انتشارات جهان فراسان

سومین ویرایش

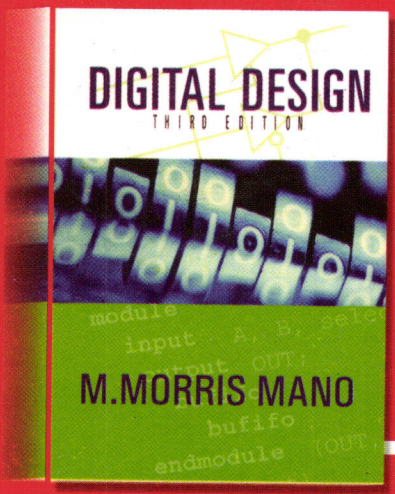
طراحی دیجیتال

(مدار منطقی)

مؤلف : موریس مانو

مترجم : دکتر قدرت سپیدنام

همراه با CD رایگان



بسم الله الرحمن الرحیم

طراحی دیجیتال (مدار منطقی)

سومین ویرایش

پروفسور موریس مانو

دانشگاه ایالت کالیفرنیا، لُس آنجلس

دکتر قدرت سپیدنام

دانشکده مهندسی دانشگاه فردوسی مشهد

انتشارات نص

مانو، موریس ۱۹۲۷ - Mano. M. Morris
طراحی دیجیتال (مدار منطقی) / مؤلف موریس مانو؛ مترجم قدرت سپیدنام،
مشهد: انتشارات خراسان ، ۱۳۴۷ .
۵۳۶ ص . مصور، جدول، نمودار .
ISBN : 964-6342-Q-1
فهرستنویسی بر اساس اطلاعات فیپا (فهرست نویسی پیش از انتشار).
Digital Design
این کتاب در سال ۱۳۶۹ توسط همین ناشر در دو جلد منتشر شده است.
کتابنامه .
چاپ اول : ۱۳۸۱
۱. کامپیوترهای رقمی. - مدارها. ۲ . مدارهای منطقی، ۳. طراحی منطقی . ۴. مدارهای مجتمع رقمی. الف. سپیدنام، قدرت، مترجم. ب . عنوان .
۶ط ۲م ۴/TKv۸۸۸ ۶۲۱/۳۵۹

انتشارات خراسان

طراحی دیجیتال (مدار منطقی)

نوشته : موریس مانو
ترجمه : دکتر قدرت سپیدنام
چاپ سوم، زمستان ۸۲ ، ۱۰۰۰۰ نسخه، انتشارات خراسان
چاپخانه سروش

قیمت با CD : ۳۰۰۰ تومان

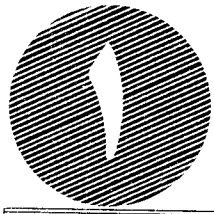
«فهرست مطالب»

سخن ناشر

"این کتاب گفتاری جدید درباره طراحی دیجیتال است و روشهای مناسبی برای طراحی مدارهای دیجیتال را به گونه‌ای روان، ساده و دقیق به خواننده می‌آموزد"

این گفتار کوتاه ، نظر مؤلف درباره کتاب است. درستی سخن مؤلف را استادان و دانشجویان ایرانی کتاب "طراحی دیجیتال" از همان سال نگارش (۱۹۸۴) تاکنون با انتخاب این کتاب و معرفی آن به خوانندگان جدید، تأیید کرده‌اند.

انتشارات خراسان ترجمه کتاب Digital design را پس از چاپ نخستین ویرایش آن با ترجمه سلیس و دقیق آقای دکتر قدرت سپیدنام استاد دانشکده مهندسی دانشگاه مشهد به خوانندگان عرضه کرده است و پس از هر بار تجدید نظر مؤلف، تجدید نظر شده آن را با ترجمه شیوای مترجم به چاپ رسانده است. اینک ترجمه سومین ویرایش آن با چاپ دو رنگ به همراه CD به اساتید و متخصصان و دانشجویان عرضه می‌شود. امید است ضمن در نظر داشتن تلاشی که مصروف آن گشته، از راهنمایی‌های ارزنده خوانندگان صاحب‌نظر بی‌نصیب نمانیم.



سیستم‌های دودویی

۱-۱ سیستم‌های دیجیتال

سیستم‌های دیجیتال در زندگی روزانه بشر نقش برجسته‌ای دارند و به این دلیل ما دوره تکنولوژی فعلی را عصر دیجیتال می‌خوانیم. سیستم‌های دیجیتال در مخابرات، تجارت، کنترل ترافیک، هدایت سفینه‌های فضایی، اعمال جراحی، هواشناسی، اینترنت و بسیاری از دیگر زمینه‌های تجاری، صنعتی و علمی به کار می‌روند. ما از تلفن‌های دیجیتال، تلویزیون‌های دیجیتال، دیسک‌های چند منظوره دیجیتال، دوربین‌های دیجیتال، و البته کامپیوترهای دیجیتال استفاده می‌کنیم. مهمترین خاصیت یک کامپیوتر دیجیتال، همگانی بودن آن است. کامپیوتر می‌تواند رشته‌ای از دستورات به نام برنامه را که روی داده‌های مفروض عمل می‌کنند، دنبال نماید. کاربر می‌تواند برنامه یا داده خود را طبق نیاز انتخاب و اجرا کند. به علت این انعطاف، کامپیوترهای همه منظوره دیجیتال می‌توانند عملیات پردازش اطلاعات را در محدوده وسیعی از کاربردها انجام دهند.

یکی از ویژگی‌های سیستم دیجیتال توانمندی آنها در دستکاری عناصر گسسته اطلاعاتی است. هر مجموعه‌ای که به تعداد متناهی از عناصر محدود باشد اطلاعات گسسته را داراست. مثال‌هایی از عناصر گسسته عبارتند از 10 رقم دهدهی، 26 حرف الفباء، 52 ورق بازی، 64 مربع بازی شطرنج. کامپیوترهای دیجیتال اولیه برای محاسبات عددی به کار می‌رفتند. در این حال، عناصر گسسته به کار رفته، ارقام بودند. نام دیجیتال یا رقمی از این مفهوم حاصل شده است. عناصر گسسته اطلاعاتی در یک سیستم دیجیتال با کمیت‌های فیزیکی به نام سیگنال نشان داده می‌شوند. رایج‌ترین سیگنال‌های الکتریکی عبارتند از ولتاژ و جریان. وسایل الکترونیکی به نام ترانزیستور در مداراتی که این سیگنال‌ها را پیاده‌سازی می‌کنند به طور چشمگیری به کار می‌روند. سیگنال‌ها در بسیاری از سیستم‌های دیجیتال الکترونیک امروزی تنها دو مقدار را دارا هستند و بنابراین گوییم دودویی‌اند. یک رقم دودویی که بیت خوانده می‌شود دو

مقدار دارد: 0 و 1 . عناصر گسسته اطلاعاتی با گروهی از بیت‌ها بـه نـام کـدهای دودویی نـمایش داده می‌شوند. مثلاً ارقام دهدهی 0 تا 9 در سیستم اعداد دیجیتال با کد چـهار بـیتی نشـان داده مـی‌شوند. با به کارگیری تکنیک‌های مختلف، گروههایی از بیت‌ها برای نمایش سمبل‌های گسسته تعریف می‌شوند و سپس در توسعه یک سیستم در قالب دیجیتال مورد استفاده قرار می‌گیرد. در نتیجه، یک سیستم دیجیتال سیستمی است که عناصر گسسته اطلاعاتی به شکل دودویی را در درون دستکاری می‌کند.

کمیت‌های اطلاعاتی یا ذاتاً گسسته‌اند و یا از نمونه‌برداری (کوانتیزه کردن) فرآیندهای پیوسته حاصل می‌شوند. به عنوان مثال یک لیست حقوق ذاتاً یک فرآیند یا رویداد گسسته بوده و حاوی نام کارمند، شماره تأمین اجتماعی، حقوق هفتگی، مالیات بردرآمد و غیره است. پرداختی به یک کارمند با استفاده از مقادیر داده گسسته مانند حروف الفبایی (نام‌ها)، ارقام (حقوق)، و نمادها یا سمبل‌های خاص (مانند $) پردازش می‌گردد. از طرف دیگر یک محقق ممکن است یک پدیده را به صورت پیوسته مشاهده کند، ولی فقط مقادیر خاصی را به صورت جدول ثبت نماید. بنابراین فرد محقق داده پیوسته را نمونه‌برداری می‌نماید ولی هر کمیت در جدول را از عناصر گسسته می‌سازد. در بسیاری از حالات نمونه‌برداری از یک فرآیند به طور خودکار به وسیله دستگاهی بنام مبدل آنالوگ به دیجیتال انجام می‌شود.

بهترین مثال از یک سیستم دیجیتال، کامپیوتر دیجیتال همه منظوره است. بخش‌های اصـلی یک کامپیوتر عبارتند از واحد حافظه، واحد پردازش مرکزی و واحدهای ورودی- خروجی. واحد حافظه برنامه‌ها و داده‌های وارده، خارج شـونده و مـیانی را ذخیره مـی‌کند. واحـد پـردازش مـرکزی اعـمال محاسباتی و دیگر عملیات روی داده‌ها را برحسب آنچه در برنامه مشخص شده، انجام می‌دهد. داده‌ها و برنامه‌هایی که به وسیله کاربر آماده شده‌اند توسط وسایل ورودی مانند صفحه کلید به حافظه انـتقال می‌یابند. یک وسیله خروجی مثل چاپگر نتایج حاصل از محاسبات را دریافت کرده و به کاربر ارائـه می‌دهد. یک کامپیوتر دیجیتال می‌تواند به چندین وسیله ورودی- خروجی وصل شود. یکی از وسایل مفید واحد مخابره است که تبادل داده را از طریق اینترنت با دیگر کاربران برقرار می‌سازد. یک کامپیوتر دیجیتال دستگاهی توانمند است که نه تنها می‌تواند محاسبات ریاضی را انجام دهد، بلکه قادر است اعمال منطقی را هم اجرا نماید. به علاوه می‌تواند جهت تصمیم‌گیری براساس شرایط داخلی یا خارجی برنامه‌ریزی شود.

برای اسـتفاده از مـدارهـای دیـجیتال در تـولیدات تـجاری دلایـل اسـاسی وجـود دارد. هـمچون کامپیوترهای دیجیتال، دستگاه‌های دیجیتال نیز قابل بـرنامه‌ریزی‌انـد. بـا تـعویض بـرنامه در وسـیله برنامه‌پذیر، سخت‌افزار یگانه‌ای قابل استفاده در کاربردهای متفاوت خواهد بود. کاهش قیمت شدید در وسایل دیجیتال به دلیل پیشرفت در تکنولوژی مدارهای مجتمع دیجیتال مرتباً روی می‌دهد. با افزایش تعداد ترانزیستورها در یک قطعه سیلیکان، توابع پیچیده‌تری قابل پیاده‌سازی شده، قیمت هـر واحـد کاهش یافته و قیمت دستگاه‌های دیجیتال روزبروز کاهش می‌یابد. دستگاه‌های ساخته شده با مدارهای مجتمع می‌توانند با سرعتی تا صد میلیون عمل در ثانیه را انجام دهند. می‌توان با اسـتفاده از کـدهای اصلاح خطا عملکرد سیستم‌های دیجیتال را به شدت اطمینان بخش نمود. مثالی از این نـوع، دیسک

چند کاره دیجیتال (DVD) است که در آن اطلاعات ویدیویی، صوتی و دیگر گونه‌ها بدون از دست رفتن یک قلم داده، ضبط می‌گردد. اطلاعات دیجیتال در DVD چنان ضبط می‌شود که هر کد در هر نمونه دیجیتال قبل از نمایش به طور خودکار خطایابی شده و اصلاح می‌گردد.

یک سیستم دیجیتال از به هم پیوستن مدول‌های دیجیتال بدست می‌آید. برای درک عمل هر مدول، دانش و آگاهی مدارهای دیجیتال و عمل منطقی آنها لازم است. هفت فصل اول این کتاب ابزار اصلی طراحی دیجیتال مانند ساختارهای منطقی گیتی، مدارهای ترکیبی و ترتیبی و وسایل منطقی برنامه‌پذیر را ارائه می‌کند. فصل ۸ طراحی دیجیتال را در سطح انتقال بین ثباتی (RTL) معرفی می‌نماید. فصل‌های ۹ و ۱۰ در مورد مدارهای ترتیبی غیرهمزمان (آسنکرون یا غیرهمگام) و دیگر خانواده‌های منطقی دیجیتال مجتمع بحث می‌کنند. فصل‌های ۱۱ و ۱۲ مدارهای مجتمع تجاری را معرفی کرده و نشان می‌دهند که چگونه در یک آزمایشگاه برای انجام آزمایشات به هم وصل می‌شوند.

یک گرایش مهم در طراحی دیجیتال، استفاده از زبان توصیف سخت‌افزاری (HDL) است. HDL نوعی زبان برنامه‌ریزی است که برای توصیف مدارهای دیجیتال به صورت متن به کار می‌رود. این زبان برای شبیه‌سازی یک سیستم دیجیتال و اطمینان از صحت عمل آن قبل از ساخت مورد استفاده قرار می‌گیرد. HDL در کنار ابزارهای طراحی منطقی دیگر برای خودکارسازی طراحی استفاده می‌شود. توصیف HDL مدارهای دیجیتال در سرتاسر این کتاب ارائه شده است.

همانطور که قبلاً گفته شد سیستم‌های دیجیتال کمیت‌های گسسته اطلاعات، که به فرم دودویی نمایش داده شده‌اند را دستکاری می‌نمایند. عملوندهای به کار رفته در محاسبات را می‌توان در سیستم اعداد دودویی بیان کرد. دیگر عناصر گسسته از جمله ارقام دهدهی به صورت کدهای دودویی نشان داده می‌شوند. پردازش داده به وسیله عناصر منطقی دودویی و با استفاده از سیگنال‌های دودویی انجام می‌گیرد. کمیات نیز در عناصر حافظه دودویی ذخیره می‌شوند. هدف این فصل معرفی مفاهیم دودویی متعدد به صورت یک مرجع برای مطالعات بعدی در فصل‌های آینده است.

۲-۱ اعداد دودویی

یک عدد دهدهی مانند 7,392 کمیتی معادل با 7 هزارتایی، بعلاوه 3 صدتایی، بعلاوه 9 ده‌تایی بعلاوه 2 واحد را نشان می‌دهد. هزارها، صدها و غیره توانی از 10 هستند که با توجه به مکان ضرایب معین می‌گردند. دقیق‌تر بگوییم، 7,392 را می‌توان به صورت زیر نوشت.

$$7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

با این وجود عرف این است که فقط ضرایب را بنویسیم و توان‌های لازم 10 را از مکان آنها استنتاج کنیم. به طور کلی یک عدد با نقطه اعشاری با یک سری ضرایب به صورت زیر نمایش داده می‌شود.

$$a_5a_4a_3a_2a_1a_0 \cdot a_{-1}a_{-2}a_{-3}$$

که ضرایب a_j هر یک از ده رقم (9, ..., 2, 1, 0) بوده و j مکان عدد را نشان می‌دهد، و از این رو توان 10

که ضریب در آن ضرب می‌گردد مشخص خواهد شد. این مطلب به صورت زیر بیان می‌شود:

$$10^5a_5 + 10^4a_4 + 10^3a_3 + 10^2a_2 + 10^1a_1 + 10^0a_0 + 10^{-1}a_{-1} + 10^{-2}a_{-2} + 10^{-3}a_{-3}$$

سیستم‌اعداد دهدهی را در مبنای 10 گوییم زیرا از 10 رقم استفاده می‌کند و ضرایب در توانی از 10 ضرب می‌گردند. سیستم دودویی یک سیستم اعداد متفاوت است. ضرایب سیستم اعداد دودویی فقط دو مقدار ممکن را دارند: 0 و 1 . هر ضریب a_j در 2^j ضرب می‌گردد. مثلاً معادل دهدهی عدد دودویی 11010.11 برابر 26.75 می‌باشد، که از ضرب ضرایب در توان‌هایی از 2 بدست می‌آید:

$$1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = 26.75$$

به طور کلی، یک عدد در مبنای r به صورت حاصلضرب توانهای r در ضرایب مربوطه‌اش بیان می‌گردد:

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \ldots + a_2 \cdot r^2 + a_1 \cdot r + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \ldots + a_{-m} \cdot r^{-m}$$

که ضرایب a_j بین 0 تا $r-1$ می‌باشند. برای تفکیک اعداد در مبناهای مختلف، ضرایب را در داخل پرانتزها نوشته و اندیس مبنارا در زیر آن می‌گذاریم (به جز در اعداد دهدهی که محتوا بیانگر دهدهی بودن است). مثالی از عدد مبنای 5 چنین است

$$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1} = (511.4)_{10}$$

مقدار ضرایب در مبنای 5 فقط 0، 1، 2، 3، 4 می‌باشند. سیستم اعداد هشت هشتی یک سیستم مبنای 8 با هشت رقم 0، 1، 2، 3، 4، 5، 6، 7 می‌باشد. مثالی از یک عدد مبنای هشت 127.4 است. برای تعیین مقدار معادل دهدهی، عدد را به صورت یک سری از توانها با مبنای 8 بسط می‌دهیم.

$$(127.4)_8 = 1 \times 8^2 + 2 \times 8^1 + 7 \times 8^0 + 4 \times 8^{-1} = (87.5)_{10}$$

توجه کنید که ارقام 8 و 9 نمی‌توانند در یک عدد هشت هشتی ظاهر شوند.

هنگامی که تعداد ارقام کمتر از 10 باشد مرسوم است که r رقم مورد نیاز برای ضرایب از سیستم دهدهی گرفته شود. هنگامی که مبنای عدد از 10 بزرگتر است از حروف الفبا برای تکمیل 10 رقم دهدهی استفاده می‌گردد. مثلاً در سیستم اعداد شانزده شانزدهی (مبنای 16)، ده رقم اول از سیستم دهدهی گرفته می‌شوند. حروف A، B، C، D، E و F به ترتیب به جای ارقام 10، 11، 12، 13، 14 و 15 به کار می‌روند. مثالی از یک عدد در مبنای 16 به صورت زیر است.

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46,687)_{10}$$

همانطورکه قبلاً اشاره شد، ارقام در یک عدد دودویی بیت خوانده می‌شوند. وقتی که یک بیت برابر 0 است در عمل جمع تبدیل مبنا نقشی ندارد. بنابراین تبدیل دودویی به دهدهی با جمع توان‌هایی از 2 که ضرایب آن 1 است صورت می‌گیرد. مثلاً،

$$(110101)_2 = 32 + 16 + 4 + 1 = (53)_{10}$$

در عدد فوق چهار عدد 1 دیده می‌شود. دهدهی مربوطه جمع چهار توان از 2 می‌باشد. 24 عدد اول حاصل از 2 به توان n در جدول ۱-۱ لیست شده‌اند. در کارهای کامپیوتری 2^{10} را K (کیلو)، 2^{20} را

جدول ۱-۱. توانهایی از 2

n	2n	n	2n	n	2n
0	1	8	256	16	65,536
1	2	9	512	17	131,072
2	4	10	1,024	18	262,144
3	8	11	2,048	19	524,288
4	16	12	4,096	20	1,048,576
5	32	13	8,192	21	2,097,152
6	64	14	16,384	22	4,194,304
7	128	15	32,768	23	8,388,608

M (مگـــا)، 2^{30} را G (گـــیگا) و 2^{40} را T (تـــرا) مـــی‌گویند. بـــنابرایـــن $4K = 2^{12} = 4096$ و $16M = 2^{24} = 16777216$ می‌باشد. معمولاً ظرفیت کامپیوتر به بایت داده می‌شود. یک بایت برابر با هشت بیت بوده و می‌تواند یک کاراکتر از صفحه کلید را در خود جای دهد. یک دیسک سخت (هارد) کامپیوتر با ظرفیت 4 گیگا دارای ظرفیت $4G = 2^{32}$ بایت می‌باشد.

اعمال حسابی با اعدادی در مبنای r از همان قواعد اعداد دهدهی استفاده می‌کنند. هنگامی که از مبنایی به جز 10 استفاده می‌شود باید دقت کرد که تنها r رقم مجاز به کار گرفته شود. در زیر مثال‌هایی برای جمع، تفریق و ضرب دودویی آورده شده است.

مضروب :	1011	مفروق منه :	101101	مضاف :	101101
مضروب فیه :	× 101	مفروق :	−100111	مضاف الیه :	+100111
	1011	باقیمانده :	000110	حاصل جمع :	1010100
	0000				
	1011				
حاصل ضرب :	10111				

جمع دو عدد دودویی مشابه قوانین دهدهی محاسبه می‌شود، به جز این که ارقام جمع در هر مکان باارزش فقط می‌تواند 0 یا 1 باشد. هر رقم نقلی حاصل در یک مکان مفروض، به وسیله جفت رقم‌های مرتبه بالاتر (باارزش‌تر) مورد استفاده قرار می‌گیرد. تفریق کمی پیچیده‌تر است. قوانین باز هـم هـمان قوانین دهدهی هستند، به جز این که قرض در یک مکان باارزش، 2 را به رقم مفروق منه مـی‌افـزایـد. (قرض در سیستم دهدهی، 10 واحد به رقم مفروق منه اضافه می‌کند.) عمل ضرب خیلی ساده است. ارقام مضروب فیه همیشه 1 یا 0 هستند. بنابراین حاصلضرب‌های جزیی برابر 0 یا برابر با مضروب می‌باشند.

۳-۱ تبدیل مبنای اعداد

همانطور که قبلاً مشاهده شد، تبدیل یک عدد در مبنای r به مبنای ده با بسط عدد به صورت یک سری از توانها و جمع همه جملات انجام می‌شود. اکنون برای تبدیل معکوس یک عدد دهدهی به یک عدد در مبنای r روالی کلی را ارائه می‌کنیم. اگر عدد حاوی نقطه ممیز باشد، لازم است تا عدد بـه دو

بخش صحیح و کسری تفکیک گردد زیرا هر بخش باید به طور جداگانه تبدیل شود. تبدیل یک عدد صحیح دهدهی به یک عدد در مبنای r با تقسیم عدد و همه خارج قسمت‌های متوالی بر r و جمع‌آوری باقیمانده‌ها انجام می‌گردد. بهتر است روال را با مثالی تشریح کنیم.

مثال ۱-۱

عدد 41 را به دودویی تبدیل کنید. ابتدا 41 را بر 2 تقسیم می‌کنیم تا خارج قسمت 20 و باقیمانده 1/2 بدست آید. خارج قسمت مجدداً بر 2 تقسیم می‌گردد تا خارج قسمت و باقیمانده جدیدی بدست آید. این روال تا رسیدن به خارج قسمت 0 ادامه می‌یابد. ضرایب عدد دودویی موردنظر به طریق زیر از باقیمانده‌ها بدست می‌آید:

	خارج قسمت صحیح		باقیمانده	ضریب عدد دودویی
$41/2 =$	20	+	$\frac{1}{2}$	$a_0 = 1$
$20/2 =$	10	+	0	$a_1 = 0$
$10/2 =$	5	+	0	$a_2 = 0$
$5/2 =$	2	+	$\frac{1}{2}$	$a_3 = 1$
$2/2 =$	1	+	0	$a_4 = 0$
$1/2 =$	0	+	$\frac{1}{2}$	$a_5 = 1$

جواب : $(41)_{10} = (a_5a_4a_3a_2a_1a_0)_2 = (101001)_2$

بنابراین، پاسخ $(101001)_2 = (a_5a_4a_3a_2a_1a_0) = (41)_{10}$ می‌باشد.

روال فوق را می‌توان به طریق ساده‌تر زیر دستکاری کرد:

خارج قسمت صحیح	باقیمانده
41	
20	1
10	0
5	0
2	1
1	0
0	1

جواب = 101001

تبدیل اعداد صحیح دهدهی به هر سیستم مبنای r مشابه مثال فوق است به جز این که تقسیم در عوض 2 بر r انجام می‌گردد.

❑

مثال ۲-۱

عدد 153 را به مبنای هشت ببرید. مبنای موردنظر r برابر 8 است. ابتدا 153 بر 8 تقسیم می‌شود تا خارج قسمت صحیح 19 و باقیمانده 1 حاصل گردد. سپس 19 بر 8 تقسیم می‌شود تا خارج قسمت 2 و باقیمانده 3 را بدست دهد. بالاخره 2 بر 8 تقسیم گردیده تا خارج قسمت 0 و باقیمانده 2 بدست آید. این روند به صورت مناسب زیر انجام می‌گردد:

153	
19	1
2	3
0	2

$= (231)_8$

در تبدیل قسمت کسری مبنای ده به دودویی از روش مشابه با بخش صحیح استفاده می‌شود. با این وجود به جای تقسیم از ضرب و به جای باقیمانده‌ها، بخش‌های صحیح انتخاب می‌گردند. مجدداً بهتر است این روش با مثالی تشریح شود.

❑

مثال ۳-۱

عدد $(0.6875)_{10}$ را به دودویی تبدیل کنید. ابتدا 0.6875 در 2 ضرب می‌شود تا یک عدد صحیح و یک کسر حاصل گردد. کسر دوباره در 2 ضرب می‌شود تا یک عدد صحیح جدید و یک کسر جدید بدست آید. این فرآیند ادامه می‌یابد تا بخش کسری صفر گردد و یا تعداد ارقام دقت مناسبی را ارائه دهند. ضرایب عدد دودویی از اعداد صحیح به صورت زیر بدست می‌آید.

	صحیح		کسری	ضریب
$0.6875 \times 2 =$	1	+	0.3750	$a_{-1} = 1$
$0.3750 \times 2 =$	0	+	0.7500	$a_{-2} = 0$
$0.7500 \times 2 =$	1	+	0.5000	$a_{-3} = 1$
$0.5000 \times 2 =$	1	+	0.0000	$a_{-4} = 1$

بنابراین پاسخ $(0.1011)_2 = (0.a_{-1}a_{-2}a_{-3}a_{-4})_2 = (0.6875)_{10}$ خواهد بود.

برای تبدیل یک عدد کسری از مبنای 10 به یک عدد در مبنای r، روش مشابهی به کار می‌رود. با این تفاوت که به جای ضرب در 2، ضرب در r انجام می‌گردد و ضرایب به جای 0، 1، از محدوده 0 تا $r-1$ خواهند بود.

❑

مثال ۴-۱

$(0.513)_{10}$ را به مبنای هشت ببرید.

$$0.513 \times 8 = 4.104$$
$$0.104 \times 8 = 0.832$$
$$0.832 \times 8 = 6.656$$
$$0.656 \times 8 = 5.248$$
$$0.248 \times 8 = 1.984$$
$$0.984 \times 8 = 7.872$$

جواب تا هفت رقم با معنی که از بخش صحیح حاصلضرب‌ها بدست می‌آید برابر است با

$$(0.513)_{10} = (0.406517\ldots)_8$$

تبدیل اعداد دهدهی که دارای هر دو بخش صحیح و کسری هستند با تبدیل جداگانه دو بخش و ترکیب جواب‌ها صورت می‌گیرد. با استفاده از مثال‌های ۱-۱ و ۳-۱ داریم:

$$(41.6875)_{10} = (101001.1011)_2$$

با استفاده از مثال‌های ۲-۱ و ۴-۱ داریم:

$$(153.513)_{10} = (231.406517)_8$$

❑

۴-۱ اعداد مبنای هشت و شانزده

تبدیل از مبنای دو به مبنای هشت و شانزده، و بالعکس نقش عمده‌ای در کامپیوترهای دیجیتال بازی می‌کند. چون $8 = 2^3$ و $16 = 2^4$ است، هر رقم در مبنای هشت متعلق به سه رقم دودویی و هر رقم مبنای شانزده متعلق به چهار رقم دودویی است. در جدول ۲-۱ شانزده عدد اول سیستم اعداد دهدهی، دودویی، هشت هشتی و شانزده شانزدهی لیست شده‌اند.

تبدیل از دودویی به هشت هشتی به سادگی با تفکیک عدد دودویی به گروه‌های سه رقمی در دو طرف نقطه دودویی بدست می‌آید. سپس به هر گروه یک رقم مبنای هشت تعلق می‌گیرد. مثال زیر روال مربوطه را نشان می‌دهد:

$$(10\ 110\ 001\ 101\ 011 \cdot 111\ 100\ 000\ 110)_2 = (26153.7406)_8$$
$$\quad 2\ \ 6\ \ 1\ \ 5\ \ 3\ \ \ 7\ \ 4\ \ 0\ \ 6$$

تبدیل از مبنای دو به مبنای شانزده نیز مشابه با روند فوق است، با این تفاوت که عدد دودویی به گروه‌های چهار رقمی تفکیک می‌شوند:

$$(10\ 1100\ 0110\ 1011 \cdot 1111\ 0010)_2 = (2C6B.F2)_{16}$$
$$\quad 2\ \ C\ \ 6\ \ B\ \ F\ \ 2$$

پس از مطالعه مقادیر لیست شده در جدول ۲-۱، به سادگی می‌توان رقم مبنای شانزده را برای هر گروه از ارقام دودویی به خاطر سپرد.

جدول ۲-۱ . اعداد با مبناهای متفاوت

دهدهی (مبنای 10)	دودویی (مبنای 2)	هشتایی (مبنای 8)	شانزده تایی (مبنای 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

تبدیل از مبنای هشت یا شانزده به دودویی با روشی عکس روش بالا انجام می‌گردد. هر رقم مبنای هشت با سه رقم مبنای دو معادل خود جایگزین می‌شود. به طور مشابه، هر رقم مبنای شانزده با چهار رقم دودویی معادلش جایگزین خواهد شد. این مطلب در مثال‌های زیر تشریح شده است:

$$(673.124)_8 = (\underset{6}{110}\ \underset{7}{111}\ \underset{3}{011}\ .\ \underset{1}{001}\ \underset{2}{010}\ \underset{4}{100})_2$$

و

$$(306.D)_{16} = (\underset{3}{0011}\ \underset{0}{0000}\ \underset{6}{0110}\ .\ \underset{D}{1101})_2$$

اساساً کار با اعداد دودویی، به دلیل این‌که تعداد ارقامشان سه یا چهار برابر معادلشان در مبنای ده می‌باشد، مشکل است. مثلاً عدد دودویی 1111 1111 1111 معادل 4095 است. با این وجود کامپیوترهای دیجیتال اعداد دودویی را به کار می‌برند و گاهی نیز لازم است تا کاربر مستقیماً به وسیله اعداد دودویی با ماشین ارتباط برقرار کند. یک راه برای حفظ سیستم دودویی در کامپیوتر، که در ضمن تعداد ارقام را برای انسان کاهش می‌دهد، استفاده از رابطه بین سیستم اعداد دودویی و هشت هشتی یا شانزده شانزدهی است. با این روش، انسان برحسب اعداد مبنای هشت یا شانزده فکر کرده و در مواقعی که ارتباط مستقیم با ماشین لازم است، تبدیل لازمه را با بررسی این اعداد انجام خواهد داد. به این ترتیب عدد دودویی 1111 1111 1111 که دارای 12 رقم است در مبنای هشت به صورت چهار رقم 7777 و یا در مبنای شانزده به شکل FFF در می‌آید. به هنگام تبادل اطلاعات با انسان، نمایش مبنای هشت یا شانزده اعداد دودویی مطلوب‌تر است زیراکه در این مبناها اعداد با 1/3 یا 1/4 تعداد ارقامشان در دودویی قابل نمایش‌اند. بنابراین اغلب کتابچه‌های راهنمای کامپیوتر از اعداد مبنای هشت یا شانزده برای نمایش کمیت‌های دودویی استفاده

می‌کنند. گرچه نمایش مبنای شانزده مناسب‌تر به نظر می‌رسد ولی انتخاب یکی از این دو کاملاً اختیاری است.

۱-۵ متمم‌ها

متمم‌ها در کامپیوترهای دیجیتال برای ساده کردن عمل تفریق و یا عملیات منطقی به کار می‌روند. در هر مبنایی چون r، دو نوع متمم وجود دارد: یکی متمم مبنا و دیگری متمم مبنای کاهش یافته. فرم اول به متمم r و دومی به متمم $r-1$ موسوم است. وقتی که مقدار مبنا یا پایه را جایگزین کنیم، برای اعداد دودویی، متمم‌های 2 و 1 و برای اعداد دهدهی، متمم‌های 10 و 9 را خواهیم داشت.

متمم در مبنای کاهش یافته

با فرض داشتن عددی n رقمی مانند N در مبنای r، متمم $(r-1)$ عدد به صورت $(r^n-1) - N$ تعریف می‌شود. برای اعداد دهدهی، $r = 10$ و $r-1 = 9$ است، و به این ترتیب متمم 9 عدد N برابر $(10^n-1) - N$ خواهد بود. در اینجا 10^n نمایشگر عددی است که متشکل از یک 1 و به دنبال آن n عدد 0 می‌باشد. 10^n-1 عددی است که با n عدد 9 نشان داده می‌شود. مثلاً اگر $n = 4$ باشد، داریم $10^4 = 10000$ و $10^4-1 = 9999$. به این ترتیب نتیجه می‌شودکه متمم 9 یک عدد دهدهی با تفریق هر رقم از 9 حاصل خواهد شد. به چند مثال عددی زیر توجه کنید.

متمم 9 عدد 546700 برابر است با 999999 - 546700 = 453299
متمم 9 عدد 012398 برابر است با 999999 - 012398 = 987601

برای اعداد دودویی، $r = 2$ و $r-1 = 1$ است، بدین ترتیب متمم 1 عدد N، $(2^n-1) - N$ خواهد بود. مجدداً، 2^n برابر با یک عدد دودویی است که از یک 1، و n عدد 0 تشکیل شده است. 2^n-1 یک عدد دودویی متشکل از n عدد 1 می‌باشد. مثلاً اگر $n = 4$ باشد، داریم $(10000)_2 = 2^4$ و $(1111)_2 = 2^4-1$. بنابراین متمم 1 یک عدد دودویی از تفریق هر رقم از 1 بدست می‌آید. با این وجود، هنگام تفریق ارقام دودویی از 1، یکی از دو حالت $1 - 0 = 1$ و یا $0 = 1 - 1$ را خواهیم داشت، که سبب می‌شود هر بیت از 0 به 1 و از 1 به 0 تبدیل شود. بنابراین متمم یک عدد دودویی با تغییر 1ها به 0 و 0ها به 1 حاصل می‌گردد. در زیر مثال‌هایی آورده شده است:

متمم 1 عدد 1011000 برابر است با 0100111
متمم 1 عدد 0101101 برابر است با 1010010

متمم $(r-1)$ اعداد مبنای هشت و شانزده بترتیب از تفریق ارقام آنها از 7 یا F (15 دهدهی) حاصل می‌شود.

متمم مبنا

متمم r یک عدد n رقمی مانند N در مبنای r به صورت r^n-N به ازاء $N \neq 0$ و برابر با 0 در ازاء $N=0$ تعریف می‌شود. از مقایسه این متمم با متمم $(r-1)$ نتیجه می‌شود که متمم r از جمع 1 با متمم $(r-1)$

حاصل می‌شود زیرا $r^n-N = [(r^n-1) - N] + 1$ می‌باشد. به این ترتیب متمم 10 یک عدد دهدهی مانند 2389 برابر است با $7610 + 1 = 7611$ که از جمع 1 به مقدار متمم 9 حاصل می‌گردد. متمم 2 عدد دودویی 101100 برابر است با $010011 + 1 = 010100$ و از جمع 1 با مقدار متمم 1 بدست می‌آید.

چون 10^n عددی است که با یک 1 و n عدد 0 به دنبال آن نمایش داده می‌شود، 10^n-N که متمم 10 عدد N است نیز با تغییر ندادن 0های کم‌ارزش‌تر و تفریق اولین رقم غیر صفر کم‌ارزش‌تر از 10 و تفریق همه رقم‌های باارزش‌تر از 9 حاصل می‌گردد.

متمم 10 عدد 012398 برابر 987602 می‌باشد

متمم 10 عدد 246700 برابر 753300 است

متمم 10 اولین عدد با تفریق 8 از 10 در کم‌ارزش‌ترین مکان و تفریق دیگر ارقام از 9 حاصل شده است. متمم 10 دومین عدد بدین ترتیب حاصل گشته است که دو 0 کم‌ارزش‌تر رها می‌شوند، 7 از 10 و دیگر ارقام از 9 تفریق می‌گردند.

به طور مشابه متمم 2 می‌تواند با رها کردن همه 0های کم‌ارزش‌تر و نیز تغییر نکردن اولین 1 و جایگزینی همه 0ها با 1ها و 1ها با 0ها در دیگر ارقام باارزش‌تر حاصل شود.

متمم 2 عدد 1101100 برابر 0010100 است

متمم 2 عدد 0110111 برابر 1001001 است

متمم 2 اولین عدد با رها کردن دو 0 کم‌ارزش‌تر و اولین 1 و سپس جایگزینی همه 1ها با 0 و 0ها با 1 در چهار رقم با ارزش‌تر باقیمانده بدست می‌آید. متمم 2 دومین عدد با رها کردن اولین 1 و متمم کردن دیگر ارقام حاصل می‌گردد.

در تعاریف قبلی، فرض شد که اعداد دارای نقطه ممیز نیستند. اگر عدد اولیه N حاوی ممیز باشد باید آن را موقتاً حذف نمود تا متمم r و $(r-1)$ بدست آید. آنگاه آن را به مکان مربوطه‌اش باز می‌گردانیم. توجه داشته باشید که متمم یک متمم، عدد را به حالت اولیه‌اش باز می‌گرداند. متمم r عدد N برابر r^n-N است. متمم یک متمم برابر است با $r^n - (r^n-N) = N$، که همان عدد اولیه است.

تفریق به کمک متمم‌ها

روش مستقیم تفریق که در مدارس ابتدایی بیان شد از مفهوم قرض کردن استفاده می‌نماید. در این روش وقتی که یک رقم در مفروق‌منه کوچکتر از مفروق باشد یک 1 از رقم باارزش‌تر قرض گرفته می‌شود. این روش هنگامی که تفریق با قلم و کاغذ انجام شود به خوبی کار می‌کند. با این وجود، هنگام پیاده‌سازی تفریق با سخت‌افزار دیجیتال، روش کمتر از روش‌های متمم کارآیی دارد.

تفریق دو عدد n رقمی بی‌علامت $M - N$ در مبنای r به صورت زیر انجام می‌شود.

۱- مفروق منه M را به متمم r مفروض N، اضافه کنید. یعنی

$$M + (r^n-N) = M - N + r^n$$

۲- اگر $M \geq N$ باشد، عمل جمع یک رقم نقلی انتهایی تولید می‌کند که باید چشم‌پوشی شود؛ آنچه

باقی می‌ماند نتیجه M − N است.

۳- اگر M < N باشد، عمل جمع هیچگونه رقم نقلی انتهایی تولید نمی‌کند و جواب برابر با $r^n - (M - N)$ است که همان متمم r عمل (M − N) می‌باشد. برای یافتن جواب معمول، متمم r حاصل جمع را بدست می‌آوریم و سپس یک علامت منفی در جلو آن می‌گذاریم.

مثال‌های زیر روال را تشریح می‌کنند.

مثال ۵-۱

با استفاده از متمم 10 تفریق 3250 − 72532 را انجام دهید.

M	=	72532
متمم 10 عدد N	= +	96750
حاصل جمع	=	169282
چشم‌پوشی از رقم نقلی انتهایی 10^5	= −	100000
جواب	=	69282

توجه کنید که M دارای 5 رقم ولی N فقط 4 رقمی است. چون هر دو عدد باید دارای تعداد ارقام برابری باشند، پس N به صورت 03250 نوشته می‌شود. متمم 10 این عدد N ، یک 9 در باارزش‌ترین مکان تولید می‌نماید. تولید رقم نقلی در باارزش‌ترین مکان دلالت بر M ≥ N دارد و نتیجه نیز مثبت است.

❑

مثال ۶-۱

با استفاده از متمم 10 ، تفریق 72532 − 3250 را بدست آورید.

M	=	03250
متمم 10 عدد N	= +	27468
حاصل جمع	=	30718

که رقم نقلی در آن وجود ندارد.

بنابراین جواب (متمم 10 عدد 30718)− = 69282− است.

❑

توجه کنید که چون 72532 > 3250 است نتیجه منفی است. نظر به این که ما با اعداد بی‌علامت سروکار داریم، نمی‌توان برای این حالت نتیجه بدون علامتی بدست آورد. وقتی تفریق را با متمم‌ها انجام می‌دهیم، جواب منفی از نبود رقم نقلی انتهایی و نتیجه متمم تشخیص داده می‌شود. هنگام کار با کاغذ و قلم، می‌توانیم جواب را به یک عدد منفی علامت‌دار تغییر دهیم تا به فرم معمول‌تر در آید. تفریق با متمم‌ها برای اعداد دودویی به روش مشابهی در مثال‌های زیر آمده است.

مثال ۷-۱

با فرض دو عدد دودویی X = 1010100 و Y = 1000011، تفریق‌های زیر را انجام دهید.

(الف) X − Y و (ب) Y − X با استفاده از متمم 2

(الف)

X	=	1010100
متمم 2 عدد Y	=	+ 0111101
حاصل جمع	=	10010001
چشم‌پوشی از نقلی انتهایی 2^7	=	−10000000
X − Y	=	0010001 جواب :

(ب)

Y	=	1000011
متمم 2 عدد X	=	+ 0101100
حاصل جمع	=	1101111

رقم نقلی انتهایی وجود ندارد.

بنابراین، جواب Y − X = (متمم 2 عدد 1101111) − = 0010001− است. ❑

تفریق اعداد بی‌علامت را می‌توان با متمم (r−1) نیز انجام داد. به خاطر دارید که متمم (r−1) ، یک واحد کمتر از متمم r است. به این علت، نتیجه جمع مفروق منه با متمم مفروق حاصل جمعی تولید می‌کند که یکی کمتر از تفاضل صحیح به هنگام رخداد نقلی انتهایی است. حذف رقم نقلی انتهایی و افزودن 1 به حاصل جمع را رقم نقلی چرخشی می‌خوانند.

مثال ۸-۱

مثال ۷-۱ را با استفاده از متمم 1 انجام دهید.

(الف) X − Y = 1010100 − 1000011

X	=	1010100
Y	=	+ 0111100
حاصل جمع	=	10010000
رقم نقلی انتهایی	=	+ 1
X − Y	=	0010001 جواب :

(ب) Y − X = 1000011 − 1010100

Y	=	1000011
X	=	+ 0101011
حاصل جمع	=	1101110

که رقم نقلی انتهایی وجود ندارد.

بنابراین جواب Y − X = (متمم 1 عدد 1101110)− = 0010001− است. ❑

توجه کنید که نتیجه منفی با اخذ متمم 1 از حاصل جمع بدست آمده است زیرا این نوع متمم به کار رفت. روال رقم نقلی انتهایی چرخشی در تفریق اعداد دهدهی بی‌علامت با متمم 9 نیز قابل استفاده است.

۶-۱ اعداد دودویی علامت‌دار

اعداد صحیح مثبت و از آن جمله صفر را می‌توان با اعداد بی‌علامت نشان داد. با این وجود برای نمایش اعداد صحیح منفی به علامتی نیاز داریم. در حساب معمولی، یک عدد منفی را با یک علامت منها و عدد مثبت را با علامت بعلاوه نشان می‌دهند. به دلیل محدودیت در سخت‌افزار، کامپیوترها باید هر چیزی را با ارقام دودویی نشان دهند. مرسوم است که علامت را با یک بیت واقع در سمت چپ‌ترین مکان عدد نمایش دهند. معمولاً اعداد مثبت را با گذاشتن 0 و اعداد منفی را با گذاشتن 1 در محل بیت مزبور معرفی می‌نمایند.

لازم است بدانیم که هر دو گروه اعداد دودویی علامت‌دار و بی‌علامت هنگام ارائه به کامپیوتر از رشته بیت‌ها تشکیل شده‌اند. معمولاً کاربر علامت‌دار بودن یا نبودن عدد را معین می‌نماید. اگر عدد دودویی علامت‌دار باشد سمت چپ‌ترین بیت، علامت، و بقیه بیت‌ها عدد هستند. اگر عدد دودویی بدون علامت فرض شود، سمت چپ‌ترین بیت، باارزش‌ترین بیت عدد خواهد بود. مثلاً رشته بیت‌های 01001 می‌تواند به عنوان 9 (دودویی بی‌علامت) و یا 9+ (دودویی علامت‌دار) در نظر گرفته شود زیرا سمت چپ‌ترین بیت 0 است. رشته بیت‌های 11001 به عنوان 25 بی‌علامت و یا 9- علامت‌دار خواهد بود زیرا در سمت چپ‌ترین مکان عدد، رقم 1 وجود دارد که بیانگر منفی بودن عدد، و بقیه چهار بیت عدد 9 را نشان می‌دهد. معمولاً اگر نوع عدد از قبل مشخص باشد هیچگونه اشتباهی در تشخیص وجود نخواهد داشت.

نمایش اعداد علامت‌دار در آخرین مثال فوق، نمایش **مقدار- علامت‌دار** نامیده می‌شود. در این نام‌گذاری، عدد شامل مقدار و یک سمبل (+ یا -) یا یک بیت (0 یا 1) برای مشخص نمودن علامت است. این روش در محاسبات معمولی مورد استفاده می‌باشد. وقتی که عملیات حسابی در یک کامپیوتر پیاده‌سازی می‌شوند، بهتر است روش دیگری به نام سیستم متمم- علامت‌دار برای ارائه اعداد منفی به کار گرفته شود. در این سیستم، یک عدد منفی با متمم‌اش مشخص می‌شود. در حالی که سیستم مقدار- علامت، عدد را با تغییر علامتش منفی می‌کند، **سیستم متمم- علامت‌دار**، منفی عدد را با متمم‌سازی‌اش تهیه می‌نماید. چون اعداد مثبت همواره با 0 در سمت چپ‌شان شروع می‌شوند متمم همواره با 1 آغاز می‌گردد، که بیانگر عدد منفی خواهد بود. سیستم متمم- علامت‌دار می‌تواند از متمم 1 یا متمم 2 استفاده کند ولی متمم 2 رایج‌تر است.

به عنوان مثال فرض کنید که عدد 9 با هشت بیت در دودویی نشان داده شده باشد. 9+ با یک بیت 0 در سمت چپ‌ترین مکان و به دنبال آن معادل دودویی 9 می‌آید که نتیجه 00001001 خواهد بود. توجه داشته باشید که تمام هشت بیت باید مقدار داشته باشند، بنابراین پس از بیت علامت بقیه مکان‌ها تا اولین 1 از سمت چپ با 0 پر می‌شوند. هر چند که برای نمایش 9+ فقط یک راه وجود دارد، برای

نمایش 9- سه روش موجود است.

نمایش مقدار- علامت‌دار 10001001
نمایش متمم 1 _ علامت‌دار 11110110
نمایش متمم 2 _ علامت‌دار 11110111

در سیستم مقدار- علامت‌دار، با تغییر بیت علامت در سمت چپ‌ترین مکان، از 0 به 1 ، عدد 9- به 9+ تبدیل می‌شود. در متمم 1 _ علامت‌دار، 9- را با متمم کردن همه بیت‌های 9+ ، از جمله بیت علامت بدست می‌آوریم. در متمم 2 _ علامت‌دار، 9- با متمم 2کردن تمام بیت‌های عدد مثبت از جمله بیت علامت حاصل می‌شود.

جدول ۳-۱ همه اعداد دودویی 4 بیت را به هر سه فرم نمایش، نشان می‌دهد. عدد دهدهی معادل نیز به منظور وجود مرجع آورده شده است. توجه کنید که اعداد مثبت در هر سه نمایش یکسانند و دارای یک 0 در سمت چپ‌ترین مکان می‌باشند. سیستم متمم 2_ علامت‌دار تنها یک نمایش برای 0 دارد که همیشه مثبت است. دو سیستم دیگر دارای 0 مثبت و 0 منفی‌اند، چیزی که در محاسبات معمولی با آن مواجه نمی‌شویم. مجدداً توجه کنید که همه اعداد منفی دارای 1 در سمت چپ‌ترین بیت‌اند. به این ترتیب ما آنها را از اعداد مثبت تفکیک می‌نماییم. با چهار بیت قادریم 16 عدد دودویی را نشان دهیم. در سیستم مقدار- علامت‌دار و متمم 1 ، هشت عدد مثبت و هشت عدد منفی و از جمله دو عدد صفر وجود دارد. در نمایش متمم 2 ، هشت عدد مثبت از جمله صفر و هشت عدد منفی موجود است.

سیستم مقدار- علامت‌دار در حساب معمولی مورد استفاده است و هنگامی که در کامپیوتر به کار

جدول ۳-۱ . اعداد دودویی علامت‌دار

دهدهی	متمم 2 - علامت دار	متمم 1 - علامت دار	مقدار - علامت دار
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000
−0	—	1111	1000
−1	1111	1110	1001
−2	1110	1101	1010
−3	1101	1100	1011
−4	1100	1011	1100
−5	1011	1010	1101
−6	1010	1001	1110
−7	1001	1000	1111
−8	1000	—	—

رود، مشکلاتی به همراه دارد زیرا باید علامت و مقدار به طور جداگانه دستکاری شوند. بنابراین، معمولاً متمم- علامت به کار گرفته می‌شود. متمم 1 نیز مشکلاتی را به بار می‌آورد و به ندرت در محاسبات به کار می‌رود. متمم 1 برای اعمال منطقی مفید است چون تبدیل 0 به 1 و یا 1 به 0 معادل با عمل متمم منطقی است. این مطلب در فصل بعدی نشان داده خواهد شد. بحث زیر در مورد محاسبات دودویی علامت‌دار، اختصاصاً به نمایش اعداد منفی متمم 2_ علامت‌دار تعلق دارد. روال‌های مشابهی قابل اعمال به سیستم متمم 1_ علامت‌دار بوده و در آن رقم نقلی چرخشی همچون اعداد بدون علامت به حساب خواهد آمد.

جمع حسابی

جمع دو عدد در سیستم مقدار- علامت از قوانین معمول در حساب تبعیت می‌نماید. اگر علامت‌ها یکسان باشند دو مقدار را با هم جمع کرده و به حاصل جمع علامت مشترک را تخصیص می‌دهیم. اگر علامت‌ها یکی نباشند، مقدار کوچکتر را از بزرگتر کم می‌کنیم و به نتیجه حاصل علامت عدد بزرگتر را اختصاص می‌دهیم. مثلاً $-12 = -(37 - 25) = -(37) + (+25)$ است، که با تفریق مقدار کوچکتر 25 از مقدار بزرگتر 37 و استفاده از علامت 37 برای علامت نتیجه انجام شده است. این فرآیند به مقایسه علامت‌ها و اندازه‌ها و سپس اجرای جمع و تفریق نیاز دارد. روال مشابهی در نمایش مقدار- علامت برای اعداد دودویی قابل اعمال است. برعکس قوانین جمع اعداد در سیستم مقدار- علامت نیازی به مقایسه و تفریق ندارد، بلکه فقط باید آنها را جمع کرد. روال بسیار ساده بوده و برای اعداد دودویی به صورت زیر بیان می‌گردد:

جمع دو عدد دودویی علامت‌دار با اعداد منفی که به فرم متمم 2 نمایش داده شده‌اند از جمع دو عدد از جمله بیت‌های علامت حاصل می‌شود. رقم نقلی حاصل از بیت علامت چشم‌پوشی می‌گردد. مثال‌های عددی برای جمع در زیر آمده است.

+ 6	00000110	− 6	11111010
+13	00001101	+13	00001101
+19	00010011	+ 7	00000111
+ 6	00000110	−6	11111010
−13	11110011	−13	11110011
− 7	11111001	−19	11101101

توجه کنید که اعداد منفی باید از ابتدا به صورت متمم 2 باشند و حاصل جمع اگر منفی باشد به صورت متمم 2 خواهد بود.

در هر یک از چهار حالت فوق، عمل انجام شده جمعی است که در آن بیت علامت هم لحاظ شده است. در این روش هر رقم نقلی خروجی نادیده گرفته می‌شود و نتایج منفی به فرم متمم 2 هستند.

برای یافتن یک جواب صحیح، باید مطمئن باشیم که برای جای دادن نتیجه، تعداد کافی بیت وجود

دارد. اگر با دو عدد n بیت شروع کنیم و حاصل جمع n+1 بیت را اشغال کند گوییم سرریز رخ داده است. هنگامی که جمع با کاغذ و قلم انجام می‌شود سرریز مسئله‌ای نیست زیرا ما از نظر عرض صفحه محدودیت نداریم. در این گونه موارد فقط یک 0 به بالاترین مکان عدد مثبت و یا یک 1 به بالاترین مکان عدد منفی می‌افزاییم تا آنها را به n+1 بیت گسترش دهیم و سپس جمع را اجرا نماییم. ولی سرریز در کامپیوتر مشکل‌ساز است زیرا تعداد بیت‌هایی که عدد را نگه می‌دارند محدود می‌باشد، و نتیجه‌ای که از مقدار نهایی به میزان 1 واحد تجاوز کند را نمی‌توان در آن جای داد.

فرم متمم اعداد منفی برای کسانی که به سیستم مقدار- علامت عادت کرده‌اند، ناآشناست. برای تعیین یک عدد منفی وقتی که به فرم متمم 2 علامت‌دار باشد، لازم است که آن را به یک عدد مثبت تبدیل کنیم تا به شکل آشناتری در آید. مثلاً عدد دودویی علامت‌دار 11111001 یک عدد منفی است زیرا سمت چپ‌ترین بیت برابر 1 است. متمم 2 آن 00000111 می‌باشد که معادل دودویی عدد 7+ است. به این ترتیب تشخیص می‌دهیم که عدد منفی اولیه برابر 7− بوده است.

تفریق حسابی

تفریق دو عدد دودویی علامت‌دار، وقتی که اعداد منفی به صورت متمم 2 باشند بسیار ساده است و می‌تواند به صورت زیر بیان شود:

متمم 2 مفروق (از جمله بیت علامت) را بدست آورید و آن را به مفروق منه (از جمله بیت علامت) اضافه کنید. رقم نقلی خروجی از مکان بیت علامت چشم‌پوشی شود.

این پدیده به این علت رخ می‌دهد که اگر علامت مفروق عوض شود، تفریق به جمع تبدیل خواهد شد. این نکته با روابط زیر نشان داده شده است:

$$(\pm A) - (+B) = (\pm A) + (-B);$$
$$(\pm A) - (-B) = (\pm A) + (+B).$$

اما تغییر یک عدد مثبت به یک عدد منفی به سادگی با بدست آوردن متمم 2 آن امکان‌پذیر است. عکس مطلب فوق نیز صحیح می‌باشد زیرا متمم یک عدد منفی متمم، یک عدد مثبت معادل تولید می‌نماید. تفریق 7+ = (13−) − (6−) را در نظر بگیرید. در دودویی با هشت بیت، این تفریق به صورت (11111010 − 11110011) است. با یافتن متمم 2 مفروق (13−)، یعنی (13+)، تفریق به فرم جمع در می‌آید. در دودویی این عمل به صورت 100000111 = 00001101 + 11111010 می‌باشد. با حذف رقم نقلی انتهایی پاسخ صحیح 00000111 (7+) خواهد بود.

لازم به تذکر است که جمع و تفریق اعداد دودویی در سیستم متمم- علامت مشابه با قوانین جمع و تفریق معمولی است. بنابراین کامپیوترها دارای سخت‌افزار مشترک برای هر دو نوع عمل حسابی می‌باشند. کاربر یا برنامه‌نویس باید نتایج چنین جمع یا تفریقی را به طور متفاوت تفسیر کند و این تفسیر به فرض اولیه وی یعنی علامت‌دار بودن یا بی‌علامت بودن اعداد بستگی دارد.

۱-۷ کدهای دودویی

سیستم‌های دیجیتال از سیگنال‌هایی که دو مقدار مجزا و عناصری از مدار که دو حالت با ثبات دارند استفاده می‌کنند. بین سیگنال‌های دودویی، عناصر مدار دودویی و ارقام دودویی رابطه مستقیمی وجود دارد. مثلاً یک عدد دودویی n رقمی را می‌توان با n عنصر مدار دودویی که هر یک دارای یک سیگنال خروجی معادل 0 یا 1 اند، نشان داد. سیستم‌های دیجیتال نه تنها اعداد دودویی بلکه بسیاری از اجزاء گسسته اطلاعات را هم دستکاری و نمایش می‌دهند و روی آنها عمل می‌کنند. هر عنصر گسسته اطلاعاتی را در میان یک گروه از مقادیر می‌توان با استفاده از کد دودویی نشان داد. کدها باید به صورت دودویی باشند زیرا کامپیوترها فقط قادرند 1ها و 0ها را نگه دارند. با ید توجه داشت که کدها فقط نماد یا سمبل نمایش اطلاعات را عوض می‌کنند و نه مفهوم آنها را. اگر بیت‌های یک کامپیوتر را به طور تصادفی مورد بررسی قرار دهیم، ملاحظه خواهیم کرد که اغلب به جای اعداد دودویی، اطلاعات کد شده در آنها وجود دارد.

یک کد دودویی n بیت، گروهی متشکل از n بیت است که 2^n ترکیب ممکن از 1ها و 0ها را داراست، و هر ترکیب یک عنصر از مجموعه کد شده را نمایش می‌دهد. یک مجموعه چهار عنصری با دو بیت کد می‌شود، که به هر عنصر یکی از ترکیبات بیتی زیر تخصیص می‌یابد: 00، 01، 10، 11. یک مجموعه هشت عضوی به کد 3 بیت نیاز دارد و برای یک مجموعه 16 عنصری یک کد 4 بیت لازم است. ترکیب بیتی یک کد n بیتی با شمارش دودویی از 0 تا 2^n-1 حاصل می‌گردد. به هر عنصر باید یک ترکیب بیتی دودویی منحصر به فرد اختصاص یابد و هیچ دو عنصر دارای مقدار یکسانی نمی‌باشند؛ در غیر این صورت تخصیص کد گنگ و بی‌معنی خواهد بود.

گرچه حداقل تعداد بیت‌های لازم برای 2^n کد مجزا، برابر n است، حداکثر تعداد بیت‌ها برای تعریف یک کد دودویی وجود ندارد. مثلاً 10 رقم دهدهی با 10 بیت قابل کد شدن است، و هر رقم دهدهی به یکی از ترکیبات نُه 0 و یک 1 تخصیص می‌یابد. در این کد دهدهی، رقم 6 به ترکیب بیتی 0001000000 اختصاص می‌یابد.

کد BCD

گرچه سیستم اعداد دودویی طبیعی‌ترین سیستم برای یک کامپیوتر است، ولی بسیاری از مردم به سیستم دهدهی عادت دارند. یکی از راه‌های حل این مشکل تبدیل اعداد دهدهی به دودویی، اجرای همه محاسبات به دودویی و سپس تبدیل نتایج دودویی به دهدهی است. این روش لازم می‌دارد تا اعداد دهدهی را در کامپیوتر ذخیره کنیم تا بتوانند به دودویی تبدیل شوند. چون کامپیوتر فقط می‌تواند مقادیر دودویی را قبول کند، باید ارقام دهدهی را با کدی مرکب از 1ها و 0ها نشان دهیم. هنگامی که این ارقام به فرم کد شده در کامپیوتر ذخیره شوند، می‌توان مستقیماً عملیات حسابی را روی این اعداد دهدهی اجرا نمود.

اگر تعداد عناصر در مجموعه به صورت توانی از 2 نباشد، کد دودویی دارای ترکیبات بیتی تخصیص نیافته خواهد بود. 10 رقم دهدهی چنین مجموعه‌ای را می‌سازند. یک کد دودویی که بتواند 10 عنصر را

جدول ۴-۱ . دهدهی کد شده به دودویی (BCD)

سمبل دهدهی	رقم BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

از هم تفکیک کند باید حداقل 4 بیت داشته باشد، ولی 6 ترکیب از 16 ترکیب ممکن تخصیص نیافته باقی می‌ماند. با مقداردهی 10گانه به 4 بیت می‌توان کدهای دودویی متفاوتی بدست آورد. کدی که برای ارقام دهدهی معمولاً به‌کار می‌رود در جدول ۴-۱ دیده می‌شود. این کد را دهدهی کد شده به دودویی می‌خوانند و به طور خلاصه آن را با BCD نمایش می‌دهند. چند کد دهدهی دیگر بعداً در این بخش نمایش داده خواهند شد.

جدول ۴-۱ هر کد 4 بیتی را به یک رقم دهدهی نسبت می‌دهد. یک عدد k رقمی در BCD به 4k بیت نیاز دارد. عدد دهدهی 396 در BCD با 12 بیت به صورت 0110 1001 0011 نمایش داده می‌شود، که در آن هر گروه 4 بیتی یک رقم دهدهی را نشان می‌دهد. هرگاه عدد دهدهی در BCD بین 0 تا 9 باشد با عدد دودویی‌اش معادل است. یک عدد BCD بزرگتر از 10 با عدد دودویی معادلش، هر چند که از 0ها و 1ها تشکیل شده، متفاوت است. به علاوه ترکیبات دودویی 1010 تا 1111 به‌کار نرفته‌اند و در BCD مفهومی ندارند. عدد دهدهی 185 و مقدار مربوطه آن را در BCD و دودویی ملاحظه کنید:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

مقدار BCD دارای 12 بیت است، ولی عدد دودویی معادل آنها تنها 8 بیت لازم دارد. واضح است یک عدد BCD نسبت به مقدار دودویی به بیت‌های بیشتری احتیاج دارد. با این وجود، استفاده از اعداد دهدهی دارای مزیت است زیرا داده‌های ورودی و خروجی به وسیله انسان‌هایی که سیستم‌های دهدهی را به‌کار می‌برند تولید می‌شود.

توجه داشته باشید که اعداد BCD اعداد دهدهی هستند و نه اعداد دودودویی، هر چند که آنها در ساختارشان از بیت‌ها استفاده می‌کنند. تنها تفاوت بین یک عدد دهدهی و BCD این است که اعداد دهدهی سمبل‌های 0، 1، 2، ...، 9 و اعداد BCD سمبل‌های 0000، 0001، 0010، ...، 1001 را به‌کار می‌برند. مقدار دهدهی دقیقاً یکی است. عدد 10 دهدهی در BCD با هشت بیت 0000 0001 و عدد 15 با 0101 0001 نمایش داده می‌شوند. مقادیر دودویی معادل آنها به ترتیب 1010 و 1111 است که تنها چهار بیت دارند.

جمع BCD

جمع دو رقم دهدهی در BCD، همراه با رقم نقلی احتمالی از جفت رقم کم‌ارزش‌تر قبلی را در نظر بگیرید. چون هر رقم از 9 تجاوز نمی‌کند، جمع نمی‌تواند بزرگتر از 19 = 1 + 9 + 9 باشد که در آن 1، رقم نقلی قبلی است. فرض کنید می‌خواهیم ارقام BCD را به شکل اعداد دودویی با هم جمع کنیم. جمع دودویی، نتیجه‌ای بین 0 تا 19 را تولید خواهد کرد. این مقادیر به دودویی برابرند با 0000 تا 10011، ولی به فرم BCD برابر با 0000 تا 1001 1 می‌باشند. اولین رقم، رقم نقلی و چهار بیت بعدی رقم جمع BCD است. وقتی حاصل جمع دودویی برابر یا کم‌تر از 1001 است (بدون نقلی)، رقم BCD مربوطه صحیح است. با این وجود، وقتی جمع دودویی بزرگتر یا مساوی 1010 باشد، نتیجه یک رقم BCD نامعتبر است. جمع $6 = (0110)_2$ با حاصل جمع دودویی، آن را به رقم صحیح بدل کرده و در صورت لزوم رقم نقلی نیز تولید خواهد کرد. دلیل این است که اختلاف بین یک رقم نقلی در باارزش‌ترین مکان بیتی حاصل از جمع دودویی و نقلی دهدهی برابر است با $16 - 10 = 6$. جمع BCD زیر را در نظر بگیرید:

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

در هر حالت دو رقم BCD و نیز فرم دودویی آنها با هم جمع می‌شوند. اگر جمع دودویی بزرگتر یا برابر 1010 باشد، به آن 0110 را می‌افزاییم تا رقم BCD حاصل جمع و نقلی صحیح حاصل شود. در مثال اول حاصل جمع برابر 9 می‌باشد که یک رقم حاصل جمع BCD صحیح است. در مثال دوم، جمع دودویی یک رقم BCD نامعتبر تولید می‌کند. افزایش 0110 به آن رقم حاصل جمع BCD صحیح 0010 (2) را همراه با یک رقم نقلی به وجود می‌آورد. در مثال سوم، جمع دودویی یک رقم نقلی خواهد داشت. این وضعیت هنگامی رخ می‌دهد که حاصل جمع مساوی یا بزرگتر از 16 باشد. گرچه چهار بیت دیگر کمتر از 1001 است، حاصل جمع نیاز به اصلاح دارد زیرا دارای رقم نقلی است. با جمع 0110 رقم حاصل جمع BCD 0111 (7) و یک نقلی BCD حاصل می‌گردد.

جمع دو عدد BCD بی‌علامت n رقمی روال مشابهی دارد. جمع 760 = 576 + 184 را به BCD در نظر بگیرید:

BCD carry	1	1		
	0001	1000	0100	184
	+0101	0111	0110	+576
Binary sum	0111	10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

اولین جفت رقم BCD کم‌ارزش‌تر، یک رقم BCD برابر با 0000 و یک رقم نقلی برای جفت رقم بعدی را

تولید می‌کند. جفت رقم دوم بعلاوه نقلی قبلی حاصل جمع 0110 و یک رقم نقلی برای جفت رقم بعدی را به وجود می‌آورد. جفت رقم سوم بعلاوه یک رقم نقلی حاصل جمع دودویی 0111 را تولید کرده و نیاز به اصلاح ندارد.

حساب دهدهی

نمایش اعداد دهدهی علامت‌دار در BCD مشابه اعداد علامت‌دار در دودویی است. ما می‌توانیم از هر یک از هر دو سیستم مقدار- علامت‌دار یا متمم- علامت‌دار استفاده کنیم. علامت یک عدد دهدهی معمولاً با چهار بیت نمایش داده می‌شود تا با کد 4 بیت ارقام دهدهی همسان باشد. معمولاً علامت مثبت با چهار 0 و علامت منها با 9 BCD یعنی 1001 نشان داده می‌شود.

سیستم مقدار- علامت‌دار به ندرت در کامپیوترها به کار می‌رود. این سیستم می‌تواند متمم 9 یا متمم 10 باشد، ولی اغلب متمم 10 به کار گرفته می‌شود. برای بدست آوردن متمم 10 یک عدد BCD، ابتدا متمم 9 را بدست اورده و به کم‌ارزش‌ترین رقم 1 واحد می‌افزاییم. متمم 9 نیز با کسر هر رقم از 9 حاصل می‌گردد.

روالی که در بخش قبل برای سیستم متمم 2 علامت‌دار بنا نهاده شد به سیستم متمم 10 علامت‌دار در اعداد دهدهی نیز قابل اعمال است. جمع با افزودن همه ارقام، از جمله رقم علامت و چشم‌پوشی از رقم نقلی انتهایی انجام می‌شود. در اینجا فرض می‌شود که همه اعداد منفی به فرم متمم 10 باشند. جمع +135 = (-240) + (+375) را در نظر بگیرید که در سیستم متمم علامت‌دار انجام شده است.

$$\begin{array}{r} 0\ 375 \\ +9\ 760 \\ \hline 0\ 135 \end{array}$$

عدد 9 واقع در سمت چپ‌ترین مکان عدد دوم نمایشگر یک علامت منفی و 9760 متمم 10 عدد 0240 است. برای بدست آوردن 135+ ، دو عدد با هم جمع و رقم نقلی نادیده گرفته می‌شود. البته اعداد دهدهی داخل کامپیوتر، از جمله ارقام علامت باید به BCD باشند. همانطور که قبلاً اشاره شد جمع با ارقام BCD انجام می‌شود.

تفریق اعداد دهدهی اعم از علامت‌دار یا بی‌علامت در سیستم متمم 10 مشابه با حالت دودویی است. متمم 10 مفروق را بدست آورید و آن را به مفروق منه اضافه کنید. بسیاری از کامپیوترها برای انجام محاسبات حسابی اعداد دهدهی در BCD، سخت‌افزار خاصی دارند. کاربر کامپیوتر می‌تواند برای انجام عمل حسابی با اعداد دهدهی، بدون نیاز به تبدیل آنها، به دودویی برنامه‌نویسی کند.

دیگر کدهای دهدهی

کدهای دودویی برای ارقام دهدهی به حداقل چهار بیت در قبال هر رقم نیاز دارند. با ایجاد 10 ترکیب مختلف در چهار بیت کدهای مختلف را می‌توان ایجاد کرد. کدهای BCD و سه نوع کد دیگر در جدول۵-۱ نشان داده شده‌اند. هر کد تنها 10 ترکیب بیتی از 16 ترکیب ممکن را در چهار بیت به کار

جدول ۵-۱. چهار کد دودویی متفاوت برای ارقام دهدهی

رقم دهدهی	BCD 8421	2421	افزونی 3	8 4-2-1
0	0000	0000	0011	0 0 0 0
1	0001	0001	0100	0 1 1 1
2	0010	0010	0101	0 1 1 0
3	0011	0011	0110	0 1 0 1
4	0100	0100	0111	0 1 0 0
5	0101	1011	1000	1 0 1 1
6	0110	1100	1001	1 0 1 0
7	0111	1101	1010	1 0 0 1
8	1000	1110	1011	1 0 0 0
9	1001	1111	1100	1 1 1 1
ترکیبات بیتی بکارنرفته	1010	0101	0000	0 0 0 1
	1011	0110	0001	0 0 1 0
	1100	0111	0010	0 0 1 1
	1101	1000	1101	1 1 0 0
	1110	1001	1110	1 1 0 1
	1111	1010	1111	1 1 1 0

می‌برد. شش ترکیبی که در هر حال به کار نروند دارای مفهوم نیستند و باید از آنها اجتناب کرد.

کدهای BCD و 2421 از جمله کدهای وزین هستند. در یک کد وزین به هر مکان از بیت وزنی تخصیص داده شده است به نحوی که هر رقم با جمع اوزان تمام 1ها در ترکیب کد بدست می‌آید. کد BCD دارای وزن‌های 8، 4، 2، 1 است که مربوط به توازنی از دو برای هر بیت است. مثلاً تخصیص بیتی 0110 با توجه به وزن 1ها برای 6 تفسیر می‌شود زیرا $8 \times 0 + 4 \times 1 + 2 \times 1 + 1 \times 0 = 6$. ترکیب بیتی 1101 وقتی با کد 2421 وزین شود معادل دهدهی $2 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 = 7$ را خواهد داد. توجه کنید که در کد 2421 بعضی از ارقام به دو طریق کدگذاری می‌شوند. عدد 4 دهدهی به ترکیب‌های بیتی 0100 یا 1010 متعلق است زیرا هر دو ترکیب عدد 4 را نشان می‌دهند.

کدهای 2421 و افزونی 3- مثال‌هایی از کدهای خود متمم هستند. این کدها خواصی دارند که با توجه به آن متمم 9 عدد دهدهی مستقیماً از تغییر 0ها به 1 و 1ها به 0 در کد حاصل می‌شود. مثلاً عدد دهدهی 395 در افزونی 3- به صورت 1000 1100 0110 می‌باشد. متمم 9 آن یعنی 604 به صورت 0111 0011 1001 می‌باشد که در واقع با متمم هر بیت از کد بدست می‌آید (مثل متمم 1 عدد دودویی).

کد افزونی 3- به دلیل خود متممی‌اش در کامپیوترهای قدیمی به کار می‌رفت. این کد بی‌وزن است و هر ترکیب کدی در آن از جمع مقدار دودویی متناظرش با 3 حاصل می‌شود. توجه داشته باشید که کد BCD خود متمم نیست.

کد 8، 4، 2- و 1- مثالی از تخصیص هر دو نوع وزن مثبت و منفی به یک کد دهدهی است. در این حال، ترکیب بیتی 0110 برای 2 دهدهی تخصیص یافته و از رابطه $8\times0+4\times1+(-2)\times1+(-1)\times0=2$ محاسبه می‌شود.

کدگری

خروجی داده بسیاری از سیستم‌های فیزیکی کمیت‌های پیوسته را فراهم می‌کنند. این داده‌ها باید قبل از اعمال به یک سیستم دیجیتال، به فرم دیجیتال تبدیل شوند. اطلاعات پیوسته یا آنالوگ با وسیله‌ای به نام مبدل آنالوگ به دیجیتال به فرم دیجیتال در می‌آیند. گاهی بهتر است برای نمایش داده دیجیتال از کد گری در جدول ۶-۱ استفاده شود. مزیت کد گری نسبت به کد دودویی این است که وقتی از یک کد به کد دیگری می‌رویم تنها یک بیت تغییر می‌یابد. مثلاً در رفتن از 7 به 8، کد گری از 0100 به 1100 تغییر می‌کند. دیده می‌شود که تنها بیت اول سمت چپ از 0 به 1 عوض شده است و سه بیت دیگر ثابت باقی می‌مانند. هنگامی که این کد را با کد دودویی مقایسه نماییم، تغییر از 7 به 8، به صورت 0111 به 1000 خواهد بود و ملاحظه می‌گردد که هر چهار بیت تغییر مقدار داده‌اند.

کد گری در کاربردهایی مورد استفاده است که رشته معمول اعداد دودویی ممکن است در طول انتقال یا تبدیل از یک عدد به دیگری خطایی تولید کنند. اگر اعداد دودویی استفاده شوند، تغییر از 0111 به 1000 ممکن است در صورت تغییر طولانی‌تر سمت راست‌ترین بیت، عدد میانی 1001 را تولید کند که خطاست. کد گری این نوع خطا را حذف می‌کند زیرا تنها یک بیت تغییر در حین انتقال بین دو عدد وجود دارد.

نوعی کاربرد کد گری به هنگام نمایش داده آنالوگ ناشی از تغییرات پیوسته مکان یک شفت موتور می‌باشد. شفت به قطعات تقسیم می‌گردد و به هر قطعه یک عدد تخصیص می‌یابد. اگر قطعات مجاور منطبق با رشته کد گری باشد، ابهام به هنگام تشخیص خطای ناشی از جدایی دو قطعه حذف می‌گردد.

جدول ۶-۱. کد گری 4 بیتی

معادل دهدهی	کد گری
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

کدکارا کتراسکی

در بسیاری از کاربردهای کامپیوترهای دیجیتال نه تنها نیاز به دستکاری روی داده‌های عددی بلکه روی حروف نیز وجود دارد. برای مثال یک کمپانی بیمه با میلیونها سند، از یک کامپیوتر دیجیتال برای پردازش فایل‌هایش استفاده می‌کند. برای نمایش نام و سایر مشخصات طرف‌های قرارداد، داشتن یک کد دودویی برای حروف الفبا ضروری است. به علاوه همان کد دودویی می‌باید اعداد دهدهی و بعضی کاراکترهای خاص دیگر مانند $ را نیز نمایش دهد. یک مجموعه کاراکتر الفبا عددی مجموعه‌ای از عناصر، متشکل از 10 رقم عدد، 26 حروف الفبا و تعداد معینی از علائم خاص است. چنین مجموعه‌ای بین 36 تا 64 عنصر برای حروف بزرگ و یا بین 64 تا 128 عنصر با حروف بزرگ و کوچک برای هر کلید دارد. در حالت اول به شش بیت و در حالت دوم به هفت بیت نیاز است.

کد دودویی استاندارد برای کاراکترهای الفبا عددی، اَسکی (ASCII) است. این کد از هفت بیت برای کد کردن 128 کاراکتر، طبق جدول (۷-۱) استفاده می‌کند. هفت بیت کد با b_1 تا b_7 مشخص شده‌اند که b_7 باارزش‌ترین بیت را تشکیل می‌دهد. مثلاً حروف A در اسکی به صورت 1000001 (ستون 100 و سطر 0001) می‌باشد. کد اسکی حاوی 94 کاراکتر گرافیکی و 34 کاراکتر غیرچاپی برای عملیات کنترلی مختلف است. کاراکترهای گرافیکی نیز از 26 حرف بزرگ (A تا Z)، 26 حرف کوچک (a تا z)، 10 عدد (0 تا 9) و 32 کاراکتر قابل چاپ مانند %، * و $ تشکیل شده است.

34 کاراکتر کنترل در جدول اسکی با اسامی خلاصه شده‌ای مشخص شده‌اند. این کاراکترها در پایین شکل همراه با نوع عملشان ذکر شده‌اند. این کاراکترها برای رده‌دهی داده‌ها و مرتب نمودن مطالب چاپ شدنی در قالب موردنظر به کار می‌روند. سه نوع کاراکترهای کنترلی وجود دارند: افکتورهای فورمت، جداسازی اطلاعات و کاراکترهای کنترل تبادل اطلاعات. افکتور فورمت قالب آنچه راکه باید چاپ شود کنترل می‌نماید. این گروه شامل پَسبَر (BS)، جدول‌بندی افقی (HT) و بازگشت نورد (CR) است. جداسازی‌های اطلاعات، داده‌ها را به صورت پاراگراف‌ها و صفحات دسته‌بندی می‌کند. از جمله آنها می‌توان از جداساز رکورد (RS) و جداساز فایل (FS) نام برد. کاراکترهای کنترل تبادل اطلاعات در حین انتقال متن بین پایانه‌های دور از هم مفیدند. مثال‌هایی از این نوع عبارتند از کاراکتر شروع متن (STX) و ختم متن (ETX) که برای قاب‌بندی یک پیام متنی به هنگام انتقال از خط تلفن به کار می‌رود.

اسکی یک کد 7 بیتی است ولی اغلب کامپیوترها واحدهای هشت بیتی اطلاعات که بایت نام دارند را دستکاری می‌کنند. بنابراین کاراکترهای اسکی اغلب هر کدام در یک بایت ذخیره می‌شوند. بیت اضافی گاهی برای اهداف دیگری به کار می‌رود و اغلب به کاربرد بستگی دارد. مثلاً بعضی از چاپگرها کاراکترهای اسکی را به صورت 8 بیتی می‌شناسند که در آن باارزش‌ترین بیت برابر 0 است. 128 کاراکتر 8 بیتی دیگر را با قرار دادن 1 در باارزش‌ترین مکان برای فونت‌های نوع ایتالیک یا الفبای یونانی می‌توان به کار برد.

کدتشخیص خطا

برای تشخیص خطاها در مخابره یا پردازش داده، گاهی بیت هشتمی به نام بیت توازن به کاراکتر

جدول ۷-۱ . کد استاندارد امریکایی برای تبادل اطلاعات (ASCII)

$b_4b_3b_2b_1$	$b_7b_6b_5$ = 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

کاراکترهای کنترل

NUL	Null	DLE	Data-link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End-of-transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

اسکی اضافه می‌شود. بیت توازن، بیتی اضافی است که حاوی پیامی بوده و طی آن تعداد 1های کل، زوج یا فرد خواهد شد. دو کاراکتر زیر به همراه توازن زوج یا فرد دیده می‌شوند.

	با توازن زوج	با توازن فرد
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

در هر حالت یک بیت به سمت چپ‌ترین مکان کد می‌افزاییم تا تعداد 1ها در کاراکتر برای توازن زوج، زوج و یا این که تعداد 1ها در کاراکتر برای توازن فرد، فرد گردد. به طور کلی یکی از دو توازن اختیار می‌شود ولی توازن زوج معمول‌تر می‌باشد.

بیت توازن در تشخیص خطا در حین انتقال اطلاعات از یک مکان به مکان دیگر مفید است. این کار با تولید یک بیت توازن زوج برای هر کاراکتر در سمت فرستنده انجام می‌گردد. کاراکترهای 8 بیتی که به همراه بیت‌های توازن می‌باشند به مقصد ارسال می‌گردند. سپس توازن هر کاراکتر در سمت گیرنده چک می‌شود. اگر توازن کاراکتر دریافتی زوج نباشد، حداقل یک بیت در حین انتقال تغییر کرده است. این روش یک، سه یا هر تعداد فردی از خطا را در هر کاراکتر انتقال یافته تشخیص می‌دهد. در این حالت تعداد زوجی از خطاها قابل تشخیص نخواهد بود. کدهای خطای دیگر برای محافظت از خطاهای زوج لازم است.

این‌که پس از شناسایی خطا چه کاری باید انجام داد به کاربرد مربوطه بستگی دارد. یک امکان این است که با فرض اتفاقی بودن خطا و عدم تکرار، تقاضای ارسال مجدد گردد. در این حالت اگر گیرنده یک خطای توازن را شناسایی کند، یک کاراکتر کنترل اسکی، ASCII NAK ، (تصدیق نفی) را متشکل از هشت بیت با توازن زوج 0101 1001 بازپس می‌فرستد. اگر خطایی شناسایی نشد، گیرنده کاراکتر کنترل ACK (تصدیق) را با کد 00000110 باز می‌فرستد. سمت فرستنده دوباره با ارسال پیام NAK پاسخ می‌دهد تا این که توازن صحیح دریافت شود. اگر پس از چند نوبت تکرار، انتقال همچنان دارای خطا بود، پیام خطایی به اپراتور برای چک کردن عامل خطا در خط انتقال فرستاده می‌شود.

۸-۱ ذخیره‌سازی دودویی و ثبات‌ها

اطلاعات دودویی در یک کامپیوتر دیجیتال، نوعی موجودیت فیزیکی در یک محیط ذخیره‌سازی اطلاعات برای ذخیره تک‌تک بیت‌ها باید داشته باشد. یک سلول دودویی وسیله‌ای است که از خود دو حالت باثبات را به نمایش می‌گذارد و قابل استقرار در یکی از دو حالت است. ورودی سلول سیگنال‌های تحریک کننده‌ای را برای استقرار در یکی از دو حالت دریافت می‌کند. خروجی سلول کمیتی فیزیکی است که دو حالت را از هم تفکیک می‌نماید. وقتی خروجی در سلول در یکی از دو حالت باشد اطلاعات ذخیره شده 1 و در حالت با ثبات دیگر 0 است.

ثبات‌ها

ثبات گروهی از سلول‌های دودویی است. یک ثبات با n سلول، هر کمیت گسسته اطلاعاتی را که حاوی n بیت باشد، می‌تواند ذخیره کند. حالت یک ثبات عددی n تایی از 1ها و 0ها است که هر بیت حالت یک سلول را در ثبات بیان می‌کند. محتوای یک ثبات تابعی از تفسیر اطلاعات ذخیره شده در آن است. مثلاً یک ثبات 16 بیتی را با محتوای زیر در نظر بگیرید:

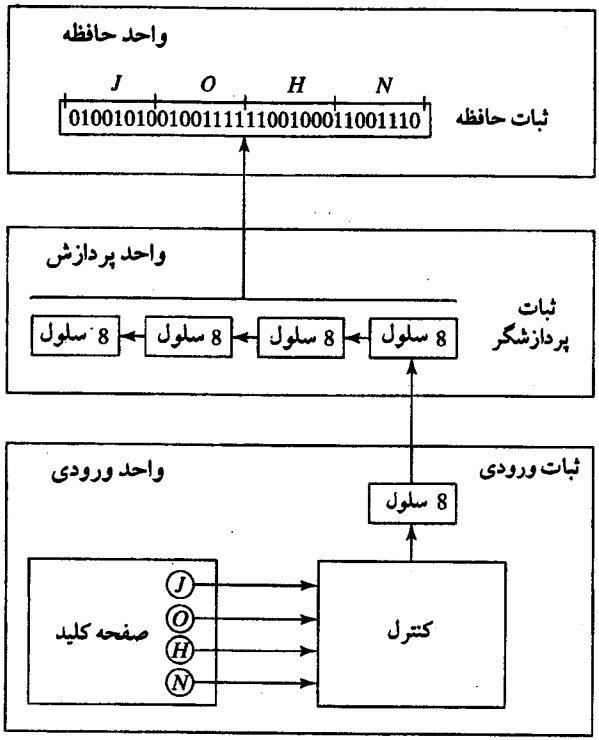
1100001111001001

یک ثبات با 16 سلول می‌تواند در یکی از 2^{16} حالت ممکن باشد. اگر فرض کنیم که محتوای یک ثبات

عدد صحیح دودویی را نشان می‌دهد، آنگاه ثبات می‌تواند هر عدد دودویی از 0 تا $2^{16}-1$ را ذخیره کند. برای مثال خاص فوق، محتوای ثبات عدد دودویی معادل با 50121 دهدهی است. اگر فرض کنیم که ثبات کاراکترهای کد هشت بیتی الفبا عددی را ذخیره کرده است، محتوای ثبات می‌تواند هر دو کاراکتر با معنی باشد. برای کد اسکی با توازن زوج واقع در هشتمین بیت باارزش، ثبات حاوی دو کاراکتر C (هشت بیت سمت چپ) و I (هشت بیت سمت راست) می‌باشد. از طرف دیگر اگر محتوای ثبات بصورت چهار رقم دهدهی تفسیر شود، محتوای ثبات یک عدد دهدهی چهار رقمی خواهد بود. درکد افزونی -3 مثال بالا عدد دهدهی 9096 است. درکد BCD این محتوا بی‌معنی است زیرا ترکیب بیتی 1100 به هیچ رقم دهدهی تخصیص نیافته است. با توجه به مثال، واضح است که یک ثبات قادر است اجزاء گسسته‌ای از اطلاعات را در خود ذخیره نماید و نیز آرایش بیتی یکسانی ممکن است برای انواع دیگر داده، تفسیر متفاوتی داشته باشد.

انتقال بین ثباتی

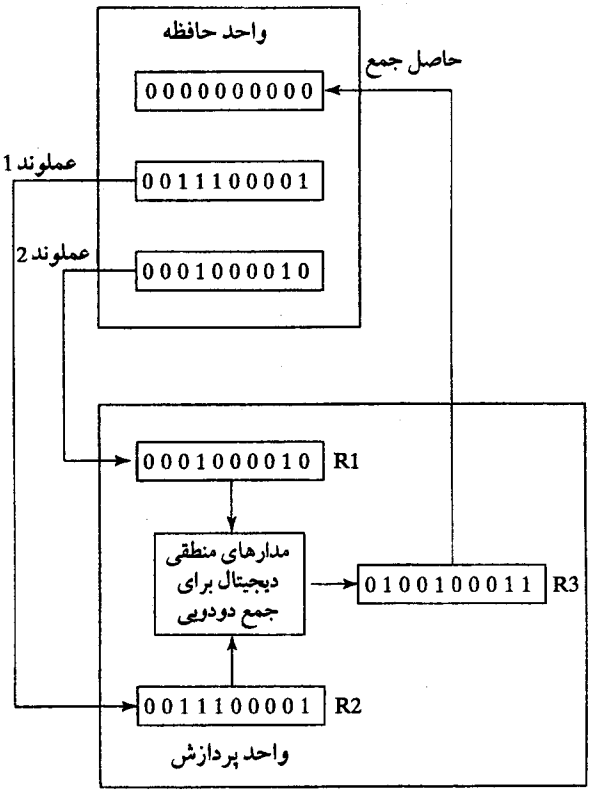
یک سیستم دیجیتال با ثبات‌هایش و قطعاتی که پردازش داده را اجرا می‌کنند مشخص می‌شود. عمل انتقال بین ثباتی یک عمل بنیادی در سیستم‌های دیجیتال است. این عمل متشکل از انتقال اطلاعات دودویی از یک مجموعه ثبات به مجموعه دیگر است. انتقال ممکن است مستقیماً از یک ثبات به دیگری باشد و یا از طریق مدارهای پردازش داده برای انجام یک عمل صورت گیرد. شکل ۱-۱ انتقال



شکل ۱-۱ . انتقال اطلاعات به وسیله ثباتها

اطلاعات را در میان ثبات‌ها نشان می‌دهد و نیز انتقال اطلاعات دودویی از یک صفحه کلید به یک حافظه به تصویر کشیده شده است. فرض می‌شود واحد ورودی دارای یک صفحه کلید، مدار کنترل و یک ثبات ورودی است. هر بار کلیدی فشرده شود، کنترل یک کد کاراکتر الفبا عددی هشت بیتی معادل را وارد ثبات ورودی می‌نماید. فرض می‌کنیم کد وارده از نوع اسکی و دارای توازن فرد باشد. اطلاعات از ثبات ورودی وارد هشت سلول کم‌ارزش‌تر یک ثبات پردازنده می‌گردد. پس از هر انتقال، ثبات ورودی پاک می‌شود تا کنترل بتواند پس از زدن کلید، کد هشت بیت جدید را وارد کند. قبل از ورود یا انتقال هر هشت بیت کاراکتر به ثبات پردازنده، کاراکتر قبلی به هشت سلول بعدی در سمت چپ خود منتقل می‌شود. وقتی کار انتقال چهار کاراکتر کامل شد، ثبات پردازنده پر شده و محتوای آن به یک ثبات حافظه منتقل می‌گردد. محتوای ذخیره شده در ثبات حافظه که در شکل ۱-۱ ملاحظه می‌گردد از انتقال کاراکترهای "J"، "O"، "H" و "N" پس از زدن چهار کلید مناسب حاصل شده است.

برای پردازش کمیت‌های گسسته‌ای از اطلاعات به فرم دودویی، کامپیوتر باید مجهز به وسایلی باشد تا داده مورد پردازش را حفظ و نیز بیت‌های اطلاعات را دستکاری کند. وسیله‌ای که برای نگهداری داده به کار می‌رود ثبات است. دستکاری متغیرهای دودویی به کمک مدارهای منطقی دیجیتال انجام می‌شود. شکل ۲-۱ فرآیند جمع دو عدد دودویی 10 بیتی را نشان می‌دهد. واحد حافظه که معمولاً



شکل ۲-۱ . مثالی برای پردازش اطلاعات دودویی

متشکل از میلیون‌ها ثبات است، در نمودار تنها با سه ثباتش نشان داده شده است. بخشی از واحد پردازشگر که در شکل آمده شامل سه ثبات R_1، R_2 و R_3 به همراه مدارهای منطقی دیجیتال است که بیت‌های ثبات R_1 و ثبات R_2 را دستکاری کرده و نتیجه را که یک حاصل جمع حسابی است به R_3 منتقل می‌سازد. ثبات‌های تشکیل دهنده حافظه اطلاعات را ذخیره می‌کنند و قادر نیستند دو عملوند را پردازش نمایند. با این وجود، اطلاعات ذخیره شده در حافظه قابل انتقال به ثبات‌های پردازشگر است. نتایج حاصل در ثبات‌های پردازنده می‌تواند مجدداً به ثبات‌های حافظه برای ذخیره کاربر بعدی باز فرستاده شود. نمودار، محتوای ارسال شده دو عملوند در ثبات‌های حافظه را به R_1 و R_2 نشان می‌دهد. مدارهای منطقی حاصل جمع را تولید می‌کنند، که بعد به ثبات R_3 منتقل می‌گردد. اکنون محتوای R_3 قابل بازگشت به یکی از ثبات‌های حافظه است.

دو مثال آخر، چگونگی توانمندی جریان اطلاعات را به صورت خیلی ساده تشریح کردند. ثبات‌های سیستم، اجزاء مبنا و اصلی ذخیره و نگهداری اطلاعات دودویی‌اند. مدارهای منطقی دیجیتال اطلاعات دودویی ذخیره شده در ثبات‌ها را پردازش می‌نمایند. مدارهای منطقی دیجیتال و ثبات‌ها در فصل‌های ۲ الی ۶ پوشش یافته‌اند. واحد حافظه در فصل ۷ توصیف شده است. انتقال در سطح ثبات که مورد استفاده در توصیف و طراحی سیستم‌هاست نیز در فصل ۸ بیان شده است.

۹-۱ منطق دودویی

منطق دودویی با متغیرهایی که دو ارزش گسسته و عملیاتی که مفهوم منطقی دارند، سروکار دارد. دو ارزشی که متغیرها اختیار می‌کنند ممکن است با اسامی مختلفی نامگذاری شوند (مثل صحیح و غلط، بله و خیر و غیره)، اما برای ما بهتر است آن را برحسب بیت تصور کنیم و مقادیر 1 و 0 را به آن تخصیص دهیم. منطق دودویی معرفی شده در این بخش معادل با جبری به نام جبر بول است. جبر بول در فصل ۲ به طور مشروح‌تر بیان شده است. در این بخش جبر بول به روشی غیرمستدل بوده و ارتباط آن با مدارهای منطقی دیجیتال و سیگنال‌های دودویی بیان شده است.

تعریف منطق دودویی

منطق دودویی شامل متغیرهای دودویی و عملیات منطقی است. متغیرها با حروف الفبایی مانند A، B، C، x، y، z و غیره نامگذاری می‌شوند، که هر متغیر فقط و فقط دو مقدار مجزای 0 و 1 دارد. سه نوع عملیات منطقی اصلی وجود دارند: AND، OR و NOT.

۱- AND: این عمل به وسیله یک "." یا بدون ذکر هر عملگری نمایش داده می‌شود. مثلاً $x.y = z$ یا $xy = z$ را چنین می‌خوانیم "x AND y برابر است با z". عمل منطقی AND چنین تفسیر می‌شود که، $z = 1$ است اگر و فقط اگر $x = 1$ و $y = 1$ باشد؛ در غیر این صورت $z = 0$ است. (به یاد داشته باشید که x و y و z متغیرهای دودویی هستند و نمی‌توانند به جز 1 و 0 چیز دیگری باشند.)

۲- OR: عملی است که با علامت بعلاوه نشان داده می‌شود. مثلاً $x + y = z$ را چنین می‌خوانیم

"x OR y برابر است با z" و به این معنی است که $z = 1$ است به شرطی که $x = 1$ یا $y = 1$ و یا هر دو $x = 1$ و $y = 1$ باشند. اگر هر دو $x = 0$ و $y = 0$ باشد آنگاه $z = 0$ خواهد بود.

۳- NOT : این عمل با یک علامت پریم نشان داده می‌شود (و گاهی با یک خط بار). مثلاً $x' = z$ (یا $\bar{x} = z$) و چنین خوانده می‌شود، "not x برابر است با z" و به این معنی است که z چیزی است که x نیست. به بیان دیگر اگر $x = 1$ باشد آنگاه $z = 0$؛ اما اگر $x = 0$ باشد، آنگاه $z = 1$ است. عمل NOT را متمم هم می‌گویند چون 1 را به 0 و 0 را به 1 تبدیل می‌کند.

منطق دودویی شبیه حساب دودویی است، و اعمال AND و OR به ترتیب به اعمال ضرب و جمع شباهت دارند. در حقیقت سمبل‌های به کار رفته برای AND و OR همان‌هایی هستند که برای ضرب و جمع مورد استفاده قرار می‌گیرند. معهذا منطق دودویی را نباید با حساب دودویی اشتباه کرد. مسئله‌ای که باید مورد توجه قرار گیرد این است که یک متغیر حسابی، عددی را مشخص می‌کند که ممکن است دارای چندین رقم باشد. یک متغیر منطقی همیشه 0 و یا 1 است. مثلاً در حساب دودویی داریم $1 + 1 = 10$ (می‌خوانیم: "یک بعلاوه یک برابر است با 2")، در صورتی که در منطق دودویی، داریم $1 + 1 = 1$ (می‌خوانیم: "یک OR یک، برابر است با 1").

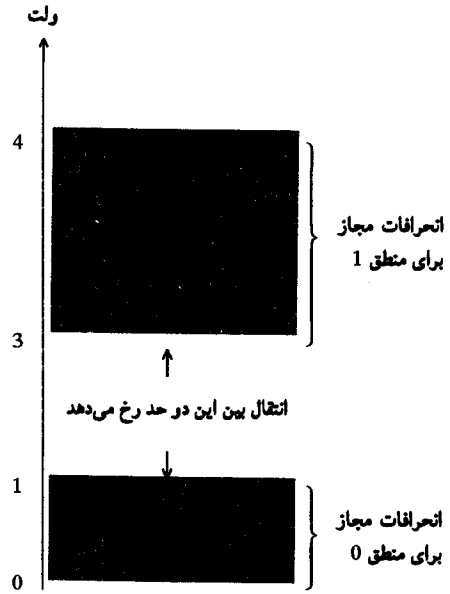
برای هر ترکیبی از مقادیر x و y، مقدار معینی برای z وجود دارد که این مقدار پس از اِعمال یا تعریف عمل منطقی مشخص می‌گردد. این تعاریف را می‌توان به صورت خلاصه یا استفاده از جدول درستی فهرست کرد. یک جدول درستی، جدولی است متشکل از تمام ترکیبات ممکن متغیرها و بیانگر ارتباط بین مقادیر آنها و نتایج حاصل از عمل مربوطه روی آنها می‌باشد. به عنوان مثال جداول درستی برای عملگرهای AND و OR با متغیرهای x و y، با لیست کردن همه مقادیر ممکن آنها وقتی به صورت زوج ترکیب شده‌اند، حاصل می‌شود. نتیجه عمل برای هر ترکیب به طور جداگانه آمده است. جداول درستی AND و OR و NOT در جدول ۸-۱ نشان داده شده‌اند. این جداول تعریف عملیات مذکور را به طور شفاف بیان می‌دارند.

گیت‌های منطقی

گیت‌های منطقی، مدارهایی الکترونیک هستند که روی یک یا چند سیگنال ورودی عمل می‌کنند تا یک سیگنال خروجی تولید نمایند. سیگنال‌های الکترونیکی مانند ولتاژها یا جریان‌هایی که در سرتاسر یک سیستم دیجیتال وجود دارند دو مقدار جدا از هم را اختیار می‌کنند. مدارهایی که با ولتاژ کار می‌کنند

جدول ۸-۱ . جدول درستی عملیات منطقی

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

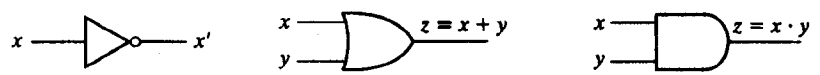


شکل ۳-۱ . مثالی از سیگنال‌های دودویی

به دو سطح ولتاژ که نمایشگر یک متغیر دودویی و برابر با منطق 1 و منطق 0اند واکنش نشان می‌دهند. مثلاً یک سیستم دیجیتال خاص ممکن است منطق 0را به عنوان سیگنالی برابر با 0ولت و منطق 1را به صورت سیگنالی برابر با 4 ولت تعریف کند. در عمل، هر سطح ولتاژ، محدوده مورد قبولی مانند شکل۳-۱ را داراست.

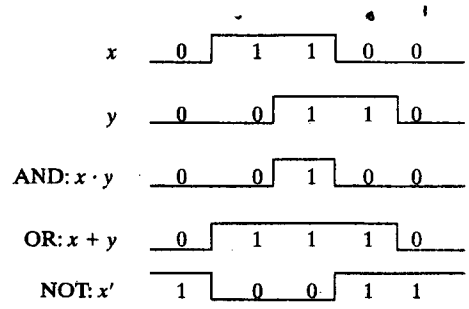
پایانه‌های ورودی مدارهای دیجیتال سیگنال‌های دودویی را در محدوده مُجازی می‌پذیرند و در پایانه‌های خروجی در محدوده مجازی پاسخ می‌دهند. ناحیه میانی بین دو ناحیه مجاز تنها هنگام گذر از یک حالت به حالت دیگر قطع می‌شود. هر اطلاعات محاسباتی یا کنترلی موردنظر را می‌توان با عبور سیگنال‌هایی دودویی از میان ترکیباتی از گیت‌ها مورد استفاده قرار داد، که هر سیگنال بیانگر یک متغیر دودویی بوده و یک بیت از اطلاعات را حمل می‌کند.

سمبل‌های گرافیکی مورد استفاده برای سه نوع گیت در شکل ۴-۱ دیده می‌شوند. گیت‌ها، بلوک‌هایی سخت‌افزاری‌اند که با ورودی منطقی مناسبی، در خروجی خود 0 یا 1 تولید می‌نمایند. سیگنال ورودی xو yدرگیت‌های ANDو OR در یکی از چهار حالت ممکن قرار دارند: 00، 01، 10 و



(الف) گیت ANDدو ورودی (ب) گیت OR دو ورودی (پ) گیت NOTبرای معکوس کردن (وارونگر)

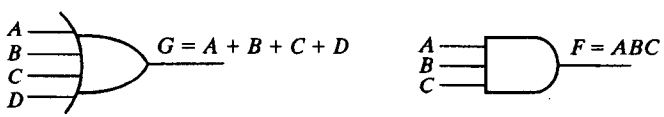
شکل ۴-۱. سمبل‌های مداری منطقی دیجیتال



شکل ۵-۱ . سیگنال‌های ورودی - خروجی برای گیت‌های (الف)، (ب) ، (پ) از شکل ۴-۱

11. این سیگنال‌ها همراه با خروجی خود در شکل ۵-۱ دیده می‌شوند. نمودارهای زمانی پاسخ هر گیت را برای چهار گیت فوق نشان می‌دهند. محور افقی نمودار زمانی، زمان و محور عمودی سیگنال‌ها را ضمن تغییر بین دو سطح ولتاژ ممکن نمایش می‌دهد. سطح پایین منطق 0 و سطح بالا منطق 1 را نشان می‌دهد. هنگامی در خروجی گیت AND منطق 1 وجود دارد که هر دو سیگنال ورودی در منطق 1 باشند. گیت OR هنگامی خروجی 1 دارد که یکی از سیگنال‌های ورودی در منطق 1 باشد. گیت NOT را معمولاً وارونگر یا معکوس‌گر هم می‌گویند. دلیل انتخاب این نام با توجه به پاسخ سیگنال در نمودار زمانی مشخص است، و در آن نشان داده شده است که سیگنال خروجی مفهوم منطق ورودی را معکوس کرده است.

گیت‌های AND و OR ممکن است بیش از دو ورودی داشته باشند. یک گیت AND با سه ورودی و یک OR با چهار ورودی در شکل ۶-۱ ملاحظه می‌شود. گیت AND سه ورودی به شرطی خروجی 1 دارد که هر سه ورودی آن 1 باشد. اگر هر یک از ورودی‌ها 0 باشند، خروجی AND برابر 0 خواهد بود. گیت OR چهار ورودی هنگامی خروجی 1 تولید می‌کند که یکی از ورودی‌ها در 1 منطقی باشد. خروجی هنگامی 0 می‌شود که همه ورودی‌ها در منطق 0 باشند.



(الف) گیت AND سه ورودی (ب) گیت OR چهار ورودی

شکل ۶-۱. مدار سوئیچینگ نمایش دهنده منطق دودویی

مسائل

۱ - ۱ اعداد مبنای 8 و 16 بین 16 تا 32 را لیست کنید. با استفاده از کاراکترهای A و B برای دو رقم آخر، اعداد 10تا 26 را در مبنای 12 لیست نمایید.

۱ - ۲ تعداد دقیق بایت‌ها در یک سیستم حاوی (الف) 32K بایتی، (ب) 64M بایتی و (پ) 6.4G بایتی چقدر است؟

۱ - ۳ بزرگترین عددی که می‌توان با 12 بیت نشان داد چند است؟ معادل دهدهی و شانزده شانزدهی آن چند است؟

۱ - ۴ اعداد زیر با مبناهای داده شده را به مبنای دهدهی تبدیل کنید: $(4310)_5$ و $(198)_{12}$.

۱ - ۵ برای صحیح بودن عملیات زیر، مبنای اعداد را در هر حالت معین کنید.
(الف) 14/2 = 5 ؛ (ب) 54/4 = 13 ؛ (پ) 24 + 17 = 40

۱ - ۶ حل معادله درجه دوم $x^2 - 11x + 22 = 0$ برابر $x = 3$ و $x = 6$ است. مبنای اعداد چیست؟

۱ - ۷ اعداد زیر را به دهدهی بیان کنید: $(10110.0101)_2$ ، $(16.5)_{16}$ و $(26.24)_8$.

۱ - ۸ اعداد دودویی زیر را به مبنای شانزده و مبنای 10 تبدیل نمایید:
(الف) 1.11010 ، (ب) 1110.10
بگویید چرا جواب دهدهی در (ب) 8 برابر (الف) است.

۱ - ۹ عدد مبنای شانزده 68BE را به دودویی و سپس از دودویی به مبنای هشت تبدیل کنید.

۱ - ۱۰ عدد دهدهی 345 را به دو روش به دودویی تبدیل نمایید:
(الف) مستقیماً به دودویی تبدیل نمایید. (ب) ابتدا به مبنای شانزده و سپس از مبنای شانزده به دودویی تبدیل سازید. کدام روش سریع‌تر است؟

۱ - ۱۱ تبدیل‌های زیر را عمل کنید:
(الف) عدد دهدهی 34.4375 را به دودویی تبدیل کنید.
(ب) معادل دودویی $\frac{1}{3}$ را تا 8 رقم بعد از نقطه ممیز بدست آورید. سپس از دودویی به دهدهی تبدیل کنید. جواب چقدر به $\frac{1}{3}$ نزدیک است؟
(پ) نتیجه دودویی در (ب) را به مبنای 16 ببرید. سپس نتیجه را به دهدهی تبدیل کنید. آیا جواب یکی است.

۱ - ۱۲ بدون تبدیل دهدهی جمع و ضرب‌های زیر را انجام دهید.
(الف) اعداد دودویی 1011 و 101 .
(ب) اعداد مبنای شانزده 2E و 34 .

۱ - ۱۳ تقسیم زیر را در دودویی انجام دهید: 101 : 1011111

۱ - ۱۴ متمم‌های 9 و 10 اعداد دهدهی زیر را بدست آورید.
(الف) 98127634 (ب) 72049900 (پ) 10000000 (ت) 00000000

۱ - ۱۵ (الف) متمم 16 عدد AF3B را بدست آورید.
(ب) AF3B را به دودویی تبدیل کنید.

(پ) متمم 2 جواب (ب) را پیدا کنید.

(ت) جواب در (پ) را به مبنای 16 تبدیل کنید و با پاسخ در (الف) مقایسه نمایید.

۱-۱۶ متمم 1 و 2 اعداد دودویی زیر را بدست آورید.

(الف) 11101010 (ب) 01111110 (پ) 00000001 (ت) 10000000 (ث) 00000000

۱-۱۷ روی اعداد بی‌علامت زیر با متمم 10 مفروق، تفریق انجام دهید. جایی که جواب منفی می‌شود، متمم 10 آن را بدست آورید و جلو آن علامت منها بگذارید. صحت پاسخ خود را تحقیق کنید.

(الف) 3049-7188 (ب) 2100-150 (پ) 7992-2997 (ت) 375-1321

۱-۱۸ روی اعداد دودویی بی‌علامت زیر با استفاده از متمم 2 مفروق، تفریق را اجرا کنید.

(الف) 11001−11011 (ب) 10101−110100

(پ) 110000−1011 (ت) 101011−101010

۱-۱۹ اعداد دهدهی زیر به فرم مقدار- علامت نشان داده شده است: 9826+ و 801+ . آنها را به متمم 10 تبدیل کنید و اعمال زیر را انجام دهید: (توجه کنید که حاصل جمع 10627+ بوده و شش رقم نیاز دارد).

(الف) (801+) + (9826+) (ب) (801−) + (9826+)

(پ) (801+) + (9826−) (ت) (801−) + (9826−)

۱-۲۰ عدد دهدهی 61+ و 27+ را با متمم 2_ علامت‌دار به دودویی تبدیل کنید و ارقام لازم را برای ذخیره اعداد در نظر بگیرید. سپس عبارات (61−) + (27+)، (61+) + (27−) و (61−) + (27−) را اجرا کنید. جواب‌ها را به دهدهی بازگردانید و تحقیق کنید که آنها صحیح‌اند.

۱-۲۱ دهدهی 9126 را به هر دو نوع کد BCD و اسکی تبدیل نمایید. برای اسکی، بیت توازن فرد در سمت چپ قرار داده می‌شود.

۱-۲۲ اعداد بی‌علامت دهدهی 965 و 672 را به BCD نشان دهید و سپس مراحل لازم برای جمع زدن آنها را طی کنید.

۱-۲۳ با استفاده از وزن‌های 1، 1، 3، 6 برای ارقام دهدهی یک کد دودویی وزین بسازید.

۱-۲۴ عدد دهدهی 6027 را (الف) به کد BCD، (ب) افزونی −3 و (پ) کد 2421 نشان دهید.

۱-۲۵ متمم 9 عدد 6027 را پیدا کنید و آن را به کد 2421 بیان نمایید. نشان دهید که نتیجه متمم 1، پاسخ (پ) در مسئله ۱-۲۴ است. این نشان می‌دهد که کد 2421 خود متمم است.

۱-۲۶ برای 52 ورق بازی نوعی کد دودویی تخصیص دهید.

۱-۲۷ عبارت "G.Boole" را با استفاده از یک کد هشت بیتی به اسکی بنویسید. بین حروف قسمت‌های مختلف نام و نام‌خانوادگی فاصله بگذارید. (جورج بول یک ریاضی‌دان قرن ۱۹ بود. جبر بول، که در فصل بعد معرفی شده است به اسم او نام‌گذاری شده است)

۱-۲۸ کدهای اسکی زیر را دیکد کنید: 1001010، 1100001، 1101110، 1100101، 0100000، 1000100، 1101111، 1100101.

۱-۲۹ در زیر رشته‌ای از کاراکترهای اسکی آمده‌اند که الگوی بیتی آن به منظور فشردگی به مبنای 16 تبدیل شده

است: 4A ، EF ، 68 ، 6E ، 20 ، C4 ، EF ، E5 . در میان 8 بیت در هر جفت رقم، سمت چپ‌ترین بیت مربوط به بیت توازن است. دیگر بیت‌ها کد اسکی را تشکیل می‌دهند.
(الف) هر یک را به فرم بیتی تبدیل کنید و سپس آن را به اسکی ببرید.
(ب) توازن به کار رفته را مشخص نمایید: فرد یا زوج.

۱-۳۰ در اسکی چند کاراکتر چاپ وجود دارد؟ چند تا از آنها کاراکترهای خاص هستند (نَه عدد و نَه حرف)؟

۱-۳۱ برای تبدیل حروف بزرگ به کوچک در اسکی کدام بیت باید عوض شود، و بالعکس؟

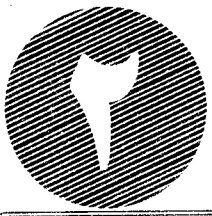
۱-۳۲ حالت یک ثبات 12 بیتی 100010010111 است. این محتوا برای حالات زیر چه چیز را نشان می‌دهد.
(الف) سه رقم دهدهی در BCD ؟ (ب) سه رقم دهدهی در کد افزونی 3 ؟
(پ) سه رقم دهدهی در کد 1–2–84 ؟ (ت) یک عدد دودویی؟

۱-۳۳ با بیت توازن زوج در سمت چپ‌ترین مکان، کد اسکی 10 رقم دهدهی را لیست کنید.

۱-۳۴ یک گیت AND سه ورودی با خروجی F و یک گیت OR سه ورودی با خروجی G را فرض کنید. ورودی‌ها A ، B و C اند. سیگنال‌های خروجی F و G ، به عنوان تابعی از سه ورودی ABC اند. هر 8 ترکیب ممکن را برای ABC به کار ببرید. (مشابه با نمودار زمان‌بندی شکل ۵-۱ عمل کنید).

مراجع

1. CAVANAGH, J. J. 1984. *Digital Computer Arithmetic.* New York: McGraw-Hill.
2. SCHMID, H. 1974. *Decimal Computation.* New York: John Wiley.
3. MANO, M. M. 1988. *Computer Engineering: Hardware Design.* Englewood Cliffs, NJ: Prentice-Hall.
4. NELSON, V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL 1997. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.



جبر بول و گیت‌های منطقی

۲-۱ تعاریف اولیه

جبر بول را می‌توان مانند هر سیستم منتجه ریاضی، به وسیله مجموعه‌ای از عناصر، یک مجموعه از الگوها و تعدادی اصول اثبات نشده یا بدیهیات تعریف نمود. یک مجموعه از عناصر کلکسیونی از اشیاء است که دارای خواص مشترکی باشند. اگر S یک مجموعه و x و y عناصر مشخصی از آن باشند، آنگاه $x \in S$ به این معنی است که x عضوی از مجموعه S و $y \notin S$ یعنی y عضوی از S نیست. یک مجموعه با تعداد قابل شمارشی از عناصر با یک جفت آکولاد مشخص می‌شود: $A = \{1, 2, 3, 4\}$، یعنی عناصر مجموعه A عبارتند از 1 ,2 ,3 و 4. یک عملگر دودویی روی یک مجموعه از عناصر، S، قانونی است که به هر جفت از عناصر S، یک عنصر منحصر به فرد از S را تخصیص دهد. به عنوان مثال رابطه $a * b = c$ را در نظر بگیرید. (*) را یک عملگر دودویی می‌خوانیم به شرطی که بتواند عنصر c را به جفت عنصر a و b منتسب نماید ضمن اینکه رابطه $a, b, c \in s$ معتبر باشد. با این وجود اگر $a, b \in S$ و $c \notin S$ باشد، (*) یک عملگر دودویی نیست.

اصول یک سیستم ریاضی، فرضیات اولیه را تشکیل می‌دهند که با استفاده از آنها می‌توان قوانین و تئوری‌ها و خواص سیستم را نتیجه گرفت. مهمترین اصول بکار رفته در فرموله کردن ساختارهای جبری عبارتند از:

۱- بسته بودن : یک مجموعه S نسبت به عملگر دودویی بسته است به شرطی که برای هر جفت عنصر از S، این عملگر عنصر منحصر به فردی از آن را به جفت عنصر منتسب نماید. به عنوان مثال، مجموعه اعداد طبیعی $N = \{1, 2, 3, 4, ...\}$ را نسبت به عملگر جمع (+) بسته گوییم زیرا برای هر دو عنصر $a \in s$ و b عنصر دیگری مانند $c \in N$ می‌توان یافت بطوری که $a+b= c$ باشد. مجموعه اعداد طبیعی نسبت به عملگر تفریق بسته نیست چون داریم $2 - 3 = -1$ در

حالی که $2, 3 \in N$ و $(-1) \notin N$ است.

۲- اصل شرکت‌پذیری : یک عملگر دودویی (*) روی مجموعه S شرکت‌پذیر است اگر داشته باشیم

$$(x * y) * z = x * (y * z) \qquad \text{به ازای همه مقادیر } x, y, z \in S$$

۳- اصل جابجایی : یک عملگر (*) روی یک مجموعه دارای خاصیت جابجایی است هرگاه

$$x * y = y * x \qquad \text{به ازای هر } x, y \in S$$

۴- عنصر شناسه : مجموعه S نسبت به عملگر (*) روی مجموعه S دارای عنصر شناسه است، اگر عنصر $e \in S$ با خاصیت زیر موجود باشد.

$$e * x = x * e = x \qquad \text{به ازای هر } x \in S$$

مثال: عنصر 0 یک عنصر شناسه نسبت به عملگر (+) روی مجموعه اعداد صحیح $I = \{..., -3, -2, -1, 0, 1, 2, 3, ...\}$ است، چون

$$x + 0 = 0 + x = x \qquad \text{به ازای هر } x \in I$$

مجموعه اعداد طبیعی N دارای عنصر شناسه نیست زیرا 0 جزو مجموعه نمی‌باشد.

۵- معکوس : مجموعه‌ای چون S با عنصر شناسه e نسبت به عملگر (*) دارای معکوس است به شرطی که برای هر $x \in S$، یک $y \in S$ وجود داشته باشد به نحوی که

$$x * y = e$$

مثال: در مجموعه اعداد صحیح I، با e=0، معکوس عنصر a برابر $(-a)$ است چون $a+(-a)=0$

۶- اصل توزیع‌پذیری : اگر (*) و (.) دو عملگر روی مجموعه S باشند، (*) را روی (.) توزیع‌پذیر گوییم هرگاه

$$x * (y \cdot z) = (x * y) \cdot (x * z)$$

مثالی جبری در این مورد میدان یا حوزه است. میدان مجموعه‌ای از عناصر است، همواره با دو عملگر دودویی، که هر یک دارای خواص 1 تا 5 بوده و هر دو عملگر برای تشکیل خاصیت 6 با یکدیگر ترکیب می‌شوند. مجموعه اعداد حقیقی، همراه با عملگرهای دودویی (+) و (.)، میدان اعداد حقیقی را تشکیل می‌دهند. میدان اعداد حقیقی مبنای جبر معمولی و حساب است. عملگرها و اصول دارای مفاهیم زیر هستند:

عملگر دودویی (+) جمع را تعریف می‌کند.

شناسه جمع، 0 است.

معکوس جمع، تفریق می‌باشد.

عملگر دودویی (.) ضرب را تعریف می‌نماید.

شناسه ضرب 1 می‌باشد.

معکوس ضرب $a = 1/a$ تقسیم را تعریف می‌کند، یعنی $a \cdot (1/a) = 1$.

تنها اصل توزیع‌پذیری قابل اعمال مربوط به عملگر (.) روی (+) است:

$$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$$

۲-۲ تعریف اصول اساسی جبر بول

در سال ۱۸۵۴ جورج بول یک برخورد سیستماتیک با منطق را معرفی نمود و برای اهداف خود یک سیستم جبری راکه امروزه آن را جبر بول می‌نامیم پایه‌ریزی کرد. در سال ۱۹۳۸ نیز سی.ای. شانون یک جبر بول دو مقداری به نام جبر سوئیچینگ را معرفی کرد که در آن خواص مدارهای سوئیچینگ با این جبر قابل ارائه است. برای تعریف مستدل جبر بول، ما اصول فرموله شده به وسیله ای. .ی. هانتینگتون در سال ۱۹۰۴ را بکار می‌بریم.

جبر بول یک ساختار جبری است که با عناصر مجموعه، یعنی B، همراه با دو عملگر دودویی (+) و (.) تعریف می‌شود به شرطی که اصول زیر (هانتینگتون) در آن معتبر باشد.

۱- (a) مجموعه نسبت به عملگر (+) بسته باشد.

(b) مجموعه نسبت به عملگر (.) بسته باشد.

۲- (a) یک عنصر شناسه 0 برای (+) وجود داشته باشد.

(b) یک عنصر شناسه 1 برای (.) وجود داشته باشد.

۳- (a) مجموعه نسبت به (+) دارای خاصیت جابجایی باشد: $x + y = y + x$

(b) مجموعه نسبت به (.) دارای خاصیت جابجایی باشد: $x \,.\, y = y \,.\, x$

۴- (a) (.) نسبت به (+) توزیع‌پذیر است: $x \,.\, (y + z) = (x \,.\, y) + (x \,.\, z)$

(b) (+) نسبت به (.) توزیع‌پذیر است: $x + (y \,.\, z) = (x + y) \,.\, (x + y)$

۵- برای هر عنصر $x \in B$، عنصری مثل $x' \in B$ وجود دارد (به آن متمم x می‌گوییم) به نحوی که:

(a) $x + x' = 1$ و (b) $x \,.\, x' = 0$

۶- حداقل دو عنصر $x, y \in B$ وجود دارد به نحوی که $x \neq y$ باشد.

با مقایسه جبر بول با حساب و جبر معمولی (حوزه یا میدان اعداد حقیقی) تفاوت‌های زیر قابل ملاحظه‌اند:

۱- اصول هانتینگتون فاقد اصل شرکت‌پذیری است. با این وجود، این اصول برای جبر بول معتبر و برای هر دو عملگر از دیگر اصول قابل استنتاج است.

۲- اصل توزیع‌پذیری (+) روی (.)، یعنی $x + (y \,.\, z) = (x + y) \,.\, (x + z)$، برای جبر بول معتبر است، ولی در جبر معمولی قابل قبول نیست.

۳- جبر بول دارای معکوس‌های جمع و ضرب نیست؛ بنابراین عملگرهای تفریق و تقسیم وجود ندارند.

۴- اصل ۵ عملگری به نام متمم را معرفی می‌نماید که در جبر معمولی وجود ندارد.

۵- جبر معمولی در مورد اعداد حقیقی بحث می‌کند، که یک مجموعه با بی‌نهایت عنصر را شامل می‌شود. جبر بول در مورد مجموعه‌ای از عناصر، B، بحث می‌نماید که هنوز آن را معرفی نکرده‌ایم، ولی بعداً در جبر بول دو مقداری یا دو ارزشی معرفی خواهد شد (کاربرد بعدی ما از این جبر مورد توجه است)، و در آن B بصورت مجموعه‌ای از دو عنصر 0 و 1 تعریف می‌شود.

جبر بول از بعضی لحاظ با جبر معمولی هماهنگی دارد. انتخاب سمبل‌های (+) و (.) به این علت مورد توجه قرار گرفته‌اند تا دستکاری جبر بول به وسیله شخصی که با جبر معمولی آشنایی دارد، ساده گردد. گرچه وی می‌تواند بخشی از دانش خود از جبر معمولی را در جبر بول بکار ببرد، ولی مبتدیان باید مراقب باشند تا قوانین جبر معمولی را در جاهایی که قابل استفاده نیستند، بکار نبرند.

لازم است تا اختلاف بین یک مجموعه از عناصر متعلق به یک ساختار جبری و متغیرهای یک سیستم جبری را بدانیم. مثلاً، اجزاء حوزه اعداد حقیقی، اعداد هستند، در صورتی که متغیرهایی مانند a ، b و c که در جبر معمولی بکار می‌روند، سمبل‌هایی هستند که به جای اعداد حقیقی بکار می‌روند. بطور مشابه در جبر بول مجموعه B دارای متغیرهایی مانند x، y و z می‌باشد که صرفاً سمبل هستند و عناصر را نشان می‌دهند. در اینجا لازم است بدانیم که به منظور داشتن یک جبر بول باید:

۱- عناصر مجموعه B مشخص باشند.

۲- قوانین عمل دو عملگر دودویی معین باشند.

۳- مجموعه عناصر، B ، همراه با دو عملگر، برای شش اصل هانتینگتون معتبر باشد.

بسته به انتخاب عناصر B و قوانین عملیات، می‌توان چندین جبر بول را فرموله کرد. در ادامه کار ما فقط با جبر دو ارزشی که تنها دو عنصر دارد، سروکار خواهیم داشت. جبر بول دو ارزشی در تئوری مجموعه‌ها و منطق کاربرد دارد. هدف ما در این کتاب کاربرد جبر بول در مدارهای منطقی گیتی است.

جبر بول دو ارزشی

جبر بول دو ارزشی روی مجموعه دو عنصری، $B = \{0, 1\}$ ، به همراه قوانین برای دو عملگر دودویی (+) و (.) که در جدول زیر نشان داده شده تعریف می‌شود (قانون عملگر متمم برای تصدیق اصل ۵ است):

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

x	x′
0	1
1	0

این قوانین دقیقاً مثل اعمال AND، OR و NOT در جدول ۸-۱ می‌باشند. اکنون نشان می‌دهیم که اصول هانتینگتون برای مجموعه $B=\{0, 1\}$ و دو عملگرهای دودویی که قبلاً تعریف شده‌اند، معتبر است.

۱- با توجه به جداول، بسته بودن کاملاً روشن است زیرا نتیجه هر عملگر 1 یا 0 بوده و $1,0 \in B$ می‌باشند.

۲- با توجه به جداول می‌بینیم که $0 + 1 = 1 + 0 = 1$ (a) $0 + 0 = 0$

(b) $1.1 = 1$ $1.0 = 0.1 = 0$

این روابط بیانگر وجود عناصر شناسه 0 برای (+) و 1 برای (.) ، طبق تعریف اصل ۲ می‌باشند.

۳- اصول جابجایی با توجه به تقارن جداول عملگرها آشکار است.

۴- (a) صحت اصل توزیع‌پذیری $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ را با ایجاد جدول برای تمام مقادیر ممکن x, y و z، می‌توان تحقیق کرد. برای هر ترکیب، $x \cdot (y + z)$ را بدست آورده و نشان می‌دهیم که برابر $(x \cdot z) + (x \cdot y)$ است.

x	y	z	y + z	x · (y + z)	x · y	x · z	(x · y) + (x · z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

(b) صحت اصل توزیع‌پذیری (+) روی (.) را نیز می‌توان مثل بند قبل تحقیق نمود.

۵- با استفاده از جدول متمم، به سادگی می‌توان دید که:

(a) $x + x' = 1$، زیرا $1 + 1 = 0 + 0' = 0$ و $1 + 1' = 1 + 0 = 1$ است

(b) $x \cdot x' = 0$، زیرا $0.1 = 0.0' = 0$ و $1.1' = 1.0 = 0$ است، که اصل ۵ را تصدیق می‌کند.

۶- اصل ۶ نیز صادق است زیرا جبر بول دو ارزشی دارای دو مقدار مجزای 0 و 1 با $0 \neq 1$ است.

تا اینجا ما یک جبر دو ارزشی با عملگرهای AND و OR و یک عملگر متمم معادل با NOT ایجاد کردیم. بنابراین جبر بول به روش مستدل ریاضی بنا گردید و نشان داده شد که معادل با منطق دودویی غیرمستدل بخش ۹-۱ است. بیان غیرمستدل برای درک کاربرد جبر بول در مدارهای گیتی مفید است. روش مستدل برای بیان و ایجاد تئوری‌ها و خواص سیستم جبری مورد توجه است. جبر بول دو ارزشی تعریف شده در این بخش را "جبر سوئیچینگ" نیز می‌نامند. برای تأکید بر تشابه بین جبر بول دو ارزشی و دیگر سیستم‌های دودویی، این جبر در بخش ۹-۱ "منطق دودویی" نامیده شد. از این پس، کلمه "دو ارزشی" را در بحث‌های بعدی از جبر بول حذف می‌کنیم.

۳-۲ قضایای اصلی و خواص جبر بول

دوگانگی

اصول هانتینگتون بصورت جفت جفت لیست شده‌اند و بصورت بخش‌های (a) و (b) مشخص شدند. هر یک از این دو را با تعویض عملگرها و عنصر شناسه می‌توان از دیگری بدست آورد. این خاصیت در جبر بول به اصل دوگانگی معروف است. خصوصیات فوق بیان می‌دارد که هر عبارت جبری منتج از اصول جبر بول با تعویض عملگرها و شناسه‌ها باز هم معتبر باقی می‌ماند. در جبر بول دو ارزشی، عناصر شناسه و عناصر مجموعه B یکسانند: یعنی 1 و 0اند. اصل دوگانگی کاربردهای متعددی دارد. اگر دوگان

یک عبارت جبری موردنظر باشد تنها کافی است عملگرهای AND و OR تعویض و 0ها به 1 و 1ها به 0 تبدیل گردند.

تئوری‌های اساسی

جدول (۱-۲) شش تئوری و چهار اصل از جبر بول را لیست کرده است. به خاطر سادگی از علامت (.) مادامی که اشتباه ایجاد نشود، صرف‌نظر شده است. تئوری‌ها و اصول لیست شده اساسی‌ترین روابط در جبر بول‌اند. تئوری‌ها نیز مانند اصول بصورت جفت جفت ارائه شده‌اند و هر رابطه دوگان زوج خود است. اصول، بدیهیات ساختار جبری بوده و اثباتی لازم ندارند. تئوری‌ها باید با توجه به اصول ثابت شوند، اثبات تئوری‌ها با یک متغیر در زیر نشان داده شده‌اند. در سمت مقابل روابط ، شماره اصل بکار رفته نوشته شده است.

تئوری ۱ (a): $x + x = x$

$$
\begin{aligned}
x + x &= (x + x) \cdot 1 && \text{به وسیله اصل: ۲ (b)}\\
&= (x + x)(x + x') && \text{(a) ۵}\\
&= x + xx' && \text{(b) ۴}\\
&= x + 0 && \text{(b) ۵}\\
&= x && \text{(a) ۲}
\end{aligned}
$$

تئوری ۱ (b): $x \cdot x = x$

$$
\begin{aligned}
x \cdot x &= xx + 0 && \text{به وسیله اصل: ۲ (a)}\\
&= xx + xx' && \text{(b) ۵}\\
&= x(x + x') && \text{(a) ۴}\\
&= x \cdot 1 && \text{(a) ۵}\\
&= x && \text{(b) ۲}
\end{aligned}
$$

جدول ۱-۲. اصول و تئوری‌های جبر بول

اصل ۲	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
اصل ۱	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
تئوری ۱	(a)	$x + x = x$	(b)	$x \cdot x = x$
تئوری ۲	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
تئوری ۳ رجعت		$(x')' = x$		
اصل ۳ جابجایی	(a)	$x + y = y + x$	(b)	$xy = yx$
تئوری ۴ شرکت پذیری	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
اصل ۴ توزیع پذیری یا پخش	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
تئوری ۵ دمورگان	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
تئوری ۶ جذب	(a)	$x + xy = x$	(b)	$x(x + y) = x$

توجه کنید که تئوری ۱ (b) دوگان ۱ (a) است و هر مرحله از اثبات در بخش (b) دوگان بخش (a) می‌باشد. به این ترتیب هر تئوری دوگان از اثبات زوجش حاصل می‌گردد.

تئوری ۲ (a): $x + 1 = 1$

$x + 1 = 1 \cdot (x + 1)$	به وسیله اصل : ۲ (b)
$= (x + x')(x + 1)$	۵ (a)
$= x + x' \cdot 1$	۴ (b)
$= x + x'$	۲ (b)
$= 1$	۵ (a)

تئوری ۲ (b): براساس دوگانگی $x \cdot 0 = 0$

تئوری ۳: $x = (x')'$. با توجه به اصل ۵، داریم $x + x' = 1$ و $x \cdot x' = 0$، که متمم x را تعریف می‌کند. متمم x' برابر x است و در نتیجه همان $(x')'$ می‌باشد. بنابراین، چون متمم منحصر به فرد است داریم $(x')' = x$.

با استفاده از اصول و تئوری‌های اثبات شده قبلی می‌توان تئوری‌های دو یا سه متغیره را بصورت جبری ثابت کرد. مثلاً: تئوری جذب را در نظر بگیرید.

تئوری ۶ (a): $x + xy = x$

$x + xy = x \cdot 1 + xy$	به وسیله اصل : ۲ (b)
$= x(1 + y)$	۴ (a)
$= x(y + 1)$	۳ (a)
$= x \cdot 1$	۲ (a)
$= x$	۲ (b)

تئوری ۶ (b): براساس دوگانگی $x(x + y) = x$

صحت تئوری‌های جبر بول را می‌توان با کمک جداول درستی هم تحقیق کرد. در جداول درستی، هر دو طرف رابطه چک می‌شوند تا نتایج مشابهی در ازاء همه ترکیبات متغیرها حاصل شود. تئوری جذب زیر صحت اولین جدول درستی را نشان می‌دهد.

x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

اثبات جبری اصل شرکت‌پذیری و تئوری دمورگان طولانی بوده و در اینجا آورده نمی‌شوند. با این وجود اعتبار آنها با جداول درستی نشان داده می‌شود. مثلاً جدول درستی برای اولین تئوری دمورگان

$(x + y)' = x'y'$ در زیر نشان داده شده است.

x	y	$x + y$	$(x + y)'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

x'	y'	$x'y'$
1	1	1
1	0	0
0	1	0
0	0	0

تقدم عملگرها

در ارزیابی عبارات جبر بول تقدم اول با پرانتز، دوم با NOT، سوم با AND و چهارم با OR است. به بیان دیگر، عبارت داخل پرانتز باید قبل از سایر عملگرها ارزیابی شود. عملگر مقدم بعدی متمم است. پس از آن AND و بالاخره OR قرار دارد. به عنوان مثال، جدول درستی را برای تئوری دمورگان تشکیل می‌دهیم سمت چپ عبارت $(x + y)'$ است. بنابراین داخل پرانتز ابتدا ارزیابی می‌شود و سپس نتیجه متمم می‌گردد. سمت راست عبارت $x'y'$ است. بنابراین متمم x و y هر دو ابتدا ارزیابی شده و حاصل AND می‌گردد. توجه کنید که در محاسبات معمولی هم روال مشابهی (به جز برای متمم) برای ضرب و جمع به ترتیب به جای AND و OR برقرار است.

۲-۴ توابع بول

جبر بول جبری است که با متغیرهای دودویی و عملیات منطقی سروکار دارد. یک تابع به وسیله یک عبارت جبری متشکل از متغیرهای دودویی، ثابت‌های 0 و 1 و سمبل‌های عملیاتی منطقی تشکیل شده است. برای مقدار مفروضی از متغیرهای دودویی، تابع می‌تواند 1 یا 0 باشد. به عنوان مثال تابع بولی زیر را در نظر بگیرید:

$$F_1 = x + y'z$$

اگر x برابر 1 باشد و یا هر دو y' و z برابر 1 باشند تابع F_1 برابر 1 است. در غیر این صورت F_1 برابر 0 خواهد بود. عمل متمم، وقتی $y' = 1$ باشد، $y = 0$ را دیکته می‌کند. بنابراین، می‌توانیم بگوییم که اگر $x = 1$ یا اگر $y = 0$ و $z = 1$ باشد آنگاه $F_1 = 1$ خواهد بود. یک تابع بول رابطه‌ای منطقی را بین متغیرها بیان می‌کند. این تابع با تعیین مقدار دودویی عبارت برحسب همه مقادیر ممکن متغیرها ارزیابی می‌شود.

یک جدول بولی بصورت یک جدول درستی هم می‌تواند نشان داده شود. جدول درستی لیستی از 1ها و 0ها است که به متغیرهای دودویی تخصیص می‌یابد، و ستونی که مقدار نتایج را برای هر ترکیب نشان می‌دهد. تعداد سطرها در جدول درستی 2^n است، که n تعداد متغیرها در تابع است. ترکیبات دودویی برای جدول درستی از شمارش اعداد دودویی و از 0 تا $2^n - 1$ بدست می‌آید. جدول (۲-۲)، جدول درستی تابع F_1 را نشان می‌دهد. در این جدول هشت ترکیب دودویی ممکن برای تخصیص بیتی

جدول ۲-۲. جدول درستی برای F_1 و F_2

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

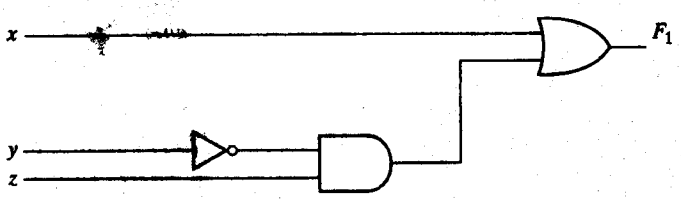
به سه متغیر x و y و z وجود دارد. ستونی که برچسب F_1 دارد در ازاء هر ترکیب 0 یا 1 است. جدول نشان می‌دهد که وقتی $x = 1$ یا $yz = 01$ باشد تابع F_1 برابر 1 است. در غیر این صورت 0 خواهد بود.

یک تابع بول را می‌توان از یک عبارت جبری به یک نمودار مداری متشکل از گیت‌های منطقی تبدیل کرد. نمودار مدار منطقی F_1 در شکل ۱-۲ نشان داده شده است. برای تولید متمم ورودی y وارونگری (معکوس‌گر) وجود دارد. برای جمله $y'z$ یک گیت AND و برای ترکیب آن دو یک گیت OR بکار رفته است. در نمودارهای مدار منطقی، متغیرهای تابع به عنوان ورودی مدار و متغیر دودویی F_1 به عنوان خروجی مدار در نظر گرفته می‌شوند.

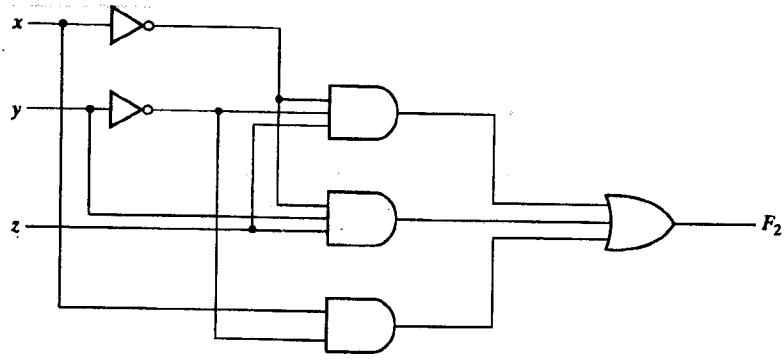
برای نمایش F_1 در یک جدول درستی تنها یک راه وجود دارد. با این وجود، وقتی تابع به فرم یک عبارت جبری است، می‌تواند به فرم‌های متفاوتی نشان داده شود. عبارت خاصی که برای مشخص کردن تابع مورد استفاده قرار می‌گیرد اتصالات میان گیت‌ها در نمودار مدار منطقی را دیکته می‌نماید. گاهی اوقات ممکن است با دستکاری یک عبارت بولی توسط قوانین جبر بول، عبارت ساده‌تری برای یک تابع بدست آوریم و بنابراین تعداد گیت‌ها در مدار و تعداد ورودی‌ها به هر گیت را کاهش دهیم. مثلاً تابع بولی زیر را در نظر بگیرید:

$$F_2 = x'y'z + x'yz + xy'$$

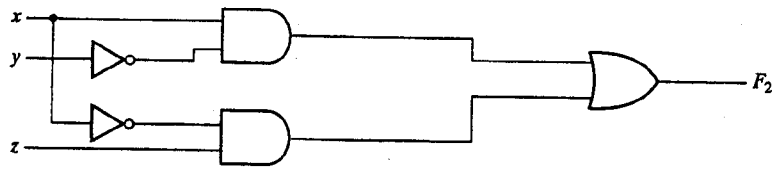
این تابع در شکل ۲-۲ (الف) پیاده‌سازی شده است. متغیرهای x و y به کمک وارونگر متمم شده‌اند تا x' و y' بدست آیند. سه جمله در عبارت با سه گیت AND پیاده‌سازی شده‌اند. گیت OR نیز، OR منطقی سه جمله را فراهم می‌سازد. جدول درستی F_2 در جدول ۲-۲ آمده است. وقتی که $xyz = 001$ یا 011 و یا $xy = 10$ باشد (بدون توجه به z)، تابع برابر 1 است، در غیر این صورت 0 است. این شرایط چهار 1 و چهار 0 برای F_2 تولید می‌کنند.



شکل ۱-۲. پیاده‌سازی با گیت $F_1 = x + y'z$



$F_2 = x'y'z + x'yz + xy'$ (الف)



$F_2 = xy' + x'z$ (ب)

شکل ۲-۲. پیاده‌سازی تابع بول F_2 با گیت

اکنون ساده‌سازی ممکن برای تابع را با اعمال بعضی از ویژگی‌های جبر بول ملاحظه کنید:

$$F_2 = x'y'z + x'yz + xy' = x'z(y' + y) + xy' = x'z + xy'$$

تابع تنها به دو جمله کاهش یافته و قابل پیاده‌سازی با گیت مطابق شکل ۲-۲ (ب) است. بدیهی است که مدار شکل (ب) ساده‌تر از (الف) می‌باشد، ولی هر دو یک تابع را پیاده‌سازی می‌کنند. تساوی دو عبارت را می‌توان به کمک جدول درستی هم تحقیق کرد. عبارت ساده شده، وقتی xz = 01 یا xy = 10 باشد، برابر 1 است. این تابع هم همان چهار 1 را در جدول تولید می‌کند. چون هر دو عبارت جدول درستی یکسانی را تولید می‌کنند به آنها معادل گوییم. بنابراین، دو مدار به ازاء همه ترکیبات ممکن متغیرهای ورودی، خروجی‌های یکسانی دارند. هر دو عبارت تابع یکسانی را تولید می‌کنند ولی یکی از آنها گیت‌ها و ورودی‌های کمتری نسبت به دیگری دارد و بنابراین چون سیم‌بندی و قطعات کمتری نیاز است بر دیگری ترجیح داده می‌شود.

دستکاری جبری

وقتی که یک عبارت بولی با گیت‌های منطقی پیاده‌سازی شود، هر جمله به یک گیت نیاز دارد و هر متغیر در جمله یک ورودی به یک گیت است. ما لیترال را یک متغیر تک در یک جمله می‌نامیم که ممکن است متمم شود یا نشود. مثلاً تابع شکل ۲-۲ (الف) دارای سه جمله و هشت لیترال است، دو تابع شکل ۲-۲ (ب) دو جمله و چهار لیترال دارد. اغلب در تابع بول با کاهش تعداد جملات، تعداد لیترال‌ها، یا

هر دو مدار ساده‌تری حاصل می‌شود. هدف از دستکاری جبر بول غالباً کاهش یک عبارت به منظور دستیابی به یک مدار ساده‌تر است. توابعی که تا پنج متغیر دارند قابل ساده‌سازی با روش نقشه که در فصل بعد آمده، هستند. برای توابع بول پیچیده‌تر، طراحان دیجیتال از برنامه‌های کامپیوتر کوچک‌سازی استفاده می‌کنند. تنها روش دستی موجود، روال سعی و کاهش می‌باشد که از روابط ساده و تکنیک‌های دستکاری آشنا استفاده می‌کند. مثال‌های زیر دستکاری جبری جبر بول را نشان می‌دهد.

مثال ۱-۲

توابع بولی زیر را با حداقل لیترال‌ها ساده کنید.

۱- $x(x' + y) = xx' + xy = 0 + xy = xy.$

۲- $x + x'y = (x + x')(x + y) = 1(x + y) = x + y.$

۳- $(x + y)(x + y') = x + xy + xy' + yy' = x(1 + y + y') = x.$

۴-
$$\begin{aligned} xy + x'z + yz &= xy + x'z + yz(x + x') \\ &= xy + x'z + xyz + x'yz \\ &= xy(1 + z) + x'z(1 + y) \\ &= xy + x'z. \end{aligned}$$

۵- از دوگان تابع ۴ $(x + y)(x' + z)(y + z) = (x + y)(x' + z)$

❑

توابع ۱ و ۲ دوگان یکدیگرند، و در مراحل حلی خود نیز از خاصیت دوگانگی استفاده کرده‌اند. راه ساده‌تری برای ساده‌سازی تابع ۳ به کمک اصل ۴ (b) در جدول ۱-۲ آمده است: یعنی $(x + y)(x + y') = x + yy' = x$. چهارمین تابع این واقعیت را بیان می‌دارد که یک افزایش در تعداد لیترال‌ها گاهی منجر به عبارت نهایی ساده‌تر می‌گردد. تابع ۵ مستقیماً ساده نشده است، ولی می‌تواند با استفاده از دوگان مراحل مربوط به تابع ۴ بدست آید. توابع ۴ و ۵ را به عنوان تئوری وفاق می‌شناسند.

متمم یک تابع

متمم یک تابع F برابر F' است و از تعویض 0ها با 1 و 1ها با 0 در مقدار F بدست می‌آید. متمم یک تابع را می‌توان بصورت جبری از تئوری دمورگان نیز بدست آورد. جفت تئوری مذکور در جدول ۱-۲ برای دو متغیر داده شد. تئوری‌های دمورگان به سه یا چند متغیر هم قابل گسترش‌اند. با استفاده از اصول و تئوری‌های لیست شده در جدول ۱-۲، فرم سه متغیره اولین تئوری دمورگان به طریق زیر ثابت می‌شود.

$$\begin{aligned} (A + B + C)' &= (A + x)' && \text{با فرض } B + C = x \\ &= A'x' && \text{با تئوری ۵ (a) دمورگان} \\ &= A'(B + C)' && B + C = x \text{ را جایگزین کنید} \\ &= A'(B'C') && \text{با تئوری ۵ (a) دمورگان} \\ &= A'B'C' && \text{با تئوری ۴ (b) شرکت‌پذیری} \end{aligned}$$

تئوری‌های دمورگان برای هر تعدادی از متغیرها، مشابه حالت دو متغیره بوده و با روش جایگزینی متوالی، مشابه روشی که در فوق مشاهده شد، می‌توان آن را بدست آورد. فرم عمومی تئوری دمورگان به صورت زیر است:

$$(A + B + C + D + \ldots + F)' = A'B'C'D' \ldots F'$$
$$(ABCD \ldots F)' = A' + B' + C' + D' + \ldots + F'$$

این تئوری بیان می‌دارد که متمم یک تابع با تعویض عملگرهای AND و OR و متمم کردن هر لیترال حاصل می‌شود.

مثال ۲-۲

متمم توابع $F_1 = x'yz' + x'y'z$ و $F_2 = x(y'z' + yz)$ را بدست آورید. متمم‌ها را با اعمال هر تعداد تئوری دمورگان بصورت زیر بدست آورید:

$$F_1' = (x'yz' + x'y'z)' = (x'yz')'(x'y'z)' = (x + y' + z)(x + y + z')$$
$$F_2' = [x(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')'(yz)'$$
$$= x' + (y + z)(y' + z')$$

□

روال ساده‌تری برای بدست آوردن متمم یک تابع این است که دوگان تابع و متمم هر لیترال بدست آید. این روش با توجه به فرم عمومی تئوری دمورگان نتیجه می‌شود. به خاطر داشته باشید که دوگان یک تابع با تبدیل عملگر AND به OR و تبدیل 1ها و 0ها به یکدیگر بدست می‌آید.

مثال ۲-۳

متمم توابع F_1 و F_2 مثال ۲-۲ را با استفاده از دوگان‌ها و متمم‌های هر لیترال بدست آورید.

۱) $F_1 = x'yz' + x'y'z'$

دوگان تابع F_1 برابر است با: $(x' + y' + z')\ (x' + y' + z)$

متمم هر لیترال برابر است با: $(x + y' + z)\ (x + y + z') = F_1'$

۲) $F_2 = x(y'z' + yz)$

دوگان تابع F_2 برابر است با: $x + (y' + z')\ (y + z)$

متمم هر لیترال برابر است با: $x' + (y + z)\ (y' + z') = F_2'$

□

۲-۵ فرم‌های استاندارد و متعارف

مینترم‌ها و ماکسترم‌ها

یک متغیر دودویی ممکن است به فرم معمولی (x) یا متمم (x') ظاهر شود. اکنون تصور کنید که دو

جدول ۳-۲ . مینترم‌ها و ماکسترم‌ها برای سه متغیر دودویی

			مینترم‌ها		ماکسترم‌ها	
x	y	z	جمله	علامت	جمله	علامت
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

متغیر دودویی x و y با عملگر AND با هم ترکیب شوند. چون هر متغیر ممکن است به هر یک از دو شکل فوق ظاهر گردد، چهار ترکیب برای آنها متصور است: $x'y'$، $x'y$، xy' و xy. هر یک از این چهار جمله AND را یک مینترم یا یک جمله ضرب استاندارد گویند. بطور مشابه n متغیر را می‌توان ترکیب کرده و 2^n مینترم به وجود آورد. 2^n مینترم مختلف را می‌توان با روشی مشابه با آنچه در جدول ۳-۲ آمده، نشان داد. اعداد دودویی از صفر تا $2^n - 1$ زیر ستون n متغیر لیست شده‌اند. هر مینترم از AND تمام n متغیر بدست می‌آید که در آن هر متغیر پریم‌دار متعلق به بیت 0 و بدون پریم با 1 نشان داده می‌شود. سمبل هر مینترم نیز در جدول با m_j نشان داده شده است. که در آن J معادل دهدهی عدد دودویی مربوط به مینترم است.

به طریق مشابهی، n متغیر یک جمله OR تشکیل می‌دهند که هر متغیر ممکن است پریم‌دار یا بدون پریم باشد. 2^n ترکیب ممکن را ماکسترم یا جمع استاندارد گویند. هشت ماکسترم برای سه متغیر، همراه با سمبل آنها در جدول ۳-۲ لیست شده‌اند. هر 2^n ماکسترم برای n متغیر بطریق مشابهی حاصل می‌شود. هر ماکسترم از یک جمله OR با n متغیر بدست می‌آید که در آن متغیر پریم‌دار با 1 و بدون پریم با 0 نشان داده می‌شود. توجه کنید که هر ماکسترم، متمم مینترم مربوطه‌اش می‌باشد و بالعکس.

یک تابع بول می‌تواند بصورت جبری با استفاده از جدول درستی و با تشکیل مینترم‌های هر ترکیب از متغیرهایی که برای تابع، 1 را تولید می‌کنند، و اجرای عملگر روی OR همه این جملات ایجاد شود. مثلاً f_1 در جدول ۴-۲ با ترکیبات 001، 100 و 111 بصورت $x'y'z$, $xy'z'$ و xyz بیان می‌شود. چون هر

جدول ۴-۲ . توابع سه متغیره تابع f_1 تابع f_2

x	y	z	تابع f_1	تابع f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

یک از این مینترم‌ها $f_1 = 1$ را ایجاد می‌نمایند پس:

$$f_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

به سادگی می‌توان نشان داد که:

$$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

این مثال‌ها خصوصیت مهمی از جبر بول را به نمایش می‌گذارند: یعنی هر تابع بولی را می‌توان بصورت جمع مینترم‌ها نشان داد "جمع به معنی OR جملات است".

اکنون متمم تابع بول را ملاحظه نمایید، می‌توان آن را با تشکیل مینترم‌هایی در جدول درستی که 0 تابع را تولید می‌کنند، ایجاد کرد و سپس آنها را OR نمود. متمم f_1 چنین است:

$$f_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

اگر متمم f'_1 را بدست آوریم تابع f_1 بدست خواهد آمد.

$$\begin{aligned} f_1 &= (x + y + z)(x + y' + z)(x' + y + z')(x' + y' + z) \\ &= M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \end{aligned}$$

بطور مشابه، می‌توان عبارت f_2 را از جدول بدست آورد:

$$\begin{aligned} f_2 &= (x + y + z)(x + y + z')(x + y' + z)(x' + y + z) \\ &= M_0 M_1 M_2 M_4 \end{aligned}$$

این مثال‌ها نیز دومین خاصیت جبر بول را به نمایش می‌گذارند: هر تابع بول را می‌توان بصورت ضرب ماکسترم‌ها (ضرب به معنی AND است) در آورد. روال تهیه ضرب ماکسترم‌ها مستقیماً از جدول درستی به فرم زیر میسر است. برای هر ترکیبی از متغیرها ماکسترم‌هایی که در تابع 0 تولید می‌کنند را تشکیل دهید، و سپس AND همه ماکسترم‌ها را بدست آورید. توابع بول که بصورت جمع مینترم‌ها یا ضرب ماکسترم بیان شوند را فرم متعارف نامند.

مجموع مینترم‌ها

قبلاً بیان شد که برای هر n متغیر دودویی 2^n مینترم مجزا وجود دارد و هر تابع بولی می‌تواند بصورت مجموعی از مینترم‌ها در آید. مینترم‌هایی که جمع آنها توابع بول را تعریف می‌کنند، آنهایی هستند که 1های تابع را در جدول درستی تشکیل می‌دهند. چون تابع در قبال هر مینترم می‌تواند 0 و یا 1 باشد، و چون 2^n مینترم وجود دارد، می‌توان تعداد توابع ممکن که با n متغیر ایجاد می‌شود را 2^{2^n} دانست. گاهی بهتر است تابع بول را برحسب جمع مینترم‌ها بیان کنیم. اگر در این فرم نبود، می‌توان ابتدا آن را بصورت جمع جملات AND در آورد. آنگاه هر ترم برای یافتن همه متغیرها در آن وارسی می‌شود. اگر یک یا چند متغیر وجود نداشته باشند، می‌توان جمله را در عبارتی مثل $x + x'$، AND نمود، که x یکی از متغیرهای مفقود شده است. مثال زیر مطلب را روشن می‌کند.

مثال ۴-۲

تابع بولی $F = A + B'C$ را بصورت جمع مینترم‌ها در آورید. تابع سه متغیر A، B و C دارد. در اولین جمله A، دو متغیر مفقود است؛ بنابراین:

$$A = A(B + B') = AB + AB'$$

این تابع هنوز هم یک متغیر کسر دارد

$$\begin{aligned} A &= AB(C + C') + AB'(C + C') \\ &= ABC + ABC' + AB'C + AB'C' \end{aligned}$$

جمله دوم $B'C$ یک متغیر کم دارد

$$B'C = B'C(A + A') = AB'C + A'B'C$$

با ترکیب همه جملات داریم:

$$\begin{aligned} F &= A + B'C \\ &= ABC + ABC' + AB'C + AB'C' + A'B'C \end{aligned}$$

دیده می‌شود که $AB'C$ دوبار تکرار شده است و برحسب تئوری $(x + x = x)$ می‌توان یکی از آنها را حذف کرد. با مرتب نمودن مینترم‌ها به ترتیب صعودی داریم:

$$\begin{aligned} F &= A'B'C + AB'C + AB'C + ABC' + ABC \\ &= m_1 + m_4 + m_5 + m_6 + m_7 \end{aligned}$$

❑

گاهی بهتر است تابع بول را وقتی بصورت جمع مینترم‌هاست به فرم خلاصه زیر نشان دهیم:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

سمبل جمع Σ به معنی OR جملات است؛ اعدادی که به دنبال آن می‌آیند نیز مینترم‌های تابع هستند. حروف داخل پرانتز در جلو F، لیستی از متغیرهای تشکیل دهنده جملات مینترم را نشان می‌دهند. روش دیگری برای تشکیل مینترم‌های تابع بول تهیه مستقیم جدول درستی تابع از عبارت جبری و سپس خواندن مینترم‌ها از جدول درستی است. تابع بول مثال ۴-۲ را در نظر بگیرید:

$$F = A + B'C$$

جدول درستی در جدول ۵-۲ مستقیماً از عبارت جبری با لیست هشت ترکیب زیر متغیرهای A، B و C و اعمال 1 زیر ستون F برای ترکیباتی که در آن $A = 1$ و $BC = 01$ است فراهم شده است. سپس از جدول درستی می‌توان مشاهده کرد که مینترم‌های تابع، جملات 1، 4، 5، 6 و 7 می‌باشند.

ضرب ماکسترم‌ها

هر یک از 2^{2^n} تابع متشکل از n متغیر را می‌توان بصورت ضرب ماکسترم‌ها نیز بیان داشت. برای بیان توابع بول به عنوان ضرب ماکسترم‌ها، ابتدا باید جملات OR را تشکیل دهیم. این کار را می‌توان با

جدول ۵-۲. جدول درستی برای $F = A + B'C$

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

استفاده از قانون توزیع‌پذیری $x + yz = (x + y)(x + z)$ انجام داد. سپس هر متغیر مفقود در هر جمله OR با xx'، OR می‌شود. روش با مثال زیر روشن‌تر خواهد شد.

مثال ۵-۲

تابع بول $F = xy + x'z$ را بصورت ضرب جملات ماکسترم نشان دهید. ابتدا تابع را با استفاده از اصل توزیع‌پذیری به فرم جملات OR در آورید:

$$\begin{aligned} F &= xy + x'z = (xy + x')(xy + z) \\ &= (x + x')(y + x')(x + z)(y + z) \\ &= (x' + y)(x + z)(y + z) \end{aligned}$$

تابع سه متغیر دارد: x، y و z. هر جمله OR فاقد یک متغیر است؛ بنابراین

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x + y' + z) \\ y + z &= y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

با ترکیب همه جملات و حذف تکراری‌ها، خواهیم داشت:

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0M_2M_4M_5 \end{aligned}$$

نمایش ساده‌تر به شکل زیر است:

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

سمبل ضرب، Π، بیانگر AND ماکسترم‌هاست. اعداد داخل پرانتز شماره ماکسترم‌های تابع‌اند. ❑

تبدیل فرم‌های متعارف به یکدیگر

متمم یک تابع که بصورت مجموع مینترم‌ها نشان داده شده برابر است با مجموع مینترم‌هایی که در تابع اصلی وجود ندارند. دلیل این است که تابع اصلی با آن دسته از مینترم‌ها بیان شده است که تابع را 1

می‌کنند، در صورتی که متمم آن در ازاء مینترم‌هایی 1 می‌شود که تابع را 0 نموده‌اند. به عنوان مثال تابع زیر را در نظر بگیرید:

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

متمم این تابع به شکل زیر است:

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

اکنون اگر متمم F' را با روش تئوری دمورگان بدست آوریم، F را به فرم متفاوتی خواهیم داشت:

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

آخرین تبدیل در رابطه فوق از تعریف مینترم‌ها و ماکسترم‌ها در جدول (۳-۲) حاصل می‌شود. با توجه به جدول درستی رابطه زیر معتبر است:

$$m_j' = M_j$$

یعنی ماکسترم jام، متمم مینترم jام است و بالعکس.

آخرین مثال تبدیل یک تابع مینترمی به معادل ماکسترمی را نشان می‌دهد. بحث مشابهی نشان می‌دهد که تبدیل ضرب ماکسترم‌ها به جمع مینترم‌ها نیز به طریق فوق است. اکنون یک روال کلی را بیان می‌کنیم. برای تبدیل یک فرم متعارف به فرم متعارف دیگر، سمبل‌های Σ و Π را با هم عوض کنید و شماره‌های مفقود شده را از فرم اصلی تابع، لیست نمایید. برای یافتن جملات مفقود، باید بدانیم که تعداد کل جملات 2^n است، که در آن n تعداد متغیرهای دودویی در تابع می‌باشد.

یک تابع بولی می‌تواند از یک عبارت جبری به کمک جدول درستی و روال تبدیل متعارف به ضربی از ماکسترم‌ها تبدیل شود. به عنوان مثال عبارت بولی زیر را ملاحظه نمایید.

$$F = xy + x'z$$

ابتدا جدول درستی تابع را طبق جدول ۶-۲ بدست می‌آوریم. 1 های زیر ستون F از ترکیب $xy = 11$ یا $xz = 01$ بدست می‌آیند. مینترم‌های تابع در جدول درستی شماره‌های 1 ، 3 ، 6 و 7 می‌باشند. تابع برحسب مجموع مینترم‌ها چنین است:

$$F(x, y, z) = \Sigma(1, 3, 6, 7)$$

جدول ۶-۲ . جدول درستی برای $F = xy + x'z$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

چون جمعاً در یک تابع از سه متغیر هشت مینترم یا ماکسترم وجود دارد، جملات مفقود عبارتند از 0، 2، 4 و 5. تابعی که برحسب ضرب ماکسترم‌ها بیان شود برابر زیر است:

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

و این همان جوابی است که در مثال ۵-۲ بدست آمد.

فرم‌های استاندارد

دو نمایش متعارف جبر بول، فرم‌های اصلی تابع‌اند که هر کس می‌تواند با توجه به جدول درستی به آنها دسترسی پیدا کند. این فرم‌ها معمولاً دارای حداقل متغیرها نیستند زیرا هر مینترم یا ماکسترم باید بنا به تعریف دارای تمام متغیرها اعم از متمم یا غیرمتمم باشد.

راه دیگری برای بیان تابع بول، فرم استاندارد است. در این فرم، جمله‌هایی که تابع را تشکیل می‌دهند ممکن است یک، دو یا هر تعدادی از متغیرها را دارا باشند. دو نوع فرم استاندارد وجود دارد. یکی جمع حاصلضرب‌ها و دیگری ضرب حاصل جمع‌ها.

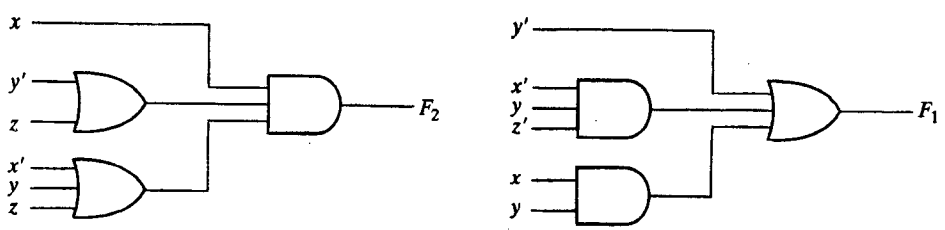
جمع حاصلضرب‌ها، یک عبارت بولی است شامل جملات AND که آنها را جملات ضرب می‌گوییم و هر یک دارای یک یا چند لیترال است. علامت جمع به معنی OR این جملات است. مثالی از یک تابع بصورت جمع حاصلضرب‌ها را در زیر ملاحظه نمایید.

$$F_1 = y' + xy + x'yz'$$

این عبارت سه جمله، با یک، دو و سه لیترال دارد. جمع آنها اثر OR را داراست.

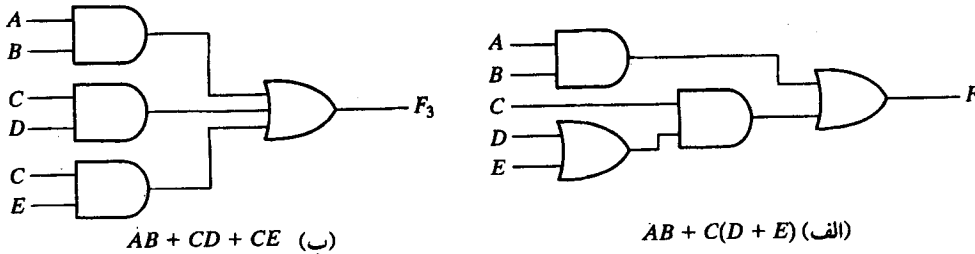
نمودار منطقی جمع حاصلضرب‌ها متشکل از گروهی گیت AND است که بدنبال آن یک گیت OR می‌آید. الگوی این آرایش در شکل ۳-۲ (الف) آمده است. هر جمله ضرب نیاز به یک گیت AND دارد. این نکته در یک ورودی تک لیترالی مستثنی است. جمع منطقی با یک گیت OR صورت می‌گیرد که ورودی‌هایش خروجی گیت‌های AND و نیز تک ورودی مذکور است. ضمناً فرض بر این است که متمم متغیرهای ورودی مستقیماً موجودند بنابراین وارونگر در نمودار لحاظ نشده است. این آرایش را پیاده‌سازی دو سطحی یا دو طبقه می‌گویند.

ضرب حاصل جمع‌ها، یک عبارت بولی حاوی جملات OR است که به آن جملات جمع می‌گویند.



(ب) ضرب حاصل جمع‌ها (الف) جمع حاصلضرب‌ها

شکل ۳-۲. پیاده‌سازی دو سطحی



شکل ۴-۲ . پیاده‌سازی سه و دو سطحی

هر جمله می‌تواند به هر تعداد لیترال داشته باشد. ضرب به معنی AND این جملات است. مثالی از یک تابع بصورت ضرب حاصل جمع‌ها چنین است:

$$F_2 = x(y' + z)(x' + y + z')$$

این تابع دارای سه جمله جمع، با یک، دو و سه لیترال است. ضرب نیز یک عملگر AND می‌باشد. استفاده از لغات ضرب و جمع از شباهت عمل AND با ضرب حسابی و شباهت عمل OR با جمع حسابی مشتق شده است. ساختار گیتی ضرب حاصل جمع متشکل از گروهی گیت OR برای جملات جمع (به جز برای تک لیترال) و به دنبال آن یک گیت AND می‌باشد. این نکته در شکل ۳-۲ (ب) دیده می‌شود، این نوع استاندارد عبارت به یک ساختار دو سطحی (یا دو طبقه) از گیت‌ها منجر می‌گردد.

یک تابع بول ممکن است بصورت غیراستاندارد نیز بیان شود. مثلاً تابع

$$F_3 = AB + C(D + E)$$

نه جمع حاصلضرب و نه ضرب حاصل جمع است. پیاده‌سازی این عبارت در شکل ۴-۲ (الف) دیده می‌شود. این مدار به دو گیت AND و دو گیت OR نیاز دارد. در این مدار سه سطح گیت وجود دارد. می‌توان در آن با استفاده از اصل توزیع‌پذیری پرانتز را حذف و آن را به فرم استاندارد در آورد.

$$F_3 = AB + C(D + E) = AB + CD + CE$$

عبارت جمع حاصلضرب در شکل ۴-۲ (ب) پیاده شده است. بطور کلی، یک پیاده‌سازی دو سطحی ترجیح داده می‌شود زیرا به هنگام انتشار ورودی‌ها به سمت خروجی‌ها حداقل مقدار تأخیر را در گیت‌ها تولید می‌کند.

۶-۲ دیگر اعمال منطقی

وقتی که عملگرهای AND و OR بین دو متغیر x و y قرار می‌گیرند، به ترتیب دو تابع بولی $x.y$ و $x+y$ را تشکیل می‌دهند. قبلاً بیان شد که برای n متغیر 2^{2^n} تابع دودویی وجود دارد. برای دو متغیر، $n=2$ و تعداد توابع بولی ممکن 16 است. بنابراین توابع AND و OR تنها دو تابع از 16 تابع ممکن با دو متغیر دودویی هستند. پیشنهاد می‌گردد 14 تابع باقیمانده مشخص شده و راجع به خصوصیت آنها تحقیق گردد.

جدول درستی 16 تابع که با دو متغیر دودویی x و y تشکیل گردیده در جدول ۷-۲ لیست شده

جدول ۷-۲ . جدول درستی برای 16 تابع از دو متغیر دودویی

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

است. هر یک از 16 ستون F_0 و F_{15} ، جدول درستی یک تابع ممکن برای دو متغیر x و y را نشان می‌دهد. توجه داشته باشید که توابع از 16 ترکیب ممکن تخصیص یافته به F معین می‌گردند. 16 تابع را می‌توان با توابع بول نشان داد. این توابع در ستون اول جدول ۸-۲ دیده می‌شوند. عبارات بولی از نظر تعداد لیترال حداقل شده‌اند.

گرچه هر تابع را می‌توان برحسب عملگرهای دودویی AND ، OR و NOT بیان کرد، اما دلیلی وجود ندارد که کسی عملگر خاصی را برای بیان سایر توابع تعیین ننماید. چنین عملگرهایی در ستون دوم جدول ۸-۲ لیست شده‌اند. با این وجود همه سمبل‌های جدید که در جدول نشان داده شده‌اند، بجز OR انحصاری (⊕) کاربرد چندانی به وسیله طراحان ندارند.

به دنبال هر یک از توابع در جدول ۸-۲، نام و توضیحی که تابع را به نحوی تشریح می‌کند، آورده شده است. 16 تابع لیست شده فوق به سه گروه زیر تقسیم می‌شوند.

۱- دو تابع که ثابت‌های 0 و 1 را تولید می‌کنند (Identity , Null)

۲- چهار تابع یکانی از نوع متمم و انتقال (Transfer , Complement)

جدول ۸-۲ . عبارات بولی 16 تابع حاصل از دو متغیر

توابع بول	سمبل عملگر	نام	توضیحات
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

۳- ده تابع باقیمانده شامل هشت عمل مختلف AND ، OR ، NAND ، NOR ، XOR (یا OR انحصاری)، XNOR (یا هم ارزی) Inhibition (نهی) و Implication (استلزام) می‌باشند.

ثابت‌ها برای توابع دودویی فقط می‌توانند 0 یا 1 باشند. تابع متمم، متمم هر متغیر دودویی را تولید می‌نماید. تابعی که برابر یک متغیر ورودی است را Transfer یا Buffer (انتقال) می‌نامند، زیرا متغیر x یا y از طریق یک گیت بدون تغییر مقدار عبور کرده است. از هشت عملگر دودویی، دوتای آنها (نهی و استلزام) به وسیله طراحان مدارات منطقی بکار می‌روند، ولی به ندرت در منطق کامپیوتر از آنها استفاده می‌شود. عملگرهای AND و OR قبلاً در جبر بول ذکر شدند. چهار تابع دیگر بطور گسترده در طراحی دستگاه‌های دیجیتال مورد استفاده‌اند.

تابع NOR متمم OR بوده و نام آن از not − OR اخذ شده است. بطور مشابه NAND ، متمم AND است و از not − AND مشتق می‌شود. OR انحصاری یا XOR مشابه با OR است ولی حالتی که در آن هر دو متغیر x و y متفقاً برابر 1 باشند، را شامل نمی‌شود. تابع XNOR یا هم‌ارزی تابعی است که هنگام مساوی بودن دو متغیر برابر 1 می‌شود، یعنی وقتی هر دو 0 یا هر دو 1 باشند. توابع XOR و XNOR متمم یکدیگرند و این خاصیت بسادگی با ملاحظه جدول ۷-۲ قابل تشخیص است. جدول درستی برای OR عبارت است از F_6 و برای XNOR نیز F_9 است. این دو تابع متمم یکدیگرند. به این دلیل تابع هم‌ارزی را NOR انحصاری هم می‌گویند و با XNOR نشان می‌دهند.

جبر بول طبق تعریف بخش ۲-۲ دو عملگر دودویی با نام‌های AND و OR و یک عملگر یکانی با نام NOT (متمم) دارد. ما با توجه به تعاریف، برخی از خواص آنها را استنتاج نمودیم و در این بخش تعدادی از عملگرهای دودویی دیگر را برحسب آنها معرفی کردیم. این روال منحصر به فرد نیست. به عنوان مثال می‌توانستیم ابتدا NOR(↓) را تعریف کرده و سپس AND و OR و NOT را برحسب آنها تعریف کنیم. به هر حال دلایل مکفی برای روش منتخب وجود دارد و در واقع مفاهیم AND ، OR و NOT بیشتر بین مردم مصطلح بوده و بر تفکرات حاکم هستند. علاوه بر آن اصول هانتینگتون منعکس کننده طبیعت دوگانی این جبر است و این خود بر خاصیت تقارن (+) و (.) نسبت به یکدیگر دلالت دارد.

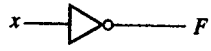
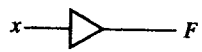
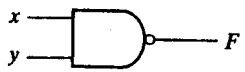
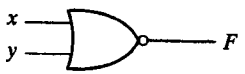
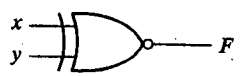
۷-۲ گیت‌های منطقی دیجیتال

چون توابع بول برحسب عملگرهای AND ، OR و NOT بیان شده‌اند، پیاده کردن آنها با استفاده از اینگونه گیت‌ها ساده‌تر خواهد بود. امکان ساخت گیت‌ها برای دیگر اعمال منطقی در عمل مورد توجه است. فاکتورهایی که باید به هنگام ساخت آنها در نظر گرفته شوند عبارتند از:

(۱) امکان سنجی و اقتصادی بودن روش ساخت به هنگام استفاده از قطعات فیزیکی. (۲) امکان گسترش ورودی گیت‌ها به بیش از دو. (۳) در نظر گرفتن خواص اصلی عملگرهای دودویی مثل جابجایی و شرکت‌پذیری. (۴) توانایی گیت در پیاده‌سازی توابع به تنهایی یا همراه با سایر گیت‌ها از شانزده تابع معرفی شده. در جدول (۸-۲) دو تابع برابر با مقدار ثابت و چهارتای دیگر دوبار تکرار شده‌اند. بنابراین تنها ده تابع برای تهیه گیت‌های منطقی کاندید هستند. دو تابع نهی و استلزام دارای

خاصیت جابجایی یا شرکت‌پذیری نیستند و لذا به عنوان گیت‌های منطقی استاندارد مورد استفاده نمی‌باشند. هشت تابع دیگر یعنی: Buffer ، NOT ، AND ، OR ، NAND ، NOR ، XOR و XNOR به عنوان گیت‌های استاندارد در طراحی سیستم‌های دیجیتال بکار می‌روند.

سمبل‌های گرافیکی و جدول درستی هشت گیت فوق در شکل ۵-۲ نشان داده شده‌اند. هر گیت

نام	سمبل گرافیکی	تابع جبری	جدول درستی
AND		$F = xy$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$
OR		$F = x + y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 1$
Inverter		$F = x'$	$x \mid F$ $0 \mid 1$ $1 \mid 0$
Buffer		$F = x$	$x \mid F$ $0 \mid 0$ $1 \mid 1$
NAND		$F = (xy)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
NOR		$F = (x + y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 0$
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$ $0\ 0 \mid 0$ $0\ 1 \mid 1$ $1\ 0 \mid 1$ $1\ 1 \mid 0$
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$ $0\ 0 \mid 1$ $0\ 1 \mid 0$ $1\ 0 \mid 0$ $1\ 1 \mid 1$

شکل ۵-۲ . گیت‌های منطقی دیجیتال نام، سمبل گرافیکی، تابع جبری، جدول درستی

دارای دو متغیر ورودی دودویی x و y و یک متغیر خروجی دودویی F می‌باشد. مدارهای AND، OR و NOT در شکل ۶-۱ معرفی شده بودند. مدار NOT یا وارونگر وضعیت منطقی یک متغیر دودویی را معکوس می‌نماید و در واقع متمم متغیر را تولید می‌کند. دایره کوچک در خروجی سمبل گرافیکی یک وارونگر (که به آن حباب می‌گویند) بیانگر متمم شدن است. سمبل مثبت بـه تـنهایی عـلامت بـافر می‌باشد. یک بافر عمل انتقال را انجام می‌دهد، ولی یک عمل منطقی تولید نمی‌کند زیرا مقدار دودویی خروجی برابر مقدا رورودی دودویی است. این مدار صرفاً در تقویت توان سیگنال‌ها استفاده شـده و معادل با دو مدار متوالی وارونگر (معکوس‌گر) است.

تابع NAND متمم AND است و همانطور که از سمبل گرافیکی آن مشخص است از یک سمبل AND و یک حباب تشکیل شده است. تابع NOR هم متمم OR است و با یک سمبل OR و به دنبال آن یک حباب نمایش داده می‌شود. گیت‌های NAND و NOR بطور گسترده‌ای به عـنوان گـیت‌های استاندارد مورد استفاده قرار گرفته و بیشتر از OR و AND مورد توجه‌اند. ایـن بـدان عـلت است کـه گیت‌های NAND و NOR به سادگی به وسیله مدارات ترانزیستوری قابل تولید بوده و می‌توان بـه راحتی توابع بول را با آنها پیاده‌سازی کرد.

گیت XOR دارای سمبل مشابهی با OR است، بجز اینکه یک خط منحنی در سمت ورودی‌اش کشیده شده است. گیت XNOR متمم XOR است و لذا حباب کوچکی در خروجی آن وجود دارد.

گسترش ورودی گیت‌ها

گیت‌هایی که در شکل ۵-۲ نشان داده شدند، بجز برای وارونگر و انتقال، قابل گسترش به بیش از دو ورودی می‌باشند. اگر عمل دودویی یک گیت جابجا و شرکت‌پذیر باشد. می‌توان ورودی‌هـای آن را گسترش داد. اعمال AND و OR که در جبر بول تعریف شده‌اند این خاصیت را از خود بـه نـمایش گذاشته‌اند. برای تابع OR داریم:

$$x + y = y + x \qquad \text{(جابجایی)}$$

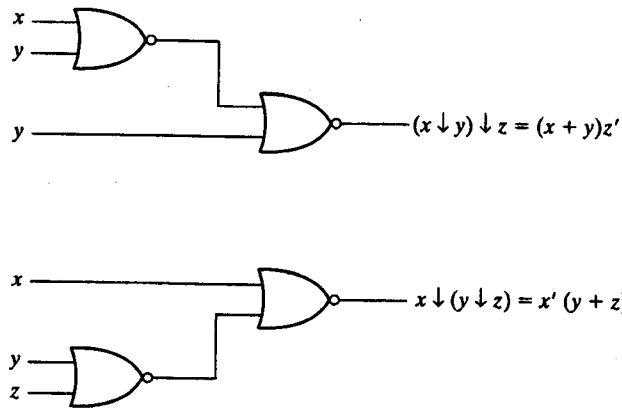
و

$$(x + y) + z = x + (y + z) = x + y + z \qquad \text{(شرکت‌پذیر)}$$

این روابط بیانگر تعویض‌پذیری ورودی‌های گیت و قابل گسترش بودن متغیرهای ورودی به بیش از دو در تابع OR است.

توابع NAND و NOR جابجاپذیرند و ورودی آنها می‌تواند به بیش از دو افزایش یابد، مشروط بر این که در تعریف تابع مختصر تغییری صورت گیرد. مشکل این است که NAND و NOR شرکت‌پذیر نیستند [یعنی $(x \downarrow y) \downarrow z \neq x(y \downarrow z)$]. این نکته در شکل ۶-۲ و معادلات زیر بنمایش گذاشته شده‌اند.

$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$
$$x \downarrow (y \downarrow z) = [x + (y + z)']' = x'(y + z) = x'y + x'z$$



شکل ۶-۲ . نمایش شرکت‌ناپذیری عملگر NOR ($x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z$)

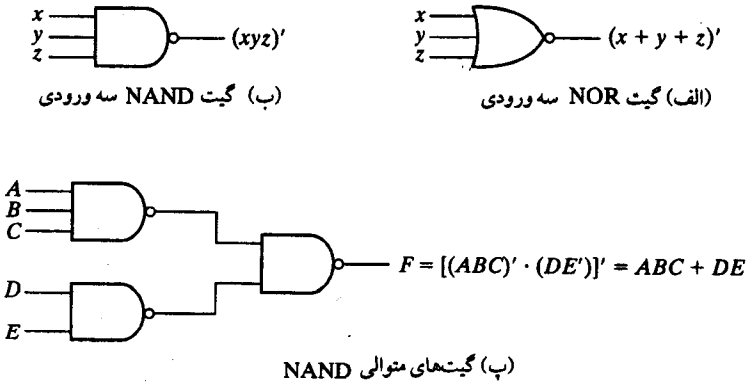
برای غلبه بر این مشکل، گیت NOR (یا NAND) چند ورودی را به عنوان متمم OR (یا AND) آن تعریف می‌کنیم. بنابراین:

$$x \downarrow y \downarrow z = (x + y + z)'$$
$$x \uparrow y \uparrow z = (xyz)'$$

سمبل‌های گرافیکی گیت‌های سه ورودی در شکل ۷-۲ نشان داده شده‌اند. در نوشتن متوالی اعمال NOR و NAND باید پرانتزها به فرم صحیحی انتخاب شوند تا بیانگر ترتیب صحیح گیت‌ها باشند. برای درک این مطلب مدار شکل ۷-۲ (پ) را ملاحظه نمایید. برای این مدار تابع بول باید به فرم زیر نوشته شود:

$$F = [(ABC)'(DE)']' = ABC + DE$$

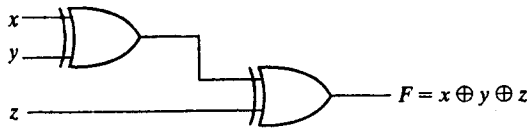
دومین عبارت از تئوری دمورگان نتیجه می‌شود. این رابطه همچنین بیان می‌دارد که جمع حاصلضرب‌ها قابل پیاده شدن با گیت‌های NAND می‌باشد. بحث بیشتر در مورد گیت‌های NAND و NOR در بخش ۶-۳ آورده شده است.



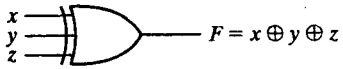
شکل ۷-۲ . گیت‌های NAND و NOR متوالی با چند ورودی

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(پ) جدول درستی



(الف) با گیت‌های دو ورودی



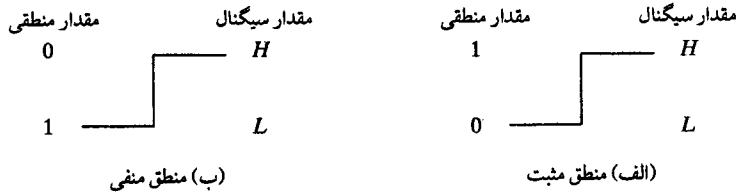
(ب) یک گیت سه ورودی

شکل ۸-۲. گیت XOR سه ورودی

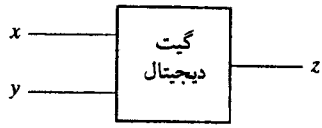
XOR و XNOR هر دو خواص جابجایی و شرکت‌پذیری را دارند و ورودی‌هایشان قابل توسعه به بیش از دو می‌باشد. با این وجود گیت‌های XOR چند ورودی از نقطه نظر سخت‌افزاری متداول نیستند. در واقع حتی فرم دو ورودی آن نیز معمولاً از سایر گیت‌ها ساخته می‌شود. علاوه بر این، تعریف این توابع باید به هنگام گسترش ورودی‌ها تصحیح گردد. تابع XOR یک تابع فرد است یعنی هرگاه ورودی‌ها تعداد فردی 1 داشته باشند، این تابع (خروجی) برابر 1 خواهد بود. ساختمان یک گیت XOR با سه ورودی در شکل ۸-۲ دیده می‌شود. این مدار معمولاً با گیت‌های دو ورودی تهیه می‌شود، شکل (الف). بصورت گرافیکی، آن را می‌توان با یک گیت سه ورودی نشان داد، شکل (ب). جدول درستی (پ) آشکارا مشخص می‌نماید که خروجی F برابر 1 است به شرطی که فقط یکی از ورودی‌ها و یا هر سه ورودی برابر 1، باشند؛ یعنی وقتی تعداد کل 1ها در متغیرهای ورودی فرد است، تابع 1 است. بحث بیشتری در مورد XOR به بخش ۸-۳ موکول شده است.

منطق مثبت و منفی

سیگنال دودویی در ورودی‌ها یا خروجی هر گیت، بجز در حالت گذرا، یکی از دو مقدار را دارد. یک مقدار سیگنال، منطق 1 و دیگری منطق 0 را نمایش می‌دهد. چون دو مقدار سیگنال متعلق به دو ارزش منطقی است، لذا دو انتساب متفاوت برای دو ارزش منطقی می‌توان اختیار کرد، شکل ۹-۲. سطح سیگنال بالاتر با H و سطح سیگنال پایین‌تر با L مشخص شده است. اگر سطح بالا، H، برای منطق 1 بکار رود یک سیستم منطق مثبت تعریف شده است. انتخاب L برای منطق 1 سیستم منطقی منفی را معرفی می‌نماید. کلمات مثبت یا منفی گاهی گمراه کننده هستند زیرا هر دو سطح سیگنال ممکن است



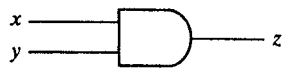
شکل ۹-۲. تخصیص سیگنال و قطبیت منطق



(ب) نمودار بلوکی گیت

x	y	F
L	L	L
L	H	L
H	L	L
H	H	H

(الف) جدول درستی با
L و H



(ت) کمیت AND با منطق مثبت

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

(پ) جدول درستی برای منطق مثبت

(ج) گیت OR با منطق منفی

x	y	z
1	1	1
1	0	1
0	1	1
0	0	0

(ث) جدول درستی برای منطقی منفی

شکل ۱۰-۲. نمایش منطق مثبت و منفی

مثبت یا هر دو منفی باشند. در واقع، این قطب‌های سیگنال نیستند که بیانگر نوع منطق می‌باشند، بلکه انتخاب مقادیر منطقی برحسب سطوح نسبی سیگنال‌ها نسبت به هم، نوع منطق را مشخص می‌کنند.

گیت‌های دیجیتال سخت‌افزاری برحسب مقادیر سیگنال H و L تعریف می‌شوند. از این پس انتخاب منطق مثبت و منفی به عهده کاربر است. به عنوان مثال گیت الکترونیک شکل ۱۰-۲ (ب) را در نظر بگیرید. جدول درستی برای این گیت در شکل ۱۰-۲ (الف) لیست شده است. این جدول رفتار فیزیکی گیت را وقتی H برابر 3V و L برابر 0 ولت است نشان می‌دهد. جدول درستی شکل ۱۰-۲ (پ) منطق مثبت را فرض می‌کند که در آن H = 1 و L = 0 است. این جدول درستی همانند جدول عمل AND است. نمودار گرافیکی برای گیت AND با منطق مثبت درشکل ۱۰-۲ (ت) دیده می‌شود.

اکنون تخصیص منطق منفی را برای همان گیت فیزیکی با L = 1 و H = 0 در نظر بگیرید. نتیجه جدول درستی شکل ۱۰-۲ (ث) خواهد بود. گرچه داده‌ها معکوس شده‌اند ولی جدول عمل OR را نشان می‌دهد. سمبل گرافیکی گیت OR با منطق منفی در شکل ۱۰-۲ (ج) دیده می‌شود. مثلث‌های کوچک در ورودی‌ها و خروجی نشانگر قطبیت هستند. وجود این علائم قطبیت همراه با مشخصات پایانه بیانگر فرض منطق منفی برای سیگنال است. بنابراین گیت فیزیکی فوق می‌تواند یک گیت AND با منطق مثبت و نیز یک گیت OR با منطق منفی باشد.

تبدیل منطق مثبت به منفی و بالعکس، عملی است که طی آن در ورودی و خروجی یک گیت 0ها به 1 و 1ها به 0 تبدیل می‌شوند. چون این عمل دوگان تابع را تولید می‌کند، تعویض پایانه‌ها از یک قطبیت به قطبیت دیگر نتیجه‌اش همان یافتن دوگان تابع است. نتیجه این تبدیل این است که همه عملگرهای AND به OR و بالعکس تبدیل شوند. به علاوه نباید از ذکر مثلث در سمبل‌های گرافیکی که بیانگر قطبیت است در منطق منفی، فراموش کرد. در این کتاب از گیت‌ها با منطق منفی استفاده نمی‌کنیم و فرض خواهیم که همه گیت‌ها با منطق مثبت کار کنند.

۸-۲ مدارهای مجتمع

یک مدار مجتمع (IC) یک کریستال نیمه هادی از جنس سیلیکان است که به آن تراشه می‌گویند و حاوی اجزاء الکترونیکی در ساخت گیت‌های دیجیتال می‌باشد. انواع گیت‌ها در داخل تراشه به هم وصل می‌شوند تا مدار مورد نیاز ایجاد گردد. تراشه روی یک محفظه سرامیک یا پلاستیک نصب شده و اتصالات به پایه‌های بیرون برای ایجاد مدار مجتمع، متصل می‌گردند. تعداد پایه‌ها ممکن است از 14 در یک بسته کوچک IC تا چند هزار در یک بسته بزرگ تغییر کند. در روی هر IC یک شماره برای شناسایی چاپ می‌شود. سازندگان معمولاً کتابچه‌ها، کاتالوگ‌ها و سایت‌های اینترنتی حاوی توصیف‌ها و اطلاعات IC هایی که ساخته‌اند را در اختیار می‌گذارند.

سطوح مجتمع‌سازی

IC های دیجیتال اغلب براساس پیچیدگی مدار درونی‌شان که به تعداد گیت‌های منطقی مرتبط است دسته‌بندی می‌شوند. تفکیک تراشه‌هایی که تنها چند یا چند صد و یا چندین هزار گیت دارند با ارجاع به بسته و دسته‌بندی آنها به وسایل مجتمع با فشردگی کم، متوسط، زیاد و خیلی زیاد صورت می‌گیرد.

مدارهای مجتمع با فشردگی کم *(SSI)* دارای چند گیت مستقل در بسته‌اند. ورودی‌ها و خروجی‌های گیت‌ها مستقیماً به پایه‌های بسته متصل می‌شوند. تعداد گیت‌ها معمولاً کمتر از 10 بوده و به وسیله پایه‌های موجود در IC محدود می‌گردند.

مدارهای مجتمع با فشردگی متوسط *(MSI)* دارای فشردگی بین 10 تا 1000 گیت در یک بسته دارند. این دسته از مدارات معمولاً اعمال دیجیتال خاصی را اجرا می‌کنند. توابع دیجیتال MSI با عناوین دیکدرها، جمع کننده‌ها و مولتی پلکسرها در فصل ۴ و ثبات‌ها و شمارنده‌ها در فصل ۶ مطرح شده‌اند.

مدارهای مجتمع با فشردگی زیاد *(LSI)* حاوی هزاران گیت در یک بسته می‌باشند. این دسته از مدارات، پردازنده‌ها، حافظه‌ها و مدارات منطقی برنامه‌پذیر را شامل می‌شوند. بعضی از اجزاء LSI در فصل ۸ معرفی شده‌اند.

مدارهای مجتمع با فشردگی خیلی زیاد *(VLSI)* صدها هزار گیت در یک بسته دارند. از جمله مثال‌ها می‌توان از آرایه‌های حافظه، تراشه میکرو کامپیوترهای پیچیده نام برد. به دلیل کوچکی و ارزانی، وسایل VLSI تکنولوژی طراحی سیستم کامپیوتر را متحول ساخته و به طراح قابلیت ساخت وسایلی را می‌دهد که قبلاً اقتصادی نبودند.

خانواده‌های منطقی دیجیتال

مدارهای مجتمع دیجیتال نه تنها براساس پیچیدگی و عمل منطقی‌شان بلکه براساس تکنولوژی مدار خاصی که به آن تعلق دارند نیز دسته‌بندی می‌گردند. تکنولوژی مدار به نام خانواده مدار منطقی خوانده می‌شود. هر خانواده منطقی دارای مدار الکترونیک مبنای خاص خود بوده و سایر توابع و مدارات پیچیده دیجیتال با استفاده از آنها ساخته می‌شوند. مدار مبنا در هر خانواده، گیت NAND، NOR یا NOT است. قطعات الکترونیک بکار رفته در ساخت مدار مبنا معمولاً برای نام‌گذاری تکنولوژی مورد استفاده قرار می‌گیرد. به لحاظ تجاری انواع متفاوتی از خانواده‌های منطقی مدارات مجتمع معرفی شده‌اند. انواع رایج آنها در زیر لیست شده‌اند.

transistor-transistor logic	TTL
emitter-coupled logic	ECL
metal-oxide semiconductor	MOS
complementary metal-oxide semiconductor	CMOS

TTL مدت مدیدی است که مورد استفاده بوده و به عنوان یک گیت استاندارد شناخته شده است. ECL در سیستم‌هایی که به سرعت بالا نیاز دارد ارجحیت دارد. MOS در مدارهایی که نیاز به چگالی قطعه بالایی دارند مورد استفاده است و CMOS در مواقعی که توان مصرفی باید کم باشد مورد توجه می‌باشد. نظر به این که توان مصرفی کم در طراحی VLSI از اصول است، CMOS تبدیل به یک خانواده منطقی غالب شده است در حالی که از کاربرد خانواده‌های TTL و ECL به تدریج کاسته می‌شود. تحلیل مدار گیت دیجیتال مبنای الکترونیکی در هر خانواده منطقی در فصل ۱۰ ارائه شده است.

مشخصه‌های خانواده‌های منطقی معمولاً با تحلیل مدار گیت مبنا در هر خانواده مقایسه می‌شوند. مهمترین پارامترهای مورد ارزیابی و مقایسه در بخش ۲-۱۰ بحث شده‌اند. در اینجا نیز به خاطر ارجاع مختصراً تعریف می‌گردند.

گنجایش خروجی نشان دهنده تعداد بارهای استانداردی است که خروجی یک نمونه گیت بدون تخریب عملکردش، بتواند راه بیندازد. یک بار استاندارد عبارت است از مقدار جریانی که برای ورودی گیت مشابه دیگر از همان خانواده لازم می‌باشد.

گنجایش ورودی تعداد ورودی‌های موجود در یک گیت است.

توان مصرفی توان تلف شده‌ای است که باید به وسیله منبع تغذیه برای گیت فراهم شود.

تأخیر انتشار عبارت است از متوسط زمان تأخیر در انتقال سیگنال از ورودی به خروجی.

حد پارازیت حداکثر ولتاژ بیرونی است که به ورودی سیگنال اضافه می‌شود ولی موجب تغییر ناخواسته در خروجی مدار نگردد.

طراحی با کمک کامپیوتر (CAD)

طراحی سیستم‌های دیجیتال با مدارهای دیجیتال حاوی میلیون‌ها ترانزیستور کار دشواری است.

سیستم‌هایی با این پیچیدگی را نمی‌توان بدون کمک ابزارهای طراحی کامپیوتری (CAD) ، ایجاد و از صحت عمل آنها مطمئن شد. ابزارهای CAD شامل برنامه‌های نرم‌افزاری است که نمایش‌های کامپیوتری را پشتیبانی کرده و در ایجاد سخت‌افزار دیجیتال به خودکارسازی کمک می‌کنند. خودکارسازی طراحی الکترونیک همه فازهای طراحی مدارهای مجتمع را شامل می‌شود. روالی از یک نمونه طراحی برای ساخت مدارهای VLSI شامل رشته‌ای از مراحل است که از وارده‌های طراحی آغاز و با تولید یک بانک اطلاعاتی حاوی ماسک‌های ساخت IC پایان می‌پذیرد. در ایجاد فیزیکی یک مدار دیجیتال در یک سیلیکان روش‌های مختلفی وجود دارد. طرح می‌تواند، یک مدار مجتمع با کاربرد خاص (ASIC) ، یک آرایه گیتی برنامه‌پذیر موردی (FPGA) ، یک وسیله منطقی برنامه‌پذیر (PLD) یا یک IC سفارشی باشد. با هر یک از این وسایل یک مجموعه ابزار CAD همراه است و نرم‌افزار لازم برای خلق سخت‌افزار را فراهم می‌سازد.

بعضی از سیستم‌های CAD حاوی یک برنامه ادیتور یا ویرایشگر است که نمودار تصویری را روی صفحه نمایش کامپیوتر خلق و اصلاح می‌کند. این فرآیند را ضبط تصویری یا وارده تصویری نامند. با کمک منوها، فرامین صفحه کلید و ماوس یک ادیتور تصویر می‌تواند نمودارهای مدارهای دیجیتال را روی صفحه کامپیوتر ترسیم کند. قطعات از لیستی در کتابخانه درونی، قابل استقرار در روی صفحه کامپیوتر بوده و سپس با خطوطی که سیم‌ها را نمایش می‌دهند به هم وصل می‌شوند. نرم‌افزار وارده تصویر یک بانک اطلاعاتی حاوی اطلاعات تولید شده با شماتیک را خلق و مدیریت می‌کند. گیت‌های ساده و بلوک‌های عملیاتی دارای مدل‌هایی هستند که رفتار و زمانبندی مدار مورد تست را ممکن می‌سازند. این کار با اعمال ورودی‌ها به مدار و استفاده از شبیه‌ساز منطقی برای تعیین خروجی‌ها صورت می‌گیرد.

یک ابزار ساخت مهم در طراحی سیستم‌های دیجیتال استفاده از زبان طراحی سخت‌افزار (HDL) است. HDL یک زبان برنامه‌نویسی است ولی خصوصاً برای توصیف سخت‌افزار دیجیتال آماده شده است. این زبان نمودارهای منطقی و دیگر اطلاعات دیجیتال را به فرم متن نشان می‌دهد. از آن برای شبیه‌سازی و تست و تصدیق عملیاتی قبل از ساخت استفاده می‌شود. کاربرد مهمی از آن نرم‌افزار طراحی منطقی آن است، که طراحی سیستم‌های دیجیتال را خودکار می‌سازد. HDL در سال‌های اخیر اهمیت به سزایی یافته است و بهترین روش موجود در طراحی سیستم‌های دیجیتال پیشرفته است. HDL در بخش ۹-۳ معرفی شده است و به دلیل اهمیت آن، ما توصیف HDL مدارهای دیجیتال، قطعات و روال طراحی را در سرتاسر این کتاب لحاظ کرده‌ایم.

۲-۱ به کمک جداول درستی اعتبار موارد زیر را تحقیق کنید:

(الف) تئوری دمورگان برای سه متغیر: $(xyz)' = x' + y' + z'$ و $(x + y + z)' = x'y'z'$

(ب) اصل توزیع‌پذیری: $x + yz = (x + y)(x + z)$

۲-۲ عبارات بولی زیر را با تعداد حداقل لیترال ساده کنید:

(الف) $xy + xy'$ (ب) $(x + y)(x + y')$

(پ) $xyz + x'y + xyz'$ (ت) $(A + B)'(A' + B')'$

۲-۳ عبارات بولی زیر را با تعداد حداقل لیترال ساده نمایید:

(الف) $ABC + A'B + ABC'$ (ب) $x'yz + xz$

(پ) $(x + y)'(x' + y')$ (ت) $xy + x(wz + wz')$

(ث) $(BC' + A'D)(AB' + CD')$

۲-۴ عبارات بولی زیر را به تعداد لیترال‌های ذکر شده ساده کنید.

(الف) $A'C' + ABC + AC'$ سه لیترال

(ب) $(x'y' + z)' + z + xy + wz$ سه لیترال

(پ) $A'B(D' + C'D) + B(A + A'CD)$ یک لیترال

(ت) $(A' + C)(A' + C')(A + B + C'D)$ چهار لیترال

۲-۵ متمم $F = x + yz$ را پیدا کنید و سپس نشان دهید که $FF' = 0$ و $F + F' = 1$ است.

۲-۶ متمم عبارات زیر را پیدا کنید:

(الف) $xy' + x'y$ (ب) $(AB' + C)D' + E$

(پ) $(x + y' + z)(x' + z')(x + y)$

۲-۷ با فرض توابع F_1 و F_2:

(الف) نشان دهید که تابع بولی $E = F_1 + F_2$ حاوی مجموع مینترم‌های F_1 و F_2 است.

(ب) نشان دهید که تابع بولی $G = F_1F_2$ فقط حاوی مینترم‌های مشترک F_1 و F_2 است.

۲-۸ جدول درستی تابع زیر را لیست کنید:

$$F = xy + xy' + y'z$$

۲-۹ اعمال منطقی اجرا شده بر روی رشته‌ای از بیت‌ها با ملاحظه هر جفت بیت انجام می‌شود (به آن عمل بیتی می‌گویند) با فرض داشتن دو رشته 8 بیتی $A = 10101101$ و $B = 10001110$، نتایج بیتی را پس از انجام اعمال منطقی زیر ارزیابی کنید: (الف) AND، (ب) OR، (پ) XOR، (ت) NOT A، (ث) NOT B.

۲-۱۰ نمودارهای منطقی را برای عبارات بولی زیر رسم نمایید:

(الف) $Y = A'B' + B(A + C)$ (ب) $Y = BC + AC'$

(پ) $Y = A + CD$ (ت) $Y = (A + B)(C' + D)$

۲-۱۱ با فرض تابع بولی زیر:

$$F = xy + x'y' + y'z$$

(الف) آن را با گیت‌های AND ، OR و وارونگر (NOT) پیاده‌سازی کنید.
(ب) آن را با گیت‌های OR و NOT پیاده کنید.
(پ) آن را با گیت AND و NOT پیاده نمایید.

۲-۱۲ تابع بول T_1 و T_2 را به حداقل لیترالهای ساده نمایید.

A	B	C	T_1	T_2
0	0	0	1	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

۲-۱۳ جمع منطقی همه مینترم‌های تابع بول n متغیره 1 است.
(الف) جمله بالا را برای $n = 3$ ثابت کنید.
(ب) روالی عمومی، برای اثبات پیشنهاد نمایید.

۲-۱۴ جدول درستی توابع زیر را بدست آورده و هر تابع را بصورت جمع مینترم‌ها و ضرب ماکسترم‌ها بیان کنید:
(الف) $(xy + z)(y + xz)$ (ب) $(A' + B)(B' + C)$
(پ) $y'z + wxy' + wxz' + w'x'z$

۲-۱۵ با فرض تابع بولی زیر

$$F = xy'z + x'y'z + w'xy + wx'y + wxy$$

(الف) جدول درستی آن را پیدا کنید.
(ب) نمودار منطقی را با استفاده از عبارت بولی اولیه بدست آورید.
(پ) تابع را با استفاده از جبر بول به حداقل لیترال‌ها ساده کنید.
(ت) جدول درستی تابع را از عبارت ساده شده بدست آورید و نشان دهید که مشابه بخش (الف) است.
(ث) نمودار منطقی را از عبارت ساده شده رسم کنید و تعداد گیت‌های کل آن را با نمودار (ب) مقایسه نمایید.

۲-۱۶ تابع زیر را بصورت جمع مینترم‌ها و ضرب ماکسترم‌ها بیان کنید:

$$F(A, B, C, D) = B'D + A'D + BD$$

۲-۱۷ متمم توابع زیر را به جمع مینترم‌ها بنویسید:
(الف) $F(A, B, C, D) = \Sigma(0, 2, 6, 11, 13, 14)$ (ب) $F(x, y, z) = \Pi(0, 3, 6, 7)$

۲-۱۸ توابع زیر را به فرم متعارف دیگری بنویسید:
(الف) $F(x, y, z) = \Sigma(1, 3, 7)$ (ب) $F(A, B, C, D) = \Pi(0, 1, 2, 3, 4, 6, 12)$

۲-۱۹ عبارات بولی زیر را به جمع حاصلضرب‌ها و ضرب حاصل جمع‌ها تبدیل کنید.
(الف) $(AB + C)(B + C'D)$ (ب) $x' + x(x + y')(y + z')$

۲-۲۰ بدون ساده‌کردن توابع زیر نمودار منطقی آنها را رسم نمایید:

(الف) $BC' + AB + ACD$ (ب) $(A + B)(C + D)(A' + B + D)$

(پ) $(AB + A'B')(CD' + C'D)$

۲-۲۱ نشان دهید که دوگان XOR با متمم آن برابر است.

۲-۲۲ با جایگزینی عبارت بولی معادل اعمال دودویی در جدول ۸-۲ نشان دهید که:

(الف) عمل نهی نه جابجاپذیر و نه شرکت‌پذیر است.

(ب) عمل XOR جابجاپذیر و اشتراک‌پذیر است.

۲-۲۳ نشان دهید که گیت NAND با منطق مثبت، یک گیت NOR با منطق منفی است و بالعکس.

مراجع

1. BOOLE, G. 1954. *An Investigation of the Laws of Thought.* New York: Dover.
2. SHANNON, C. E. A symbolic analysis of relay and switching circuits. *Trans. AIEE* 57 (1938): 713-723.
3. HUNTINGTON, E. V. Sets of independent postulates for the algebra of logic. *Trans. Am. Math. Soc.,* 5 (1904): 288-309.
4. MANO, M. M. and C. R. Kime. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.
5. DIETMEYER, D. L. 1988. *Logic Design of Digital Systems,* 3rd ed. Boston: Allyn Bacon.

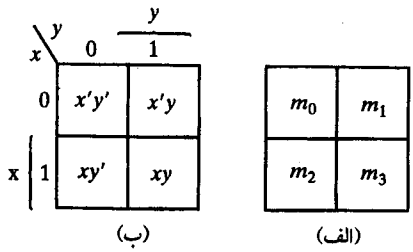
حداقل سازی در سطح گیت

۳-۱ روش نقشه

پیچیدگی گیت‌های منطقی دیجیتال که یک تابع بول را پیاده‌سازی می‌کنند، مستقیماً به پیچیدگی عبارات جبری که توسط آن تابع پیاده‌سازی می‌شوند بستگی دارد. گرچه جدول درستی یک تابع نمایش منحصر به فردی دارد، اما وقتی به صورت جبری بیان شود، می‌تواند فرم‌های متفاوتی داشته باشد. عبارات بول را می‌توان همان‌طور که در بخش ۴-۲ بحث شد، به صورت جبری ساده کرد. با این وجود، این روش حداقل‌سازی به دلیل کمبود قوانین خاص در پیشگویی مرحله بعدی فرآیند دستکاری، مشکل است. روش نقشه، روالی ساده را برای ساده‌سازی توابع بول پیش پا می‌گذارد. این روش را می‌توان فرم مصور جدول درستی تصور کرد. روش نقشه را نقشه کارنو یا نقشه K هم می‌نامند.

نقشه نموداری است متشکل از مربعات که هر مربع یک مینترم از تابع را نشان می‌دهد. چون هر تابع بول را می‌توان به مجموعی از مینترم‌ها نشان داد، بنابراین نتیجه می‌شود که یک تابع بولی در نقشه را می‌توان با مربعاتی که مینترم‌های متعلق به آنها در تابع وجود دارد به صورت گرافیکی شناسایی کرد. در واقع نقشه، نمایشی عینی از همه راه‌هایی است که یک تابع ممکن است در فرم استاندارد داشته باشد. با تشخیص همه الگوهای مختلف، کاربر می‌تواند عبارات جبری مختلفی برای یک تابع بدست آورده و از میان آنها ساده‌ترین را انتخاب کند.

عبارات ساده شده حاصل از نقشه همیشه به یکی از دو فرم استاندارد جمع حاصلضرب‌ها و ضرب حاصل جمع‌ها می‌باشد. فرض بر این است که ساده‌ترین عبارت جبری، دارای حداقل جملات با کمترین لیترال در هر جمله باشد. این فرض نموداری با حداقل گیت را فراهم نموده و تعداد ورودی‌ها به گیت نیز حداقل خواهد بود. بعد خواهیم دید که ساده‌ترین عبارت منحصر به فرد نیست. گاهی ممکن است دو یا چند عبارت بیابیم که معیار حداقل‌سازی را برآورد. در این حالت هر یک از دو حل رضایت‌بخش خواهد بود.



شکل ۱-۳. نقشه دو متغیره

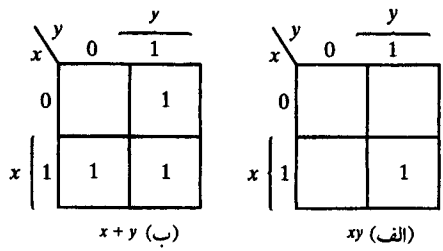
نقشه دو متغیره

نقشه دو متغیره در شکل ۱-۳ (الف) نشان داده شده است. در این نقشه چهار مینترم برای دو متغیر وجود دارد؛ از این رو نقشه متشکل از چهار مربع است، که هر یک متعلق به یک مینترم می‌باشد. نقشه (ب) برای نمایش ارتباط بین مربعات و دو متغیر x و y دوباره ترسیم شده است. 0 و 1 موجود در هر سطر و ستون مقدار متغیر را نشان می‌دهند. متغیر x در سطر 0 پریم‌دار و در سطر 1 بدون پریم است. به طور مشابه y در ستون 0 پریم‌دار و در ستون 1 بدون پریم می‌باشد.

اگر مربع‌هایی را که مینترم آنها متعلق به تابع مفروضی است با علامتی مشخص کنیم، روش مفید دیگری برای نمایش هر یک از 16 تابع ممکن از دو متغیر بدست می‌آید. به عنوان مثال تابع xy در شکل ۲-۳ (الف) دیده می‌شود. چون xy برابر m_3 است، یک 1 در داخل مربع متعلق به m_3 قرار می‌دهیم. به طور مشابه تابع x+y در نقشه شکل ۲-۳ (ب) نشان داده شده است که در آن سه مربع با 1 علامت زده شده‌اند. این مربعات از مینترم‌های تابع به دست آمده‌اند:

$$m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

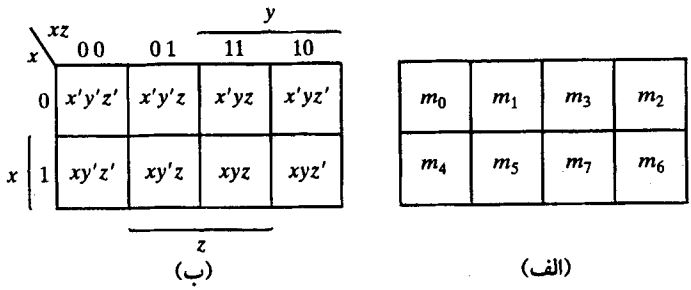
سه مربع از تلاقی x در سطر دوم و متغیر y در ستون دوم، که ناحیه متعلق به x یا y را پوشش می‌دهند، نیز بدست می‌آید.



شکل ۲-۳. نمایش توابع در نقشه

نقشه سه متغیره

یک نقشه سه متغیره در شکل ۳-۳ مشاهده می‌شود. برای سه متغیر هشت مینترم وجود دارد. بنابراین نقشه از هشت مربع تشکیل یافته است. توجه کنید که مینترم‌ها براساس ترتیب دودویی مرتب



شکل ۳-۳. نقشه سه متغیره

نشده‌اند، بلکه ترتیبشان براساس کدگری جدول (۴-۱) فهرست شده است. ویژگی این ترتیب این است که هنگام عبور از یک ستون به ستون مجاور تنها یک بیت از نظر مقدار تغییر می‌کند. برای نشان دادن رابطه بین مربع‌ها و سه متغیر نقشه، بخش (ب) با اعدادی در هر سطر و هر ستون علامت‌گذاری شده است. مثلاً مربع متعلق به m_5 مربوط به سطر 1 و ستون 01 است. وقتی دو عدد در کنار هم قرار گیرند عدد دودویی 101 حاصل می‌شود که معادل دهدهی آن عدد 5 می‌باشد. به طریقی دیگر هم می‌توان به مربع $m_5 = xy'z$ نگاه کرد به این ترتیب که بگوییم m_5 در سطر مربوط به x و ستون متعلق به $y'z$ است (ستون 01). توجه کنید که هر متغیر در چهار مربع مقدار 0 و در چهار مربع دیگر مقدار 1 را دارد. به منظور تفکیک، هر متغیر را در خانه‌های 1 بدون پریم و در خانه‌های 0 با پریم نشان می‌دهیم. برای سادگی، متغیر را با سمبل حرفی‌اش در زیر مربعاتی که بدون پریم هستند می‌نویسیم.

جهت درک برتری‌های نقشه کارنو در ساده‌سازی توابع بول، باید خاصیت مربع‌های همجوار را مشخص کنیم. تنها اختلاف بین هر دو مربع مجاور در نقشه این است که در یکی متغیری با پریم و در دیگری بدون پریم ظاهر می‌شود. مثلاً، m_5 و m_7 در دو مربع مجاور قرار دارند. متغیر y در m_5 پریم‌دار و در m_7 بدون پریم است، ضمن این که دو متغیر دیگر در هر دو مربع یکسانند. با توجه به اصول جبر بول، نتیجه می‌گیریم که جمع دو مینترم در مربع‌های مجاور را می‌توان به یک جمله AND متشکل از دو لیترال ساده کرد. برای روشن شدن مطلب، مجموع دو مربع همجوار مانند m_5 و m_7 را ملاحظه کنید.

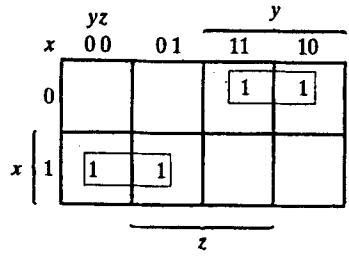
$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

در اینجا دو مربع در متغیر y با هم اختلاف دارند که هنگام تشکیل جمع دو مینترم حذف می‌شود. بنابراین هر دو مینترمی که در دو مربع مجاور با هم OR شوند موجب حذف متغیری می‌گردند که در آن دو مینترم متفاوت‌اند. مثال‌های زیر روال حداقل‌سازی یک تابع بول را با یک نقشه توضیح می‌دهد.

مثال ۱-۳

تابع بولی زیر را ساده کنید.

$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$



شکل ۴-۳. نقشه مثال ۱-۳ $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

ابتدا یک 1 در هر مربعی که مینترم تابع را نشان دهد قرار می‌دهیم. این کار در شکل ۴-۳ به این ترتیب انجام شده است که مربعات مینترم‌های 010، 011، 100 و 101 با 1 علامت زده شده‌اند. قدم بعدی یافتن مربع‌های مجاور است. این کار در نقشه با زیرمربع‌هایی که هر یک دو عدد 1 را در بر می‌گیرند صورت گرفته است. زیرمربع یا مستطیل بالای سمت راست ناحیه پوشش یافته با $x'y$ را شامل می‌شود. این دو مربع در سطر 0 قرار دارند که با x' و نیز در دو ستون آخر با y نشان داده می‌شوند. به طور مشابه مستطیل پایین سمت چپ جمله ضرب xy' را نشان می‌دهد (سطر دوم نشان دهنده x و دو ستون چپ نیز y' است). جمع منطقی این دو جمله ضرب، عبارت ساده شده را نتیجه می‌دهد.

$$F = x'y + xy'$$

❑

مواردی وجود دارد که در آنها دو مربع همجوارند ولی به هم نچسبیده‌اند. در شکل ۳-۳، m_0 مجاور m_2 و m_4 مجاور m_6 است زیرا مینترم‌ها تنها در یک متغیر با هم اختلاف دارند. این مطلب به راحتی با کمک جبر قابل اثبات است.

$$m_0 + m_2 = x'y'z' + x'yz' = x'z'(y' + y) = x'z'$$
$$m_4 + m_6 = xy'z' + xyz' = xz' + (y' + y) = xz'$$

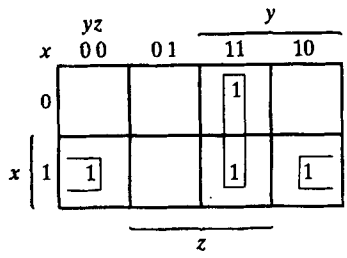
در نتیجه ما باید تعریف مربع‌های همجوار را اصلاح کنیم تا این حالت و دیگر حالات مشابه را نیز شامل شود. این تصحیح بدین صورت انجام می‌گیرد که نقشه کشیده شده در یک سطح از دو لبه سمت چپ و راست مجاور تصور شوند.

مثال ۲-۳

تابع بول زیر را ساده کنید.

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

نقشه این تابع در شکل ۵-۳ ترسیم شده است. در این شکل 4 مربع با 1 علامت خورده‌اند که هر کدام متعلق به یک مینترم است. دو مربع همجوار در ستون سوم با هم ترکیب شده‌اند تا جمله دو لیترال yz را به وجود آورند.



شکل ۵-۳. نقشه مثال ۲-۳ $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

دو مربع باقیمانده هم براساس تعریف جدید مجاورند و در نمودار با نیم مربع‌ها محصور شده‌اند. این دو مربع، وقتی ترکیب شوند جمله دو لیترالی xz' را بدست می‌دهند. بنابراین تابع ساده شده به فرم زیر است.

$$F = yz + xz'$$

❑

اکنون به ترکیب چهار مربع همجوار در نقشه سه متغیره توجه نمایید. چنین ترکیبی نشان دهنده جمع منطقی چهار مینترم مجاور است و نتیجه این ترکیب، تولید عبارتی با تنها یک متغیر است. به عنوان مثال جمع منطقی چهار مینترم مجاور 0، 2، 4 و 6 عبارت را به جمله یک لیترالی z' کاهش می‌دهد.

$$\begin{aligned} m_0 + m_2 + m_4 + m_6 &= x'y'z' + x'yz' + xy'z' + xyz' \\ &= x'z'(y' + y) + xz'(y' + y) \\ &= x'z' + xz' = z'(x' + x) = z' \end{aligned}$$

تعداد مربعات مجاوری که ممکن است ترکیب شوند همواره برابر توانی از 2، مانند 1، 2، 4 و 8 می‌باشد. هر چقدر تعداد بیشتری از مربعات همجوار ترکیب شوند جمله حاصلضرب نتیجه، تعداد کمتری لیترال خواهد داشت.

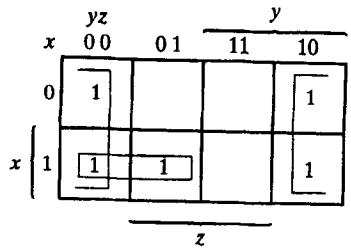
یک مربع یک مینترم را نمایش می‌دهد و دارای سه لیترال است.
دو مربع مجاور یک جمله دو لیترال را نشان می‌دهند.
چهار مربع همجوار یک جمله با یک لیترال را نشان می‌دهند.
هشت مربع همجوار که تمام نقشه را می‌پوشانند همواره تابع 1 را تولید می‌کنند.

مثال ۳-۳

تابع بول زیر را ساده کنید.

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

نقشه تابع F در شکل ۶-۳ نشان داده شده است. ابتدا چهار مربع مجاور در اولین و آخرین ستون را با هم ترکیب می‌کنیم تا جمله تک لیترال z' بدست آید. تنها مینترم باقیمانده که متعلق به مینترم 5 است با مربع



شکل ۶-۳. نقشه مثال ۳-۲ $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

مجاورش که قبلاً به کار رفته، ترکیب می‌گردد. این کار نه تنها مجاز است بلکه مفید نیز می‌باشد، زیرا دو مربع مجاور جمله دو لیترالی xy' را تولید می‌کنند در حالی که یک مربع تنها، جمله سه لیترال $xy'z$ را نمایش می‌دهد. تابع ساده به صورت زیر است.

$$F = z' + xy'$$

❑

اگر تابعی به صورت مجموع مینترم‌ها بیان نشود می‌توان از نقشه برای بدست آوردن مینترم‌های تابع استفاده کرد و سپس تابع را به صورت جملاتی با حداقل لیترال‌ها ساده نمود. البته باید عبارت جبری حتماً به صورت جمع حاصلضرب‌ها باشد. هر جمله ضرب را می‌توان با نقشه‌ای متشکل از یک، دو یا چند مربع در نقشه نشان داد. آنگاه مینترم‌های تابع مستقیماً از جدول استخراج می‌شوند.

مثال ۴-۳

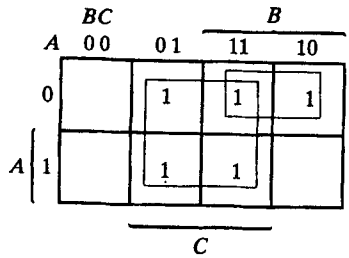
تابع بول زیر را ساده کنید.

$$F = A'C + A'B + AB'C + BC$$

(الف) آن را به مجموع مینترم‌ها نشان دهید.

(ب) و سپس عبارت مجموع حاصلضرب حداقل را پیدا کنید.

سه جمله ضرب در عبارت دو لیترال دارند و در نقشه سه متغیره، هر یک با دو مربع نشان داده شده‌اند. دو مربع مربوط به جمله اول، $A'C$، در شکل ۷-۳ از تلاقی A' (اولین سطر) و C (دو ستون میانی) بدست



شکل ۷-۳. نقشه مثال ۴-۳ $F = A'C + A'B + AB'C + BC = C + A'B$

می‌آید تا مربعات 001 و 011 را بدهند. توجه کنید که وقتی 1ها را در مربعات می‌گذارید، ممکن است یک 1 را که از جمله قبلی در آن قرار داده شده بیابید. این نکته برای دومین جمله، A′B، رخ می‌دهد که دو عدد 1 در مربع‌های 01 و 010 قرار دارند. مربع 011 با جمله اول، A′C، مشترک است، بنابراین تنها یک 1 در آن قرار داده می‌شود. به همین ترتیب می‌بینیم که جمله AB′C متعلق به مربع 101، یعنی مینترم 5 است، و جمله BC متعلق به دو مربع 011 و 111 می‌باشد. تابع جمعاً پنج مینترم دارد و در نقشه شکل ۷-۳ هم با پنج عدد 1 نشان داده شده است. مینترم‌هایی که مستقیماً از نقشه خوانده می‌شوند عبارتند از 1، 2، 3، 5 و 7. تابع را می‌توان به صورت جمع مینترم‌ها نشان داد.

$$F(A, B, C) = \Sigma(1, 2, 3, 5, 7)$$

عبارت جمع حاصلضرب مفروض اولیه چندین جمله دارد. همانطور که در نقشه مشاهده می‌شود می‌توان آن را ساده کرده و عبارتی دو جمله‌ای بدست آورد.

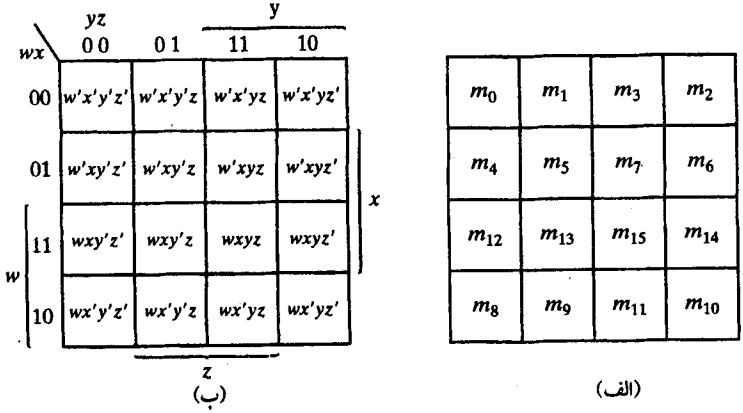
$$F = C + A'B$$

❑

۳-۲ نقشه چهار متغیره

نقشه توابع بول چهار متغیره در شکل ۸-۳ نشان داده شده است. در (الف) 16 جمله مینترم فهرست شده و به هر یک مربعی تخصیص داده شده است. در (ب) نقشه دوباره رسم شده تا بیانگر ارتباط بین چهار متغیر باشد. سطرها و ستون‌ها براساس کدگری شماره‌گذاری شده‌اند، و بین هر دو سطر یا ستون مجاور تنها یک رقم تغییر می‌کند. مینترم متعلق به هر مربع از ترکیب شماره سطر و شماره ستون آن بدست می‌آید. مثلاً وقتی اعداد سطر سوم (11) و ستون دوم (01) ترکیب شوند عدد دودویی 1101 حاصل می‌گردد، که معادل 13 دهدهی است. بنابراین، مربع در سطر سوم و ستون دوم مینترم m_{13} را نمایش می‌دهد.

ساده کردن توابع بول چهار متغیره مشابه با روش به کار رفته برای توابع سه متغیره است. مربعات مجاور مربعاتی هستند که در کنار یکدیگرند. به علاوه نقشه در سطحی واقع است و لبه‌های بالا و پایین و چپ و راست آن نیز مجاور است تا به این ترتیب مربعات همجوار را بسازند. مثلاً m_0 و m_2 و نیز m_{11}.



شکل ۸-۳. نقشه چهار متغیره

و m_3 هر کدام مربعات مجاور را می‌سازند. ترکیب مربعات همجوار به راحتی با بررسی نقشه چهار متغیره قابل تشخیص است:

یک مربع یک مینترم را نمایش می‌دهد، و جمله آن چهار لیترالی است.
دو مربع همجوار یک جمله سه لیترالی را می‌سازند.
چهار مربع همجوار یک جمله دو لیترالی را نشان می‌دهند.
هشت مربع همجوار یک جمله یک لیترالی را نمایش می‌دهند.
شانزده مربع همجوار تابعی برابر 1 را تولید می‌کنند.

هیچ ترکیب دیگری از مربع‌ها نمی‌تواند تابع را ساده کند. دو مثال زیر روال بکار رفته برای توابع بول چهار متغیره را نشان می‌دهد.

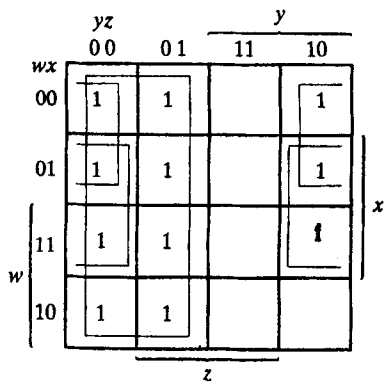
مثال ۵-۳

تابع بول زیر را ساده کنید.

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$$

چون تابع چهار متغیر دارد، باید از نقشه چهار متغیره استفاده کرد. مینترم‌های لیست شده در مجموع فوق با 1ها در نقشه شکل ۹-۳علامت زده شده‌اند. هشت 1 مجاور می‌توانند با هم ترکیب شده و جمله تک لیترالی y' را نتیجه دهند. سه 1 باقیمانده در سمت راست نمی‌توانند با هم ترکیب و جمله ساده‌ای بدهند. آنها باید به صورت دو یا چهار مربع مجاور با هم ترکیب شوند. هر چقدر تعداد مربعات ترکیب شده بیشتر باشد، تعداد لیترال‌ها در جمله کمتر خواهد بود.

در این مثال دو 1 فوقانی سمت راست با دو 1 فوقانی در سمت چپ ترکیب شده و جمله $w'z'$ را می‌دهند. توجه داشته باشید که می‌توان یک مربع را بیش از یک بار به کار برد. حال فقط یک مربع در سطر سوم و ستون چهارم (مربع 1110) باقیمانده است. در عوض انتخاب این مربع به تنهایی، آن را با مربع‌هایی که قبلاً به



شکل ۹-۳. نقشه مثال ۵-۳

$$F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$$

کار رفته‌اند برای ایجاد مربع‌های مجاور ترکیب می‌کنیم. این مربعات شامل دو سطر میانی و دو ستون انتهایی بوده و جمله xz' را تولید می‌کنند. تابع ساده شده به صورت زیر است:

$$F = y' + w'z' + xz'$$

❑

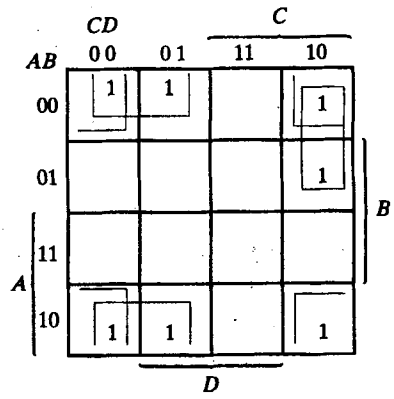
مثال ۶-۳

تابع زیر را ساده کنید.

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

ناحیه مفروش شده با این تابع شامل مربعاتی است که در شکل ۱۰-۳ با 1 علامت زده شده است. این تابع دارای چهار متغیر بوده و همانطور که دیده می‌شود سه جمله سه لیترالی و یک جمله چهار لیترالی دارد. هر جمله سه لیترالی در نقشه با دو مربع نمایش داده شده است. مثلاً $A'B'C'$ در مربعات 0000 و 0001 نشان داده شده است. تابع را می‌توان با انتخاب چهار 1 در گوشه‌ها و ترکیب آنها برای بدست آوردن جمله $B'D'$ ساده کرد. این عمل مجاز است زیرا وقتی نقشه را سطحی تصور کنیم که لبه‌های چپ و راست و لبه‌های پایین و بالای آن با هم مجاورند، این چهار مربع همجوار خواهند بود. دو 1 سمت چپ در سطر بالا با دو 1 در سطر پایین ترکیب می‌شوند تا جمله $B'C'$ حاصل شود. تنها 1 باقیمانده را به صورت دو مربع ترکیب می‌کنیم تا $A'CD'$ حاصل گردد. تابع ساده شده به صورت زیر خواهد بود.

$$F = B'D' + B'C' + A'CD'$$

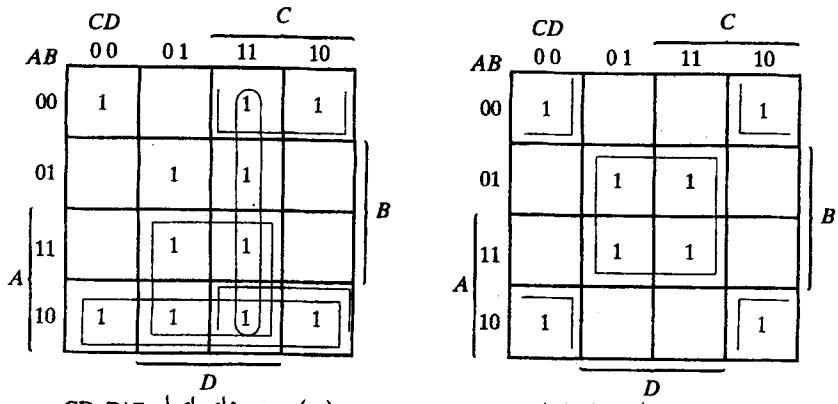


شکل ۱۰-۳. نقشه مثال ۶-۲

❑ $F = A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

موجب‌های اصلی

هنگام انتخاب مربع‌های مجاور در یک نقشه باید مطمئن شویم که همه مینترم‌های تابع هنگام



شکل ۱۱-۳. ساده‌سازی با موجب‌های اصلی

ترکیب مربع‌ها پوشش داده شده‌اند. همچنین باید تعداد جملات در عبارت حداقل شود و هر جمله‌ای که مینترم آن قبلاً به وسیله دیگر جملات به کار رفته نیز کنار گذاشته شود. گاهی نیز ممکن است دو یا سه عبارت بر معیار ساده‌سازی صحّه بگذارند. روش ترکیب مربع‌ها در نقشه را می‌توان سینماتیک‌تر کرد به شرطی که مفهوم جملات موجب‌های اصلی و موجب‌های اصلی اساسی خوب فهمیده شوند. یک موجب اصلی جمله‌ای حاصلضربی است که از ترکیب حداکثر مربعات مجاور به هم حاصل می‌گردد. اگر مینترمی در یک مربع تنها با یک موجب اصلی پوشش یابد، به آن موجب اصلی اساسی گوییم.

موجب‌های اصلی یک تابع را می‌توان با ترکیب حداکثر تعداد مربعات ممکن بدست آورد. این بدان معنی است که یک 1 تنها اگر در مجاورت هر 1 دیگر در نقشه نباشد، یک موجب اصلی است. دو 1 مجاور به شرطی یک موجب اصلی را ایجاد می‌کنند که در داخل یک گروه چهارتایی مربع‌ها واقع نباشند. چهار 1 مجاور یک موجب اصلی را تشکیل می‌دهند بشرطی که در یک گروه از هشت مربع همجوار نباشند و بهمین ترتیب. موجب اصلی اساسی با نظاره بر مربعات 1 و وارسی تعداد موجب‌های اصلی که آن را پوشش می‌دهد تعیین می‌گردد. یک موجب اصلی، اساسی است اگر تنها موجب اصلی باشد که مینترم را پوشش می‌دهد.

تابع چهار متغیره زیر را در نظر بگیرید:

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

مینترم‌های تابع با 1 در نقشه‌های شکل ۱۱-۳ علامت زده شده‌اند. بخش (الف) از شکل، دو موجب اصلی اساسی را نشان می‌دهد. یک موجب، اساسی است زیرا تنها یک راه برای پوشش m_0 در چهار مربع مجاور وجود دارد. این چهار مربع جمله $B'D'$ را تعریف می‌کنند. به طور مشابه، برای ترکیب m_5 با چهار مربع مجاور تنها یک راه وجود دارد و جمله BD از آن حاصل می‌گردد. این دو موجب اصلی اساسی هشت مینترم را پوشش می‌دهند. سه مینترم باقیمانده m_3، m_9 و m_{11} باید بعد ملاحظه شوند.

شکل ۱۱-۳ (ب) همه راه‌های ممکن که سه مینترم با موجب‌های اصلی پوشش می‌یابند را نشان می‌دهد. مینترم m_3 می‌تواند با موجب اصلی CD یا $B'C$ پوشش یابد. مینترم m_9 با هر یک از AD یا AB' پوشش می‌یابد. مینترم m_{11} نیز با هر یک از چهار موجب اصلی می‌تواند پوشش پیدا کند. عبارت ساده شده از جمع منطقی دو موجب اصلی اساسی، و هر دو موجب اصلی دیگر که مینترم‌های m_3 و m_9 و m_{11} را پوشش دهند بدست می‌آید. چهار امکان برای بیان تابع با چهار جمله ضرب که هر یک دو لیترال دارند وجود دارد:

$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

مثال فوق نشان داد که شناسایی موجب‌های اصلی در نقشه در تعیین صور متفاوت تابع ساده شده کمک مؤثری می‌نمایند.

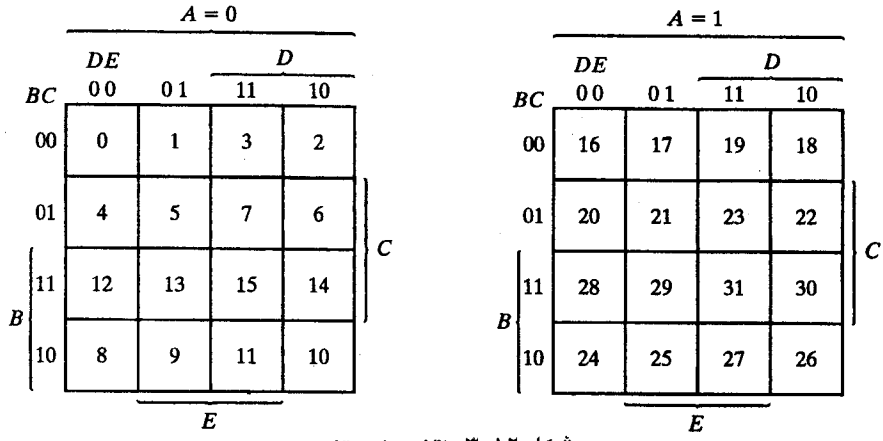
روال یافتن عبارت ساده شده از نقشه لازم می‌دارد که ابتدا تمام موجب‌های اصلی اساسی را معین کنیم. تابع ساده شده از جمع منطقی همه موجب‌های اصلی اساسی، به علاوه دیگر موجب‌های اصلی حاصل می‌گردد. این موجب‌های اصلی ممکن است برای پوشش مینترم‌های باقیمانده‌ای که در موجب اصلی اساسی وجود ندارند لازم باشد. گاهی بیش از یک راه برای ترکیب مربعات وجود دارد و هر ترکیب هم ممکن است عبارت ساده شده یکسانی را تولید کند.

۳-۳ نقشه پنج متغیره

استفاده از نقشه‌هایی که بیش از چهار متغیر دارند چندان ساده نیست. یک نقشه پنج متغیره به 32 مربع و نقشه شش متغیره به 64 مربع نیاز دارد. وقتی تعداد متغیرها زیاد شود، تعداد مربعات هم به طور بی‌رویه‌ای افزایش می‌یابند و یافتن مربعات همجوار بیش از پیش به شکل هندسی وابسته می‌گردد.

یک نقشه پنج متغیره در شکل ۱۲-۳ نشان داده شده است. این نقشه، از دو نقشه چهار متغیره با متغیرهای A ، B ، C ، D و E تشکیل یافته و متغیر A آن دو را از هم تفکیک کرده است. نقشه چهار متغیره سمت چپ 16 مربعی را نشان می‌دهد که در آن $A = 0$ است، و دیگر نقشه چهار متغیره، مربعات مربوط به $A = 1$ را نمایش می‌دهد. مینترم‌های 0 تا 15 متعلق به A=0 و مینترم‌های 16 تا 31 متعلق به A=1 است. هر نقشه چهار متغیره وقتی جداگانه بررسی شود همجواری تعریف شده قبلی خود را حفظ می‌کند. به علاوه هر مربع از نقشه A=0 با مربع متناظرش در مربع A=1 همجوار است. مثلاً مینترم 4 با مینترم 20 و مینترم 15 با 31 مجاور است. بهترین راه تجسم این قانون برای مربع‌های همجوار این است که این دو نیم نقشه را بر روی یکدیگر تصور کنیم. هر دو مربعی که روی هم قرار گیرند مجاور شناخته می‌شوند.

با پیگیری روشی که برای نقشه پنج متغیره به کار رفت، می‌توان نقشه شش متغیره را با 4 نقشه چهار متغیره بدست آورد تا 64 مربع مورد نیاز حاصل گردد. نقشه‌هایی با شش یا تعداد بیشتری متغیر، نیاز به



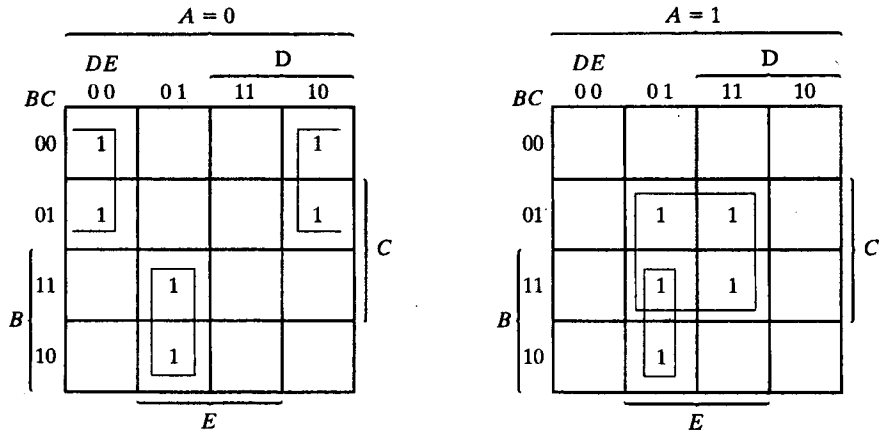
شکل ۱۲-۳. نقشه پنج متغیر

تعداد بی‌شماری مربع داشته و استفاده از آنها غیرعملی است. روش دیگر، استفاده از برنامه‌های کامپیوتری در ساده‌سازی توابع بول با متغیرهای بی‌شمار می‌باشد.

با بررسی و در نظر گرفتن تعریف جدید همجواری مربعات، می‌توان نشان داد که 2^k مربع همجوار به ازاء $k = (0, 1, 2, ..., n)$ در یک نقشه n متغیره ناحیه را مشخص می‌کند که نمایش‌دهنده جمله‌ای با $n - k$ لیترال است. برای این که عبارت فوق مفهوم داشته باشد باید همیشه n بزرگتر از k باشد. وقتی $n = k$ است، تمام سطح نقشه ترکیب شده و تابع یگانی (1) را تولید می‌کند. جدول ۱-۳ رابطه بین تعداد مربعات مجاور و تعداد لیترال در هر جمله را نشان می‌دهد. مثلاً هشت مربع مجاور ناحیه‌ای را در نقشه پنج متغیره ترکیب می‌کنند تا یک جمله دو متغیره حاصل شود.

جدول ۱-۳. رابطه بین تعداد مربعات مجاور و تعداد لیترال‌ها در یک جمله

	تعداد مربعات مجاور	تعداد لیترال‌ها در یک جمله در یک نقشه n متغیره			
K	2^k	$n = 2$	$n = 3$	$n = 4$	$n = 5$
0	1	2	3	4	5
1	2	1	2	3	4
2	4	0	1	2	3
3	8		0	1	2
4	16			0	1
5	32				0



شکل ۱۳-۳. نقشه مثال ۷-۲ $F = A'B'E' + BD'E + ACE$

مثال ۷-۳

تابع بول زیر را ساده کنید.

$$F(A, B, C, D, E) = (0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$$

نقشه پنج متغیره برای این تابع در شکل ۱۳-۳ دیده می‌شود. در بخشی از نقشه که متعلق به مینترم‌های 0 تا 15 است، A = 0 بوده و در آن شش مینترم مقدار 1 را دارند. پنج مینترم دیگر به بخش A = 1 متعلق است. چهار مربع مجاور در نقشه A = 0 با هم ترکیب شده‌اند تا جمله A′B′E′ را بدهند. توجه کنید که باید A′ را نیز در جمله منظور کنیم زیرا تمام مربع‌ها متعلق به نقشه A = 0 می‌باشند. دو مربع در ستون 01 و دو سطر آخر در هر دو بخش نقشه مشترکند. بنابراین آنها چهار مربع مجاور را تشکیل داده و جمله سه متغیره BD′E را می‌سازند. در اینجا متغیر A آورده نشده است زیرا مربع‌های مجاور به هر دو A = 0 و A = 1 متعلق‌اند. جمله ACE از چهار مربع همجوار در نقشه A = 1 بدست می‌آید. تابع ساده شده جمع منطقی سه جمله می‌باشد.

$$F = A'B'E' + BD'E + ACE$$

❑

۳-۴ ساده‌سازی با ضرب حاصل جمع‌ها

در تمام مثال‌های قبلی توابع بول حاصل از نقشه به فرم جمع حاصلضرب‌ها بیان شدند. با کمی اصلاح می‌توان فرم ضرب حاصل جمع‌ها را بدست آورد.

روال تهیه یک تابع حداقل برحسب ضرب حاصل جمع‌ها از خواص اصلی توابع بول حاصل می‌گردد. 1های واقع در مربع‌های نقشه نشانگر مینترم‌های تابع است. مینترم‌هایی که در تابع ذکر نشوند

متمم تابع را بیانگرند. با توجه به این مطلب مشاهده می‌کنیم که متمم یک تابع به وسیله مربع‌هایی که با 1 علامت‌زنی نشده‌اند بیان می‌گردد. اگر در مربع‌های خالی 0 قرار داده و آنها را با روش مربع‌های همجوار ترکیب کنیم عبارت ساده شده متمم تابع یعنی F' را بدست خواهیم آورد. متمم F' به ما تابع F را باز می‌گرداند. به دلیل عمومیت تئوری دمورگان تابع حاصل به طور خودکار به صورت ضرب حاصل جمع‌هاست. بهترین روش برای تشریح این مطلب ارائه یک مثال است.

مثال ۸-۳

تابع بولی زیر را (الف) به صورت جمع حاصلضرب‌ها، (ب) ضرب حاصل جمع‌ها ساده کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

1های موجود در نقشه شکل ۱۴-۳ همه مینترم‌های تابع را نمایش می‌دهند. مربع‌هایی که با 0 علامت زده شده‌اند مینترم‌های غایب در F را نشان می‌دهند، بنابراین متمم F را بیانگر هستند. ترکیب مربعات حاوی 1ها تابع ساده شده را به صورت جمع حاصلضرب‌ها بدست می‌دهد:

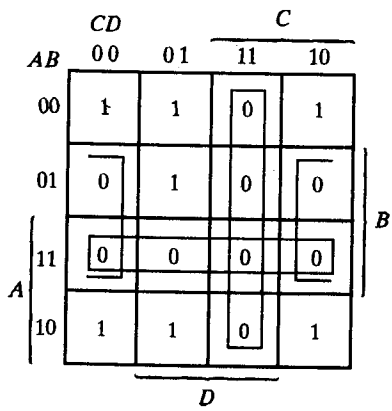
(الف) $$F = B'D' + B'C' + A'C'D$$

اگر مربعات حاوی 0ها را ترکیب کنیم، تابع متمم ساده شده بدست خواهد آمد:

$$F' = AB + CD + BD'$$

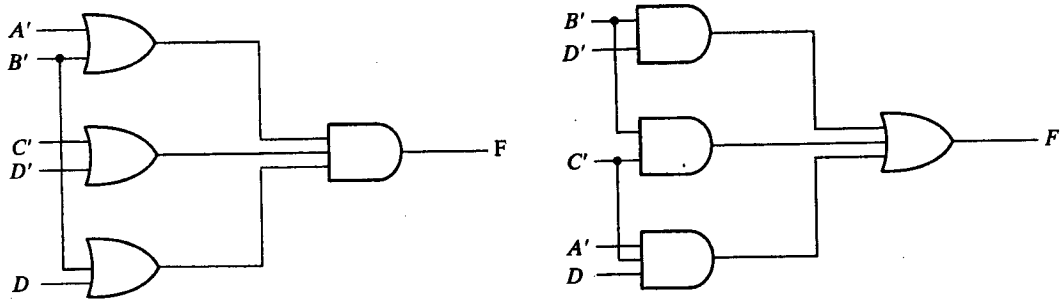
با اعمال تئوری دمورگان (استفاده از دوگان و متمم کردن هر متغیر طبق آنچه در بخش ۴-۲ دیدیم) تابع ساده شده را به صورت ضرب حاصل جمع‌ها بدست می‌آوریم:

(ب) $$F = (A' + B')(C' + D')(B' + D)$$ ❑



شکل ۱۴-۳. نقشه برای مثال ۸-۳

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10) = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$$



$F = (A' + B')(C' + D')(B' + D)$ (ب)

$F = B'D' + B'C' + A'C'D$ (الف)

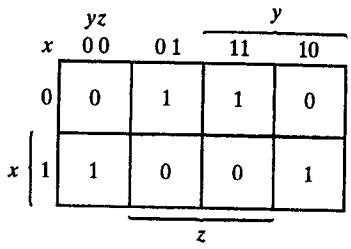
شکل ۱۵-۳. پیاده‌سازی با گیت تابع مثال ۸-۳

پیاده‌سازی عبارت ساده شده حاصل از مثال ۸-۳ در شکل ۱۵-۳ دیده می‌شود. عبارت جمع‌حاصلضرب‌ها در بخش (الف) با گروهی از گیت‌های AND پیاده‌سازی شده است. خروجی گیت‌های AND نیز به ورودی‌های یک گیت OR متصل گردیده است. همان تابع به صورت ضرب حاصل جمع‌ها در شکل (ب) با تعدادی گیت OR ، که هر یک متعلق به یک جمله OR است، پیاده‌سازی شده و خروجی آنها به یک AND منتهی گشته است. در هر دو حال فرض بر این است که متمم متغیرها نیز مستقیماً در دسترسند و بنابراین نیازی به وارونگر نمی‌باشد. الگوهای ایجاد شده در شکل ۱۵-۳ یک سری روش‌های کلی می‌باشند که به وسیله آنها هر تابع بول استاندارد قابل پیاده‌سازی است. در جمع حاصلضرب‌ها، گیت‌های AND به یک OR ختم می‌شوند و در ضرب حاصل جمع‌ها گیت‌های OR به یک AND متصل می‌گردند. هر یک از دو پیکربندی فوق دارای دو سطح از گیت‌ها می‌باشند. بنابراین پیاده‌سازی یک تابع استاندارد را پیاده‌سازی دو سطحی می‌گویند.

مثال ۸-۳ روالی را برای محاسبه فرم ساده شده یک تابع برحسب ضرب حاصل جمع‌ها، وقتی تابع ابتدا به صورت جمع مینترم‌ها است، نشان می‌دهد. این روال هنگامی که تابع در آغاز برحسب ماکسترم‌ها بیان شود نیز معتبر است. برای مثال به جدول درستی (۲-۳) که تابع F را تعریف می‌کند

جدول ۲-۳. جدول درستی تابع F

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0



شکل ۱۶-۳. نقشه برای تابع جدول (۲-۳)

توجه نمایید. این تابع به صورت جمع مینترم‌ها چنین بیان می‌شود:

$$F(x, y, z) = \Sigma(1, 3, 4, 6)$$

در ضرب ماکسترم‌ها تابع به صورت زیر است:

$$F(x, y, z) = \Pi(0, 2, 5, 7)$$

به بیان دیگر، 1های تابع، مینترم‌ها را نشان می‌دهند و 0های آن بیانگر جملات ماکسترم هستند. نقشه این تابع در شکل ۱۶-۳ دیده می‌شود. برای ساده کردن این تابع، در مربع مربوط به هر جمله مینترم که تابع به ازاء آن، مقدار 1 دارد، عدد 1گذاشته و بقیه مربع‌ها را با 0 پر می‌کنیم. از طرف دیگر اگر تابع به فرم ضرب ماکسترم‌ها داده شده باشد در ابتدا در مربعاتی که جملات آن در تابع است 0 قرار می‌دهیم و بقیه مربع‌ها با 1 پر می‌شوند. سپس تابع می‌تواند به یکی از فرم‌های استاندارد ساده شود. برای جمع حاصلضرب‌ها، 1ها را با هم ترکیب می‌کنیم و خواهیم داشت:

$$F = x'z + xz'$$

برای ضرب حاصل جمع‌ها، 0ها را با هم ترکیب می‌کنیم تا متمم تابع ساده شده به صورت زیر حاصل شود:

$$F' = xz + x'z'$$

که نشان می‌دهد تابع XOR متمم تابع هم‌ارزی (XNOR) است (بخش ۶-۲). با متمم‌گیری مجدد از F' تابع ساده شده را به ضرب حاصل جمع‌ها بدست خواهیم آورد.

$$F = (x' + z')(x + z)$$

برای وارد کردن یک تابع در یک نقشه که برحسب ضرب حاصل جمع‌ها بیان شده است، می‌باید متمم تابع را بدست آورد و در آن مربع‌های مربوطه را با 0 پر کرد. مثلاً تابع

$$F = (A' + B' + C')(B + D)$$

را می‌توان با متمم‌گیری از آن وارد جدول کرد.

$$F' = ABC + B'D'$$

آنگاه مربع‌هایی که مینترم‌های F' را تشکیل می‌دهند با 0 پر می‌کنیم. بقیه مربع‌ها را با 1 پر می‌نماییم.

۳-۵ حالات بی‌اهمیت

جمع منطقی مینترم‌های مربوط به یک تابع شرایطی را که تحت آن تابع برابر 1 است، مشخص

می‌نماید. تابع در ازاء بقیه مینترم‌ها 0 است. در این حالت فرض بر این است که همه ترکیبات مقادیر برای متغیرهای تابع معتبرند. در عمل کاربردهایی وجود دارند که در آنها در ازاء ترکیبات معینی از متغیرها، تابع مشخص نیست. مثلاً یک کد دودویی چهار بیتی برای ارقام دهدهی دارای شش ترکیب است که به کار نرفته‌اند و در نتیجه نامشخص تصور می‌گردند. توابعی که در ازاء ترکیبی از ورودی‌ها خروجی‌های نامشخص دارند، تابع غیرکامل نامیده می‌شود. در بسیاری از کاربردها، توجهی به مقدار منتسب به تابع در ازاء مینترم‌های نامعین نخواهیم داشت. به این دلیل مرسوم است که همه مینترم‌های نامشخص در تابع را حالات بی‌اهمیت بخوانیم. از حالات بی‌اهمیت می‌توان برای ساده‌سازی بیشتر عبارت بول در یک نقشه استفاده کرد.

باید توجه داشت که یک مینترم بی‌اهمیت ترکیبی از متغیرهاست که مقدار منطقی آن نامشخص است. به این دلیل نمی‌توان یک حالت بی‌اهمیت را در نقشه با 1 نشان داد زیرا این عمل به این معنی است که تابع برای ترکیب خاص از ورودیها همواره برابر 1 می‌باشد. به طور مشابه گذاشتن 0 در مربع‌های نقشه به معنی 0 بودن همیشگی تابع در آن حالت است. برای تفکیک حالت بی‌اهمیت از 1ها و 0ها از X استفاده می‌کنیم. بنابراین هر X در داخل یک مربع از نقشه به این معنی است که تخصیص 1 یا 0 به تابع به ازاء یک مینترم خاص فاقد اهمیت است.

وقتی مربع‌های مجاور انتخاب می‌گردند تا تابع در جدول ساده شود، مینترم‌های بی‌اهمیت با این ایده که ساده‌ترین فرم برای تابع بدست آید، برابر 1 یا 0 فرض می‌شوند. در ساده‌سازی تابع می‌توانیم با توجه به ساده‌ترین فرم ممکن برای تابع، به حالات بی‌اهمیت 0 یا 1 دهیم.

مثال ۹-۳

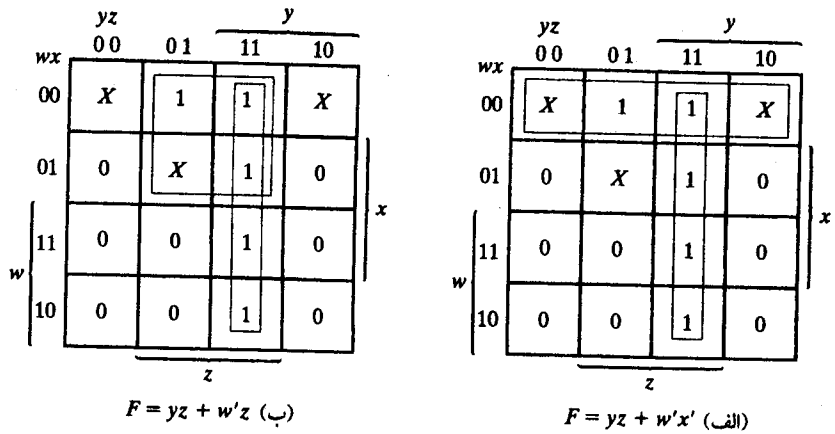
تابع بول زیر را ساده کنید.

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

که حالات بی‌اهمیت زیر را داراست.

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

مینترم‌های F ترکیباتی از متغیرها هستند که تابع را برابر 1 می‌کنند. مینترم‌های d مینترم‌های بی‌اهمیتی هستند که ممکن است به آنها 0 یا 1 تخصیص داده شود. ساده‌سازی نقشه در شکل ۱۷-۳ نشان داده شده است. مینترم‌های F با 1 علامت زده شده‌اند، مینترم‌های d با X علامت‌گذاری شده‌اند و بقیه مربع‌ها با 0 پر شده‌اند. برای بدست آوردن عبارت جمع حاصلضرب‌های ساده شده باید هر پنج 1 موجود در نقشه به حساب آیند، ولی بسته به روش ساده‌سازی ممکن است Xها را در نظر بگیریم و یا نگیریم. جمله yz چهار مینترم در سومین ستون را پوشش می‌دهد. مینترم باقیمانده m_1 می‌تواند با مینترم m_3 ترکیب شده و جمله سه لیترالی $w'x'z$ را بدهند. با این وجود با احتساب یک یا دو X همجوار، می‌توانیم چهار مربع مجاور را ترکیب نماییم تا جمله دو متغیره حاصل گردد. در بخش (الف) از نمودار، مینترم‌های بی‌اهمیت 0 و 2 با 1 جایگزین شده‌اند



شکل ۱۷-۳. مثال با حالات بی‌اهمیت

و تابع ساده شده به صورت زیر است.

$$F = yz + w'x'$$

در بخش (ب) از نمودار، مینترم بی‌اهمیت 5 با 1 جایگزین شده و آنگاه تابع ساده شده به فرم زیر است:

$$F = yz + w'z$$

هر یک از دو عبارت شرایط بیان شده برای این مثال را دارا هستند. ❑

مثال قبل نشان داد که مینترم‌های بی‌اهمیت در نقشه در ابتدا با Xها علامت خورده‌اند و فرض می‌شود که بتوانند 0 و یا 1 بشوند. انتخاب 0 و یا 1 به روش ساده کردن تابع غیرکامل وابسته است. پس از انتخاب، تابع ساده شده حاصل متشکل از مجموع مینترم‌ها است و در آنها مینترم‌هایی که در آغاز نامعلوم بوده ولی بعد به عنوان 1 انتخاب شده‌اند نیز وجود خواهند داشت. دو عبارت ساده شده حاصل در مثال ۹-۳ را در نظر بگیرید:

$$F(w, x, y, z) = yz + w'x' = \Sigma(0, 1, 2, 3, 7, 11, 15)$$
$$F(w, x, y, z) = yz + w'z = \Sigma(1, 3, 5, 7, 11, 15)$$

هر دو عبارت شامل مینترم‌های 1، 3، 7، 11 و 15 می‌باشند که تابع F را برابر 1 می‌کنند. مینترم‌های بی‌اهمیت در آن دو به طور متفاوتی به کار گرفته شده‌اند و در اولین عبارت مینترم‌های 0 و 2 برابر 1 گرفته شده و مینترم 5 با انتخاب 0 حذف شده است. در دومین عبارت مینترم 5 برابر 1 و مینترم‌های 0 و 2 با مقدار 0 جایگزین شده‌اند. دو عبارت توابعی را نشان می‌دهند که فرم جبری متفاوتی دارند. هر دو مینترم‌های مشخص شده را می‌پوشانند ولی هر یک مینترم‌های بی‌اهمیت متفاوتی را پوشش می‌دهند. مادامی که تابع مشخص شده غیرکامل است، هر دو عبارت قابل قبول‌اند زیرا تنها اختلاف در مقدار F مینترم‌های بی‌اهمیت می‌باشند.

می‌توان عبارت ضرب حاصل جمع‌ها را هم برای تابع شکل ۱۷-۳ بدست آورد. در این حال، تنها راه برای ترکیب 0ها جایگزینی مینترم‌های بی‌اهمیت شماره 0 و 2 با مقدار 0 می‌باشد و به این ترتیب تابع متمم ساده شده بدست می‌آید:

$$F' = z' + wy'$$

با متمم‌گیری از طرفین، عبارت ساده شده به صورت ضرب حاصل جمع‌ها خواهد بود:

$$F(w, x, y, z) = z(w' + y) = \Sigma(1, 3, 5, 7, 11, 15)$$

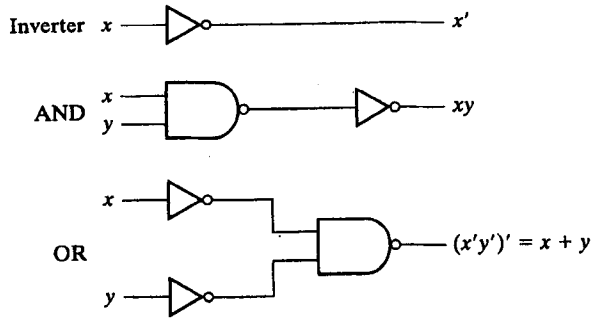
در این حال، ما مینترم‌های شماره 0 و 2 را با مقدار 0 و مینترم 5 را با 1 جایگزین کرده‌ایم.

۳-۶ پیاده‌سازی با NAND و NOR

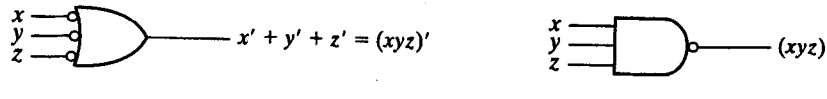
مدارهای دیجیتال اغلب به جای AND و OR با گیت‌های NAND و NOR ساخته می‌شوند. ساختن گیت‌های NAND و NOR با اجزاء الکترونیکی ساده‌تر بوده و به عنوان گیت‌های پایه در تمام خانواده‌های IC های دیجیتال به کار می‌روند. به دلیل مزیت گیت‌های NAND و NOR در طراحی مدارهای دیجیتال، قواعد و روال‌هایی برای تبدیل توابع بول بیان شده برحسب AND ، OR و NOT به نمودارهای منطقی معادل برحسب NAND و NOR بوجود آمده است.

مدارهای NAND

گیت NAND را یک گیت یونیورسال می‌گویند زیرا هر سیستم دیجیتالی را می‌توان با آن پیاده‌سازی کرد. برای این که نشان دهیم هر تابع بولی قابل پیاده‌سازی با گیت‌های NAND می‌باشد، کافی است فقط نشان دهیم که اعمال منطقی AND ، OR و NOT را می‌توان با NAND پیاده‌سازی کرد. این کار درشکل ۱۸-۳ نشان داده شده است. عمل متمم از یک گیت NAND یک ورودی که دقیقاً مثل NOT عمل می‌کند حاصل می‌گردد. عمل AND نیاز به دو گیت NAND دارد. اولی عمل NAND و دومی عمل NOT را انجام می‌دهد. عمل OR از طریق یک گیت NAND و دو NOT در هر ورودی حاصل می‌شود.



شکل ۱۸-۳. عملیات منطقی با گیت‌های NAND



Invert-OR (ب)　　AND-invert (الف)

شکل ۱۹-۳. دو سمبل گرافیکی برای گیت NAND

راهی مناسب برای پیاده‌سازی یک تابع بول با گیت‌های NAND ، بدست آوردن تابع بول ساده شده برحسب عملگرهای بولی و سپس تبدیل تابع به منطق NAND است. تبدیل یک عبارت جبری از AND ، OR و NOT به NAND به سادگی با دستکاری نمودار AND–OR به نمودار NAND انجام می‌شود.

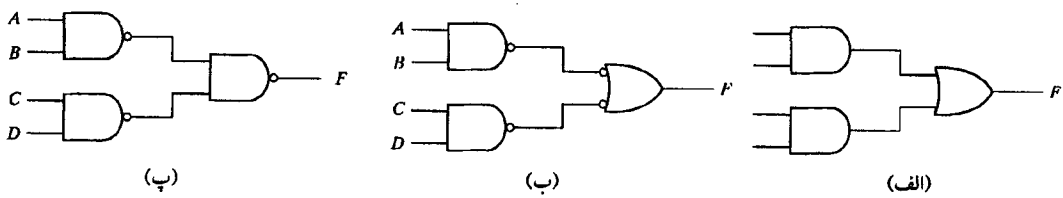
برای ساده سازی تبدیل به منطق NAND ، بهتر است سمبل گرافیکی دیگری برای گیت تعریف کنیم. دو سمبل گرافیکی معادل برای گیت NAND در شکل ۱۹-۳ دیده می‌شود. سمبل AND–invert قبلاً معرفی شد و متشکل بود از یک سمبل AND و به دنبال آن یک دایره کوچک که به آن حباب گفته شد و نقش متمم‌سازی را داشت. به همین ترتیب می‌توان گیت NAND را با سمبل گرافیکی OR با حبابی در هر ورودی نشان داد. سمبل invert–OR برای گیت NAND از تئوری دمورگان و با توجه به این قرارداد که دوایر کوچک به منزله متمم کردن هستند بدست می‌آید. دو سمبل گرافیکی فوق در طراحی و تحلیل مدارهای NAND مفیدند. وقتی هر دو سمبل در یک نمودار به کار روند گوییم مدار با علائم مخلوط نشان داده شده است.

پیاده‌سازی دو سطحی

برای پیاده‌سازی توابع بول با گیت‌های NAND ، تابع باید به فرم جمع حاصلضرب‌ها باشد. برای درک ارتباط بین عبارت جمع حاصلضرب‌ها و معادل NAND آن، به نمودارهای منطقی شکل ۲۰-۳ توجه کنید. هر سه نمودار معادل بوده و تابع زیر را پیاده می‌نمایند.

$$F = AB + CD$$

در (الف) تابع با گیت‌های AND و OR پیاده‌سازی شده است. در (ب)، گیت‌های AND با گیت‌های NAND و گیت OR نیز با یک گیت NAND که با سمبل invert–OR مشخص شده پیاده‌سازی شده است. توجه داشته باشید که یک حباب به معنی متمم و دو حباب در یک مسیر دوبار متمم‌سازی را



(پ)　　(ب)　　(الف)

شکل ۲۰-۳. سه راه پیاده‌سازی تابع F = AB + CD

نشان می‌دهند، پس می‌توانند حذف شوند. حذف حباب‌ها در گیت‌های (ب) مدار شکل (الف) را نتیجه می‌دهد. بنابراین دو نمودار یک تابع را پیاده‌سازی می‌کنند پس معادل‌اند.

در شکل ۲۰-۳ (پ)، خروجی گیت NAND با سمبل گرافیکی AND–invert ترسیم شده است. هنگام رسم نمودارهای منطقی NAND ، هر یک از دو مدار (ب) یا (پ) پذیرفته است. مدار (ب) از علائم مخلوط استفاده کرده است و رابطه مستقیم‌تری را با عبارت بول پیاده شده نشان می‌دهد. صحت پیاده‌سازی NAND در شکل ۲۰-۳ (پ) می‌تواند به صورت جبری تحقیق شود. تابعی که این شکل را پیاده کرده است به سادگی با تئوری دمورگان قابل تبدیل به جمع حاصلضرب‌هاست:

$$F = \big((AB)'(CD)'\big)' = AB + CD$$

مثال ۱۰-۳

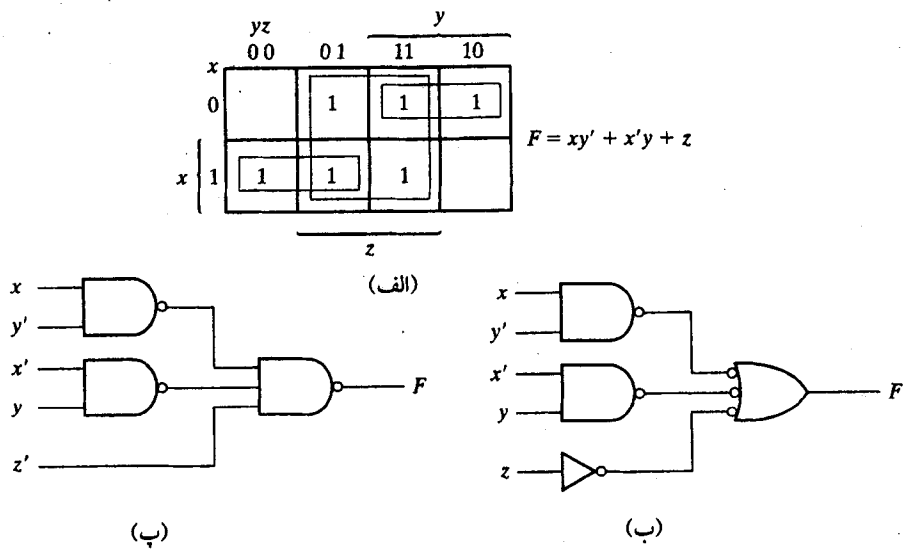
تابع بول زیر را با گیت‌های NAND پیاده کنید.

$$F(x, y, z) = (1, 2, 3, 4, 5, 7)$$

اولین قدم در تبدیل، ساده‌سازی تابع در جمع حاصلضرب‌هاست. این کار به کمک نقشه شکل ۲۱-۳ (الف) انجام شده است و تابع حاصل به صورت زیر است.

$$F = xy' + x'y + z$$

پیاده‌سازی NAND دو سطحی در شکل ۲۱-۳ (ب) به صورت علائم مخلوط دیده می‌شود. توجه کنید که ورودی z باید یک گیت NAND یک ورودی باشد تا حباب موجود در گیت سطح دوم را جبران کند. روش دیگری برای ترسیم نمودار منطقی در شکل ۲۱-۳ (پ) نشان داده شده است. در اینجا تمام



شکل ۲۱-۳. حل مثال ۱۰-۳

گیت‌های NAND با سمبل یکسان ترسیم شده‌اند. وارونگر با ورودی z حذف شده است ولی متغیر ورودی متمم شده و با z' نشان داده شده است. ❑

روالی که در مثال قبل توصیف شد بیان می‌دارد که یک تابع بول می‌تواند با دو سطح (یا دو طبقه) گیت NAND پیاده‌سازی شود. روال تهیه نمودار منطقی NAND از تابع بول به قرار زیر است:

۱- تابع را ساده کرده آن را به فرم جمع حاصل ضرب‌ها بنویسید.
۲- برای هر جمله ضرب موجود در تابع که حداقل دو لیترال دارد یک گیت NAND بکشید. ورودی به هر یک گیت NAND لیترال‌های جمله‌اند. این مجموعه، گیت‌های سطح اول را تشکیل می‌دهد.
۳- در سطح دوم، یک گیت NAND با ورودی‌هایی که از خروجی‌های سطح اول می‌آیند بکشید. از سمبل گرافیکی AND–invert یا invert–OR استفاده نمایید.
۴- یک جمله با یک لیترال نیاز به یک وارونگر در اولین سطح دارد. با این وجود، اگر تک لیترال متمم شده است می‌توان آن را مستقیماً به گیت NAND سطح دوم وصل کرد.

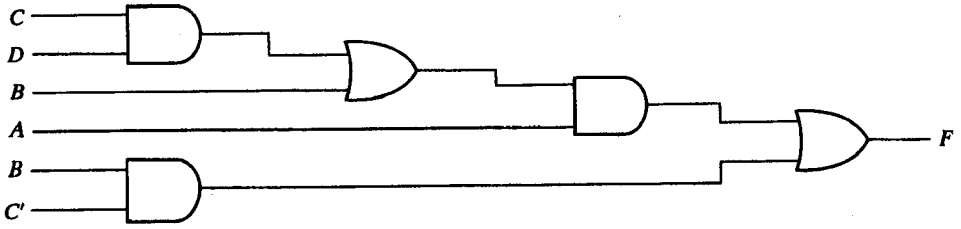
مدارهای NAND چند سطحی

فرم استاندارد بیان توابع بول، پیاده‌سازی دو سطحی (طبقه) را نتیجه می‌دهد. مواردی وجود دارد که طراحی سیستم‌های دیجیتال یک ساختار گیتی با سه یا چند طبقه را نتیجه می‌دهد. رایج‌ترین روش طراحی مدارهای چند طبقه بیان تابع بول برحسب عملیات AND ، OR و NOT می‌باشد. سپس می‌توان تابع را با گیت‌های AND و OR پیاده‌سازی کرد. آنگاه در صورت لزوم تمام مدار را می‌توان به NAND تبدیل نمود. به عنوان مثال تابع بول زیر را ملاحظه کنید:

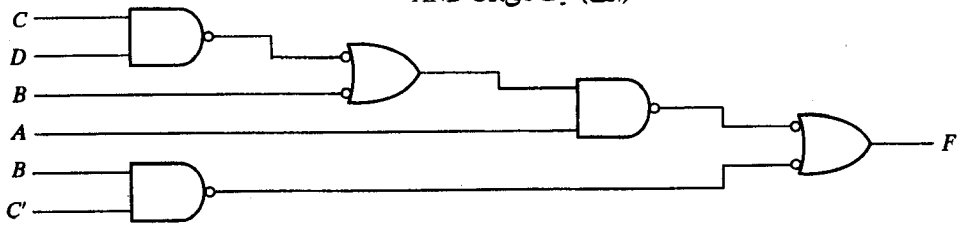
$$F = A(CD + B) + BC'$$

گرچه می‌توان پرانتزها را حذف کرده و عبارت را به صورت جمع حاصلضرب استاندارد در آورد، ولی آن را به عنوان یک مدار چند طبقه مورد بررسی قرار می‌دهیم. پیاده‌سازی AND–OR برای آن در شکل ۲۲-۳ (الف) نشان داده شده است. در مدار، چهار طبقه گیت دیده می‌شود. اولین طبقه دو گیت AND دارد. دومین طبقه یک گیت OR و به دنبال آن یک AND در طبقه سوم آمده و در طبقه چهارم هم یک OR ملاحظه می‌شود. با استفاده از علائم مخلوط، می‌توان یک نمودار منطقی با الگویی از سطوح متناوب AND و OR را به سادگی به مدار NAND تبدیل کرد. این تبدیل در شکل ۲۲-۳ (ب) دیده می‌شود. روال این است که هر گیت AND را به سمبل AND–invert و هر گیت OR را به invert–OR تبدیل کنیم. مدار NAND حاصل، عملکرد یکسانی با نمودار AND–OR دارد به شرطی که در هر مسیر دو حباب وجود داشته باشد. حباب مربوط به ورودی B موجب می‌شود تا یک متمم اضافی صورت گیرد که باید آن را با متغیر ورودی مذکور به لیترال B' جبران کرد.

روال کلی نمودار چند طبقه AND–OR به نمودار تمام NAND با استفاده از علائم مخلوط به شرح زیر است:



(الف) گیت‌های AND-OR



(ب) گیت‌های NAND

شکل ۲۲-۳. پیاده‌سازی $F = A(CD + B) + BC'$

۱- همه گیت‌های AND را با استفاده از سمبل‌های گرافیکی invert–AND به گیت NAND تبدیل کنید.
۲- همه گیت‌های OR را با سمبل‌های گرافیکی invert–OR به NAND تبدیل نمایید.
۳- همه حباب‌ها را در نمودار چک کنید. برای هر حبابی که در یک مسیر جبران نشده است یک وارونگر (گیت NAND یک ورودی) وارد کنید و یا لیترال ورودی را متمم نمایید.

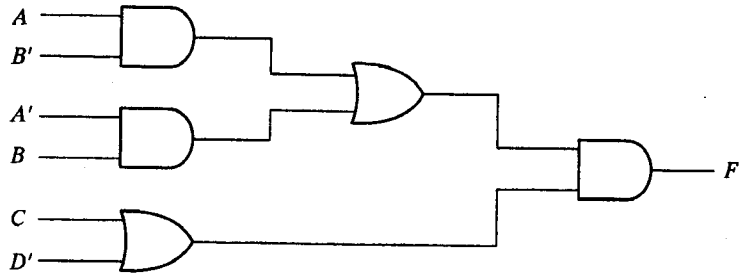
به عنوان مثالی دیگر تابع بول چند سطحی زیر را ملاحظه کنید.

$$F = (AB' + A'B)(C + D')$$

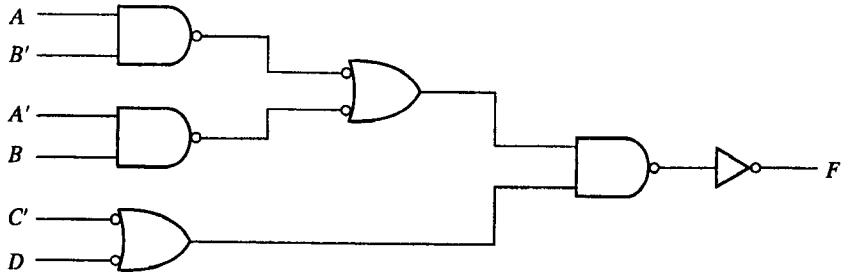
پیاده‌سازی AND–OR تابع در شکل ۲۳-۳ (الف) ملاحظه می‌شود که در آن سه طبقه گیت به کار رفته است. در بخش (ب) نمودار فرم تبدیل شده به NAND آن با علائم مخلوط دیده می‌شود. دو حباب اضافی مربوط به ورودی‌های C و D' موجب متمم شدن آنها به C' و D می‌گردد. حباب موجود در گیت NAND خروجی، مقدار خروجی را متمم می‌کند بنابراین برای بدست آوردن مقدار اصلی تابع مجبوریم یک گیت وارونگر در خروجی به کار ببریم.

پیاده‌سازی NOR

عمل NOR دوگان NAND است. بنابراین تمام روال‌ها و قوانین منطق NOR دوگان روال‌های متناظر و قوانین حاصل در منطق NAND هستند. NOR ، گیت یونیورسال دیگری است که برای پیاده‌سازی هر تابع بول به کار می‌رود. پیاده‌سازی اعمال AND ، OR و NOT با گیت‌های NOR در شکل ۲۴-۳ ملاحظه می‌گردد. عمل متمم، با گیت NOR یک ورودی حاصل شده و عیناً مثل وارونگر

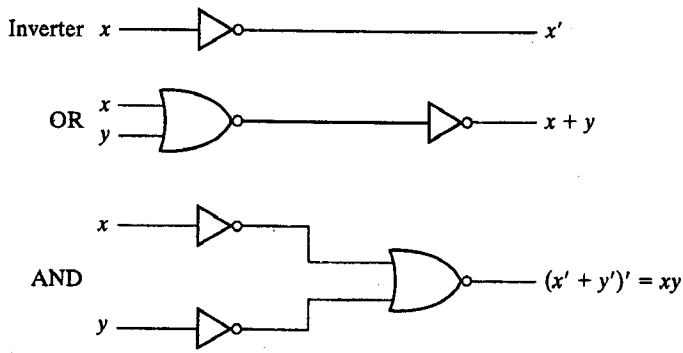


(الف) گیت‌های AND-OR



(ب) گیت‌های NAND

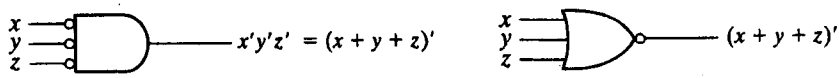
شکل ۲۳-۳. پیاده‌سازی $F = (AB' + A'B)(C + D')$



شکل ۲۴-۳. عملیات منطقی با گیت‌های NOR

عمل می‌کند. عمل OR نیاز به دو گیت NOR و عمل AND از یک گیت NOR که در هر ورودی‌اش یک وارونگر دارد حاصل می‌شود.

دو سمبل گرافیکی برای علائم مخلوط در شکل ۲۵-۳ دیده می‌شود. سمبل OR–invert عمل NOR را با یک OR و به دنبال آن یک متمم تعریف می‌کند. هر دو سمبل عمل NOR یکسانی را به نمایش می‌گذارند و از نظر منطقی با توجه به تئوری دمورگان یکی هستند.



Invert-AND (ب)　　OR-invert (الف)

شکل ۲۵-۳. دو سمبل گرافیکی برای گیت NOR

پیاده‌سازی دو طبقه با گیت‌های NOR لازم می‌دارد تا تابع به صورت ضرب حاصل جمع‌ها ساده شود. به خاطر دارید که عبارت ضرب حاصل جمع‌های ساده شده از نقشه با ترکیب 0ها و متمم کردن آنها بدست می‌آید. عبارت ضرب حاصل جمع با گیت‌های OR در اولین سطح که جملات جمع را تولید می‌کنند پیاده‌سازی می‌شود. به دنبال آنها گیت AND برای تولید ضرب دیده می‌شود. تبدیل نمودار OR–AND به NOR با تبدیل گیت‌های OR به گیت NOR با استفاده از سمبل گرافیکی invert–AND صورت می‌گیرد. یک جمله تک لیترال که به یک گیت سطح دوم برود باید متمم گردد. شکل ۲۶-۳ پیاده‌سازی یک تابع را به فرم ضرب حاصل جمع‌ها نشان می‌دهد:

$$F = (A + B)(C + D)E$$

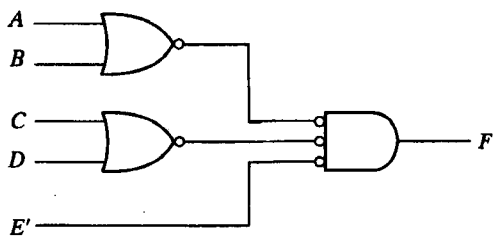
الگوی OR–AND را به سادگی با حذف حباب‌ها در طول هر مسیر می‌توان شناسایی کرد. متغیر E برای جبران سومین حباب در ورودی گیت سطح دوم متمم شده است.

روال تبدیل یک نمودار AND–OR چند سطح به نمودار تمام NOR مشابه آنچه برای گیت‌های NAND دیدیم، می‌باشد. در حالت NOR، باید هر گیت OR را به یک سمبل OR–invert و هر گیت AND را به یک گیت invert–AND تبدیل نماییم. هر حبابی که به وسیلهٔ حباب دیگر در همان مسیرجبران نشود نیاز به یک وارونگر یا متمم شدن لیترال ورودی دارد.

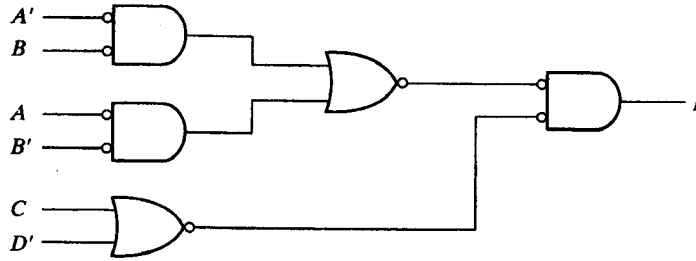
تبدیل نمودار AND-OR شکل ۲۳-۳(الف) به یک نمودار NOR در شکل ۲۷-۳ نشان داده شده است. تابع بول برای این مدار به شکل زیر است

$$F = (AB' + A'B)(C + D')$$

نمودار معادل AND-OR را می‌توان با حذف حباب‌ها تشخیص داد. برای جبران حباب‌ها در چهار ورودی، لازم است لیترال‌های ورودی مربوطه متمم شوند



شکل ۲۶-۳. پیاده‌سازی F = (A + B)(C + D) E



شکل ۲۷-۳. پیاده‌سازی $F = (AB' + A'B)(C + D')$ با گیت‌های NOR

۳-۷ دیگر پیاده‌سازی‌های دو سطحی

گیت‌هایی که بیشتر در مدارهای مجتمع یافت می‌شوند از نوع NAND و NOR هستند. به همین دلیل، پیاده‌سازی‌های منطقی NAND و NOR از دیدگاه عملی مهمتراند. در بعضی از گیت‌های NOR یا NAND (و نه همه آنها) این امکان وجود دارد تا با اتصال سیم بین خروجی‌های دو گیت، یک تابع منطقی مشخص تولید کرد. این منطق را **منطق سیم‌بندی** یا **سیمی** می‌نامند. مثلاً گیت‌های TTL NAND کلکتور باز وقتی به هم گره زده شوند تولید منطق AND سیمی (Wired–AND) را می‌نمایند. (گیت TTL کلکتور باز در فصل ۱۰ شکل ۱۱-۱۰ نشان داده شده است.) منطق AND سیمی که با گیت‌های NAND انجام می‌شود در شکل ۲۸-۳ (الف) ترسیم شده است. گیت AND با ترسیم خطوط تا مرکز گیت نشان داده شده تا بدین ترتیب از گیت‌های AND معمولی تفکیک شود. گیت AND سیمی (یا اتصالی) یک گیت فیزیکی نیست، بلکه فقط سمبلی برای نمایش تابع حاصل از اتصال سیمی است. تابع منطقی پیاده شده با مدار شکل ۲۸-۳ (الف) برابر زیر است.

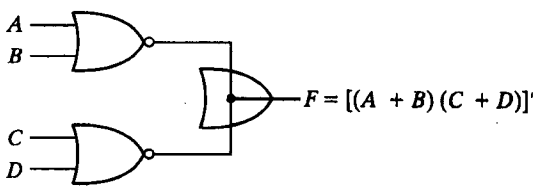
$$F = (AB)' \cdot (CD)' = (AB + CD)'$$

و به آن تابع AND–OR–INVERT می‌گویند.

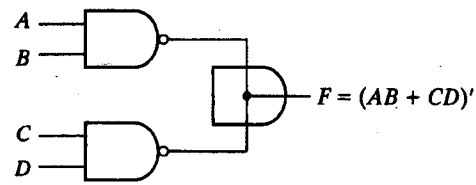
به طور مشابه خروجی NOR گیت‌های ECL برای اجرای یک تابع OR سیمی به هم گره زده می‌شوند. تابع منطقی پیاده‌سازی شده با مدار شکل ۲۸-۳ (ب) چنین است:

$$F = (A + B)' + (C + D)' = [(A + B)(C + D)]'$$

و به آن تابع OR–AND–INVERT می‌گویند.



(ب) OR سیمی در گیت‌های ECL



(الف) AND سیمی در گیت‌های TTL NAND کلکتور باز

شکل ۲۸-۳. منطق سیمی

یک گیت منطقی سیمی تولید گیت سطح دوم فیزیکی را نمی‌کند زیرا تنها یک اتصال سیمی است. با این وجود به هنگام بحث، مدارهای شکل ۲۸-۳ را به عنوان پیاده‌سازی‌های دو سطحی یا دو طبقه در نظر می‌گیریم. اولین طبقه متشکل از گیت‌های NAND (یا NOR) و دومین طبقه تنها یک گیت AND یا OR دارد. در بحث‌های بعدی اتصال سیمی در سمبل گرافیکی حذف می‌گردد.

فرم‌های مفید گیت‌ها

از نقطه نظر تئوری یافتن ترکیب‌های دو سطحی ممکن گیت‌ها آموزنده است. در اینجا چهار نوع گیت AND، OR، NAND و NOR را بررسی می‌کنیم. اگر یکی از انواع گیت‌ها را به سطح اول و نوعی دیگر را به سطح دوم نسبت دهیم، در می‌یابیم که 16 ترکیب ممکن از فرم دو سطحی وجود دارد. می‌توان در هر دو سطح یک نوع گیت مانند NAND–NAND را هم به کار برد. هشت ترکیب از آنها، فرم زاید خوانده می‌شوند زیرا در حقیقت یک عمل ساده منطقی را انجام می‌دهند. این نکته در مواردی که هر دو سطح اول و دوم از گیت‌های AND تشکیل شده‌اند به خوبی مشهود است. خروجی مدار صرفاً تابع AND از همه متغیرهای ورودی است. هشت فرم مفید دیگر نوعی پیاده‌سازی جمع حاصلضرب‌ها و یا ضرب حاصل‌جمع‌ها را تولید می‌کند. این هشت فرم مفید عبارتند از:

AND-OR	OR-AND
NAND-NAND	NOR-NOR
NOR-OR	NAND-AND
OR-NAND	AND-NOR

اولین گیت در هر یک از فرم‌های فوق سطح اول پیاده‌سازی را تشکیل می‌دهد. دومین گیت در لیست تنها گیتی است که در سطح دوم قرارگرفته است. توجه کنید هر دو فرمی که در یک سطر آمده‌اند دوگان یکدیگرند.

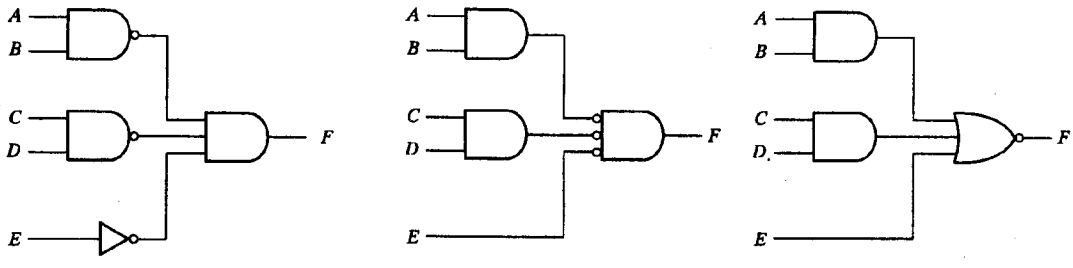
فرم‌های AND–OR و OR–AND، فرم‌های دو سطح اصلی بحث شده در بخش ۴-۳ می‌باشند. فرم‌های NAND–NAND و NOR–NOR در بخش ۶-۳ ارائه شدند. چهار فرم باقیمانده نیز در این بخش بررسی می‌شوند.

پیاده‌سازی AND–OR–INVERT

دو فرم NAND–AND و AND–NOR معادل یکدیگرند و می‌توان آنها را همزمان شرح داد. هر دو تابع طبق شکل ۲۹-۳، عمل AND–OR–INVERT را اجرا می‌کنند. فرم AND–NOR، همان عمل AND–OR با یک وارونگر در خروجی است. این فرم تابع زیر را پیاده‌سازی می‌کند.

$$F = (AB + CD + E)'$$

با استفاده از سمبل گرافیکی معادل دیگری برای گیت NOR، نمودار شکل ۲۹-۳ (ب) بدست می‌آید. توجه کنید که تک متغیر E متمم نشده است زیرا تنها تغییر، در سمبل گرافیکی گیت NOR صورت گرفته است. اکنون حباب‌ها را از پایانه‌های ورودی گیت سطح دوم به پایانه‌های خروجی گیت‌های سطح



شکل ۲۹-۳. مدارهای AND–OR–INVERT: $F = (AB + CD + E)'$

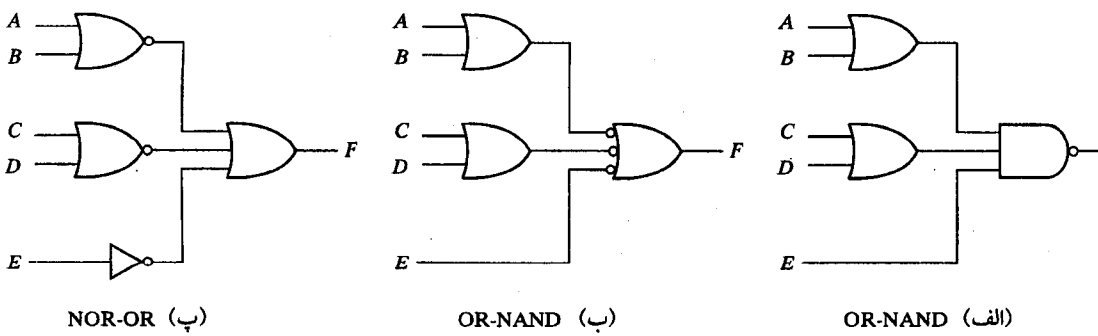
اول انتقال می‌دهیم. برای جبران هر حباب یک وارونگر در ازاء هر متغیر لازم است. به همین ترتیب می‌توان وارونگر را حذف کرد به شرطی که E متمم شود. مدار شکل ۲۹-۳ (پ)، فرم NAND–AND است و برای پیاده‌سازی تابع AND–OR–INVERT در شکل ۲۸-۳ نشان داده شده است.

پیاده‌سازی AND–OR نیاز به یک عبارت جمع حاصلضرب‌ها دارد. پیاده‌سازی AND–OR–INVERT مشابه آن است، به جز این که یک وارونگر اضافی دارد. بنابراین اگر متمم تابع به صورت جمع حاصلضرب‌ها ساده شود (با ترکیب 0ها در نقشه)، می‌توان F′ را با بخش AND–OR تابع پیاده‌سازی کرد. وقتی که F′ از داخل وارونگر عبور کند، خروجی F تابع را تولید می‌نماید. مثالی در مورد پیاده‌سازی AND–OR–INVERT به دنبال آمده است.

پیاده‌سازی OR–AND–INVERT

فرم‌های NAND–OR و NOR–OR تابع OR–AND–INVERT را اجرا می‌کنند. این فرم‌ها در شکل ۳۰-۳ نشان داده شده‌اند. فرم NAND–OR، فرم OR–AND را تداعی می‌کند. به جز این که در خروجی گیت NAND، عمل متمم با حباب انجام می‌شود. در این شکل تابع زیر پیاده‌سازی شده است.

$$F = [(A + B)(C + D)E]'$$



شکل ۳۰-۳. مدارهای OR–AND–INVERT: $F = [(A + B)(C + D)E]'$

جدول ۳-۳. پیاده‌سازی با دیگر فرم‌های دو طبقه

دریافت خروجی از	ساده کردن F' در	پیاده سازی تابع	معادل فرم مفید (b)*	معادل فرم مفید (a)
F	مجموع حاصلضرب‌ها بوسیله ترکیب 0 ها در نقشه	AND-OR-INVERT	NAND-AND	AND-NOR
F	ضرب حاصلجمع‌ها با ترکیب 1 ها در نقشه و سپس مکمل سازی	OR-AND-INVERT	NOR-OR	OR-NAND

* فرم (b) برای یک جمله تک لیترال به یک وارونگر نیاز دارد.

با استفاده از فرم دیگر گیت NAND نمودار شکل ۳۰-۳ (ب) بدست می‌آید. مدار در (پ) با انتقال دوایر کوچک از ورودی‌های گیت سطح دوم به خروجی‌های گیت‌های سطح اول حاصل می‌گردد. مدار شکل ۳۰-۳ (پ) یک فرم NOR-OR است و قبلاً برای پیاده‌سازی تابع OR-AND-INVERT در شکل ۲۸-۳ نشان داده شد.

پیاده‌سازی OR-AND-INVERT به عبارتی به فرم ضرب حاصل جمع‌ها احتیاج دارد. اگر متمم تابع به صورت ضرب حاصل جمع‌ها ساده شود می‌توانیم F' را با بخش OR-AND تابع پیاده‌سازی کنیم. پس از عبور F' از بخش INVERT، متمم F' یعنی F در خروجی حاصل خواهد شد.

خلاصه جدول وار و مثال

جدول ۳-۳ روش‌های پیاده‌سازی یک تابع بول را در هر یک از چهار فرم دو سطحی خلاصه کرده است. بدلیل بخش INVERT در هر حالت، استفاده از ساده سازی F' تابع مناسب‌تر است. وقتی که F' به یکی از این فرم‌ها پیاده سازی شود، متمم تابع را در فرم AND-OR یا OR-AND پیاده سازی کرده‌ایم. چهار فرم دو سطحی این تابع را معکوس نموده و خروجی متمم F' خواهد بود. این خروجی نرمال یعنی F است.

مثال ۱۱-۳

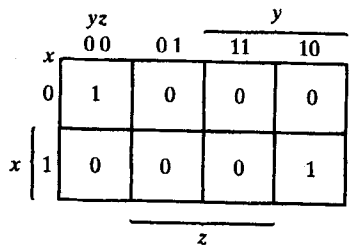
تابع شکل ۳۱-۳ (الف) را به فرم دو سطحی لیست شده در جدول ۳-۳ پیاده‌سازی کنید.

متمم تابع با ترکیب 0ها در نقشه به فرم جمع حاصلضرب‌های ساده شده بدست می‌آید.

$$F' = x'y + xy' + z$$

خروجی نرمال این تابع می‌تواند به صورت زیر بیان شود.

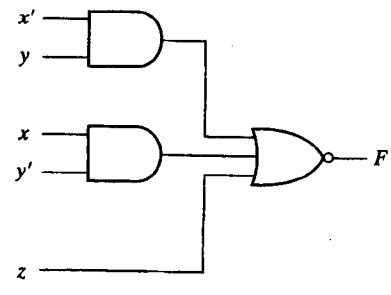
$$F = (x'y + xy' + z)'$$



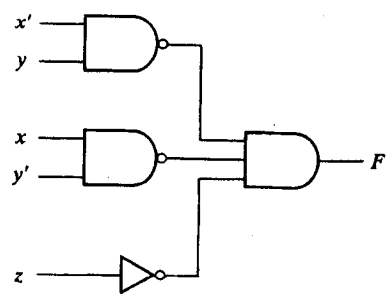
$F = x'y'z' + xyz'$

$F' = x'y + xy' + z$

(الف) ساده سازی نقشه در جمع حاصلضرب ها

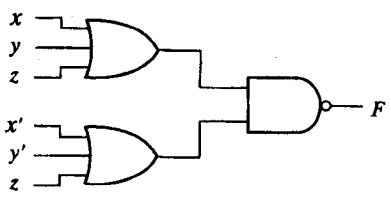


AND-NOR

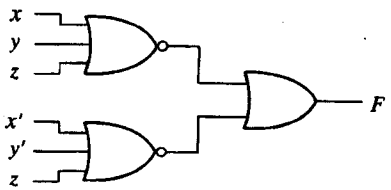


NAND-AND

$F = (x'y + xy' + z)'$ (ب)



OR-NAND



NOR-OR

$F = [(x + y + z)(x' + y' + z)]'$ (پ)

شکل ۳۱-۳. دیگر پیاده‌سازی‌های دو طبقه

که به فرم AND–OR–INVERT است. پیاده‌سازی‌های AND–NOR و NAND–AND در شکل ۳۱-۳ (ب) دیده می‌شوند. توجه کنید که در پیاده‌سازی NAND–AND به گیت NAND یک ورودی یا گیت وارونگر نیاز است، ولی در حالت AND–NOR به آن نیازی نیست. اگر در عوض z از zé استفاده کنیم به وارونگر احتیاجی نیست.

فرم‌های OR–AND–INVERT نیاز به عملیات ساده شده‌ای از متمم تابع به فرم ضرب حاصل جمع‌ها دارد. برای تهیه این عبارت، ابتدا 1ها را در نقشه ترکیب می‌کنیم.

$$F = x'y'z' + xyz'$$

آنگاه متمم تابع را بدست می‌آوریم.

$$F' = (x + y + z)(x' + y' + z)$$

خروجی نرمال F اکنون به فرم زیر نوشته می‌شود.

$$F = [(x + y + z)(x' + y' + z)]'$$

که به فرم OR–AND–INVERT بیان شده است. با استفاده از این عبارت تابع را می‌توانیم به فرم OR–NAND یا NOR–OR نیز پیاده‌کنیم، شکل ۳۱-۳(پ).

❑

۳-۸ تابع OR انحصاری

OR انحصاری (XOR) که با علامت ⊕ نشان داده می‌شود یک عملگر منطقی است که تابع بولی زیر را اجرا می‌نماید:

$$x \oplus y = xy' + x'y$$

این تابع هنگامی برابر 1 است که فقط x یا y برابر 1 باشند، ولی هر دو آنها به طور همزمان 1 نباشند. NOR انحصاری (XNOR) که به آن هم‌ارزی هم می‌گویند عمل زیر را اجرا می‌نماید.

$$(x \oplus y)' = xy + x'y'$$

هنگامی این تابع برابر 1 است که هر دو متغیر x و y به طور همزمان برابر 1 یا برابر 0 باشند. به کمک جدول درستی یا دستکاری جبری می‌توان نشان داد که NOR انحصاری متمم OR انحصاری است:

$$(x \oplus y)' = (xy' + x'y)' = (x' + y)(x + y') = xy + x'y'$$

روابط زیر در مورد OR انحصاری معتبرند

$$x \oplus 0 = x$$
$$x \oplus 1 = x'$$
$$x \oplus x = 0$$
$$x \oplus x' = 1$$
$$x \oplus y' = x' \oplus y = (x \oplus y)'$$

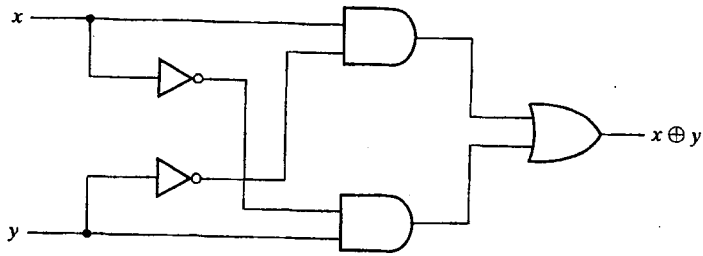
هر یک از این روابط می‌تواند با به کارگیری جدول درستی و جایگزینی ⊕ با عبارت بولی هم‌ارزی ثابت شود. و نیز می‌توان نشان داد که عمل OR انحصاری خاصیت جابجایی و شرکت‌پذیری را دارد. یعنی:

$$A \oplus B = B \oplus A$$

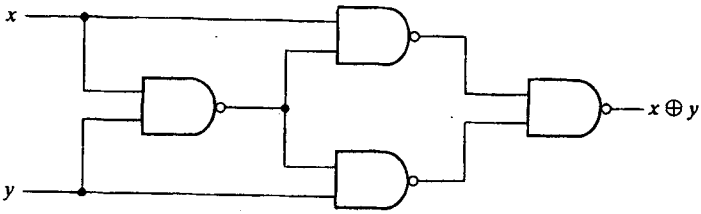
و

$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

این بدان معنی است که دو ورودی گیت OR انحصاری بدون تأثیر بر عمل قابل تعویض‌اند. و نیز به این معنی است که یک عمل OR انحصاری سه متغیره را نیز می‌توانیم ارزیابی کنیم و به همین دلیل سه متغیر یا بیشتر را بدون پرانتز بیان می‌نماییم. با این وجود گیت‌های OR انحصاری با ورودی‌های متعدد مشکل ساخت سخت‌افزاری را دارند. در واقع، حتی تابع دو ورودی آن هم با انواع گیت‌های دیگر ساخته



(الف) با گیت‌های AND-OR-NOT



(ب) با گیت‌های NAND

شکل ۳۲-۳. پیاده‌سازی XOR

می‌شود. یک تابع OR انحصاری دو ورودی که با گیت‌های معمولی AND، OR و NOT ساخته شده در شکل ۳۲-۳ (الف) دیده می‌شود. شکل ۳۲-۳ (ب) پیاده‌سازی OR انحصاری را با چهار گیت NAND نشان می‌دهد. گیت NAND اول عمل $(xy)' = (x' + y')$ را اجرا می‌کند. مدارهای NAND دو طبقه دیگر جمع حاصلضرب ورودی‌ها را تهیه می‌نماید:

$$(x' + y')x + (x' + y')y = xy' + x'y = x \oplus y$$

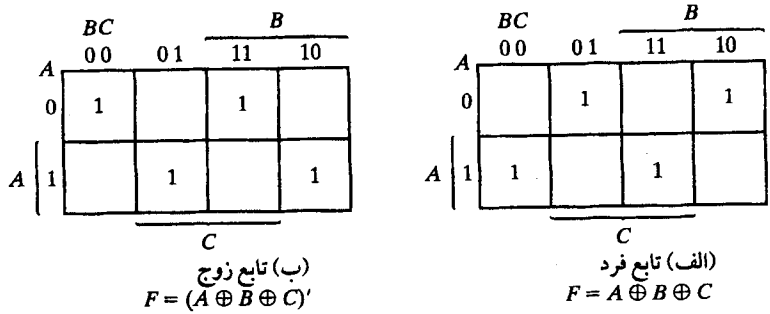
تنها توابع بولی محدودی برحسب OR انحصاری بیان می‌شوند. با این وجود این تابع به کرّات ضمن طراحی سیستم‌های دیجیتال به کار گرفته می‌شود. خصوصاً در عملیات حسابی و خطایابی و تصحیح خطا بسیار مفید است.

تابع فرد

عملگر OR انحصاری با سه متغیر یا بیشتر را می‌توان با جایگزینی سمبل $\oplus$ با عبارت بولی معادلش به یک تابع بولی معمولی تبدیل کرد. بخصوص، حالت سه متغیره را می‌توان به یک عبارت بولی مطابق زیر تبدیل نمود.

$$\begin{aligned} A \oplus B \oplus C &= (AB' + A'B)C' + (AB + A'B')C \\ &= AB'C' + A'BC' + ABC + A'B'C \\ &= \Sigma(1, 2, 4, 7) \end{aligned}$$

عبارت بول بوضوح نشان می‌دهد که تابع OR انحصاری سه متغیره برابر با 1 است به شرطی که فقط



شکل ۳۳-۳. نقشه تابع XOR سه متغیر

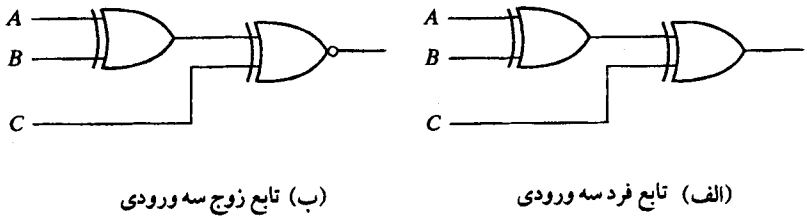
یک متغیر 1 باشد و یا هر سه متغیر برابر 1 باشند. برخلاف حالت دو متغیره، که فقط یک متغیر باید برابر 1 می‌بود، در حالت سه یا چند متغیر، نیاز این است که تعداد فردی از متغیرها برابر 1 باشند. در نتیجه عمل XOR چند متغیره را تابع فرد می‌خوانند.

تابع بول حاصل از عمل XOR سه متغیره به صورت جمع چهار مینترم است که مقادیر عددی آنها 001، 010، 100 و 111 می‌باشد. هر یک از این اعداد دودویی تعداد فردی 1 دارند. چهار مینترم دیگری که در تابع لحاظ نشده‌اند 000، 011، 101 و 110 بوده و تعداد زوجی 1 در مقدار دودویی آنها وجود دارد. به طور کلی تابع XOR با n متغیر یک تابع فرد است که به صورت جمع $2^n/2$ مینترم که مقادیر عددی آنها تعداد فردی 1 دارد بیان می‌شود.

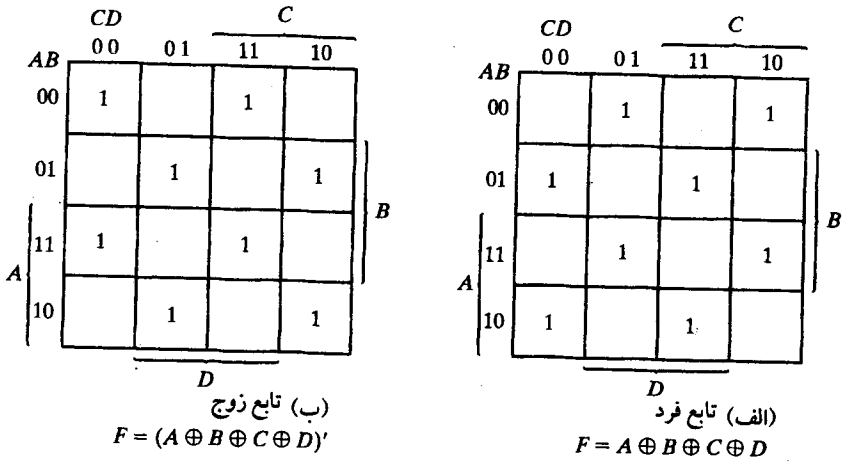
تعریف یک تابع فرد را با ترسیم آن در یک نقشه شفاف‌تر می‌کنیم. شکل ۳۳-۳ (الف) نقشه را برای تابع XOR سه متغیره نشان می‌دهد. چهار مینترم تابع یک مربع در میان با هم فاصله دارند. تابع فرد از چهار مینترمی که مقادیر دودویی‌اش تعداد فردی 1 دارند شناسایی می‌شود. متمم یک تابع فرد، یک تابع زوج است. طبق شکل ۳۳-۳ (ب)، تابع زوج سه متغیره هنگامی 1 است که تعداد زوجی متغیر در یک مینترم، 1 باشد (از جمله مینترمی که هیچ یک از متغیرها در آن 1 نیست).

تابع فرد 3 ورودی را می‌توان با گیت دو ورودی هم طبق شکل ۳۴-۳ (الف) پیاده‌سازی کرد. متمم یک تابع فرد با جایگزینی گیت خروجی با یک گیت XNOR طبق شکل ۳۴-۳ (ب) بدست می‌آید.

اکنون عملکرد XOR چهار متغیره را ملاحظه کنید. با دستکاری جبری، می‌توانیم جمع مینترم‌های



شکل ۳۴-۳. نمودار منطقی توابع فرد و زوج



شکل ۳۵-۳. نقشه برای تابع XOR چهار متغیره

این تابع را بدست آوریم.

$$\begin{aligned} A \oplus B \oplus C \oplus D &= (AB' + A'B) \oplus (CD' + C'D) \\ &= (AB' + A'B)(CD + C'D') + (AB + A'B')(CD' + C'D) \\ &= \Sigma(1, 2, 4, 7, 8, 11, 13, 14) \end{aligned}$$

برای تابع بول چهار متغیره 16 مینترم وجود دارد. نیمی از مینترم‌ها دارای تعداد فردی 1 در مقادیر عددی خود هستند؛ نیمه دیگر دارای تعداد زوجی 1 در مینترم می‌باشند. هنگام ترسیم تابع در نقشه، مقدار عدد دودویی هر مینترم از اعداد سطر و ستون مربعی که مینترم را نمایش می‌دهد بدست می‌آید. نقشه شکل ۳۵-۳ (الف) مربوط به تابع XOR چهار متغیره است. این یک تابع فرد است زیرا مقادیر دودویی همه مینترم‌ها تعداد فردی 1 دارند. متمم یک تابع فرد هم یک تابع زوج است. طبق شکل ۳۵-۳ (ب) تابع زوج چهار متغیره هنگامی 1 است که تعداد زوجی از متغیرها در مینترم برابر 1 باشد.

تولید و چک توازن

توابع XOR در سیستم‌هایی که به کدهای عیب‌یاب و تصحیح‌کننده خطا نیاز دارند بسیار مفیدند. همانطور که در بخش ۷-۱ ملاحظه شد، یک بیت توازن به منظور تشخیص خطا در حین انتقال اطلاعات دودویی به آن اضافه می‌شود. بیت توازن، بیتی اضافی است که با پیام دودویی همراه می‌شود تا تعداد 1ها را زوج یا فرد کند. پیام، از جمله بیت توازن، ارسال و سپس در مقصد برای تشخیص خطا چک می‌شود. اگر توازن چک شده با آنچه ارسال شده است تطابق نداشت. یک خطا اعلام می‌گردد. مداری که بیت توازن را در فرستنده تولید می‌کند، مولد توازن نامیده می‌شود. مداری که توازن را در سمت گیرنده چک می‌کند چک کننده توازن خوانده می‌شود.

جدول ۴-۳. جدول درستی مولد توازن زوج

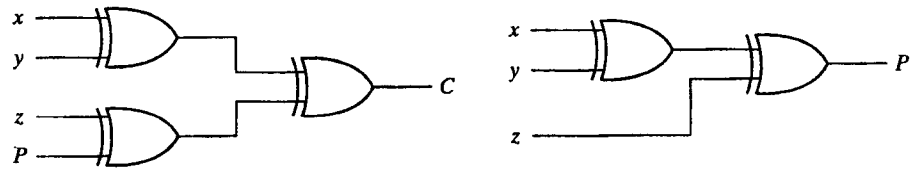
پیام سه بیتی			بیت توازن
x	**y**	**z**	**P**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

به عنوان مثال، فرض کنید بخواهیم یک پیام سه بیتی را همراه با یک بیت توازن زوج ارسال کنیم. جدول ۴-۳، جدول درستی را برای مولد توازن نشان می‌دهد. سه بیت x و y و z که پیام را تشکیل می‌دهند ورودی به مدار هستند. بیت توازن P، خروجی است. برای توازن زوج، بیت P باید طوری باشد که تعداد کل 1 ها را زوج کند (از جمله P). از جدول درستی می‌بینیم که P یک تابع فرد را تشکیل می‌دهد زیرا برای مینترم‌هایی که تعداد فردی 1 دارند باید برابر 1 شود. بنابراین P به صورت یک تابع XOR سه متغیره بیان می‌شود.

$$P = x \oplus y \oplus z$$

نمودار منطقی مولد توازن در شکل ۳۶-۳ (الف) ملاحظه می‌شود.

سه بیت پیام، همراه با بیت توازن به مقصد ارسال می‌شوند و در آنجا به مدار چک کننده توازن برای چک کردن خطای محتمل به هنگام ارسال، وارد می‌گردند. چون اطلاعات با توازن زوج ارسال شده است، چهار بیت دریافتی باید تعداد زوجی 1 داشته باشد. خطا در حین انتقال هنگامی رخ می‌دهد که چهار بیت دریافتی تعداد فردی 1 دارد، و این به معنی رخداد خطا در حین انتقال است. خروجی چک کننده توازن که با C مشخص شده است به هنگام رخداد خطا برابر 1 می‌شود. یعنی اگر چهار بیت دریافتی تعداد فردی 1 داشته باشد خطا رخ داده است. جدول ۵-۳ جدول درستی برای چک کننده توازن زوج است. با توجه به آن ملاحظه می‌شود که تابع C متشکل از هشت مینترم با مقادیر دودویی دارای تعداد فردی 1 است. این مطالب مربوط به نقشه ۳۵-۳ (الف) است که تابع فرد را نشان می‌دهد. می‌توان



(الف) مولد توازن زوج 3 بیتی (ب) چک کننده توازن زوج 4 بیتی

شکل ۳۶-۳. نمودار منطقی مولد و چک کننده توازن

جدول ۵-۳. جدول درستی چک کننده توازن زوج

چهار بیت دریافتی				چک خطای توازن
x	y	z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

چک کننده توازن را با گیت‌های XOR پیاده‌سازی کرد:

$$C = x \oplus y \oplus z \oplus P$$

نمودار منطقی چک کننده توازن در شکل ۳۶-۳ (ب) ملاحظه می‌گردد.

لازم به تذکر است که مولد توازن را با مدار شکل ۳۶-۳ (ب) نیز می‌توان تولید کرد به شرطی که ورودی P به منطق 0 متصل گردد و خروجی نیز با P نام‌گذاری شود. دلیل این است که $z \oplus 0 = z$ بوده و موجب می‌شود تا z از گیت بدون تغییر عبور کند. مزیت این است که برای هر دو مدار تولید و چک کننده توازن از یک مدار مشابه می‌توان استفاده کرد.

با توجه به مثال قبل واضح است که تابع مولد توازن و نیز چک کننده دارای نیمی از کل مینترم‌ها هستند و مقادیر عددی آنها تعداد زوج یا فردی 1 دارند. در نتیجه می‌توان آنها را با گیت‌های XOR پیاده‌سازی کرد. تابعی با تعداد زوجی 1 متمم تابع فرد است. این تابع با XOR پیاده‌سازی می‌شود ولی گیت واقع در خروجی باید XNOR باشد تا متمم لازم را تولید نماید.

۳-۹ زبان توصیف سخت‌افزاری (HDL)

زبان توصیف سخت‌افزاری، زبانی است که سخت‌افزار سیستم‌های دیجیتال را به فرم متنی توصیف می‌نماید. در واقع این زبان، یک زبان برنامه‌نویسی است، ولی خصوصاً حول توصیف ساختارهای سخت‌افزاری و رفتار آنها بنا نهاده شده است. می‌توان از آن برای نمایش نمودارهای منطقی، عبارات بولی و دیگر مدارهای دیجیتال پیچیده استفاده کرد. HDL به عنوان یک زبان مستندسازی برای نمایش و

مستند کردن سیستم‌های دیجیتال به کار می‌رود به نحوی که قابل خواندن به وسیله انسان‌ها و کامپیوترها می‌باشد. این زبان به عنوان زبان تبادل بین دو طراح هم به کار می‌رود. محتوای زبان به طور مؤثر و نیز به سادگی قابل ذخیره، بازیابی و پردازش به وسیله نرم‌افزار کامپیوتر است. در پردازش HDL دو کاربرد وجود دارد: شبیه‌سازی و سنتز.

شبیه‌سازی منطقی نمایشی از ساختار و رفتار یک سیستم منطقی دیجیتال به کمک کامپیوتر است. یک شبیه‌ساز توصیف HDL را تفسیر کرده و یک خروجی قابل خواندن مانند نمودار زمانی، تولید می‌نماید و بدین وسیله رفتار سخت افزار را قبل از ساخت پیش بینی می‌نماید. HDL امکان تشخیص خطای عملیاتی در طراحی را بدون نیاز به خلق فیزیکی آن، فراهم می‌سازد. خطاهایی که در حین شبیه‌سازی شناسایی می‌شوند را می‌توان با اصلاح عبارت مربوطه در HDL رفع کرد. امکاناتی که عملیات طرح را تست می‌کند، **برنامه تست** می‌نامند. بنابراین برای شبیه‌سازی یک سیستم، طرح ابتدا در HDL توصیف شده و سپس صحت عمل آن با شبیه‌سازی طرح و تست آن به وسیله برنامه تست که در HDL نوشته می‌شود، تحقیق می‌گردد.

سنتز منطقی فرآیندی است که طی آن از قطعات و اتصال بین آنها به نام netlist در مدل سیستم دیجیتالی که در HDL توصیف شده است لیستی تهیه می‌گردد. netlist سطح گیت را می‌توان در ساخت یک مدار مجتمع یا طرح بورد مدار چاپی به کار برد. سنتز منطقی مشابه با کامپایل یک برنامه در یک زبان سطح بالاست. تفاوت در این است که، در عوض تولید کد منتج، یک بانک اطلاعاتی تولید می‌نماید که در آن دستورالعمل‌های ساخت یک قطعه سخت‌افزار دیجیتال فیزیکی توصیف شده با کد HDL آمده است. سنتز منطقی بر روال‌هایی مبتنی است که مدارهای دیجیتال را پیاده‌سازی می‌کنند و شامل آن بخش از یک طراحی دیجیتال است که قابل اتوماتیک شدن با نرم‌افزار کامپیوتر باشد.

در صنعت HDL های انحصاری متعددی وجود دارند که به وسیله کمپانی‌ها برای طراحی یا کمک به طراحی مدارهای مجتمع ساخته شده‌اند. دو استاندارد HDL به وسیله IEEE پشتیبانی می‌شوند: VHDL و Verilog HDL . VHDL یک زبان تحت کنترل وزارت دفاع بود که در حال حاضر به صورت تجاری و در دانشگاه‌ها استفاده می‌شود. Verilog به عنوان یک زبان انحصاری که به وسیله کمپانی Cadence Data System ارتقاء یافت، ولی بعد کنترل آن را به مجموعه‌ای از کمپانی‌ها به نام Open Verilog International (OVI) محول کرد. VHDL نسبت به Verilog زبان سخت‌تری است. چون Verilog برای یادگیری ساده‌تر است، ما آن را در این کتاب انتخاب کرده‌ایم. با این وجود، توصیف‌های Verilog HDL لیست شده در سرتاسر این کتاب تنها درباره Verilog نیست، بلکه معرفی مفهوم نمایش سیستم‌های دیجیتال به کمک کامپیوتر به وسیله نوعی زبان توصیف سخت‌افزاری است.

نمایش مدول

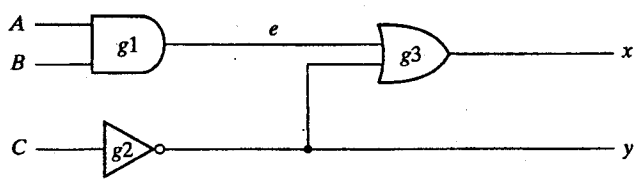
Verilog HDL دارای دستور زبانی است که دقیقاً ساختارهای مُجازی که در زبان می‌توانند به کار روند را توصیف می‌نماید. خصوصاً، Verilog حدود 100 کلمه کلیدی از پیش‌تعریف شده، حروف

کوچک و شناسه‌هایی دارد که ساختار زبان را تعریف می‌کنند. مثال‌هایی از کلمات کلیدی عبارتند از module، endmodule، input، output، wire، and، or، not و غیره. هر متن بین دو اسلش (//) و انتهای خط به عنوان یک توضیح تفسیر می‌گردد. فاصله‌های Blank و نام‌ها حساس به اندازه هستند، و این بدان معنی است که حروف بزرگ و کوچک با هم متفاوتند. در Verilog مدول یک بلوک ساختاری است. این دستور با کلمه کلیدی module آغاز و با کلمه کلیدی endmodule پایان می‌یابد. اکنون برای تشریح بعضی از مفاهیم زبان مثال ساده‌ای را تشریح می‌کنیم.

توصیف HDL مدار شکل ۳۷-۳ در مثال ۱-۳ HDL نشان داده شده است. خطی که دو اسلش دارد توضیحات است و عمل مدار را توضیح می‌دهد. دومین خط، مدول را همراه با نام و لیستی از پورت‌ها مشخص می‌کند. نام (در اینجا smpl–circuit) یک شناسه است که برای ارجاع به مدول به کار رفته است. شناسه‌ها نام‌هایی هستند که به متغیرها داده می‌شوند و به این ترتیب در طراحی قابل ارجاع می‌گردند. آنها از کاراکترهای الفبا عددی و زیرخط (_) ساخته می‌شوند و حساس به اندازه‌اند. شناسه‌ها باید با کاراکتر الفبایی و یا خط تیره شروع شوند. آنها را نمی‌توان با عدد شروع کرد. Port List رابطی بین مدول برای تبادل اطلاعات (مخابره) با محیط است. در این مثال پورت‌ها، ورودی‌ها و خروجی‌های مدارند. Port List بین پرانتزها محصور شده و از ویرگول برای جدا کردن عناصر لیست استفاده می‌شود. عبارت با نقطه ویرگول (;) پایان می‌یابد. همه کلمات کلیدی که باید به حروف کوچک باشند به منظور وضوح با خط پررنگ چاپ می‌شوند، ولی این نیاز زبان نیست. سپس input و output بیان می‌دارند که کدام پورت‌ها ورودی و کدام خروجی هستند. اتصالات درونی در نقش سیم‌ها می‌باشند. مدار دارای یک اتصال داخلی در e بوده و با کلمه کلیدی wire بیان می‌شود. ساختار مدار با گیت‌های اصلی از پیش تعریف شده به عنوان کلمه کلیدی مشخص می‌گردد. معرفی هر گیت با یک نام اختیاری مثل g1، g2 و غیره و به دنبال آن خروجی و ورودی‌هایی که با ویرگول از هم جدا شده و در داخل

مثال ۱-۳، HDL

```
//Description of simple circuit Fig. 3-37
module smpl_circuit(A,B,C,x,y);
   input A,B,C;
   outputx,y;
   wire e;
   and g1(e,A,B);
   not g2(y, C);
   or g3(x,e,y);
endmodule
```



شکل ۳۷-۳. مداری برای نمایش HDL

پرانتزاند، صورت می‌گیرد. همواره خروجی در ابتدا معرفی می‌شود و پس از آن ورودی ذکر می‌گردد. مثلاً گیت OR که g3 نامیده شده، دارای خروجی x و ورودی‌های e و y است. توصیف مدول با کلمه کلیدی endmodule خاتمه می‌یابد. توجه کنید که هر عبارت با یک نقطه ویرگول (;) پایان می‌پذیرد، ولی پس از endmodule نقطه ویرگول گذاشته نمی‌شود.

تأخیر در گیت‌ها

هنگام استفاده از HDL در شبیه‌سازی، گاهی لازم است مقداری تأخیر بین ورودی تا خروجی گیت در نظر گرفته شود. در Verilog تأخیر برحسب واحدهای زمانی و سمبل # معین می‌گردد. ارتباط یک واحد زمانی با زمان فیزیکی با استفاده از رهنمون کامپایلر timescale انجام می‌شود. رهنمون‌های کامپایلر با سمبل " ` " (backqoute) شروع می‌شوند. چنین رهنمونی قبل از اعلان مدول مشخص می‌گردد. مثالی از رهنمون timescale در زیر آمده است.

`timescale 1ns/100ps

عدد اول نشان‌دهنده واحد اندازه‌گیری برای زمان‌های تأخیر است. عدد دوم دقتی که تحت آن تأخیرها گرد شده‌اند را نشان می‌دهد که در این حالت 0.1ns است. اگر timescale مشخص نشود، شبیه‌ساز واحد زمان معینی را، مثل 1ns، پیش‌فرض می‌کند. در این کتاب، واحد زمان پیش‌فرض را انتخاب خواهیم کرد.

مثال ۲-۳ HDL توصیف مثال قبلی را همراه با تأخیر در هر گیت مشخص می‌نماید. گیت‌های AND، OR و NOT به ترتیب زمان تأخیر 30ns، 20ns و 10ns را دارند. اگر مدار شبیه‌سازی شود و ورودی‌ها از 000 به 111 تغییر یابند، خروجی‌ها طبق جدول ۶-۳ تغییر می‌کنند. خروجی وارونگر در y پس از تأخیر 10ns از 1 به 0 تغییر می‌یابد. خروجی گیت AND در e پس از 30ns تأخیر از 0 به 1 تغییر می‌کند. خروجی گیت OR در x در t=30ns از 1 به 0 می‌رود و سپس در t=50ns به 1 باز می‌گردد. در هر دو حالت، تغییر در خروجی گیت OR از تغییری که در 20ns قبل در ورودی‌اش اتفاق می‌افتد، ناشی می‌شود. واضح است که هر چند خروجی x پس از تغییرات ورودی نهایتاً در 1 ثبات پیدا می‌کند، تأخیرهای گیتی قبل از آن برای مدت 20ns یک جرقه منفی ایجاد می‌نمایند.

برای شبیه‌سازی یک مدار با HDL، لازم است ورودی‌ها را برای شبیه‌ساز به مدار اعمال کنیم تا پاسخ خروجی تولید گردد. یک توصیف HDL که محرک را برای یک طرح فراهم می‌کند **برنامه تست**

مثال ۲-۳، HDL

```
//Description of circuit with delay
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```

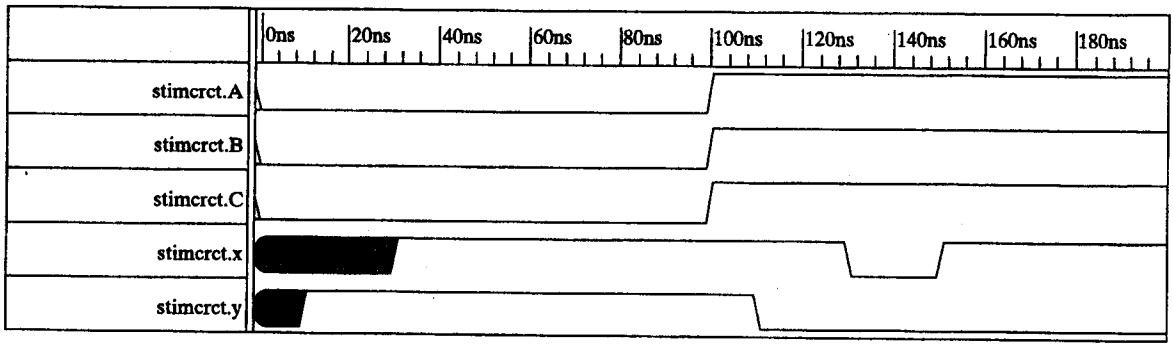
جدول ۶-۳. خروجی گیت‌ها پس از تأخیر

	واحد زمانی	ورودی	خروجی
	(ns)	ABC	y e x
تغییر اولیه	–	000	1 0 1
	–	111	1 0 1
	10	111	0 0 1
	20	111	0 0 1
	30	111	0 1 0
	40	111	0 1 0
	50	111	0 1 1

خوانده می‌شود. نوشتن برنامه‌های تست در انتهای بخش ۱۱-۴ توضیح داده شده است. در اینجا بدون آن که توضیحاتی اضافی را ارائه کنیم روال را با مثال ساده‌ای شرح می‌دهیم. مثال ۳-۳ HDL یک برنامه تست را برای شبیه‌سازی مدار تأخیردار نشان می‌دهد. دو مدول در این برنامه تست لحاظ شده است: مدول محرک و مدول توصیف مدار. مدول محرک stimcrct پورت ندارد. ورودی‌ها به مدار با کلمه کلیدی reg و خروجی نیز با کلمه کلیدی wire معرفی می‌شوند. circuit_with_delay با cwd نام‌گذاری یا ذکر شده است. (واکنش متقابل بین مدول محرک و مدول مدار در شکل ۳۳-۴ نشان داده شده است.)

مثال ۳-۳، HDL

```
//Stimulus for simple circuit
module stimcrct;
reg A,B,C;
wire x,y;
circuit_with_delay cwd(A,B,C,x,y);
initial
   begin
        A = 1'b0; B = 1'b0; C = 1'b0;
     #100
        A = 1'b1; B = 1'b1; C = 1'b1;
     #100  $finish;
   end
endmodule

//Description of circuit with delay
module circuit_with_delay (A,B,C,x,y);
   input A,B,C;
   output x,y;
   wire e;
   and #(30) g1(e,A,B);
   or  #(20) g3(x,e,y);
   not #(10) g2(y,C);
endmodule
```



شکل ۳۸-۳. خروجی شبیه‌سازی شده مثال ۲-۳ HDL

عبارت initial ورودی‌های بین کلمات کلیدی begin و end را مشخص می‌نماید. در آغاز ABC=000 است. (هر یک از A، B و C با 1'b0 تنظیم شده‌اند، و به معنی یک رقم دودویی با مقدار 0 است.) پس از 100ns ورودی‌ها به ABC = 111 تبدیل می‌شوند. پس از 100ns ثانیه دیگر شبیه‌سازی خاتمه می‌یابد. ($finish یک تکلیف در سیستم است.) نمودار زمانی که از شبیه‌سازی حاصل می‌گردد در شکل ۳۸-۳ نشان داده شده است. زمان کل شبیه‌سازی 200ns طول می‌کشد. ورودی‌های A، B و C پس از 100ns، از 0 به 1 تغییر می‌یابند. در اولین 10ns، خروجی y غیرمشخص است، خروجی x هم در اولین 30ns نامعین می‌باشد. خروجی y پس از 110ns از 1 به 0 می‌رود. خروجی x پس از 130ns از 1 به 0 می‌رود و در 150ns به 1 باز می‌گردد که این مقادیر دقیقاً در جدول ۶-۳ پیش‌بینی شده بود.

عبارات بولی

عبارات بولی در Verilog HDL با عبارت تخصیص مداوم یا پیوسته متشکل از کلمه کلیدی assign که پس از آن یک عبارت بولی آمده مشخص می‌گردد. برای تفکیک علامت جمع حسابی از علامت OR ، Verilog HDL از سمبل (&) ، (|) و (~) به ترتیب برای AND ، OR و NOT استفاده می‌کند. بنابراین برای توصیف مدار ساده شکل ۳۷-۳ با یک عبارت بولی عبارت زیر را به کار می‌بریم.

```
assign x = (A & B) | ~C);
```

مثال ۴-۳ HDL توصیف مداری که با دو عبارت بولی زیر بیان شده را نشان می‌دهد:

$$x = A + BC + B'D$$

$$y = B'C + BC'D'$$

مدار دارای دو خروجی x و y و چهار ورودی A، B، C و D است. دو عبارت assign معادلات بول را توصیف می‌نمایند.

دیدیم که یک مدار دیجیتال می‌تواند با عبارات HDL درست مثل ترسیم در یک نمودار مداری، یا با عبارات بولی توصیف گردد. مزیت HDL این است که برای پردازش به وسیله کامپیوتر مناسب است.

مثال ۴-۳، HDL

```
//Circuit specified with Boolean expressions
module circuit_bln (x,y,A,B,C,D);
   input A,B,C,D;
   output x,y;
   assign x = A | (B & C) | (~B & D);
   assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

مواردی که به وسیله کاربر تعریف می‌شود (UDP)

گیت‌های به کار رفته در توصیف‌های HDL ، با لغات کلیدی and ، or و غیره به وسیله سیستم تعریف می‌شوند و primitives سیستم نام‌گذاری می‌گردند. کاربر می‌تواند primitive های دیگری را با تعریف آنها به صورت جدول اضافه نماید. این نوع مدارها را تعریف شده بوسیله کاربر یا user–defind می‌نامند. یکی از راه‌های معرفی مدار به فرم جدول، معرفی آن با جدول درستی است. توصیف‌های UDP از کلمه کلیدی module استفاده نمی‌کنند. در عوض با کلمه کلیدی primitive (اصلی) تعریف می‌شوند. بهترین راه معرفی primitive ارائه یک مثال می‌باشد.

مثال ۵-۳ HDL یک UDP را با یک جدول درستی تعریف می‌کند. حل آن براساس قواعد کلی زیر است:

مثال ۵-۳، HDL

```
//User defined primitive(UDP)
primitive crctp (x,A,B,C);
   output x;
   input A,B,C;
//Truth table for x(A,B,C) = Σ(0,2,4,6,7)
   table
//      A   B   C  :  x   (Note that this is only a comment)
        0   0   0  :  1;
        0   0   1  :  0;
        0   1   0  :  1;
        0   1   1  :  0;
        1   0   0  :  1;
        1   0   1  :  0;
        1   1   0  :  1;
        1   1   1  :  1;
   endtable
endprimitive

//Instantiate primitive
module declare_crctp;
  reg x,y,z;
  wire w;
  crctp (w,x,y,z);
endmodule
```

- از کلمه کلیدی primitive استفاده شده و به دنبال آن یک نام و لیست پورت‌ها آورده می‌شود.
- تنها یک خروجی می‌تواند وجود داشته باشد که با بکارگیری کلمه کلیدی output و قبل از همه در لیست پورت اعلام می‌شود.
- به هر تعداد ورودی (input) می‌تواند تعریف شود. ترتیب معرفی آنها با اعلام input با ترتیب مقادیرشان در جدولی که به دنبال می‌آید، باید همخوانی داشته باشد.
- جدول درستی باید در داخل کلمات کلیدی table و endtable محصور شود.
- مقادیر ورودی با (:) پایان می‌یابند. خروجی همواره آخرین وارده در هر سطر است و بعد از آن (;) می‌آید.
- در پایان endprimitive ذکر می‌شود.

توجه کنید که متغیرهای لیست شده در بالای جدول بخشی از توضیحات بوده و به منظور آشنایی ذکر شده‌اند. سیستم متغیرها را به ترتیبی که در بخش ورودی ذکر شده‌اند تشخیص می‌دهد. یک UDP نیز مثل primitive سیستم به کار گرفته می‌شود. مثلاً

```
crctp (w,x,y,z)
```

مداری با تابع

$$w(x, y, z) = \Sigma(0, 2, 4, 6, 7)$$

و ورودی‌های x و y و z و خروجی w را پیاده می‌کند.

گرچه Verilog HDL این نوع توصیف را فقط برای UDP به کار می‌برد. دیگر HDL ها و سیستم‌های طراحی کامپیوتری (CAD) روال‌های دیگری را برای مشخص کردن مدارهای دیجیتال به صورت جدول استفاده می‌کنند. جداول می‌توانند با نرم‌افزار CAD برای بدست آوردن یک ساختار گیتی بهینه پردازش شوند.

در این بخش، HDL را معرفی و مثال‌های ساده‌ای از مدل‌سازی ساخت یافته را ارائه دادیم. مشروح مطالب فوق را برای Verilog HDL می‌توان در فصل بعدی یافت. خوانندگانی که با مدارهای ترکیبی آشنا هستند می‌توانند مستقیماً به بخش ۱۱-۴ مراجعه کرده و این موضوع را ادامه دهند.

مسائل

۳-۱ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

(الف) $F(x, y, z) = \Sigma(0, 2, 6, 7)$ (ب) $F(A, B, C) = \Sigma(0, 2, 3, 4, 6)$

(پ) $F(a, b, c) = \Sigma(0, 1, 2, 3, 7)$ (ت) $F(x, y, z) = \Sigma S(3, 5, 6, 7)$

۳-۲ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده کنید.

(الف) $F(x, y, z) = \Sigma(0, 1, 5, 7)$ (ب) $F(x, y, z) = \tilde{\Sigma}(1, 2, 3, 6, 7)$

۳-۳ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

(الف) $xy + x'y'z' + x'yz'$ (ب) $x'y' + yz + x'yz'$

(پ) $A'B + BC' + B'C'$

۳-۴ توابع بولی زیر را با استفاده از نقشه‌های چند متغیره ساده کنید.

(الف) $F(x, y, z) = \sum (2, 3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(4, 6, 7, 15)$

(پ) $F(A, B, C, D) = \sum (3, 7, 11, 13, 14, 15)$

(ت) $F(w, x, y, z) = \sum(2, 3, 12, 13, 14, 15)$

۳-۵ با استفاده از نقشه چهار متغیره توابع زیر را ساده نمایید.

(الف) $F(w, x, y, z) = \sum(1, 4, 5, 6, 12, 14, 15)$

(ب) $F(A, B, C, D) = \sum(0, 1, 2, 4, 5, 7, 11, 15)$

(پ) $F(w, x, y, z) = \sum(2, 3, 10, 11, 12, 13, 14, 15)$

(ت) $F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

۳-۶ با استفاده از نقشه‌های چهار متغیره توابع زیر را ساده کنید.

(الف) $A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$

(ب) $x'z + w'xy' + w(x'y + xy')$

۳-۷ عبارات بولی زیر را با نقشه‌های چهار متغیره ساده کنید.

(الف) $w'z + xz + x'y + wx'z$

(ب) $B'D + A'BC' + AB'C + ABC'$

(پ) $AB'C + B'C'D' + BCD + ACD' + A'B'C + A'BC'D$

(ت) $wxy + yz + xy'z + x'y$

۳-۸ با رسم هر تابع در یک نقشه کارنو، مینترم‌های عبارات بولی زیر را بدست آورید.

(الف) $xy + yz + xy'z$

(ب) $C'D + ABC' + ABD' + A'B'D$

(پ) $wxy + x'z' + w'xz$

۳-۹ موجب‌های اصلی عبارات بولی زیر را بدست آورید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۰ با یافتن موجب‌های اصلی اساسی توابع بول زیر را ساده کنید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۱ توابع بولی زیر را با نقشه‌های پنج متغیره ساده کنید.

(الف) $F(A, B, C, D, E) = \sum(0, 1, 4, 5, 16, 17, 21, 25, 29)$

(ب) $F = A'B'CE' + A'B'C'D' + B'D'E' + B'CD' + CDE' + BDE'$

۳-۱۲ توابع بولی زیر را به صورت ضرب حاصل جمع‌ها ساده کنید.

(الف) $F(w, x, y, z) = \sum (0, 2, 5, 6, 7, 8, 10)$ (ب) $F(A, B, C, D) = \prod(1, 3, 5, 7, 13, 15)$

۳-۱۳ عبارات بولی زیر را به صورت (۱) جمع حاصلضرب‌ها و (۲) ضرب حاصل جمع‌ها ساده کنید.

(الف) $x'z' + y'z' + yz' + xy$

(ب) $AC' + B'D + A'CD + ABCD$

(پ) $(A' + B' + D')(A + B' + C')(A' + B + D')(B + C' + D')$

۳-۱۴ به سه طریق تابع بولی زیر را با هشت لیترال یا کمتر نشان دهید:

$$F = A'B'D' + AB'CD' + A'BD + ABC'D$$

۳-۱۵ تابع بولی F زیر را با حالات بی‌اهمیت d ساده کرده و سپس تابع ساده شده را به صورت جمع مینترم‌ها در آورید:

(الف) $F(x, y, z) = \sum (0, 1, 2, 4, 5)$, $d(x, y, z) = \sum (3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(0, 6, 8, 13, 14)$, $d(A, B, C, D) = \sum(2, 4, 10)$

(پ) $F(A, B, C, D) = \sum(1, 3, 5, 7, 9, 15)$, $d(A, B, C, D) = \sum(4, 6, 12, 13)$

۳-۱۶ عبارات زیر را ساده کنید و آنها را با مدارات دو طبقه گیت NAND پیاده‌سازی نمایید:

(الف) $AB' + ABD + ABD' + A'C'D' + A'BC'$

(ب) $BD + BCD' + AB'C'D'$

۳-۱۷ یک نمودار گیت NAND رسم کنید به نحوی که تابع زیر را پیاده‌سازی کرده آن را متمم نماید:

$$F(A, B, C, D) = \sum(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۱۸ یک نمودار منطقی با استفاده از گیت‌های NAND دو ورودی برای پیاده‌سازی عبارت زیر رسم کنید.

$$(AB + A'B')(CD' + C'D)$$

۳-۱۹ توابع زیر را ساده کنید و آنها را با مدارهای دو طبقه گیت NOR پیاده نمایید:

(الف) $F = wx' + y'z' + w'yz'$ (ب) $F(w, x, y, z) = \sum(5, 6, 9, 10)$

۳-۲۰ یک مدار NAND چند طبقه برای عبارت زیر رسم کنید.

$$(AB' + CD')E + BC(A + B)$$

۳-۲۱ یک مدار چند طبقه NOR برای عبارت زیر رسم کنید.

$$w(x + y + z) + xyz$$

۳-۲۲ نمودار مدار شکل ۴-۴ را به مدار چند طبقه NAND تبدیل نمایید.

۳-۲۳ تابع بول F را همراه با حالات بی‌اهمیت d با کمتر از دو گیت NOR پیاده‌سازی نمایید.

$$F(A, B, C, D) = \sum(0, 1, 2, 9, 11)$$
$$d(A, B, C, D) = \sum(8, 10, 14, 15)$$

فرض کنید که هر دو نوع ورودی معمولی و متمم موجودند.

۳-۲۴ تابع بول F را با فرم‌های دو طبقه (الف) NAND–AND، (ب) AND–NOR، (پ) OR–NAND و (ت) NOR–OR پیاده‌سازی کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۲۵ هشت فرم دو طبقه غیرمفید (زاید) نمایش تابع را لیست کنید و نشان دهید که به یک عمل کاهش می‌یابند. نشان دهید که این فرم‌ها چگونه برای گسترش تعداد ورودی‌ها به کار می‌روند.

۳-۲۶ با استفاده از نقشه‌ها، ساده‌ترین فرم جمع حاصلضرب تابع F = fg را بیابید، که در آن f و g برابرند با

$$f = wxy' + y'z + w'yz' + x'yz'$$

و

$$g = (w + x + y' + z')(x' + y' + z)(w' + y + z')$$

۳-۲۷ نشان دهید که دوگان XOR، متمم آن هم هست.

۳-۲۸ با توازن فرد، مدارهای مولد توازن سه بیتی و چک‌کننده توازن چهار بیتی را بدست آورید.

۳-۲۹ عبارات بولی زیر را با نیم جمع‌کننده پیاده‌سازی کنید.

$$D = A \oplus B \oplus C$$
$$E = A'BC + AB'C$$
$$F = ABC' + (A' + B')C$$
$$G = ABC$$

۳-۳۰ عبارت بولی زیر را با گیت‌های XOR و AND پیاده کنید.

$$F = AB'CD' + A'BCD' + AB'C'D + A'BC'D$$

۳-۳۱ توصیف ساختار گیتی HDL مدار شکل ۳-۲۲ (الف) را بنویسید.

۳-۳۲ مدار XOR شکل ۳-۲۲ (الف) دارای گیت‌هایی با تأخیر 10ns برای هر وارونگر، 20ns تأخیر برای هر گیت AND و 30ns برای هر گیت OR است. ورودی مدار از xy = 00 به xy = 01 می‌رود.
(الف) سیگنال‌ها را در خروجی هر گیت از t = 0 تا t = 50ns معین کنید.
(ب) توصیف HDL را برای مدار با در نظر گرفتن تأخیرها بنویسید.
(پ) یک مدول محرک (مشابه مثال ۳-۳ HDL) بنویسید و مدار را برای تأیید پاسخ بخش (الف) شبیه‌سازی کنید.

۳-۳۳ توصیف HDL مدار شکل ۳-۳۷ را با دو عبارت بنویسید.

۳-۳۴ توصیف HDL مداری را که با توابع بولی زیر مشخص شده بنویسید. از عبارات تخصیص یافته پیوسته استفاده نمایید.

$$x = A(CD + B) + BC'$$
$$y = (AB' + A'B)(C + D')$$
$$z = [(A + B)(C' + D'B)]'$$

۳-۳۵ خطاهای نحوی را در اعلان‌های زیر بیابید (توجه کنید که نام گیت‌های primitive اختیاری‌اند):

```
module Exmpl-3(A,B,C,D,F)
inputs A,B,C,
Output D,F;
and g1(A,B,D);
not (D,B,A);
OR (F,B,C);
endmodule;
```

۳-۳۶ نمودار منطقی مدار دیجیتالی که با توصیف HDL زیر بیان شده را رسم کنید.

```
module circt (A,B,C,D,F);
input A,B,C,D;
output F;
wire w,x,y,z,a,d;
and (x,B,C,d);
and (y,a,C);
and (w,z,B);
or (z,y,A);
or (F,x,w);
not (a,A);
not (d,D);
endmodule
```

۳-۳۷ یک تابع منطقی اکثریت، تابعی است که اگر اکثریت متغیرها 1 باشند برابر 1 می‌شود در غیر این صورت 0 است. UDP را برای تابع اکثریت 3 بیتی بنویسید.

مراجع

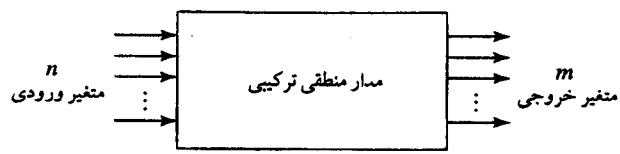
1. BHASKER, J. 1997. *A Verilog HDL Primer.* Allentown, PA: Star Galaxy Press.
2. HILL, F. J., and G. R. PETERSON. 1981. *Introduction to Switching Theory and Logical Design,* 3rd ed. New York: John Wiley.
3. *IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language* (IEEB Std 1364–1995). 1995. New York: The Institute of Electrical and Electronics Engineers.
4. KARNAUGH, M. A Map Method for Synthesis of Combinational Logic Circuits. *Transactions of AIEE, Communication and Electronics,* 72, part I (Nov. 1953): 593–99.
5. KOHAVI, Z. 1978. *Switching and Automata Theory,* 2nd ed. New York: McGraw-Hill.
6. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.
7. MCCLUSKEY, E. J. 1986. *Logic Design Principles.* Englewood Cliffs, NJ: Prentice-Hall.
8. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis.* SunSoft Press (A Prentice Hall Title).

منطق ترکیبی

۴-۱ مدارهای ترکیبی

مدارهای منطقی در سیستم‌های دیجیتال می‌توانند از نوع ترکیبی و یا ترتیبی باشند. یک مدار ترکیبی متشکل از تعدادی گیت منطقی است که خروجی آنها در هر لحظه از زمان مستقیماً به وسیله ورودی‌های همان لحظه معین می‌شود و به ورودی‌های قبلی بستگی ندارد. این نوع مدار، پردازشی را انجام می‌دهد که با مجموعه‌ای از توابع بولی مشخص می‌گردد. مدارهای ترتیبی علاوه بر گیت‌های منطقی از عناصر حافظه نیز استفاده می‌کنند. خروجی‌های آنها تابعی از ورودی‌ها و حالت عناصر حافظه است. در نتیجه خروجی یک مدار ترتیبی نه تنها به مقادیر فعلی ورودی‌ها بلکه به ورودی‌های قبلی وابسته بوده و عملکرد مدار باید به وسیله حالات داخلی و ترتیب زمانی ورودی‌ها مشخص گردد. مدارهای ترتیبی در فصل‌های ۵ و ۹ بحث شده‌اند.

یک مدار ترکیبی از متغیرهای ورودی، گیت‌های منطقی، و متغیرهای خروجی تشکیل شده است. گیت‌های منطقی سیگنال‌هایی را از ورودی‌ها دریافت کرده و سیگنال‌هایی را برای خروجی‌ها تولید می‌نمایند. این فرآیند اطلاعات دودویی مفروض در ورودی را به اطلاعات موردنیاز در خروجی تبدیل می‌کند. نمودار کلی یک مدار ترکیبی در شکل ۴-۱ دیده می‌شود. n متغیر دودویی ورودی از منبع



شکل ۴-۱ . نمودار بلوکی یک مدار ترکیبی

بیرونی دریافت و m متغیر خروجی به مقصد بیرونی ارسال می‌شوند. هر متغیر ورودی و یا خروجی به طور فیزیکی به صورت یک سیگنال نشان داده می‌شوند و این سیگنال‌ها نیز 0 و 1 منطقی را نمایش می‌دهند. در بسیاری از کاربردها، منبع و مقصد، ثبات‌های ذخیره‌سازی هستند. اگر ثبات‌ها به همراه گیت‌های منطقی به کار روند، کل مدار با نام مدار ترتیبی شناخته خواهد شد.

برای n متغیر ورودی، 2^n ترکیب ممکن دودویی از ورودی‌ها وجود دارد. برای هر ترکیب ممکن از ورودی‌ها فقط یک مقدار برای خروجی موجود است. بنابراین، یک مدار ترکیبی با یک جدول درستی، که مقادیر خروجی‌ها را در برابر هر ترکیب از متغیرهای ورودی لیست می‌نماید، نشان داده می‌شود. یک مدار ترکیبی با m تابع بولی نیز قابل نمایش است، که هر یک متعلق به یک خروجی است. هر تابع خروجی برحسب n متغیر ورودی بیان می‌گردد.

در فصل ۱ در مورد اعداد و کدهای دودویی که کمیت‌های گسسته‌ای از اطلاعات را نمایش می‌دهند، مطالبی آموختیم. متغیرهای دودویی به طور فیزیکی با ولتاژها و دیگر انواع سیگنال‌ها نشان داده می‌شوند. سیگنال‌ها نیز در سیستم‌های منطقی دیجیتال برای اجرای توابع موردنیاز دستکاری می‌شوند. در فصل ۲، جبر بول را به عنوان روشی جبری در بیان توابع منطقی معرفی نمودیم. در فصل ۳ آموختیم که چگونه عبارت بول را برای دستیابی به یک پیاده‌سازی اقتصادی، ساده کنیم. در این فصل با استفاده از دانش فصل‌های قبل، تحلیل و طراحی مدارهای ترکیبی را فرموله می‌نماییم. با حل مثال‌های نمونه فهرستی از توابع اصلی مهم برای درک سیستم‌های دیجیتال فراهم خواهد شد.

مدارهای ترکیبی متعددی وجود دارند که در طراحی سیستم‌های دیجیتال به کرّات به کار می‌روند. این مدارها به صورت مجتمع در دسترس بوده و به عنوان قطعات استاندارد دسته‌بندی شده‌اند. آنها توابع دیجیتال خاصی را که عموماً در طراحی سیستم‌های دیجیتال مورد نیازند، اجرا می‌کنند. در این فصل ما مهمترین مدارهای ترکیبی استاندارد مانند جمع‌کننده‌ها، تفریق‌گرها، مقایسه‌گرها، دیکدرها، انکدرها و مولتی پلکسرها را معرفی می‌کنیم. این قطعات به صورت مدارهای مجتمع MSI (مجتمع با فشردگی متوسط) در دسترسند. به آنها در مدارهای پیچیده VLSI ، مانند مدارات مجتمع خاص (ASIC) ، سلول‌های استاندارد هم می‌گویند. توابع سلول‌های استاندارد در داخل مدارهای VLSI به همان شکل به هم متصل می‌شوند که در طراحی MSI متشکل از چند IC ، وصل شدند.

۲-۴ روش تحلیل

تحلیل یک مدار ترکیبی به این معنی است که ما تابعی را که مدار پیاده‌سازی می‌کند، معین نماییم. این کار با یک نمودار منطقی مفروض آغاز شده و با مجموعه‌ای از توابع بول، یک جدول درستی، یا توضیحاتی از عمل مدار پایان می‌یابد. اگر نمودار منطقی مورد بررسی با نام تابع یا توضیحی از کار آن همراه باشد، آنگاه تحلیل به تصدیق تابع بیان شده کاهش می‌یابد. تحلیل را می‌توان به طور دستی با یافتن توابع بول یا جدول درستی، و یا با استفاده از یک برنامه شبیه‌سازی کامپیوتری اجرا نمود.

اولین قدم در تحلیل این است که مطمئن شویم مدار از نوع ترکیبی است و نه ترتیبی. نمودار یک

مدار ترکیبی حاوی گیت‌هایی است که فاقد مسیرهای پسخورد یا حافظه است. یک مسیر پسخورد، اتصالی است از خروجی یک گیت به ورودی گیت دیگری که خود بخش ورودی آن را (گیت خروجی) تشکیل می‌دهد. مسیرهای پسخوردی در یک مدار دیجیتال مدار ترتیبی را تعریف می‌کنند و باید طبق روال فصل ۹ با آنها رفتار کرد.

به محض این که محقق شد مدار از نوع ترکیبی است، می‌توان برای بدست آوردن توابع بول خروجی یا جداول درستی پیش رفت. اگر تابع مدار تحت بررسی است، لازم است عمل مدار را از توابع بول حاصل یا جداول درستی تفسیر کرد. موفقیت در چنین بررسی‌هایی به شرطی میسر است که فرد تجربه قبلی و آشنایی لازم با چنین مدارهایی داشته باشد.

برای بدست آوردن توابع بول خروجی از یک مدار منطقی به ترتیب زیر باید عمل کرد.

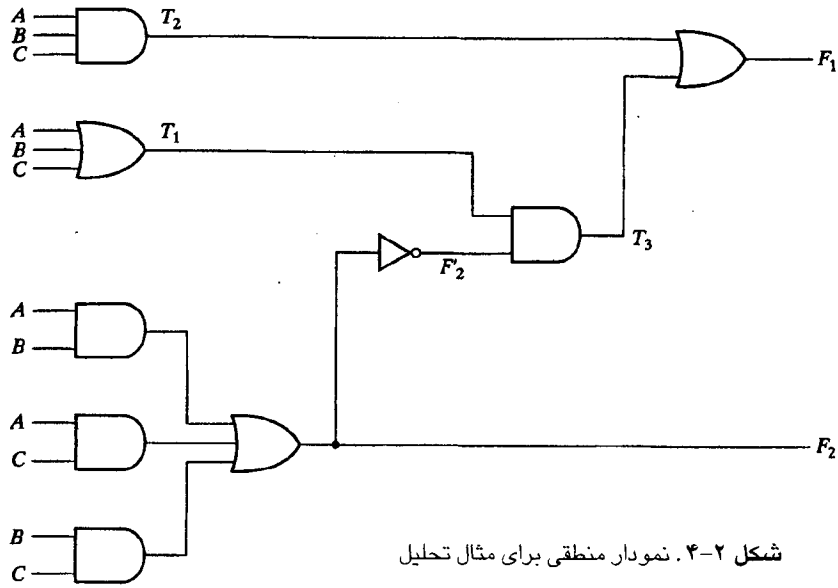
۱- تمام خروجی‌های گیت‌هایی که تابعی از ورودی هستند باید با سمبل‌های دلخواه نام‌گذاری شوند. برای خروجی هر گیت تابع بول را معین کنید.

۲- گیت‌هایی که تابعی از متغیرهای ورودی و گیت‌های برچسب خورده قبلی‌اند را با سمبل‌های اختیاری دیگری برچسب بزنید. برای این گیت‌ها نیز توابع بول خروجی را بدست آورید.

۳- فرآیند مرحله ۲ را تا دستیابی به خروجی‌های مدار ادامه دهید.

۴- با جایگزینی توابع بدست آمده در قبل، توابع بولی خروجی را برحسب متغیرهای ورودی اولیه بدست آورید.

تحلیل مدارهای ترکیبی شکل ۲-۴ روال پیشنهادی را تشریح می‌نماید. توجه دارید که مدار دارای سه ورودی دودویی A، B و C و دو خروجی F_1 و F_2 است. خروجی گیت‌هایی که تابعی از متغیرهای ورودی‌اند عبارتند از T_1 و T_2. خروجی F_2 به سادگی از متغیرهای ورودی بدست می‌آید. توابع بول



شکل ۲-۴. نمودار منطقی برای مثال تحلیل

برای این سه خروجی عبارتند از:

$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$

اکنون خروجی گیت‌هایی که تابعی از سمبل‌های قبلی می‌باشند را ملاحظه می‌نماییم:

$$T_3 = F_2'T_1$$
$$F_1 = T_3 + T_2$$

برای بدست آوردن F_1 برحسب A، B و C، یک سری جایگزینی‌ها را به فرم زیر انجام می‌دهیم:

$$\begin{aligned} F_1 &= T_3 + T_2 = F_2'T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC \end{aligned}$$

اگر بخواهیم این بررسی را دنبال کرده و عمل انتقال اطلاعات را با این مدار مشخص نماییم، می‌توانیم مدار را از عبارات بولی فوق رسم کرده و سعی کنیم عملیات آشنا را تشخیص دهیم. توابع بول F_1 و F_2 با مدار شکل ۷-۴ پیاده‌سازی می‌شوند (بخش ۴-۴) و معادل با یک مدار جمع کننده (یا تمام جمع کننده) است.

بدست آوردن جدول درستی برای مدار به محض شناختن توابع بولی خروجی روندی ساده است. برای تهیه مستقیم جدول درستی از نمودار منطقی و بدون نیاز به توابع بول به طریق زیر عمل کنید.

۱- تعداد متغیرهای ورودی در مدار را مشخص کنید. برای n ورودی 2^n ترکیب از ورودی‌ها را تشکیل دهید. آنگاه اعداد دودویی را در جدول از 0 تا 2^n-1 لیست نمایید.

۲- خروجی‌های گیت‌های انتخابی را با سمبل‌های دلخواه برچسب بزنید.

۳- برای آن دسته از خروجی گیت‌ها که فقط تابعی از متغیرهای ورودی هستند جدول درستی را بدست آورید.

۴- برای بدست آوردن خروجی گیت‌هایی که تابعی از مقادیر تعریف شده قبلی هستند پیش بروید تا ستون همه خروجی‌ها معین شود.

این فرآیند با استفاده از مدار شکل ۲-۴ تشریح می‌شود. در جدول ۱-۴، هشت ترکیب ممکن را برای

جدول ۱-۴. جدول درستی برای نمودار منطقی شکل ۲-۴

A	B	C	F_2	F_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

سه متغیر ورودی تشکیل می‌دهیم. جدول درستی برای F_2 مستقیماً از مقادیر A ، B و C تشکیل می‌شود که در آن برای هر ترکیبی که دو یا سه ورودی برابر با 1 دارد، F_2 برابر 1 است. جدول درستی برای F'_2 متمم F_2 است. جداول درستی برای T_1 و T_2 به ترتیب توابع OR و AND متغیرهای ورودی می‌باشند. مقدار T_3 از T_1 و F'_2 حاصل می‌شود: وقتی که هر دو T_1 و F'_2 برابر با 1 باشند T_3 نیز برابر 1 است، در غیر این صورت T_3 برابر 0 خواهد بود. بالاخره F_1 برای آن دسته از ترکیبات 1 است که در آنها T_2 یا T_3 یا هر دو برابر 1 باشند. بررسی ترکیبات جدول درستی برای A ، B و C و F_1 و F_2 نشان می‌دهد که این جدول با جدول جمع کننده کامل بخش ۴-۴ با متغیرهای x و y و z و S و C برابر است.

روش دیگر در تحلیل یک مدار ترکیبی شبیه‌سازی منطقی آن است. در بخش ۱۱-۴ شبیه‌سازی منطقی و تصدیق مدار شکل ۲-۴ را با کمک Verilog–HDL خواهیم دید (مثال ۱۰-۴ ملاحظه شود.)

۳-۴ روش طراحی

طراحی مدارهای ترکیبی با مشخصات مسئله آغاز و به فرم نمودار مدار منطقی یا مجموعه‌ای از توابع بول که به کمک آنها نمودار منطقی حاصل می‌شود، پایان می‌یابد. روال شامل موارد زیر است:

۱- با استفاده از مشخصات مدار تعداد ورودی‌ها و خروجی‌ها را معین کرده و به هر کدام سمبلی تخصیص دهید.

۲- جدول درستی مربوط به ورودی‌ها و خروجی‌های مدار را تشکیل دهید.

۳- توابع بولی ساده شده را برای هر خروجی به صورت تابعی از متغیرهای ورودی بدست آورید.

۴- نمودار منطقی را رسم کرده و صحت طراحی را تحقیق نمایید.

جدول درستی یک مدار ترکیبی، از ستون‌های ورودی و ستون‌های خروجی تشکیل می‌شود. ستون‌های ورودی از 2^n ترکیب مربوط به n متغیر ورودی بدست می‌آید. مقادیر دودویی خروجی‌ها از مشخصات بیان شده در مسئله حاصل می‌گردد. توابع خروجی مشخص شده در جدول درستی تعریف دقیقی از مدار ترکیبی را بدست می‌دهند. تفسیر لفظی صحیح جدول درستی از اهمیت خاصی برخوردار است. اغلب مشخصات لفظی کامل نیستند و تفسیر غلط ممکن است جدول درستی غلطی را تولید کند.

توابع دودویی خروجی لیست شده در جدول با روش‌های موجود مانند دستکاری جبری، نقشه کارنو یا برنامه‌های ساده‌سازی مبتنی بر کامپیوتر ساده می‌شوند. غالباً عبارات ساده شده متعددی حاصل می‌شوند که باید مناسب‌ترین را انتخاب کرد. در یک کاربرد خاص، معیارهای مختلفی در انتخاب یک پیاده‌سازی نقش دارند. یک طرح عملی قیودی چون تعداد گیت‌ها، تعداد ورودی‌ها به یک گیت، زمان انتشار سیگنال در گیت‌ها، تعداد اتصالات داخلی، محدودیت‌های مربوط به قابلیت راه‌اندازی هر گیت، و دیگر معیارهایی که باید در طراحی با مدارهای مجتمع مدّنظر باشند، را در نظر می‌گیرد. چون اهمیت هر قید را کاربرد خاص دیکته می‌کند، بیان یک عبارت جامع درباره ساده‌سازی مشکل است. در بسیاری از حالات ساده‌سازی با تصدیق و تأیید یک هدف ساده، مثل تولید توابع بولی به فرم استاندارد آغاز شده و سپس با برآورده کردن دیگر معیارهای رفتاری پیش می‌رود.

مثال تبدیل کد

وجود کدهای گوناگون و متنوع برای بیان اجزاء اطلاعات گسسته، باعث شده است تا سیستم‌های دیجیتال مختلف از کدهای متفاوتی استفاده کنند. گاهی لازم است خروجی یک سیستم به عنوان ورودی به سیستمی دیگر استفاده شود. اگر این دو سیستم از کدهای متفاوتی برای بیان اطلاعات یکسان استفاده کنند، یک مدار مبدل باید بین آن دو قرار داده شود. بنابراین یک مبدل کد مداری است که دو سیستم را، علیرغم به کارگیری کد دودویی متفاوت، با هم سازگار می‌سازد.

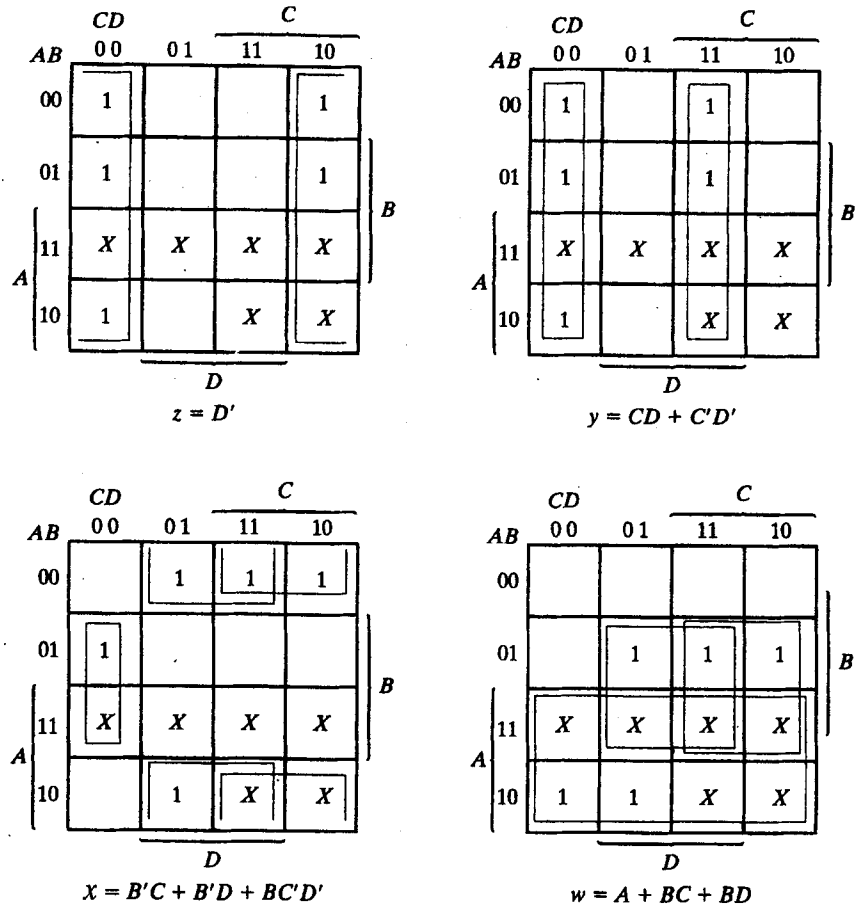
برای تبدیل کد دودیی A به کد دودیی B، خطوط ورودی باید ترکیبات بیتی اجزاء مشخص شده با کد A را تهیه نموده و خطوط خروجی نیز باید ترکیبات کد B مربوطه را تولید نمایند. یک مدار ترکیبی به کمک گیت‌ها این تبدیل را انجام می‌دهد. روش طراحی با مثالی که دهدهی کد شده به دودویی (BCD) را به کد افزونی -3 تبدیل می‌نماید، تشریح خواهد شد.

ترکیبات تخصیص داده شده به BCD و افزونی -3 در جدول ۵-۱ لیست شده است (بخش ۷-۱). چون هر کد، از چهار بیت برای نمایش یک رقم دهدهی استفاده می‌نماید، باید چهار متغیر ورودی و چهار متغیر خروجی داشته باشیم. چهار متغیر دودویی را با A، B، C و D و چهار متغیر خروجی را با w، x، y و z نام‌گذاری کنید. جدول درستی رابط بین ورودی‌ها و خروجی‌ها در جدول ۲-۴ دیده می‌شود. ترکیبات بیتی برای ورودی‌ها و خروجی‌های متناظر آنها مستقیماً در بخش ۷-۱ بدست آمد. توجه کنید که چهار متغیر دودویی دارای 16 ترکیب‌اند ولی تنها 10 عدد از آنها در جدول درستی ذکر شده‌اند. 6 ترکیب بیتی ذکر نشده برای متغیرهای ورودی ترکیبات بی‌اهمیت هستند. این مقادیر در BCD مفهوم ندارند و فرض بر این است که هرگز رخ نمی‌دهند. بنابراین به متغیرهای خروجی مربوط به آنها به دلخواه 0 یا 1 خواهیم داد و این تخصیص به نحوی خواهد بود که از آن مدار ساده‌تری حاصل گردد.

نقشه‌ها در شکل ۳-۴ برای بدست آوردن توابع بول خروجی ساده شده رسم شده‌اند. هر یک از چهار نقشه به یکی از خروجی‌های مدار به عنوان تابعی از چهار متغیر ورودی مربوط است. 1هایی که

جدول ۲-۴. جدول درستی برای مثال تبدیل کد

ورودی BCD				خروجی کد افزونی - 3			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



شکل ۳-۴. نقشه‌های تبدیل BCD به افزونی -3

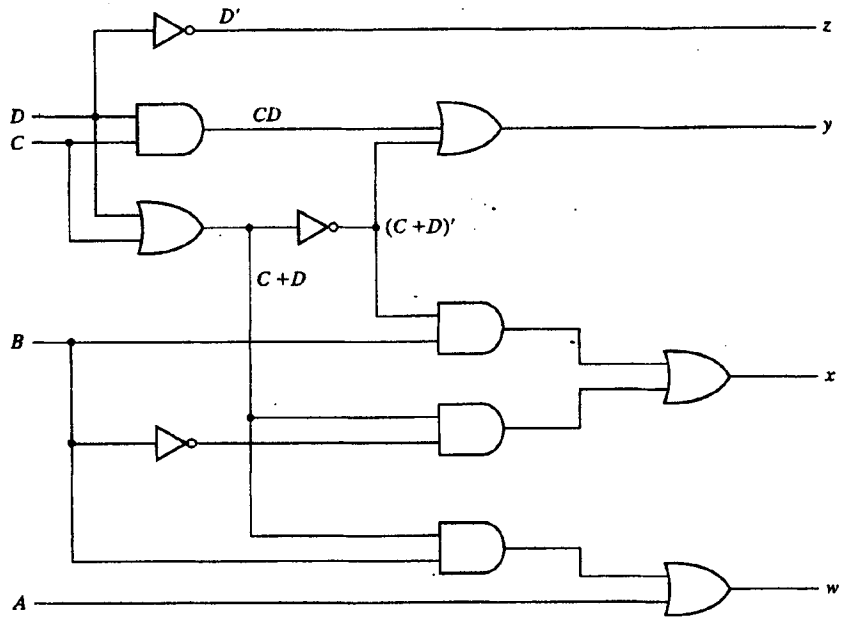
در مربع‌ها نوشته شده‌اند از مینترم‌هایی که خروجی را 1 کنند بدست می‌آیند. این 1ها با در نظر گرفتن تک‌تک ستون‌های خروجی در جدول درستی مشخص می‌شوند. مثلاً ستون زیر خروجی z دارای پنج عدد 1 است؛ بنابراین، نقشه z دارای پنج 1 می‌باشد که هر یک متعلق به مینترمی است که z توسط آن برابر 1 می‌شود. شش مینترم بی‌اهمیت از 10 تا 15 با علامت X علامت زده شده‌اند. یکی از نتایج ساده‌سازی توابع در جمع حاصلضرب‌ها در زیر نقشه هر متغیر خروجی نوشته شده است.

نمودار دو سطحی را می‌توان مستقیماً از عبارات بولی حاصل از نقشه‌ها بدست آورد. البته فرم‌های متعدد دیگری نیز برای بدست آوردن نمودار منطقی که همین مدار را پیاده‌سازی کند وجود دارد. عبارات حاصل در شکل ۳-۴ را به منظور استفاده از گیت‌های مشترک می‌توان دستکاری جبری نمود. این دستکاری جبری که در زیر آمده است انعطاف‌پذیری حاصل با سیستم‌های چند خروجی را وقتی با سه، یا چهار سطح و یا بیشتر پیاده‌سازی می‌شوند، نشان می‌دهد.

$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$

نمودار منطقی که این توابع را پیاده‌سازی می‌کند در شکل ۴-۴ دیده می‌شود. مشاهده می‌شود. گیت OR که خروجی‌اش (C + D) است به نحوی در پیاده‌سازی هر سه خروجی به کار رفته است.

بدون احتساب گیت‌های وارونگر در ورودی، پیاده‌سازی به صورت جمع حاصلضرب‌ها به هفت گیت AND و سه گیت OR نیاز دارد. در شکل ۴-۴ همین سیستم به چهار گیت AND، چهار گیت OR و یک وارونگر احتیاج دارد. اگر تنها ورودی‌های معمولی یا نرمال در دسترس باشند، پیاده‌سازی اول به وارونگرهایی برای متغیرهای B و C و D نیاز خواهد داشت، ولی در پیاده‌سازی دوم فقط B و D نیاز به وارونگر دارند.



شکل ۴-۴. نمودار منطقی برای تبدیل BCD به افزونی -3

۴-۴ جمع‌کننده- تفریق‌گر دودویی

کامپیوترهای دیجیتال اعمال پردازش اطلاعات گوناگونی را انجام می‌دهند. از آن میان می‌توان توابع مربوط به انواع اعمال حسابی را نام برد. اصلی‌ترین عمل حسابی جمع دو رقم دودویی است. این جمع

ساده شامل چهار عمل پایه است: یعنی $0 + 0 = 0$، $0 + 1 = 1$، $1 + 0 = 1$ و $1 + 1 = 10$. سه عمل اول جمعی یک رقمی تولید می‌کنند، ولی وقتی هر دو بیت مضاف و مضاف‌الیه برابر 1 باشند، جمع دودویی از دو رقم تشکیل خواهد شد. بیت باارزش‌تر این نتیجه را نقلی می‌گویند. وقتی مضاف و مضاف‌الیه دارای ارقام باارزش‌تر بیشتری باشند، نقلی حاصل از جمع دو بیت با جفت بیت باارزش‌تر بعدی افزوده می‌شود. مدار ترکیبی که جمع دو بیت را انجام می‌دهد، نیم جمع‌کننده نام دارد. مداری که سه بیت را با هم جمع کند، (دو بیت به علاوه بیت نقلی) جمع‌کننده کامل یا تمام جمع کننده خوانده می‌شود. اسم مدارها به این علت انتخاب شده است که از دو نیم جمع‌کننده می‌توان در پیاده‌سازی یک جمع کننده کامل استفاده کرد.

یک جمع- تفریق‌گر دودویی مداری ترکیبی است که عملیات حسابی جمع و تفریق را با اعداد دودویی انجام می‌دهد. ما این مدار را به صورت سلسله مراتبی طراحی خواهیم کرد. ابتدا طراحی نیم‌جمع‌کننده انجام می‌شود، و با استفاده از آن جمع کننده کامل را طراحی خواهیم کرد. با اتصال سری n جمع کننده کامل جمع دو عدد n بیتی تولید می‌گردد. مدار تفریق‌گر با افزودن مدار متمم‌ساز به آن ساخته می‌شود.

نیم جمع‌کننده

با توجه به توضیحات لفظی یک نیم‌جمع کننده، در می‌یابیم که مدار نیاز به دو ورودی دودویی و دو خروجی دودویی دارد. متغیرهای ورودی بیت‌های مضاف و مضاف‌الیه را مشخص می‌کنند. متغیرهای خروجی جمع و نقلی را تولید می‌نمایند. ما سمبل‌های x و y را به دو ورودی و S (برای جمع) و C (نقلی) را به خروجی‌ها تخصیص می‌دهیم. جدول درستی برای نیم جمع‌کننده در جدول ۳-۴ نشان داده شده است. خروجی C فقط هنگامی 1 است که هر دو ورودی 1 باشند. خروجی S، بیت کم‌ارزش‌تر حاصل جمع را نشان می‌دهد.

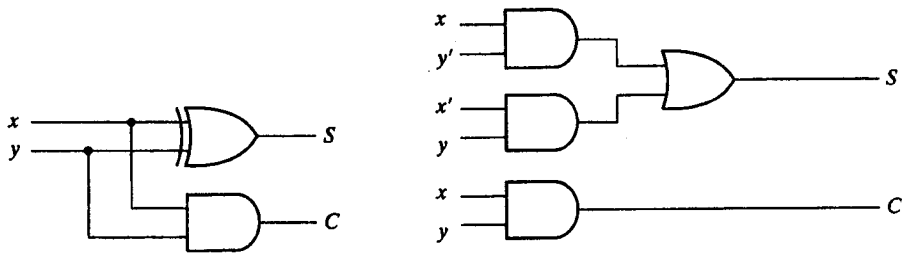
توابع بولی ساده شده برای دو خروجی مستقیماً از جدول درستی بدست می‌آیند. عبارات جمع حاصلضرب ساده شده عبارتند از:

$$S = x'y + xy'$$
$$C = xy$$

نمودار منطقی نیم‌جمع کننده پیاده شده با جمع حاصلضرب‌ها در شکل ۵-۴ (الف) دیده می‌شود. می‌توان آن را با گیت‌های XOR و AND طبق شکل ۵-۴ (ب) هم پیاده کرد. این نوع برای ساخت جمع‌کننده کامل از دو نیم جمع‌کننده به کار می‌رود.

جدول ۳-۴. نیم جمع‌کننده

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



(ب) $S = x \oplus y$, $C = xy$

(الف) $S = xy' + x'y$, $C = xy$

شکل ۵-۴. پیاده‌سازی نیم جمع‌کننده

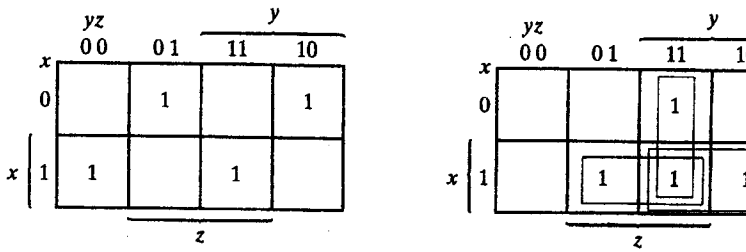
جمع‌کننده کامل

یک جمع‌کننده کامل مداری ترکیبی است که جمع حسابی سه بیت را تشکیل می‌دهد. این مدار دارای سه ورودی و دو خروجی است. دو متغیر ورودی که با x و y نشان داده شده‌اند دو بیت باارزش جمع شونده را نشان می‌دهند. ورودی سوم، z، نقلی حاصل از مکان کم‌ارزش‌تر قبلی است. دو خروجی لازم است زیرا جمع حسابی سه رقم دودویی بین 0 تا 3 می‌باشد و اعداد 2 و 3 به دو رقم دودویی نیاز دارند. دو خروجی با سمبل S برای جمع و C برای نقلی مشخص شده‌اند. متغیر دودویی S مقدار کم‌ارزش‌تر جمع را بدست می‌دهد. متغیر دودویی C نقلی خروجی را بیان‌گر است. جدول درستی جمع‌کننده کامل در جدول ۴-۴ دیده می‌شود. هشت سطر زیر سه متغیر همه ترکیبات ممکن سه متغیر را نشان می‌دهند. متغیرهای خروجی از جمع حسابی بیت‌های ورودی معین می‌شوند. وقتی همه بیت‌های ورودی 0 هستند، خروجی 0 است. خروجی S هنگامی 1 می‌شود که فقط یک ورودی برابر 1 باشد و یا اگر هر سه ورودی 1 باشند. خروجی C هم موقعی 1 است که دو یا سه ورودی برابر 1 باشند.

تفسیر بیت‌های ورودی و خروجی مدار ترکیبی در مراحل مختلف طراحی متفاوت است. به طور فیزیکی سیگنال‌های دودویی ورودی‌ها ارقامی دودویی تصور می‌شوند که به صورت حسابی باید با هم جمع شده و جمع دو رقمی را در خروجی تولید کنند. از طرف دیگر، در جدول درستی و یا هنگام

جدول ۴-۴. جمع کننده کامل

x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



شکل ۶-۴. نقشه جمع‌کننده کامل به صورت جمع حاصلضرب‌ها

پیاده‌سازی با گیت‌های منطقی، همان مقادیر به عنوان متغیرهای بول تعبیر می‌شوند. نقشه خروجی‌های جمع‌کننده کامل در شکل ۶-۴ ملاحظه می‌شود. عبارات ساده شده عبارتند از:

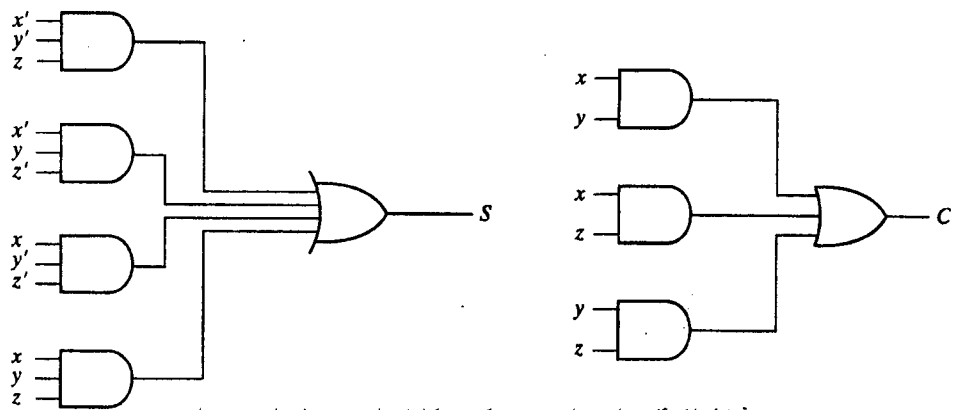
$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$C = xy + xz + yz$$

نمودار منطقی پیاده شده به صورت جمع حاصلضرب‌ها در شکل ۷-۴ مشاهده می‌شود. می‌توان با دو نیم جمع‌کننده و یک OR هم طبق شکل ۸-۴ آن را پیاده‌سازی کرد. خروجی S از نیم‌جمع‌کننده دوم XOR متغیر z و خروجی نیم‌جمع کننده اول حاصل می‌شود، زیرا

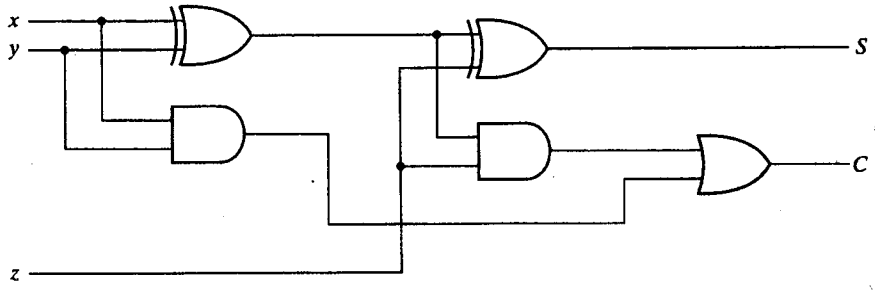
$$\begin{aligned} S &= z \oplus (x \oplus y) \\ &= z'(xy' + x'y) + z(xy' + x'y)' \\ &= z'(xy' + x'y) + z(xy + x'y') \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

و نقلی خروجی برابر است با

$$C = z(xy' + x'y) + xy = xy'z + x'yz + xy$$



شکل ۷-۴. پیاده‌سازی جمع کننده کامل با جمع حاصلضرب‌ها



شکل ۸-۴ . پیاده‌سازی یک جمع‌کننده کامل با دو نیم جمع‌کننده و یک گیت OR

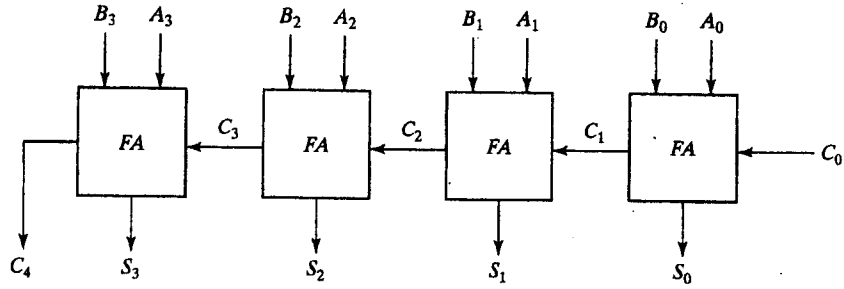
جمع‌کننده دودویی

یک جمع‌کننده دودویی مداری دیجیتال است که جمع حسابی دو عدد دودویی را تولید می‌کند. می‌توان آن را از به هم پیوستن متوالی جمع‌کننده کامل ساخت، و در آن هر خروجی نقلی از هر جمع‌کننده کامل به ورودی نقلی جمع‌کننده کامل بعدی زنجیروار بسته می‌شود. شکل ۹-۴ اتصالات درونی مدار چهار جمع‌کننده کامل (FA)، برای تهیه جمع‌کننده دودویی 4 بیت با نقلی موج‌گونه را نشان می‌دهد. بیت‌های مضاف از A و مضاف‌الیه از B با اعداد اندیس‌دار از راست به چپ و با اندیس 0 در بیت کم‌ارزش‌تر مشخص شده است. نقلی‌ها به صورت زنجیر جمع‌کننده‌های کامل را بهم وصل کرده‌اند. نقلی ورودی به جمع‌کننده C_0 وصل بوده و موج‌گونه‌وار تا نقلی خروجی C_4 انتشار می‌یابد. خروجی‌های S بیت‌های حاصل جمع را تولید می‌کنند. یک جمع‌کننده n بیت به n جمع‌کننده کامل نیاز دارد و هر خروجی نقلی به ورودی نقلی جمع‌کننده بالا رتبه‌تر وصل می‌شود.

به منظور تشریح بیشتر مثالی را با اعداد دودویی A = 1011 و B = 0011 در نظر بگیرید. حاصل جمع آنها S = 1110 است که از جمع چهار بیت مطابق زیر بدست می‌آید.

اندیس i :	3	2	1	0	
نقلی ورودی	0	1	1	0	C_i
مضاف	1	0	1	1	A_i
مضاف الیه	0	0	1	1	B_i
حاصل جمع	1	1	1	0	S_i
نقلی خروجی	0	0	1	1	C_{i+1}

بیت‌ها با کمک جمع‌کننده کامل و از کم‌ارزش‌ترین مکان (اندیس 0) با هم جمع می‌شوند تا بیت حاصل‌جمع ونقلی را تشکیل دهند. نقلی ورودی C_0 در کم‌ارزش‌ترین مکان باید 0 باشد. مقدار C_{i+1} در یک مکان مفروض، نقلی خروجی جمع‌کننده کامل است. این مقدار به نقلی ورودی تمام جمع‌کننده‌ای که بیت‌های یک مکان بالاتر در سمت چپ را جمع می‌کند انتقال می‌یابد. بنابراین بیت‌های جمع از



شکل ۹-۴. جمع‌کننده 4 بیت

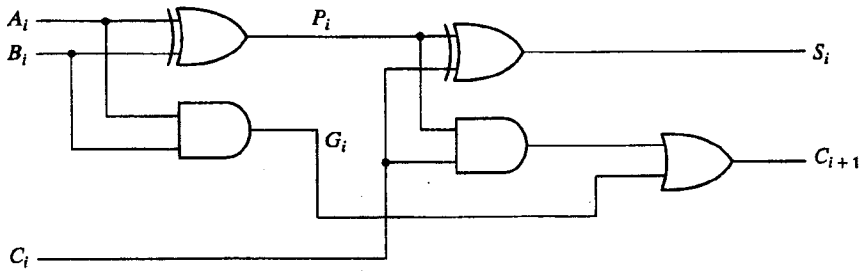
راست به چپ تولید شده و به محض تولید نقلی قبل از خود، در اختیار خواهند بود. برای داشتن خروجی جمع صحیح همه نقلی‌ها باید تولید شده باشند.

یک جمع‌کننده 4 بیت مثالی از یک قطعه استاندارد است. می‌توان از آن در کاربردهای متعددی مثل عملیات حسابی استفاده کرد. به خاطر بسپارید که طراحی این مدار با روشهای کلاسیک به یک جدول درستی با $2^9 = 512$ وارده نیاز دارد زیرا 9 ورودی به مدار موجود است. با استفاده از روش بستن متوالی یک تابع استاندارد می‌توان به یک پیاده‌سازی ساده و مستقیم دست یافت.

انتشار رقم نقلی

جمع دو عدد دودویی به صورت موازی لازم می‌دارد که مضاف و مضاف‌الیه به طور همزمان برای محاسبه موجود باشند. همچون دیگر مدارهای ترکیبی، در این مدار هم قبل از داشتن یک جواب صحیح، سیگنال باید از گیت‌ها عبور کند. زمان کل انتشار برابر است با زمان تأخیر انتشار یک نمونه گیت ضربدر طبقات گیت‌ها در مدار. طولانی‌ترین زمان تأخیر انتشار در یک جمع‌کننده زمانی است که نقلی برای انتشار در جمع‌کننده‌ای کامل لازم دارد. چون هر بیت از خروجی جمع به نقلی ورودی‌اش وابسته است مقدار S_i در هر طبقه مفروض در جمع‌کننده تنها موقعی به مقدار پایدار نهایی خود می‌رسند که نقلی ورودی به آن طبقه رسیده باشد. مثلاً خروجی S_3 را در شکل ۹-۴ در نظر بگیرید. به محض اعمال ورودی‌ها به جمع‌کننده، ورودی‌های A_3 و B_3 در دسترسند. با این وجود نقلی ورودی C_3 تا تولید C_2 به وسیله طبقه قبل در مقدار نهایی‌اش پایدار نمی‌شود. به طور مشابه C_2 منتظر C_1 و C_1 منتظر C_0 خواهد بود. بنابراین پس از انتشار موج‌گونه نقلی در همه طبقات، خروجی S_3 و نقلی C_4 در مقادیر صحیح نهایی خود پایدار خواهند شد.

تعداد طبقات گیت برای انتشار نقلی را باید از مدار هر جمع‌کننده کامل بدست آورد. به منظور سهولت این مدار در شکل ۱۰-۴ دوباره ترسیم شده است. متغیرهای ورودی و خروجی از اندیس i برای مشخص کردن شماره طبقه جمع‌کننده استفاده کرده‌اند. سیگنال‌ها در P_i و G_i هنگامی به ثبات می‌رسند که از گیت‌های مربوطه‌شان انتشار یافته باشند. این دو سیگنال که در همه جمع‌کننده‌های کامل وجود دارند، به بیت‌های مضاف و مضاف‌الیه ورودی وابسته‌اند. سیگنال نقلی ورودی C_i از طریق گیت AND و یک



شکل ۱۰-۴. جمع‌کننده کامل با P و G

گیت OR، که دو سطح گیت را تشکیل می‌دهند، به C_{i+1} می‌رسد. اگر در مجموع، چهار جمع‌کننده کامل وجود داشته باشد بین نقلی خروجی C_0 تا C_4، 8=4×2 طبقه گیت وجود خواهد داشت. برای یک جمع کننده n بیت، 2n طبقه گیت وجود دارد تا نقلی ورودی از طریق آنها انتشار یافته و به خروجی برسد.

زمان انتشار نقلی، فاکتور محدود کننده‌ای روی سرعت جمع دو عدد می‌باشد. گرچه جمع‌کننده، یا هر مدار ترکیبی دیگر دارای مقداری در پایانه‌اش هست، ولی خروجی‌ها صحیح نخواهند بود مگر این که فرصتی کافی برای انتشار سیگنال از گیت‌های متصل به هم از ورودی تا خروجی داده شود. چون همه عملیات حسابی با جمع تکراری صورت می‌گیرد، زمان مصرف شده در طول فرآیند جمع بسیار حساس خواهد بود. راه‌حل روشنی برای کاهش زمان تأخیر انتشار استفاده از گیت‌های سریع‌تر است. با این وجود، مدارهای فیزیکی در قابلیت خود محدودیت دارند. راه‌حل دیگر افزایش پیچیدگی مدار به طریقی است که زمان تأخیر نقلی کاهش یابد. چند تکنیک برای کاهش زمان انتشار نقلی در جمع‌کننده‌های موازی وجود دارد. رایج‌ترین تکنیک استفاده از اصل پیش‌بینی نقلی می‌باشد. مدار جمع‌کننده کامل شکل ۱۰-۴ را ملاحظه نمایید. اگر دو متغیر دودویی جدید زیر را معرفی کنیم:

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

حاصل جمع خروجی و نقلی آن را می‌توان به صورت زیر تعریف کرد.

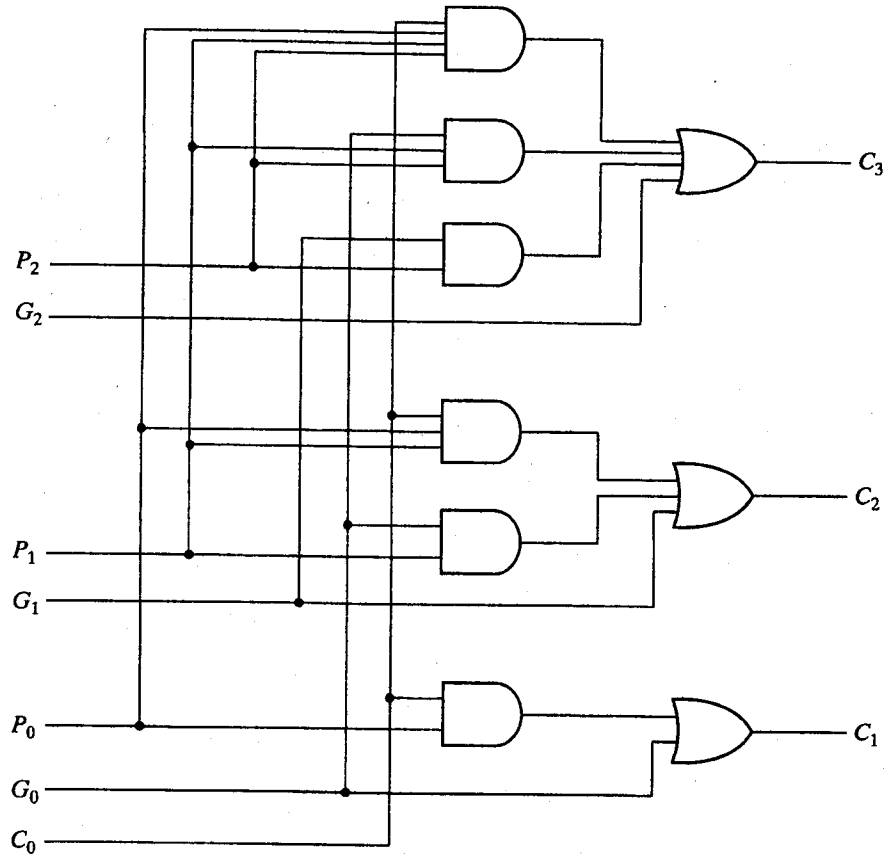
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

G_i را مولد نقلی نامند که وقتی هر دو A_i و B_i برابر 1 باشند، یک نقلی 1 را تولید می‌نماید، و این تولید مستقل از C_i می‌باشد. P_i را انتشار نقلی گویند زیرا جمله‌ای است که در انتشار نقلی از C_i به C_{i+1} نقش دارد.

اکنون توابع بول را برای خروجی‌های نقلی هر طبقه نوشته و هر C_i را با مقدار معادل قبلی جایگزین می‌کنیم:

$C_0 =$ نقلی ورودی

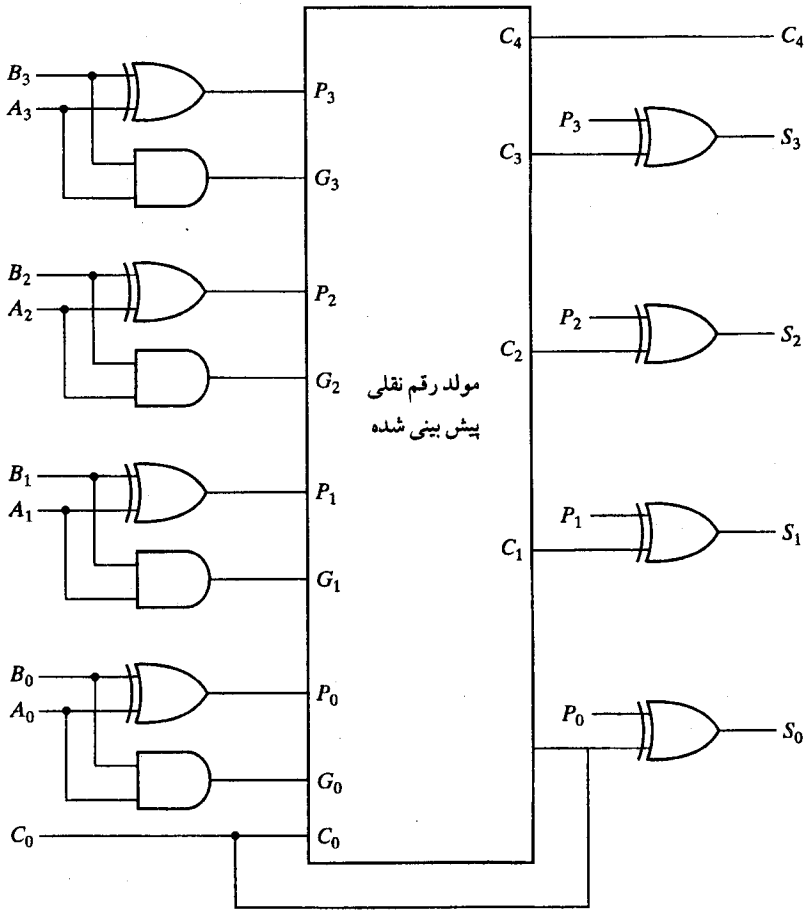
$$C_1 = G_0 + P_0 C_0$$
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$



شکل ۱۱-۴. نمودار منطقی مولد پیش‌بینی نقلی

چون تابع بول برای هر نقلی خروجی برحسب جمع حاصلضرب‌ها بیان شده است، هر تابع قابل پیاده‌سازی با یک طبقه گیت AND و به دنبال آن گیت OR (یا با دو طبقه NAND) است. سه تابع بول برای C_1 ، C_2 و C_3 در مولد نقلی پیش‌بینی شونده و در شکل ۱۱-۴ دیده می‌شوند. توجه کنید که C_3 نیاز ندارد به انتظار C_2 و C_1 بماند. در واقع C_3 با C_1 و C_2 همزمان منتشر می‌گردد.

ساخت یک جمع‌کننده 4 بیت با پیش بینی نقلی در شکل ۱۲-۴ مشاهده می‌شود. خروجی اولین گیت XOR متغیر P_i و گیت AND متغیر G_i را تولید می‌نماید. نقلی‌ها از درون مولد پیش بینی نقلی انتشار می‌یابند (مشابه شکل ۱۱-۴) و به عنوان ورودی به گیت XOR دوم اعمال می‌گردند. همه نقلی‌های خروجی پس از یک تأخیر در دو طبقه گیت به طور همزمان تولید می‌شوند. بنابراین S_1 تا S_3 دارای زمان تأخیر انتشار یکسانی هستند. مدار دو طبقه برای نقلی خروجی C_4 نشان داده نشده است. این مدار هم به سادگی با روش جایگزینی قابل دستیابی است.

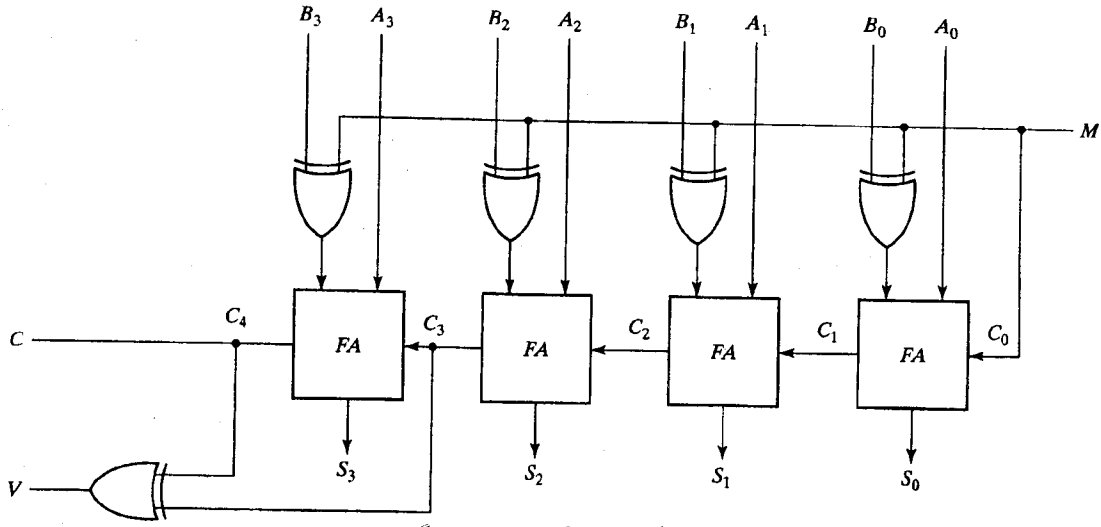


شکل ۱۲-۴. جمع‌کننده 4 بیت با پیش‌بینی نقلی

تفریق دودویی

در بخش ۵-۱ دیدیم که تفریق اعداد دودویی بی‌علامت با استفاده از متمم راحت‌تر انجام می‌گیرد. به خاطر دارید که A−B را می‌توان با محاسبه متمم 2 عدد B و جمع آن با A معین کرد. متمم 2 را با بدست آوردن متمم 1 و جمع آن با 1 محاسبه می‌کنند. متمم 1 را هم با وارونگر بدست آورده و عدد 1 را هم از طریق ورودی نقلی به آن اضافه می‌نمایند.

مدار تفریق‌گر A−B متشکل از یک جمع‌کننده با وارونگرهای واقع در بین ورودی B و ورودی مربوطه‌اش در تمام جمع‌کننده می‌باشد. نقلی ورودی C_0 به هنگام تفریق باید برابر با 1 شود. بنابراین عمل به صورت A بعلاوه متمم 1 عدد B بعلاوه 1 اجرا می‌شود. این عمل برابر با جمع A با متمم 2 عدد B خواهد بود. برای اعداد بی‌علامت اگر A≥B باشد، عمل فوق A−B و اگر A<B باشد (B−A) است. برای اعداد علامت‌دار، نتیجه A−B است به شرطی که سرریز وجود نداشته باشد (بخش ۶-۱ ملاحظه شود).



شکل ۱۳-۴ . جمع- تفریق‌گر

عملیات جمع و تفریق را می‌توان با یک مدار در هم ادغام کرده و با یک جمع‌کننده دودویی مشترک انجام داد. این کار با افزودن یک گیت XOR در هر جمع‌کننده کامل صورت می‌گیرد. یک مدار جمع- تفریق‌گر در شکل ۱۳-۴ دیده می‌شود. وقتی $M = 0$ است، مدار یک جمع‌کننده و وقتی $M = 1$ باشد، مدار یک تفریق‌گر خواهد بود. هر گیت XOR ورودی M و یکی از ورودی‌های B را دریافت می‌کند. وقتی $M = 0$ است داریم، $B \oplus 0 = B$. جمع‌کننده کامل B را دریافت می‌نماید، نقلی ورودی 0 است و بنابراین مدار عمل A بعلاوه B را اجرا می‌کند. اگر $M = 1$ باشد، $B \oplus 1 = B'$ بوده و $C_0 = 1$ است. ورودی‌های B همگی متمم شده و از طریق ورودی نقلی، یک 1 به آن اضافه می‌شود. در این حالت مدار یک عمل A بعلاوه متمم 2 عدد B را انجام می‌دهد. (XOR با خروجی V ، یک سرریز را شناسایی می‌نماید).

لازم است متذکر شویم که در سیستم متمم علامت‌دار، اعداد دودویی هم چون اعداد بی‌علامت، با قوانین جمع و تفریق یکسانی ترکیب می‌شوند. بنابراین، کامپیوترها نیاز به یک سخت‌افزار مشترک دارند تا هر دو نوع محاسبه را انجام دهند. کاربر یا برنامه‌نویس باید نتایج چنین جمع یا تفریقی را متفاوت تفسیر کنند و این به علامت‌دار یا بی‌علامت بودن اعداد بستگی دارد.

سرریز

هرگاه دو عدد n رقمی با هم جمع شوند و حاصل جمع n+1 رقم را اشغال کند، گوییم سرریز رخ داده است. این مطلب جدا از علامت‌دار بودن یا نبودن برای اعداد دهدهی یا دودویی صحیح است. وقتی که جمع با کاغذ و قلم انجام می‌شود، سرریز مسئله‌ای نیست زیرا محدودیتی برای عرض صفحه جهت نوشتن جمع وجود ندارد. ولی سرریز در کامپیوترهای دیجیتال مشکلاتی ایجاد می‌کند، زیرا تعداد

بیت‌های نگهدارنده عدد محدود بوده و نتیجه‌ای را که n+1 بیت دارد نمی‌توانند در خود جای دهند. به این دلیل، بسیاری از کامپیوترها وقوع یک سرریز را، اگر رخ دهد، شناسایی می‌کنند و فیلپ فلاپ مربوطه را در 1 می‌نشانند تا بعد به وسیله کاربر چک شود.

تشخیص یک سرریز پس از جمع دو عدد دودویی به این بستگی دارد که آیا اعداد علامت دارند یا بی‌علامت‌اند. وقتی دو عدد بی‌علامت با هم جمع شوند، یک سرریز از نقلی باارزش‌ترین مکان تشخیص داده می‌شود. در حالتی که اعداد علامت‌دار باشد، سمت چپ‌ترین بیت همواره علامت را نشان داده و اعداد منفی هم به صورت متمم 2 هستند. وقتی دو عدد علامت‌دار جمع شوند، با بیت علامت به عنوان بخشی از عدد رفتار می‌شود و رقم نقلی انتهایی هیچ سرریزی را مشخص نمی‌کند.

در جمع وقتی که یکی از اعداد مثبت و دیگری منفی باشد، سرریز رخ نمی‌دهد، زیرا جمع یک عدد مثبت با یک عدد منفی نتیجه‌ای تولید می‌کند که از بزرگترین آن دو کوچکتر است. سرریز هنگامی رخ می‌دهد که هر دو عدد جمع شونده مثبت یا منفی باشند. برای درک بهتر موضوع مثال زیر را ملاحظه کنید. دو عدد علامت‌دار دودویی 70+ و 80+ دودویی در دو ثبات 8 بیتی ذخیره شده‌اند. محدوده اعدادی که هر یک از ثبات‌ها داراست از 127+ تا 128− دودویی است. چون مجموع دو عدد 150+ است، حاصل از ظرفیت ثبات 8 بیتی تجاوز خواهد کرد. این مطالب هنگامی که هر دو عدد مثبت یا منفی باشند صحت دارد. دو جمع مذکور همراه با ارقام نقلی در زیر نشان داده شده‌اند:

نقلی‌ها :	0	1		نقلی‌ها :	1	0	
+70		0	1000110	−70		1	0111010
+80		0	1010000	−80		1	0110000
+150		1	0010110	−150		0	1101010

توجه کنید که حاصل جمع هشت بیتی که باید مثبت باشد یک بیت علامت منفی دارد و نتیجه 8 بیتی که باید منفی باشد دارای بیت علامت مثبت است. با این وجود اگر رقم نقلی خارج شده از بیت علامت به عنوان بیت علامت در نظر گرفته شود، آنگاه جواب 9 بیتی حاصل صحیح خواهد بود. چون پاسخ نمی‌تواند در 8 بیت جای داده شود، گوییم سرریز رخ داده است.

وضعیت سرریز را می‌توان با وجود رقم نقلی به بیت علامت و نقلی خروجی از بیت علامت مشاهده کرد. اگر این دو نقلی یکی نباشند، یک سرریز رخ داده است. این نکته در مثال‌های فوق که در آن دو نقلی به طور جداگانه نشان داده شده‌اند دیده می‌شود. اگر دو رقم نقلی را به یک گیت XOR اعمال کنیم، وقوع سرریز با 1 شدن خروجی این گیت شناسایی می‌شود. برای این که روش به خوبی کار کند متمم 2 باید از طریق بدست آوردن متمم 1 و جمع آن با 1 انجام گردد. این کار موجب مراقبت از حالتی می‌شود که در آن عدد منفی ماکزیمم متمم شود.

مدار جمع- تفریق‌گر با خروجی‌های C و V در شکل ۱۳-۴ دیده می‌شود. اگر دو عدد دودویی بی‌علامت تصور شوند، آنگاه بیت C، نقلی بعد از جمع یا قرض بعد از تفریق است. اگر اعداد علامت‌دار

فرض شوند، آنگاه بیت V یک سرریز را مشخص می‌کند. اگر $V = 0$ بعد از یک جمع یا تفریق باشد، بیانگر نبود سرریز بوده و نتیجه n بیتی حاصل صحیح است. اگر $V = 1$ باشد، در این صورت نتیجه عمل حاوی n+1 بیت می‌باشد، ولی بیت n+1ام علامت واقعی است که به یک مکان بیرونی منتقل شده است.

۴-۵ جمع‌کننده دهدهی

کامپیوترها یا ماشین‌های حسابی که اعمال محاسباتی را مستقیماً در سیستم اعداد دهدهی انجام می‌دهند، اعداد دهدهی را به فرم کد دودویی ارائه می‌کنند. یک جمع‌کننده در این نوع کامپیوترها ، از نوعی مدار محاسباتی استفاده می‌کند که اعداد دهدهی کد شده را می‌پذیرد و نتایج را در همان کد ارائه می‌نماید. برای جمع دودویی کافی است جفت بیت باارزش را همراه با رقم نقلی قبلی در نظر بگیرید. یک جمع‌کننده دهدهی به حداقل ده ورودی و پنج خروجی نیاز دارد زیرا برای کد هر رقم دهدهی چهار بیت لازم است و مدار باید ورودی و خروجی نقلی هم داشته باشد. برای انجام این‌گونه جمع، مدارهای جمع‌کننده دهدهی متعددی وجود دارند که انتخاب آنها به کد به کار رفته در نمایش ارقام دهدهی بستگی دارد. در اینجا ما جمع‌کننده دهدهی را برای کد BCD بررسی می‌کنیم.

جمع‌کننده BCD

جمع حسابی دو رقم دهدهی در BCD را همراه با یک رقم نقلی از مرحله قبل در نظر بگیرید. چون هر رقم ورودی از 9 تجاوز نمی‌کند، حاصل جمع خروجی از $19 = 1 + 9 + 9$ بیشتر نخواهد شد. عدد 1 در جمع فوق، نقلی ورودی است. فرض کنید که دو رقم BCD را به جمع‌کننده دودویی 4 بیتی اعمال نماییم. جمع‌کننده، حاصل جمع را به فرم دودویی اجرا می‌کند و نتیجهٔ تولید شده بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول (۵-۴) مشاهده می‌شود که با K ، z_8 ، z_4 ، z_2 و z_1 برچسب خورده‌اند. K یک رقم نقلی است و اندیس زیر حرف z وزن‌های 8 ، 4 ، 2 و 1 می‌باشند که به چهار بیت کد BCD تخصیص یافته‌اند. ستون زیر حاصل جمع دودویی، مقادیر دودویی ظاهر شده در خروجی‌های جمع‌کننده چهار بیت را نشان می‌دهد. حاصل جمع خروجی دو رقم دهدهی باید به فرم BCD درآید و نیز باید آن طور که در زیر ستون جمع BCD ملاحظه می‌شود ظاهر گردد. مسئله این است که برای تبدیل جمع دودویی به رقم BCD عدد که در ستون جمع BCD مشاهده می‌شود باید قانونی پیدا شود.

ضمن بررسی محتوای جدول، ملاحظه می‌شود که وقتی جمع دودویی برابر با یا کمتر از 1001 باشد، با عدد BCD نظیر خود برابر است، و بنابراین تبدیلی لازم نیست. وقتی جمع دودویی بزرگتر از 1001 باشد، نمایش بی‌اعتباری را برای BCD خواهیم داشت. افزایش دودویی 6 (0110) به جمع دودویی آن را به نمایش BCD صحیح تبدیل می‌کند، ضمن این که یک رقم نقلی نیز در صورت لزوم تولید خواهد کرد.

مدار منطقی برای تشخیص این اصلاح، می‌تواند از وارده‌های جدول حاصل گردد. واضح است که وقتی نقلی خروجی $K = 1$ باشد نیاز به اصلاح جمع دودویی وجود دارد. دیگر ترکیبات شش‌گانه از

جدول ۵-۴. طریقه طراحی جمع کننده BCD

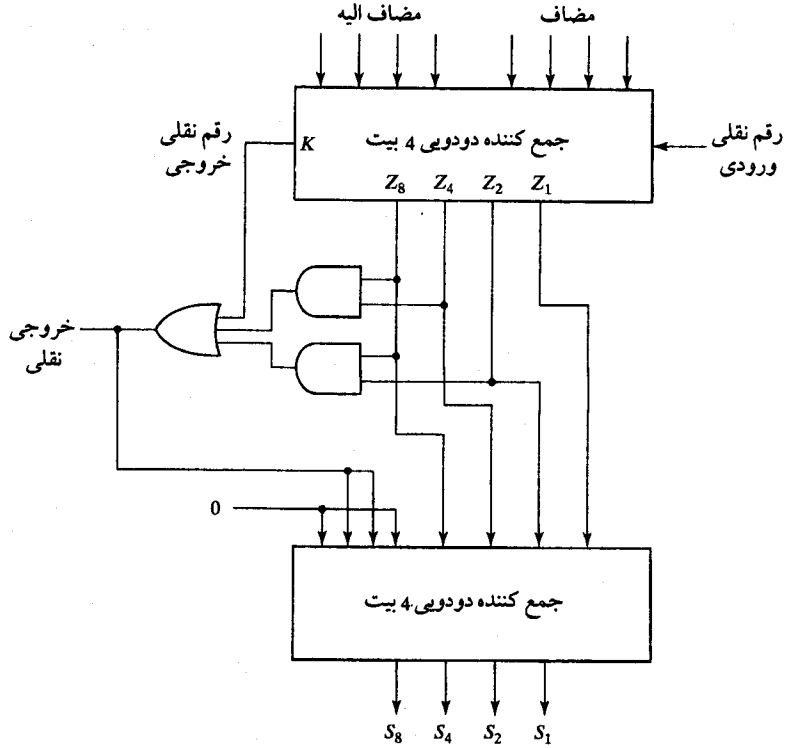
جمع دودویی					جمع BCD					دهدهی
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

1010 تا 1111 که به اصلاح نیاز دارند دارای 1 در مکان z_8 می‌باشند. برای تفکیک این شش حالت از 1000 و 1001، که آنها نیز دارای 1 در مکان z_8 هستند، به z_4 و z_2 مراجعه می‌کنیم که در هر حال حداقل یکی از آنها 1 است. به این ترتیب شرط اصلاح و داشتن یک نقلی خروجی را می‌توان با تابع بولی زیر بیان کرد:

$$C = K + Z_8Z_4 + Z_8Z_2$$

وقتی $C = 1$ است، لازم است 0110 به جمع دودویی اضافه شود تا یک نقلی خروجی برای طبقه بعدی فراهم شود.

یک جمع‌کننده BCD که دو رقم BCD را با هم جمع کرده و ارقام جمع را به BCD نشان می‌دهد در شکل ۱۴-۴ ملاحظه می‌گردد. دو رقم دهدهی همراه با نقلی ورودی ابتدا در جمع‌کننده 4 بیت فوقانی جمع شده و حاصل جمع دودویی تولید می‌کنند. وقتی نقلی خروجی برابر 0 باشد، چیزی به جمع دودویی اضافه نمی‌شود. وقتی این نقلی برابر 1 باشد، عدد دودویی 0110 از طریق جمع‌کننده 4 بیت پایینی به جمع دودویی اضافه می‌گردد. نقلی خروجی تولید شده در جمع‌کننده پایین می‌تواند صرف‌نظر شود زیرا اطلاعاتی را حمل می‌کند که قبلاً در پایانه نقلی خروجی وجود داشته است. یک جمع‌کننده دهدهی موازی که n رقم دهدهی را جمع می‌کند به n طبقه جمع‌کننده BCD نیاز دارد. نقلی خروجی هر طبقه باید به ورودی طبقه بالاتر متصل گردد.



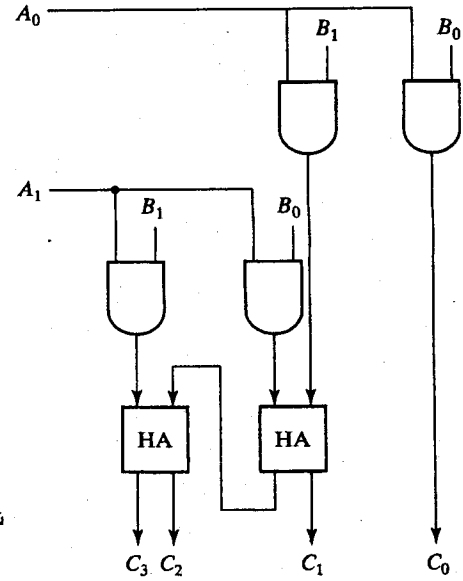
شکل ۱۴-۴. نمودار بلوکی یک جمع‌کننده BCD

۴-۶ ضرب دودویی

ضرب اعداد دودویی همچون ضرب اعداد دهدهی انجام می‌شود. هر بیت مضروب، در کم‌ارزش‌ترین بیت مضروب فیه ضرب می‌شود. چنین حاصلضربی، حاصلضرب جزیی خوانده می‌شود. حاصلضرب‌های جزیی هر بار یک مکان به چپ انتقال می‌یابند. حاصلضرب نهایی از جمع حاصلضرب‌های جزیی بدست می‌آید.

برای این که ببینیم که یک ضرب کننده چگونه با یک مدار ترکیبی پیاده می‌شود، ضرب اعداد دو بیت را طبق شکل ۱۵-۴ در نظر بگیرید. بیت‌های مضروب، B_0 و B_1 و بیت‌های مضروب‌فیه A_0 و A_1 و حاصلضرب $C_3C_2C_1C_0$ فرض می‌شوند. اولین حاصلضرب جزیی با ضرب A_0 در B_1B_0 حاصل می‌گردد. ضرب دو بیت مثل A_0 و B_0 هنگامی 1 تولید می‌کند که هر دوی آنها 1 باشند؛ در غیر این صورت 0 تولید خواهد کرد. این پاسخ مشابه با عمل AND است. بنابراین حاصل ضرب جزیی را می‌توان با گیت‌های AND مطابق شکل پیاده کرد. دومین حاصلضرب جزیی از ضرب A_1 در B_1B_0 بدست می‌آید که باید یک مکان هم به چپ جابجا شود. دو حاصلضرب جزیی به وسیله مدار دو نیم جمع‌کننده (HA) با هم جمع می‌شوند.

مضروب		B_1	B_0
مضروب فیه		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0



شکل ۱۵-۴. ضرب دودویی 2 بیت در 2 بیت

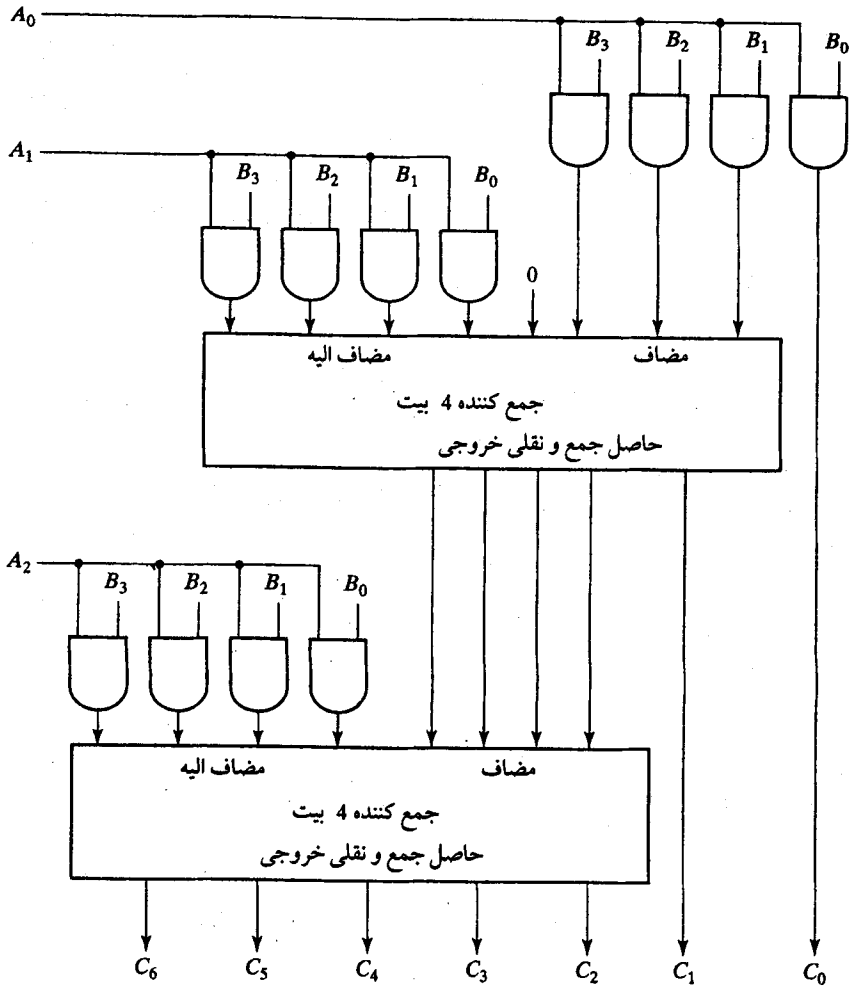
معمولاً در حاصلضرب‌های جزیی بیت‌های بیشتری وجود دارند و لازم است از تمام جمع‌کننده برای تولید جمع حاصلضرب‌های جزیی استفاده شود. توجه کنید لزومی ندارد که کم‌ارزش‌ترین بیت از جمع‌کننده عبور کند زیرا با خروجی اولین گیت AND، تشکیل شده است.

به طریقی مشابه می‌توان یک مدار ترکیبی ضرب دودویی با بیت‌های بیشتر ساخت. هر بیت از مضروب‌فیه در بیت‌های مضروب، AND می‌گردد. خروجی دودویی در هر سطحی از گیت‌های AND با حاصلضرب جزیی سطح قبلی برای تشکیل حاصلضرب جزیی جدید جمع می‌شود. آخرین سطح حاصلضرب کل را تولید می‌کند. برای J بیت مضروب فیه و K بیت مضروب به (J × K) گیت AND و (J−1) عدد جمع‌کننده K بیت نیاز است تا حاصلضرب J + K بیتی تولید شود.

به عنوان دومین مثال مدار ضرب کننده‌ای را ملاحظه نمایید که یک عدد دودویی 4 بیتی را در یک عدد 3 بیتی ضرب می‌کند. فرض کنید مضروب با $B_3B_2B_1B_0$ و مضروب‌فیه $A_2A_1A_0$ باشد. چون $K = 4$ و $J = 3$ است 12 گیت AND و دو جمع‌کننده 4 بیت برای تولید حاصلضرب 7 بیتی لازم است. نمودار منطقی ضرب کننده در شکل ۱۶-۴ دیده می‌شود.

۴-۷ مقایسه‌گر مقدار

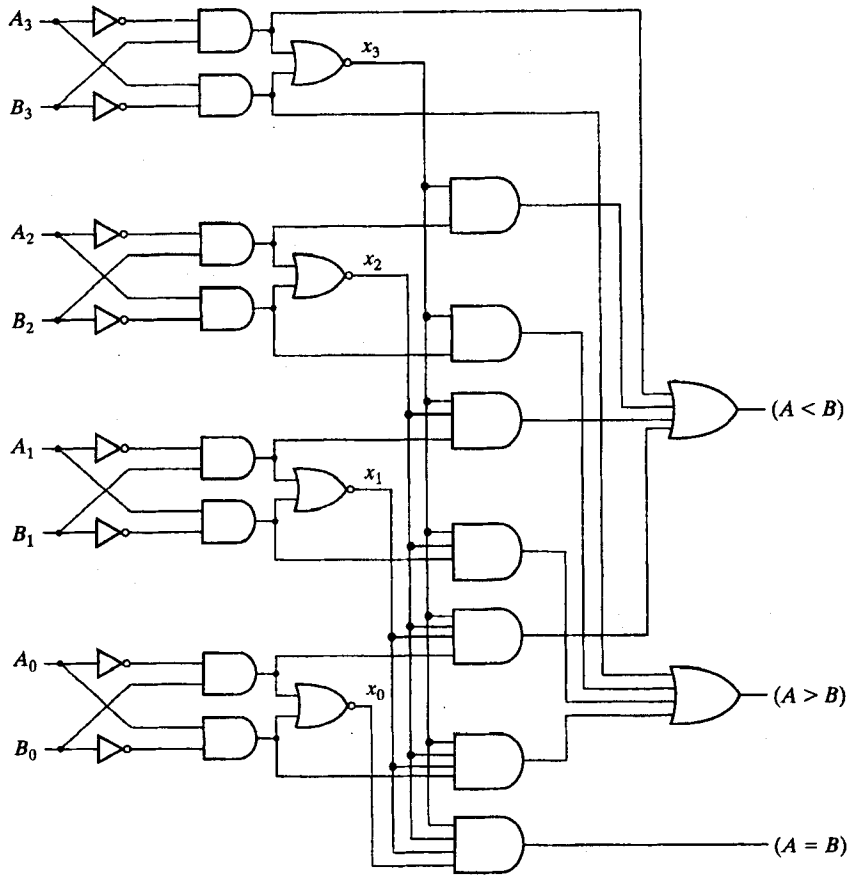
مقایسه دو عدد عملی است که توسط آن بزرگتر بودن، کوچکتر بودن یا مساوی بودن آنها معین می‌شود. یک مقایسه گر مقدار مداری ترکیبی است که دو عدد A و B را مقایسه می‌نماید و اندازه نسبی آنها را تعیین می‌کند. نتیجه این مقایسه با سه متغیر دودویی که بیانگر $A > B$ یا $A < B$ می‌باشد، مشخص می‌گردد.



شکل ۱۶-۴. ضرب کننده دودویی 4 بیت در 3 بیت

مدار مقایسه دو عدد n بیت 2^{2^n} وارده در جدول دارد و حتی با n=3 خسته کننده خواهد شد. از طرف دیگر، یک مدار مقایسه‌گر ممکن است مقدار قابل توجهی نظم در خود داشته باشد. توابع دیجیتال که ذاتاً دارای نظمی درونی هستند، معمولاً با روال‌های الگوریتمی قابل طراحی‌اند. یک الگوریتم روالی است که مجموعه مراحل معینی را مشخص می‌نماید و اگر دنبال شوند، از آن حلی برای مسئله حاصل می‌گردد. ما در اینجا از این روش برای ارائه یک الگوریتم جهت پیاده‌سازی مقایسه‌گر مقدار 4 بیتی استفاده خواهیم کرد.

در اینجا الگوریتم، کاربرد مستقیم روالی است که فرد برای مقایسه نسبی اندازه‌های دو عدد به کار می‌برد. دو عدد A و B را که هر کدام چهار رقم دارند در نظر بگیرید. ضرایب اعداد را به ترتیب نزولی زیر



شکل ۱۷-۴. مقایسه گر مقدرا چهار بیتی

هم می‌نویسیم:

$$A = A_3A_2A_1A_0$$
$$B = B_3B_2B_1B_0$$

هر حرف اندیس‌دار یک رقم را در عدد نشان می‌دهد. دو عدد هنگامی مساوی‌اند که همه جفت ارقام متناظر با هم برابر باشند: یعنی $A_3 = B_3$، $A_2 = B_2$، $A_1 = B_1$ و $A_0 = B_0$. وقتی که اعداد دودویی باشند، ارقام 1 یا 0اند و رابطه تساوی هر جفت بیت به طور منطقی با یک تابع *XOR* نمایش داده می‌شود.

$$x_i = A_iB_i + A_i'B_i' \qquad \text{for } i = 0, 1, 2, 3$$

که در آن $x_i = 1$ به شرطی صحت دارد که بیت‌های مکان iام برابر باشند (یعنی اگر هر دو 1 یا هر دو 0).

برابری دو عدد A و B در مدار ترکیبی با یک متغیر خروجی و با علامت (A = B) نشان داده می‌شود. این متغیر دودویی هنگامی 1 است که همه اعداد ورودی A و B مساوی باشند، در غیر این صورت 0 است. برای این که شرایط برابری برقرار باشد همه متغیرهای x_i باید برابر 1 شوند. در این

صورت AND همه متغیرها دیکته خواهد شد:

$$(A = B) = x_3 x_2 x_1 x_0$$

عدد دودویی (A = B) هنگامی 1 است که فقط همه جفت ارقام دو عدد برابر باشند.

برای این که معین کنیم آیا A بزرگتر یا کوچکتر از B است، اندازه‌های نسبی دو رقم را با شروع از باارزش‌ترین مکان آغاز می‌نماییم. اگر دو رقم مساوی باشند، دو رقم پایین‌تر را مقایسه می‌کنیم. این مقایسه تا رسیدن به یک جفت غیرمساوی ادامه خواهد داشت. اگر در این هنگام بیت متعلق به A برابر 1 و B برابر 0 باشد، نتیجه می‌گیریم A > B است. اگر برعکس رقم مربوط به A برابر 0 و B برابر با 1 باشد، خواهیم داشت A < B. مقایسه فوق را می‌توان با کمک دو تابع بولی به صورت زیر نوشت:

$$(A > B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$
$$(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

سمبل‌های (A > B) و (A < B) متغیرهای خروجی دودویی هستند که بترتیب هنگام A > B یا A<B برابر 1 می‌شوند.

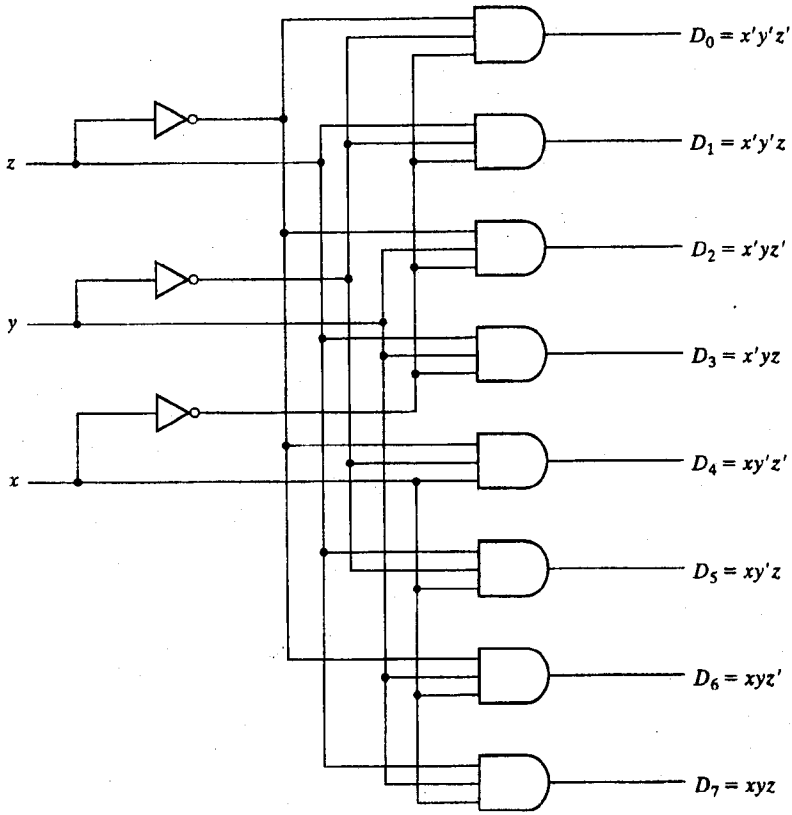
پیاده‌سازی گیتی سه متغیر خروجی ساده‌تر از آنچه به نظر می‌رسد انجام می‌شود زیرا شامل مقدار قابل توجهی اعمال تکراری است. خروجی‌های نامساوی می‌توانند از گیت‌هایی که برای تولید خروجی مساوی لازم بود استفاده کنند. نمودار منطقی یک مقایسه‌گر مقدار 4 بیتی در شکل ۱۷-۴ ملاحظه می‌شود. چهار خروجی x با مدارهای XNOR تولید شده و به گیت AND اعمال شده‌اند تا متغیر دودویی خروجی (A = B) تولید گردد. دو خروجی دیگر از متغیر x برای تولید توابع بولی لیست شده قبلی استفاده می‌کنند. این یک پیاده‌سازی چند طبقه است که الگوی منظمی دارد. روال برای بدست آوردن مدارهای مقایسه‌گر اندازه برای اعداد دودویی با بیش از چهار بیت از این مثال کاملاً آشکار است.

۸-۴ دیکدرها

کمیت‌های گسسته اطلاعاتی در سیستم‌های دیجیتال با کدهای دودویی نشان داده می‌شوند. یک کد دودویی n بیتی قادر است تا 2^n عنصر گسسته اطلاعات کد شده را نشان دهد. یک دیکدر مداری ترکیبی است که اطلاعات دودویی را از n خط ورودی به حداکثر 2^n خط خروجی منحصر به فرد تبدیل می‌کند اگر کد n بیتی دارای ترکیبات بی‌استفاده باشد، دیکدر ممکن است خروجی‌های کمتر از 2^n داشته باشد.

دیکدرهایی که در اینجا ارائه شده‌اند دیکدرهای n به m خوانده می‌شوند که $m \leq 2^n$ است. هدف از آنها تولید 2^n مینترم (یا کمتر) از n متغیر ورودی است. نام دیکدر همراه با دیگر مبدل‌های کد مانند دیکدر BCD به هفت قسمتی هم به کار می‌رود.

به عنوان مثال دیکدر 3 به 8 قسمتی شکل ۱۸-۴ را ملاحظه نمایید. سه ورودی به هشت خروجی دیکدر شده است که هر یک نمایشگر یکی از مینترم‌های متعلق به سه متغیر ورودی است. سه وارونگر، متمم ورودی‌ها را تهیه کرده و هشت گیت AND هر کدام یک مینترم تولید می‌کنند. کاربرد رایج این نوع دیکدر، تبدیل دودویی به هشت هشتی است. متغیرهای ورودی یک عدد دودویی را نشان می‌دهند، و



شکل ۱۸-۴. دیکدر 3 به 8

خروجی بیانگر هشت رقم در سیستم اعداد مبنای هشت است. با این وجود دیکدر 3 به 8 خط را می‌توان برای دیکد کردن هر کد 3 بیت در تولید هشت خروجی، یکی برای هر عنصر از کد، به کار برد.

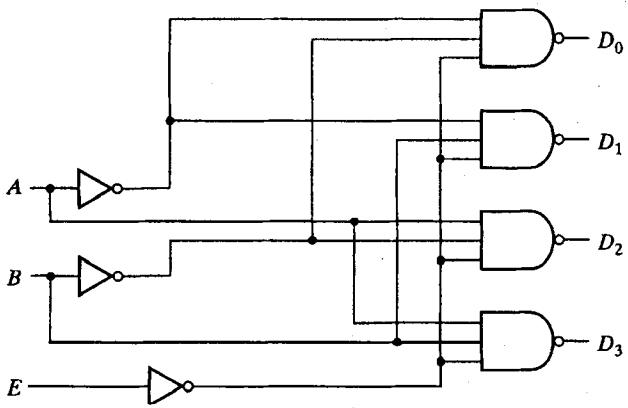
طرز کار یک دیکدر می‌تواند با لیستی در جدول درستی ۶-۴ آشکار شود. برای هر ترکیب ورودی

جدول ۶-۴. جدول درستی دیکدر 3 به 8 خط

ورودی‌ها			خروجی ها							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(ب) جدول درستی



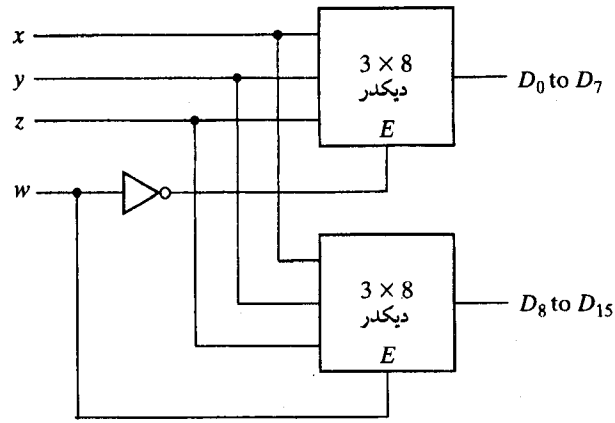
(الف) دیاگرام منطقی

شکل ۱۹-۴. دیکدر 2 به 4 خط با ورودی فعال ساز

ممکن، هفت خروجی وجود دارد که برابر 0 هستند و فقط یکی از آنها برابر 1 است. خروجی حاوی 1 بیانگر مینترم عدد دودیی حاضر در خطوط ورودی است.

بعضی از دیکدرها با گیت‌های NAND ساخته می‌شوند. چون گیت NAND عمل AND را با یک خروجی معکوس تولید می‌کند، تولید مینترم‌های دیکدر در شکل متمم اقتصادی‌تر است. به علاوه، دیکدر معمولاً دارای یک یا دو ورودی تواناساز یا فعال‌ساز برای کنترل کار مدار می‌باشند. یک دیکدر 2 به 4 با یک ورودی فعال‌ساز که با گیت‌های NAND ساخته شده در شکل ۱۹-۴ دیده می‌شود. مدار با خروجی‌های متمم شده و یک ورودی فعال‌ساز متمم شده کار می‌کند. دیکدر هنگامی که E برابر 0 باشد فعال می‌گردد. همانطور که توسط جدول درستی مشاهده می‌شود، هر بار تنها یک خروجی برابر 0 بوده و دیگر خروجی‌ها در وضعیت 1 قرار دارند. وقتی $E = 1$ باشد مدار غیرفعال است و به دو ورودی دیگر بستگی ندارد. هنگام غیرفعال شدن مدار، هیچ یک از خروجی‌ها در 0 نبوده و هیچ یک از مینترم‌ها انتخاب نمی‌شوند. به طور کلی، یک دیکدر ممکن است خروجی‌های متمم شده یا متمم نشده داشته باشد. ورودی فعال‌ساز ممکن است با سیگنال 0 یا 1 فعال گردد. بعضی از دیکدرها دارای دو یا چند ورودی فعال‌ساز می‌باشند که باید یک شرط منطقی مفروضی را برآورده سازند تا مدار فعال شود.

یک دیکدر با ورودی فعال‌ساز می‌تواند به عنوان یک دی‌مولتی پلکسر عمل کند. دی‌مولتی پلکسر مداری است که اطلاعات را از یک خط دریافت کرده و آن را به یکی از 2^n خط خروجی ممکن هدایت می‌نماید. انتخاب یک خروجی خاص با ترکیب بیتی n خط انتخاب صورت می‌گیرد. دیکدر شکل ۱۹-۴ را می‌توان به عنوان یک دی‌مولتی پلکسر 1 به 4 به کار برد. در این مدار E به عنوان ورودی داده و A و B ورودی‌های انتخاب هستند. تنها متغیر ورودی E مسیری به تمام چهارخروجی دارد، ولی اطلاعات ورودی تنها به یکی از خروجی‌ها هدایت می‌شود. این خروجی با ترکیب دودویی دو خط



شکل ۲۰-۴. دیکدر 16 × 4 ساخته شده با دو دیکدر 8 × 3

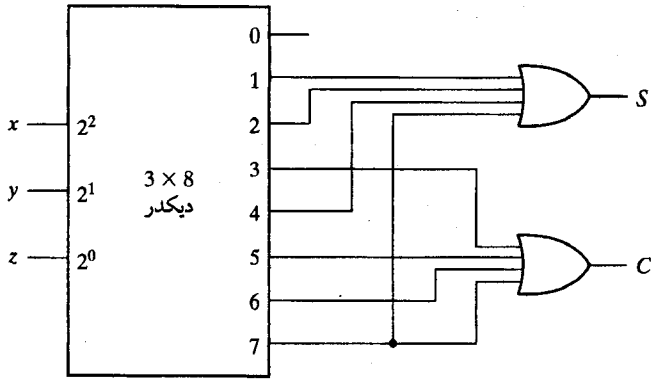
انتخاب A و B انتخاب می‌گردد. می‌توان انتخاب مسیر را از جدول درستی تحقیق کرد. مثلاً اگر خطوط انتخابی AB = 10 باشند، خروجی D_2 مثل ورودی E خواهد بود در حالی که دیگر خروجی‌ها در 1 نگهداشته خواهند شد. چون عمل دیکدر و دی‌مولتی پلکسر با استفاده از یک مدار حاصل می‌شود، یک دیکدر با ورودی فعال‌ساز را دیکدر/ دی‌مولتی پلکسر هم می‌خوانند.

برای تهیه مدار دیکدر بزرگتر می‌توان دیکدرها با ورودی‌های فعال‌ساز را به هم متصل کرد. شکل ۲۰-۴ دو دیکدر 3 به 8 را با ورودی‌های فعال‌ساز به هم پیوسته برای تشکیل یک دیکدر 4 به 16 خط نشان می‌دهد. وقتی $w = 0$ است، دیکدر فوقانی فعال می‌شود و دیگری غیرفعال است. خروجی‌های دیکدر پایینی همگی در 0 خواهند بود و هشت خروجی بالایی مینترم‌های 0000 تا 0111 را تولید می‌کنند. وقتی $w = 1$ باشد، وضعیت فعال شدن معکوس میگردد. خروجی‌های دیکدر پایینی مینترم‌های 1000 تا 1111 را تولید می‌نمایند، در حالی که خروجی‌های فوقانی همه 0 هستند. این مثال حسن ورودی‌های فعال‌ساز را در دیکدرها و دیگر قطعات منطقی ترکیبی نشان می‌دهد. به طورکلی ورودی‌های فعال‌ساز ابزارهای مناسبی برای اتصالات درونی دو یا چند قطعه استاندارد برای گسترش آنها با عملکردی مشابه و ورودی‌ها و خروجی‌های بیشتر است.

پیاده‌سازی مدار منطقی ترکیبی

یک دیکدر، 2^n مینترم را برای n متغیر ورودی تهیه می‌کند. چون هر تابع بولی می‌تواند برحسب جمع مینترم‌ها بیان شود، می‌توان از دیکدر برای تولید مینترم استفاده کرده و با یک گیت OR بیرونی جمع منطقی آنها را تشکیل داد. به این ترتیب هر مدار ترکیبی n ورودی و m خروجی با یک دیکدر n به 2^n و m گیت OR قابل پیاده‌سازی است.

روال پیاده‌سازی یک مدار ترکیبی با دیکدر و گیت‌های OR لازم می‌دارد که تابع بول مدار برحسب



شکل ۲۱-۴. پیاده‌سازی یک جمع‌کننده کامل با دیکدر

جمع مینترم‌ها بیان شود. سپس یک دیکدر برای تولید همه مینترم‌های حاصل از متغیرهای ورودی انتخاب می‌گردد. ورودی‌های هر گیت OR از خروجی‌های دیکدر برحسب لیست مینترم هر تابع انتخاب می‌گردند. این روال با مثالی که مدار جمع‌کننده کامل را به کار می‌برد تشریح می‌گردد.

با توجه به جدول درستی جمع‌کننده کامل (جدول ۴-۴)، توابع مدار ترکیبی را به صورت مجموع مینترم‌ها بدست می‌آوریم:

$$S(x, y, z) = \sum(1, 2, 4, 7)$$
$$C(x, y, z) = \sum(3, 5, 6, 7)$$

چون سه ورودی و جمعاً هشت مینترم وجود دارد، به یک دیکدر 3 به 8 خط احتیاج است. پیاده‌سازی در شکل ۲۱-۴ ملاحظه می‌گردد. دیکدر هشت مینترم را برای x، y و z تولید می‌کند. گیت OR برای خروجی S، جمع منطقی مینترم‌های 1، 2، 4 و 7 را تشکیل می‌دهد. گیت OR جمع منطقی مینترم‌های 3، 5، 6 و 7 را برای تولید خروجی C بکار می‌برد.

یک تابع با لیست طویلی از مینترم‌ها نیاز به یک گیت OR با ورودی‌های متعدد دارد. تابعی که k مینترم دارد می‌تواند به فرم متمم خود، F′، با $2^n - k$ مینترم نشان داده شود. اگر تعداد مینترم‌های موجود در تابع بزرگتر از $2^n/2$ باشد، آنگاه می‌توان F′ را با تعداد مینترم کمتری بیان کرد. در چنین وضعیتی، استفاده از گیت NOR برای تشکیل جمع مینترم‌های F′ مزیت دارد. خروجی گیت NOR این جمع را متمم کرده و تولید خروجی نرمال F را خواهد کرد. اگر از گیت‌های NAND، مثل شکل ۱۹-۴ استفاده شود، آنگاه گیت‌های خروجی در عوض OR باید از نوع NAND باشند. دلیل این است که یک مدار گیتی دو طبقه NAND تابع جمع مینترم‌ها را پیاده‌سازی می‌کند و معادل با مدار دو طبقه AND-OR است.

۹-۴ انکدرها

یک انکدر مداری است که عمل عکس یک دیکدر را انجام می‌دهد. یک انکدر دارای 2^n (یا کمتر) خط ورودی و n خط خروجی است. خطوط خروجی کد دودویی مربوط به مقدار دودویی ورودی را

تولید می‌نمایند.

مثالی از یک انکدر، انکدر هشت هشتی به دودویی است که جدول درستی آن در جدول ۷-۴ داده شده است. این مدار دارای هشت ورودی (یک ورودی برای هر رقم هشت هشتی) و سه خروجی است که عدد دودویی مربوطه را تولید می‌نماید. فرض بر این است که در هر لحظه‌ای از زمان تنها یک ورودی مقدار 1 را داشته باشد.

انکدر را می‌توان با گیت‌های OR که ورودی‌هایشان مستقیماً از جدول درستی تهیه می‌شود، پیاده‌سازی کرد. خروجی z هنگامی 1 است که رقم هشت هشتی ورودی در 1، 3، 5 یا 7 برابر 1 باشد. خروجی y به ازاء ارقام هشت هشتی ورودی 2، 3، 6 و 7، برابر 1 می‌شود. این شرایط را می‌توان با معادلات بولی خروجی بیان کرد.

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

این انکدر با سه گیت OR قابل پیاده‌سازی است.

انکدری که در جدول ۷-۴ تعریف شد دارای این محدودیت است که در آن در هر لحظه از زمان فقط یک ورودی فعال می‌باشد. اگر دو ورودی به طور همزمان فعال شوند، خروجی ترکیبات نامفهومی را تولید خواهد کرد. مثلاً اگر D_3 و D_6 همزمان برابر 1 شوند، خروجی انکدر 111 خواهد شد زیرا در این حالت هر سه خروجی برابر 1 است. این خروجی نه 3 و نه 6 را نمایش می‌دهد. برای حل این مشکل، مدارهای انکدر باید اولویتی در ورودی ایجاد کنند تا مطمئن شویم که فقط یک ورودی انکد شده است. اگر اولویت بالاتر را با اندیس‌های بالاتر ایجاد نماییم، و اگر در یک زمان D_6 و D_3 برابر 1 شوند، خروجی 110 می‌شود زیرا، D_6 اولویت بالاتری نسبت به D_3 دارد.

مشکل دیگری که در انکدر هشت هشتی به دودویی وجود دارد این است که در آن یک خروجی تمام 0 به ازاء حالتی که همه ورودی‌ها 0 هستند تولید می‌شود. این خروجی برابر با حالتی است که در آن $D_0 = 1$ است. ایراد را می‌توان با تهیه یک خروجی بیشتر حل کرد تا به این ترتیب نشان دهد که حداقل یک ورودی 1 است.

جدول ۷-۴. جدول درستی یک انکدر هشت هشتی به دودویی

ورودی‌ها								خروجی‌ها		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

جدول ۸-۴ . جدول درستی انکدر اولویت

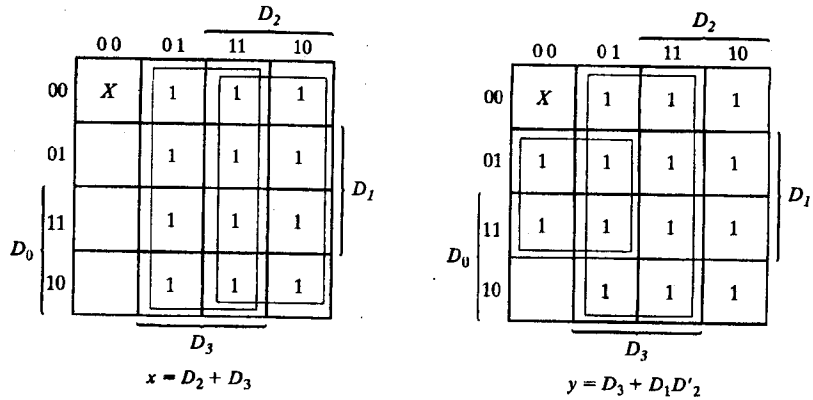
ورودی‌ها				خروجی ها		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

انکدر اولویت

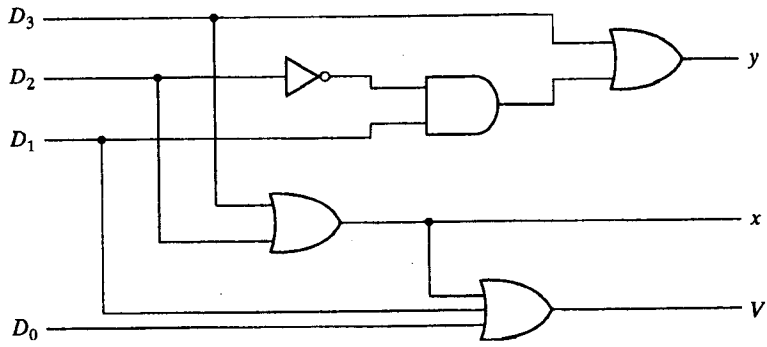
انکدر اولویت مداری است که تابع اولویت‌دهی را در خود دارد. عملکرد این نوع انکدر چنان است که اگر دو یا چند ورودی به طور همزمان برابر 1 شوند، ورودی با اولویت بالاتر پیش خواهد افتاد. جدول درستی یک انکدر با تقدم چهار ورودی در جدول ۸-۴ ملاحظه می‌گردد. علاوه بر دو خروجی x و y، مدار دارای سومین خروجی با علامت V است؛ که به معنی بیت معتبر می‌باشد و هرگاه یک یا چند ورودی برابر 1 شوند این خروجی 1 می‌گردد. اگر همه ورودی‌ها 0 باشند بیت معتبر وجود نخواهد داشت و $V = 0$ است. وقتی $V = 0$ باشد در خروجی دیگر وارسی نمی‌شوند و حالت بی‌اهمیت را خواهند داشت. توجه کنید که گرچه حالات بی‌اهمیت در ستون‌های خروجی اهمیت ندارند ولی X ها در ستون ورودی‌ها درکاهش جدول درستی نقش عمده‌ای دارند. به جای ذکر هر 16 مینترم برای چهار متغیر، جدول درستی از X که می‌تواند 0 یا 1 باشد استفاده می‌کند. مثلاً X100 دو مینترم 1100 ، 0100 را می‌پوشاند.

طبق جدول ۸-۴، هر عدد با مقدار بالاتر اولویت بالاتری را داراست. در این جدول D_3 اولویت بالاتری دارد، بنابراین بدون توجه به مقادیر دیگر ورودی‌ها، وقتی این ورودی 1 شود، خروجی xy برابر 11 می‌گردد (دودویی 3). D_2 دارای اولویت بعدی است. اگر $D_2 = 1$ شود خروجی 10 می‌گردد به شرطی که $D_3 = 0$ باشد، ولی این خروجی مستقل از دو ورودی با اولویت پایین‌تر است. اگر دیگر ورودی‌های با اولویت‌تر 0 باشند، خروجی مربوط به D_1 تولید می‌گردد و به همین ترتیب.

نقشه ساده‌سازی خروجی‌های x و y در شکل ۲۲-۴ نشان داده شده است. مینترم‌های دو تابع از



شکل ۲۲-۴ . نقشه یک انکدر اولویت‌دار



شکل ۲۳-۴ . پیاده‌سازی تابع بول با یک مولتی پلکسر

جدول ۷-۴ استنتاج شده‌اند. گرچه جدول دارای تنها پنج سطر است، وقتی هر X در هر سطر ابتدا با صفر و سپس با 1 جایگزین شود، آنگاه تمام 16 ترکیب ورودی را خواهیم داشت. مثلاً سطر چهارم در جدول با XX10 چهار مینترم 0010 ، 0110 ، 1010 و 1110 را نمایش می‌دهد. عبارات بول ساده شده برای انکدر اولویت از نقشه‌ها حاصل می‌شود. شرط خروجی V یک تابع OR از همه متغیرهای ورودی است. انکدر اولویت بر طبق توابع بولی زیر در شکل ۲۳-۴ پیاده‌سازی شده است.

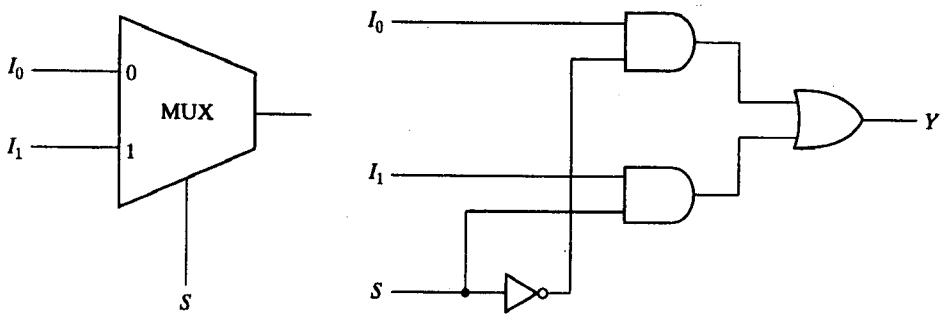
$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$

۴-۱۰ مولتی پلکسرها

یک مولتی پلکسر مداری ترکیبی است که اطلاعات دودویی را از تعدادی خط ورودی دریافت کرده و آنها را به یک خط خروجی هدایت می‌نماید. انتخاب یک ورودی خاص به وسیله مجموعه‌ای از خطوط انتخاب انجام می‌شود. معمولاً 2^n خط ورودی و n خط انتخاب وجود دارد و ترکیب بیتی تعیین کننده ورودی انتخاب شده است.

یک مولتی پلکسر 2 به 1 یکی از دو منبع 1 بیت را طبق شکل ۲۴-۴ به یک مقصد مشترک متصل می‌کند. مدار دارای دو خط ورودی داده، یک خروجی و یک خط انتخاب S است. وقتی S = 0 باشد، گیت AND فوقانی فعال شده و I_0 به خروجی راه می‌یابد. وقتی S = 1 باشد، گیت AND تحتانی فعال شده و I_1 به خروجی متصل می‌شود. مولتی پلکسر مثل یک کلید الکترونیک عمل کرده و یکی از دو منبع را انتخاب می‌نماید. نمودار بلوکی یک مولتی پلکسر گاهی به شکل ذوزنقه شکل ۲۴-۴ (ب) نشان داده می‌شود. این مدار چگونگی انتخاب و هدایت منابع متعدد داده را به یک مقصد نشان می‌دهد. مولتی پلکسر اغلب با نمودارهای بلوکی و کلمه MUX نشان داده می‌شود.

یک مولتی پلکسر 4 به 1 در شکل ۲۵-۴ دیده می‌شود. هر یک از چهار ورودی I_0 تا I_3 به یک



(ب) نمودار بلوکی

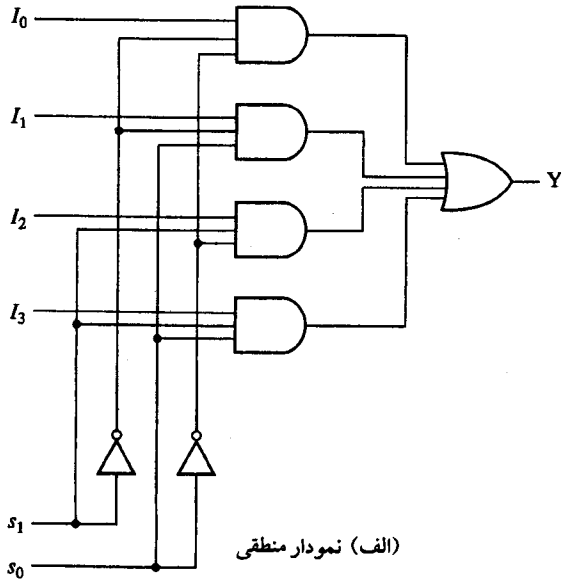
(الف) نمودار منطقی

شکل ۲۴-۴. مولتی‌پلکسر 2 به 1

ورودی گیت AND اعمال می‌شود. خطوط انتخاب S_1 و S_0 برای انتخاب گیت AND خاص دیکد می‌شوند. خروجی گیت‌های AND به یک گیت OR اعمال می‌شوند تا خروجی 1 خط را ایجاد کنند. جدول تابع، ورودی راکه از مولتی پلکسر عبور کرده نشان می‌دهد. برای نمایش عمل مدار، حالتی راکه $S_1S_0 = 10$ است ملاحظه کنید. گیت مربوط به ورودی I_2 دارای دو ورودی 1 و یک ورودی متصل به I_2 است. سه گیت دیگر هر یک حداقل یک 0 در ورودی خود دارند و بنابراین خروجی‌شان 0 می‌شود. خروجی گیت OR ، اکنون برابر مقدار I_2 است و به این ترتیب مسیری از ورودی انتخابی به خروجی ایجاد شده است. یک مولتی پلکسر را انتخابگر داده هم می‌خوانند، زیرا یکی از چند ورودی را انتخاب کرده و اطلاعات دودویی را به خط خروجی هدایت می‌کند.

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(ب) جدول تابع

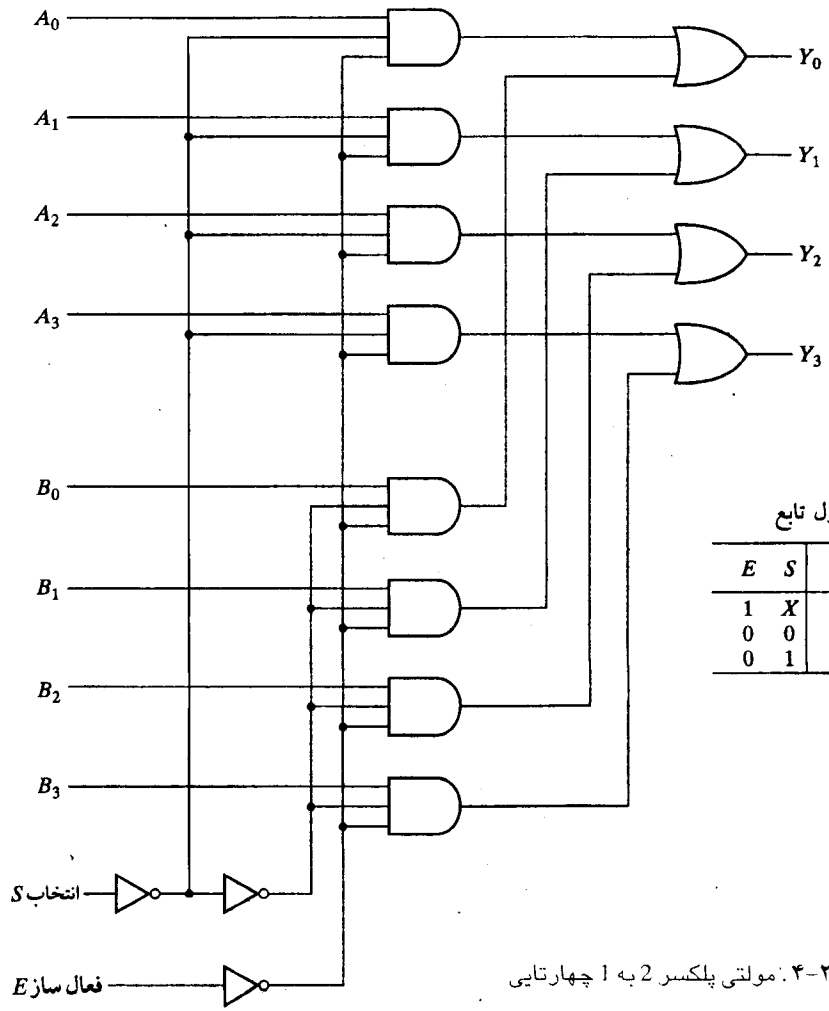


(الف) نمودار منطقی

شکل ۲۵-۴. مولتی پلکسر 4 به 1

وجود گیت‌های AND و وارونگرها در مولتی پلکسر، مدار دیکدر را به خاطر می‌آورد، و به علاوه آنها خطوط ورودی انتخاب را دیکد می‌کنند. به طور کلی یک مولتی پلکسر 2^n به 1 از یک دیکدر n به 2^n، ساخته شده که در آن 2^n خط به 2^n گیت AND، یعنی هر خط به یک گیت وصل شده است. خروجی گیت‌های AND به تنها گیت OR اعمال می‌گردند. سایز مولتی پلکسر با 2^n خط ورودی داده و تنها خط خروجی‌اش مشخص می‌شود. همچون دیکدر، مولتی پلکسرها هم ممکن است خط فعال‌سازی داشته باشند تا عملکرد کل قطعه را کنترل کنند. وقتی که ورودی فعال‌ساز در وضعیت غیرفعال قرار دارد، خروجی‌ها غیرفعالند، و وقتی در حالت فعال خود قرار گیرد، مدار به عنوان یک مولتی پلکسر معمولی عمل می‌کند.

مدارهای مولتی پلکسر را می‌توان برای تهیه مولتی پلکسر چند بیتی با هم ترکیب کرد. به منظور تشریحی بر این مطلب، یک مولتی پلکسر 2 به 1 چهارتایی در شکل ۲۶-۴ نشان داده شده است. مدار



جدول تابع

E	S	Y
1	X	تمام 0
0	0	انتخاب A
0	1	انتخاب B

شکل ۲۶-۴: مولتی پلکسر 2 به 1 چهارتایی

دارای چهار مولتی پلکسر است که هر یک قادر است یکی از دو خط را انتخاب نماید. خروجی Y_0 می‌تواند به یکی از ورودی‌های A_0 یا B_0 وصل شود و به طور مشابه Y_1 هم می‌تواند مقدار A_1 یا B_1 را داشته باشند، و به همین ترتیب. خط انتخاب ورودی یکی از خطوط ورودی را در هر یک از چهار مولتی پلکسر انتخاب می‌کند. خط فعال‌ساز E باید به هنگام کار معمولی فعال شود. گرچه مدار حاوی چهار مولتی پلکسر 2 به 1 است، ولی ما بیشتر علاقمندیم به آن به عنوان مداری که یکی از دو مجموعه چهار بیتی خطوط داده را انتخاب می‌کند، بنگریم. همانطور که در جدول تابع دیده می‌شود، مدار وقتی که $E = 0$ است فعال می‌شود. آنگاه اگر $S = 0$ باشد چهار ورودی A مسیری به چهار خروجی دارند. از طرف دیگر اگر $S = 1$، چهار ورودی B به خروجی‌ها اعمال می‌شوند، وقتی $E = 1$ باشد، مستقل از وضعیت S، همه خروجی‌ها 0 خواهند بود.

پیاده‌سازی تابع بول

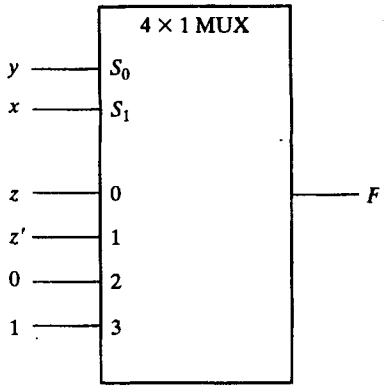
در بخش ۸-۴ دیده شد که با افزودن یک گیت OR به خروجی‌های یک دیکدر می‌توان از آن برای پیاده‌سازی توابع بول استفاده کرد. با بررسی نمودار منطقی یک مولتی پلکسر ملاحظه می‌شود که این مدار در واقع همان دیکدر است که یک گیت OR به آن اضافه شده است. مینترم‌های یک تابع با خطوط انتخاب مولتی پلکسر تولید می‌شوند. هر مینترم به وسیله ورودی‌های داده انتخاب می‌شود. این مطلب روشی را برای پیاده‌سازی هر تابع n متغیره به وسیله یک مولتی پلکسر که دارای 2^n ورودی داده و n خط ورودی انتخاب است، فراهم می‌سازد.

اکنون روش کاراتری را برای پیاده‌سازی یک تابع بول n متغیره با مولتی پلکسری که $n-1$ ورودی انتخاب دارد، معرفی می‌نماییم. ابتدا $n-1$ متغیر به ورودی‌های انتخاب مولتی پلکسر وصل می‌شود. تنها متغیر باقیمانده تابع برای ورودی‌های داده مورد استفاده قرار می‌گیرد. اگر متغیر باقیمانده را z بنامیم هر ورودی داده مولتی پلکسر برابر z، z'، 1 و یا 0 خواهد بود. برای نمایش این رویه، تابع سه متغیره زیر را ملاحظه کنید:

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

تابع را می‌توان مطابق شکل ۲۷-۴ با یک مولتی پلکسر 4 به 1 پیاده‌سازی کرد. دو متغیر x و y به خطوط انتخاب وصل می‌شود، به این ترتیب که x به ورودی s_1 و y به ورودی s_0 متصل می‌گردد. مقادیر خطوط ورودی داده از جدول درستی تابع معین می‌شود. وقتی $xy = 00$ باشد خروجی F برابر z است زیرا وقتی $z = 0$ است F هم برابر 0 می‌باشد و وقتی $z = 1$ شود F نیز برابر 1 می‌گردد. این وضع لازم می‌دارد تا متغیر z به ورودی داده 0 متصل شود. عملکرد مولتی پلکسر به نحوی است که وقتی $xy = 00$ گردد، خط داده شماره 0 به خروجی وصل شده F را برابر z می‌نماید. به طریقی مشابه می‌توان نشان داد که خطوط ورودی 1، 2 و 3 وقتی که $xy = 01, 10$ و 11 است به ترتیب به F وصل می‌شوند. این مثال خاص هر چهار حالت ممکن را برای ورودی‌های داده تهیه می‌کند.

روالی کلی برای پیاده‌سازی هر تابع بول n متغیره با یک مولتی پلکسر که $n-1$ خط ورودی انتخاب



(ب) پیاده سازی مولتی پلکسر

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(الف) جدول درستی

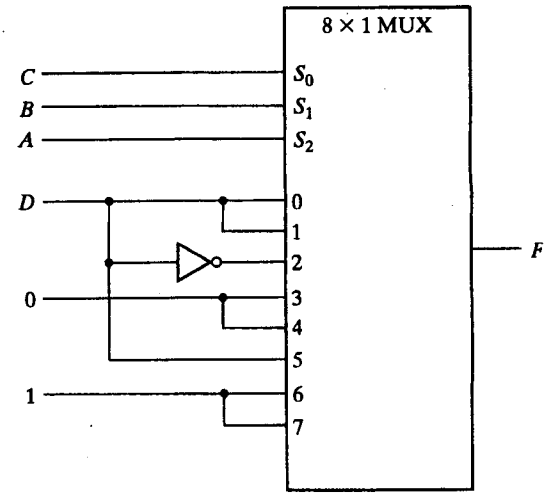
شکل ۲۷-۴ . پیاده سازی یک تابع بول با یک مولتی پلکسر

و 2^{n-1} ورودی داده دارد از مثال قبل نتیجه می‌گردد. ابتدا تابع بول را در جدول درستی لیست می‌کنیم. اولین n−1 متغیر در جدول به ورودی‌های انتخاب مولتی پلکسر وصل می‌شوند. برای هر ترکیبی از متغیرهای انتخاب، خروجی را به عنوان تابعی از آخرین متغیر ارزیابی می‌کنیم. این تابع می‌تواند 0، 1، متغیر و یا متمم متغیر باشد. آنگاه این مقادیر به ورودی‌های داده به نحوی صحیح اعمال می‌شوند. به عنوان دومین مثال، تابع بول زیر را ملاحظه نمایید:

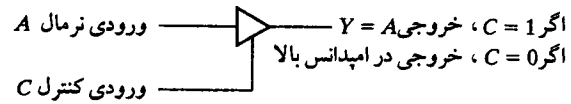
$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15).$$

این تابع با مولتی پلکسری که سه ورودی انتخاب دارد، مطابق شکل ۲۸-۴ پیاده‌سازی می‌شود. توجه کنید که اولین متغیر A باید به ورودی انتخاب S_2 و دو خط B و C به ترتیب به S_1 و S_0 وصل شوند.

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



شکل ۲۸-۴ . پیاده‌سازی یک تابع 4 ورودی با یک مولتی پلکسر



شکل ۲۹-۴. سمبل گرافیکی برای یک بافر سه حالته

مقادیر ورودی‌های داده از جدول درستی لیست شده در شکل معین می‌گردند. شماره خط داده مربوطه از ترکیب دودویی ABC حاصل می‌شود. مثلاً وقتی ABC = 101 باشد، جدول نشان می‌دهد که F = D است، بنابراین متغیر D به ورودی داده 5 وصل می‌شود. ثابت‌های دودویی 0 و 1 مربوط به دو مقدار سیگنال ثابت است. وقتی از مدارهای مجتمع استفاده کنیم، منطق 0 مربوط به سیگنال زمین و منطق 1 معادل با سیگنال تغذیه است که معمولاً 5 ولت می‌باشد.

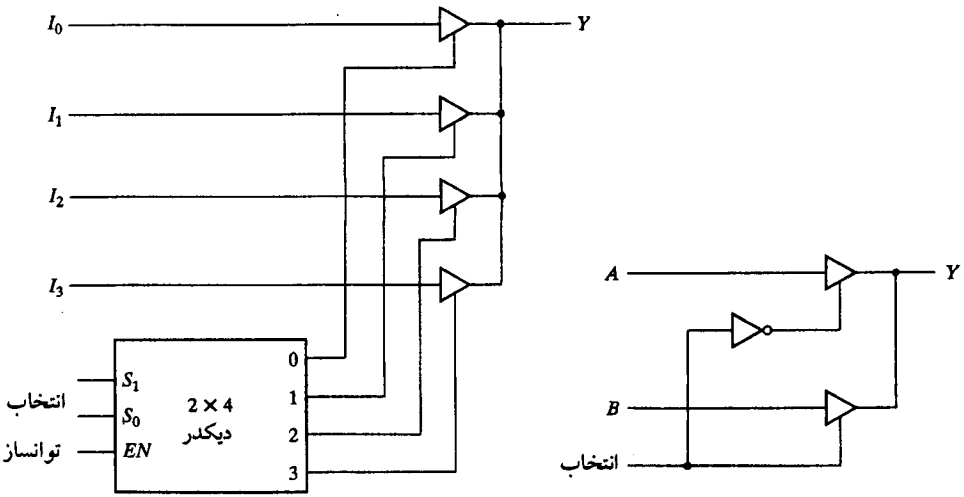
گیت‌های سه حالته

یک مولتی پلکسر را می‌توان با گیت‌های سه حالته ساخت. یک گیت سه حالته مداری دیجیتال است که سه حالت را از خود به نمایش می‌گذارد. دو حالت، همچون گیت‌های معمولی همان منطق 1 و 0 است. حالت سوم، حالت امپدانس بالاست. حالت امپدانس بالا مثل مدار باز عمل می‌کند و به این معنی است که خروجی از درون قطع بوده و مدار دارای مفهوم منطقی باارزشی نیست. گیت‌های سه حالته ممکن است به عنوان گیت‌های AND و NAND نیز عمل کنند. با این وجود اغلب به عنوان بافر مورد استفاده قرار می‌گیرند.

نمودار گرافیکی یک گیت بافر سه حالته در شکل ۲۹-۴ دیده می‌شود. این قطعه با ورودی کنترلی که وارد ضلع پایینی آن می‌شود از نوع معمولی‌اش تفکیک می‌شود. یک بافر معمولی دارای یک ورودی، یک خروجی و یک خط کنترل می‌باشد که وضع خروجی را مشخص می‌نماید. وقتی که ورودی کنترل برابر 1 است، خروجی فعال شده و گیت مانند یک بافر معمولی عمل می‌کند و در این حالت خروجی برابر ورودی اصلی است. وقتی که ورودی کنترل 0 شود، خروجی غیرفعال شده و گیت بدون توجه به مقدار ورودی اصلی به حالت امپدانس بالا می‌رود. حالت امپدانس بالای یک گیت سه حالته ویژگی خاصی را فراهم می‌کند که در دیگر گیت‌ها وجود ندارد. به علت این ویژگی، تعداد زیادی از خروجی‌های سه حالته می‌توانند به هم وصل و بدون تأثیر بر روی بار شدن، یک خط مشترکی را تشکیل دهند.

ساخت مولتی پلکسرها با بافرهای سه حالته در شکل ۳۰-۴ دیده می‌شود. بخش (الف) شکل، یک مولتی پلکسر 2 به 1 را با دو بافر سه حالته و یک وارونگر نشان می‌دهد. دو خروجی به هم وصل شده‌اند تا یک خروجی مشترک را به وجود آورند. باید اشاره کرد که این گونه اتصالات را با گیت‌هایی که خروجی سه حالته ندارند نمی‌توان اجرا کرد. وقتی ورودی انتخاب 0 است، بافر فوقانی به وسیله ورودی کنترلش فعال می‌گردد و در این حال بافر پایینی غیرفعال است.

ساختار یک مولتی پلکسر 4 به 1 در شکل ۳۰-۴ (ب) ملاحظه می‌شود. خروجی‌های چهار بافر سه



شکل ۳۰-۴. مولتی پلکسرها با گیت‌های سه حالته

حالته به هم متصل شده‌اند تا یک خروجی مشترک را بسازند. ورودی‌های کنترل به بافر مشخص می‌کنند که کدام یک از چهار ورودی نرمال I_0 تا I_3 به خط خروجی متصل خواهند شد. در هر لحظه از زمان تنها یکی از بافرها در حالت فعال قرار خواهد داشت. بافرهای متصل باید طوری وصل شوند که تنها یکی از بافرهای سه حالته با خروجی ارتباط داشته باشد، ضمن این که همه دیگر بافرها در حالت امپدانس بالا قرار خواهند گرفت. برای اطمینان از این که تنها یک ورودی کنترل در هر لحظه فعال است، از دیکدری طبق نمودار استفاده می‌کنیم. وقتی که ورودی فعال‌ساز دیکدر 0 است، هر چهار خروجی آن 0 خواهد بود و خط گذرگاه در حالت امپدانس بالاست زیرا هر چهار بافر غیرفعالند. وقتی که ورودی فعال‌ساز فعال گردد، یکی از بافرها، بسته به مقدار دودویی در ورودی‌های انتخاب دیکدر، فعال خواهد شد. با بررسی دقیق در می‌یابیم که این مدار راهی دیگر در ساخت یک مولتی پلکسر 4 به 1 است.

۴-۱۱ HDL برای مدارهای ترکیبی

زبان سخت‌افزاری verilog (HDL) در بخش ۹-۳ معرفی شد. در این بخش، روش دیگری را برای توصیف مدارهای ترکیبی HDL نشان خواهیم داد. مدارهای ترتیبی در فصل بعد توصیف خواهند شد. همانطور که قبلاً ذکر شد، مدول، یک بلوک ساختاری پایه در Verilog HDL است. مدول می‌تواند در هر یک از تکنیک‌های مدل‌سازی زیر توصیف گردد.

- مدل‌سازی سطح گیت با ذکر گیت‌های اصلی (Primitive) و مدول‌های تعریف شده بوسیله کاربر.
- مدل‌سازی روند داده با بکارگیری عبارات تخصیص مداوم (پیوسته) که با کلمه کلیدی assign انجام می‌شود.

- مدل‌سازی رفتاری با استفاده از عبارات تخصیص اجرایی (رویه‌ای) که با کلمه کلیدی always صورت می‌گیرد.

مدل‌سازی سطح گیت، مدار را با تعیین گیت‌ها و این که چگونه به هم وصل شده‌اند توصیف می‌نماید. مدل‌سازی روند داده اغلب برای توصیف مدارهای ترکیبی به کار می‌رود. مدل‌سازی رفتاری برای سیستم‌های دیجیتال در سطح بالاتر مورد استفاده است. مدل دیگری به جز روش‌های فوق وجود دارد که به آن مدل‌سازی سطح سوئیچ گویند. این نوع مدل‌سازی قابلیت طراحی را در سطح ترانزیستور MOS فراهم می‌سازد که در بخش ۱۰-۱۰ بررسی شده است.

مدل‌سازی سطح گیت

مدل‌سازی در سطح گیت در بخش ۹-۳ همراه با یک مثال ساده معرفی شد. در این نوع نمایش، یک مدار با گیت‌های منطقی و اتصالات بین آنها نشان داده می‌شود. این مدل توصیف متنی برای نمودار مداری را فراهم می‌کند. Verilog قادر است تا 12 گیت را به عنوان گیت‌های اصلی پیش تعریف شده تشخیص دهد. چهار گیت از آنها از نوع سه حالته است. هشت نوع دیگر آنهایی هستند که در بخش ۷-۲ لیست شدند. این گیت‌ها با کلمات کلید حروف کوچک زیر معرفی می‌شوند که عبارتند از: and، nand، or، nor، xor، xnor، not، buf. وقتی که گیت‌ها شبیه‌سازی شوند، سیستم به هر گیت یک مجموعه چهار مقداری را تخصیص می‌دهد. علاوه بر مقدار منطقی 0 و 1، دو مقدار نامشخص و امپدانس بالا هم لحاظ شده‌اند. مقدار نامشخص با x و امپدانس بالا با z مشخص شده است. مقدار نامشخص در حین شبیه‌سازی برای حالتی است که یک ورودی یا خروجی نامعلوم باشد و به عنوان مثال به آن مقدار 1 یا 0 تخصیص نیافته باشد. حالت امپدانس بالا در خروجی گیت‌های سه حالته هنگامی رخ می‌دهد که یک سیستم ناخودآگاه باز رها گردد. جدول درستی برای and، or، xor و not در جدول ۹-۴ دیده می‌شود. جدول درستی چهار گیت دیگر مشابه است با این تفاوت که خروجی‌ها متمم شده‌اند. توجه کنید که برای گیت and، خروجی فقط هنگامی 1 است که هر دو ورودی 1 باشند و هنگامی 0 است که هر یک از دو ورودی 0 باشد. در غیر این صورت اگر یک ورودی x یا z باشد خروجی x است. وقتی هر دو ورودی گیت or برابر 0 باشد

جدول ۹-۴. جدول درستی برای گیت‌های اصلی پیش‌تعریف شده

and	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

or	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

xor	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

not	خروجی ورودی
	0 1
	1 0
	x x
	z x

خروجی آن 0است و اگر هر یک از ورودی‌ها 1 باشد خروجی هم 1است، در غیر اینصورت x است.
وقتی که یک گیت اصلی در یک مدول لحاظ شود گوییم در مدول ذکر شده است. به طور کلی ذکر قطعات عباراتی هستند که به قطعات سطح پایین‌تری در طراحی ارجاع می‌دهند، و کپی‌هایی اساسی (یا نمونه) از آن قطعات در مدول سطح بالاتر ایجاد می‌نمایند. بنابراین مدولی که یک گیت را در توصیف خود به کار می‌برد گیت را **ذکر** کرده است.

اکنون دو مثال از مدل‌سازی سطح گیت را نمایش می‌دهیم. هر دو مثال از عرض چند بیتی که بردار نامیده می‌شود، استفاده می‌کنند. یک بردار در داخل یک جفت کروشه مشخص می‌شود که از دو عدد و دو نقطه در بین آنها تشکیل شده است. کد زیر دو بردار را نشان می‌دهد:

```
output [0:3]D;
wire [7:0]SUM;
```

عبارت اول یک بردار خروجی D را با چهار بیت 0 تا 3 نشان می‌دهد. دومین عبارت یک بردار SUM سیمی (wire) با هشت بیت که از 0 تا 7 شماره‌گذاری شده‌اند را نشان می‌دهد. اولین عدد لیست شده باارزش‌ترین بیت بردار است. بیت‌های خاص در داخل کروشه ذکر می‌گردند، بنابراین D[2]، بیت 2 از D را نشان می‌دهد. همچنین ممکن است بخش‌هایی از یک بردار را ذکر کند. مثلاً SUM[2 : 0] سه بیت کم‌ارزش‌تر بردار SUM را بیان می‌کند.

مثال ۱-۴ HDL توصیف سطح گیت یک دیکدر 2 به 4 را نشان می‌دهد. این مدار دو ورودی داده A و B و یک ورودی فعال‌ساز E دارد. چهار خروجی با بردار D تعریف شده‌اند. wire برای اتصالات درونی به کار می‌رود. گیت‌های not متمم ورودی‌ها و چهار گیت nand خروجی D را تهیه می‌نماید. به یاد داشته باشید که همواره در لیست گیت، خروجی ابتدا ذکر می‌شود و سپس ورودی‌ها نوشته می‌شوند. این مثال دیکدر شکل ۱۹-۴ را توصیف می‌کند و رویه‌های بیان شده در بخش ۹-۳ را دنبال می‌نماید.

مثال ۱-۴، HDL

```
//Gate-level description of a 2-to-4-line decoder
//Figure 4-19
module decoder_gl (A,B,E,D);
   input A,B,E;
   output [0:3]D;
   wire Anot,Bnot,Enot;
   not
      n1 (Anot,A),
      n2 (Bnot,B),
      n3 (Enot,E);
   nand
      n4 (D[0],Anot,Bnot,Enot),
      n5 (D[1],Anot,B,Enot),
      n6 (D[2],A,Bnot,Enot),
      n7 (D[3],A,B,Enot);
endmodule
```

```
//Gate-level hierarchical description of 4-bit adder
// Description of half adder (see Fig 4-5b)
module halfadder (S,C,x,y);
   input x,y;
   output S,C;
//Instantiate primitive gates
   xor (S,x,y);
   and (C,x,y);
endmodule

//Description of full adder (see Fig 4-8)
module fulladder (S,C,x,y,z);
   input x,y,z;
   output S,C;
   wire S1,D1,D2; //Outputs of first XOR and two AND gates
//Instantiate the halfadder
    halfadder HA1 (S1,D1,x,y),
              HA2 (S,D2,S1,z);
    or g1(C,D2,D1);
endmodule

//Description of 4-bit adder (see Fig 4-9)
module _4bit_adder (S,C4,A,B,C0);
   input [3:0] A,B;
   input C0;
   output [3:0] S;
   output C4;
   wire C1,C2,C3;  //Intermediate carries
//Instantiate the fulladder
   fulladder  FA0 (S[0],C1,A[0],B[0],C0),
              FA1 (S[1],C2,A[1],B[1],C1),
              FA2 (S[2],C3,A[2],B[2],C2),
              FA3 (S[3],C4,A[3],B[3],C3);
endmodule
```

توجه کنید که کلمات کلیدی not و nand فقط یک بار نوشته می‌شوند و لزومی ندارد که برای هر گیت تکرار شوند، ولی ویرگول‌ها باید در انتهای هر سری گیت‌ها گذاشته شوند به جز در آخرین عبارت که با نقطه ویرگول پایان می‌یابد.

دو یا چند مدول را می‌توان با هم ترکیب نموده و توصیف سلسله مراتبی یک طرح را ساخت. دو روش طراحی اساسی وجود دارد: یکی بالا به پایین و دیگری پایین به بالا. در طراحی بالا به پایین، بلوک سطح بالا تعریف شده و سپس بلوک‌های جزیی لازم برای ساخت بلوک سطح بالا مشخص می‌شوند. در طراحی پایین به بالا، بلوک‌های ساختاری ابتدا مشخص شده و سپس برای ساخت بلوک بالایی ترکیب

می‌شوند. مثلاً جمع‌کننده شکل ۹-۴ را در نظر بگیرید. می‌توان آن را بلوکی تصور کرد که با چهار بلوک جمع‌کننده کامل ساخته شده است، در حالی که هر جمع‌کننده کامل از دو بلوک نیم‌جمع‌کننده تشکیل می‌شود. در طراحی بالا به پایین، ابتدا جمع کننده 4 بیت تعریف می‌شود و سپس بلوک‌های دو نیم‌جمع کننده توصیف می‌شوند. در طراحی پایین به بالا ، نیم جمع کننده‌ها اول تعریف می‌شوند، سپس جمع کننده 4 بیت از آن ساخته می‌شود.

توصیف سلسه مراتبی (پایین - بالا) جمع‌کننده 4 بیت در مثال ۲-۴ HDL نشان داده شده است. نیم‌جمع‌کننده با ذکر گیت‌های اصلی تعریف می‌شود. مدول بعدی با ذکر دو نیم جمع کننده، جمع کننده کامل را ذکر کرده است. مدول سوم یک جمع‌کننده 4 بیت را با ذکر چهار جمع‌کننده کامل توصیف می‌نماید. (دقت کنید که شناسه‌ها نمی‌توانند با یک عدد آغاز شوند ولی می‌توانند با یک زیر خط تیره در پایین شروع شوند، بنابراین نام مدول 4bitadder_ خواهد بود.) ذکر هم با نام مدولی که یک مجموعه نام پورت جدید یا قبلی را بکار می‌برد انجام می‌شود. مثلاً، HA1 در داخل مدول جمع‌کننده با x، y و D_1 ، S_1 ذکر، می‌شود. این ذکر یک نیم‌جمع‌کننده با خروجی‌های D_1، S_1 و ورودی‌های y و x را تولید می‌نماید.

توجه کنید که در Verilog ، تعریف یک مدول نمی‌تواند در توصیف دیگری قرار گیرد. به بیان دیگر، یک مدول نمی‌تواند در بین یک کلمه کلیدی module و endmodule واقع شود. تنها راه تعریف یک مدول در دیگری، ذکر آن است. بنابراین مدول‌ها در مدول‌های دیگر برای توصیف سلسله مراتبی طرح ذکر می‌شوند. همچنین توجه کنید که نام‌ها باید هنگام تعریف مدول‌ها ذکر شوند. (مثل FA0 برای اولین جمع‌کننده کامل در سومین جدول)، ولی ذکر نام در گیت‌های اصلی اختیاری است.

گیت‌های سه حالته

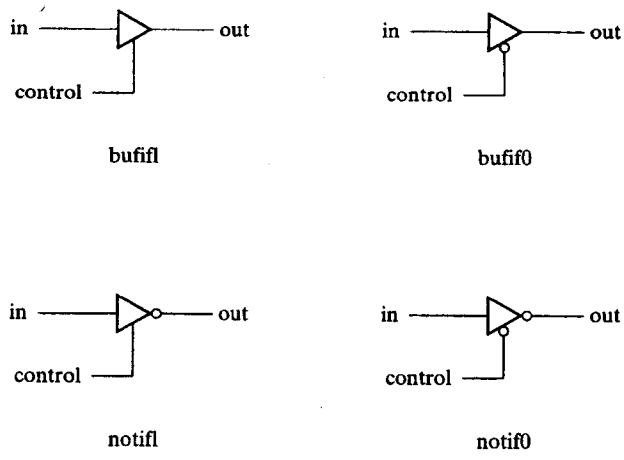
همانطور که در بخش ۱۰-۴ ذکر شد، هر گیت سه حالته دارای یک ورودی کنترل است که می‌تواند آن را به حالت امپدانس بالا ببرد. این حالت در HDL با سمبل z مشخص می‌شود. چهار نوع گیت سه حالته طبق شکل ۳۱-۴ وجود دارد. گیت bufif1 مانند یک بافر معمولی عمل می‌کند به شرطی که control = 1 باشد. وقتی control = 0 است، خروجی به حالت امپدانس بالای z می‌رود. گیت bufif0 رفتاری مشابه دارد با این تفاوت که امپدانس بالا در control = 1 رخ می‌دهد. دو گیت not به همین ترتیب کار می‌کنند با این تفاوت که وقتی در امپدانس بالا نیستند، خروجی آنها متمم ورودی است. گیت‌ها با عبارت زیر ذکر می‌شوند:

gate name (output, input, control);

gate name می‌تواند یکی از چهار گیت سه حالته انتخاب گردد. خروجی می‌تواند 0، 1 یا z باشد دو مثال از ذکر گیت در زیر آمده است:

```
bufif1  (OUT,A,control);
notif0  (Y,B,enable);
```

در مثال اولی، ورودی A هنگامی که control = 1 است به خروجی منتقل می‌گردد. وقتی control=0



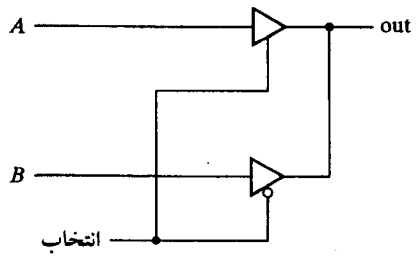
شکل ۳۱-۴ . گیت‌های سه حالته

باشد خروجی out به z می‌رود. در مثال دوم وقتی enable = 1 است Y = z خواهد شد و خروجی Y = B′ در enable = 0 رخ می‌دهد.

خروجی گیت‌های سه حالته می‌توانند برای تشکیل یک خروجی مشترک بهم وصل شوند. برای شناخت چنین اتصالی، HDL از کلمه کلید tri (برای tristate) استفاده می‌کند تا بیان نماید که خروجی دارای قابلیت راه‌اندازی چندگانه است. مثلاً مولتی پلکسر 2 به 1 را در شکل ۳۲-۴ با گیت‌های سه حالته در نظر بگیرید.

توصیف HDL باید برای خروجی از داده نوع tri استفاده کند.

```
module muxtri (A,B,select,OUT);
   input A,B,select;
   output OUT;
   tri OUT;
   bufif1 (OUT,A,select);
   bufif0 (OUT,B,select);
endmodule
```



شکل ۳۲-۴ . مولتی پلکسر 2 به 1 با بافرهای سه حالته

دو بافر سه حالته دارای خروجی مشترک‌اند. برای این که نشان دهیم آنها اتصال مشترک دارند، باید خروجی OUT را با کلمه کلیدی tri همراه کنیم.

کلمات کلیدی wire و tri مثال‌هایی از داده نوع net می‌باشند. Net ها اتصال بین دوالمان یا عنصر را نشان می‌دهند. خروجی آنها با خروجی وسیله‌ای که نشان می‌دهند مرتباً راه‌اندازی می‌شود. کلمه net (شبکه) یک کلمه کلیدی نیست و کلاسی از انواع داده مانند wire ، wor ، wand ، tri ، supply1 و supply0 را نشان می‌دهد. اعلان wire مکرّر به کار برده می‌شود. مدل‌های wor پیاده‌سازی سخت‌افزاری آرایش wired–OR را نشان می‌دهند. مدل‌های wand هم برای آرایش wired–AND در نظر گرفته شده‌اند (شکل ۲۸-۳). شبکه‌های supply1 و supply0 منبع تغذیه و زمین هستند. از آنها در توصیف سطح- سوئیچ استفاده می‌شود. (بخش ۱۰-۱۰ ملاحظه گردد).

مدل‌سازی روند داده

مدل‌سازی روند داده از تعدادی عملگر روی عملوندها استفاده می‌کند تا خروجی موردنظر را تولید کند. Verilog HDL حدود 30 عملگر در اختیار می‌گذارد. جدول ۱۰-۴ بخشی از این عملگرها، سمبل‌های آنها و عملی که اجرا می‌کنند را نشان می‌دهند. لیست کامل عملگرها را در جدول ۱-۸ در بخش ۲-۸ ملاحظه کنید. لازم است تا بین اعمال حسابی و منطقی تفکیک به عمل آید، بنابراین برای هر یک، از سمبل جداگانه‌ای استفاده شده است. سمبل (+) برای جمع حسابی به کار رفته و AND منطقی از سمبل & استفاده می‌کند. برای OR ، NOT و XOR سمبل‌های خاصی وجود دارد. سمبل برابری از دو علامت مساوی و بدون فضا در بین آنها استفاده می‌کند تا با علامت برابری به کار رفته با عبارت تخصیص اشتباه نشود. عملگر ادغام مکانیزمی است که برای ضمیمه کردن چند عملوند استفاده می‌شود. مثلاً دو عملوند دو بیتی را می‌توان برای ایجاد یک عملوند چهار بیتی در هم ادغام کرد. عملگر شرطی بعد همراه با مثال ۶-۴ HDL توضیح داده شده است.

جدول ۱۰-۴. عملگرهای Verilog HDL

Symbol	Operation
+	binary addition
−	binary subtraction
&	bit-wise AND
|	bit-wise OR
∧	bit-wise XOR
~	bit-wise NOT
==	equality
>	greater than
<	less than
{ }	concatenation
? :	conditional

مثال ۳-۴ HDL

```
//Dataflow description of a 2-to-4-line decoder
//See Fig. 4-19
module decoder_df (A,B,E,D);
   input A,B,E;
   output [0:3] D;
   assign D[0] = ~(~A & ~B & ~E),
          D[1] = ~(~A & B & ~E),
          D[2] = ~(A & ~B & ~E),
          D[3] = ~(A & B & ~E);
endmodule
```

مدل‌سازی روند داده از تخصیص‌های مداوم و کلمه کلیدی assign استفاده می‌کند. یک تخصیص مداوم عبارتی است که یک مقدار را به یک net تخصیص می‌دهد. *net* نوع داده در HDL برای نمایش اتصالات بین عناصر مدار به کار می‌رود. یک net خروجی یک گیت را که با عبارت output یا wire بیان شده، تعریف می‌کند. مقدار تخصیص داده شده به net با یک عبارت که عملگر و عملوند را استفاده می‌کند بیان می‌گردد. به عنوان مثال با فرض اعلان متغیرها، یک مولتی پلکسر 2 به 1 با ورودی‌های داده A و B، ورودی انتخاب S و خروجی Y با عبارت مداوم زیر تعریف می‌گردد:

```
assign Y = (A & S) | (B & ~S);
```

عبارت با کلمه کلیدی assign و به دنبال آن خروجی موردنظر Y و یک علامت مساوی شروع می‌شود. به دنبال علامت مساوی، یک عبارت بول آورده شده است. در سخت‌افزار، این عبارت معادل با اتصال خروجی گیت OR (|) به سیم Y است.

مثال‌های بعدی مدل‌های روند داده دو مثال سطح گیت قبلی را نشان می‌دهند. توصیف روند داده یک دیکدر 2 به 4 در مثال ۳-۴ HDL نشان داده شده است. مدار با توجه به عبارات بولی با چهار عبارت تخصیص مداوم تعریف شده است که هر یک متعلق به یک خروجی است. توصیف روند داده یک جمع کننده 4 بیت در مثال ۴-۴ HDL آورده شده است. منطق جمع با یک عبارت و استفاده از عملگرهای جمع و ادغام بیان شده است. سمبل جمع (+) بیانگر جمع دودویی 4 بیت A با چهار بیت B و یک رقم نقلی Cin است. خروجی موردنظر ادغام نقلی خروجی Cout و چهار بیت SUM است.

مثال ۴-۴ HDL

```
//Dataflow description of 4-bit adder
module binary_adder (A,B,Cin,SUM,Cout);
   input [3:0] A,B;
   input Cin;
   output [3:0] SUM;
   output Cout;
   assign {Cout,SUM} = A + B + Cin;
endmodule
```

مثال ۵-۵، HDL

```
//Dataflow description of a 4-bit comparator.
module magcomp (A,B,ALSB,AGTB,AEQB);
   input [3:0] A,B;
   output ALTB,AGTB,AEQB;
   assign ALTB=(A < B),
          AGTB = (A > B),
          AEQB = (A == B);
endmodule
```

ادغام علموندها در پرانتز بیان شده و یک ویرگول آنها را از هم جدامی‌سازد. بنابراین پنج بیت {Cout, SUM} ، نتیجه عمل جمع را به نمایش می‌گذارد.

مدل‌سازی روند داده امکان توصیف مدارهای ترکیبی را با تابع به جای ساختار گیتی‌اش فراهم می‌سازد. برای ملاحظه چگونگی انجام طراحی دیجیتال با روند داده، مقایسه‌گر مقدار چهار بیتی توصیف شده در مثال ۵-۴ HDL را در نظر بگیرید. مدل دو گروه ورودی چهار بیت A و B و سه خروجی را مشخص می‌کند. اگر A کوچکتر از B باشد خروجی (ALTB) در منطق 1 و اگر A بزرگتر از B باشد خروجی (AGTB) در منطق 1 قرار می‌گیرد. ضمن این که اگر A برابر با B است خروجی (AEQB) وجود داشته باشد. توجه کنید که تساوی با دو علامت مساوی تعریف می‌شود. کامپایلر طراحی Verilog HDL می‌تواند این توصیف مدول را به عنوان ورودی بپذیرد و یک خروجی netlist از یک مدار معادل با شکل ۱۷-۴ را فراهم سازد.

مثال بعدی از عملگر شرطی (: ?) استفاده می‌نماید. این عملگر سه عملوند را اختیار می‌کند.

Condition ? true– expression: false–expression;

شرط همواره ارزیابی می‌شود. اگر نتیجه منطق 1 بود عبارت true–expression ارزیابی می‌گردد. اگر نتیجه منطق 0 بود، false-expression ارزیابی خواهد شد. این معادل با یک شرط if– else است.

مثال ۶-۴ HDL مولتی پلکسر 2 به 1 خط را با عملگر شرطی توصیف می‌نماید. تخصیص مداوم

```
assign OUT = select ? A : B ;
```

شرط زیر را بیان می‌کند.

OUT = A if select = 1, else OUT = B if select = 0

مثال ۶-۴، HDL

```
//Dataflow description of 2-to-1-line multiplexer
module mux2x1_df (A,B,select,OUT);
   input A,B,select;
   output OUT;
   assign OUT = select ? A : B;
endmodule
```

مثال ۴-۷، HDL

```
//Behavioral description of 2-to-1-line multiplexer
module mux2x1_bh(A,B,select,OUT);
   input A,B,select;
   output OUT;
   reg OUT;
   always @ (select or A or B)
         if (select == 1) OUT = A;
         else OUT = B;
endmodule
```

مدل‌سازی رفتاری

مدل‌سازی رفتاری مدارهای دیجیتال را در سطح الگوریتمی و عملیاتی نمایش می‌دهد. این مدل اغلب در توصیف مدارهای ترتیبی به کار برده می‌شود، ولی قابل استفاده در توصیف مدارهای ترکیبی هم می‌باشد. در اینجا دو مثال از مدارهای ترکیبی ساده برای معرفی موضوع ارائه می‌گردد. این بحث پس از مطالعه مدارهای ترتیبی در بخش ۵-۵ مورد بحث بیشتری قرار گرفته است.

توصیف‌های رفتاری از کلمه کلیدی always و به دنبال آن لیستی از عبارات تخصیص اجرایی (رویه‌ای) استفاده می‌کنند. خروجی موردنظر این عبارات باید نوع داده reg باشد. برخلاف داده نوع wire که خروجی موردنظر یک تخصیص ممکن است مرتباً به روز شود، داده نوع reg مقدارش را تا تخصیص مقدار جدید حفظ می‌کند.

مثال ۴-۷ HDL توصیف رفتاری مولتی پلکسر 2 به 1 را نشان می‌دهد (آن را با مثال ۴-۶ HDL مقایسه کنید). چون OUT یک خروجی موردنظر یا مقصد است، باید علاوه بر اعلان output ، به صورت داده reg هم اعلام گردد. عبارات تخصیص اجرایی در داخل بلوک always هر زمان که تغییری در هر متغیر بعد از علامت @ رخ دهد دوباره اجرا می‌گردد. توجه کنید که در انتهای عبارت always علامت (;) وجود ندارد. در این حال، آنها عبارتند از A ، B و select . توجه کنید که کلمه کلیدی or در بین متغیرها به جای عملگر منطقی OR ، "|" ، استفاده شده است. عبارت شرطی if– else تصمیمی را که مبتنی بر مقدار ورودی selectاست فراهم می‌کند. عبارت if را می‌توان بدون ذکر سمبل کمیت نیز نوشت:

```
if (select) OUT = A ;
```

و به این معنی است که select برای منطق 1 چک می‌شود.

مثال ۴-۸ HDL یک مولتی پلکسر 4 به 1 را توصیف می‌نماید. ورودی select به صورت یک بردار 2 بیت توصیف شده و خروجی y هم با داده reg اعلان شده است. عبارت always دارای یک بلوک ترتیبی در بین کلمات کلیدی case و endcase است. این بلوک هر وقت که هر ورودی بعد از علامت @ تغییر کند، اجرا خواهد شد. عبارت case یک انشعاب شرطی چند مسیری است. عبارت case (select) با مقادیر لیست عباراتی که به دنبالش می‌آیند ارزیابی و مقایسه می‌شود. اولین مقداری که با شرط

مثال ۸-۴، HDL

```
//Behavioral description of 4-to-1- line multiplexer
//Describes the function table of  Fig. 4-25(b).
module mux4x1_bh (i0,i1,i2,i3,select,y);
   input i0,i1,i2,i3;
   input [1:0] select;
   output y;
   reg y;
   always @ (i0 or i1 or i2 or i3 or select)
            case (select)
               2'b00: y = i0;
               2'b01: y = i1;
               2'b10: y = i2;
               2'b11: y = i3;
            endcase
endmodule
```

صحیح تطبیق کند اجرا می‌گردد. چون select یک عدد دوبیتی است، می‌تواند 00، 01، 10 و یا 11 باشد. اعداد دودویی با حرف b و قبل از آن یک علامت پریم مشخص می‌گردند. سایز عدد ابتدا نوشته می‌گردد و سپس مقدار آن ذکر می‌شود. بنابراین 2' b 0 1 به معنی عدد دودویی دو رقمی است که مقدارش 01 است. اعداد را می‌توان به دهدهی، هشت هشتی یا شانزده شانزدهی و به ترتیب با 'd، 'o و 'h مشخص کرد. اگر مبنای عدد مشخص نباشد، پیش‌فرض دهدهی خواهد بود. اگر سایز عدد نامشخص باشد، سیستم سایز را 32 بیت فرض خواهد کرد.

در اینجا مثال‌های ساده‌ای را از توصیف‌های رفتاری مدارهای ترکیبی نشان دادیم. مدل‌سازی رفتاری و عبارات تخصیص اجرایی دانش مدارهای ترتیبی را لازم دارد و در بخش ۵-۵ به طور مشروح ارائه شده است.

نوشتن یک برنامه تست ساده

یک برنامه تست (T.B) برنامه‌ای HDL است که برای اعمال محرک به طرح HDL برای تست و مشاهده پاسخ شبیه‌ساز آن به کار می‌رود. T.B می‌تواند بسیار پیچیده و طولانی باشند تا حدی که ساخت آن از طرح مورد تست بیشتر طول بکشد. با این وجود، برنامه‌ای که در اینجا بررسی می‌شود نسبتاً ساده است زیرا ما فقط مایل به تست مدارهای ترکیبی هستیم. مثال‌ها برای نمایش توصیف‌های نمونه مدول‌های محرک HDL ارائه شده‌اند.

علاوه بر عبارت always، برنامه تست از عبارت initial برای تهیه محرک به مدار تحت تست استفاده می‌کند. عبارت always حلقه‌ای را به صورت تکراری اجرا می‌کند. عبارت initial فقط یک بار با شروع از t = 0 شبیه‌سازی را انجام داده و ممکن است هر عملی را با تأخیری که مضربی از واحدهای زمانی است و با سمبل # مشخص شده ادامه دهد. مثلاً بلوک initial زیر را ملاحظه نمایید.

```
initial
    begin
            A = 0; B= 0;
       #10  A = 1;
       #20  A = 0; B=1;
    end
```

بلوک، بین begin و end محصور شده است. در $t = 0$، A و B در 0 قرار گرفته‌اند. 10 واحد زمان بعد، A به 1 تغییر یافته است، پس از 20 واحد زمانی، (در $t = 30$) A به 0 و B به 1 تغییر پیدا می‌کند. ورودی‌ها به جدول درستی 3 بیتی می‌توانند با بلوک initial تولید شوند:

```
initial
    begin
            D = 3'b000;
            repeat (7)
       #10  D = D + 3'b001;
    end
```

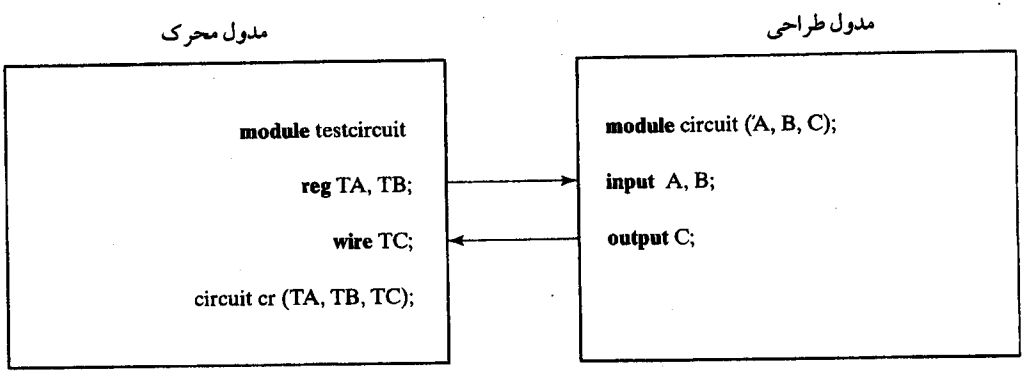
بردار 3 بیت D در $t = 0$ با 000 مقداردهی اولیه می‌شود. کلمه کلیدی repeat یک عبارت حلقه‌ای را تداعی می‌کند: یعنی عدد 1 هفت بار به D اضافه شده و این کار هر 10 واحد زمانی یک بار تکرار شده است. نتیجه یک رشته اعدادی از 000 تا 111 خواهد بود.

یک مدول محرک در یک برنامه HDL ساختار زیر را دارد.

نام **module** تست
اعلان شناسه‌های **reg** و **wire** محلی
ذکر مدول طراحی مورد تست
ایجاد عبارات محرک **initial** و **always**
نمایش پاسخ خروجی
endmodule

یک مدول تست معمولاً ورودی یا خروجی ندارد. سیگنال‌هایی که به عنوان ورودی به مدول طراحی برای شبیه‌سازی اعمال می‌شوند معمولاً در مدول محرک به عنوان نوع داده reg محلی اعلان می‌گردند. خروجی‌های مدول طراحی که برای تست نمایش داده می‌شوند در مدول محرک به عنوان داده نوع wire اعلان می‌شوند. آنگاه مدول تحت تست با به‌کارگیری شناسه‌های محلی ذکر می‌گردد. شکل ۳۳-۴ این ارتباط را نشان می‌دهد. مدول محرک ورودی‌ها را برای مدول طراحی با اعلان شناسه‌های TA و TB به عنوان نوع reg تولید می‌کند و خروجی طرح را با شناسه نوع wire، یعنی TC چک می‌نماید. سپس شناسه‌های محلی برای ذکر مدول زیر تست به کار می‌روند.

پاسخ به محرک تولید شده با بلوک‌های intial و always در خروجی محرک به صورت نمودار زمان‌بندی ظاهر می‌شود. و نیز می‌توان با استفاده از Verilog system tasks خروجی عددی نیز تولید



شکل ۳۳ - مدول‌های محرک و طراحی به‌طور جدا

کرد. این کار در سیستم با شناسایی کلمات کلیدی که با سمبل $ آغاز می‌شوند ساخته می‌شود. بعضی از این وظایف مفید در نمایش در زیر آمده است:

$ display	یک بار نمایش مقدار متغیر یا رشته‌هایی با بازگشت از انتهای خط
$ write	مثل $ display ولی بدون رفتن به خط بعد
$ monitor	هر وقت در حین اجرای شبیه‌سازی متغیری تغییر کند، آن را نمایش می‌دهد
$ time	زمان شبیه‌سازی را نشان می‌دهد
$ finish	شبیه‌سازی را پایان می‌دهد

قاعده نوشتن $ display، $write و $ monitor به شکل زیر است.

Task– name (format specification, argument list);

و یا

;(لیست آرگومان و مشخصات قالب) نام تکلیف

مشخصات قالب شامل مبنای اعدادی است که با استفاده از سمبل (%) نمایش داده می‌شوند و ممکن است دارای رشته‌ای در داخل (") باشد. مبنا می‌تواند دودویی، دهدهی، هشت هشتی و یا شانزده شانزدهی فرض شود که به ترتیبی با سمبل‌های %b، %d، %o و %h نشان داده می‌شوند. مثلاً عبارت:

```
$display (%d %b %b, C,A,B);
```

به این معنی که C به دهدهی و A و B به دودویی نمایش داده شوند. توجه کنید که در مشخصات قالب، علامت ویرگول وجود ندارد ولی برای جداسازی مشخصات قالب و لیست آرگومان و نیز بین متغیرهای لیست آرگومان، ویرگول وجود دارد. مثالی که یک رشته را داخل علامت کوتیشن یا نقل قول محصور کند مشابه زیر است:

```
$display ("time = %0d A = %b B = %b", $time,A,B);
```

و نمایش زیر را تولید خواهد کرد

```
time = 3   A = 10   B = 1
```

```
//Stimulus for mux2x1_df.
module testmux;
  reg TA,TB,TS;  //inputs for mux
  wire Y;        //output from mux
  mux2x1_df mx (TA,TB,TS,Y);  // instantiate mux
     initial
        begin
              TS = 1; TA = 0; TB = 1;
          #10 TA = 1; TB = 0;
          #10 TS = 0;
          #10 TA = 0; TB = 1;
        end
     initial
      $monitor("select = %b A = %b B = %b OUT = %b time = %0d",
               TS, TA, TB, Y, $time);
endmodule

//Dataflow description of 2-to-1-line multiplexer
//from Example 4-6
module mux2x1_df (A,B,select,OUT);
   input A,B,select;
   output OUT;
   assign OUT = select ? A : B;
endmodule

Simulation log:

select = 1 A = 0 B = 1 OUT = 0 time = 0
select = 1 A = 1 B = 0 OUT = 1 time = 10
select = 0 A = 1 B = 0 OUT = 0 time = 20
select = 0 A = 0 B = 1 OUT = 1 time = 30
```

که (time =)، (A =) و (B =) بخشی از رشته مورد نمایش‌اند. قالب %0d، %b و %b به ترتیب مبنای $ time، A و B را مشخص می‌کنند. هنگام نمایش مقادیر تابع، بهتر است قالب %0d را به جای %d به کار ببریم. این کار رقم‌های باارزش‌تر را بدون فضای خالی تولید می‌نماید (%d حدود 10 فضای خالی را تولید می‌کند زیرا زمان با عدد 32 بیت تولید می‌گردد).

مثالی از مدول محرک در مثال HDL ۴-۹ نشان داده شده است. مدار مورد تست یک مولتی پلکسر 1 × 2 است که در مثال ۶-۴ توصیف گردید مدول testmux پورت ندارد. ورودی‌های mux با کلمه کلیدی reg و خروجی‌ها با wire مشخص می‌شوند. mux با متغیرهای محلی ذکر شده است. بلوک initial رشته‌ای از اعداد دودویی را که در حین شبیه‌سازی اعمال می‌گردند، مشخص می‌کند. پاسخ خروجی با تکلیف monitor $ چک می‌شود. هر بار یک متغیر تغییر مقدار دهد، شبیه‌ساز ورودی‌ها، خروجی و زمان را نمایش می‌دهد. نتیجه شبیه‌سازی در مثال زیر تیتر simulation ذکر شده است.

```
//Gate-level description of circuit of Fig. 4-2
module analysis (A,B,C,F1,F2);
   input   A,B,C;
   output  F1,F2;
   wire    T1,T2,T3,F2not,E1,E2,E3;
   or  g1 (T1,A,B,C);
   and g2 (T2,A,B,C);
   and g3 (E1,A,B);
   and g4 (E2,A,C);
   and g5 (E3,B,C);
   or  g6 (F2,E1,E2,E3);
   not g7 (F2not,F2);
   and g8 (T3,T1,F2not);
   or  g9 (F1,T2,T3);
endmodule

//Stimulus to analyze the circuit
module test_circuit;
  reg [2:0]D;
  wire F1,F2;
  analysis fig42(D[2],D[1],D[0],F1,F2);
  initial
    begin
     D = 3'b000;
     repeat(7)
      #10 D = D + 1'b1;
    end
  initial
     $monitor ("ABC = %b F1 = %b F2 =%b ",D, F1, F2);
endmodule


Simulation log:

ABC = 000 F1 = 0 F2 =0
ABC = 001 F1 = 1 F2 =0
ABC = 010 F1 = 1 F2 =0
ABC = 011 F1 = 0 F2 =1
ABC = 100 F1 = 1 F2 =0
ABC = 101 F1 = 0 F2 =1
ABC = 110 F1 = 0 F2 =1
ABC = 111 F1 = 1 F2 =1
```

می‌بینیم که وقتی $S = 1$ باشد $OUT = A$ و وقتی $S = 0$ باشد $OUT = B$ است، که بدین ترتیب عملکرد مولتی پلکسر را تأیید می‌کند.

شبیه‌سازی منطقی، روشی سریع و دقیق در تحلیل مدارهای ترکیبی جهت اطمینان از عملکرد

صحیح آنهاست. دو نوع تصدیق وجود دارد: عملیاتی و زمانی. در تصدیق عملیاتی، ما عملکرد مدار را جدا از ملاحظات زمانی مورد بررسی قرار می‌دهیم. این کار با تهیه جدول درستی مدار ترکیبی انجام می‌شود. در تصدیق زمانی عملکرد مدار را با احتساب آثار تأخیر در گیت‌ها مطالعه می‌کنیم. این کار با مشاهده امواج در خروجی گیت‌ها وقتی به یک ورودی مفروض پاسخ می‌دهند، صورت می‌گیرد.
مثالی از یک مدار با تأخیر در گیت‌ها در بخش ۹-۳ و با مثال ۳-۳ HDL ملاحظه شد. اکنون یک مثال HDL، که جدول درستی یک مدار ترکیبی را تولید می‌کند، ملاحظه می‌شود.

تحلیل مدارهای ترکیبی در بخش ۲-۴ ملاحظه شد. یک مدار جمع‌کننده کامل چند طبقه تحلیل شد و جدول درستی آن بدست آمد. توصیف سطح گیت این مدار در مثال ۱۰-۴ HDL آورده شده است. مدار دارای سه ورودی دو خروجی و نه گیت است. به دنبال توصیف مدار، اتصالات درونی بین گیت‌ها طبق نمودار شماتیک ۲-۴ آمده است. محرک مدار در مدول دوم ذکر شده است. ورودی‌ها برای شبیه‌سازی مدار با بردار D سه بیتی reg مشخص شده‌اند. D[2] با ورودی A و D[1] با ورودی B، و D[0] با ورودی C معادل است. خروجی‌های مدار F_1 و F_2 بوده و از نوع wire اعلان می‌شوند. این رویه مراحل مشخص شده در شکل ۳۳-۴ را دنبال می‌کند. حلقه repeat هفت عدد دودویی را پس از 000 برای جدول درستی تهیه می‌کند. نتایج شبیه‌سازی جدول درستی خروجی را نمایش می‌دهد. جدول درستی لیست شده نشان می‌دهد که مدار یک جمع‌کننده کامل است.

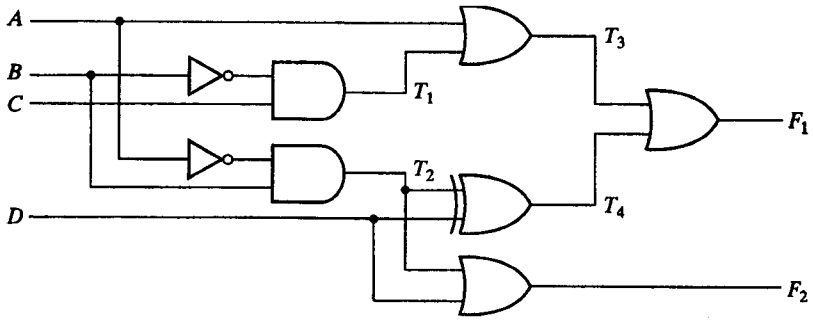
مسائل

۱-۴ مدار ترکیبی شکل (م ۱-۴) را در نظر بگیرید.

(الف) عبارات بولی را برای T_1 تا T_4 بدست آورید. خروجی‌های F_1 و F_2 را به عنوان یک تابع چهار ورودی ارزیابی کنید.

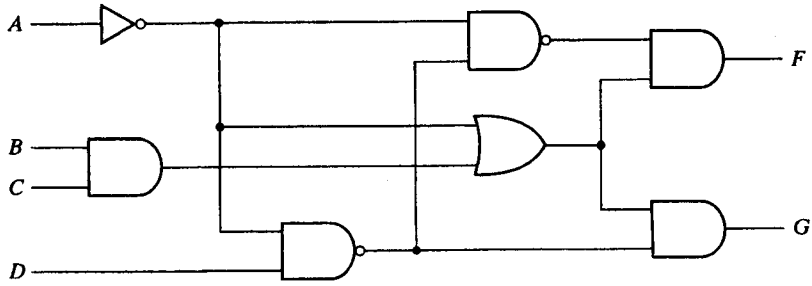
(ب) جدول درستی را با 16 ترکیب ورودی تشکیل دهید. آنگاه مقادیر دودویی T_1 تا T_4 و خروجی‌های F_1 و F_2 را در جدول لیست نمایید.

(پ) توابع بولی خروجی حاصل در بخش (ب) را روی نقشه‌ها رسم کنید و نشان دهید که عبارات بولی ساده شده معادل با آنهایی هستند که در بخش (الف) بدست آمد.



شکل (م ۱-۴)

۴-۲ عبارات بولی ساده شده را برای خروجی F و G برحسب متغیرهای ورودی در مدار شکل (م ۲-۴) بدست آورید.



شکل (م ۲-۴)

۴-۳ برای مدار شکل ۲۶-۴ (بخش ۱۰-۴)

(الف) توابع بولی را برای چهار خروجی برحسب متغیرهای ورودی بنویسید.

(ب) اگر مدار در جدول درستی لیست شده باشد، چند سطر و ستون در جدول وجود دارد؟

۴-۴ یک مدار ترکیبی با سه ورودی و یک خروجی طراحی کنید.

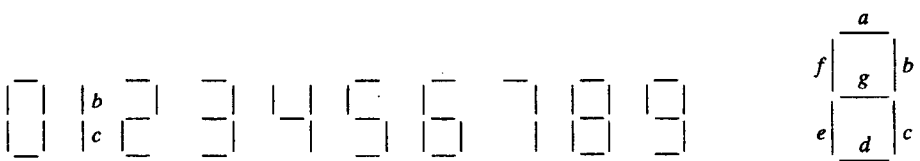
۴-۵ یک مدار ترکیبی با سه ورودی x و y و z و سه خروجی A و B و C طراحی کنید. وقتی ورودی دودویی 0 ، 1 ، 2 یا 3 است، خروجی دودویی یکی بیشتر از ورودی است. وقتی ورودی دودیی 4 ، 5 ، 6 و یا 7 است خروجی دودویی یکی کمتر از ورودی است.

۴-۶ یک مدار اکثریت مداری ترکیبی است که اگر متغیرها بیش از 0 ، مقدار 1 را داشته باشند خروجی‌اش 1 می‌گردد. در غیر این صورت خروجی 0 است. یک مدار اکثریت 3 بیت طراحی کنید.

۴-۷ یک مدار ترکیبی که چهار بیت از کد گری (جدول ۶-۱) را به 4 بیت عدد دودویی تبدیل نماید، طراحی کنید. مدار با XOR طراحی گردد.

۴-۸ یک مبدل کد که یک رقم دهدهی از کد 1- ، 2- ، 4 و 8 را به BCD (جدول ۵-۱) تبدیل کند، طراحی نمایید.

۴-۹ یک دیکدر BCD به هفت قسمتی مداری ترکیبی است که یک رقم دهدهی در BCD را به کد مناسبی جهت انتخاب قطعات در یک نمایشگر تبدیل می‌کند تا ارقام دهدهی به صورت مفهوم قابل مشاهده باشند. هفت خروجی دیکدر (g, f, e, d, c, b, a) قطعات مربوطه را طبق شکل (م ۹-۴ الف) انتخاب می‌کنند. عدد مورد نمایش در شکل (م ۹-۴ ب) دیده می‌شود. یک دیکدر BCD به هفت قسمتی با حداقل گیت طراحی کنید. شش حالت نامعتبر باید به صورت خاموش ظاهر شوند.



(الف) علامت سگمنت (ب) علامت عدد برای نمایش گر

شکل (م ۹-۴)

۴-۱۰ یک مدار ترکیبی مولد متمم 2 چهار بیت طراحی کنید. (خروجی، متمم 2 عدد ورودی را تولید می‌کند) نشان دهید که مدار می‌تواند با گیت‌های XOR ساخته شود. آیا می‌توانید توابع خروجی یک مولد متمم 2 پنج بیت را حدس بزنید.

۴-۱۱ یک مدار ترکیبی افزایش‌گر 4 بیت طراحی نمایید (مداری که 1 را به عدد دودویی 4 بیت اضافه می‌کند.) این مدار را می‌توان با چهار نیم جمع‌کننده طراحی کرد.

۴-۱۲ (الف) یک مدار نیم تفریق‌گر با ورودی‌های x و y و خروجی‌های D و B طراحی کنید. مدار تفریق x−y را انجام می‌دهد سپس تفاضل را در D و رقم قرضی را در B قرار می‌دهد.
(ب) یک مدار تفریق‌گر کامل با سه ورودی x، y و z و دو خروجی D و B طراحی نمایید. مدار تفریق x−y−z را اجرا می‌کند، که z قرضی ورودی، B قرضی خروجی و D تفاضل باقیمانده است.

۴-۱۳ مدار جمع- تفریق‌گر شکل ۴-۱۳ دارای مقادیر زیر برای ورودی مد M و ورودی‌های داده A و B است. در هر حال، مقادیر چهار خروجی SUM، رقم نقلی C و سرریز V را معین کنید.

	M	A	B
(a)	0	0111	0110
(b)	0	1000	1001
(c)	1	1100	1000
(d)	1	0101	1010
(e)	1	0000	0001

۴-۱۴ فرض کنید که گیت XOR دارای یک تأخیر انتشار 20ns و گیت‌های AND و OR دارای تأخیر انتشار 10ns هستند. زمان کل تأخیر انتشار در جمع‌کننده 4 بیت شکل ۴-۱۲ چقدر است؟

۴-۱۵ عبارت بولی دو طبقه را برای نقلی خروجی C_4 در مولد نقلی پیش‌بینی شونده شکل ۴-۱۲ بدست آورید.

۴-۱۶ نشان دهید که نقلی خروجی در یک مدار جمع‌کننده کامل را می‌توان به فرم AND−OR−INVERT

$$C_{i+1} = G_i + P_iC_i = (G_i'P_i + G_i'C_i')'$$

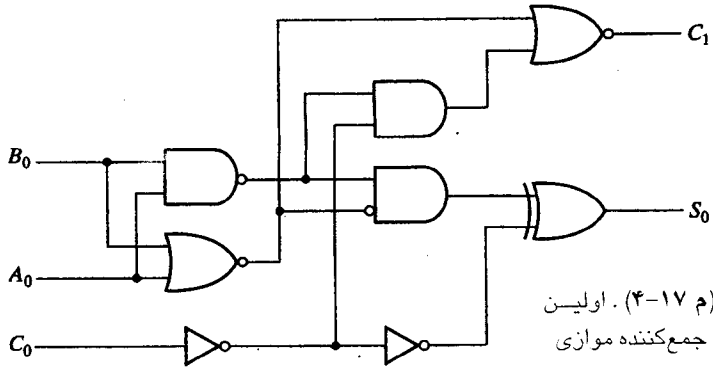
نشان داد. آی‌سی 74182 یک مدار مولد نقلی پیش‌بینی شونده است که نقلی‌ها را با گیت‌های AND−OR−INVERT تولید می‌کند (بخش ۳-۷ ملاحظه شود). فرض بر این است که پایانه‌های ورودی متمم‌های G و P و C_1 را دارند. توابع بول را برای نقلی‌های پیش‌بینی شونده C_2، C_3 و C_4 در این آی‌سی، بدست آورید. (توجه: از روش جایگزینی در بدست آوردن نقلی‌ها برحسب C'_1 استفاده کنید).

۴-۱۷ تولید و انتشار نقلی را به صورت زیر تعریف کنید:

$$P_i = A_i + B_i$$
$$G_i = A_iB_i$$

نشان دهید که نقلی خروجی و جمع خروجی یک جمع‌کننده کامل به صورت زیر است:

$$C_{i+1} = (C_i'G_i' + P_i')'$$
$$S_i = (P_iG_i') \oplus C_i$$



شکل (م ۱۷-۴) . اولین طبقه یک جمع‌کننده موازی

نمودار منطقی اولین طبقه یک جمع‌کننده موازی 4 بیت با استفاده از آی‌سی 74283 در شکل (م ۱۷-۴) نشان داده شده است. پایانه‌های P'_1 ، G'_1 را شناسایی کنید و نشان دهید که مدار یک جمع‌کننده کامل را پیاده‌سازی می‌نماید.

۱۸-۴ یک مدار ترکیبی طراحی کنید تا متمم 9 یک رقم BCD را تولید نماید.

۱۹-۴ یک مدار جمع‌کننده کامل BCD بسازید. جمع‌کننده BCD شکل ۱۴-۴ و متمم 9 مسئله ۱۸-۴ را به کار ببرید. برای قطعات از نمودارهای بلوکی استفاده کنید.

۲۰-۴ یک ضرب‌کننده دودویی که دو عدد 4 بیت را در هم ضرب می‌کند طراحی نمایید.

۲۱-۴ یک مدار ترکیبی که دو عدد 4 بیت را برای مساوی بودن مقایسه می‌کند طراحی کنید. خروجی مدار هنگامی 1 است که دو عدد مساوی باشند. در غیر این صورت 0 است.

۲۲-۴ یک دیکدر افزونی 3 به دودویی با استفاده از ترکیبات به‌کار نرفته بی‌اهمیت طراحی کنید.

۲۳-۴ نمودار منطقی یک دیکدر 2 به 4 با فقط گیت NOR بسازید. یک ورودی فعال‌ساز در آن لحاظ کنید.

۲۴-۴ یک دیکدر BCD به دهدهی با ترکیبات بی‌اهمیت کد BCD طراحی کنید.

۲۵-۴ یک دیکدر 5 به 32 خط با دیکدرهای 3 به 8 خط و فعال‌ساز و یک دیکدر 2 به 4 طراحی نمایید. از نمودار بلوکی برای عناصر استفاده کنید.

۲۶-۴ یک دیکدر 4 به 16 خط با پنج دیکدر 2 به 4 همراه با فعال‌ساز بسازید.

۲۷-۴ یک تابع ترکیبی با سه تابع بولی زیر تعریف شده است.

$$F_1(A, B, C) = \Sigma(2, 4, 7)$$
$$F_2(A, B, C) = \Sigma(0, 3)$$
$$F_3(A, B, C) = \Sigma(0, 2, 3, 4, 7)$$

مدار را با یک دیکدر که ازگیت‌های NAND ساخته شده بسازید (مشابه شکل ۱۹-۴). گیت‌های NAND یا AND به خروجی‌های دیکدر متصل‌اند. از نمودار بلوکی برای دیکدر استفاده نمایید. تعداد ورودی‌ها را در گیت‌های بیرونی حداقل نمایید.

۴-۲۸ یک مدار ترکیبی با سه تابع بولی زیر تعریف شده است.

$$F_1 = x'y'z' + xz$$
$$F_2 = xy'z' + x'y$$
$$F_3 = x'y'z + xy$$

مدار را با دیکدر و گیت‌های بیرونی بسازید.

۴-۲۹ یک انکدر اولویت 4 ورودی با ورودی‌هایی مثل جدول ۸-۴ طراحی کنید، ولی D_0 بالاترین اولویت و D_3 پایین‌ترین اولویت را داشته باشد.

۴-۳۰ جدول درستی یک انکدر اولویت هشت هشتی به دودویی را معین کنید. یک خروجی V برای مشخص نمودن این‌که حداقل یکی از ورودی‌ها وجود دارد، فراهم نمایید. ورودی‌که بزرگترین عدد را در اندیس دارد بالاترین اولویت را داراست. اگر ورودی‌های D_5 و D_3 به طور همزمان 1 شوند، مقدار چهار خروجی چیست؟

۴-۳۱ یک مولتی پلکسر 1 × 16 با دو مولتی پلکسر 1 × 8 و یک مولتی پلکسر 1 × 2 بسازید. نمودارهای بلوکی را به‌کار ببرید.

۴-۳۲ تابع بولی زیر را با یک مولتی پلکسر پیاده سازی کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$$

۴-۳۳ یک جمع‌کننده کامل را با دو مولتی پلکسر 1 × 4 پیاده کنید.

۴-۳۴ یک مولتی پلکسر 1 × 8 دارای ورودی‌های A، B و C متصل به ورودی‌های انتخاب S_2، S_1 و S_0 است. ورودی‌های داده I_0 تا I_7 به قرار زیراند: $I_1 = I_2 = I_7 = 0$؛ $I_3 = I_5 = 1$؛ $I_0 = I_4 = D$ و $I_6 = D'$. تابع بولی که با آن مولتی پلکسر پیاده شده را معین نمایید.

۴-۳۵ تابع بولی زیر را با یک مولتی‌پلکسر 1 × 4 و گیت‌های بیرونی پیاده سازی کنید. ورودی‌های A و B را به خطوط انتخاب وصل کنید. ورودی‌های چهار خط داده تابعی از متغیرهای C و D است. این مقادیر با بیان F به عنوان تابعی از C و D برای هر چهار حالت وقتی AB = 00 ، 01 ، 10 ، 11 باشد، بدست می‌آید. این توابع را می‌توان با گیت‌های بیرونی پیاده‌سازی کرد.

$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$$

۴-۳۶ یک توصیف HDL در سطح گیت برای مدار انکدر اولویت شکل ۲۳-۴ بنویسید.

۴-۳۷ یک توصیف سلسله مراتبی HDL در سطح گیت برای جمع- تفریق‌گر 4 بیت با اعداد بی‌علامت بنویسید. مدار مشابه شکل ۱۳-۴ است ولی خروجی V در آن وجود ندارد. می‌توانید جمع‌کننده کامل 4 بیت مثال ۲-۴ HDL را ذکر کنید.

۴-۳۸ یک توصیف روند داده را برای مولتی پلکسر 2 به 1 چهارتایی با فعال‌ساز بنویسید (شکل ۲۶-۴).

۴-۳۹ یک توصیف رفتاری HDL از یک مقایسه‌گر 4 بیت با 6 خروجی Y[5 : 0] بنویسید. بیت 5 از Y برای مساوی، بیت 4 برای نامساوی، بیت 3 بزرگتر از، بیت 2 برای کمتر از، بیت 1 برای بزرگتر یا مساوی، و بیت 0 برای کوچکتر یا مساوی در نظر گرفته شده است.

۴-۴۰ یک توصیف روند داده برای جمع - تفریق‌گر 4 بیت برای اعداد بی‌علامت بنویسید. از مقایسه‌گر شرطی (: ?) استفاده کنید.

۴-۴۱ مسئله ۴-۴۰ را با مدل‌سازی رفتاری تکرار نمایید.

۴-۴۲ (الف) یک توصیف HDL سطح گیت مدار مبدل BCD به افزونی -3 مطابق شکل ۴-۴ بنویسید.

(ب) یک توصیف HDL روند داده برای مبدل BCD به افزونی -3 با استفاده از عبارات بولی لیست شده در شکل ۴-۳ بنویسید.

(پ) یک توصیف رفتاری برای مدار مبدل BCD به افزونی -3 بنویسید.

(ت) یک برنامه تست برای شبیه‌سازی مدار مبدل BCD به افزونی -3 بنویسید تا صحت جدول درستی تحقیق شود. هر سه مدار را چک کنید.

۴-۴۳ عمل مدار را با توصیف HDL زیر توضیح دهید.

```
module Prob438 (A,B,S,E,Q);
   input  [1:0] A, B;
   input  S, E;
   output [1:0] Q;
   assign Q = E ? (S ? A : B) : 'bz;
endmodule
```

۴-۴۴ یک توصیف مداری HDL برای یک مدار حسابی- منطقی 4 بیت (ALU) بنویسید. مدار دو عمل حسابی و دو عمل منطق را اجرا می‌کند که با ورودی 2 بیت انتخاب می‌شوند. چهار عمل عبارتند از جمع، تفریق، AND و OR.

۴-۴۵ یک توصیف رفتاری HDL برای یک انکدر اولویت 4 بیتی بنویسید. از یک بردار 4 بیت برای ورودی‌های D و یک بلوک always با عبارت if –else استفاده کنید. فرض نمایید که ورودی D[3] دارای بالاترین اولویت باشد.

مراجع

1. DIETMEYER, D. L. 1988. *Logic Design of Digital Systems.* 3rd ed. Boston: Allyn Bacon.
2. GAJSKI, D. D. 1997. *Principles of Digital Design.* Upper Saddle River, NJ: Prentice Hall.
3. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.
4. KATZ, R. H. 1994. *Contemporary Logic Design.* Upper Saddle River, NJ: Prentice Hall.
5. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.
6. NELSON V. P., H. T. NAGLE, J. D. IRWIN and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.
7. ROTH, C. H. 1992. *Fundamentals of Logic Design.* 4th ed. St. Paul: West.
8. WAKERLY, J. F. 2000. *Digital Design, Principles and Practices.* 3rd ed. Upper Saddle River, NJ: Prentice Hall.
9. BHASKER, J. 1997. *A Verilog HDL Primer.* Allentown, PA: Star Galaxy Press.
10. BHASKER, J. 1998. *Verilog HDL Synthesis.* Allentown, PA: Star Galaxy Press.
11. CILETTI, M. D. 1999. *Modeling, Synthesis, and Rapid Prototyping with Verilog HDL.* Upper Saddle River, NJ: Prentice Hall.
12. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis.* SunSoft Press (A Prentice Hall Title).

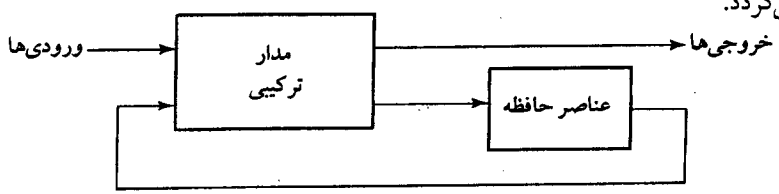
۵

مدارهای منطقی ترتیبی همزمان

۵–۱ مدارهای ترتیبی

مدارهای دیجیتالی که تاکنون بررسی شدند از نوع ترکیبی بودند. در این مدارها خروجی‌ها همه به ورودی‌های جاری وابسته‌اند. گرچه به نظر می‌رسد که هر سیستم دیجیتال دارای مدارهای ترکیبی است، بسیاری از سیستم‌هایی که در عمل با آن مواجه هستیم حاوی عناصر حافظه هم می‌باشند و بنابراین لازم است تا این سیستم‌ها برحسب منطق ترتیبی مورد بررسی قرار گیرند.

نمودار بلوکی یک مدار ترتیبی در شکل ۵–۱ نشان داده شده است. این مدار متشکل از مداری ترکیبی است که عناصر حافظه برای ایجاد یک مسیر پسخورد به آن وصل شده‌اند. عناصر حافظه قطعاتی هستند که می‌توانند اطلاعات دودویی را ذخیره کنند. اطلاعات دودویی ذخیره شده در این عناصر در هر لحظه از زمان حالت مدار ترتیبی در آن زمان است. مدار ترتیبی اطلاعات دودویی را از ورودی‌های بیرونی دریافت می‌کند. این ورودی‌ها همراه با حالت فعلی عناصر حافظه، مقدار دودویی خروجی‌ها را معین می‌نماید. آنها شرط تغییر حالت در عناصر حافظه را نیز معین می‌سازند. نمودار بلوکی نشان می‌دهد که خروجی‌های یک مدار ترتیبی نه فقط تابعی از ورودی‌ها هستند، بلکه به حالت فعلی عناصر حافظه نیز وابسته می‌باشند. حالت بعدی عناصر حافظه نیز تابعی از ورودی‌های بیرونی و حالت فعلی است. بنابراین یک مدار ترتیبی با ترتیب زمانی ورودی‌ها، خروجی‌ها و حالات داخلی مشخص می‌گردد.

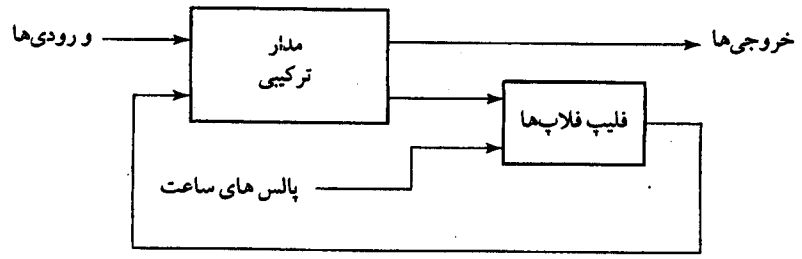


شکل ۵–۱ نمودار بلوکی یک مدار ترتیبی

دو نوع مدار ترتیبی وجود دارد که دسته‌بندی آنها به زمانبندی سیگنال آنها وابسته است. مدار ترتیبی همزمان یا همگام سیستمی است که رفتار آن با توجه به دانش و آگاهی از سیگنال‌هایش در هر لحظه گسسته‌ای از زمان قابل تعریف می‌باشد. رفتار یک مدار ترتیبی غیرهمزمان به ترتیب تغییر سیگنال‌های ورودی آن که می‌توانند در هر لحظه از زمان روی مدار تأثیر کنند وابسته می‌باشد. عناصر حافظه‌ای که به طور معمول در مدارهای ترتیبی غیرهمزمان به کار می‌روند نوعی وسایل تأخیر زمانی هستند. قابلیت نگهداری یک وسیله تأخیر زمانی به زمان انتشار سیگنال در وسیله بستگی دارد. در عمل تأخیر انتشار در گیت‌های منطقی درونی برای ایجاد تأخیر کفایت می‌کند بنابراین واحد تأخیر واقعی می‌تواند مورد نیاز نباشد. در سیستم‌های غیرهمزمان نوع گیتی، عناصر حافظه متشکل از گیت‌های منطقی است که در واقع تأخیر انتشار آنها عمل ذخیره‌سازی را تداعی می‌نماید. بنابراین در چنین مواقعی یک مدار ترتیبی غیرهمزمان را می‌توان مداری ترکیبی با پسخورد دانست. به دلیل وجود پسخورد در بین گیت‌های منطقی، هر مدار ترتیبی غیرهمزمان هر لحظه ممکن است ناپایدار شود. مسئله بی‌ثباتی حاکم مشکلات عدیده‌ای را بر طراح تحمیل خواهد کرد. مدارهای ترتیبی غیرهمزمان در فصل ۹ مورد بحث قرار گرفته‌اند.

با توجه به تعریف، یک مدار ترتیبی همزمان سیگنال‌هایی را مورد استفاده قرار می‌دهد که فقط در لحظات گسسته‌ای از زمان روی عناصر حافظه‌اش اثر می‌گذارد. در این مدارها همزمانی با وسیله‌ای به نام مولد ساعت تحقق می‌یابد و طی آن رشته متناوبی از پالس ساعت به وسیله این دستگاه تولید می‌گردد. پالس‌های ساعت در سرتاسر سیستم توزیع می‌گردند به نحوی که عناصر حافظه تنها هنگام رسیدن هر پالس تحت تأثیر ورودی خود قرار می‌گیرند. در عمل پالس‌های ساعت به همراه دیگر پالس‌ها که تغییرات لازم را در حافظه ایجاد می‌کنند همراه هستند. مدارهای ترتیبی همزمانی که پالس‌های ساعت را در ورودی عناصر ذخیره‌ساز خود به کار می‌برند، مدارهای ترتیبی ساعت‌دار خوانده می‌شوند. ما غالباً در عمل با مدارهای ترتیبی ساعت‌دار مواجه هستیم. آنها مشکل ناپایداری را ندارند و موضوع زمانبندی در آنها به راحتی به مراحلی گسسته و مستقل شکسته می‌شود. هر یک از این مراحل یا برش‌های زمانی مستقلاً قابل بررسی می‌باشند.

عناصر ذخیره‌سازی در مدارهای ترتیبی ساعت‌دار را فلیپ فلاپ می‌گویند. فلیپ فلاپ یک وسیله ذخیره‌سازی دودویی بوده و قادر است یک بیت از اطلاعات را در خود ذخیره نماید. یک مدار ترتیبی ممکن است در صورت لزوم تعداد قابل توجهی از این فلیپ فلاپ‌ها را به کار ببرد. نمودار بلوکی یک مدار ترتیبی ساعت‌دار همزمان در شکل ۲-۵ دیده می‌شود. خروجی‌ها می‌توانند از یک مدار ترکیبی، یا از فلیپ فلاپ‌ها و یا هر دو حاصل شوند. فلیپ فلاپ‌ها ورودی‌های خود را از مدار ترکیبی و نیز از سیگنال ساعت که با فواصل زمانی رخ می‌دهند، طبق نمودار زمانی دریافت می‌کنند. حالت فلیپ فلاپ‌ها تنها هنگام تغییر وضعیت یک پالس ساعت عوض می‌شود. وقتی یک پالس ساعت فعال نیست، حلقه پسخورد قطع می‌شود زیرا حتی اگر خروجی‌های مدار ترکیبی که ورودی آنها را تغذیه می‌کند عوض شود خروجی‌های فلیپ فلاپ تغییر نمی‌نمایند. بنابراین تغییر وضعیت از یک حالت به بعدی فقط در فواصل زمانی دیکته شده به وسیله پالس‌های ساعت امکان‌پذیر است.



(الف) نمودار بلوکی

(ب) نمودار زمان بندی پالس‌های ساعت

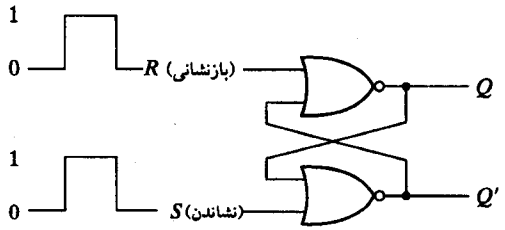
شکل ۲-۵ . مدار ترتیبی ساعت‌دار همزمان

۲-۵ لچ‌ها

یک فلیپ فلاپ می‌تواند یک حالت دودویی را مادامی که تغذیه به مدارش اعمال شود، تا مدتی نامحدود نگهدارد. تفاوت عمده بین انواع فلیپ فلاپ‌ها، در تعداد ورودی‌ها و نحوه تأثیر آنها در تغییر حالت دودویی است. ساده‌ترین انواع فلیپ فلاپ‌ها که با سطوح سیگنال عمل می‌کنند، لچ نامیده می‌شوند. لچ‌ها (یا نگهدارها) مدارهای مبنایی هستند که همه فلیپ فلاپ‌ها با آنها ساخته می‌شوند. گرچه لچ‌ها برای ذخیره اطلاعات دودویی و طراحی مدارهای ترتیبی غیرهمزمان (بخش ۳-۹) مفیدند، ولی عملاً در مدارهای ترتیبی همزمان به کار نمی‌روند. انواع فلیپ فلاپ‌هایی که در مدارهای ترتیبی مورد استفاده قرار می‌گیرند در بخش بعد معرفی شده‌اند.

لچ SR

لچ SR مداری با دو گیت NAND یا NOR است که به طور متقاطع به هم وصل شده‌اند. این مدار دو ورودی دارد که با S به معنی نشاندن (set) و R برای بازنشانی (Reset) نام‌گذاری شده‌اند. لچ SR ساخته شده از دو گیت NOR در شکل ۳-۵ دیده می‌شود. لچ دارای دو حالت مفید است. وقتی



(الف) نمودار منطقی

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(ب) جدول تابع

شکل ۳-۵ . لچ SR با گیت‌های NOR

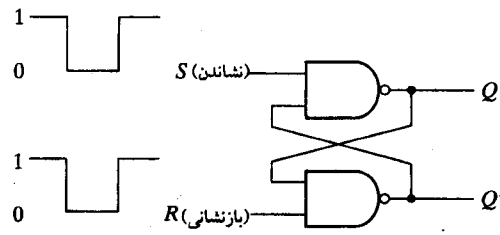
خروجی $Q = 1$ و $Q' = 0$ باشند گوییم که لچ در حالت نشانده (منطق 1) است. اگر $Q = 0$ و $Q' = 1$ باشد گوییم در حالت بازنشانی (منطق 0) است. خروجی‌های Q و Q' متمم یکدیگرند. با این وجود، وقتی هر دو ورودی به طور همزمان 1 شوند، حالت تعریف نشده 0 برای دو خروجی رخ می‌دهد.

تحت شرایط معمولی، هر دو ورودی در 0 نگهداری می‌شوند مگر این که بخواهیم حالت لچ را عوض کنیم. اعمال یک لحظه 1 به ورودی S موجب می‌شود که لچ به حالت 1 برود. قبل از این که حالت تعریف نشده‌ای رخ دهد، ورودی S باید به 0 بازگردد. طبق جدول تابع در شکل ۳-۵ (ب)، دو حالت از ورودی موجب می‌شود تا مدار در حالت 1 قرار گیرد. اولین حالت ($S = 1$ ، $R = 0$) نقشی است که طی آن ورودی S، مدار را به حالت 1 می‌برد. حذف ورودی فعال از S، مدار را در همان حالت باقی می‌گذارد. پس از بازگشت هر دو ورودی به 0، امکان رفتن به حالت 0 میسر خواهد شد، به این ترتیب که برای یک لحظه یک 1 به R اعمال می‌گردد. سپس می‌توان 1 را از R حذف کرد و در این حال مدار در حالت 0 باقی خواهد ماند. بنابراین وقتی هر دو ورودی S و R برابر 0اند، بسته به این که کدام ورودی اخیراً 1 شده است، لچ می‌تواند در حالت 1 یا 0 قرار گیرد.

اگر به طور همزمان به هر دو ورودی R و S، 1 دودویی را اعمال کنیم، هر دو خروجی به 0 می‌روند. این ورودی‌ها حالت تعریف نشده‌ای را در خروجی ایجاد می‌کنند، زیرا حالت بعدی پیش‌بینی نشده‌ای را به هنگام بازگشت دو ورودی به 0 نتیجه می‌دهد. در حالت کار معمولی لچ، با اطمینان از اعمال نشدن همزمان 1 به ورودی‌ها، این وضعیت پرهیز می‌گردد.

لچ SR با دو گیت NAND متقاطع در شکل ۴-۵ مشاهده می‌شود. این مدار به طور معمول با 1 در هر دو ورودی‌اش کار می‌کند مگر این که بخواهیم حالت لچ را تغییر دهیم. اعمال 0 به S موجب می‌شود Q به 1 برود، و لچ را به حالت نشانده وا دارد. وقتی که ورودی S به 1 بازگردد، مدار در همان حالت 1 باقی می‌ماند. پس از بازگشت هر دو ورودی به 1، ما مجاز به تغییر حالت لچ با استقرار 0 در R هستیم. این موجب می‌شود تا مدار به حالت بازنشانی برود و حتی پس از بازگشت هر دو ورودی به 1، لچ در همان حال بماند. حالتی که برای لچ NAND غیر مجاز است، هنگامی است که هر دو ورودی به طور همزمان در 0 باشند. بنابراین از وقوع این حالت باید ممانعت کرد.

با مقایسه لچ NOR با NAND مشاهده می‌شود که سیگنال‌های ورودی برای NAND متمم

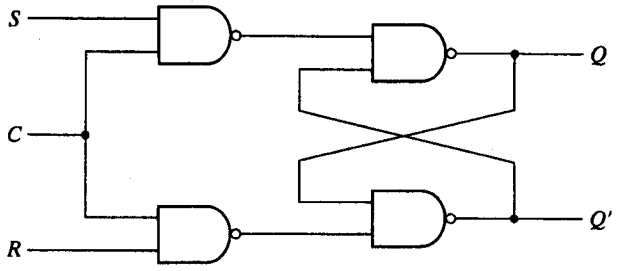


(الف) نمودار منطقی

S	R	Q	Q′	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(ب) جدول تابع

شکل ۴-۵ . لچ SR با گیت‌های NAND



(الف) نمودار منطقی

C	S	R	حالت بعدی Q
0	X	X	بلاتغییر
1	0	0	بلاتغییر
1	0	1	حالت بازنشانی; $Q = 0$
1	1	0	حالت نشاندن ; $Q = 1$
1	1	1	نامعین

(ب) جدول درستی

شکل ۵-۵. لچ SR با ورودی کنترل

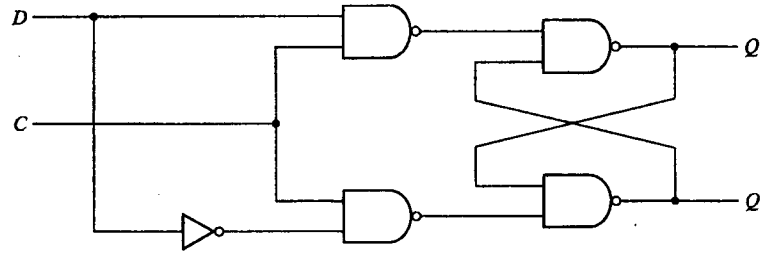
ورودی‌های لچ NOR است. چون لچ NAND برای تغییر حالت خود به سیگنال 0 نیاز دارد، گاهی آن را لچ $S'-R'$ می‌خوانند. علامت پریم یا خط بار بر روی حروف، بیانگر این حقیقت است که ورودی‌ها باید در حالت متمم خود باشند تا مدار را فعال کنند.

عملکرد لچ SR با افزودن یک ورودی کنترل برای تعیین زمان تغییر حالت لچ اصلاح می‌گردد. یک لچ کنترل‌دار در شکل ۵-۵ مشاهده می‌شود. این مدار شامل یک لچ SR پایه و دو گیت NAND اضافی است. ورودی کنترل C به عنوان یک سیگنال فعال‌ساز برای دو ورودی عمل می‌کند. مادامی که ورودی کنترل در 0 باقی بماند، خروجی گیت‌های NAND در سطح منطقی 1 باقی می‌مانند. این وضعیت حالت سکون برای لچ SR است. حالت نشاندن با $S = 1$ ، $R = 0$ و $C = 1$ حاصل می‌گردد. برای تغییر وضعیت باید $S = 0$ ، $R = 1$ و $C = 1$ باشد. در هر حال، وقتی که C به 0 بازگردد، مدار در حالت جاری باقی می‌ماند. در ورودی کنترل با اعمال 0 به C ، مدار غیرفعال می‌شود، به نحوی که عدم تغییر حالت مستقل از مقادیر S و R ، نیز می‌باشد. به علاوه وقتی $C = 1$ باشد، و هر دو ورودی S و R برابر 0 باشند، باز هم حالت مدار تغییر نخواهد کرد. این حالات در جدول تابع در کنار نمودار، ملاحظه می‌شوند.

حالت نامعین هنگامی رخ می‌دهد که هر سه ورودی برابر 1 باشند. این وضعیت، مقدار 0 را روی هر دو ورودی لچ SR پایه قرار می‌دهد، که این ورودی‌ها حالت نامعین را برقرار می‌نمایند. وقتی که ورودی کنترل به 0 بازمی‌گردد، نمی‌توان حالت بعدی را معین کرد زیرا بستگی دارد که کدام یک از دو ورودی S و R زودتر به 0 بروند. این حالت نامعین موجب می‌گردد تا اداره مدار مشکل باشد و بنابراین به ندرت به کار گرفته می‌شود. با این وجود، مدار از اهمیت لازم برخوردار است زیرا دیگر لچ‌ها و فلیپ فلاپ‌ها با آن ساخته می‌شوند.

لچ D

یکی از راه‌های حذف حالت نامطلوب یعنی حالت نامعین یا غیر مجاز در لچ SR این است که مطمئن شویم S و R هرگز به طور همزمان به 1 نمی‌روند. این کار با لچ D شکل ۶-۵ میسراست. این لچ تنها دو ورودی دارد: D (داده) و C (کنترل). ورودی D مستقیماً به ورودی S و متمم آن به ورودی R وصل می‌شود. مادامی که ورودی کنترل در 0 قرار دارد، لچ SR متقاطع دارای 1 در هر دو ورودی بوده و



(الف) نمودار منطقی

C	D	حالت بعدی Q
0	X	بلاتغییر
1	0	حالت بازنشانی; Q = 0
1	1	حالت نشاندن; Q = 1

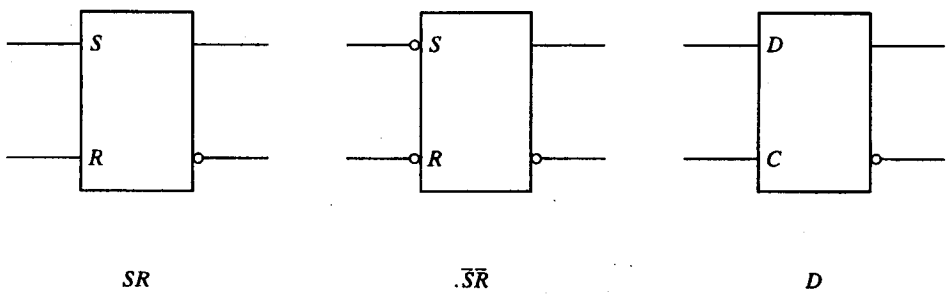
(ب) جدول درستی

شکل ۶-۵. لچ D

مدار نمی‌تواند تغییر حالت دهد. در واقع مقدار D هم نقشی ندارد. وقتی C = 1 باشد ورودی D نمونه‌برداری می‌شود. اگر D = 1 باشد خروجی Q به 1 می‌رود، به این ترتیب مدار در حالت نشانده است. اگر D = 0، خروجی Q به 0 رفته و مدار را به حالت بازنشانی می‌برد.

لچ D نامش را از قابلیت نگهداری داده در درون دریافت کرده است. این لچ برای ذخیره موقت اطلاعات دودویی بین یک محیط و یک واحد مناسب است. اطلاعات دودویی حاضر در ورودی داده لچ D هنگامی که ورودی کنترل فعال شود، به خروجی Q منتقل می‌گردد. مادامی که ورودی کنترل فعال است، خروجی تغییرات ورودی را دنبال می‌کند. این وضعیت مسیری از D به خروجی ایجاد می‌کند، و به این دلیل مدار را لچ شفاف هم می‌خوانند. وقتی ورودی کنترل غیرفعال شود، اطلاعات دودویی حاضر قبلی در ورودی، در خروجی Q باقی می‌ماند تا دوباره ورودی کنترل فعال گردد.

سمبل گرافیک برای انواع لچ در شکل ۷-۵ آمده است. لچ با یک بلوک مستطیلی مشخص می‌شود، که در آن ورودی‌ها در سمت چپ و خروجی‌ها در سمت راست نشان داده می‌شوند. یکی از خروجی‌ها، خروجی معمولی و دیگری متمم خروجی معمولی را نشان می‌دهد. نمودار گرافیک لچ SR دارای ورودی‌های S و R می‌باشد که در داخل بلوک ذکر شده‌اند. در لچ گیت NAND به ورودی‌ها حباب‌هایی اضافه می‌شود که بیانگر نشانده شدن و بازنشانی با سیگنال منطقی 0 است. نمودار گرافیکی برای ورودی‌های D دارای ورودی‌های D و C است که در داخل بلوک مشخص شده‌اند.



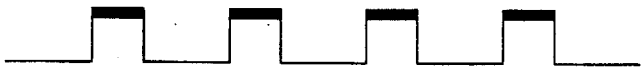
شکل ۷-۵. سمبل‌های گرافیکی لچ

۵-۳ فلیپ فلاپ‌ها

حالت یک لچ یا یک فلیپ فلاپ با تغییر در ورودی کنترل عوض می‌شود. این تغییر لحظه‌ای را ژیگرگویند و انتقال مربوط به آن را تریگر کردن فلیپ فلاپ خوانند. لچ D با پالس‌ها در ورودی کنترلش اساساً یک فلیپ فلاپ است که هر زمان پالس به سطح منطقی 1 برود تریگر می‌شود. مادامی که پالس ورودی کنترل در این سطح بماند هر تغییری در ورودی داده، خروجی و حالت لچ را عوض خواهد کرد.

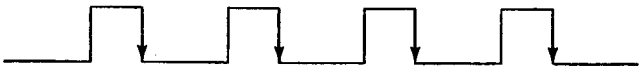
همانطور که از نمودار بلوکی شکل ۲-۵ ملاحظه می‌شود، یک مدار ترتیبی از خروجی‌های فلیپ فلاپ به ورودی‌های مدار ترکیبی دارای مسیر پسخورد است. در نتیجه ورودی‌های فلیپ فلاپ ممکن است از خروجی همان و یا دیگر فلیپ فلاپ‌ها راه‌اندازی شوند. وقتی که لچ‌ها به عنوان عناصر مورد استفاده قرار گیرند، مشکلی اساسی به وجود می‌آید. به محض تغییر پالس ساعت به منطق 1، انتقال حالت لچ‌ها آغاز می‌شود. در حالی که پالس ساعت هنوز فعال است، حالت جدید لچ در خروجی ظاهر می‌گردد. این خروجی به ورودی لچ‌ها از طریق مدار ترکیبی وصل می‌شود. اگر پالس ساعت در منطق 1، باشد و ورودی اعمال شده به لچ‌ها تغییر کند، لچ به مقادیر جدید واکنش نشان داده و خروجی جدیدی رخ خواهد داد. نتیجه این واکنش وضعیت پیش‌بینی نشده‌ای است زیرا حالت لچ‌ها ممکن است با قرار داشتن پالس ساعت در سطح فعال همچنان به تغییر خود ادامه دهد. به دلیل این عملکرد غیرمطلوب، خروجی یک لچ وقتی همه لچ‌ها به منبع ساعت مشترکی وصلند نمی‌تواند مستقیماً و یا از طریق یک مدار منطقی به همان لچ یا دیگر لچ‌ها وصل شود.

فلیپ فلاپ‌ها طوری ساخته می‌شوند که وقتی بخشی از نوع مدار ترتیبی‌اند و از ساعت مشترکی استفاده می‌کنند، عملکردشان صحیح باشد. مشکل لچ این است که به سطح پالس ساعت پاسخ می‌دهد. همانطور که در شکل ۸-۵ (الف) مشاهده می‌شود، وقتی که پالس ساعت در منطق 1 قرار دارد، هر تغییر مثبت در ورودی کنترل موجب می‌شود تا به ازاء هر تغییر در ورودی D، تغییری در خروجی به وجود



(الف) پاسخ به سطح مثبت

(ب) پاسخ به لبه مثبت



(پ) پاسخ به لبه منفی

شکل ۸-۵. پاسخ ساعت در لچ و فلیپ فلاپ

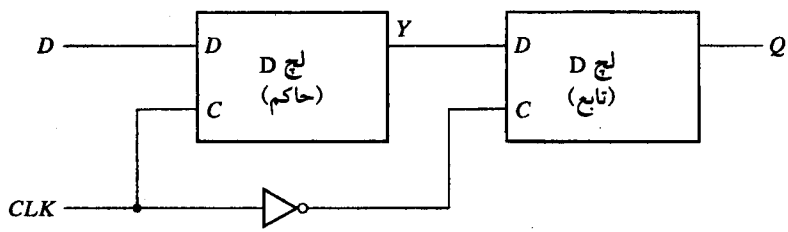
آید. نکته کلیدی در یک عملکرد صحیح فلیپ فلاپ‌ها تریگر شدن آنها در زمان گذر سیگنال است. پالس ساعت از دو انتقال 1 به 0 و 0 به 1 گذر می‌کند. طبق شکل ۸-۵ گذر مثبت به عنوان لبه مثبت و گذر منفی به نام لبه منفی شناخته می‌شود. برای اصلاح یک لچ به یک فلیپ فلاپ، دو راه وجود دارد. یکی از این روش‌ها استفاده از دو لچ با آرایشی خاص است که خروجی فلیپ فلاپ را در حین تغییر ورودی، از آن جدا می‌سازد. راه دیگر تهیه فلیپ فلاپی است که فقط در حین گذر سیگنال تریگر می‌شود (از 0 به 1 یا از 1 به 0) و در بقیه لحظات پالس ساعت غیرفعال است. اکنون هر دو روش را مطالعه می‌کنیم.

فلیپ فلاپ D حساس به لبه

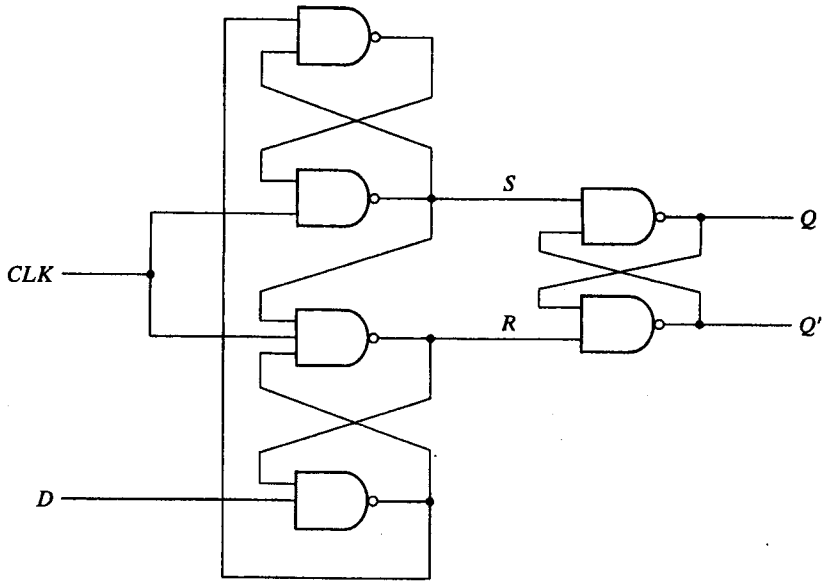
ساخت یک فلیپ فلاپ D با دو لچ D و یک وارونگر در شکل ۹-۵ ملاحظه می‌گردد. اولین لچ را حاکم و دومی را تابع می‌گویند. مدار، ورودی D را نمونه‌برداری کرده و خروجی Q را فقط در لبه منفی پالس کنترل ساعت (CLK) تغییر می‌دهد. وقتی که پالس ساعت در 0 است، خروجی وارونگر 1 می‌باشد. لچ تابع فعال شده و خروجی آن، Q، برابر با خروجی حاکم یعنی Y خواهد شد. لچ حاکم غیرفعال است زیرا CLK = 0 می‌باشد. وقتی که پالس ساعت ورودی به سطح 1 تغییر وضعیت می‌دهد، داده از ورودی بیرونی D به حاکم منتقل می‌گردد. در این حال، مادامی که ساعت در سطح 1 بماند، تابع غیرفعال خواهد بود زیرا ورودی C آن برابر 0 است. هر تغییر در ورودی، خروجی Y را عوض می‌کند، ولی نمی‌تواند خروجی تابع را عوض کند. وقتی که پالس ساعت به 0 بازگردد، حاکم غیرفعال شده و از ورودی D جدا می‌شود. در همان زمان تابع فعال شده و مقدار Y به خروجی فلیپ فلاپ در Q انتقال می‌یابد. بنابراین خروجی فلیپ فلاپ فقط در حین گذر پالس ساعت از 1 به 0 تغییر می‌کند.

رفتار فلیپ فلاپ حاکم - تابع که در بالا توصیف شد نشان می‌دهد که خروجی فقط در لبه منفی پالس ساعت تغییر می‌نماید. این تغییر را می‌توان در لبه مثبت پالس ساعت هم انجام داد. این کار به این ترتیب صورت می‌گیرد که یک وارونگر اضافی بین پایانه CLK و اتصال بین وارونگر دیگر و ورودی C لچ حاکم قرار گیرد. چنین فلیپ فلاپی با لبه منفی پالس عمل کرده و به این ترتیب لبه منفی حاکم و لبه مثبت نیز تابع و پایانه خروجی را عوض می‌کند.

نمونه دیگری از فلیپ فلاپ D حساس به لبه از سه لچ SR، مطابق شکل ۱۰-۵ استفاده می‌کند. دو



شکل ۹-۵. فلیپ فلاپ D حاکم- تابع

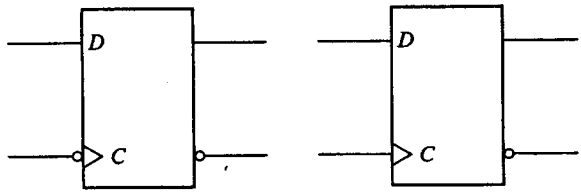


شکل ۱۰-۵. فلیپ فلاپ D حساس به لبه مثبت

لچ موجود در این شکل به ورودی‌های بیرونی D (داده) و CLK (ساعت) پاسخ می‌دهند. لچ سوم خروجی را برای فلیپ فلاپ تهیه می‌کند. ورودی‌های S و R لچ خروجی در CLK = 0 در سطح منطق 1 نگهداری می‌شوند. این موجب می‌شود تا خروجی در حالت فعلی خود باقی بماند. ورودی D ممکن است برابر 0 یا 1 باشد. اگر هنگام 1 شدن CLK ، D = 0 برقرار باشد، R به 0 تغییر می‌کند. یعنی فلیپ فلاپ به حالت بازنشان رفته و در آن Q = 0 می‌گردد. اگر در زمان CLK = 1 تغییری در ورودی رخ دهد پایانه R در 0 می‌ماند. بنابراین فلیپ فلاپ علیرغم تغییر در ورودی خود به حالت قفل باقی خواهد ماند. وقتی که ساعت به 0 بازگردد، R به 1 می‌رود و لچ خروجی در وضعیت ساکن و بدون تغییر در خروجی باقی می‌ماند. به طور مشابه وقتی CLK از 0 به 1 می‌رود، اگر D = 1 باشد، S به 0 تغییر می‌کند. این موجب می‌شود تا مدار به حالت 1 رفته و Q = 1 گردد. هر تغییر در D ، مادامی که CLK = 1 است، نمی‌تواند خروجی را تحت تأثیر قرار دهد.

به طور خلاصه، وقتی ساعت ورودی در فلیپ فلاپ حساس به لبه مثبت یک انتقال مثبت انجام دهد، مقدار D به Q منتقل می‌شود. یک لبه منفی از 1 به 0 تأثیری بر روی خروجی ندارد. به همین ترتیب سطح منطق 1، و نیز سطح منطق 0 هم خروجی را عوض نمی‌کنند. از این رو این نوع فلیپ فلاپ تنها به لبه 0 به 1 و لاغیر پاسخ می‌دهد.

هنگام استفاده از فلیپ فلاپ حساس به لبه باید زمانبندی پاسخ فلیپ فلاپ تحت بررسی قرار گیرد. در این زمانبندی، حداقل زمانی به نام زمان برپایی وجود دارد که طی آن قبل از وقوع گذر ساعت، ورودی باید در مقدار ثابت خود نگهداری شود. به همین ترتیب حداقل زمانی به نام زمان نگهداری وجود دارد



(الف) لبه مثبت (ب) لبه منفی

شکل ۱۱-۵ . سمبل گرافیکی فلیپ فلاپ D حساس به لبه

که طی آن ورودی D نباید پس از اعمال لبه مثبت ساعت، تغییر کند. تأخیر انتشار به صورت فاصله زمانی بین لبه تریگر شدن و تثبیت خروجی در حالت جدید تعریف می‌گردد. این و دیگر پارامترها در برگه‌های اطلاعاتی سازندگان برای هر خانواده منطقی ارائه می‌شوند.

سمبل گرافیکی فلیپ فلاپ D حساس به لبه در شکل ۱۱-۵ مشاهده می‌شود. این سمبل مشابه با سمبل لچ D است به جز این که در جلو حرف C علامت فلشی وجود دارد که دینامیکی بودن ورودی را نشان می‌دهد. نشانگر دینامیک به این معنی است که فلیپ فلاپ به گذر لبه ساعت حساس است. وجود یک حباب در ورودی دینامیکی به معنی نیاز به لبه منفی ساعت است. عدم وجود حباب پاسخ به لبه مثبت را نشان می‌دهد.

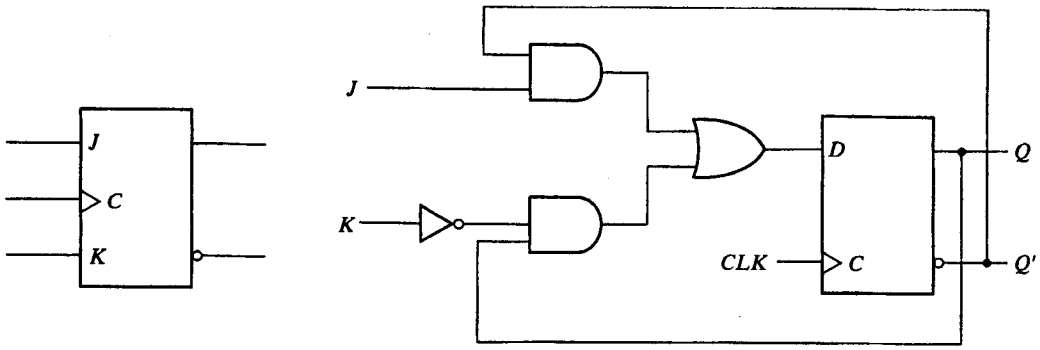
دیگر فلیپ فلاپ‌ها

مدارهای مجتمع VLSI حاوی هزاران گیت در داخل بسته اند. در ساخت یک سیستم دیجیتال انواع گیت‌های درون بسته به هم مرتبط می‌شوند. اقتصادی‌ترین و بهترین فلیپ فلاپ قابل ساخت، نوع D حساس به لبه می‌باشد که به تعداد کمتری گیت نیاز دارد. دیگر فلیپ فلاپ‌ها را می‌توان با فلیپ فلاپ D و مقداری مدار بیرونی به وجود آورد. دو فلیپ فلاپ رایج در طراحی سیستم‌های دیجیتال عبارتند از: فلیپ فلاپ JK و T .

با یک فلیپ فلاپ سه عمل را می‌توان انجام داد: نشاندن در 1 ، بازنشانی در 0 و متمم شدن خروجی. فلیپ فلاپ JK هر سه کار را انجام می‌دهد. نمودار مدار یک فلیپ فلاپ JK که از یک فلیپ فلاپ D ساخته شده است، در شکل ۱۲-۵ (الف) دیده می‌شود. ورودی J، فلیپ فلاپ را در 1 ، ورودی K ، آن را در 0 می‌نشاند، و وقتی هر دو ورودی در 1 قرار گیرند خروجی متمم می‌شود. صحت این مطلب را می‌توان با بررسی مداری که به ورودی D اعمال شده تحقیق کرد:

$$D = JQ' + K'Q$$

وقتی $J = 1$ و $K = 0$ است، $D = Q' + Q = 1$ بوده و بنابراین لبه ساعت بعدی خروجی را در 1 می‌نشاند. وقتی که $J = 0$ و $K = 1$ باشد، لبه پالس بعدی خروجی را به 0 باز می‌نشاند. وقتی هر دو



(ب) سمبل گرافیکی

(الف) نمودار مدار

شکل ۱۲-۵. فلیپ فلاپ D

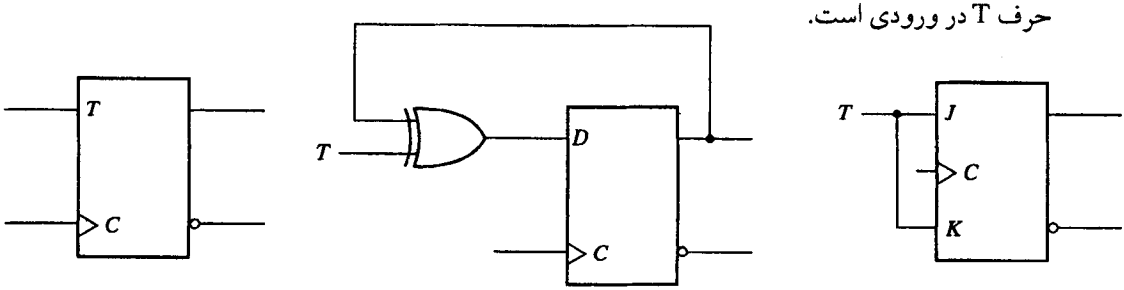
ورودی $J = K = 1$ باشد، $D = Q'$ است و بنابراین لبه ساعت بعدی خروجی را متمم می‌کند. هنگامی که $J = K = 0$ باشد، $D = Q$ است و لبه پالس ساعت بعدی خروجی را بدون تغییر رها خواهد کرد. سمبل گرافیکی برای فلیپ فلاپ JK در شکل ۱۲-۵ (ب) ملاحظه می‌گردد. این سمبل مشابه فلیپ فلاپ D است به جز این که اکنون ورودی‌ها با J و K نام گذاری شده‌اند.

فلیپ فلاپ T (دگر وضع) یک فلیپ فلاپ متمم‌ساز است و می‌توان آن را با گره زدن دو ورودی J و K ایجاد کرد. این عمل در شکل ۱۳-۵ (الف) نشان داده شده است. وقتی $T = 0$ باشد، $(J = K = 0)$، لبه ساعت، خروجی را عوض نمی‌کند. وقتی که $T = 1$ است $(J = K = 1)$، لبه ساعت، خروجی را متمم می‌نماید. فلیپ فلاپ متمم‌ساز در طراحی شمارنده‌های دودویی بسیار مورد توجه است.

یک فلیپ فلاپ T را می‌توان با یک فلیپ فلاپ D و یک گیت XOR مطابق شکل ۱۳-۵ (ب) ساخت. عبارت ورودی D در این حالت برابر است با:

$$D = T \oplus Q = TQ' + T'Q$$

وقتی $T = 0$ است، آنگاه $D = Q$ می‌باشد و بنابراین تغییری در خروجی رخ نمی‌دهد. وقتی $T = 1$ باشد، آنگاه $D = Q'$ بوده و خروجی متمم می‌گردد. سمبل گرافیکی برای این نوع فلیپ فلاپ دارای حرف T در ورودی است.



(پ) سمبل گرافیکی

(ب) با فلیپ فلاپ *D*

(الف) با فلیپ فلاپ *JK*

شکل ۱۳-۵. فلیپ فلاپ D

جدول مشخصه

جدول مشخصه خواص منطقی یک فلیپ فلاپ را تعریف می‌کند و بدین ترتیب عملکرد آن به صورت جدول توصیف می‌گردد. جداول مشخصه سه نوع فلیپ فلاپ در جدول ۱-۵ نشان داده شده است. آنها حالت بعدی را به صورت تابعی از ورودی‌ها و حالت فعلی تعریف می‌نمایند. Q(t) به معنی حالت فعلی و یا حالت قبل از اعمال لبه ساعت است. Q(t+1) ، حالت بعدی پس از اعمال ساعت می‌باشد. توجه کنید ورودی لبه ساعت در جدول مشخصه ذکر نشده است ولی فرض بر این است که بین t و t+1 رخ می‌دهد.

جدول مشخصه فلیپ فلاپ JK نشان می‌دهد که حالت بعدی برابر است با حالت فعلی، به شرطی که J = K = 0 باشد. این وضع را می‌توان به صورت Q(t+1) = Q(t) نشان داد و بیان می‌دارد که تغییری در حالت آن ایجاد نمی‌شود. وقتی که K = 1 و J = 0 باشد، ساعت فلیپ فلاپ را به 0 بازنشانی می‌کند و بنابراین Q(t+1)=0 خواهد شد. اگر J = 1 و K = 0 گردد فلیپ فلاپ به Q(t+1)= 1 می‌رود. وقتی که هر دو ورودی J و K برابر 1 شوند، حالت بعدی متمم حالت فعلی خواهد بود و می‌توان آن را با Q(t+1) = Q′(t) نشان داد.

حالت بعدی فلیپ فلاپ فقط به ورودی D بستگی دارد و مستقل از حالت فعلی است. این حالت را با Q(t+1) = D نشان می‌دهیم. این بدان معنی است که مقدار حالت بعدی برابر با مقدار فعلی (قبل از لبه پالس ساعت) ورودی D است. البته باید توجه کرد که فلیپ فلاپ D حالت بی‌تغییر را دارا نیست. ولی این کار با غیرفعال کردن ساعت و یا با اتصال خروجی به ورودی D انجام می‌شود و طی آن خروجی یا حالت فلیپ فلاپ همواره بی‌تغییر خواهد ماند.

جدول درستی فلیپ فلاپ T فقط دو حالت دارد. وقتی T = 0 باشد، لبه ساعت حالت را تغییر نمی‌دهد. وقتی T = 1 باشد، لبه ساعت حالت فلیپ فلاپ را متمم می‌کند.

جدول ۱-۵ . جداول مشخصه فلیپ فلاپ

فلیپ فلاپ *JK*

J	*K*	*Q*(*t* + 1)	
0	0	*Q*(*t*)	بلاتغییر
0	1	0	بازنشانی
1	0	1	نشاندن
1	1	*Q*′(*t*)	متمم

فلیپ فلاپ *T*

T	*Q*(*t* + 1)	
0	*Q*(*t*)	بلاتغییر
1	*Q*′(*t*)	متمم

فلیپ فلاپ *D*

D	*Q*(*t* + 1)	
0	0	بازنشانی
1	1	نشاندن

معادلات مشخصه

خواص منطقی یک فلیپ فلاپ که در جدول مشخصه ملاحظه شد را می‌توان به صورت معادله مشخصه هم بیان کرد. برای فلیپ فلاپ D، این معادله به صورت زیر است:

$$Q(t + 1) = D$$

این رابطه بیان می‌کند که حالت بعدی خروجی برابر با مقدار ورودی D در حال حاضر است. معادله مشخصه برای فلیپ فلاپ JK را از جدول مشخصه و یا از مدار شکل ۱۲-۵ می‌توان بدست آورد. یعنی

$$Q(t + 1) = JQ' + K'Q$$

که Q مقدار خروجی فلیپ فلاپ قبل از اعمال یک پالس ساعت است. معادله مشخصه برای فلیپ فلاپ T از شکل ۱۳-۵ حاصل می‌شود.

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

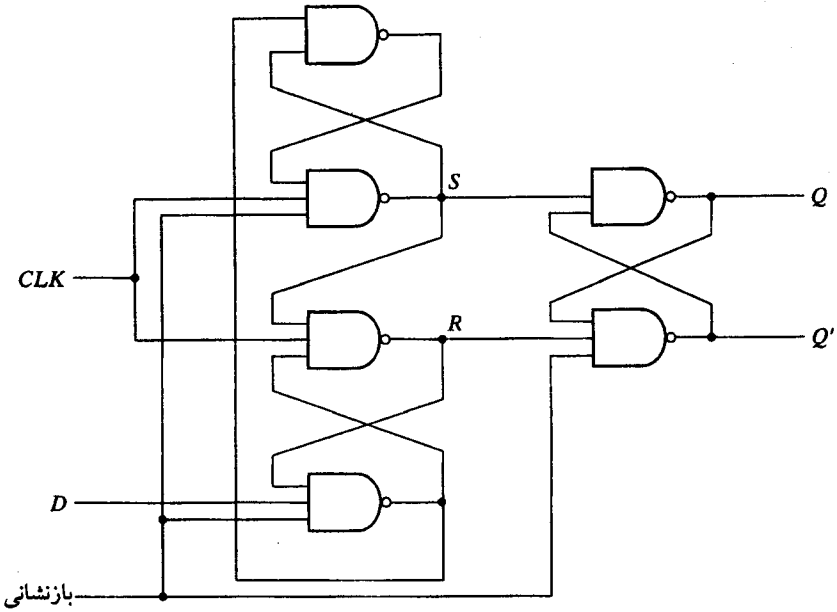
ورودی‌های سیستم

بعضی از فلیپ فلاپ‌ها دارای ورودی‌های غیرهمزمان برای واداشتن آن به یک حالت خاص مستقل از پالس ساعت می‌باشند. ورودی که فلیپ فلاپ را در 1 می‌نشاند، پیش تنظیم یا تنظیم مستقیم می‌نامند. ورودی که فلیپ فلاپ را به 0 پاک می‌کند، ورودی پاک یا بازنشان مستقیم (غیر همزمان) می‌خوانند. وقتی تغذیه در یک سیستم دیجیتال روشن شود، حالت فلیپ فلاپ نامعلوم است. ورودی‌های مستقیم در استقرار همه فلیپ فلاپ‌های سیستم به یک حالت آغازین معلوم، قبل از اعمال پالس ساعت مفید هستند.

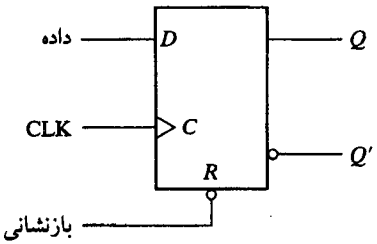
یک فلیپ فلاپ D حساس به لبه مثبت با بازنشانی غیرهمزمان R در شکل ۱۴-۵ ملاحظه می‌شود. نمودار مدار مشابه شکل ۱۰-۵ است با این تفاوت که یک ورودی بازنشانی اضافی، به سه گیت NAND متصل شده است. وقتی که این ورودی در 0 است، Q' را به ماندن در 1 وا می‌دارد، و این به نوبه خود به معنی پاک شدن خروجی Q به 0 است و بنابراین فلیپ فلاپ بازنشانی می‌شود. دو اتصال دیگر از ورودی بازنشان بقاء سومین لچ SR را در منطق 1 تضمین می‌کند. این وضع هنگامی رخ می‌دهد که ورودی بازنشانی، بدون توجه به مقادیر D و CLK، در 0 باشد.

سمبل گرافیکی فلیپ فلاپ D با یک ورودی بازنشان مستقیم دارای یک ورودی اضافی است که با R علامت‌گذاری شده است. وجود حباب در ورودی به این معنی است که بازنشانی با سطح منطق 0 فعال می‌گردد. فلیپ فلاپ‌هایی که از نشاندن مستقیم استفاده می‌کنند از سمبل S در ورودی نشاندن غیرهمزمان استفاده می‌کنند.

جدول تابع، عملکرد مدار را مشخص می‌کند. وقتی $R = 0$ باشد، خروجی به 0 بازنشانی می‌شود. این حالت مستقل از D و C است. هنگامی مدار می‌تواند به روند عادی خود بازگردد که ورودی بازنشانی به 1 برود. ساعت در C با یک فلش رو به بالا، که به معنی عملکرد فلیپ فلاپ در لبه مثبت ساعت می‌باشد، نشان داده شده است. مقدار D با هر لبه مثبت سیگنال ساعت، به شرطی که $R = 1$ باشد، به خروجی Q منتقل می‌گردد.



(الف) نمودارمدار



(ب) سمبل گرافیکی

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(پ) جدول درستی

شکل ۱۴-۵. فلیپ فلاپ D با بازنشانی غیرهمزمان

۵-۴ تحلیل مدارهای ترتیبی ساعت‌دار

رفتار یک مدار ترتیبی ساعت‌دار با ورودی‌ها، خروجی‌ها و حالت فلیپ فلاپ‌ها مشخص می‌گردد. خروجی‌ها و حالت بعدی هر دو تابعی از ورودی‌ها و حالت فعلی‌اند. تحلیل یک مدار ترتیبی به معنی تهیه جدول یا نموداری از رشته زمانی ورودی‌ها، خروجی‌ها و حالات درونی است. می‌توان عبارات بول را نوشت و به وسیله آنها رفتار مدار را توصیف کرد. این عبارات باید رشته زمانی لازم را چه مستقیماً و چه غیرمستقیم مشخص کند.

یک نمودار منطقی، وقتی دارای فلیپ فلاپ‌ها با ورودی‌های ساعت باشد، مدار ترتیبی ساعت‌دار

خوانده می‌شود. فلیپ فلاپ‌ها می‌توانند از هر نوع، و نمودار منطقی هم ممکن است شامل گیت‌های ترکیبی باشد یا نباشد. در این بخش، ما یک نمایش جبری را برای تعیین حالت بعدی برحسب حالت فعلی و ورودی‌ها ارائه می‌کنیم. آنگاه برای توصیف رفتار یک مدار ترتیبی، یک جدول حالت و یک نمودار حالت ارائه می‌شود. یک عبارت جبری دیگر هم برای مشخص کردن نمودار منطقی مدارهای ترتیبی بیان می‌گردد. برای تشریح روال‌های مختلف، مثال‌های خاصی آورده شده است.

معادلات حالت

رفتار مدار ترتیبی ساعت‌دار را می‌توان با معادلات حالت توصیف کرد. یک معادله حالت (که به آن معادله گذر هم می‌گویند) حالت بعدی را برحسب تابعی از حالات فعلی و ورودی‌ها بیان می‌نماید. مدار شکل ۱۵-۵ را ملاحظه نمایید. این مدار از دو فلیپ فلاپ A و B نوع D و یک ورودی x و یک خروجی y تشکیل شده است. چون ورودی D یک فلیپ فلاپ، مقدار حالت بعدی را معین می‌کند، می‌توان مجموعه معادلاتی را به صورت زیر برای مدار نوشت:

$$A(t + 1) = A(t)x(t) + B(t)x(t)$$
$$B(t + 1) = A'(t)x(t)$$

یک معادله حالت معادله‌ای است که شرایط گذر حالت را برای یک فلیپ فلاپ بیان می‌کند. سمت چپ معادله با (t+1) حالت بعدی فلیپ فلاپ را پس از یک لبه ساعت معین می‌نماید. سمت راست معادله عبارتی است بولی که حالت فعلی و وضعیت ورودی‌هایی را مشخص می‌نمایند که در قبال آنها حالت بعدی 1 می‌گردد. چون همه متغیرها در عبارت بول تابعی از حالت فعلی هستند، ما از نوشتن (t) پس از متغیر صرف‌نظر کرده و معادلات حالت را به صورت فشرده‌تری مطابق زیر می‌نویسیم:

$$A(t + 1) = Ax + Bx$$
$$B(t + 1) = A'x$$

عبارات بولی برای معادلات حالت مستقیماً از گیت‌های تشکیل‌دهنده بخش ترکیبی در مدار ترتیبی بدست می‌آیند، زیرا مقادیر D در مدار ترکیبی حالت بعدی را تعیین می‌کنند. به طور مشابه مقدار فعلی خروجی نیز قابل ارائه به صورت جبری زیر است:

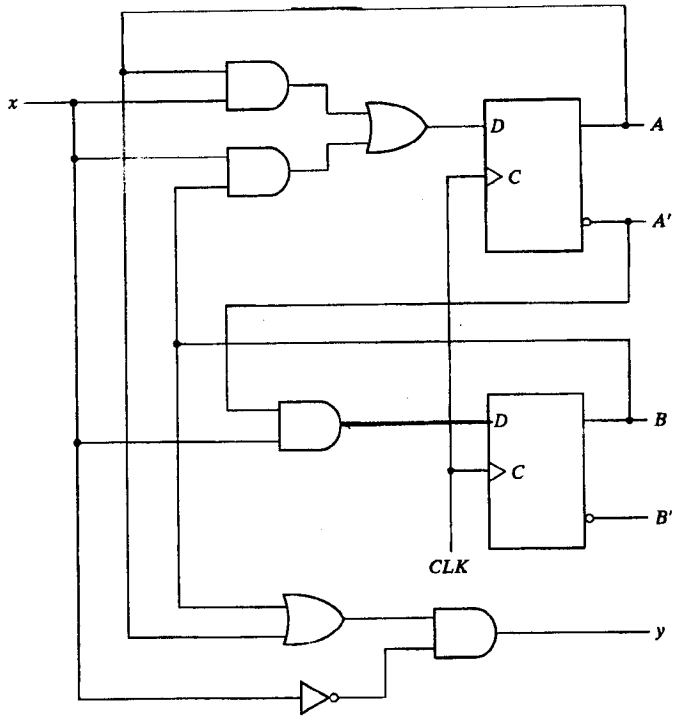
$$y(t) = [A(t) + B(t)]x'(t)$$

با حذف سمبل (t) از مقدار فعلی، معادله بولی خروجی زیر بدست می‌آید:

$$y = (A + B)x'$$

جدول حالت

رشته‌های زمانی ورودی‌ها، و خروجی‌ها و حالات فلیپ فلاپ را می‌توان در یک جدول حالت (به آن جدول گذر هم می‌گویند) جمع‌آوری کرد. جدول حالت برای مدار شکل ۱۵-۵ در جدول ۲-۵ دیده می‌شود. جدول متشکل از چهار بخش با نام‌های حالت فعلی، ورودی، حالت بعدی و خروجی است.



شکل ۱۵-۵. مثال مدار ترتیبی

بخش حالت فعلی، حالت فلیپ فلاپ‌های A و B را در هر لحظه از زمان t نشان می‌دهد. بخش ورودی مقدار x را برای هر حالت فعلی ممکن بدست می‌دهد. بخش حالت بعدی، وضعیت فلیپ فلاپ‌ها را یک سیکل ساعت بعد، در زمان t+1 بیان می‌دارد. بخش خروجی مقدار y را در هر زمان t در قبال هر حالت فعلی با توجه به شرایط ورودی، مشخص می‌کند.

تهیه جدول حالت به لیستی از همه ترکیبات دودویی حالت فعلی و ورودی‌ها نیاز دارد. در این حال،

جدول ۲-۵. جدول حالت برای شکل ۱۵-۵

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

ما هشت ترکیب دودویی 000 تا 111 را داریم. سپس مقادیر حالت بعدی از نمودار منطقی یا از معادلات حالت بدست می‌آیند. حالت بعدی فلیپ فلاپ A باید در معادله زیر صدق کند.

$$A(t + 1) = Ax + Bx$$

بخش حالت بعدی در جدول حالت در زیر ستون A دارای سه عدد 1 است که در قبال آنها حالت فعلی و مقدار ورودی شرایطی را که حالت فعلی A و ورودی x هر دو برابر 1 هستند و یا حالت فعلی B و ورودی x هر دو برابر 1 می‌باشند برآورده می‌سازند. به طور مشابه حالت بعدی فلیپ فلاپ B از معادله حالت زیر حاصل می‌گردد.

$$B(t + 1) = A'x$$

و هنگامی برابر 1 است که حالت فعلی $A = 0$ و ورودی $x = 1$ باشد. ستون خروجی از معادله زیر حاصل می‌شود:

$$y = Ax' + Bx'$$

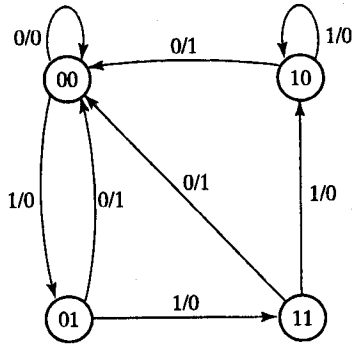
جدول حالت یک مدار ترتیبی با فلیپ فلاپ‌های نوع D با روال مشابهی بدست می‌آید. به طور کلی، یک مدار ترتیبی با m فلیپ فلاپ و n ورودی نیاز به 2^{m+n} سطر در جدول حالت دارد. اعداد دودویی از 0 تا $2^{m+n}-1$ در زیر ستون‌های حالت فعلی و ورودی لیست شده‌اند. بخش حالت بعدی دارای m ستون، یعنی یک ستون در ازاء هر فلیپ فلاپ، می‌باشد. مقادیر دودویی برای حالت بعدی مستقیماً از معادلات حالت حاصل می‌گردند. بخش خروجی دارای ستونهایی به تعداد خروجی‌هاست. مقدار دودویی این بخش مشابه با جدول درستی از مدار یا تابع بولی آن بدست می‌آید. گاهی بهتر است تا جدول حالت را با کمی تغییر نشان دهیم. در آرایشی دیگر، جدول حالت تنها سه بخش دارد که عبارتند از: حالت فعلی، حالت بعدی و خروجی. حالات ورودی در زیر ستون حالت بعدی و ستون خروجی ذکر می‌شود. جدول حالت ۲-۵ با توجه به این روش به جدول ۳-۵ تبدیل شده است. برای هر حالت فعلی، بسته به مقدار ورودی، دو حالت ممکن برای حالت بعدی و خروجی وجود دارد. بسته به نوع کاربرد هر یک از دو روش فوق بر دیگری ارجحیت دارد.

نمودار حالت

اطلاعات موجود در جدول حالت را می‌توان به صورت گرافیکی با نمودار حالت نشان داد. در این

جدول ۳-۵ . فرم دیگر جدول حالت

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0



شکل ۱۶-۵. نمودار حالت مدار شکل ۱۵-۵

نوع نمودار، یک حالت با یک دایره نشان داده می‌شود و گذر در بین حالات با خطوط جهت‌داری که دو دایره را به هم وصل می‌کنند نمایش داده می‌شود. نمودار حالت مدار ترتیبی شکل ۱۵-۵ در شکل ۱۶-۵ ملاحظه می‌گردد. نمودار حالت همان اطلاعات جدول حالت را بیان می‌کند که مستقیماً از جدول ۲-۵ یا ۳-۵ بدست می‌آید. عدد دودویی داخل هر دایره حالت فلیپ فلاپ‌ها را بیان می‌نماید. خطوط جهت‌دار با دو عدد که با یک خط مورب از هم جدا شده‌اند، برچسب خورده‌اند. مقدار ورودی در حالت فعلی در سمت چپ این خط و عدد پس از خط مورب، خروجی را در حالت فعلی در قبال ورودی مربوطه‌اش نشان می‌دهد. باید توجه داشت مقدار خروجی ذکر شده در کنار خطوط جهت‌دار، در حین حالت فعلی و ورودی مربوطه رخ می‌دهد و هیچ ارتباطی به حالت بعدی ندارد. مثلاً خط واصل جهت‌دار که از حالت 00 به 01 می‌رود با 1/0 برچسب خورده است و به این معنی است که وقتی مدار ترتیبی در حالت فعلی 00 است ورودی 1 و خروجی 0 می‌باشد، در پالس ساعت بعدی، مدار به حالت بعدی 01 می‌رود. اگر ورودی به 0 تغییر یابد، آنگاه خروجی 1 می‌گردد ولی اگر ورودی در 1 باقی بماند، خروجی در 0 خواهد ماند. این اطلاعات از نمودار حالت و خطوط جهت‌دار که از دایره 01 سرچشمه گرفته، حاصل شده است. یک خط جهت‌دار که دایره را به خودش وصل کند، به معنی عدم وجود تغییر در حالت است.

بین جدول حالت و نمودار حالت تفاوتی به جز نحوه ارائه وجود ندارد. جدول درستی به راحتی از یک معادله حالت و نمودار منطقی حاصل می‌گردد. نمودار حالت مستقیماً از جدول حالت بدست می‌آید. نمودار حالت تصویری از گذر حالات را مجسم می‌کند و برای تفسیر عملکرد مدار مناسب‌تر است. مثلاً، نمودار حالت شکل ۱۶-۵ به وضوح نشان می‌دهد که، با شروع از حالت 00 ، مادامی که ورودی در 1 باشد خروجی برابر 0 است. اولین ورودی 0 بعد از رشته‌ای از 1 ، خروجی 1 را تولید کرده و مدار را به 00 اولیه باز می‌گرداند.

معادلات ورودی فلیپ فلاپ

نمودار منطقی یک مدار ترتیبی متشکل از فلیپ فلاپ‌ها و گیت‌هاست. اتصالات میان گیت‌ها

مدار ترکیبی را می‌سازند و ممکن است با عبارات بولی نشان داده شوند. آگاهی از نوع فلیپ فلاپ‌ها و لیست عبارات بولی مدار ترکیبی، اطلاعات لازم را برای ترسیم نمودار منطقی مدار ترتیبی فراهم می‌سازد. بخشی از مدار ترکیبی که خروجی‌های بیرونی را تولید می‌کند و به صورت توابع بولی توصیف می‌گردند معادلات خروجی نامیده می‌شوند. بخشی از مدار که ورودی‌های فلیپ فلاپ‌ها را تولید می‌کنند با توابع بولی به نام معادلات ورودی فلیپ فلاپ نام‌گذاری شده‌اند (گاهی به آنها معادلات تحریک هم می‌گویند). ما از سمبل ورودی فلیپ فلاپ برای نام‌گذاری متغیر معادله ورودی و نام خروجی فلیپ فلاپ‌ها به عنوان اندیس استفاده خواهیم کرد. مثلاً معادله ورودی زیر یک گیت OR، با ورودی‌های x و y که به ورودی D از فلیپ فلاپ متصلند و خروجی آن با Q نام‌گذاری شده است را نشان می‌دهد.

$$D_Q = x + y$$

مدار ترتیبی شکل ۱۵-۵ متشکل از دو فلیپ فلاپ A و B از نوع D، یک ورودی x و یک خروجی y است. نمودار منطقی مدار می‌تواند به صورت جبری با دو معادله ورودی و یک معادله خروجی بیان شود:

$$D_A = Ax + Bx$$
$$D_B = A'x$$
$$y = (A + B)x'$$

سه معادله فوق اطلاعات لازم را برای ترسیم نمودار منطقی مدار ترتیبی فراهم می‌سازند. سمبل D_A یک فلیپ فلاپ D با نام A را مشخص می‌نماید. به همین ترتیب D_B فلیپ فلاپ B از نوع D است. عبارات بولی مربوط به این دو متغیر و عبارت خروجی y، بخش ترکیبی مدار ترتیبی را معین می‌کنند.

معادلات ورودی فلیپ فلاپ‌ها فرم جبری مناسبی را برای نمودار منطقی یک مدار ترتیبی تشکیل می‌دهند. آنها نوع فلیپ فلاپ را با توجه به سمبل فلیپ فلاپ مشخص می‌نمایند و مدار ترکیبی که فلیپ فلاپ‌ها را راه می‌اندازند هم با آنها مشخص می‌شود. توجه کنید که عبارت معادله ورودی با عبارت مربوط به معادله حالت یکی است. دلیل این است که معادله مشخصه با مقدار ورودی به D برابر است: یعنی $Q(t+1) = D_Q$.

تحلیل با کمک فلیپ فلاپ‌های D

در اینجا روال تحلیل یک مدار ترتیبی متشکل از فلیپ فلاپ‌های D را با یک مثال ساده خلاصه می‌کنیم. مداری که برای این هدف در نظر گرفته شده با معادله ورودی زیر توصیف گردیده است.

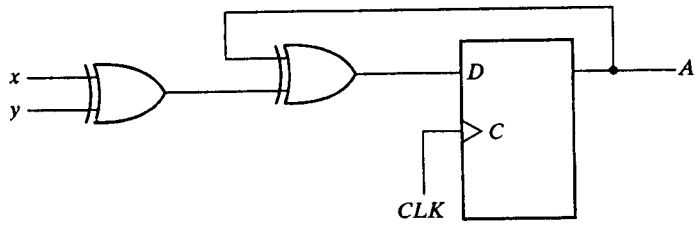
$$D_A = A \oplus x \oplus y$$

سمبل D_A یک فلیپ فلاپ نوع D با خروجی A را بیان می‌کند. متغیرهای x و y، ورودی‌ها به مدار هستند. هیچ معادله خروجی مشخص نشده، بنابراین خروجی مدار از خروجی فلیپ فلاپ اخذ شده است. نمودار منطقی از معادله ورودی حاصل و در شکل ۱۷-۵ (الف) رسم شده است.

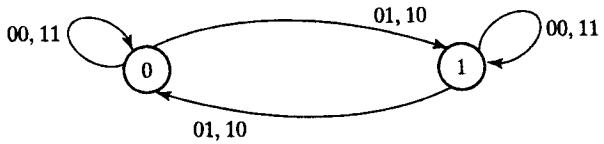
جدول حالت برای حالت فعلی یک ستون داشته و متعلق به فلیپ فلاپ A است، دو ستون هم برای ورودی‌ها و یک ستون برای حالت بعدی A لازم است. اعداد دودویی زیر ستون Axy از 000 تا 111

حالت فعلی	ورودی‌ها		حالت بعدی
A	x	y	A
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(ب) جدول حالت



(الف) نمودار مدار



(پ) نمودار حالت

شکل ۱۷-۵. مدار ترتیبی با فلیپ فلاپ D

مطابق شکل ۱۷-۵ (ب) لیست شده‌اند. مقادیر حالت بعد، از معادله حالت زیر حاصل می‌شوند:

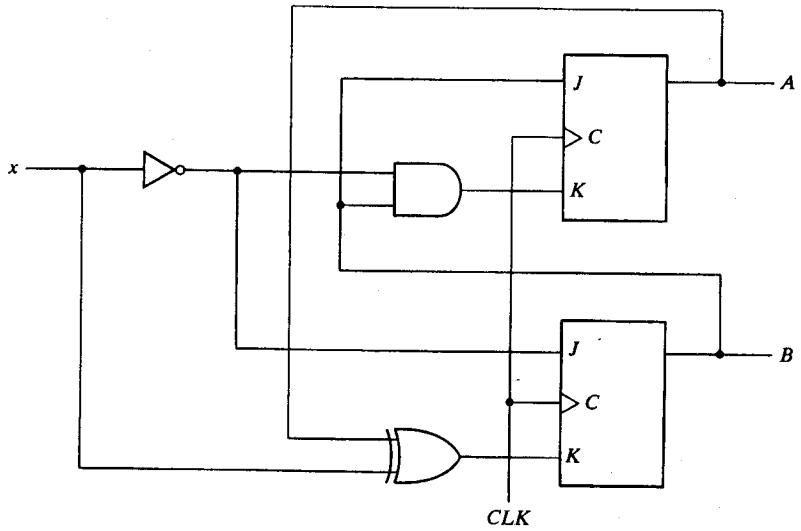
$$A(t + 1) = A \oplus x \oplus y$$

این عبارت یک تابع فرد را بیان می‌دارد و هنگامی برابر 1 است که فقط یک یا سه متغیر برابر 1 باشد. این نکته در ستون حالت بعدی A قابل ملاحظه است.

مدار دارای یک فلیپ فلاپ و دو حالت است. نمودار حالت از دو دایره که هر یک مطابق شکل ۱۷-۵ (پ) متعلق به یک حالت می‌باشد تشکیل گردیده است. حالت فعلی و خروجی، همانطور که با اعداد داخل دوایر نشان داده شده، می‌تواند 0 یا 1 باشد. روی خطوط جهت‌دار به خطوط مورب نیازی نیست زیرا برای مدار ترکیبی هیچ خروجی در نظر گرفته نشده است. دو ورودی، چهار ترکیب ممکن را برای هر حالت ممکن می‌سازند. دو ترکیب ورودی برای هر گذر حالت با یک ویرگول از هم جدا شده‌اند تا شکل مفهوم‌تر باشد.

تحلیل فلیپ فلاپ‌های JK

یک جدول حالت متشکل از چهار بخش، حالت فعلی، ورودی‌ها، حالت بعدی و خروجی‌هاست. دو مورد اول با لیست حاصل از همه ترکیبات بدست می‌آیند. بخش خروجی از معادلات خروجی حاصل می‌شوند. مقادیر حالت بعدی از معادلات حالت ارزیابی می‌گردند. در فلیپ فلاپ نوع D معادله



شکل ۱۸-۵. مدار ترتیبی با فلیپ فلاپ JK

حالت با معادله ورودی یکی است. هنگامی که فلیپ فلاپ‌هایی به جز D مثل JK یا T به کار روند، لازم است به جدول مشخصه یا معادله مشخصه آنها مراجعه شود تا مقادیر حالت بعدی بدست آیند. ما رویه را ابتدا با به‌کارگیری جدول مشخصه و سپس با معادله مشخصه تشریح خواهیم کرد.

مقادیر حالت بعدی یک مدار ترتیبی که از فلیپ فلاپ‌هایی چون نوع JK و T استفاده می‌کنند از رویه زیر بدست می‌آید.

۱- تعیین معادلات ورودی برحسب حالت فعلی و متغیرهای ورودی

۲- لیست مقادیر دودویی هر معادله ورودی

۳- استفاده از جدول مشخصه فلیپ فلاپ برای تعیین مقادیر حالت در جدول حالت.

به عنوان یک مثال، مدار ترتیبی متشکل از دو فلیپ فلاپ A و B از نوع JK و یک ورودی x را طبق شکل ۱۸-۵ ملاحظه نمایید. مدار دارای خروجی خاص نیست و بنابراین نیازی به ستون خروجی در جدول حالت وجود ندارد (خروجی‌های فلیپ فلاپ را می‌توان در این حالت به عنوان خروجی در نظر گرفت). می‌توان مدار را با معادلات ورودی زیر بیان کرد.

$$J_A = B \qquad K_A = Bx'$$
$$J_B = x' \qquad K_B = A'x + Ax' = A \oplus x$$

جدول حالت مدار ترتیبی در جدول ۴-۵ نشان داده شده است. ستون‌های حالت فعلی و ورودی، هشت حالت ممکن را لیست کرده‌اند. مقادیر دودویی زیرستون‌های "ورودی فلیپ فلاپ‌ها" بخشی از جدول حالت نیستند، ولی برای ارزیابی حالت بعدی که در مرحله ۲ از رویه ذکر شد لازم‌اند.

جدول ۴-۵. جدول حالت برای مدار ترتیبی با فلیپ فلاپ JK

حالت فعلی		ورودی	حالت بعدی		ورودی های فیلیپ فلاپ			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

این مقادیر دودویی مستقیماً از چهار معادله ورودی مشابه با آنچه برای جدول درستی یک عبارت بول حاصل می‌شوند، بدست آمده‌اند. حالت بعدی هر فلیپ فلاپ از ورودی‌های J و K و جدول مشخصه فلیپ فلاپ JK در جدول ۱-۵ حاصل می‌گردند. چهار حالت برای بررسی وجود دارد. وقتی J = 1 و K = 0 باشد، حالت بعدی 1 است. وقتی J = 0 و K = 1 است، حالت بعدی 0 می‌باشد. با J = K=0، تغییری در حالت وجود ندارد و حالت بعدی با حالت فعلی یکی است. وقتی J = K = 1 باشد. بیت حالت بعدی متمم بیت حالت فعلی است. مثال‌های دو حالت فوق‌الذکر در جدول هنگام AB=10 و x=0 رخ می‌دهند. بنابراین $J_A = K_A = 0$ بوده و حالت فعلی A = 1 است. به این ترتیب حالت بعدی A با حالت فعلی تفاوتی نداشته و مقدار آن 1 است. در همان سطر از جدول $J_B = K_B = 1$ است. چون حالت فعلی B = 0 می‌باشد، حالت بعدی B متمم شده و به 1 تغییر می‌یابد.

می‌توان مقادیر حالت بعدی را از ارزیابی معادلات حالت در معادله مشخصه هم بدست آورد. این کار با دنبال کردن روال زیر میسر است.

۱- معادلات ورودی فلیپ فلاپ را برحسب حالت فعلی و متغیرهای ورودی بدست آورید.

۲- معادلات ورودی را در معادلات مشخصه فلیپ فلاپ جایگزین نمایید تا معادلات حالت حاصل شود.

۳- از معادلات حالت برای تعیین مقادیر حالت بعدی در جدول حالت استفاده نمایید.

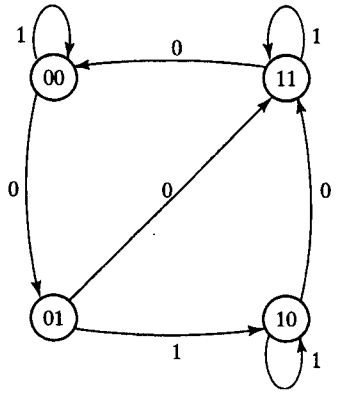
معادلات ورودی فلیپ فلاپ JK شکل ۱۸-۵ در فوق ملاحظه گردید. معادلات مشخصه برای فلیپ فلاپ‌ها از جایگزینی A و B به جای اسم Q بدست می‌آیند:

$$A(t + 1) = JA' + K'A$$
$$B(t + 1) = JB' + K'B$$

از جایگزینی J_A و K_A از معادلات ورودی، معادله حالت برای A بدست می‌آید:

$$A(t + 1) = BA' + (Bx')'A = A'B + AB' + Ax$$

معادله حالت مقادیر بیتی ستون زیر "حالت بعدی" A را در جدول حالت فراهم می‌سازد. به طور مشابه،



شکل ۱۹-۵. نمودار حالت شکل ۱۸-۵

معادله حالت برای فلیپ فلاپ B با جایگزینی مقادیر J_B و K_B بدست می‌آید.

$$B(t+1) = x'B' + (A \oplus x)'B = B'x' + ABx + A'Bx'$$

معادله حالت مقادیر بیتی را برای ستون زیر "حالت بعدی" B در جدول حالت فراهم می‌نماید. توجه کنید که وقتی معادله حالت به کار رود ستون‌های زیر ورودی‌های "فلیپ فلاپ" در جدول ۴-۵ لازم نیستند.

نمودار حالت مدار ترتیبی در شکل ۱۹-۵ نشان داده شده است. چون مدار دارای خروجی نیست اعداد روی خطوط جهت‌دار خارج شده از دوایر، تنها مقادیر ورودی x را بیانگر هستند.

تحلیل با فلیپ فلاپ‌های T

تحلیل یک مدار ترتیبی با فلیپ فلاپ‌های T روال یکسانی با نوع JK دارد. مقادیر حالت بعدی در جدول حالت با جدول مشخصه ۱-۵ یا با معادله مشخصه زیر بدست می‌آیند.

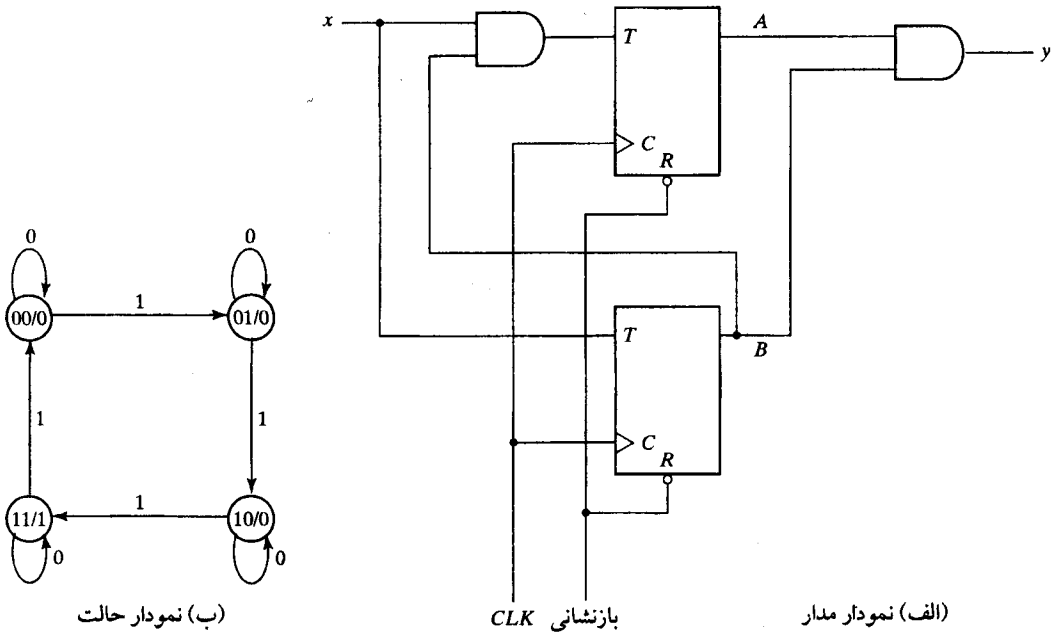
$$Q(t+1) = T \oplus Q = T'Q + TQ'$$

مدار ترتیبی شکل ۲۰-۵ را ملاحظه نمایید. این مدار دارای دو فلیپ فلاپ A و B، یک ورودی x و یک خروجی y است. این مدار با دو معادله ورودی و یک معادله خروجی قابل توصیف می‌باشد.

$$T_A = Bx$$
$$T_B = x$$
$$y = AB$$

جدول حالت برای مدار در جدول ۵-۵ لیست شده است. مقادیر y از معادله خروجی بدست می‌آیند. مقادیر حالت بعدی از معادلات حالت و با جایگزینی T_A و T_B در معادلات مشخصه حاصل می‌شوند، یعنی:

$$A(t+1) = (Bx)'A + (Bx)A' = AB' + Ax' + A'Bx$$
$$B(t+1) = x \oplus B$$



شکل ۲۰-۵ . مدار ترتیبی با فلیپ فلاپ‌های T

مقادیر حالت بعدی در جدول حالت از عبارات مربوط به دو معادله حالت بدست می‌آید.

نمودار حالت مدار در شکل ۲۰-۵ (ب) ملاحظه می‌شود. مادامی که ورودی x برابر 1 است، مدار به عنوان یک شمارنده دودویی با رشته 00، 01، 10 و 11 عمل می‌کند و در نهایت به 00 باز می‌گردد. وقتی $x = 0$ است، مدار در همان حال باقی می‌ماند. در حالت 11، خروجی $y = 1$ است. در اینجا خروجی فقط به حالت فعلی وابسته بوده و مستقل از ورودی است. دو مقدار داخل هر دایره با یک خط مورب از هم جدا شده‌اند تا حالت فعلی و خروجی از هم تفکیک شوند.

جدول ۵-۵ . جدول حالت برای مدار ترتیبی با فلیپ فلاپ‌های T

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

مدل‌های میلی و مور

عمومی‌ترین مدل یک مدار ترتیبی دارای ورودی‌ها، خروجی‌ها و حالات داخلی است. معمولاً مدارهای ترتیبی به دو صورت مدل مور و میلی ارائه می‌شوند. تفاوت آنها در نحوه تولید خروجی است. در مدل میلی، خروجی تابعی از ورودی و حالت فعلی است. در مدل مور، خروجی فقط تابعی از حالت فعلی می‌باشد. وقتی این دو مدل در کتب و دیگر منابع تکنیکی مورد بحث قرار می‌گیرند، مدارهای ترتیبی را ماشین حالت متناهی یا FSM گویند. مدل میلی یک مدار ترتیبی را FSM میلی یا ماشین میلی نامند. مدل مور را نیز FSM مور یا ماشین مور خوانند.

مثالی از مدل میلی در شکل ۱۵-۵ نشان داده شد. خروجی y تابعی از هر دو ورودی x و حالت فعلی A و B است. نمودار حالت مربوطه در شکل ۱۶-۵ نشان می‌دهد که مقادیر ورودی‌ها و خروجی‌ها با خط موربی در امتداد خطوط جهت‌دار بین حالت‌ها از یکدیگر جدا شده‌اند.

مثالی از مدل مور در شکل ۱۸-۵ نشان داده شد. در اینجا خروجی فقط تابعی از حالت فعلی است. نمودار حالت مربوطه در شکل ۱۹-۵ تنها دارای ورودی‌هایی است که در امتداد خطوط جهت‌دار ذکر شده‌اند. خروجی‌ها همان حالات فلیپ فلاپ‌ها هستند که در داخل دوایر نشان داده شده‌اند. مثال دیگری از یک مدل مور در شکل ۲۰-۵ ملاحظه می‌شود. خروجی فقط به مقادیر موجود در فلیپ فلاپ‌ها وابسته است و بنابراین تنها تابعی از حالت فعلی است. مقدار ورودی در نمودار حالت در امتداد خط جهت‌دار ذکر شده، ولی خروجی‌های مدار در داخل دایره همراه با حالت فعلی نشان داده شده‌اند.

در مدل مور، خروجی‌های مدار ترتیبی با ساعت همزمان است زیرا آنها تنها به خروجی‌های فلیپ فلاپ بستگی دارند که به پالس ساعت وابسته‌اند. در مدل میلی، خروجی‌ها فقط هنگامی که ورودی‌ها در طول پالس ساعت تغییر کنند، عوض می‌شوند. به علاوه، خروجی‌ها ممکن است مقادیر غلط لحظه‌ای نیز داشته باشند زیرا بین لحظه ورود داده و تغییر خروجی‌ها تأخیر وجود دارد. برای همزمان کردن مدار نوع میلی، ورودی‌های مدار ترتیبی باید با پالس ساعت همزمان گردند و خروجی‌ها فقط باید در لبه پالس نمونه‌برداری شوند.

۵-۵ HDL برای مدارهای ترتیبی

زبان توصیف سخت‌افزاری Verilog HDL در بخش ۹-۳ معرفی شد. توصیف مدارهای ترکیبی و معرفی مدل‌سازی رفتاری در بخش ۱۱-۴ ارائه گردید. در این بخش بحث مدل‌سازی رفتاری را ادامه می‌دهیم و مثال‌هایی از فلیپ فلاپ‌ها و مدارهای ترتیبی را ارائه خواهیم داد.

مدل‌سازی رفتاری

در Verilog HDL دو نوع عبارت رفتاری موجود است: initial و always. رفتار initial تنها یک بار و در $t = 0$ اجرا می‌شود. رفتار always مرتباً اجرا شده و تا پایان شبیه‌سازی تکرار می‌گردد. روند رفتاری با استفاده از کلمه کلیدی initial یا always و به دنبال آن عبارت یا بلوکی از عبارت در بین

کلمات کلیدی begin و end آغاز می‌شود. هر مدول می‌تواند حاوی تعداد دلخواهی عبارت initial و یا always باشد. این عبارات به صورت همروند نسبت به یکدیگر از زمان 0 اجرا می‌گردند.

عبارت initial فقط یک بار اجرا می‌شود. این عبارت در شروع شبیه‌سازی آغاز و پس از تکمیل اجرای همه عبارات پایان می‌پذیرد. همانطور که در انتهای بخش ۱۱-۴ مطرح شد، عبارت initial برای تولید سیگنال‌های ورودی در شبیه‌سازی طرح مفید است. در هنگام شبیه‌سازی یک مدار ترتیبی، لازم است یک منبع ساعت برای تریگر کردن فلیپ فلاپ‌ها تولید شود. در زیر دو راه تولید ساعت آزاد گرد (آزادکار) نشان داده شده است:

```
initial
   begin
      clock = 1'b0 ;
      repeat (30)
      #10 clock = ~ clock;
   end
```

```
initial
    begin
      clock = 1'b0;
      #300  $finish;
    end
always
      #10   clock = ~clock;
```

در نوع اول، بلوک initial بین کلمات کلیدی begin و end قرار دارد. ساعت در زمان 0 در 0 تنظیم می‌شود. ساعت هر 10 واحد زمانی یک بار متمم شده و 30 مرتبه تکرار می‌گردد. این کار 15 سیکل ساعت را با سیکل 20 واحد زمانی تولید می‌نماید. در نوع دوم بلوک initial ساعت را در زمان 0 در 0 قرار می‌دهد. پس از 10 واحد زمانی، عبارت always مرتباً ساعت را هر 10 واحد زمانی یک بار متمم می‌کند و در نتیجه زمان سیکل 20 واحد زمانی خواهد بود. شبیه‌سازی در پاسخ به $finish در زمان 300 پایان می‌پذیرد.

عبارت always را می‌توان با تأخیرهایی که برای مدت معین رخ می‌دهند و یا صحت شرایط معین و رخداد وقایع کنترل کرد. در اینجا شرط براساس رخداد واقعه مورد بحث قرار گرفته است. نوع عبارت به شکل زیر است:

```
always  @ (event control expression)
Procedural assignment statements.
```

عبارت کنترل واقعه، شرطی را که باید رخ دهد تا اجرای عبارات تخصیص اجرایی فعال شود، مشخص می‌کند. متغیرها در سمت چپ عبارات اجرایی باید از نوع داده reg و به همان ترتیب نیز باید اعلان شود. سمت راست می‌تواند هر عبارتی باشد که با عملگرهای تعریف شده در Verilog تولید مقدار بنماید.

عبارت کنترل رخداد واقعه که به آن لیست حساسیت هم می‌گویند، وقایعی را مشخص می‌کند که باید رخ دهد تا اجرای عبارات اجرایی واقع در بلوک always آغاز گردد. عبارات داخل بلوک به ترتیب اجرا می‌شوند و پس از آخرین اجرا، عملیات متوقف می‌گردد. آنگاه عبارت always منتظر رخداد واقعه یا پدیده دیگری می‌گردد. در اینجا دو نوع رخداد واقعه وجود دارد: وقایع حساس به سطح و وقایع حساس به لبه. وقایع حساس به سطح در مدارهای ترکیبی و لچ‌ها رخ می‌دهند. مثلاً عبارت

```
always @ (A or B or Reset)
```

موجب اجرای عبارات اجرایی در بلوک always می‌گردد به شرطی که تغییری در A یا B یا ورودی

بازنشان (Reset) رخ دهد. در مدارهای ترتیبی همزمان، تغییرات باید فقط در پاسخ به یک گذر پالس ساعت رخ دهد. گذر می‌تواند در یک لبه مثبت یا لبه منفی اتفاق بیفتد. Verilog HDL با دو کلمه کلیدی negedge و posedege مراقب این شرایط می‌باشد. مثلاً عبارت

```
always @(posedge clock or negedge reset)
```

فقط عبارات اجرایی را در قبال یک گذر مثبت پالس ساعت و یا در لبه منفی بازنشانی اجرا می‌کند.

تخصیص اجرایی تخصیصی است که در یک عبارت initial یا always ذکر می‌گردد. این برخلاف تخصیص مداوم یا پیوسته مورد بحث در بخش ۱۱-۴ همراه با مدل‌سازی روند داده است که در آن عبارت مرتباً ارزیابی می‌شود. دو نوع تخصیص اجرایی وجود دارد: بلوکی و غیربلوکی . این دو با سمبل‌هایی که به کار می‌روند از یکدیگر تفکیک می‌گردند. تخصیص بلوکی از سمبل (=) به عنوان عملگر تخصیص استفاده می‌نماید و تخصیص‌های غیربلوکی سمبل (= <) را به کار می‌برد.

عبارات تخصیص بلوکی به طور متوالی و برحسب لیست شدن در بلوک ترتیبی، اجرا می‌شوند. تخصیص‌های غیربلوکی عبارات را از سمت راست ارزیابی می‌کند و تخصیص را تا اتمام ارزیابی عبارات در سمت چپ پدید نمی‌آورند. دو نوع تخصیص را می‌توان با تشریح آن بهتر درک نمود. دو تخصیص بلوک اجرایی زیر را در نظر بگیرید:

```
B = A
C = B + 1
```

عبارت اول A را به B منتقل می‌کند. دومین عبارت مقدار جدیدی را با افزودن B به میزان یک واحد و انتقال آن به C انجام می‌دهد. در پایان C برابر با A + 1 است. اکنون دو عبارت تخصیص غیربلوکی را در نظر بگیرید:

```
B <= A
C <= B + 1
```

وقتی عبارات اجرا شوند، عبارات سمت راست ارزیابی شده و در مکانی موقت نگهداری می‌شوند. مقدار A در یک مکان از حافظه نگهداری می‌شود در حالی که B + 1 در مکانی دیگر حفظ می‌گردد. در پایان عبارات در بلوک متوالی ارزیابی شده و ذخیره می‌شود، و تخصیص به مقصد سمت چپ صورت می‌گیرد. در این حال C حاوی مقدار اولیه B به علاوه 1 است. اغلب مثال‌ها در این فصل و فصل بعد می‌توانند از تخصیص‌های بلوکی استفاده کنند. هنگامی که با طراحی سطح انتقال ثباتی، مثل فصل ۸، سروکار داریم تخصیص غیربلوکی بهتر و تحکم آمیز است.

فلیپ فلاپ‌ها و لچ‌ها

مثال‌های ۱-۵ تا ۴-۵ HDL توصیفی از انواع فلیپ فلاپ‌ها و لچ D را نشان داد. لچ D شفاف است و در حین فعال بودن کنترل ورودی، پاسخ لازم به یک تغییر در ورودی داده، را با یک تغییر در خروجی فراهم می‌سازد. توصیف مدولی یک لچ D در مثال ۱-۵ HDL نشان داده شد. این مدول دارای دو ورودی، D و کنترل C ، و یک خروجی Q است. چون Q در یک عبارت اجرایی ارزیابی می‌شود، باید از نوع reg تصور شود. لچ‌ها به سطوح سیگنال ورودی پاسخ می‌دهند بنابراین ورودی‌ها در عبارت کنترل

مثال ۵-۱، HDL

```
//Description of D latch (See Fig. 5-6)
module D_latch (Q,D,control);
   output Q;
   input D,control;
   reg Q;
   always @ (control or D)
   if (control) Q = D;      //Same as: if (control == 1)
endmodule
```

واقعه بدون نشانه لبه و به دنبال @ در عبارت always ذکر می‌شوند. در اینجا تنها یک عبارت تخصیص اجرایی بلوکی وجود دارد و انتقال ورودی D را به خروجی Q اگر کنترل صحیح (در منطق 1) باشد مشخص می‌نماید. توجه کنید که این عبارت هر بار که تغییر در D رخ دهد و کنترل در 1 باشد، اجرا خواهد شد.

مثال ۵-۲ HDL دو فلیپ فلاپ D را با دو مدول توصیف می‌کند. اولی فقط به پالس ساعت واکنش نشان می‌دهد و دومی دارای یک ورودی بازنشانی غیرهمزمان نیز هست. خروجی Q باید علاوه بر خروجی بودن به عنوان داده نوع reg هم اعلان شود. دلیل این است که در یک عبارت تخصیص اجرایی این خروجی، هدف می‌باشد. کلمه کلیدی posedge انتقال ورودی D را به Q فقط در لبه مثبت گذر CLK تضمین می‌نماید. هر تغییر در D در هر زمان دیگر Q را تغییر نخواهد داد.

مدول دوم علاوه بر ساعت همزمان کننده، دارای ورودی بازنشان غیرهمزمان نیز هست. برای تولید چنین فلیپ فلاپی فرم خاصی از عبارت if به‌کار می‌رود. عبارت واقعه یا رخداد پس از سمبل @ در عبارت always ممکن است هر تعداد از وقایع لبه از نوع posedge یا negedge باشد. یکی از وقایع باید رخداد یک ساعت در نظر گرفته شود. بقیه وقایع شرایطی را مشخص می‌نمایند که تحت آن منطق غیرهمزمان اجرا

مثال ۵-۲، HDL

```
//D flip-flop
module D_FF (Q,D,CLK);
   output Q;
   input D,CLK;
   reg Q;
   always @ (posedge CLK)
     Q = D;
endmodule

//D flip-flop with asynchronous reset.
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @(posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;     // Same as: if (RST == 0)
     else Q = D;
endmodule
```

می‌شود، هر if یا else if در عبارات تخصیص اجرایی مربوط به یک واقعه غیرهمزمان است. آخرین عبارت else مربوط به رخداد ساعت است. در مدول دوم مثال ۲-۵ دو رخداد لبه وجود دارد.

واقعه negedge RST (بازنشانی) از نوع غیرهمزمان است زیرا با عبارت if(~RST) انطباق دارد. مادامی که RST = 0 است، Q به 0 پاک می‌گردد. اگر CLK گذر مثبت داشته باشد، اثرش بلوکه می‌شود. تنها وقتی که RST = 1 است واقعه ساعت posedge به طور همزمان D را به Q منتقل می‌نماید.

معمولاً لازم است فلیپ فلاپ‌ها دارای سیگنال ورودی بازنشانی یا پیش تنظیم باشند، در غیر این صورت، حالت اولیه مدار ترتیبی را نمی‌توان معین کرد. یک مدار ترتیبی را نمی‌توان با HDL شبیه‌سازی کرد مگر این که حالت اولیه‌ای همراه با یک سیگنال ورودی به آن تخصیص یابد.

مثال ۳-۵ HDL ساخت فلیپ فلاپ T یا JK را با فلیپ فلاپ D و تعدادی گیت نشان می‌دهد. مدار با استفاده از معادلات مشخصه فلیپ فلاپ‌ها توصیف شده است.

$$Q(t+1) = Q \oplus T \qquad \text{برای یک فلیپ فلاپ } T$$
$$Q(t+1) = JQ' + K'Q \qquad \text{برای یک فلیپ فلاپ } JK$$

مثال ۳-۵ HDL

```
//T flip-flop from D flip-flop and gates
module TFF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   wire DT;
   assign DT = Q ^ T ;
//Instantiate the D flip-flop
   DFF TF1 (Q,DT,CLK,RST);
endmodule

//JK flip-flop from D flip-flop and gates
module JKFF (Q,J,K,CLK,RST);
   output Q;
   input J,K,CLK,RST;
   wire JK;
   assign JK = (J & ~Q) | (~K & Q);
//Instantiate D flipflop
   DFF JK1 (Q,JK,CLK,RST);
endmodule

//D flip-flop
module DFF (Q,D,CLK,RST);
   output Q;
   input D,CLK,RST;
   reg Q;
   always @ (posedge CLK or negedge RST)
     if (~RST) Q = 1'b0;
     else Q = D;
endmodule
```

مثال ۴-۵، HDL

```
// Functional description of JK flip-flop
module  JK_FF (J,K,CLK,Q,Qnot);
   output Q,Qnot;
   input  J,K,CLK;
   reg  Q;
   assign Qnot = ~ Q ;
   always @ (posedge CLK)
          case ({J,K})
            2'b00: Q = Q;
            2'b01: Q = 1'b0;
            2'b10: Q = 1'b1;
            2'b11: Q = ~ Q;
          endcase
endmodule
```

مدول اول TFF یک فلیپ فلاپ T ساخته شده با DFF را ذکر می‌کند. سیم DT طبق نیاز برای تبدیل فلیپ فلاپ D به T به XOR مربوط به Q و T تخصیص یافته است. ذکر مقدار DT به جای D در مدول DFF، تولید فلیپ فلاپ T موردنظر را می‌نماید. فلیپ فلاپ JK هم به روشی مشابه با استفاده از معادله مشخصه آن انجام می‌شود تا به این ترتیب جایگزینی برای D در DFF باشد.

مثال ۴-۵ HDL روش دیگری برای توصیف یک فلیپ فلاپ JK را نشان می‌دهد. در اینجا ما به جای معادله مشخصه از جدول مشخصه استفاده می‌کنیم. شرط انشعاب چندگانه case، عدد دو بیتی حاصل از بیت‌های J و K را چک می‌کند. مقدار case ({K و J}) ارزیابی شده و با مقادیر در لیست عباراتی که به دنبال می‌آیند مقایسه می‌شوند. اولین مقداری که با شرط صحیح تطابق داشته باشد اجرا می‌گردد. چون ترکیب J و K یک عدد دو بیتی را تولید می‌کند، عدد می‌تواند 00، 01، 10 یا 11 باشد. اولین عدد مقدار J و دومی مقدار K را بدست می‌دهد. چهار شرط ممکن مقدار حالت بعدی Q پس از اعمال لبه مثبت پالس ساعت را مشخص می‌نماید.

نمودار حالت

عملکرد مدارهای ترتیبی در HDL به وسیله قالب مشابهی با نمودار حالت توصیف می‌گردد. نمودار حالت مدل میلی در مثال ۵-۵ HDL ارائه شده است. ورودی، خروجی، ساعت و بازنشانی به روش معمولی اعلان می‌شوند. حالت فلیپ فلاپ‌ها با شناسه Prstate و Nxtstate اعلان می‌گردد. این متغیرها مقدار حالت یک مدار ترتیبی را نگه می‌دارند. تخصیص دودویی حالت با استفاده از یک عبارت parameter صورت می‌گیرد. Verilog تعریف ثابت‌ها را در مدول با کلمه کلیدی parameter اجازه می‌دهد. به چهار حالت S0 تا S3 اعداد دودویی 00 تا 11 تخصیص داده شده است. عبارت S2=2'b10 بر S2 = 2 ترجیح داده می‌شود. اولی از دو بیت برای ذخیره یک ثابت استفاده می‌کند. دومی اعداد دودویی 32 (یا 64) بیت را نتیجه می‌دهد.

```
//Mealy state diagram (Fig. 5-16)
module Mealy_mdl (x,y,CLK,RST);
  input x,CLK,RST;
  output y;
  reg y;
  reg [1:0] Prstate, Nxtstate;
  parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
   always @ (posedge CLK or negedge RST)
      if (~RST) Prstate = S0;  //Initialize to state S0
      else Prstate = Nxtstate; //Clock operations
   always @ (Prstate or x)     //Determine next state
         case (Prstate)
            S0: if (x) Nxtstate = S1;
                  else Nxtstate = S0;
            S1: if (x) Nxtstate = S3;
                  else Nxtstate = S0;
            S2: if (~x)Nxtstate = S0;
                  else Nxtstate = S2;
            S3: if (x) Nxtstate = S2;
                  else Nxtstate = S0;
         endcase
   always @ (Prstate or x)     //Evaluate output
         case (Prstate)
            S0: y = 0;
            S1: if (x) y = 1'b0; else y = 1'b1;
            S2: if (x) y = 1'b0; else y = 1'b1;
            S3: if (x) y = 1'b0; else y = 1'b1;
         endcase
  endmodule
```

توصیف HDL از سه بلوک always استفاده می‌نماید که به طور همروند اجرا می‌شوند و از طریق متغیرهای مشترک با هم واکنش نشان می‌دهند. اولین عبارت always مدار را به حالت اولیه S0 = 00 باز می‌نشاند و عملکرد ساعت‌دار همزمان را بیان می‌دارد. عبارت Prstate = Nxtstate تنها در پاسخ به لبه مثبت پالس ساعت اجرا می‌گردد. این بدان معنی است که هر تغییر در مقدار Nxtsate در بلوک always دوم به عنوان یک واقعه posedge به prstate منتقل می‌شود. دومین بلوک always حالت بعدی را به عنوان تابعی از حالت فعلی ورودی معین می‌نماید. شرط انشعاب چند مسیری، رشته حالات را در نمودار حالت شکل ۱۶-۵ دنبال می‌کند. سومین بلوک always خروجی را به عنوان یک تابع از حالت فعلی و ورودی ارزیابی می‌نماید. هر چند این بلوک به طور جداگانه به خاطر وضوح لیست شده است، ولی می‌تواند همراه با بلوک دوم ذکر شود. توجه کنید که اگر ورودی x در هر حالتی تغییر کند، مقدار خروجی y ممکن است تغییر یابد.

مثالی از نمودار حالت مدل مور در مثال ۶-۵ HDL آورده شده است. این مثال نشان می‌دهد که می‌توان گذرهای حالت را تنها با یک بلوک always نشان داد. حالت فعلی مدار با متغیر state بیان

```
//Moore state diagram (Fig. 5-19)
module Moore_mdl (x,AB,CLK,RST);
   input x,CLK,RST;
   output [1:0]AB;
   reg [1:0] state;
   parameter S0 = 2'b00, S1 = 2'b01, S2 = 2'b10, S3 = 2'b11;
      always @ (posedge CLK or negedge RST)
         if (~RST) state = S0;  //Initialize to state S0
         else
         case (state)
           S0: if (~x) state = S1; else state = S0;
           S1: if (x)  state = S2; else state = S3;
           S2: if (~x) state = S3; else state = S2;
           S3: if (~x) state = S0; else state = S3;
         endcase
   assign AB = state;          //Output of flip-flops
endmodule
```

می‌شود. گذرهای حالت با لبه مثبت ساعت یعنی posedge CLK طبق شرایط عبارت case رخ می‌دهند. خروجی مدار مستقل از ورودی است و مستقیماً از خروجی‌های فلیپ فلاپ اخذ می‌شوند. خروجی 2 بیت AB با عبارت assign معین می‌شود و برابر با مقدار حالت فعلی است.

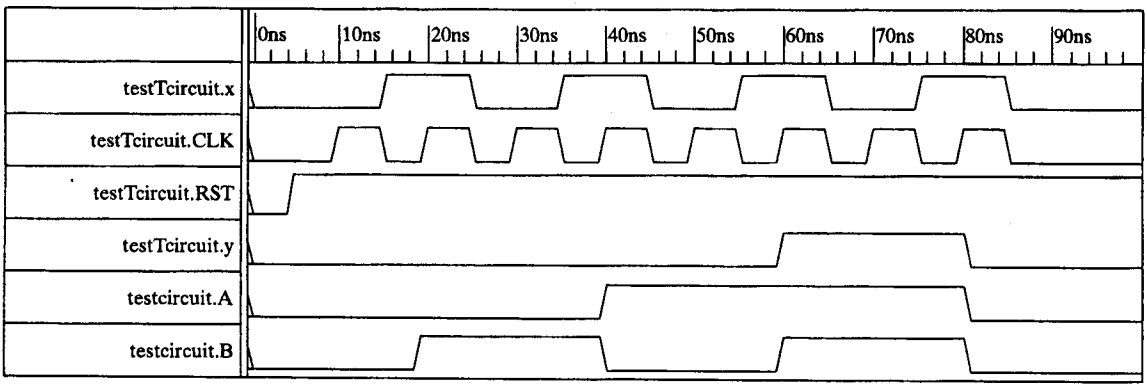
توصیف ساختاری

مدارهای ترکیبی را می‌توان در HDL با استفاده از عبارات سطح گیت یا روند داده توصیف کرد. مدارهای ترتیبی از عبارات رفتاری استفاده می‌کنند تا عملکرد فلیپ فلاپ‌ها را توصیف نمایند. چون یک مدار ترتیبی از فلیپ فلاپ‌ها و گیت‌ها استفاده می‌کند، ساختارش با ترکیبی از عبارات روند داده و رفتاری توصیف می‌گردد. فلیپ فلاپ‌ها با عبارت always توصیف می‌شوند. بخش ترکیبی را می‌توان با عبارات assign و معادلات بول توصیف کرد. آنگاه می‌توان بلوک‌های جداگانه را با ذکر با هم ترکیب کرد.

توصیف ساختاری یک مدار ترتیبی در مثال ۵-۷ HDL نشان داده شده است. در مثال دو مدول وجود دارد. اولی مدار شکل ۵-۲۰ (الف) را توصیف می‌کند. دومی فلیپ فلاپ T را توصیف می‌نماید. مدول دیگری نیز وجود دارد که نیروی محرکه لازم را برای تست عملکرد مدار فراهم می‌سازد. مدار ترتیبی یک شمارنده دودویی است که با ورودی x کنترل می‌شود. خروجی y وقتی که شمارش دودویی به 11 برسد، فعال می‌شود. فلیپ فلاپ‌های A و B نیز به عنوان خروجی‌ها به حساب آمده‌اند تا بتوان عملکرد آنها را چک کرد. معادلات ورودی فلیپ فلاپ‌ها و معادله خروجی با عبارات assign با توجه به عبارات بولی ارزیابی می‌شوند. آنگاه فلیپ فلاپ T با TA و TB و معادلات ورودی ذکر می‌گردد.

مدول دوم فلیپ فلاپ T را توصیف می‌کند. ورودی RST فلیپ فلاپ را در لبه منفی به 0 باز می‌نشاند. عملکرد مدار با معادله مشخصه‌اش، یعنی $Q(t+1) = Q \oplus T$ مشخص می‌شود.

```
//Structural description of sequential circuit
//See Fig. 5-20(a)
module Tcircuit (x,y,A,B,CLK,RST);
   input x,CLK,RST;
   output y,A,B;
   wire TA,TB;
//Flip-flip input equations
   assign TB = x,
          TA = x & B;
//Output equation
   assign y = A & B;
//Instantiate T flip-flops
   T_FF BF (B,TB,CLK,RST);
   T_FF AF (A,TA,CLK,RST);
endmodule

//T flip-flop
module T_FF (Q,T,CLK,RST);
   output Q;
   input T,CLK,RST;
   reg Q;
     always @ (posedge CLK or negedge RST)
       if (~RST) Q = 1'b0;
       else Q = Q ^ T;
endmodule

//Stimulus for testing sequential circuit
module testTcircuit;
  reg x,CLK,RST;  //inputs for circuit
  wire y,A,B;     //output from circuit
  Tcircuit TC (x,y,A,B,CLK,RST);  // instantiate circuit
  initial
     begin
         RST = 0;
         CLK = 0;
      #5 RST = 1;
         repeat (16)
      #5 CLK = ~CLK;
     end
  initial
     begin
          x = 0;
      #15 x = 1;
          repeat (8)
      #10 x = ~ x;
     end
endmodule
```



شکل ۲۱-۵. شبیه سازی خروجی با مثال ۷-۵ HDL

مدول محرک، ورودی‌ها به مدار را برای چک کردن پاسخ فراهم می‌سازد. اولین بلوک initial هشت سیکل ساعت را با پریود 10ns تهیه می‌کند. دومین بلوک inital هر تغییر در ورودی x راکه در لبه منفی پالس ساعت رخ دهد معین می‌سازد. نتیجه شبیه‌سازی در شکل ۲۱-۵ ملاحظه می‌گردد. خروجی A و B وارد رشته‌ای از ترکیبات دودویی 00 ، 01 ، 10 و 11 و سپس به 00 می‌شود. تغییر در شمارش در هنگام لبه مثبت ساعت و با شرط $x = 1$ رخ می‌دهد. اگر $x = 0$ باشد شمارش تغییری نمی‌کند. وقتی هر دو A و B برابر 1 باشد خروجی y برابر 1 خواهد بود. این خود صحت عمل مدار را تأیید می‌نماید.

۶-۵ کاهش و تخصیص حالت

تحلیل مدارهای ترتیبی از نمودار مدار آغاز و به جدول یا نمودار حالت پایان می‌یابد. طراحی یک مدار ترتیبی با مجموعه‌ای از مشخصات و ویژگی‌ها شروع و به یک نمودار منطقی ختم می‌شود. روال‌های طراحی از بخش ۷-۵ به بعد ارائه خواهند شد. این بخش خواص معینی از مدارهای ترتیبی که ممکن است تعداد گیت‌ها و فلیپ فلاپ‌ها را در طراحی کاهش دهد مورد بحث قرار خواهد داد.

کاهش حالت

کاهش تعداد فلیپ فلاپ‌ها در یک مدار ترتیبی را کاهش حالت می‌نامند. الگوریتم‌های کاهش حالت رویه‌هایی را شامل می‌شود که در آن تعداد حالات در یک جدول حالت کاهش می‌یابد، ضمن این که نیازهای ورودی و خروجی حفظ می‌گردند. چون m فلیپ فلاپ، 2^m حالت را تولید می‌کنند، کاهش در تعداد حالات ممکن است گاهی به کاهش در تعداد فلیپ فلاپ‌ها منجر گردد. یک تأثیر پیش‌بینی نشده در کاهش تعداد فلیپ فلاپ‌ها این است که گاهی مدار معادل با چند فلیپ فلاپ کمتر ممکن است گیت‌های بیشتری در بخش ترکیبی لازم داشته باشد.

ما روال کاهش حالت را با مثالی تشریح می‌کنیم. برای این کار ویژگی‌های نمودار حالت شکل ۲۲-۵

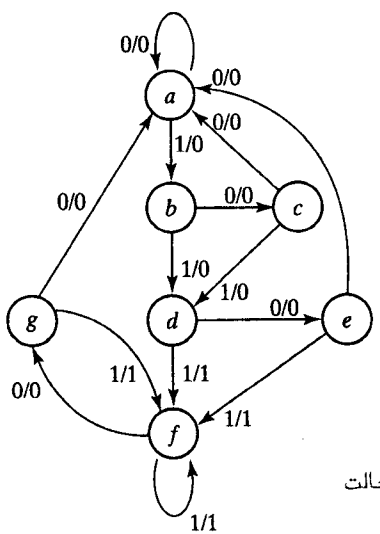
را برمی‌گزینیم. در این مثال تنها رشته‌های ورودی- خروجی اهمیت دارند؛ و از حالات داخلی فقط برای تهیه رشته موردنظر استفاده می‌شود. به این دلیل حالات درون دایره‌ها با سمبل‌های حرفی، به جای مقادیر دودویی ذکر شده‌اند. این برخلاف یک شمارنده دودویی است که در آن مقدار دودویی رشته حالات به عنوان خروجی به بیرون برده می‌شوند.

تعداد رشته‌های ورودی قابل اعمال به مدار بی‌نهایت است؛ و هر کدام، رشته خروجی منحصر به فرد خود را دارد. مثلاً رشته ورودی 01010110100 را با حالت اولیه a در نظر بگیرید. هر ورودی 0 یا 1، یک خروجی 0 یا 1 تولید کرده و موجب می‌شود مدار به حالت بعدی برود. با توجه به نمودار حالت ملاحظه می‌شود که رشته خروجی و حالت برای رشته ورودی مفروض به طریق زیر بدست می‌آید. با فرض قرار داشتن مدار در حالت اولیه a، یک ورودی 0 یک خروجی 0 تولید کرده و مدار در همان حالت a باقی می‌ماند. با حالت فعلی a اگر ورودی 1 باشد، خروجی 0 و حالت بعدی b است. با حالت فعلی b و ورودی 0، خروجی 0 و حالت بعدی c است. با ادامه این روند رشته کامل را مطابق زیر خواهیم یافت.

حالت	a	g	f	g	f	f	e	d	c	b	a	a
ورودی	0	0	1	0	1	1	0	1	0	0	1	0
خروجی	0	0	1	0	1	1	0	1	0	0	0	0

در هر ستون، حالت فعلی، مقدار ورودی و مقدار خروجی را داریم. حالت بعدی هر حالت در ستون بعدی نوشته شده است. باید متذکر شد که در این مدار، حالات از درجه دوم اهمیت برخوردارند زیرا ما فقط به رشته خروجی حاصل از رشته ورودی علاقمندیم.

اکنون اجازه بدهید فرض کنیم که یک مدار ترتیبی را که نمودار حالت آن کمتر از 7 حالت دارد در اختیار داریم و مایلیم آن را با مداری که نمودار حالتش در شکل ۲۲-۵ داده شده است مقایسه نماییم. اگر ورودی یکسانی به هر دو مدار اعمال شود و خروجی‌های مشابهی برای تمام رشته ورودی تولید گردد،



شکل ۲۲-۵. نمودار حالت

جدول ۶-۵ . جدول حالت

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

می‌توان یکی را به جای دیگری جایگزین کرد. موضوع کاهش حالت یافتن راهی برای کاهش حالات در مدار ترتیبی بدون تغییر رابطه ورودی- خروجی است.

اکنون به منظور کاهش حالات این مثال به پیش می‌رویم. ابتدا به جدول حالت نیاز داریم. در اعمال رویه‌ها برای کاهش حالت استفاده از جدول، مناسب‌تر از استفاده از نمودار است. جدول حالت مدار در جدول ۶-۵ لیست شده و مستقیماً از نمودار حالت بدست آمده است. الگوریتم کاهش یک جدول حالت کامل در اینجا بدون اثبات ارائه شده است: یعنی "دو حالت را معادل گوییم اگر برای هر عنصر از مجموعه ورودی‌ها، دقیقاً خروجی یکسانی را تولید کنند و مدار را به همان حالت یا به یک حالت معادل ببرند". وقتی دو حالت معادل هستند، بدون تغییر رابطه ورودی- خروجی، می‌توان یکی از آن دو را حذف کرد.

حال این الگوریتم را به جدول ۶-۵ اعمال می‌نماییم. با مراجعه و پیشروی در جدول حالت به دنبال دو حالتی می‌گردیم که به حالت بعدی یکسانی بروند و به ازاء هر دو ترکیب، ورودی- خروجی یکسانی داشته باشند. حالت g و e این چنین حالت‌هایی هستند. هر دو به حالات a و f رفته و خروجی‌های 0 و 1 را برای $x = 0$ و $x = 1$ تولید می‌کنند. بنابراین حالات g و e معادلند و یکی از این حالات قابل حذف است. روال حذف یک حالت و جایگزینی با معادلش در جدول ۷-۵ ملاحظه می‌شود. سطری که حالت فعلی g را دارد حذف و حالت g در ستون حالت بعدی با e جایگزین می‌گردد. اکنون حالت فعلی f دارای حالت بعدی e و f و خروجی‌های 0 و 1 به ترتیب برای $x = 0$ و $x = 1$

جدول ۷-۵ . کاهش جدول حالت

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	e	f	0	1

جدول ۸-۵ . جدول حالت کاهش یافته

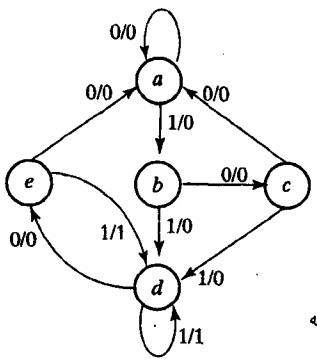
حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

می‌باشد. حالات بعدی و خروجی‌های یکسانی در سطر مربوطه به حالت فعلی d ملاحظه می‌شود. بنابراین حالات f و d معادلند و می‌توان حالت f را حذف و آن را با d جایگزین کرد. جدول کاهش یافته نهایی در جدول ۸-۵ ملاحظه می‌گردد. نمودار حالت برای جدول کاهش یافته فقط دارای پنج حالت بوده و در شکل ۲۳-۵ نشان داده شده است. این نمودار حالت مشخصات اولیه ورودی- خروجی را داراست و رشته خروجی موردنظر را برای هر رشته ورودی فراهم می‌سازد. لیست حاصل از نمودار حالت شکل ۲۳-۵ مربوط به رشته ورودی به کار رفته قبلی است. توجه کنید که گرچه رشته حالت متفاوت است، ولی رشته خروجی یکسانی تولید شده است.

حالت	a	a	b	c	d	e	d	d	e	d	e	a
ورودی		0	1	0	1	0	1	1	0	1	0	0
خروجی		0	0	0	0	0	1	1	0	1	0	0

در حقیقت اگر ما g را با e و f را با d جایگزین کنیم، این رشته همان رشته شکل ۲۱-۵ است.

یافتن حالات معادل با چک کردن هر جفت حالت با روالی که از جدول ایجاب استفاده می‌کند صورت می‌گیرد. جدول ایجاب متشکل از مربعاتی است که هر یک از این مربعات به یک جفت حالت معادل تعلق دارد. با به‌کارگیری صحیح جدول، می‌توان همه حالات معادل را در جدول حالت بدست آورد. کاربرد جدول ایجاب در کاهش تعداد حالات در جدول حالت در بخش ۵-۹ نشان داده شده است. مدار ترتیبی این مثال تعداد حالات را از هفت به پنج کاهش داد. به طورکلی، کاهش در تعداد حالات در جدول حالت ممکن است به مداری با عناصر کمتر منجر شود. با این وجود کاهش حالات در جدول حالت، تضمینی در کاهش تعداد فلیپ فلاپ یا تعداد گیت‌ها ندارد.



شکل ۲۳-۵ . نمودار حالت کاهش یافته

جدول ۹-۵. سه تخصیص ممکن دودویی حالت

حالت	تخصیص 1 دودویی	تخصیص 2 کد گری	تخصیص 3 یک بارز
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

تخصیص حالت

هنگام طراحی یک مدار ترتیبی با عناصر فیزیکی، لازم است به حالات، مقادیر دودویی کد شده‌ای تخصیص یابد. برای مداری با m حالت، باید کدی n بیتی اختیار شود که در آن $2^n \geq m$ است. مثلاً، با سه بیت می‌توان کدی را برای هشت حالت از 000 تا 111 تعریف کرد. اگر جدول حالت شکل ۶-۵ به کار برود، باید مقادیر دودویی را به هفت حالت اختصاص دهیم؛ تنها حالت باقیمانده هم مورد استفاده نیست. اگر جدول حالت ۸-۵ به کار برود، تنها پنج حالت به تخصیص دودویی نیاز دارند و در این مورد سه حالت به کار نرفته وجود خواهد داشت. در طراحی با حالات به کار نرفته به عنوان حالات بی‌اهمیت برخورد می‌شود. چون حالات بی‌اهمیت معمولاً در یافتن مدار ساده‌تر مفیدند، به نظر می‌رسد مدار پنج حالته گیت‌های ترکیبی کمتری از مدار هفت حالته داشته باشد.

ساده‌ترین راه کد دادن به پنج حالت استفاده از پنج عدد صحیح در شمارش دودویی در تخصیص 1 مطابق جدول ۹-۵ است. تخصیص مناسب دیگر کدگری در تخصیص 2 است. در این ستون به هنگام تغییر از یک عدد به یک عدد دیگر تنها یک بیت تغییر می‌یابد. این کد استقرار توابع بول را در نقشه کارنو برای ساده‌سازی آسان‌تر می‌کند. تخصیص بعدی که در طراحی سیستم‌های کنترل به کار می‌رود تخصیص 1 بارز است. در این آرایش تعداد بیت‌ها با تعداد حالات مدار برابر است. در هر زمان مشخص، تنها یک بیت برابر 1 و بقیه بیت‌ها در 0 نگهداری می‌شوند. در این تخصیص برای هر حالت یک فلیپ فلاپ به کار می‌رود.

جدول ۱۰-۵، جدول حالت کاهش یافته با به‌کارگیری تخصیص 1 به سمبل‌های حرفی حالات است. یک تخصیص متفاوت دیگر، جدول حالت دیگری با مقادیر دودویی متفاوت را نتیجه می‌دهد. فرم دودویی جدول حالت برای بخش ترکیبی مدار ترتیبی مورد استفاده است. پیچیدگی مدار ترکیبی به انتخاب تخصیص حالت دودویی بستگی دارد.

جدول ۱۰-۵. جدول حالت کاهش یافته با تخصیص 1، دودویی

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

گاهی نام جدول گذر در عوض جدول حالت به کار می‌رود. معمولاً نام جدول حالت در مواردی مورد استفاده قرار می‌گیرد که از حروف برای معرفی حالت استفاده بشود. در این کتاب، برای هر دو نوع جدول نام مشترکی را به کار خواهیم برد.

۷-۵ روش طراحی

طراحی یک مدار ترتیبی ساعت‌دار با بیان مجموعه‌ای از مشخصات و ویژگی‌های شروع شده و با نمودار منطقی یا لیستی از توابع بول که از آنها نمودار منطقی حاصل می‌شود پایان می‌یابد. برخلاف یک مدار ترکیبی که کلاً با جدول درستی مشخص می‌گردد، مدار ترتیبی برای مشخصات خود نیاز به جدول حالت دارد. اولین قدم در طراحی مدارهای ترتیبی تهیه جدول حالت یا نمایش معادلی از آن، مانند نمودار حالت است.

یک مدار ترتیبی همزمان، از فلیپ فلاپ‌ها و گیت‌های ترکیبی ساخته می‌شود. طراحی مدار شامل انتخاب فلیپ فلاپ‌ها و سپس یافتن ساختار گیتی بخش ترکیبی آن است تا مداری که مشخصات موردنظر را داراست حاصل گردد. تعداد فلیپ فلاپ‌ها با توجه به تعداد حالات لازم در مدار بدست می‌آید. مدار ترکیبی هم با استفاده از جدول حالات و ارزیابی معادلات ورودی و خروجی آن معین می‌گردد. در واقع، به محض این که نوع و تعداد فلیپ فلاپ‌ها معین شد، روند طراحی از مدار ترتیبی به مدار ترکیبی تبدیل می‌شود. بدین ترتیب تکنیک‌های مدار ترکیبی را می‌توان به کار گرفت.

روال طراحی مدارهای ترتیبی همزمان را می‌توان به صورت زیر خلاصه کرد.

۱- با توجه به عبارات و مشخصات مربوط به عملکرد موردنظر، یک نمودار حالت برای مدار بدست آورید.

۲- در صورت لزوم تعداد حالات را کاهش دهید.

۳- به حالات مقادیر دودویی را تخصیص دهید.

۴- جدول حالت کد شده دودویی را بدست آورید.

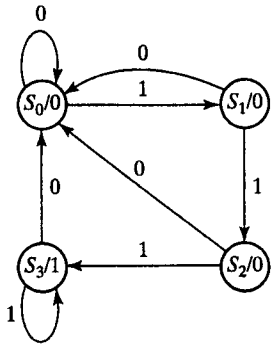
۵- نوع فلیپ فلاپ‌های به کار رفته را مشخص کنید.

۶- معادلات ساده شده ورودی فلیپ فلاپ‌ها و نیز معادلات خروجی را بدست آورید.

۷- نمودار منطقی را ترسیم کنید.

در مشخصاتی که رفتار مدار بیان می‌شود معمولاً فرض بر این است که خواننده با مفهوم منطق دیجیتال آشنایی دارد. در رسیدن به یک تفسیر صحیح از مشخصات مدار، تجربه و هوشمندی طراح نقش عمده‌ای دارد، زیرا توصیف لفظی ممکن است غیرکامل و غیرواقعی باشد. به محض دستیابی به چنین مشخصات و نمودار حالت، می‌توان از روش‌های شناخته شده طراحی برای تکمیل آن استفاده کرد. گرچه روال‌های ویژه‌ای در کاهش حالت و نیز تخصیص آن موجود است، ولی کمتر به وسیله طراحان با تجربه مورد استفاده قرار می‌گیرند.

مراحل ۴ تا ۷ در طرحی را می‌توان با الگوریتم‌های موجود پیاده‌سازی کرد و بنابراین قابل



شکل ۲۴-۵. نمودار حالت برای آشکارساز رشته

خودکارسازی است. بخشی از طراحی که روالی منظم را دنبال کند، سنتز یا ترکیب و یا ادغام می‌گویند.

اولین بخش، پرچالش‌ترین بخش طراحی است. ما در اینجا مثال ساده‌ای را ارائه خواهیم کرد تا نشان دهیم چگونه یک نمودار حالت از بیان لفظی بدست می‌آید.

می‌خواهیم مداری طراحی کنیم که به کمک آن سه 1 (و یا بیشتر) متوالی در یک رشته بیتی وارده از طریق یک خط ورودی شناسایی شوند. نمودار حالت مدار در شکل ۲۴-۵ نشان داده شده است. این نمودار با شروع از S_0 بدست آمده است. اگر ورودی 0 باشد، مدار در همان حالت قبل باقی می‌ماند، ولی اگر ورودی 1 شود، مدار به حالت S_1 می‌رود تا دریافت یک 1 را مشخص کند. اگر ورودی بعدی هم 1 باشد، تغییر به حالت S_2 به معنی دریافت دو 1 متوالی است، ولی اگر ورودی 0 باشد، به حالت قبلی S_0 باز می‌گردد. سومین 1 متوالی، مدار را به S_3 می‌برد. اگر 1های بیشتری تشخیص داده شود، مدار همچنان در S_3 باقی می‌ماند. هر ورودی 0 مدار را در این حالت به S_0 باز می‌گرداند. و به این ترتیب در ازاء 1های متوالی بیشتر مدار در همان S_3 باقی خواهد ماند. این مدار ترتیبی از نوع مدل مور است زیرا وقتی مدار در حالت S_3 است خروجی 1 و در غیر این صورت 0 است.

سنتز با فلیپ فلاپ‌های D

به محض این که نمودار حالت بدست آمد، بقیه طراحی از رویه سنتز سرراستی استفاده می‌کند. در واقع ما می‌توانیم مدار را به کمک توصیف HDL نمودار حالت و ابزارهای مناسب HDL برای یافتن یک netlist، طراحی کنیم. لازم به یادآوری است که توصیف HDL نمودار حالت مشابه با مثال ۶-۵ HDL در بخش ۵-۵ است. برای طراحی دستی مدار، باید کدهای دودویی را به حالات تخصیص دهیم و جدول حالت را لیست نماییم. این کار در جدول ۱۱-۵ انجام شده است. جدول از نمودار حالت شکل ۲۴-۵ با تخصیص مستقیم دودویی بدست می‌آید. برای نمایش چهار حالت از دو فلیپ فلاپ D استفاده می‌کنیم و آنها را با A و B نامگذاری می‌نماییم. در مدار یک ورودی x و یک خروجی y وجود دارد. معادله مشخصه فلیپ فلاپ D برابر $Q(t+1) = D_Q$ است و به این معنی است که مقادیر حالت بعدی در جدول حالت همان حالت ورودی D در فلیپ فلاپ است. معادلات ورودی فلیپ فلاپ

جدول ۱۱-۵ . جدول حالت برای آشکارساز رشته

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

مستقیماً از ستون‌های حالت بعدی A و B بدست آمده و برحسب مینترم به صورت زیر بیان می‌شوند.

$$A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$$
$$B(t+1) = D_B(A, B, x) = \Sigma(1, 5, 7)$$
$$y(A, B, x) = \Sigma(6, 7)$$

که در آن A و B حالت فعلی فلیپ فلاپ‌های A و B ، x ورودی، و D_A و D_B معادلات ورودی‌اند. مینترم‌های خروجی y از ستون خروجی در جدول حالت بدست می‌آیند.

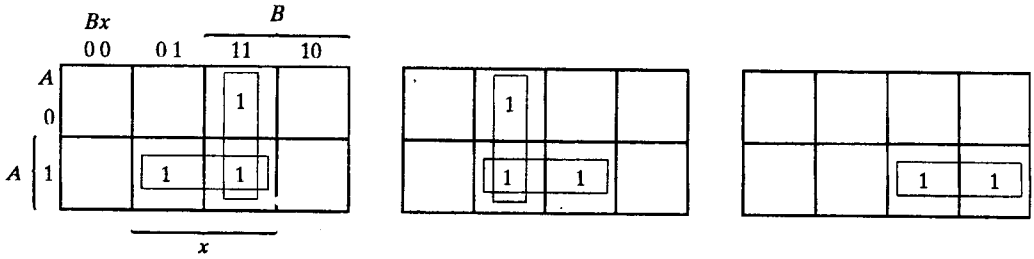
معادلات بولی که به کمک نقشه کارنو ساده شده‌اند در شکل ۲۵-۵ دیده می‌شوند. این معادلات عبارتند از:

$$D_A = Ax + Bx$$
$$D_B = Ax + B'x$$
$$y = AB$$

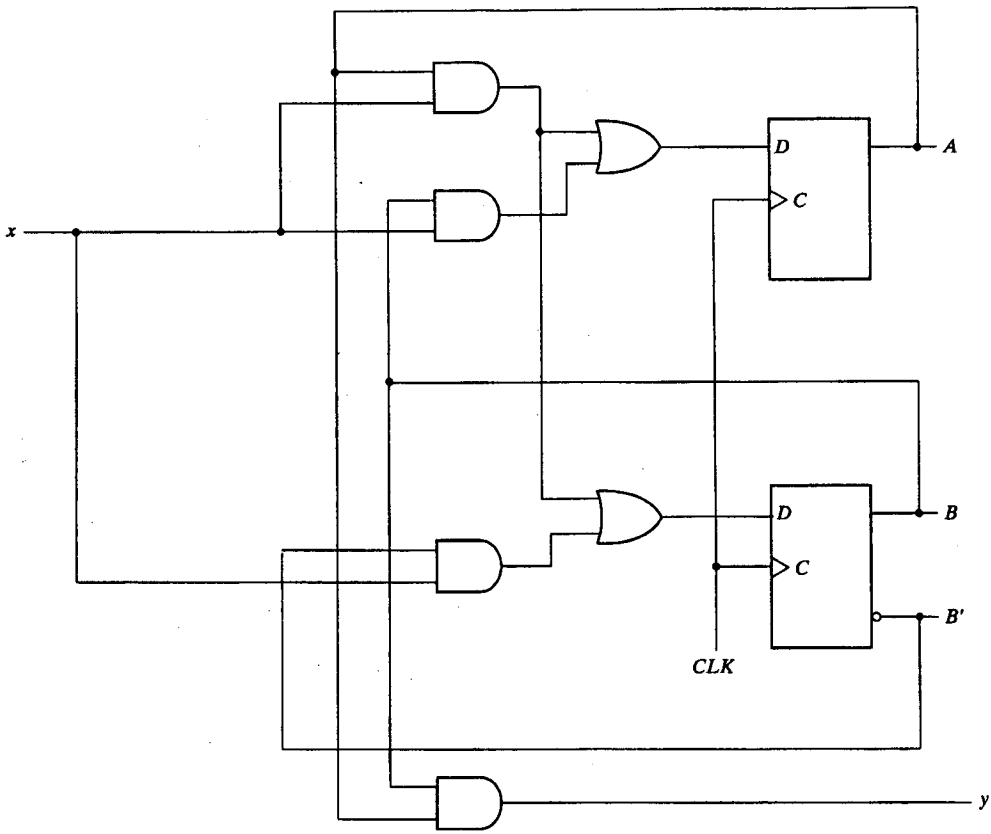
نمودار منطقی مدار ترتیبی در شکل ۲۶-۵ ترسیم شده است.

جداول تحریک

طراحی یک مدار ترتیبی با فلیپ فلاپ‌هایی به جز نوع D پیچیده است زیرا معادلات ورودی برای مدار باید به طور غیرمستقیم از جدول حالت بدست آید. وقتی که فلیپ فلاپ‌های D به کار می‌روند،



شکل ۲۵-۵ . نقشه‌ها برای آشکارساز رشته



شکل ۲۶-۵. نمودار منطقی آشکارساز رشته

معادلات ورودی مستقیماً از حالت بعدی بدست می‌آیند. این مطلب برای فلیپ فلاپ‌های JK و T چنین نیست. برای تعیین معادلات حالت این فلیپ فلاپ‌ها لازم است رابطه عملیاتی بین جدول حالت و معادلات ورودی را بدست آوریم.

جداول مشخصه فلیپ فلاپ‌ها که در جدول ۱-۵ مشاهده شد وقتی ورودی‌ها و حالات بعدی معلوم‌مند، مقدار حالت بعدی را فراهم می‌کنند. این جداول در تحلیل مدارهای ترتیبی و تعریف کار فلیپ فلاپ‌ها مفیدند. در حین فرآیند طراحی، ما معمولاً گذر از حالت فعلی به حالت بعدی را می‌دانیم و مایلیم وضعیت ورودی فلیپ فلاپ که موجب این گذر می‌گردد را پیدا کنیم. به این دلیل جدولی را بدست می‌آوریم که در آن برای هر تغییر مورد نظری در حالت، ورودی‌های لازم لیست شده باشند. چنین جدولی را جدول تحریک می‌گویند.

جدول ۱۲-۵ جداول تحریک را برای دو فلیپ فلاپ نشان می‌دهد. هر جدول ستونی برای حالت فعلی Q(t) ، حالت بعدی Q(t+1) و نیز ستونی برای هر ورودی دارد تا چگونگی وقوع گذر

جدول ۱۲-۵. جداول تحریک فلیپ فلاپ

$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

(ب) T

$Q(t)$	$Q(t+1)$	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

(الف) JK

را نشان دهد. از حالت فعلی به حالت بعدی چهارگذر می‌تواند وجود داشته باشد. شرایط ورودی لازم برای هر چهارگذر از اطلاعات موجود در جدول مشخصه بدست می‌آید. سمبل X در جدول حالت بی‌اهمیت را نشان می‌دهد و به این معنی است که 1 یا 0 بودن ورودی اهمیتی ندارد.

جدول تحریک فلیپ فلاپ JK در بخش (الف) مشاهده می‌شود. وقتی که هر دو حالت فعلی و بعدی 0 باشند، ورودی J باید در 0 بماند ولی K می‌تواند 0 یا 1 باشد. به طور مشابه هرگاه هر دو حالت فعلی و بعدی 1 باشند، K در 0 ولی J می‌تواند 0 یا 1 باشد. اگر فلیپ فلاپ قرار باشد از حالت 0 به حالت 1گذر کند، J باید 1 باشد زیرا ورودی J فلیپ فلاپ را در 1 می‌نشاند. با این وجود K ممکن است 0 و یا 1 باشد. اگر $K = 0$ باشد، $J = 1$ فلیپ فلاپ را مطابق میل در 1 می‌نشاند؛ اگر $K = 1$ و $J = 1$ باشد خروجی فلیپ فلاپ متمم شده و از حالت 0 به 1 می‌رود. بنابراین K با علامت بی‌اهمیت X برای گذر از 0 به 1، علامت خورده است. برای گذر از حالت 1 به حالت 0، باید $K = 1$ باشد، زیرا ورودی K فلیپ فلاپ را پاک می‌کند. با این وجود ورودی J می‌تواند 0 یا 1 باشد، زیرا $J = 0$ تأثیری ندارد و $J = 1$ همراه با $K = 1$ فلیپ فلاپ را متمم می‌کند که حاصل آن گذر از 1 به 0 است.

جدول تحریک برای فلیپ فلاپ T در بخش (ب) ملاحظه می‌شود. با توجه به جدول مشخصه، می‌بینیم که وقتی $T = 1$ است، فلیپ فلاپ متمم می‌گردد ولی اگر $T = 0$ باشد، حالت فلیپ فلاپ بی‌تغییر باقی خواهد ماند. بنابراین اگر قرار است فلیپ فلاپ بی‌تغییر بماند، لازم است $T = 0$ باشد و اگر قرار باشد حالت فلیپ فلاپ متمم شود $T = 1$ خواهد بود.

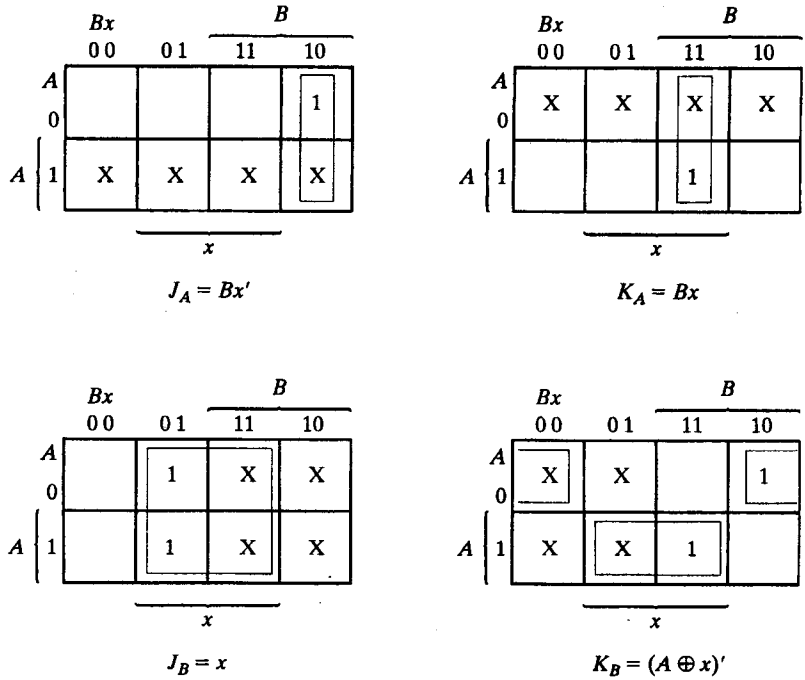
سنتز با استفاده از فلیپ فلاپ‌های JK

روال سنتز برای مدار ترتیبی حاوی فلیپ فلاپ JK مانند فلیپ فلاپ‌های D است، با این تفاوت که معادلات ورودی باید از حالت فعلی به بعدی ارزیابی شوند، که این کار با توجه به جدول تحریک صورت می‌گیرد. برای تشریح روال، مدار ترتیبی معرفی شده با جدول ۱۳-۵ را طراحی می‌کنیم. علاوه بر ستون‌های حالت فعلی، ورودی و حالت خروجی در جدول حالت معمولی، این جدول وضعیت ورودی فلیپ فلاپ، که معادلات ورودی از آن بدست می‌آیند را نیز نشان می‌دهد. این ورودی‌ها از جدول حالت و به همراهی جدول تحریک فلیپ فلاپ J معین می‌گردند. مثلاً در اولین سطر جدول ۱۳-۵ برای فلیپ فلاپ A گذری از حالت فعلی 0 به حالت بعدی 0 داریم. در جدول ۱۲-۵ برای فلیپ فلاپ J، لازمه این گذر یا انتقال این است که $J = 0$ و K به صورت بی‌اهمیت باشد. بنابراین 0 و X در

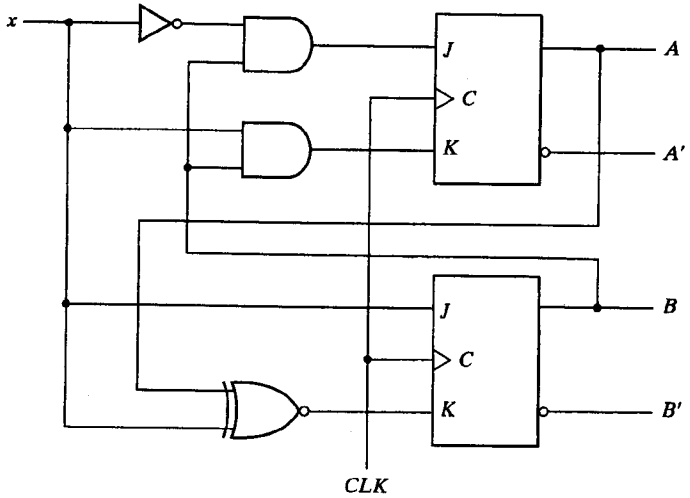
جدول ۱۳-۵. جدول حالت ورودی‌های فلیپ فلاپ JK

حالت فعلی		ورودی	حالت بعدی		ورودی‌های فلیپ فلاپ			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

اولین سطر زیر J_A و K_A وارد می‌شوند. چون در سطر اول فلیپ فلاپ B هم از حالت فعلی 0 به حالت بعدی 0 می‌رود، 0 و X به ترتیب زیر J_B و K_B وارد می‌شوند. سطر دوم جدول یک گذر از حالت فعلی 0 به حالت بعدی 1 در فلیپ فلاپ B را نشان می‌دهد. با توجه به جدول حالت، می‌بینیم که گذر از 0 به 1 لازم می‌دارد که $J = 1$ و $K = X$ باشد، بنابراین در سطر دوم و زیر ستون J_B و K_B، مقادیر 1 و X وارد می‌شوند. این فرآیند در جدول برای هر سطر و برای هر فلیپ فلاپ ادامه می‌یابد و هر وضعیت ورودی از جدول تحریک در سطر مناسبی برای هر فلیپ فلاپ کپی می‌شود.



شکل ۲۷-۵. نقشه‌ها برای معادلات ورودی J و K



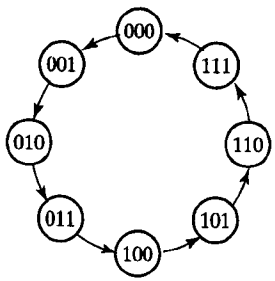
شکل ۲۸-۵ . نمودار منطقی برای مدار ترتیبی با فلیپ فلاپ‌های JK

ورودی‌های فلیپ فلاپ‌ها در جدول ۱۳-۵، در واقع جدول درستی را برای معادلات ورودی به عنوان تابعی از حالت فعلی A و B و ورودی x به نمایش می‌گذارند. معادلات ورودی در نقشه‌های شکل ۲۷-۵ ساده شده‌اند. مقادیر حالت بعدی در ضمن ساده‌سازی به کار نرفته‌اند زیرا معادلات ورودی فقط تابعی از حالات فعلی و ورودی بیرونی می‌باشند. به مزیت استفاده از فلیپ فلاپ‌های JK در طراحی توجه نمایید. چون تعداد قابل توجهی حالت بی‌اهمیت در جدول وجود دارد، مدار ترکیبی برای معادلات ورودی ساده‌تر خواهد بود، زیرا مینترم‌های بی‌اهمیت معمولاً به ساده‌سازی کمک می‌کنند. اگر حالات بی‌استفاده‌ای در جدول حالت وجود داشته باشد، حالات بی‌اهمیت دیگری نیز در نقشه وجود خواهد داشت.

چهار معادله ورودی برای دو فلیپ فلاپ JK در زیر نقشه‌های شکل ۲۷-۵ لیست شده‌اند. نمودار منطقی مدار ترتیبی در شکل ۲۸-۵ مشاهده می‌شود.

سنتز با استفاده از فلیپ فلاپ‌های T

سنتز مدارهای ترتیبی با فلیپ فلاپ‌های T به وسیله طراحی یک شمارنده دودویی نشان داده خواهد شد. یک شمارنده دودویی n بیت متشکل از n فلیپ فلاپ است که می‌تواند از 0 تا 2^n-1 را بشمارد. نمودار حالت یک شمارنده 3 بیت در شکل ۲۹-۵ دیده می‌شود. با توجه به حالات دودویی در



شکل ۲۹-۵ . نمودار حالت شمارنده دودویی 3 بیت

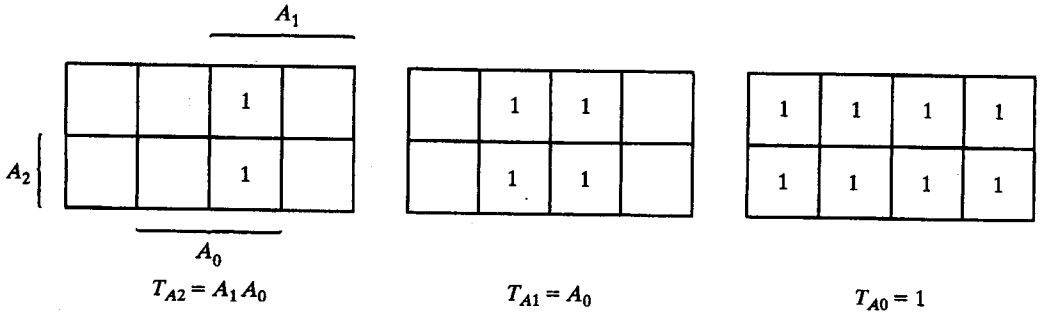
جدول ۱۴-۵. جدول حالت برای شمارنده 3 بیت

حالت فعلی			حالت بعدی			ورودی‌های فلیپ فلاپ		
A_2	A_1	A_0	A_2	A	A_0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	1	1
1	1	1	0	0	0	1	1	1

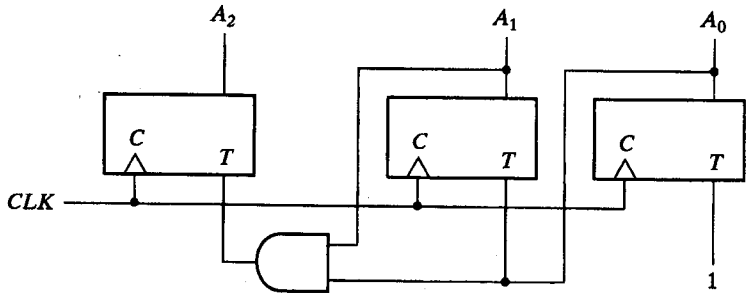
داخل دوایر، خروجی فلیپ فلاپ‌ها پس از 111 با بازگشت به 000 تکرار می‌شوند. خطوط جهت‌دار بین دوایر با مقادیر ورودی و خروجی علامت‌زنی نشده‌اند. به خاطر بسپارید که گذار حالات در مدارهای ترتیبی ساعت‌دار در لبه پالس ساعت به عنوان یک متغیر ورودی جداگانه در نمودار حالت یا جدول حالت ظاهر نمی‌گردد. از این نظر، نمودار حالت شمارنده نیازی به نشان دادن مقادیر ورودی و خروجی در امتداد خطوط جهت‌دار ندارد. در واقع تنها ورودی به مدار همان ساعت و خروجی‌ها هم حالات فعلی فلیپ فلاپ‌ها هستند. حالت بعدی شمارنده کلاً به حالت فعلی‌اش بستگی دارد و گذار حالت هر بار که پالس ساعت تغییر وضعیت دهد، رخ خواهد داد.

جدول ۱۴-۵، جدول حالت برای یک شمارنده دودویی 3 بیت است. سه فلیپ فلاپ با A_2 و A_1 و A_0 نام‌گذاری شده‌اند. شمارنده‌های دودویی غالباً با فلیپ فلاپ‌های T ساخته می‌شوند زیرا دارای خاصیت متمم‌سازی هستند. جدول تحریک برای ورودی‌های T از جدول تحریک فلیپ فلاپ T و وارسی گذر از حالت فعلی به حالت بعدی حاصل می‌گردد. به منظور تشریح، ورودی‌های فلیپ فلاپ را در سطر 001 ملاحظه کنید. حالت فعلی 001 و حالت بعدی 010 است که حالت بعدی مربوط به رشته شمارش است. از مقایسه این دو شماره مشاهده می‌شود که A_2 از 0 به 0 می‌رود، بنابراین T_{A2} با 0 نشان‌گذاری شده است، زیرا فیلپ فلاپ A_2 نباید به هنگام رخداد پالس ساعت عوض شود. A_1 از 0 به 1 می‌رود بنابراین T_{A1} با 1 علامت زده شده است زیرا فلیپ فلاپ باید با لبه پالس ساعت متمم شود. به طور مشابه A_0 از 1 به 0 می‌رود که بیانگر متمم شدن است و بنابراین T_{A0} با 1 نشان‌گذاری شده است. آخرین ردیف با حالت فعلی 111 با اولین مقدار شمارش یعنی 000 مقایسه می‌شود که در این هنگام حالت بعدی است. رفتن از تمام 1 به تمام 0 لازم می‌دارد تا هر سه فلیپ فلاپ متمم گردد.

معادلات ورودی فلیپ‌فلاپ در نقشه شکل ۳۰-۵ ساده شده است. توجه کنید که در T_{A0} همه مینترم‌ها 1 هستند زیراکم‌ارزش‌ترین بیت در هر شمارش متمم می‌گردد. تابع بولی که در آن همه مینترم‌ها لحاظ شود برابر مقدار ثابت 1 است. معادلات ورودی نوشته شده در زیر هر نقشه بخش ترکیبی شمارنده را مشخص می‌نماید. با استفاده از این توابع و سه فلیپ فلاپ، نمودار منطقی شمارنده مطابق شکل ۳۱-۵ خواهد بود.



شکل ۳۰-۵. نقشه برای شمارنده دودویی



شکل ۳۱-۵. نمودار منطقی شمارنده دودویی 3 بیت

مسائل

۵-۱ لچ D شکل ۶-۵ با چهار گیت NAND و یک وارونگر ساخته شده است. سه راه دیگر ساخت آن را در زیر ملاحظه کنید. در هر حالت، نمودار منطقی را رسم و عملکرد مدار را اصلاح نمایید.
(الف) گیت‌های NOR برای بخش لچ SR و گیت‌های AND برای دو گیت دیگر ممکن است به یک وارونگر هم نیاز داشته باشید.
(ب) از گیت‌های NOR برای هر چهار گیت استفاده نمایید. ممکن است وارونگر هم لازم داشته باشد.
(پ) فقط از چهار گیت NAND استفاده کنید (بدون وارونگر) این کار را می‌توانید با اتصال خروجی گیت فوقانی در شکل ۶-۵ که به ورودی گیت پایینی می‌رود، در عوض خروجی وارونگر انجام دهید. (خروجی گیت بالایی به لچ SR می‌رود).

۵-۲ با فلیپ فلاپ D یک فلیپ فلاپ JK بسازید. از یک مولتی پلکسر و یک وارونگر استفاده کنید.

۵-۳ نشان دهید که معادله مشخصه برای خروجی متمم شده فلیپ فلاپ JK مطابق زیر است.

$$Q'(t+1) = J'Q' + KQ$$

۵-۴ یک فلیپ فلاپ PN چهار نوع عمل را داراست: پاک‌کردن به 0، بدون تغییر، متمم و نشاندن به 1. هر یک از موارد فوق به ترتیب با PN برابر با 00، 01، 10 و 11 انجام می‌شود.

(الف) جدول مشخصه را پیاده کنید. (ب) معادله مشخصه را بدست آورید.

(پ) جدول تحریک را پیاده نمایید.

(ت) نشان دهید که می‌توان فلیپ فلاپ PN را به فلیپ فلاپ D تبدیل کرد.

۵-۵ تفاوت جدول درستی، جدول حالت، جدول مشخصه و جدول تحریک را بیان کنید. همچنین تفاوت تابع بول، معادله حالت، معادله مشخصه و معادله ورودی فلیپ فلاپ را توضیح دهید.

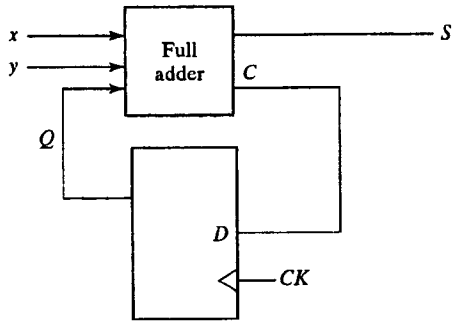
۵-۶ یک مدار ترتیبی با دو فلیپ فلاپ A و B از نوع D؛ دو ورودی x و y، و یک خروجی z با معادلات حالت بعدی و خروجی مشخص شده است.

$$A(t+1) = x'y + xA$$
$$B(t+1) = x'B + xA$$
$$z = B$$

(الف) نمودار منطقی مدار را رسم کنید. (ب) جدول حالت را برای مدار ترتیبی لیست کنید.

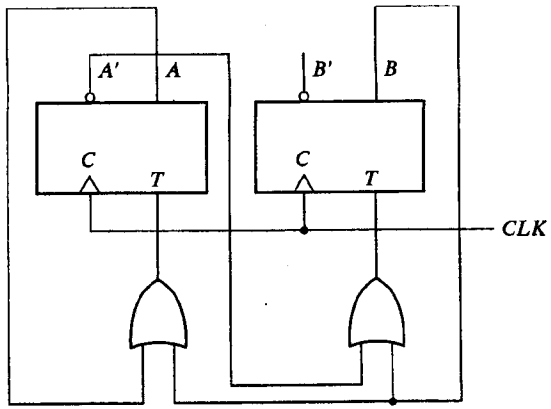
(پ) نمودار حالت مربوطه را بکشید.

۵-۷ یک مدار ترتیبی دارای یک فلیپ فلاپ Q، دو ورودی x و y و یک خروجی S است. این مدار دارای یک جمع‌کننده کامل متصل به یک فلیپ فلاپ D است، شکل (م ۵-۷). جدول حالت و نمودار حالت مدار ترتیبی را بدست آورید.



شکل (م ۵-۷)

۵-۸ جدول و نمودار حالت مدار ترتیبی شکل (م ۵-۸) را بدست آورید. عملکرد مدار را توضیح دهید.



شکل (م ۵-۸)

۵-۹ یک مدار ترتیبی دو فلیپ فلاپ A و B از نوع JK و یک ورودی x دارد. مدار با معادلات ورودی فلیپ فلاپ زیر تعریف شده است.

$$J_A = x \qquad K_A = B'$$
$$J_B = x \qquad K_B = A$$

(الف) معادلات حالت A(t+1) و B(t+1) را با جایگزینی معادلات ورودی در J و K بدست آورید.
(ب) نمودار حالت مدار را رسم کنید.

۵-۱۰ یک مدار ترتیبی دارای دو فلیپ فلاپ A و B از نوع JK، دو ورودی x و y و یک خروجی z است. معادلات ورودی فلیپ فلاپ و معادله خروجی عبارتند از:

$$J_A = Bx + B'y' \qquad K_A = B'xy'$$
$$J_B = A'x \qquad K_B = A + xy'$$
$$z = Ax'y' + Bx'y'$$

(الف) نمودار منطقی مدار را رسم نمایید. (ب) جدول حالت را پیاده کنید.
(پ) معادلات حالت را برای A و B بدست آورید.

۵-۱۱ با شروع از 00 در شکل ۵-۱۶، گذار حالت و رشته خروجی حاصل از یک رشته ورودی 0101101101110111110 را بدست آورید.

۵-۱۲ تعداد حالات را در جدول حالت زیر کاهش داده و جدول کاهش یافته را نشان دهید.

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	f	b	0	0
b	d	c	0	0
c	f	e	0	0
d	g	a	1	0
e	d	c	0	0
f	f	b	1	1
g	g	h	0	1
h	g	a	1	0

۵-۱۳ با شروع از حالت a و رشته ورودی 01110010011، رشته خروجی را برای موارد زیر بدست آورید.
(الف) جدول حالت مسئله قبل.
(ب) جدول حالت کاهش یافته در مسئله قبل. نشان دهید که برای هر دو رشته خروجی یکسانی بدست می‌آید.

۵-۱۴ تخصیص دودویی 2 را از جدول ۵-۹ در حالات جدول ۵-۸ جایگزین کنید و جدول حالت دودویی را بدست آورید.

۵-۱۵ یک جدول حالت برای فلیپ فلاپ JK با Q به عنوان حالت فعلی و حالت بعدی و ورودی‌های J و K لیست نمایید. مدار ترتیبی حاصل از جدول حالت را طراحی کنید و نشان دهید که معادل با شکل ۵-۱۲ (الف) است.

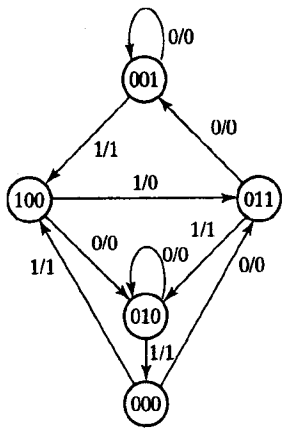
۵-۱۶ یک مدار ترتیبی با دو فلیپ فلاپ A و B از نوع D و یک ورودی x طراحی نمایید. وقتی x = 0 است، حالت مدار بدون تغییر باقی بماند. وقتی x = 1 است مدار وارد حالات 00، 01، 11 و 10 و بازگشت به 00 شده و کار را تکرار کند.

۵-۱۷ یک متمم 2 ساز یک ورودی، یک خروجی بسازید. مدار رشته‌ای از بیت‌ها را دریافت کرده و متمم 2 آن را در خروجی تولید می‌نماید. مدار قابل بازنشانی غیرهمزمان است تا عملیات را شروع و پایان دهد.

۵-۱۸ یک مدار ترتیبی با دو فلیپ فلاپ A و B نوع JK و دو ورودی E و x طراحی نمایید. اگر E = 0 باشد، مدار بدون توجه به x در حالت فعلی خود می‌ماند. وقتی E = 1 و x = 1 است وارد حالت 00، 01، 10 و 11 و بازگشت به 00 گشته و کار را تکرار کند. وقتی E = 1 و x = 0 است مدار وارد حالات 00، 11، 10 و 01 و به 00 رفته و عمل را تکرار نماید.

۵-۱۹ یک مدار ترتیبی دارای سه فلیپ فلاپ A، B و C؛ یک ورودی x و یک خروجی y است. نمودار حالت در شکل (م ۱۹-۵) مشاهده می‌گردد. قرار است مدار با استفاده از حالات بی‌اهمیت طراحی گردد. مدار طراحی شده را تحلیل کنید تا اثر حالات بی‌اهمیت ملاحظه شود.

(الف) از فلیپ فلاپ D استفاده کنید. (ب) از فلیپ فلاپ JK استفاده نمایید.



شکل (م ۱۹-۵)

۵-۲۰ مدار ترتیبی نمودار حالت شکل (۱۹-۵) را با فلیپ فلاپ‌های T طراحی کنید.

۵-۲۱ تفاوت اصلی بین یک عبارت initial و یک عبارت always در Verilog HDL چیست؟

۵-۲۲ شکل موج تولید شده با عبارت initial زیر را رسم کنید.

```
initial
  begin
  w = 0;  #20 w = 1;  # 50 w = 0;  # 30 w = 1;  #10 w = 0;
  end
```

۵-۲۳ عبارات زیر را با فرض این که Reg A در ابتدا مقدار 30 را دارد ملاحظه نمایید.

(الف)
```
RegA = 125
RegB = RegA
```

(ب)
```
RegA <= 125
RegB <= RegA
```

پس از اجرا مقادیر Reg A و Reg B چه هستند.

۵-۲۴ یک توصیف رفتاری HDL برای فلیپ فلاپ D بنویسید. برای آن ورودی پیش تنظیم و پاک کردن پیش‌بینی نمایید (به شکل ۱۳-۱۱ مراجعه شود).

۵-۲۵ یک فلیپ فلاپ حساس به لبه مثبت خاص دو ورودی D1 و D2 و یک ورودی کنترل دارد که یکی از آن دو را انتخاب می‌کند. توصیف رفتاری HDL آن را بنویسید.

۵-۲۶ یک توصیف رفتاری HDL با استفاده از عبارت if-else مبتنی بر مقدار حالت فعلی، برای فلیپ‌فلاپ JK بنویسید.

۵-۲۷ توصیف HDL مثال ۵-۵ را با ترکیب گذر حالات و خروجی در یک بلوک always بازنویسی کنید.

۵-۲۸ مدار ترتیبی شکل ۵-۱۷ را شبیه‌سازی نمایید.

(الف) توصیف HDL نمودار حالت را بنویسید.

(ب) توصیف HDL نمودار مدار را بنویسید.

(پ) یک HDL محرک با رشته‌ای از ورودی‌های 00، 01، 11، 10 بنویسید. نشان دهید که پاسخ برای هر دو توصیف یکی است.

۵-۲۹ توصیف HDL یک شمارنده دودویی 2 بیت در شکل ۵-۲۰ نشان داده شد. از مدول محرک مثال ۵-۷ HDL استفاده کنید و نشان دهید که پاسخ خروجی مشابه شکل موج ۵-۲۱ است.

۵-۳۰ نمودار منطقی را برای مدار ترتیبی توصیف شده با مدول HDL زیر رسم کنید.

```
module Seqcrt (A,B,C,Q,CLK);
   input A,B,C,CLK;
   output Q;
   reg Q,E;
   always @ (posedge CLK)
      begin
     E <= A & B;
     Q <= E | C;
      end
endmodule
```

چه تغییر، اگر لازم است، باید در مدار لحاظ شود اگر دو عبارت آخر از تخصیص بلوکی به جای غیربلوکی استفاده نماید.

مراجع

1. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.

2. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices,* 3rd ed. Upper Saddle River, NJ: Prentice Hall.

3. KATZ, R. H. 1994. *Contemporary Logic Design.* Upper Saddle River, NJ: Prentice Hall.

4. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.

5. NELSON V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.

ثَبّات‌ها و شمارنده‌ها

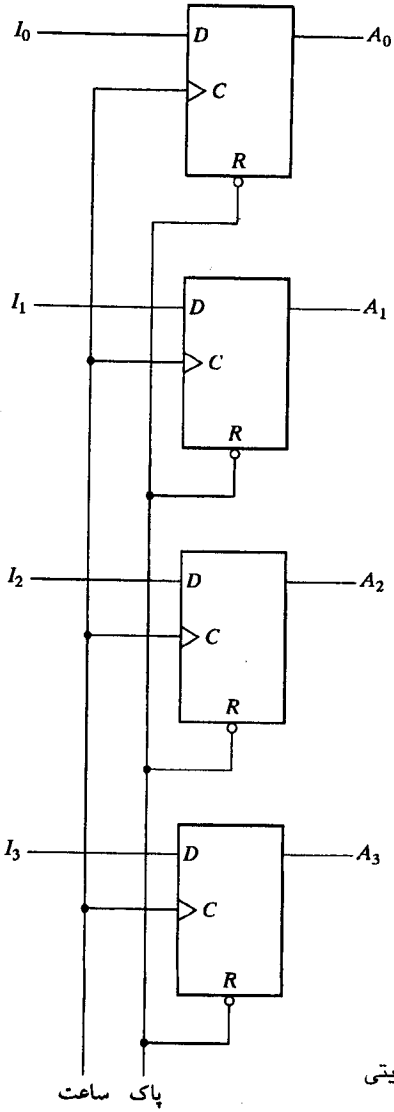
۶-۱ ثَبات‌ها

یک مدار ترتیبی ساعت‌دار متشکل از گروهی از فلیپ فلاپ‌ها و گیت‌های ترکیبی است که به منظور تشکیل یک مسیر پسخورد به هم متصل شده‌اند. فلیپ فلاپ‌ها عناصر ضروری مدار هستند زیرا در غیاب آنها، مدار به یک مدار ترکیبی محض تقلیل می‌یابد. (به شرطی که بین گیت‌ها هم مسیر پسخورد وجود نداشته باشد.) اما مداری با فلیپ فلاپ حتی در نبود گیت‌های ترکیبی باز هم یک مدار ترتیبی است. مدارهای حاوی فلیپ فلاپ‌ها معمولاً برحسب کارشان و نه با نام مدار ترتیبی دسته‌بندی می‌شوند. دو نوع از این مدارها ثبات‌ها و شمارنده‌ها هستند.

یک ثبات گروهی از فلیپ فلاپ‌هاست. هر فلیپ فلاپ قادر است یک بیت از اطلاعات را در خود ذخیره نماید. یک ثبات n بیت، مجموعه‌ای از n فلیپ فلاپ می‌باشد که قادر است n بیت از اطلاعات دودویی را در خود ذخیره نماید. علاوه بر فلیپ فلاپ، یک ثبات ممکن است گیت‌های ترکیبی را نیز برای اجرای کارهای پردازشی مختلف داشته باشد. در تعریف جامع‌تر، یک ثبات متشکل از یک گروه فلیپ فلاپ‌ها و گیت‌هاست که در عمل انتقال با یکدیگر تشریک مساعی دارند. فلیپ فلاپ‌ها اطلاعات دودویی را نگه می‌دارند و گیت‌ها چگونگی انتقال اطلاعات را به ثبات معین می‌کنند.

یک شمارنده اساساً یک ثبات است که وارد یک رشته از حالات از پیش تعیین شده می‌شود. گیت‌ها در شمارنده‌ها چنان به هم متصل شده‌اند تا رشته از پیش تعیین شده‌ای از حالات را تولید نمایند. هر چند که شمارنده‌ها نوع خاصی از ثبات می‌باشند، معمولاً آنها را با نام‌های متفاوت از ثبات‌ها جدا می‌کنند.

انواع متنوعی از ثبات‌ها در بازار وجود دارند. ساده‌ترین ثبات، فقط از فلیپ فلاپ‌ها و بدون هرگونه گیتی تشکیل شده است. شکل ۶-۱ چنین ثباتی را که از چهار فلیپ فلاپ D ساخته شده نشان می‌دهد. ساعت ورودی مشترک همه فلیپ فلاپ‌ها را با لبه مثبت هر پالس تریگر می‌کند و به این ترتیب



شکل ۱-۶. ثبات 4 بیتی

اطلاعات دودویی در چهار ورودی به داخل ثبات 4 بیت منتقل می‌گردند. می‌توان هر لحظه چهار خروجی را نمونه‌برداری کرده و اطلاعات دودویی ذخیره شده در ثبات را بدست آورد. ورودی پاک به ورودی باز نشان (R) همه فلیپ فلاپ‌ها می‌رود. وقتی این ورودی به 0 رود، همه فلیپ فلاپ‌ها به طور غیرهمزمان بازنشانی (0) می‌شوند. ورودی پاک کردن برای 0کردن ثبات قبل از عمل ساعت‌زنی مفید است. در حین عمل معمول ساعت زنی، ورودی‌های R باید در منطق 1 قرارگیرند. توجه کنید که برای 0 کردن همه حالات در یک ثبات، می‌توان از پاک کردن ، یا بازنشانی استفاده کرد.

ثبات با بار شدن موازی

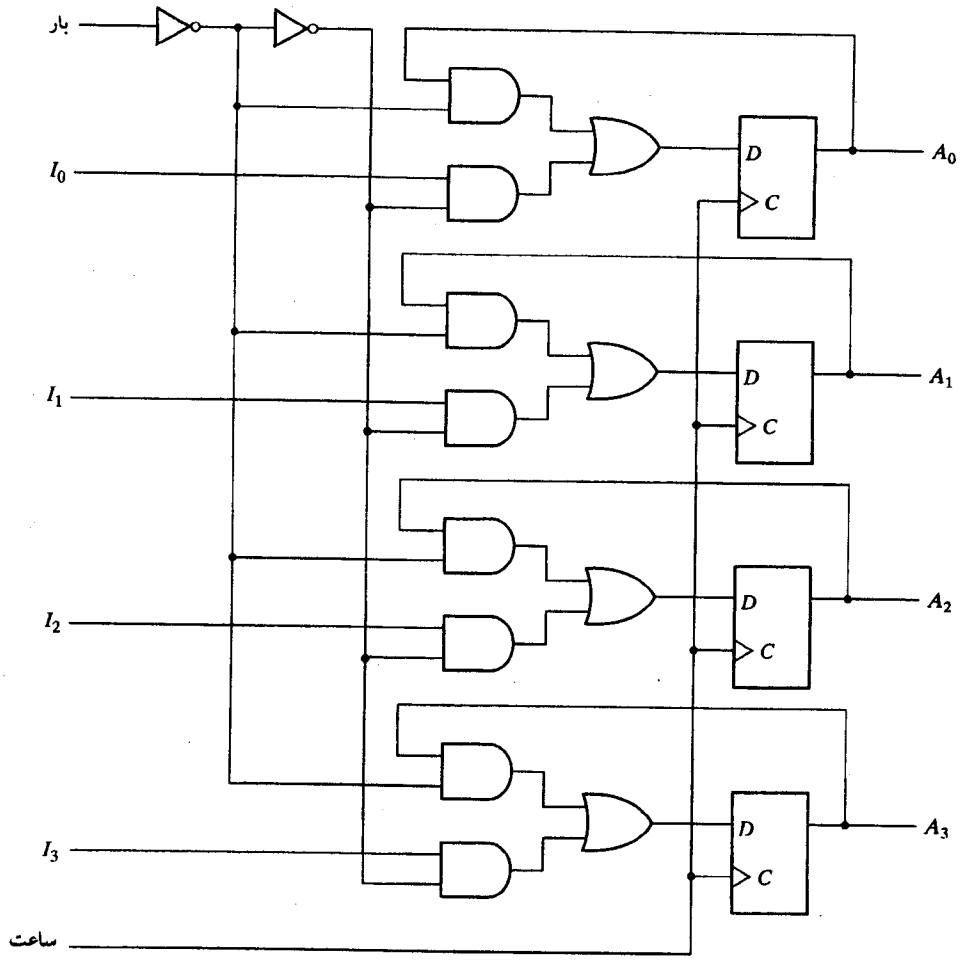
سیستم‌های دیجیتال همزمان دارای یک مولد ساعت اصلی‌اند که رشته‌ای از پالس‌های ساعت را به طور پیوسته فراهم می‌سازند. پالس‌های ساعت به همه فلیپ فلاپ‌ها و ثبات‌ها در سیستم اعمال می‌گردند. ساعت اصلی مانند پمپی است که ضربان ثابتی را برای همه بخش‌های سیستم فراهم می‌نماید. برای تأثیر یک پالس ساعت خاص بر روی یک ثبات خاص، باید یک کنترل جداگانه به کار برده شود. انتقال اطلاعات جدید به یک ثبات را بار شدن ثبات نامند. اگر همه بیت‌های ثبات به طور همزمان با یک پالس بار شوند گوییم بار شدن موازی است. لبه ساعت اعمال شده به ورودی‌های C ثبات شکل ۱-۶ موجب می‌شود تا هر چهار ورودی به طور موازی بار گردند. در این آرایش اگر بخواهیم ثبات بدون تغییر رها شود، باید ساعت از مدار قطع گردد. این کار با کنترل سیگنال ورودی ساعت به وسیله گیت فعال‌ساز انجام می‌شود. با این وجود قرار دادن گیت‌ها در مسیر ساعت به این معنی است که یک کار منطقی صورت گرفته است. استقرار گیت‌ها موجب تولید تأخیرهای نابرابر در ورودی فلیپ فلاپ‌ها می‌گردد. برای همزمانی کامل سیستم، باید مطمئن بود که همه پالس‌های ساعت به طور همزمان به هر نقطه از سیستم می‌رسد و بنابراین همه فلیپ فلاپ‌ها به طور همزمان تریگر می‌شوند. اعمال پالس ساعت از طریق گیت، تأخیرهای متغیری را موجب می‌شود و ممکن است سیستم را از همزمانی خارج کند. به این دلیل پیشنهاد می‌شود که کنترل عمل یک ثبات با ورودی‌های D به جای کنترل ساعت در ورودی‌های C فلیپ فلاپ‌ها انجام گیرد.

یک ثبات 4 بیتی با ورودی کنترل بار شدن که از طریق گیت‌ها به ورودی‌های D فلیپ فلاپ هدایت شده در شکل ۲-۶ ملاحظه می‌گردد. ورودی بار شدن به ثبات عملی را که در هر پالس ساعت اتفاق می‌افتد مشخص می‌کند. وقتی که ورودی بار برابر با 1 است، داده در چهار ورودی در لبه مثبت پالس ساعت بعدی به داخل ثبات منتقل می‌شود. وقتی ورودی بار شدن 0 است، خروجی‌های فلیپ فلاپ‌ها به ورودی‌های خودشان وصلند. اتصال پسخوردی از خروجی به ورودی لازم است زیرا فلیپ فلاپ D دارای حالت "بی‌تغییر" نیست. در هر لبه پالس ساعت، ورودی D حالت بعدی فلیپ فلاپ را مشخص می‌نماید. برای بدون تغییر نگهداشتن فلیپ فلاپ، لازم است D را با حالت فعلی فلیپ فلاپ یکسان نماییم.

پالس‌های ساعت مرتباً به ورودی‌های C اعمال می‌گردند. ورودی بار شدن، پذیرش اطلاعات جدید و یا حفظ اطلاعات فعلی را در ثبات معین می‌کند. انتقال اطلاعات از ورودی‌ها یا خروجی‌های ثبات در پاسخ به یک لبه ساعت به طور همزمان در هر چهار بیت انجام می‌گیرد.

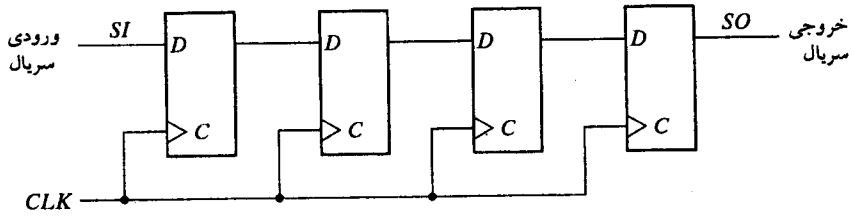
۲-۶ شیفت رجیسترها

ثباتی که بتواند اطلاعات دودویی‌اش را به سمت راست یا چپ شیفت یا جابجا کند، شیفت رجیستر یا ثبات جابجایی نامیده می‌شود. ساختار منطقی یک شیفت رجیستر، از زنجیره‌ای از فلیپ فلاپ‌ها تشکیل شده که در آن خروجی یک فلیپ فلاپ به ورودی فلیپ فلاپ دیگر متصل است. همه فلیپ فلاپ‌ها پالس ساعت مشترکی دریافت می‌کنند. پالس‌های ساعت اطلاعات را از یک طبقه به طبقه دیگر جابجا می‌کنند.



شکل ۲-۶. ثبات 4 بیتی با بار شدن موازی

ساده‌ترین شیفت رجیستر طبق شکل ۳-۶ فقط از فلیپ فلاپ‌ها استفاده می‌کند. خروجی یک فلیپ فلاپ مفروض به ورودی D فلیپ فلاپ سمت راست خود متصل است. هر پالس ساعت محتوای ثبات را یک بیت به راست جابجا می‌کند. ورودی سریال، تعیین کننده اطلاعاتی است که از منتهی‌الیه سمت چپ در حین جابجایی وارد می‌شود. خروجی سریال از خروجی سمت راست‌ترین فلیپ فلاپ اخذ می‌گردد. گاهی لازم است تا جابجایی را طوری کنترل کنیم که فقط با پالس‌های معینی رخ دهد. این کار با ممانعت از پالس ساعت در رسیدن به ثبات امکان‌پذیر است. بعد نشان خواهیم داد که عمل جابجایی می‌تواند از طریق ورودی‌های D به جای ورودی ساعت ثبات کنترل گردد. در هر صورت اگر شیفت رجیستر شکل ۳-۶ به کار رود، می‌توان عمل جابجایی را به وسیله یک گیت AND و ورودی که جابجایی را کنترل می‌کند تحت کنترل در آورد.

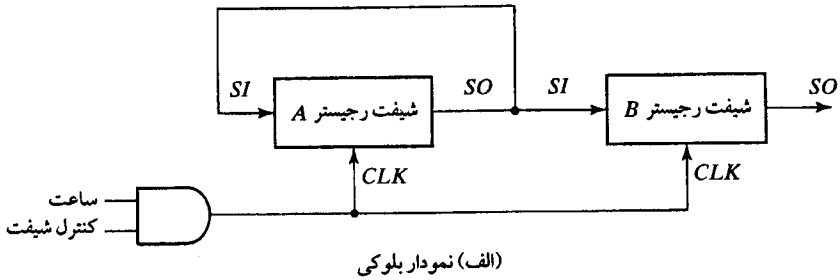


شکل ۳-۶. شیفت رجیستر 4 بیتی

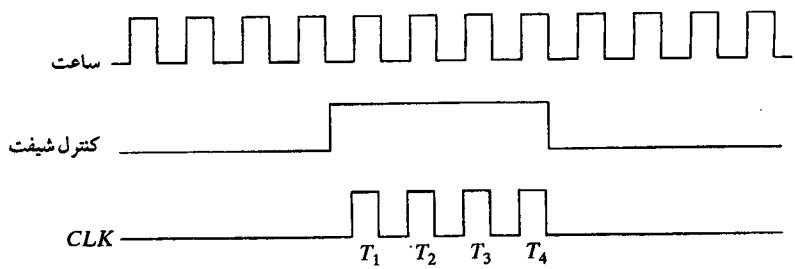
انتقال سریال

اگر یک سیستم دیجیتال هر بار یک بیت را انتقال دهد و یا دستکاری نماید، گوییم در مد سریال کار می‌کند. با جابجایی یک بیت به خارج ثبات مبدأ و ورود به ثبات مقصد، اطلاعات هر بار یک بیت انتقال می‌یابد. این برخلاف انتقال موازی است که در آن همه بیت‌های ثبات به طور همزمان انتقال می‌یابند.

انتقال سریال اطلاعات از ثبات A به ثبات B طبق نمودار بلوکی شکل ۴-۶ (الف) با شیفت رجیستر انجام می‌شود. خروجی سریال (SO) از ثبات A به ورودی سریال (SI) در ثبات B وصل است. برای پیشگیری در از دست دادن اطلاعات ذخیره شده در ثبات مبدأ، اطلاعات ثبات A از خروجی سریال به ورودی سریالش چرخانده می‌شود. در حین عمل جابجایی مقدار اولیه ثبات B به بیرون منتقل شده و از بین می‌رود، مگر این که به ثبات سومی انتقال یابد. ورودی کنترل جابجایی زمان و تعداد دفعاتی که ثبات‌ها جابجا می‌شوند را معین می‌سازد. این کار با یک گیت AND انجام می‌گردد و طی آن پالس‌های ساعت اجازه عبور به پایانه‌های CLK را به هنگام فعال بودن کنترل جابجایی خواهند داشت.



(الف) نمودار بلوکی



(ب) نمودار زمانی

شکل ۴-۶. انتقال سریال از ثبات A به ثبات B

جدول ۱-۶. مثال انتقال سریال

پالس زمانی	شیفت رجیستر **A**	شیفت رجیستر **B**
مقدار اولیه	1 0 1 1	0 0 1 0
پس از T_1	1 1 0 1	1 0 0 1
پس از T_2	1 1 1 0	1 1 0 0
پس از T_3	0 1 1 1	0 1 1 0
پس از T_4	1 0 1 1	1 0 1 1

فرض کنید که شیفت رجیسترها هر کدام دارای چهار بیت باشند. واحد کنترلی که انتقال را مدیریت می‌کند باید طوری طراحی شود که شیفت رجیسترها را در طول سیگنال کنترل جابجایی برای مدت چهار پالس ساعت فعال‌سازد. این مطلب در نمودار زمانبندی شکل ۴-۶ (ب) ملاحظه می‌شود. سیگنال کنترل جابجایی با ساعت همگام است و مقدارش درست پس از لبه منفی پالس ساعت تغییر می‌یابد. در چهار پالس ساعت بعدی سیگنال کنترل جابجایی فعال است و خروجی گیت AND متصل به ورودی‌های CLK چهار پالس T_1 ، T_2 ، T_3 و T_4 را تولید می‌نماید. هر لبه بالا رونده پالس یک جابجایی را در هر ثبات انجام می‌دهد. چهارمین پالس کنترل جابجایی را 0 نموده و موجب می‌شود تا شیفت رجیسترها غیرفعال شوند.

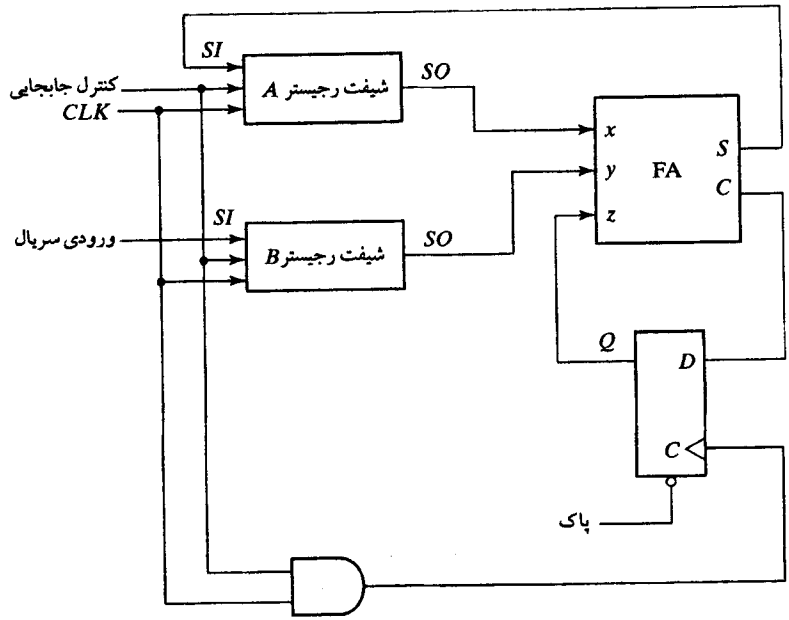
فرض کنید که محتوای دودویی A قبل از جابجایی 1011 و B برابر 0010 باشد. انتقال سریال از A به B در چهار مرحله رخ می‌دهد، جدول ۱-۶. با اولین پالس، T_1 ، سمت راست‌ترین بیت A به سمت چپ‌ترین بیت B منتقل می‌گردد و نیز به سمت چپ‌ترین بیت A می‌چرخد. در همان زمان تمام بیت‌های A و B یک مکان به راست جابجا می‌شوند. خروجی سریال قبلی از B در سمت راست‌ترین مکان از بین رفته و مقدار آن از 0 به 1 تبدیل می‌گردد. سه پالس بعدی اعمال مشابهی را انجام می‌دهند و بیت‌های A و B را هر بار یک بیت به راست جابجا می‌کنند. پس از چهارمین جابجایی، کنترل جابجایی به 0 رفته و هر دو ثبات A و B دارای مقدار 1011 خواهند بود. بنابراین محتوای A به B منتقل شده است، ضمن این که A همچنان بدون تغییر باقی می‌ماند.

با توجه به این مثال تفاوت بین مدهای سریال و موازی کاملاً آشکار است. در مد موازی، اطلاعات همه بیت‌های ثبات در دسترس است و همگی می‌توانند با یک پالس ساعت به طور همزمان انتقال یابند. در مد سریال ثبات‌ها دارای یک ورودی سریال و یک خروجی سریال هستند، اطلاعات هر بار یک بیت انتقال می‌یابد و ثبات‌ها در یک جهت جابجا می‌شوند.

جمع سریال

عملیات در کامپیوترهای دیجیتال معمولاً به صورت موازی صورت می‌گیرد زیرا این روش سریع‌ترین نوع است. عملیات سریال کندتر است، ولی به قطعات کمتری نیاز دارد. برای ارائه مد سریال، در اینجا یک جمع‌کننده سریال را نشان می‌دهیم. نوع موازی آن در بخش ۴-۴ مشاهده شد.

دو عددی که قرار است بطور سریال با هم جمع شوند در دو شیفت رجیستر ذخیره می‌شوند. بیت‌ها،



شکل ۵-۶. جمع کننده سریال

هر جفت یک بار به وسیله یک جمع‌کننده کامل سریال (FA)، با هم جمع می‌شوند، شکل ۵-۶. نقلی خروجی جمع‌کننده کامل به فلیپ فلاپ D انتقال می‌یابد. آنگاه خروجی این فلیپ فلاپ به عنوان نقلی ورودی به جفت بیت باارزش‌تر بعدی اضافه می‌شود. با انتقال حاصل جمع به A، که با جابجایی A انجام می‌گردد، می‌توان از یک ثبات برای هر دو مقدار مضاف‌الیه و حاصل جمع استفاده کرد. ورودی سریال ثبات B می‌تواند برای انتقال یک مقدار دودویی جدید به کار رود و در همان زمان بیت‌های مضاف در حین جمع، به خارج جابجا می‌شوند.

طرز کار یک جمع‌کننده سریال به شرح زیر است. در ابتدا ثبات A مقدار مضاف‌الیه و ثبات B مقدار مضاف را نگه می‌دارند، ضمن این که فلیپ فلاپ نقلی به 0 پاک شده است. خروجی Q از فلیپ فلاپ نقلی ورودی در z را تهیه می‌کند. کنترل جابجایی هر دو ثبات و فلیپ فلاپ نقلی را فعال می‌سازد، به نحوی که در پالس ساعت بعدی، هر دو ثبات یک بار به راست جابجا می‌شوند، بیت حاصل جمع از S به سمت چپ فلیپ فلاپ A می‌رود و رقم نقلی به فلیپ فلاپ Q منتقل خواهد شد. کنترل جابجایی ثبات‌ها را به تعداد پالس‌هایی برابر با تعداد بیت‌های ثبات‌ها فعال می‌کند. در قبال هر یک پالس ساعت جدید، یک بیت حاصل جمع جدید به A می‌رود. یک نقلی جدید به Q رفته و هر دو ثبات یک بار به سمت راست جابجا می‌گردند. این روند تا از کار افتادن کنترل جابجایی ادامه می‌یابد. بنابراین، جمع با عبور هر جفت بیت به همراه نقلی قبلی از یک جمع‌کننده کامل و انتقال حاصل جمع به ثبات A، در هر بار یک بیت، انجام می‌گردد.

در آغاز، ثبات A در فلیپ فلاپ نقلی به 0 پاک می‌شود و سپس اولین عدد از B به آن اضافه می‌گردد. ضمن جابجایی B به جمع‌کننده، دومین عدد از طریق ورودی سریال وارد می‌شود. این عدد به محتوای ثبات A اضافه می‌شود، و در همان هنگام سومین عدد وارد ثبات می‌گردد. این کار می‌تواند برای تشکیل جمع دو، سه یا چند عدد و ذخیره حاصل جمع در ثبات A تکرار شود.

از مقایسه جمع‌کننده سریال با موازی بخش ۴-۴، چندین تفاوت ملاحظه می‌گردد. جمع‌کننده موازی از ثبات‌هایی با امکان بار شدن موازی استفاده می‌کند، در حالی که جمع‌کننده سری شیفت رجیسترها را به کار می‌برد. تعداد جمع‌کننده‌های مدار موازی برابر تعداد بیت‌های اعداد دودویی است، در صورتی که جمع‌کننده سریال از یک جمع‌کننده کامل و یک فلیپ فلاپ برای ذخیره نقلی و خروجی استفاده می‌نماید. دلیل ذخیره نقلی این است که در اعمال سریال نتیجه یک جمع بیتی در هر زمان نه تنها به ورودی‌های فعلی بلکه به ورودی‌های قبلی که باید در فلیپ فلاپ‌ها ذخیره شوند نیز بستگی دارد.

برای نمایش روش طراحی اعمال سریال با مدارهای ترتیبی دوباره جمع‌کننده سریال را با استفاده از جدول حالت طراحی می‌کنیم. ابتدا فرض می‌نماییم که دو شیفت رجیستر برای ذخیره اعدادی که قرار است با هم به طور سریال جمع شود موجود باشد. خروجی‌های سریال از ثبات‌ها را x و y می‌نامیم. مدار ترتیبی موردنظر در حال حاضر فاقد شیفت رجیستر است ولی هنگام تکمیل آن به مدار اضافه خواهد شد. مدار ترتیبی دارای دو ورودی x و y است که دو بیت با ارزشتر دو عدد را برای مدار تهیه می‌کنند، یک خروجی S که بیت حاصل جمع را تولید می‌نماید و یک فلیپ فلاپ Q برای ذخیره رقم نقلی است. جدول حالتی که مدار ترتیبی را تعریف می‌کند در جدول ۲-۶ نشان داده شده است. حالت فعلی Q مقدار فعلی نقلی است. نقلی فعلی در Q با ورودی‌های x و y جمع می‌شود تا بیت جمع را در خروجی S تولید نماید. حالت بعدی Q برابر نقلی فعلی است. توجه کنید که وارده‌های جدول حالت، با وارده‌های جدول جمع‌کننده کامل یکی است جز این که نقلی ورودی اکنون حالت فعلی Q ، و نقلی خروجی اکنون حالت بعدی آنست .

جدول ۲-۶. جدول حالت یک جمع کننده سریال

حالت فعلی	ورودی ها		حالت بعدی	خروجی	ورودی‌های فلیپ فلاپ	
Q	X	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

اگر از فلیپ فلاپ D برای Q استفاده شود، مدار به شکل ۵-۶ کاهش می‌یابد. اگر برای Q از JK استفاده نماییم، لازم است مقادیر ورودی‌های J و K را با ارجاع به جدول تحریک (جدول ۱۲-۵) معین کنیم. این کار در دو ستون آخر جدول ۲-۶ انجام شده است. دو معادله ورودی فلیپ فلاپ و معادله خروجی با نقشه به صورت زیر ساده می‌شوند.

$$J_Q = xy$$
$$K_Q = x'y' = (x + y)'$$
$$S = x \oplus y \oplus Q$$

نمودار مدار در شکل ۶-۶ نشان داده شده است. مدار متشکل از سه گیت و یک فلیپ فلاپ JK می‌باشد. برای تکمیل جمع‌کننده سریال دو شیفت رجیستر هم به آن اضافه شده است. توجه کنید که خروجی S نه فقط تابعی از x و y است، بلکه تابعی از Q هم می‌باشد. حالت بعدی Q تابعی از حالت فعلی Q و مقادیر x و y رسیده از خروجی‌های شیفت رجیسترهاست.

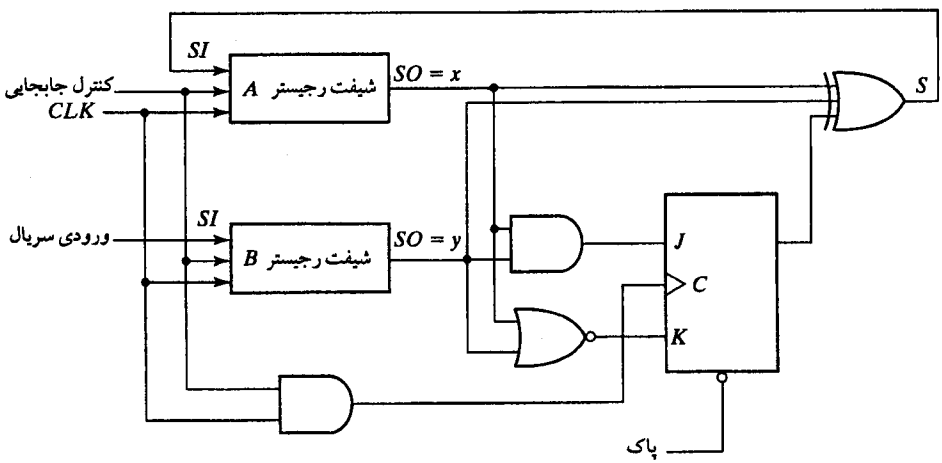
شیفت رجیسترهای یونیورسال

اگر خروجی فلیپ فلاپ‌های یک شیفت رجیستر قابل دسترسی باشد، آنگاه می‌توان اطلاعات وارده سریال را با جابجایی از خروجی فلیپ فلاپ‌ها به صورت موازی خارج کرد. اگر به شیفت رجیستر یک قابلیت بار شدن موازی اضافه شود، آنگاه داده وارده موازی به ثبات را می‌توان با جابجایی به صورت سریال خارج کرد.

بعضی از شیفت رجیسترها پایانه‌های لازم را برای انتقال موازی دارا هستند. این مدارها ممکن است قابلیت جابجایی به چپ و راست را هم داشته باشند. عمومی‌ترین شیفت رجیستر دارای امکانات زیر است:

۱- کنترل پاک برای پاک کردن ثبات به 0.

۲- ورودی ساعت برای همزمانی اعمال.



شکل ۶-۶. فرم دوم یک جمع کننده سریال

۳- کنترل جابجایی به راست برای فعال کردن عمل جابجایی به راست و خطوط ورودی و خروجی سریال مربوط به جابجایی به راست.

۴- کنترل جابجایی به چپ برای فعال کردن عمل جابجایی به چپ و خطوط ورودی و خروجی سریال مربوط به جابجایی به چپ.

۵- یک کنترل بارکردن موازی برای فعال کردن انتقال موازی و n خط ورودی مربوط به انتقال موازی.

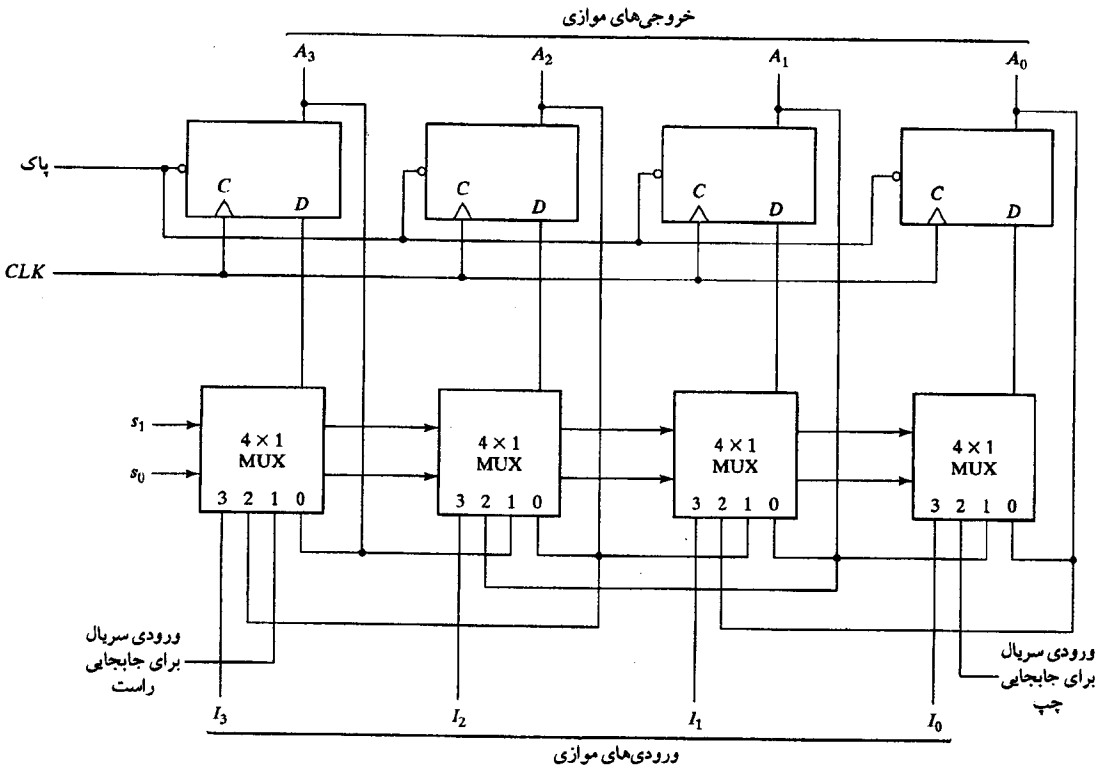
۶- n خط خروجی موازی.

۷- حالت کنترلی که علیرغم وجود پالس ساعت اطلاعات را در ثبات بدون تغییر نگه می‌دارد.

دیگر شیفت رجیسترها ممکن است بعضی از امکانات فوق را با حداقل یک عمل جابجایی یا شیفت داشته باشد.

ثباتی که فقط قادر به جابجایی داده در یک جهت باشد را شیفت رجیستر یک جهته گویند. اگر در دو جهت جابجا نماید آن را شیفت رجیستر دو جهته می‌نامند. اگر ثبات قادر به جابجایی دو جهته و بار شدن موازی باشد به آن شیفت رجیستر یونیورسال گویند.

نمودار یک شیفت رجیستر 4 بیت یونیورسال با همه فلیپ فلاپ‌های فوق‌الذکر در شکل ۷-۶ نشان داده شده است. این مدار از چهار فلیپ فلاپ D و چهار مولتی پلکسر ساخته شده است. چهار مولتی پلکسر



شکل ۷-۶. شیفت رجیستر یونیورسال 4 بیتی

جدول ۳-۶. جدول عملکرد ثبات شکل (۷-۶)

عملکرد ثبات	وضعیت کنترل	
	S_0	S_1
بلاتغییر	0	0
شیفت - راست	1	0
شیفت - چپ	0	1
بارکردن موازی	1	1

دو ورودی انتخاب مشترک S_1 و S_0 دارند. ورودی 0 در هر مولتی پلکسر وقتی انتخاب می‌شود که $S_1S_0=00$ باشد، ورودی 1 با $S_1S_0 = 01$ انتخاب می‌گردد، و به طور مشابه دو ورودی باقیمانده‌انتخاب می‌گردند. ورودی‌های انتخاب مد عملیات ثبات را طبق وارده‌های جدول (۳-۶) کنترل می‌کنند. وقتی $S_1S_0 = 00$ است، مقدار فعلی ثبات به ورودی‌های D از فلیپ فلاپ‌ها اعمال می‌گردد. این وضعیت مسیری را از خروجی هر فلیپ فلاپ به ورودی‌اش ایجاد می‌نماید. لبه ساعت بعدی، مقداری را که از قبل در آن ذخیره شده وارد فلیپ فلاپ می‌کند و بنابراین هیچ تغییری در حالت رخ نمی‌دهد. وقتی $S_1S_0 = 01$ است، پایه ورودی 1 از مولتی پلکسر دارای مسیری به ورودی‌های D فلیپ فلاپ‌هاست. این موجب عمل جابجایی به راست می‌گردد، که در آن ورودی سریال به فلیپ فلاپ A_3 وارد می‌شود. وقتی $S_1S_0 = 10$ است، یک عمل جابجایی به چپ صورت می‌گیرد و طی آن دیگر ورودی به فلیپ فلاپ A_0 خواهد رفت. بالاخره وقتی $S_1S_0 = 11$ است، اطلاعات دودویی روی خطوط ورودی موازی به طور همزمان در لبه پالس بعدی وارد ثبات می‌گردند.

شیفت رجیسترها اغلب برای اتصال و ارتباط سیستم‌های دیجیتالی که با فواصل دوری از یکدیگر قرار دارند به کار می‌روند. مثلاً فرض کنید که بخواهیم کمیتی n بیتی را بین دو نقطه جابجا کنیم. اگر فاصله زیاد باشد، استفاده از n خط موازی گران تمام می‌شود. استفاده از یک خط و انتقال سریال اطلاعات به صورت یک بیت در هر بار اقتصادی‌تر است. فرستنده داده n بیتی را به صورت موازی وارد شیفت رجیستر کرده و داده را به صورت سریال در طول خط ارسال می‌دارد. گیرنده داده را به طور سریال وارد یک شیفت رجیستر می‌نماید. وقتی هر n بیت دریافت شد، می‌توان از خروجی‌های ثبات آنها را به صورت موازی دریافت کرد. بنابراین فرستنده یک عمل تبدیل موازی به سریال داده و گیرنده یک عمل تبدیل سریال به موازی را انجام می‌دهد.

۳-۶ شمارنده‌های موج‌گونه

ثباتی که براساس اعمال پالس‌های ورودی وارد رشته حالات از پیش تعیین شده‌ای می‌گردد، شمارنده نام دارد. پالس‌های ورودی ممکن است پالس‌های ساعت و یا از یک منبع بیرونی با توالی ثابت و یا متغیر باشند. رشته حالات ممکن است رشته اعداد دودویی و یا رشته حالات دیگری باشد. شمارنده‌ای که رشته اعداد دودویی را دنبال می‌کند، شمارنده دودویی نامیده می‌شود. یک شمارنده n بیتی متشکل

از n فلیپ فلاپ بوده و می‌تواند از 0 تا 2^n-1 را بشمارد.

شمارنده‌ها به دو صورت وجود دارند: شمارنده‌های موج‌گونه و شمارنده‌های همزمان. در یک شمارنده موج‌گونه، تغییر وضعیت خروجی فلیپ فلاپ به عنوان منبع تریگر کردن دیگر فلیپ فلاپ‌ها عمل می‌کند. به بیان دیگر، ورودی C بعضی از و یا همه فلیپ فلاپ‌ها با پالس‌های ساعت مشترکی تریگر یا راه‌اندازی نمی‌شوند. برعکس در شمارنده همزمان، ورودی‌های C همه فلیپ فلاپ‌ها ساعت مشترکی، را دریافت می‌نمایند. شمارنده‌های همزمان در دو بخش بعد ارائه شده‌اند. در اینجا شمارنده‌های موج‌گونه BCD و دودویی را ارائه کرده و نحوه کار آنها را توضیح می‌دهیم.

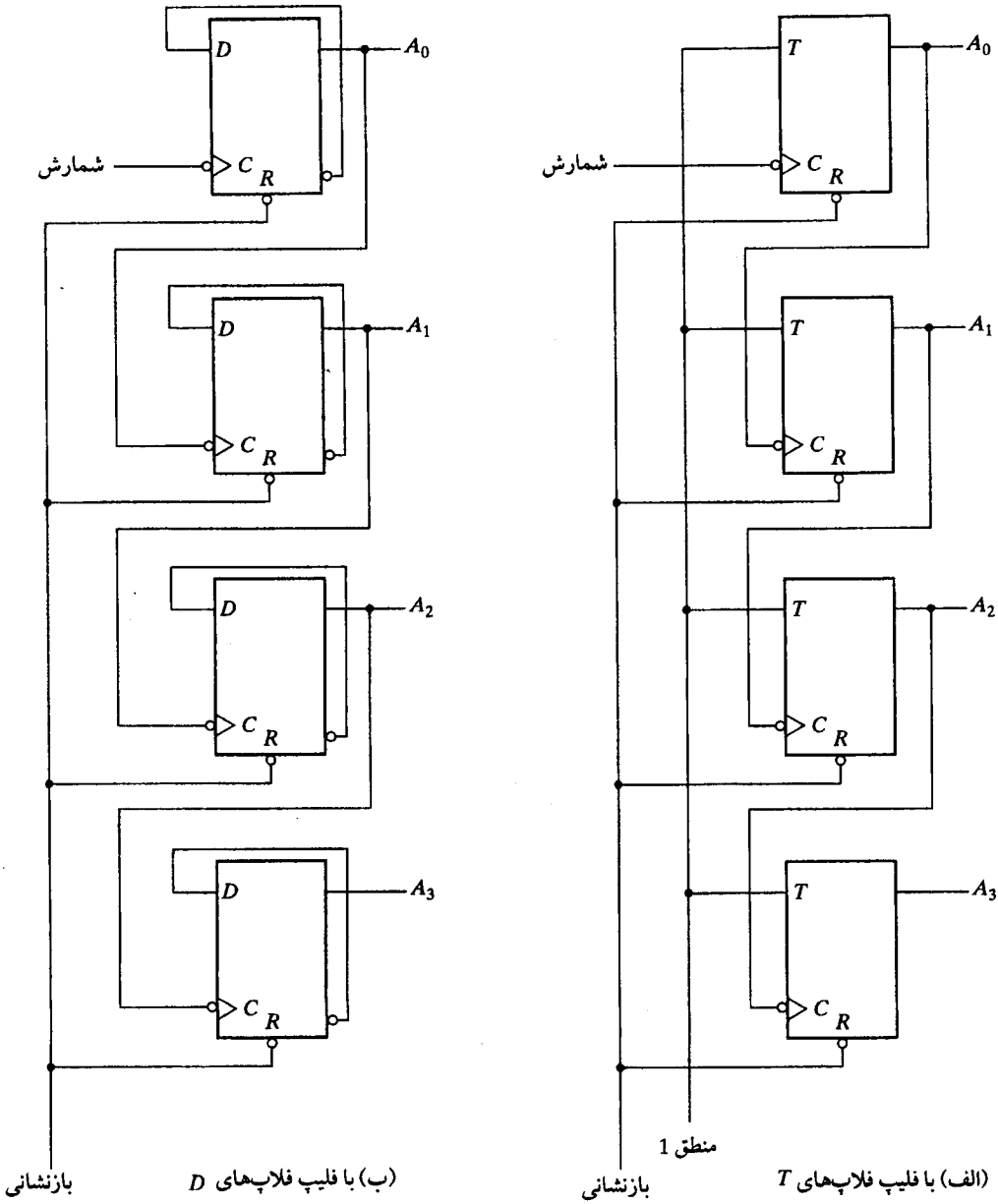
شمارنده موج‌گونه دودویی

یک شمارنده موج‌گونه دودویی از یک سری اتصال بین فلیپ فلاپ‌های متمم‌ساز تشکیل شده است. که خروجی هر فلیپ فلاپ به ورودی C فلیپ فلاپ مرتبه بالاتر وصل است. فلیپ فلاپی که کم‌ارزش‌ترین بیت را نگه می‌دارد، پالس‌های مورد شمارش را دریافت می‌کند. فلیپ فلاپ متمم‌ساز را می‌توان با یک فلیپ فلاپ JK که در آن J و K به هم وصل‌اند و یا از یک فلیپ فلاپ T ساخت. سومین امکان استفاده از فلیپ فلاپ D است که در آن خروجی متمم به ورودی D وصل است. به این ترتیب، ورودی D همواره متمم حالت فعلی بوده و پالس ساعت بعدی موجب متمم شدن خروجی اصلی آن خواهد شد. نمودار منطقی دو شمارنده دودویی 4 بیت در شکل ۸-۶ نشان داده شده است. شمارنده با فلیپ فلاپ‌های متمم‌ساز نوع T در بخش (الف) و نوع D در بخش (ب) ساخته شده است. خروجی هر فلیپ فلاپ به ورودی فلیپ فلاپ بعدی در رشته متصل است. همانطور که گفته شد فلیپ فلاپی که کم‌ارزش‌ترین بیت را نگه می‌دارد پالس‌های شمارش را دریافت می‌کند. ورودی‌های T همه فلیپ فلاپ‌ها در (الف) به طور دایم به منطق 1 متصل‌اند. این شرایط موجب می‌شود تا با گذر منفی در ورودی C فلیپ فلاپ متمم شود. حباب جلو نشانه‌گر دینامیک (◁) در کنار C به این معنی است که فلیپ فلاپ‌ها به لبه منفی ورودی واکنش نشان می‌دهند. گذر منفی هنگامی رخ می‌دهد که خروجی فلیپ فلاپ قبل که به C وصل است از 1 به 0 برود.

برای درک عملکرد شمارنده دودویی 4 بیت، به 9 عدد دودویی اول در جدول (۴-۶) مراجعه کنید.

جدول ۴-۶. رشته شمارش دودویی

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0



شکل ۸-۶. شمارنده موج‌گونه دودویی 4 بیتی

شمارش از 0 دودویی شروع و با هر پالس در ورودی افزایش می‌یابد. پس از شماره 15 شمارنده برای تکرار به 0 باز می‌گردد. بیت کم‌ارزش‌تر A_0 با هر پالس شمارش ورودی متمم می‌شود. هر بار که A_0 از 1

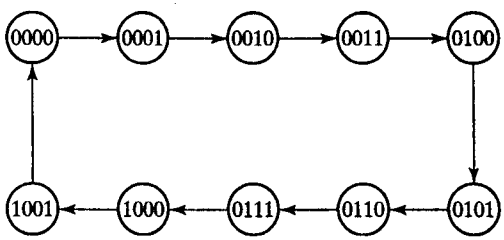
به 0 برود، A_1 را متمم می‌سازد. هر بار که A_1 از 1 به 0 برود، A_2 را متمم می‌نماید. هر بار که A_2 از 1 به 0 برود، A_3 را متمم می‌کند. و به همین ترتیب بیت‌های بالاتر در شمارنده موج‌گونه تغییر می‌کنند. به عنوان مثال، گذر از 0011 به 0100 را در نظر بگیرید. A_0 با پالس ساعت متمم می‌شود. چون A_0 از 1 به 0 می‌رود، A_1 تریگر شده و متمم می‌گردد. در نتیجه A_1 از 1 به 0 می‌رود که به نوبه خود موجب متمم شدن A_2 گشته و آن را از 0 به 1 خواهد برد. A_2 نمی‌تواند A_3 را تریگر کند زیرا A_2 یک گذر مثبت را تولید می‌کند و فلیپ فلاپ هم تنها به گذر منفی واکنش نشان می‌دهد.

بنابراین شمارش از 0011 به 0100 با تغییر نوبتی بیت‌ها رخ می‌دهد، به طوری که شمارنده از 0011 به 0010، سپس به 0000 و بالاخره به 0100 خواهد رفت. هر بار یکی از فلیپ فلاپ‌ها تغییر کرده و تغییر به پیش می‌رود و انتشار سیگنال در شمارنده از یک طبقه به طبقه دیگر مثل حرکت موج می‌ماند.

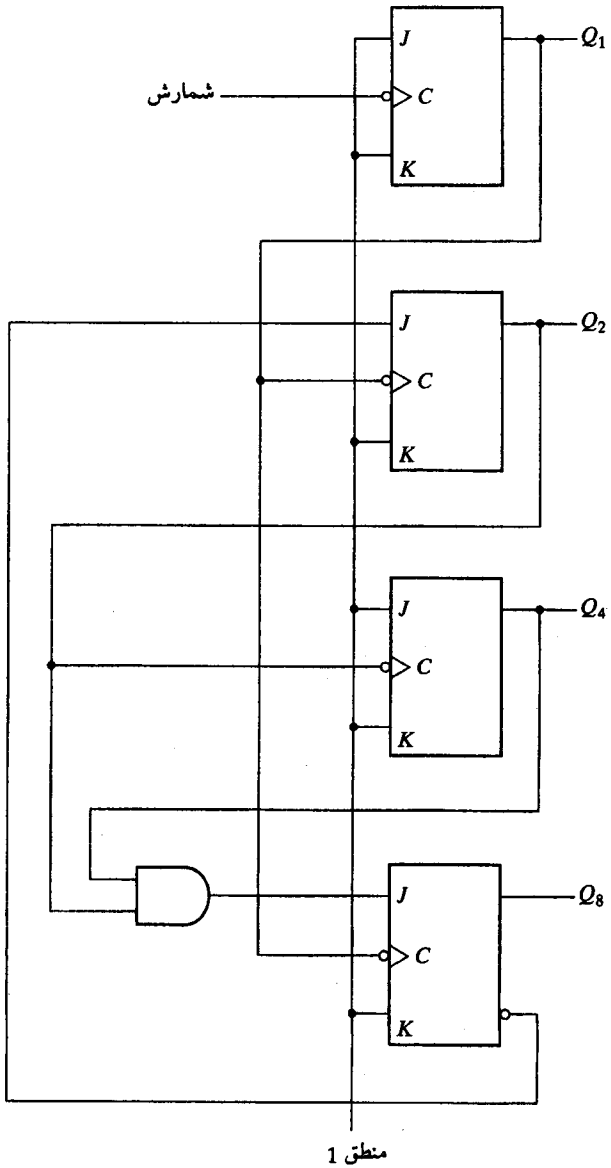
یک شمارنده دودویی با شمارش معکوس را پایین شمارگویند. در پایین شمار، شمارش با هر ورودی پالس شمارش، یک واحد کم می‌شد. شمارش یک پایین شمار 4 بیت از 15 شروع و به صورت 14، 13، 12، ... ,0 پایان یافته و سپس به 15 باز می‌گردد. لیستی از رشته شمارش یک شمارنده پایین شمار نشان می‌دهد که کم‌ارزش‌ترین بیت با هر پالس شمارش متمم شده است. هر بیت دیگر در رشته، اگر بیت کم‌ارزش‌تر قبل از آن از 0 به 1 برود، متمم می‌گردد. بنابراین نمودار یک پایین‌شمار مشابه شکل ۸-۶ خواهند بود، به شرطی که همه فلیپ فلاپ‌ها با لبه مثبت ساعت تریگر شوند. (حباب در ورودی C باید حذف شود.) اگر از فلیپ فلاپ‌های حساس به لبه منفی استفاده شود، آنگاه ورودی C هر فلیپ فلاپ باید به خروجی متمم فلیپ فلاپ قبلی وصل گردد. آنگاه وقتی که خروجی غیر متمم از 0 به 1 برود، متمم از 1 به 0 رفته و فلیپ فلاپ بعدی را آن طور که باید متمم خواهد کرد.

شمارنده BCD موج‌گونه

یک شمارنده دهدهی رشته‌ای از ده حالت را دنبال کرده و پس از 9 به 0 باز می‌گردد. چنین شمارنده‌ای باید حداقل چهار فلیپ فلاپ برای نمایش هر رقم دهدهی داشته باشد، زیرا یک رقم دهدهی با کد چهار بیتی نشان داده می‌شود. رشته حالات در شمارنده دهدهی به وسیله کد دودویی مربوطه برای نمایش هر رقم دیجیتال معین می‌گردد. اگر BCD به کار رود، رشته حالات مطابق نمودار حالت شکل ۹-۶ خواهد بود. این جدول مشابه با جدول دودویی است. به جز این که پس از 1001 برای عدد دهدهی 9، 0000 را برای رقم دهدهی 0 خواهیم داشت.

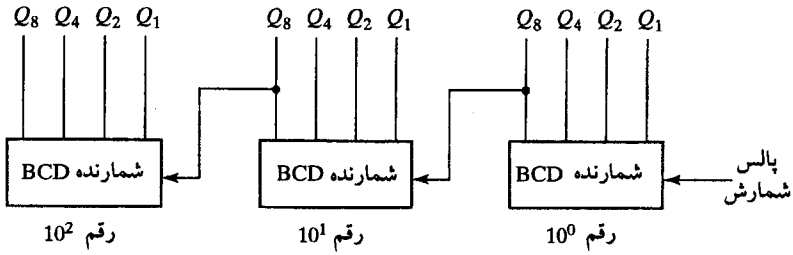


شکل ۹-۶. نمودار حالت یک شمارندهٔ دهدهی BCD



شکل ۱۰-۶. شمارندهٔ موج گونهٔ BCD

نمودار منطقی یک شمارنده BCD موج‌گونه با استفاده از فلیپ فلاپ JK در شکل ۱۰-۶ دیده می‌شود. چهار خروجی با حروف Q و اندیسی در زیر آن برای مشخص کردن وزن آن در BCD علامت‌گذاری شده است. توجه کنید که خروجی Q_1 به ورودی‌های C در هر دو ورودی Q_2 و Q_8 اعمال شده است و خروجی Q_2 هم به ورودی C از Q_4 وصل است. ورودی‌های J و K یا دائماً به 1 وصلند و یا به خروجی



شکل ۱۱-۶. نمودار بلوکی یک شمارنده دهدهی BCD با سه دهه

فلیپ فلاپ‌های دیگر وصل شده‌اند. شمارنده موج‌گونه یک مدار غیرهمزمان است. سیگنال‌ها بسته به ترتیبی که در آن از 1به 0 می‌روند، روی فلیپ فلاپ‌ها اثر می‌گذارند. عمل یک شمارنده با لیستی از حالات گذر هر فلیپ فلاپ قابل تفسیر است. این حالات از نمودار منطقی و دانستن چگونگی عملکرد یک فلیپ فلاپ JK حاصل می‌شود. به خاطر بسپارید وقتی که ورودی C از 1 به 0 می‌رود، اگر $J = 1$ باشد، فلیپ فلاپ 1 می‌شود و اگر $K = 1$ باشد به 0 پاک می‌گردد، و نیز اگر $J = K = 1$ باشد متمم شده و بالاخره با $J = K = 0$ حالت فلیپ فلاپ بی‌تغییر خواهد بود.

برای تحقیق و اطمینان از این که این حالات به ترتیب در شمارنده BCD رخ می‌دهند باید مطمئن شویم که گذر حالات فلیپ فلاپ‌ها رشته‌ای راکه به وسیله نمودار حالت شکل ۹-۶ مشخص شده دنبال می‌کند. حالت Q_1 پس از هر پالس ساعت عوض می‌شود. هر بار Q_1 از 1 به 0 برود و $Q_8 = 0$ باشد Q_2 متمم می‌شود. وقتی $Q_8 = 1$ شود، Q_2 در 0 می‌ماند. هر بار Q_2 از 1 به 0 برود Q_4 متمم می‌گردد. مادامی که Q_2 و Q_4 در 0 باشند، Q_8 در 0 خواهد ماند. وقتی هر دو Q_2 و Q_4 برابر 1 شوند، با تغییر 1 به 0 خروجی Q_1، خروجی Q_2 متمم می‌شود. با گذر بعدی Q_1، Q_8 پاک می‌شود.

شمارنده BCD شکل ۱۰-۶ یک شمارنده دهدهی است، زیرا از 0 تا 9 می‌شمارد. برای شمارش از 0 تا 99 دو شمارنده دهدهی لازم داریم. شمارش 0 تا 999 سه شمارنده دهدهی لازم دارد. شمارنده‌های دهدهی چند رقمی با اتصال سری شمارنده‌های BCD ساخته می‌شوند که هر کدام برای یک دهه است. یک شمارنده دهدهی سه رقمی در شکل ۱۱-۶ دیده می‌شود. ورودی‌ها به دومین و سومین دهه از Q_8 دهه قبل وارد می‌شوند. وقتی Q_8 در یک دهه از 1 به 0 می‌رود، شمارنده دهه بالاتر را تریگر می‌کند، ضمن این که خودش از 9 به 0 باز می‌گردد.

۶-۴ شمارنده‌های همزمان

شمارنده‌های همزمان در اعمال پالس ساعت به ورودی فلیپ فلاپ‌ها با شمارنده‌های موج‌گونه تفاوت دارند. یک ساعت مشترک همه فلیپ فلاپ‌ها را به طور همزمان تریگر می‌کند در صورتی که در نوع شمارنده‌های موج‌گونه هر بار فقط یک فلیپ فلاپ تریگر می‌شود. تصمیم بر متمم شدن یک فلیپ فلاپ از مقادیر داده‌های ورودی مانند T و J و K در لبه ساعت معین می‌شود. اگر $T=0$ یا $J=K=0$

باشد، حالت فلیپ فلاپ تغییر نمی‌نماید. اگر $T = 1$ یا $J = K = 1$ باشد، فلیپ فلاپ متمم می‌گردد.

روال طراحی شمارنده‌های همزمان در بخش ۷-۵ ارائه شد و طراحی شمارنده دودویی 3 بیت با توجه به شکل ۳۱-۵ انجام شد. در این بخش، تعدادی از مدارهای همزمان و عملکرد آنها را توضیح می‌دهیم.

شمارنده دودویی

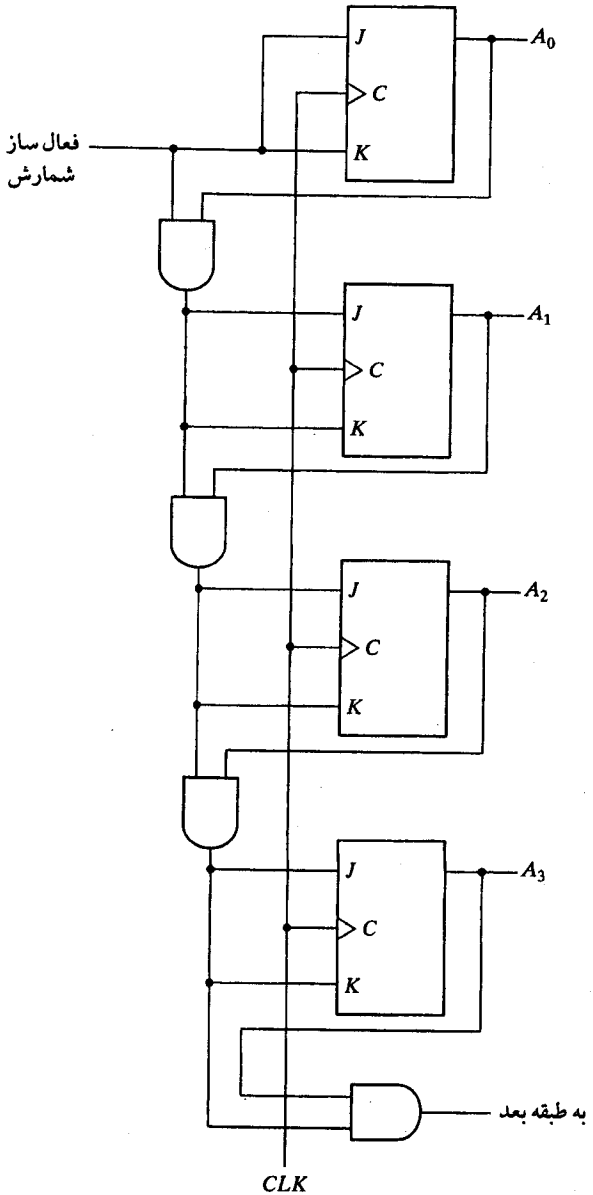
طراحی یک شمارنده دودویی آن قدر ساده است که نیازی به پیگیری مراحل طراحی را ندارد. در شمارنده دودویی همزمان، فلیپ فلاپ واقع در کم‌ارزش‌ترین مکان با هر پالس یکبار متمم می‌شود. فلیپ فلاپ‌های واقع در هر مکان هنگامی متمم می‌شود که همه فلیپ فلاپ‌های پایین‌تر 1 باشند. مثلاً اگر حالت فعلی یک شمارنده 4 بیت $A_3A_2A_1A_0 = 0011$ باشد، شماره بعدی 0100 خواهد بود. لازم به یادآوری است که در مثال فوق A_0 مرتباً متمم می‌شود. A_1 هنگامی متمم می‌گردد که A_0 برابر 1 باشد. A_2 هنگامی 1 می‌شود که $A_1A_0 = 11$ باشد. با این وجود A_3 متمم نمی‌شود زیرا حالت فعلی $A_2A_1A_0 = 011$ است، چون حالت تمام 1 وجود ندارد.

شمارنده‌های دودویی همزمان الگوی منظمی دارند و می‌توان آنها را با متمم کردن فلیپ فلاپ‌ها و گیت‌ها ساخت. نظم الگو را می‌توان با توجه به شکل ۱۲-۶ ملاحظه کرد. ورودی‌های C همه فلیپ فلاپ‌ها به ساعت مشترکی وصل‌اند. شمارنده با ورودی فعال‌ساز شمارش، فعال می‌گردد. اگر ورودی فعال‌ساز 0 باشد، ورودی همه Jها و Kها برابر 0 خواهند بود و بنابراین ساعت قادر نخواهد بود حالت شمارنده را عوض کند. در اولین طبقه، A_0، اگر شمارنده فعال شود $J = K = 1$ خواهد بود. در دیگر طبقات، Jها و Kها به شرطی 1 هستند که همه طبقات کم‌ارزش‌تر آنها برابر 1 و ورودی شمارش هم فعال شده باشد. در هر طبقه، زنجیره گیت‌های AND منطق لازم را برای ورودی‌های J و K فراهم می‌کنند. شمارنده را می‌توان به هر تعداد از طبقات گسترش داد که در آن هر طبقه یک گیت AND و یک فلیپ فلاپ اضافی خواهد داشت و هرگاه همه فلیپ فلاپ‌های طبقات قبل 1 شوند خروجی AND برابر با 1 خواهد بود.

توجه داشته باشید که فلیپ فلاپ‌ها در لبه مثبت ساعت تریگر می‌شوند. قطبیت ساعت، آنطور که در شمارنده‌های موج‌گونه مهم بود، در اینجا اهمیت ندارد. شمارنده همزمان با هر یک از دو لبه مثبت یا منفی پالس ساعت تریگر می‌گردد. فلیپ فلاپ‌های متمم‌ساز در یک شمارنده دودویی می‌توانند از نوع JK، T یا D با گیت‌های XOR باشند. هم‌ارزی سه نوع فلیپ فلاپ در شکل ۱۳-۵ مشخص شده است.

بالا- پایین شمار دودویی

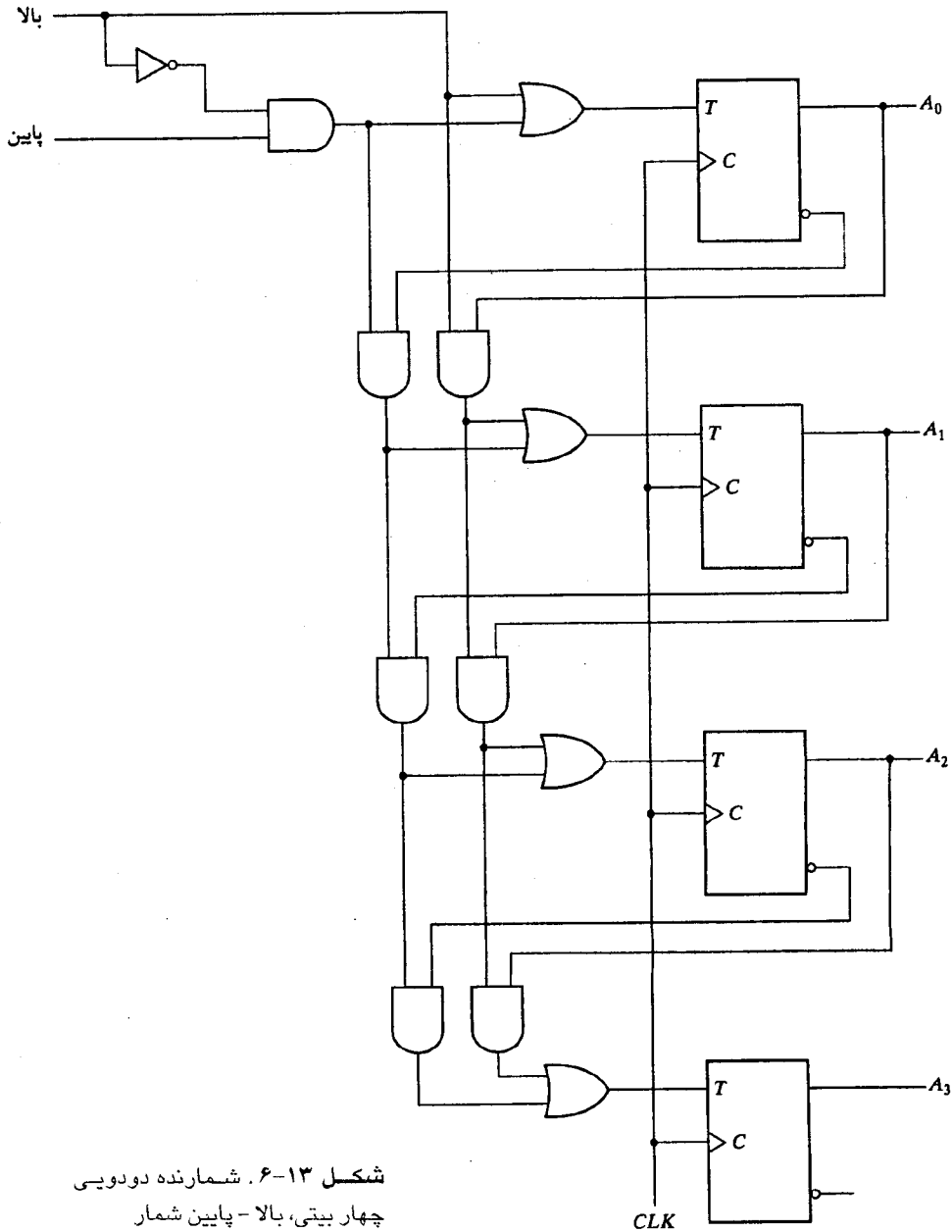
یک شمارنده پایین شمار دودویی همزمان وارد حالات معکوسی از 1111 به سمت 0000 و سپس به 1111 می‌شود تا شمارش را تکرار کند. می‌توان شمارنده پایین شماری به روش معمول ساخت، ولی نتایج از وارسی شمارش دودویی پایین شمار قابل پیش‌بینی هست. به این ترتیب که بیت مکان کم‌ارزش‌تر با هر پالس متمم می‌شود. هر بیت در هر مکان دیگر اگر همه بیت‌های کم‌ارزش‌تر 0 باشند،



شکل ۱۲-۶. شمارنده دودویی همزمان 4 بیتی

متمم می‌گردد. مثلاً پس از حالت فعلی 0100، حالت بعدی 0011 قرار دارد. کم‌ارزش‌ترین بیت همواره متمم می‌گردد. بیت باارزش‌تر دوم چون بیت اول 0 است، متمم می‌شود. سومین بیت متمم می‌شود زیرا دو بیت اول برابر 0 اند. ولی چهارمین بیت تغییر نمی‌کند چون همه بیت‌های پایین رتبه 0 نیستند.

یک شمارنده پایین شمار می‌تواند مشابه شکل ۱۲-۶ ساخته شود، با این تفاوت که ورودی گیت‌ها از خروجی‌های متمم فلیپ فلاپ‌های قبلی می‌آیند. می‌توان دو عمل بالا و پایین شمار را با هم ترکیب کرد و به این ترتیب بالا- پایین شمار ساخت. مدار چنین شمارنده‌ای که از فلیپ فلاپ‌های T استفاده می‌کند در شکل ۱۳-۶ آمده است. این مدار دارای یک ورودی کنترل بالا و یک ورودی کنترل پایین



شکل ۱۳-۶ . شمارنده دودویی چهار بیتی، بالا - پایین شمار

است. وقتی ورودی بالا برابر 1 است، مدار رو به بالا می‌شمارد، زیرا ورودی‌های T سیگنال‌های خود را از مقادیر خروجی نرمال فلیپ فلاپ‌ها دریافت می‌کنند. وقتی ورودی پایین برابر 1 است ورودی بالا برابر 0 است، مدار رو به پایین می‌شمارد زیرا خروجی‌های متمم شده فلیپ فلاپ‌های قبلی به ورودی‌های T اعمال شده‌اند. وقتی هر دو مقدار بالا و پایین 0 باشند، مدار تغییر نکرده و شماره ثابت می‌ماند. وقتی هر دو ورودی بالا و پایین 1 باشند، مدار رو به بالا می‌شمارد. این موجب می‌شود تا همیشه تنها یک عمل اجرا گردد.

شمارنده BCD

یک شمارنده BCD دهدهی کد شده به دودویی از 0000 تا 1001 و بعد 0000 می‌شمارد. به دلیل بازگشت از 9 به 0، یک شمارنده BCD دارای الگوی منظمی همچون شمارنده دودویی نیست. برای راه‌اندازی مدار یک شمارنده همزمان BCD، لازم است از روال طراحی یک مدار ترتیبی استفاده شود.

جدول حالت یک شمارنده BCD در جدول ۵-۶ لیست شده است. وضعیت ورودی فلیپ فلاپ‌های T از حالات فعلی و بعدی بدست می‌آیند. یک خروجی y هم در جدول دیده می‌شود. وقتی حالت فعلی 1001 باشد این خروجی برابر 1 است. به این ترتیب y می‌تواند شمارش دهه باارزش‌تر بعدی را فعال کرده و به طور همزمان از 1001 به 0000 برود.

معادلات ورودی فلیپ فلاپ‌ها را می‌توان به کمک نقشه ساده کرد. حالات بی‌استفاده برای مینترم‌های 10 الی 15 جملات بی‌اهمیت تلقی می‌شوند. توابع ساده شده عبارتند از:

$$T_{Q1} = 1$$
$$T_{Q2} = Q_8' Q_1$$
$$T_{Q4} = Q_2 Q_1$$
$$T_{Q8} = Q_8 Q_1 + Q_4 Q_2 Q_1$$
$$y = Q_8 Q_1$$

جدول ۵-۶. جدول حالت یک شمارنده BCD

حالت فعلی				حالت بعدی				خروجی	ورودی های فلیپ فلاپ			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

می‌توان مدار را به سادگی با چهار فلیپ فلاپ T و پنج گیت AND و یک OR طراحی کرد. شمارنده‌های BCD همزمان را می‌توان برای شمارش اعداد دهدهی با هر طول به صورت متوالی به‌یکدیگر متصل کرد. این نوع سری‌سازی در شکل ۱۱-۶ دیده شد، با این تفاوت که خروجی y باید به ورودی شمارش دهه باارزش‌تر بعدی وصل گردد.

شمارنده دودویی با بار شدن موازی

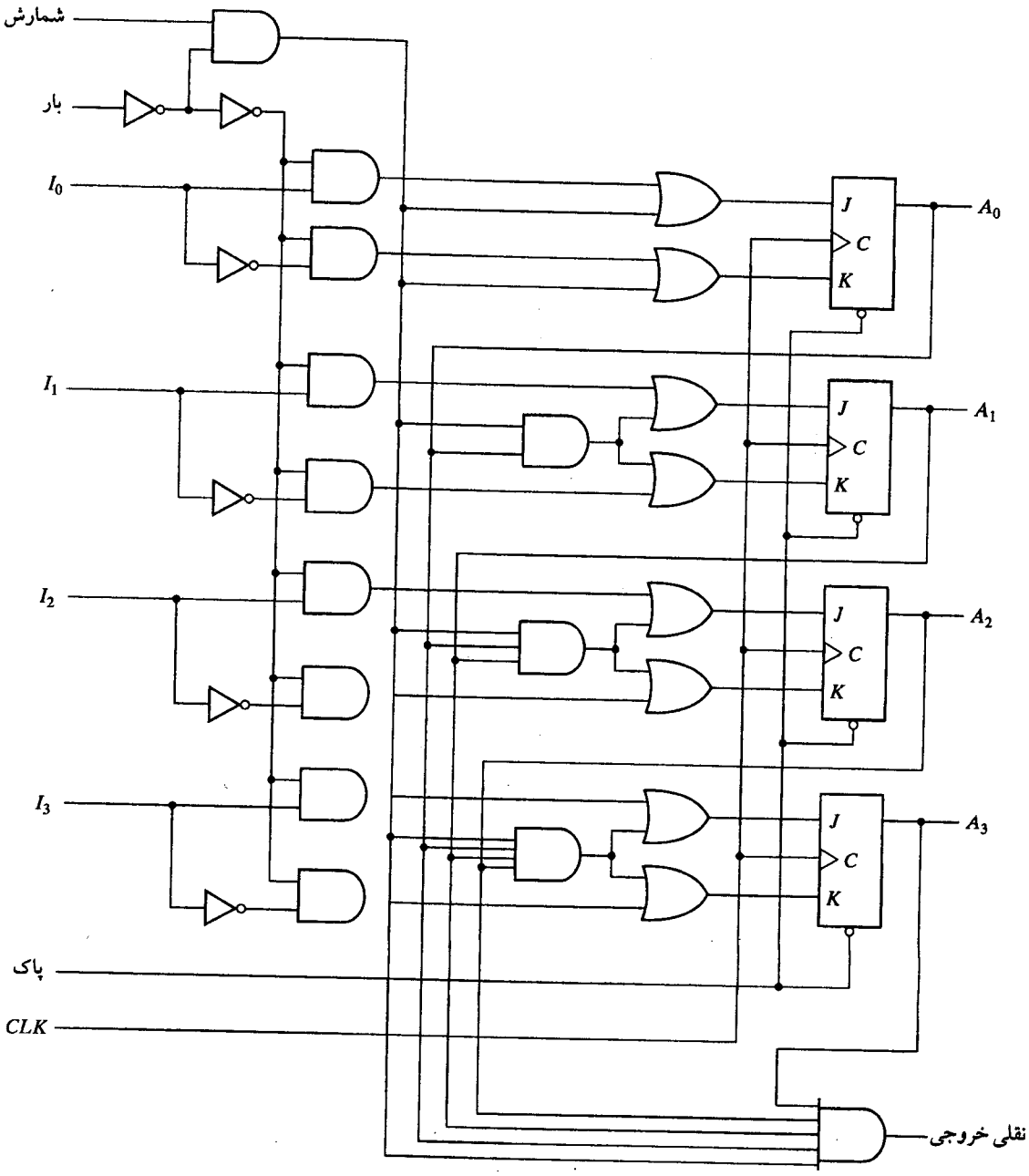
اغلب شمارنده‌هایی که در سیستم‌های دیجیتال به کار می‌روند نیاز به قابلیت انتقال موازی یک عدد دودویی اولیه به داخل شمارنده، قبل از شروع دارند. شکل ۱۴-۶ نمودار منطقی یک ثبات 4 بیت را نشان می‌دهد، که قابلیت بار شدن موازی دارد و می‌تواند به عنوان یک شمارنده به کار رود. اگر ورودی کنترل بار شدن در وضعیت 1 باشد، عمل شمارش را غیرفعال می‌کند و موجب انتقال داده از چهار ورودی داده به چهار فلیپ فلاپ می‌گردد. اگر هر دو ورودی کنترل 0 باشد، پالس‌های ساعت حالت ثبات را عوض نمی‌کند.

ضمن فعال بودن ورودی شمارش، اگر همه فلیپ فلاپ‌ها در 1 باشند، خروجی نقلی 1 می‌شود. این حالتی است که طی آن فلیپ فلاپ بیت باارزش‌تر بعدی متمم می‌گردد. خروجی نقلی برای گسترش شمارنده به بیش از چهار بیت نیز مفید است. هنگام تولید مستقیم نقلی خروجی به دلیل کاهش تأخیر آن سرعت شمارنده افزایش می‌یابد. در رفتن از 1111 به 0000، تنها یک گیت تأخیر وجود دارد، در صورتی که در زنجیره گیت AND شکل ۱۲-۶، چهار گیت تأخیر موجود است. به طور مشابه هر فلیپ فلاپ به یک گیت AND مرتبط است که مستقیماً خروجی همه فلیپ فلاپ‌ها را دریافت می‌کند.

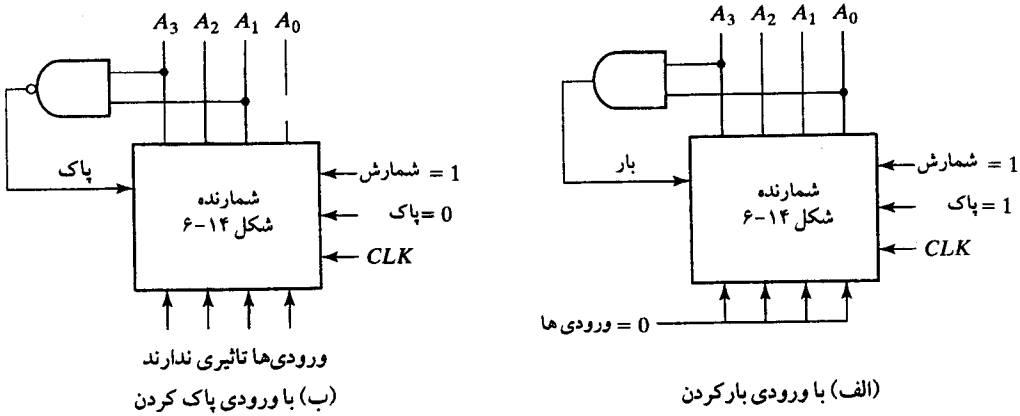
عملکرد شمارنده در جدول ۶-۶ خلاصه شده است. چهار ورودی کنترل، یعنی پاک، CLK، بار و شمارش حالت بعدی را معین می‌کنند. ورودی پاک غیرهمزمان است و هر وقت 0 شود بدون توجه به وجود پالس ساعت یا دیگر ورودی‌ها، شمارنده را پاک می‌کند. این مطلب با وارده X در جدول ذکر شده که به معنی حالات بی‌اهمیت برای ورودی‌هاست. در دیگر حالات ورودی پاک در 1 قرار دارد. با قرار داشتن ورودی‌های بار و شمارش در 0، خروجی‌ها عوض نمی‌شوند. با ورودی بار در 1، انتقال از $I_0 - I_3$ به ثبات در لبه مثبت پالس ساعت انجام می‌شود. مستقل از مقدار ورودی شمارش، داده ورودی در ثبات بار می‌شود، زیرا ورودی شمارش با فعال شدن ورودی بار، غیر فعال می‌شود. اگر ورودی شمارش کنترل شده باشد، ورودی بار در 0 خواهد بود.

جدول ۶-۶. جدول عملکرد شمارنده شکل (۱۴-۶)

تابع	شمارش	بار	CLK	پاک کننده
به 0 پاک می‌شود	X	X	X	0
ورودی بارشونده	X	1	↑	1
شمارش حالت دودویی بعدی	1	0	↑	1
بلاتغییر	0	0	↑	1



شکل ۱۴-۶ . شمارنده دودویی 4 بیتی با امکان بار شدن موازی



شکل ۱۵-۶. دو روش دسترسی به شمارنده BCD با امکان بار شدن موازی

یک شمارنده با بار شدن موازی را می‌توان برای تولید هر رشته شمارش موردنظر به کار برد. شکل۱۵-۶ دو راه تولید شمارش BCD با بار شدن موازی را نشان می‌دهد. در هر حال، برای فعال کردن شمردن از طریق ورودی CLK ، کنترل شمارش در 1 قرار داده می‌شود. همچنین به خاطر بسپارید که کنترل بار از شمردن جلوگیری می‌کند و عمل پاک مستقل از دیگر ورودی‌های کنترل است.

گیت AND در شکل ۱۵-۶ (الف) وقوع حالت 1001 را شناسایی می‌کند. شمارنده در آغاز به 0 پاک می‌شود و سپس ورودی‌های پاک و بار به 1 برده می‌شوند بنابراین شمارنده همواره فعال خواهد بود. مادامی که خروجی گیت AND برابر 0 است، هر لبه ساعت مثبت، شمارنده را یک بار افزایش می‌دهد. وقتی که خروجی به 1001 برسد، A_0 و A_3 برابر 1 خواهند شد و بنابراین خروجی گیت AND برابر 1 می‌گردد. این حالت ورودی بار را فعال می‌کند و بنابراین در لبه ساعت بعدی ثبات نمی‌شمارد، ولی از طریق چهار ورودی بار خواهد شد. چون همه ورودی‌ها به منطق 0 مرتبط هستند و همه مقادیر 0 به دنبال شماره 1001 ، وارد ثبات می‌گردند، بنابراین طبق آنچه در BCD لازم است از 0000 تا 1001 شمرده و سپس به 0000 باز می‌گردد.

در شکل ۱۵-۶ (ب)، گیت AND شماره 1010 را تشخیص می‌دهد، و به محض وقوع آن، ثبات پاک می‌شود. شماره 1010 برای ماندن فرصت چندانی نمی‌یابد زیرا ثبات بلافاصله به 0 خواهد رفت. هنگام رفتن از 1010 به 1011 در خروجی A_0 یک جرقه کوچک رخ داده و بلافاصله به 0000 خواهد رفت. این جرقه لحظه‌ای ممکن است مطلوب نباشد و به این دلیل، این آرایش پیشنهاد نمی‌گردد. اگر شمارنده دارای وروی پاک کردن باشد، می‌توان پس از 1001 آن را پاک کرد.

۶-۵ دیگر شمارنده‌ها

می‌توان شمارنده‌ها را برای تولید هر رشته از حالات طراحی کرد. یک شمارنده تقسیم بر N که به آن N شمار هم می‌گویند، شمارنده‌ای است که یک رشته N حالتی را تکرار می‌کند. رشته ممکن است

شمارش دودویی یا دیگر رشته‌های اختیاری را دنبال کند. از شمارنده‌ها برای تولید سیگنال‌های زمانبندی جهت کنترل یک رشته از اعمال در یک سیستم دیجیتال استفاده می‌شود. شمارنده را می‌توان با شیفت رجیسترها هم ساخت. به این دلیل، به معرفی چند شمارنده غیردودویی می‌پردازیم.

شمارنده با حالات بی‌استفاده

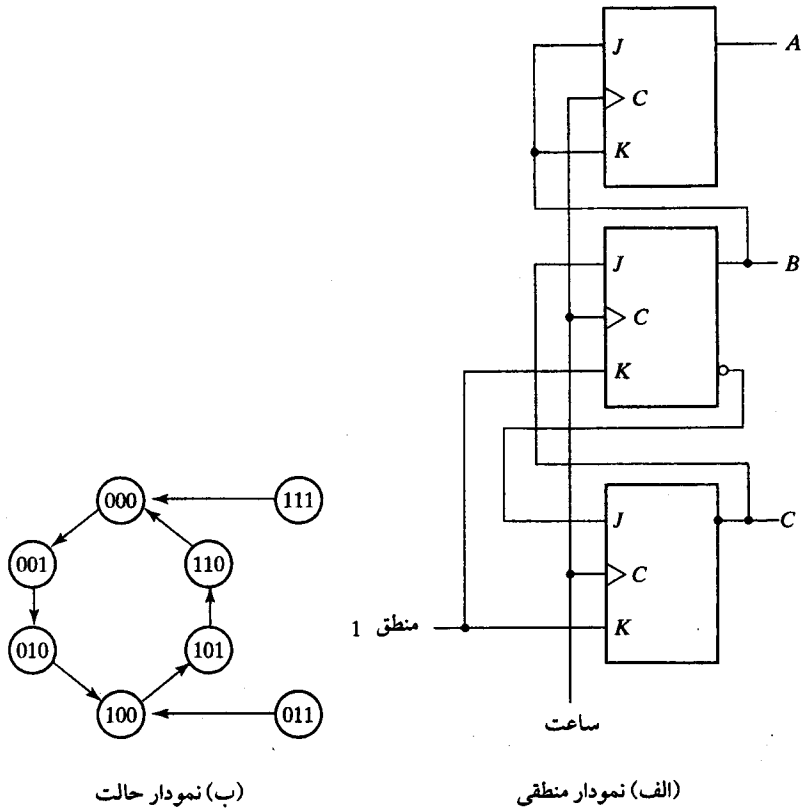
مداری با n فلیپ فلاپ دارای 2^n حالت دودویی است. مواردی وجود دارد که در آن یک مدار ترتیبی حالات کمتری از حداکثر فوق را به کار می‌برد. حالاتی که به کار نروند در جدول حالت لیست نمی‌شوند. هنگام ساده کردن معادلات ورودی، حالات به کار نرفته را می‌توان به عنوان حالت بی‌اهمیت یا حالت خاص بعدی تلقی کرد. به محض طراحی و ساخت مدار عوامل خارجی ممکن است آن را وارد یکی از حالات به کار نرفته کنند. در این حال لازم است مطمئن شویم که مدار بالاخره وارد یکی از حالات معتبر خواهد شد و بنابراین عملکرد معمول خود را دنبال خواهد کرد. در غیر این صورت، اگر مدار ترتیبی در میان حالات به کار نرفته به عنوان حالات بی‌اهمیت تصور شوند، آنگاه پس از طراحی مدار، باید در مورد اثر آنها تحقیق به عمل آید. پس از طراحی می‌توان حالت بعدی حاصل از حالت به کار نرفته را با تحلیل مدار بدست آورد.

به منظور تشریح، شمارنده مربوط به جدول ۷-۶ را ملاحظه کنید. شمارش، رشته‌ای تکراری از شش حالت است، که فلیپ فلاپ‌های B و C شماره‌های دودویی 00، 01 و 10 را تکرار می‌کنند و A نیز هر سه شماره یک بار از 0 به 1 و بالعکس تغییر می‌نماید. رشته شمارش شمارنده دودویی سر راست نیست و دو حالت 011 و 111 در شمارش لحاظ نشده‌اند. انتخاب فلیپ فلاپ‌های JK شرایط ورودی جدول را به دنبال خواهد داشت. ورودی‌های K_B و K_C تنها 1 و X را در ستون‌های خود دارند، بنابراین این حالات همیشه برابر 1 هستند. معادلات ورودی دیگر را می‌توان با مینترم‌های 3 و 7 به عنوان بی‌اهمیت مشخص کرد. معادلات ساده شده به قرار زیراند:

$$J_A = B \qquad K_A = B$$
$$J_B = C \qquad K_B = 1$$
$$J_C = B' \qquad K_C = 1$$

جدول ۷-۶. جدول حالت شمارنده

حالت فعلی			حالت بعدی			ورودی های فلیپ فلاپ					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X



شکل ۱۶-۶. شمارنده‌ای با حالات به کار رفته

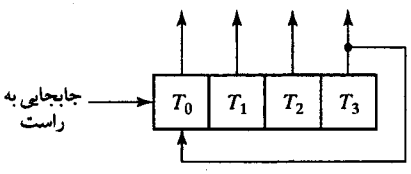
نمودار منطقی شمارنده در شکل ۱۶-۶ (الف) دیده می‌شود. چون دو حالت به کار نرفته وجود دارد، مدار را برای تعیین اثر آن‌ها تحلیل می‌کنیم. اگر مدار به علت وقوع یک سیگنال خطا به حالت 011 برود، پس از اعمال یک پالس ساعت به 100 خواهد رفت. با بررسی نمودار منطقی و توجه به این که با $B = 1$ لبه ساعت بعدی A متمم و C به 0 می‌رود، و وقتی $C = 1$ باشد لبه پالس بعدی B را متمم می‌کند، این نتیجه‌گیری به عمل آمده است. به طریقی مشابه، می‌توان حالت بعدی را از حالت فعلی 111 به 000 ارزیابی کرد.

نمودار حالت همراه با حالات به کار نرفته در شکل ۱۶-۶ (ب) ملاحظه می‌شود. اگر مدار به علت عوامل خارجی وارد یکی از حالات بی‌استفاده شود، پالس شمارش بعدی آن را به یکی از حالات معتبر خواهد برد و آنگاه مدار به طور صحیحی به شمارش خود ادامه خواهد داد. بنابراین، مدار خود تصحیح است. یک شمارنده خود تصحیح اگر در یکی از حالات به کار نرفته برود، نهایتاً پس از یک یا چند پالس ساعت به رشته شمارش طبیعی باز خواهد گشت.

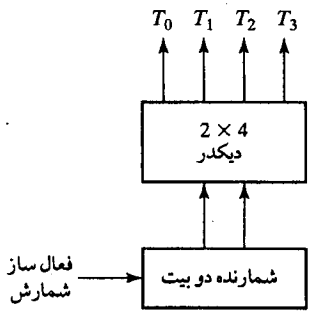
شمارنده حلقوی

سیگنال‌های زمانبندی که رشته عملیاتی را در یک سیستم دیجیتال کنترل می‌کنند با یک شیفت رجیستر یا یک شمارنده و یک دیکدر قابل تولیدند. یک شمارنده حلقوی یک شیفت رجیستر چرخشی است که در آن هر بار تنها یک فلیپ فلاپ در حالت 1 است و همه دیگر فلیپ فلاپ‌ها صفراند. تنها بیت نشانده (1) به فلیپ فلاپ بعدی جابجا می‌شود تا رشته‌ای از سیگنال‌های زمانبندی تولید گردد.

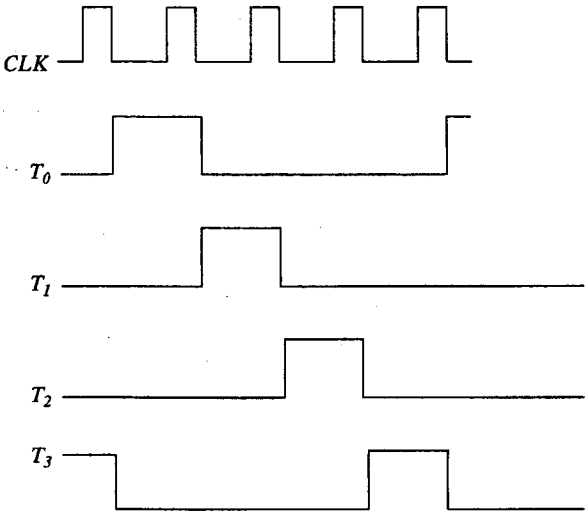
شکل ۱۷-۶ (الف) یک شیفت رجیستر 4 بیت متصل به یک شمارنده حلقوی را نشان می‌دهد. مقدار اولیه ثبات 1000 است. تنها بیت فوق‌الذکر با هر پالس ساعت به سمت راست جابجا می‌شود و چرخش هم از T_3 به T_0 اتفاق می‌افتد. هر چهار پالس ساعت یک بار یکی از فلیپ فلاپ‌ها در حالت 1 قرار می‌گیرد. هر خروجی پس از یک گذر لبه منفی پالس ساعت، 1 می‌گردد و در طول سیکل ساعت بعدی در 1 باقی می‌ماند.



(الف) شمارنده حلقوی (مقدار اولیه = 1000)



(ب) شمارنده و دیکدر



(پ) رشته‌ای از چهار سیگنال زمانی

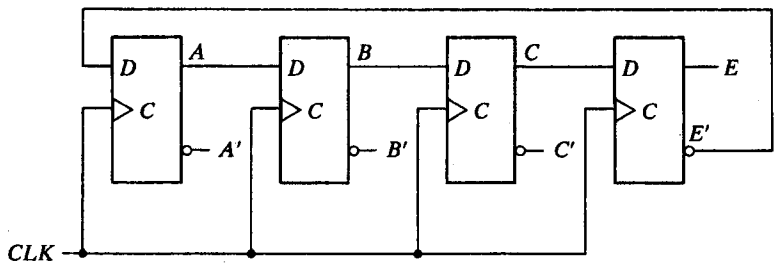
شکل ۱۷-۶. تولید سیگنالهای زمانی

سیگنال‌های زمانبندی با یک شمارنده 2 بیت که به چهار حالت جدا از هم می‌رود نیز تولید می‌گردد. دیکدر شکل ۱۷-۶ (ب) چهار حالت شمارنده را دیکد می‌کند و رشته سیگنال‌های زمانبندی لازم را ایجاد می‌نماید.

برای تولید 2^n سیگنال، به یک شیفت رجیستر با 2^n فلیپ فلاپ و یا به یک شمارنده دودویی n بیتی همراه با یک دیکدر n به 2^n نیاز است. مثلاً 16 سیگنال زمانبندی می‌تواند با یک شیفت رجیستر 16 بیتی به عنوان یک شمارنده حلقوی یا یک شمارنده 4 بیت و یک دیکدر 4 به 16 تولید گردد. در حالت اول، به 16 فلیپ فلاپ نیاز است. در دومی، 4 فلیپ فلاپ و 16 گیت AND چهار ورودی برای دیکدر لازم می‌باشد. در این حالت تعداد فلیپ فلاپ‌ها کمتر از شمارنده حلقوی است و دیکدر تنها گیت دو ورودی نیاز دارد. این ترکیب را شمارنده جانسون نامند.

شمارنده جانسون

یک شمارنده حلقوی k بیتی، یک بیت را بین k فلیپ فلاپ گردش می‌دهد تا بدین وسیله k حالت قابل تفکیک تولید شود. اگر شیفت رجیستر به صورت یک شمارنده حلقوی دنباله چرخان در آید، تعداد این حالات دو برابر می‌شود. شمارنده حلقوی دنباله چرخان یک شیفت رجیستر دوار است که خروجی متمم آخرین فلیپ فلاپ به ورودی اولین فلیپ فلاپ متصل شده است. شکل ۱۸-۶ (الف) چنین شیفت رجیستری را نشان می‌دهد. اتصال پسخوردی از خروجی متمم سمت راست‌ترین فلیپ فلاپ به



(الف) شمارنده حلقوی چهار طبقه دنباله چرخان

رشته اعداد	خروجی‌های فلیپ فلاپ A	B	C	E	گیت AND لازم برای خروجی
1	0	0	0	0	$A'E'$
2	1	0	0	0	AB'
3	1	1	0	0	BC'
4	1	1	1	0	CE'
5	1	1	1	1	AE
6	0	1	1	1	$A'B$
7	0	0	1	1	$B'C$
8	0	0	0	1	$C'E$

(ب) رشته شمارش و دیکد کردن مورد نیاز

شکل ۱۸-۶. ساختار یک شمارنده جانسون

ورودی سمت چپ‌ترین فلیپ فلاپ ایجاد می‌گردد. شیفت رجیستر مزبور محتوای خود را باهر پالس ساعت یک بار به سمت راست جابجا می‌کند، و در همان زمان مقدار متمم فلیپ فلاپ E به فلیپ فلاپ A منتقل می‌گردد. با شروع از حالت پاک شده، شمارنده حلقوی وارد رشته حالات هشتگانه‌ای می‌شود، شکل ۱۸-۶ (ب). به طور کلی، یک شمارنده حلقوی دنباله چرخان k بیتی یک رشته $2k$ حالته را دنبال می‌نماید. این شمارنده از حالتی که همه بیت‌های آن 0 است شروع می‌نماید و هر عمل شیفت یک 1 را از سمت چپ وارد می‌کند تا همه ثبات‌ها با 1 پر شود. در دنباله عملیات 0ها از سمت چپ وارد می‌شوند تا وقتی که همه بیت‌های ثبات مجدداً با 0 پر شوند.

شمارنده جانسون، شمارنده حلقوی دنباله چرخان k بیتی به همراه $2k$ گیت برای دیکدر کردن و تهیه $2k$ سیگنال زمانبندی خروجی است. گیت‌های دیکدر در شکل ۱۸-۶ نشان داده نشده‌اند، اما در آخرین ستون جدول مشخص گردیده‌اند. پس از وصل هشت گیت لیست شده در جدول، ساختار شمارنده جانسون کامل خواهد بود. نظر به این که هر گیت در حین یک حالت ویژه توانا می‌شود، خروجی گیت‌ها، به ترتیب هشت سیگنال زمانبندی را تولید خواهند کرد.

دیکد کردن یا رمزگشایی یک شمارنده k بیتی حلقوی دنباله چرخان در بدست آورن $2k$ دنباله زمانی، از یک الگوی منظم پیروی می‌کند. حالتی که همه بیت‌ها 0 هستند توسط خروجی‌های متمم دو فلیپ فلاپ واقع در منتهی‌الیه دیکد می‌شود. کلیه حالات دیگر با الگوی 0 و 1 یا 1 و 0 مجاور دیکد می‌شوند. به عنوان مثال، دنباله 7 دارای یک الگوی 0 و 1 در فلیپ فلاپ‌های B و C است. بنابراین خروجی دیکد شده با گرفتن متمم B و خروجی طبیعی C یعنی $B'C$ بدست می‌آید.

یکی از معایب مدار شکل ۱۸-۶ (الف) این است که اگر مدار به حالت بی‌استفاده وارد شود، شروع به دنبال کردن حالات نامعتبر کرده و هرگز راهش را به یک حالت معتبر نخواهد یافت. این شکل را می‌توان با تصحیح مدار به صورتی که از حالت نامعتبر دوری جوید، رفع کرد. یکی از روش‌های اصلاح، قطع خروجی فلیپ فلاپ B است که به ورودی D فلیپ فلاپ C می‌رود

$$D_C = (A + C)B$$

که D_C معادله ورودی فلیپ فلاپ برای ورودی D در فلیپ فلاپ C است. شمارنده جانسون را می‌توان با هر طول زمانی ساخت. تعداد فلیپ فلاپ‌های لازم نصف تعداد سیگنال‌های زمانبندی است. تعداد گیت‌ها برای ساخت آن نصف تعداد سیگنال‌های زمانبندی بوده و فقط گیت دو ورودی نیاز دارد.

۶-۶ HDL برای ثبات‌ها و شمارنده‌ها

ثبات‌ها و شمارنده‌ها را می‌توان در هر یک از سطوح رفتاری و ساختاری توصیف کرد. در سطح رفتاری، ثبات با انواع عملیاتی که مشابه جدول تابع انجام می‌دهد مشخص می‌گردد. یک توصیف ساختاری مدار را برحسب مجموعه‌ای از اجزاء مانند گیت‌ها، فلیپ فلاپ‌ها و مولتی پلکسرها نشان می‌دهد. انواع عناصر برای تشکیل توصیف سلسله مراتبی طرح، مشابه یک نمودار منطقی ذکر می‌گردد. در این فصل سه مدار را برای تشریح دو نوع توصیف به کار خواهیم برد.

شیفت رجیستر

شیفت رجیستر یونیورسال که در بخش ۶-۶ ارائه شد، یک شیفت رجیستر دو جهته با بار شدن موازی است. چهار عملی که به وسیله ثبات ساعت‌دار اجرا شده‌اند در جدول (۶-۶) دیده می‌شوند. می‌توان ثبات را به طور غیرهمزمان هم پاک کرد. توصیف رفتاری یک شیفت رجیستر یونیورسال در مثال ۱-۶ HDL نشان داده شده است. در مجموع، دو ورودی انتخاب، دو ورودی سریال، یک ورودی موازی 4 بیت و یک خروجی موازی 4 بیت وجود دارند. بلوک always پنج عملی که با ثبات قابل اجرا است را توصیف می‌نماید. ورودی Clr ثبات را به طور غیرهمزمان پاک می‌کند. برای این که ثبات به لبه مثبت ساعت واکنش نشان دهد، Clr باید در سطح منطقی بالا باشد. چهار عمل ساعت‌دار ثبات از مقادیر دو ورودی انتخاب در عبارت case مشخص می‌گردند (S1 و S0 بعد از کلمه کلیدی case به ترتیب در داخل کروشه قرار گرفته‌اند.) عمل جابجایی با عملیات زنجیره‌ای ورودی سریال و سه فلیپ فلاپ انجام می‌شود. مثلاً عبارت

```
A = {rtin, A[3:1]}
```

به معنی جابجایی زنجیره ورودی سریال به سمت راست (rtin) با فلیپ فلاپ‌های A_3، A_2 و A_1 است تا یک عدد 4 بیتی تشکیل شده و به A[3 : 0] انتقال یابد. این موجب تولید یک عمل جابجایی به راست می‌گردد. توجه کنید که عمل مدار مستقل از هر سخت‌افزاری بیان شد.

ساختار ثبات با مراجعه به نمودار منطقی شکل ۷-۶ قابل توصیف است. نمودار نشان می‌دهد که ثبات با چهار مولتی پلکسر و چهار فلیپ فلاپ D ساخته شده است. توصیف ساختاری ثبات در

مثال ۱-۶، HDL

```
//Behavioral description of
//Universal shift register
// Fig. 6-7 and Table 6-3
module shftreg (s1,s0,Pin,lfin,rtin,A,CLK,Clr);
   input s1,s0;                    //Select inputs
   input lfin, rtin;               //Serial inputs
   input CLK,Clr;                  //Clock and Clear
   input [3:0] Pin;                //Parallel input
   output [3:0] A;                 //Register output
   reg [3:0] A;
   always @ (posedge CLK or negedge Clr)
     if (~Clr) A = 4'b0000;
      else
        case ({s1,s0})
         2'b00: A = A;             //No change
         2'b01: A = {rtin,A[3:1]}; //Shift right
         2'b10: A = {A[2:0],lfin}; //Shift left
         2'b11: A = Pin;           //Parallel load input
        endcase
endmodule
```

```
//Structural description of
//Universal shift register(see Fig. 6-7)
module SHFTREG (I,select,lfin,rtin,A,CLK,Clr);
   input [3:0] I;            //Parallel input
   input [1:0] select;       //Mode select
   input lfin,rtin,CLK,Clr;  //Serial inputs,clock,clearn
   output [3:0] A;           //Parallel output
 //Instantiate the four stages
   stage ST0 (A[0],A[1],lfin,I[0],A[0],select,CLK,Clr);
   stage ST1 (A[1],A[2],A[0],I[1],A[1],select,CLK,Clr);
   stage ST2 (A[2],A[3],A[1],I[2],A[2],select,CLK,Clr);
   stage ST3 (A[3],rtin,A[2],I[3],A[3],select,CLK,Clr);
endmodule

//One stage of shift register
module stage(i0,i1,i2,i3,Q,select,CLK,Clr);
   input i0,i1,i2,i3,CLK,Clr;
   input [1:0] select;
   output Q;
   reg Q;
   reg D;
//4x1 multiplexer
   always @ (i0 or i1 or i2 or i3 or select)
         case (select)
            2'b00: D = i0;
            2'b01: D = i1;
            2'b10: D = i2;
            2'b11: D = i3;
         endcase
//D flip-flop
   always @ (posedge CLK or negedge Clr)
         if (~Clr) Q = 1'b0;
         else Q = D;
endmodule
```

مثال۲-۶ HDL نشان داده شده است. در این مثال دو مدول وجود دارد. اولین مدول ورودی‌ها، خروجی‌ها و ساعت سپس طبقات ثبات را مشخص می‌کند. چهار بخش فوق ارتباطات درونی بین چهار طبقه را معین می‌سازد و جزئیات ساخت ثبات را طبق نمودار منطقی مهیا می‌سازد. دومین مدول دو بلوک always دارد. اولین بلوک always ، مولتی پلکسر و دومی فلیپ‌فلاپ را توصیف می‌نماید. روی هم رفته آنها یک مرحله از ثبات را تعریف می‌نمایند.

شمارنده همزمان

مثال ۳-۶ HDL شمارنده همزمان را با بار شدن موازی، شکل ۱۴-۶، توصیف می‌کند. شمارش

```
//Binary counter with parallel load
//See Figure 6-14 and Table 6-6
module counter (Count,Load,IN,CLK,Clr,A,CO);
  input Count,Load,CLK,Clr;
  input [3:0] IN;                    //Data input
  output CO;                         //Output carry
  output [3:0] A;                    //Data output
  reg [3:0] A;
  assign CO = Count & ~Load & (A == 4'b1111);
  always @ (posedge CLK or negedge Clr)
    if (~Clr) A = 4'b0000;
    else if (Load)  A = IN;
    else if (Count) A = A + 1'b1;
    else A = A;
endmodule
```

(count) بار شدن (load)، CLK و Clr ورودی‌هایی هستند که کار ثبات را طبق عملیات جدول (۶-۶) معین می‌نمایند. شمارنده چهار ورودی داده، چهار خروجی داده و یک خروجی نقلی دارد. نقلی خروجی CO با یک مدار ترکیبی تولید می‌شود و با یک عبارت assign مشخص می‌گردد. اگر شمارش به 15 برسد CO = 1 می‌شود و در این حال شمارنده در وضعیت شمارش است. بنابراین CO = 1 خواهد بود به شرطی که Count = 1 و Load = 0 و A = 1111 باشد؛ در غیر این صورت CO = 0 است. بلوک always عملی را که بسته به مقادیر Clr، Load و Count مشخص می‌شود انجام می‌دهد. یک سیگنال منفی در Clr، A را به 0 می‌نشاند. در غیر این صورت اگر Clr = 1 باشد، یکی از سه عمل به هنگام لبه مثبت ساعت انجام می‌گیرد. عبارت if، else if و else به صورت زیر عمل تصمیم‌گیری را اجرا می‌کنند.

```
      if Clr = 0                                    Clear A to 0
else if (Clr = 1 and) Load = 1                      Load inputs to A
else if (Clr = 1 and Load = 0 and) Count = 1        Increment A
   else (Clr = 1 and Load = 0 and Count = 0)        No change in A
```

سلسله مراتب مربوط به عبارات if–else با آنچه در جدول ۶-۶ ملاحظه شد مطابقت دارد.

شمارنده موج‌گونه

توصیف ساختاری یک شمارنده موج‌گونه در مثال ۴-۶ HDL نشان داده شده است. اولین مدول چهار فلیپ فلاپ متمم‌ساز را که در مدول دوم به صورت CF(Q, CLK, Reset) تعریف شده ذکر می‌کند. ساعت (ورودی C) اولین فلیپ فلاپ به ورودی Count بیرونی وصل است (count جایگزین CLK در F0 شده است). ورودی ساعت فلیپ فلاپ دوم به خروجی اول متصل است (A_0 جایگزین

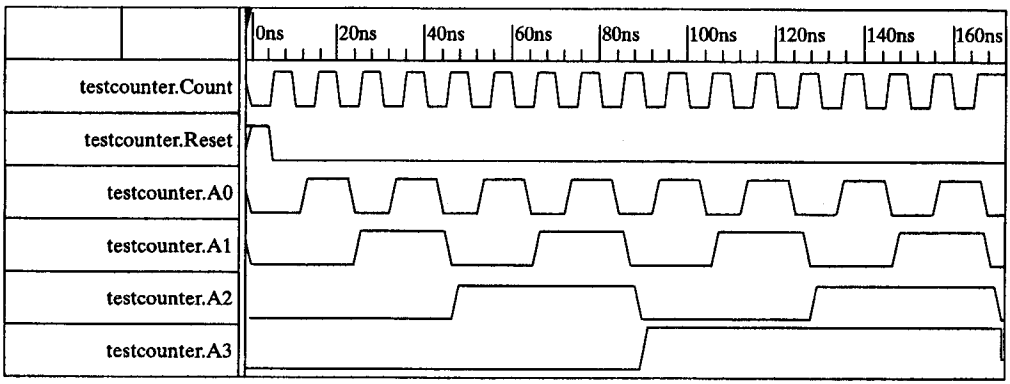
```
//Ripple counter (See Fig. 6-8(b))
module ripplecounter (A0,A1,A2,A3,Count,Reset);
   output A0,A1,A2,A3;
   input Count,Reset;
//Instantiate complementing flip-flop
   CF F0 (A0,Count,Reset);
   CF F1 (A1,A0,Reset);
   CF F2 (A2,A1,Reset);
   CF F3 (A3,A2,Reset);
endmodule

//Complementing flip-flop with delay
//Input to D flip-flop = Q'
module CF (Q,CLK,Reset);
   output Q;
   input CLK,Reset;
   reg Q;
   always @ (negedge CLK or posedge Reset)
      if (Reset) Q = 1'b0;
      else  Q = #2 (~Q);      // Delay of 2 time units
endmodule

//Stimulus for testing ripple counter
module testcounter;
   reg Count;
   reg Reset;
   wire A0,A1,A2,A3;
//Instantiate ripple counter
   ripplecounter RC (A0,A1,A2,A3,Count,Reset);
always
    #5 Count = ~Count;
initial
 begin
      Count = 1'b0;
      Reset = 1'b1;
   #4 Reset = 1'b0;
   #165 $finish;
 end
endmodule
```

CLK در F1 شده است). به طور مشابه، ساعت هر فلیپ فلاپ دیگر به خروجی فلیپ فلاپ قبلی وصل می‌باشد. به این ترتیب فلیپ فلاپ‌ها زنجیروار به هم متصل‌اند تا یک شمارنده موج‌گونه طبق شکل ۸-۶ (ب) بدست آید.

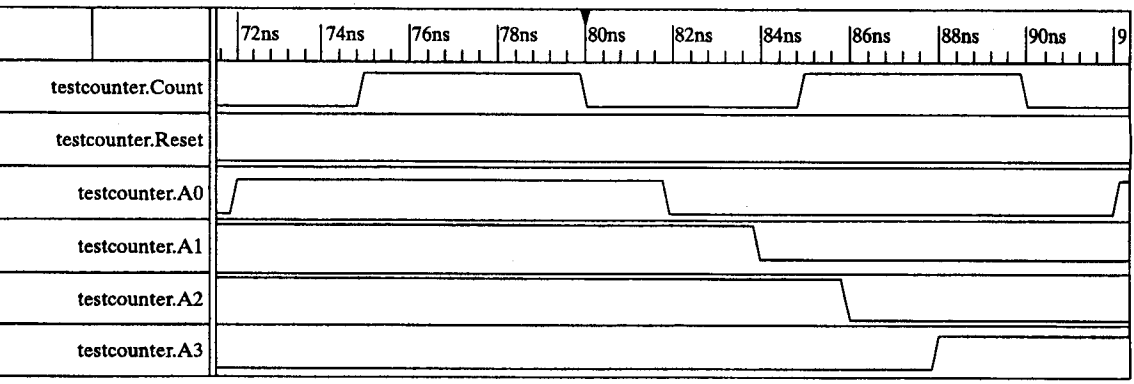
دومین مدول فلیپ فلاپ متمم‌سازی را با تأخیر نشان می‌دهد. مدار فلیپ فلاپ متمم‌ساز با اتصال خروجی متمم به ورودی D ایجاد می‌شود. یک ورودی reset (بازنشانی) در فلیپ فلاپ لحاظ شده است

تا شمارنده مقداردهی اولیه شود. شبیه‌سازهای HDL بدون مقداردهی اولیه قادر به تهیه مقادیر خروجی نیستند. به فلیپ فلاپ تأخیری برابر 2واحد زمانی تخصیص یافته است و این تأخیر از لحظه اعمال ساعت تا لحظه متمم شدن ساعت محاسبه می‌شود. این تأخیر بصورت عبارت Q=#2(~Q) نشان داده می‌شود.

مدول سوم در مثال ۴-۶ محرک شبیه‌سازی و تست شمارنده موج‌گونه را فراهم می‌نماید. عبارت always ساعتی با سیکل 10 واحد زمانی تولید می‌کند. فلیپ فلاپ‌ها در لبه منفی ساعت تریگر می‌شوند، که در زمان‌های t = 10 ، 20 ، 30 و هر 10 واحد زمانی یک بار رخ می‌دهد. شکل موج‌های حاصل برای این شبیه‌سازی در شکل ۱۹-۶ نشان داده شده است. Count هر 10ns یک بار منفی می‌شود. A0 با هر لبه منفی Count و با 2ns تأخیر متمم می‌گردد. هر فلیپ فلاپ، وقتی فلاپ فلاپ قبلی‌اش از 1 به 0 برود، متمم می‌گردد. پس از t = 80ns همه فلیپ فلاپ‌ها متمم شده‌اند زیرا شمارنده از 0111 به 1000 می‌رود. هر خروجی به اندازه 2n تأخیر دارد و به علت آن A_3 در t = 88ns از 0 به 1 و در 168ns از 1 به 0 خواهد رفت.



(الف) از 0 تا 170 ns



(ب) از 70 تا 92 ns

شکل ۱۹-۶. شبیه‌سازی خروجی مثال ۴-۶ HDL

مسائل

۶-۱ در ثبات شکل ۶-۱ یک گیت NAND دو ورودی را لحاظ نمایید و خروجی گیت را به ورودی C همه فلیپ فلاپ‌ها وصل کنید. یکی از ورودی‌های گیت NAND پالس‌های ساعت را از مولد ساعت دریافت می‌کند و ورودی دیگر گیت NAND کنترل بار شدن موازی را مهیا می‌سازد. عملکرد ثبات اصلاح شده را توضیح دهید.

۶-۲ یک ورودی پاک همزمان به ثبات شکل ۶-۲ اضافه کنید. ثبات اصلاح شده قابلیت بار شدن موازی و پاک کردن همزمان را دارد. وقتی که ساعت از لبه مثبت می‌گذرد و ورودی پاک برابر 1 است، ثبات بطور همزمان پاک می‌گردد.

۶-۳ تفاوت بین انتقال موازی و سری چیست؟ چگونگی تبدیل داده سریال به موازی و بالعکس را توضیح دهید. چه نوع ثباتی لازم است؟

۶-۴ محتوای یک ثبات 4 بیتی در آغاز 1101 است. ثبات شش بار با ورودی سریال 101101 به راست جابجا می‌شود. محتوای ثبات پس از هر جابجایی چیست؟

۶-۵ یک شیفت رجیستر 4 بیت یونیورسال شکل ۶-۷ در یک بسته IC قرار دارد.
(الف) نمودار بلوکی IC را بکشید و همه خروجی‌ها و ورودی‌ها را نشان دهید. برای منبع تغذیه دو پایه در نظر بگیرید.
(ب) نمودار بلوکی را با دو آی‌سی برای تولید یک شیفت رجیستر 8 بیت بکشید.

۶-۶ یک شیفت رجیستر 4 بیت با بار شدن موازی با فلیپ فلاپ D طراحی کنید. دو ورودی کنترل shift و Load وجود دارد. وقتی shift = 1 است، محتوای ثبات یک مکان جابجایی می‌شود. وقتی load = 1 و shift = 0 است داده جدید وارد شیفت رجیستر می‌گردد. اگر هر دو ورودی کنترل برابر 0 باشند، محتوای ثبات تغییر نمی‌نماید.

۶-۷ نمودار منطقی یک ثبات 4 بیتی با چهار فلیپ فلاپ D و چهار مولتی پلکسر و ورودی‌های انتخاب مد S1 و S0 را رسم نمایید. ثبات طبق جدول تابع زیر عمل می‌کند:

عملکرد ثبات	s_0	s_1
بلاتغییر	0	0
متمم چهارخروجی	1	0
پاک کردن ثبات به 0	0	1
(همزمان با پالس ساعت)	1	1
بارکردن داده موازی		

۶-۸ جمع‌کننده سریال شکل ۶-۶ دو ثبات 4 بیت را به کار می‌برد. ثبات A عدد دودویی 0101 و ثبات B هم 0111 را نگه می‌دارد. فلیپ فلاپ نقلی در ابتدا 0 است. مقادیر دودویی درون ثبات A و فلیپ فلاپ نقلی را پس از هر جابجایی معین کنید.

۶-۹ دو راه پیاده‌سازی جمع‌کننده سریال (A + B) در بخش ۶-۲ نشان داده شد. برای تبدیل به تفریق‌گر (A − B) باید آن را اصلاح کرد.
(الف) با استفاده از شکل ۶-۵ تغییرات لازم برای انجام تفریق $A + \overline{B} + 1$ را نشان دهید.
(ب) با استفاده از شکل ۶-۶، نشان دهید که با اصلاح جدول ۶-۲ جمع‌کننده به تفریق‌گر تبدیل می‌شود.

۶-۱۰ یک متمم 2 ساز با شیفت رجیستر و یک فلیپ فلاپ بسازید. عدد دودویی از یک سمت آن خارج و متمم 2 از سمت دیگر وارد آن می‌گردد.

۶-۱۱ یک شمارنده موج‌گونه دودویی از فلیپ فلاپ‌هایی که در لبه مثبت ساعت تریگر می‌شوند استفاده می‌کند. اگر (الف) خروجی‌های معمولی فلیپ فلاپ‌ها به ساعت و (ب) متمم خروجی‌ها به ساعت وصل شوند، شمارش چگونه است.

۶-۱۲ نمودار منطقی یک پایین شمار موج‌گونه دودویی 4 بیت را (الف) با فلیپ فلاپ‌هایی که در لبه مثبت ساعت و (ب) در لبه منفی ساعت تریگر می‌شوند، رسم نمایید.

۶-۱۳ نشان دهید که یک شمارنده موج‌گونه BCD با یک شمارنده موج‌گونه 4 بیت و پاک‌کننده غیرهمزمان و یک گیت AND که وقوع 1010 را شناسایی کند، قابل ساخت است.

۶-۱۴ چند فلیپ فلاپ در یک شمارنده موج‌گونه دودویی 10 بیت پس از رسیدن به شمارش‌های زیر متمم خواهد شد.
(الف) 1001100111 (ب) 0011111111 (پ) 1111111111

۶-۱۵ یک فلیپ فلاپ از لحظه لبه پالس تا متمم شدن خروجی دارای تأخیر 5ns است. حداکثر تأخیر در یک شمارنده موج‌گونه دودویی 10 بیت که از آن استفاده کند، چقدر است؟ حداکثر فرکانس شمارش مورد استفاده چقدر است؟

۶-۱۶ شمارنده موج‌گونه شکل ۶-۱۰ دارای چهار فلیپ فلاپ و 16 حالت است که تنها 10 حالت آن به‌کار رفته است. مدار را تحلیل کنید و حالت بعدی را برای هر یک از شش حالت به‌کار نرفته معین نمایید. اگر یک سیگنال پارازیت مدار را به حالات به‌کار نرفته بفرستد، چه پیش می‌آید.

۶-۱۷ یک شمارنده همزمان دودویی 4 بیتی را با فلیپ فلاپ‌های D طراحی نمایید.

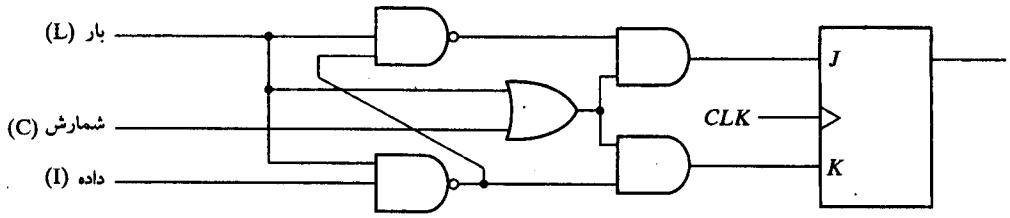
۶-۱۸ در بالا- پایین شمار شکل ۶-۱۳، وقتی هر دو ورودی بالا و پایین فعال شوند، چه رخ می‌دهد؟ مدار را طوری اصلاح کنید که هر دو ورودی 1 باشند ولی شمارنده تغییر حالت ندهد و در همان حال باقی بماند.

۶-۱۹ معادلات ورودی فلیپ فلاپ برای یک شمارنده BCD با فلیپ فلاپ‌های T در بخش ۶-۴ ملاحظه شد. معادلات ورودی را برای این شمارنده (الف) با فلیپ فلاپ‌های JK و (ب) فلیپ فلاپ‌های D بنویسید. سه طرح را مقایسه نمایید و بگویید کدام یک اقتصادی‌تر است.

۶-۲۰ شمارنده دودویی با بار شدن موازی شکل ۶-۱۴ را در یک نمودار بلوکی قرار دهید و همه ورودی‌ها و خروجی‌های آن را نشان دهید.
(الف) اتصالات چهار عدد از این بلوک‌ها را برای تولید یک شمارنده 16 بیت با بار شدن موازی نشان دهید.
(ب) یک شمارنده دودویی بسازید که از 0 تا 64 را بشمارد.

۶-۲۱ شمارنده شکل ۶-۱۴ دارای دو ورودی کنترل بار شدن (L) و شمارش (C) و یک ورودی داده (I_i) است.
(الف) معادلات ورودی فلیپ فلاپ را برای J و K از اولین طبقه برحسب L و C و I بنویسید.
(ب) نمودار منطقی اولین طبقه یک مدار مجتمع (74161) در شکل (م ۶-۲۱) دیده می‌شود. نشان دهید که این مدار معادل با مدار بخش (الف) است.

۶-۲۲ با استفاده از مدار شکل ۶-۱۴، سه روش دیگر برای شمارنده مد 12 را بدست آورید:



شکل (م ۲۱-۶)

(الف) با استفاده از گیت AND و ورودی بار شدن

(ب) با استفاده از نقلی خروجی

(پ) با استفاده از گیت NAND و ورودی پاک کردن غیرهمزمان

۲۳-۶ یک مدار زمانبندی که یک سیگنال خروجی را به مدت هشت پالس ساعت ثابت نگهدارد، طراحی نمایید. سیگنال شروع حالت 1 را خارج می‌کند و پس از هشت سیکل ساعت سیگنال به حالت 0 باز می‌گردد.

۲۴-۶ شمارنده‌ای را با فلیپ فلاپ‌های T بسازید تا رشته زیر را تکرار کند: 0، 1، 3، 7، 6، 4. نشان دهید که وقتی حالات 010 و 101 حالات بی‌اهمیتی تصور شوند، شمارنده به درستی کار نمی‌کند. راهی برای اصلاح آن بیابید.

۲۵-۶ لازم است تا شش سیگنال تکرار شونده T_0 تا T_5 را مشابه شکل ۱۷-۶ (پ) تولید کنیم. مدار را با توجه به موارد زیر طراحی نمایید.

(الف) فقط فلیپ فلاپ (ب) یک شمارنده و یک دیکدر

۲۶-۶ یک سیستم دیجیتال دارای مولد ساعتی است که پالس‌ها را با فرکانس 80MHz تولید می‌کند. مداری طراحی کنید که یک پالس ساعت با زمان سیکل 50ns را تولید نماید.

۲۷-۶ شمارنده‌ای با رشته دودویی 0، 1، 2، 3، 4، 5 و 6 طراحی کنید. از فلیپ فلاپ JK استفاده نمایید.

۲۸-۶ شمارنده‌ای که رشته 0، 1، 2، 4 و 6 را تکرار کند طراحی نمایید از فلیپ فلاپ D استفاده کنید.

۲۹-۶ هشت حالت به کار نرفته در شمارنده حلقوی دنباله چرخان شکل ۱۸-۶ (الف) را لیست کنید. حالت بعدی را برای هر یک از این حالات معین نمایید. نشان دهید که اگر شمارنده خود را در یک حالت نامعتبر بیابد، به حالت معتبر باز نمی‌گردد. مطابق متن کتاب آن را اصلاح کنید و نشان دهید که شمارنده حالات یکسانی را تولید می‌کند ضمن این که با رسیدن به حالت به کار نرفته، به یک حالت معتبر خواهد رفت.

۳۰-۶ نشان دهید که یک شمارنده جانسون با n فلیپ فلاپ یک رشته 2n حالتی تولید می‌کند. 10 حالت مربوط به شمارنده 5 فلیپ فلاپی را لیست نمایید و عبارت بولی هر یک از خروجی‌های 10 عدد گیت AND را معین کنید.

۳۱-۶ توصیف رفتاری و ساختاری HDL ثبات 4 بیت شکل ۱-۶ را بنویسید.

۳۲-۶ (الف) توصیف رفتاری HDL یک ثبات 4 بیت با بار شدن موازی و پاک شدن غیرهمزمان رابنویسید.

(ب) توصیف ساختاری HDL یک ثبات 4 بیت با بار شدن موازی در شکل ۲-۶ را بنویسید. از یک مولتی پلکسر 1 × 2 برای ورودی فلیپ فلاپ‌ها استفاده نمایید. ورودی پاک شدن را نیز لحاظ کنید.

(پ) هر دو توصیف را به کمک برنامه تست چک کنید.

۶-۳۳ برنامه محرک زیر برای شبیه‌سازی یک شمارنده دودویی با بار شدن موازی در مثال ۶-۳ HDL به کار رفته است. با توجه به برنامه، خروجی شمارنده و خروجی نقلی را از t = 0 تا t = 155ns پیش‌بینی کنید.

```
//Stimulus for testing counter
//of Example 6-3
     module testcounter;
       reg Count, Load, CLK, Clr;
       reg [3:0] IN;
       wire CO;
       wire [3:0] A;
       counter cnt (Count, Load, IN, CLK, Clr, A, CO);
       always
           #5 CLK = ~CLK;
         initial
          begin
           Clr = 0;
           CLK = 1;
           Load = 0; Count = 1;
             #5  Clr = 1;
             #50 Load = 1; IN = 4'b1100;
             #10 Load = 0;
             #70 Count = 0;
             #20 $finish;
          end
     endmodule
```

۶-۳۴ توصیف رفتاری HDL را بری شیفت رجیستر 4 بیت شکل ۶-۳ بنویسید.

۶-۳۵ توصیف رفتاری و ساختاری یک بالا- پایین‌شمار 4 بیت که در شکل ۶-۱۳ ملاحظه شد را بنویسید.

۶-۳۶ توصیف رفتاری یک بالا- پایین شمار با بار شدن موازی را برای ورودی‌های کنترل زیر بنویسید.

(الف) شمارنده سه خط کنترل برای سه عمل دارد: بالا، پایین و بار شدن. ترتیب اولویت عبارت است از: بار، بالا و پایین.

(ب) شمارنده برای چهار حالت بالا، پایین، بار و بدون تغییر دو خط انتخاب دارد.

۶-۳۷ توصیف HDL را برای یک شمارنده حلقوی 8 بیتی مشابه با شکل ۶-۱۷ (الف) بنویسید.

۶-۳۸ توصیف HDL را برای یک شمارنده حلقوی دنباله چرخان (شکل ۶-۱۸ (الف)) بنویسید.

۶-۳۹ توصیف رفتاری و ساختاری شمارنده شکل ۶-۱۶ را بنویسید.

مراجع

1. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.

2. NELSON V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.

3. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.

حافظه و منطق برنامه‌پذیر

۷-۱ مقدمه

یک واحد حافظه وسیله‌ای است که اطلاعات دودویی جهت ذخیره شدن به آن منتقل و یا اطلاعاتی که برای پردازش لازم است از آن دریافت می‌شود. وقتی که پردازش داده رخ می‌دهد، اطلاعات از حافظه به ثبات‌های انتخابی در واحد پردازش انتقال می‌یابد. نتایج میانی و پایانی حاصل در واحد پردازش برای ذخیره شدن به واحد حافظه بازگردانده می‌شوند. اطلاعات دودویی واصله از وسیله ورودی هم در حافظه ذخیره می‌گردد و اطلاعات ارسالی به خروجی هم از حافظه برداشته می‌شود. واحد حافظه مجموعه‌ای از سلول‌هاست که قادر است حجم زیادی از اطلاعات دودویی را در خود جمع کند.

در سیستم‌های دیجیتال دو نوع حافظه مورد استفاده قرار می‌گیرد: حافظه با دستیابی تصادفی (RAM) و حافظه فقط خواندنی (ROM). RAM اطلاعات جدید را برای ذخیره می‌پذیرد تا بعد مورد استفاده قرار گیرد. عمل ذخیره کردن اطلاعات جدید در حافظه را عمل نوشتن در حافظه می‌گویند. فرآیند انتقال اطلاعات ذخیره شده در حافظه به بیرون را خواندن نامند. حافظه با دستیابی تصادفی هر دو عمل خواندن و نوشتن را انجام می‌دهد. حافظه فقط خواندنی عمل خواندن را اجرا می‌نماید. این بدان معنی است که اطلاعات دودویی ذخیره شده قبلی در حافظه در هر زمان قابل بازیابی است. با این وجود، اطلاعات موجود را نمی‌توان در حافظه فقط خواندنی نوشت، بلکه فقط می‌توان آن را خواند.

حافظه فقط خواندنی یک وسیله منطقی برنامه‌پذیر است. اطلاعات دودویی که در یک وسیله منطقی برنامه‌پذیر ذخیره می‌شود، به نوعی آرایش یافته و سپس در سخت‌افزار ذخیره می‌شود. این فرآیند را برنامه‌ریزی وسیله گویند. کلمه "برنامه‌ریزی" به یک روال سخت‌افزاری گویند که طی آن بیت‌ها در آرایش سخت‌افزار وسیله وارد می‌شوند.

حافظه فقط خواندنی (ROM) نوعی وسیله منطقی برنامه‌پذیر است. وسیله‌های دیگری از این نوع

(الف) نماد معمولی (ب) نماد منطقی آرایه‌ای

شکل ۱-۷. نمودارهای منطقی معمولی و آرایه‌ای برای گیت OR

عبارتند از آرایه منطقی برنامه‌پذیر (PLA)، منطق آرایه‌ای برنامه‌پذیر (PAL)، و آرایه گیتی برنامه‌پذیر موردی (FPGA). یک وسیله منطقی برنامه‌پذیر مداری مجتمع با گیت‌های منطقی داخلی است که از طریق مسیرهای الکترونیک به هم وصل شده و مثل فیوز عمل می‌کند. در ابتدای کار، همه فیوزها دست نخورده هستند. برنامه‌ریزی وسیله به معنی سوزاندن آن دسته از فیوزهاست که باید برای یک آرایش خاصی از تابع منطقی موردنظر، سوزانده شوند. در این فصل، آرایش وسایل منطقی برنامه‌پذیر را معرفی کرده و روال‌های استفاده از آنها را در طراحی سیستم‌های دیجیتال مشخص خواهیم کرد.

در بعضی از وسایل منطقی برنامه‌پذیر ممکن است صدها میلیون گیت مرتبط از طریق صدها هزار مسیر درونی به هم وصل شده باشند. برای این که نمودار منطقی داخلی را به فرم مختصر نشان دهیم، لازم است از یک سمبل گیت خاص قابل اعمال در منطق آرایه استفاده کنیم. شکل ۱-۷ سمبل معمول و آرایه‌ای را برای گیت OR چند ورودی نشان می‌دهد. به جای داشتن چندین خط ورودی به گیت، تنها یک خط به گیت وصل می‌کنیم. خطوط ورودی عمود بر این خط بوده و از طریق فیوزها به آن وصل می‌شوند. به طریقی مشابه می‌توانیم منطق آرایه را برای گیت AND ترسیم کنیم. این نوع نمایش گرافیکی برای ورودی گیت‌ها در ترسیم نمودارهای منطقی آرایه‌ای به کار خواهد رفت.

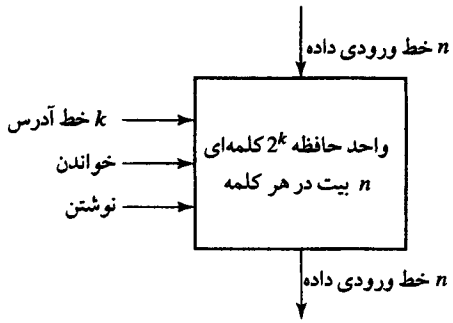
۷-۲ حافظه با دستیابی تصادفی RAM

واحد حافظه مجموعه‌ای از سلول‌های ذخیره‌سازی به همراه مدارهای مربوطه لازم برای انتقال اطلاعات به داخل یا به خارج وسیله است. زمان لازم برای هر انتقال اطلاعات و یا از هر مکان موردنظر همواره یکسان است، و به این علت نام حافظه با دستیابی تصادفی به صورت خلاصه RAM به کار رفته است.

یک واحد حافظه اطلاعات دودویی را به صورت گروهی از بیت‌ها به نام کلمه ذخیره می‌نماید. کلمه حافظه گروهی از بیت‌هاست که به صورت واحد حافظه وارد یا خارج می‌شود. کلمه حافظه ترکیبی از 1ها و 0هاست و ممکن است، یک عدد، یک دستور، یک یا چند کاراکتر الفبا عددی، یا هر اطلاعات دودویی کد شده را نمایش دهد. یک گروه هشت بیتی را بایت نامند.

بسیاری از کامپیوترها کلماتی را به کار می‌برند که طولشان مضربی از هشت بیت، یا بایت است. بنابراین کلمه 16 بیتی دو بایت، کلمه 32 بیتی از چهار بایت ساخته می‌شود. ظرفیت یک واحد حافظه براساس تعداد کل بایت‌هایی است که می‌تواند ذخیره کند.

محاوره بین حافظه و محیط از طریق خطوط داده ورودی و خروجی، خطوط انتخاب آدرس و خطوط کنترلی که جهت انتقال را مشخص می‌کند، صورت می‌گیرد. نمودار بلوکی واحد حافظه در



شکل ۲-۷. دیاگرام بلوکی یک واحد حافظه

شکل ۲-۷ ملاحظه می‌گردد. n خط داده ورودی اطلاعات ذخیره شونده را مهیا می‌سازند و n خط داده خروجی هم اطلاعات خروجی را تهیه می‌نمایند. k خط آدرس کلمه خاص انتخابی را در میان چند کلمه مشخص می‌نماید. دو ورودی کنترل، جهت انتقال موردنظر را تعیین می‌کنند. ورودی نوشتن موجب می‌شود تا وارده به داخل حافظه برود و ورودی خواندن سبب انتقال داده به خارج از حافظه می‌گردد.

واحد حافظه با تعداد کلمات و تعداد بیت‌ها در هر کلمه مشخص می‌شود. خطوط آدرس یک کلمه خاص را انتخاب می‌کنند. به هر کلمه در حافظه یک عدد شناسایی به نام آدرس تخصیص می‌یابد، که از 0 شروع و تا 2^k-1 ادامه دارد و در آن k تعداد خطوط آدرس است. انتخاب یک کلمه خاص در حافظه با اعمال آدرس k بیتی به خطوط آدرس صورت می‌گیرد. یک دیکدر این آدرس را پذیرفته و مسیرهای لازم را برای انتخاب کلمه موردنظر باز می‌کند. حافظه‌ها از لحاظ سایز بسیار متغیرند و ممکن است از 1024 کلمه، که به 10 خط آدرس نیاز دارد، تا 2^{32} کلمه که 32 خط آدرس را داراست در دسترس باشند. مرسوم است که تعداد کلمات (یا بایت) در حافظه با یکی از حروف K (کیلو)، M (مگا)، یا G (گیگا) ذکر شود. K برابر 2^{10}، M برابر با 2^{20} و G برابر 2^{30} است. بنابراین $64K=2^{16}$، $2M=2^{21}$ و $4G=2^{32}$ خواهد بود.

به عنوان مثال واحد حافظه‌ای با ظرفیت 2K کلمه و 16 بیت در هر کلمه را در نظر بگیرید. چون $1K = 1024 = 2^{10}$ و 16 بیت هم دو بایت را تشکیل می‌دهد، می‌توان گفت که حافظه قادر است $2K = 2048$ بایت را در خود جای دهد. شکل ۳-۷ محتوای سه کلمه اول و سه کلمه آخر این حافظه را

آدرس حافظه		محتویات حافظه
دودویی	دهدهی	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

شکل ۳-۷. محتویات یک حافظهٔ 16×1024

نشان می‌دهد. کلمات با آدرس دهدهی از 0 تا 1023 شناخته می‌شوند. آدرس دودویی معادل شامل 10 بیت است. اولین آدرس با ده عدد 0 مشخص می‌شود و آخرین آدرس هم با ده عدد 1 معین می‌گردد. دلیل این است که 1023 در دودویی برابر 1111111111 می‌باشد. یک کلمه در حافظه با آدرس دودویی‌اش انتخاب می‌گردد. وقتی کلمه‌ای نوشته و یا خوانده شود، حافظه روی هر 16 واحد به صورت یکجا عمل می‌نماید.

حافظه 10 × 64K دارای 10 بیت خط آدرس و 16 بیت در هر کلمه است. به عنوان مثالی دیگر، یک حافظه 10 × 64K دارای 16 بیت آدرس (چون $64K = 2^{16}$ است) و هر کلمه متشکل از 10 بیت می‌باشد. تعداد بیت‌های آدرس در حافظه وابسته به تعداد کل کلماتی است که قابل ذخیره شدن در حافظه می‌باشد و مستقل از تعداد بیت‌های هر کلمه است. تعداد بیت‌ها در آدرس از رابطه $2^k \geq m$ بدست می‌آید، که m تعداد کل کلمات و k تعداد بیت‌های آدرس لازم برای برقراری رابطه فوق است.

اعمال نوشتن و خواندن

دو عملی که یک حافظه RAM می‌تواند انجام دهد عبارتند از نوشتن و خواندن. سیگنال نوشتن عمل انتقال به داخل و سیگنال خواندن یک عمل انتقال به بیرون را مشخص می‌نماید. برای پذیرش یکی از این سیگنال‌های کنترل، مدارهای درونی داخل حافظه عمل موردنظر را مهیا می‌سازند. مراحل مربوط به وارد کردن یک کلمه به حافظه به قرار زیر است:

۱- آدرس دودویی کلمه موردنظر به خطوط آدرس اعمال می‌گردد.

۲- بیت‌های داده‌ای که باید در حافظه ذخیره شوند به خطوط ورودی داده اعمال می‌گردند.

۳- ورودی نوشتن فعال می‌شود.

آنگاه واحد حافظه بیت‌ها را از خطوط ورودی داده برداشته و آنها را در مکانی که به وسیله خطوط آدرس معین می‌گردد ذخیره می‌نماید.

مراحل مربوط به خارج کردن یک کلمه از حافظه به قرار زیر است.

۱- آدرس دودویی کلمه موردنظر به خطوط آدرس اعمال می‌شود.

۲- ورودی خواندن فعال می‌گردد.

آنگاه واحد حافظه کلمه انتخابی به وسیله آدرس را برداشته و به خطوط داده خروجی اعمال می‌کند. محتوای کلمه انتخابی پس از خوانده شدن تغییر نمی‌نماید.

گاهی قطعات حافظه تجاری موجود در تراشه‌های مدار مجتمع دو خط کنترل خواندن و نوشتن را با آرایش‌های متفاوت دارا هستند. در عوض داشتن دو خط خواندن و نوشتن جداگانه برای کنترل دو عمل، بسیاری از مدارهای مجتمع دو ورودی کنترل مجزا دارند: یکی ورودی قطعه را انتخاب می‌کند و دیگری عمل آن را کنترل می‌نماید. عملیات حافظه‌ای که از این ورودی‌های کنترل حاصل می‌شود در جدول ۱-۷ مشخص شده‌اند.

فعال‌ساز حافظه (گاهی به آن انتخاب‌گر تراشه گویند) برای توانا کردن یک تراشه خاص حافظه از میان چند تراشه موجود به کار می‌رود. وقتی که فعال‌ساز حافظه غیرفعال باشد، تراشه حافظه انتخاب

جدول ۱-۷. ورودی‌های کنترل به تراشه حافظه

عملکرد حافظه	نوشتن / خواندن	فعال ساز حافظه
هیچ	X	0
نوشتن در کلمه منتخب	0	1
خواندن از کلمه منتخب	1	1

نشده و هیچ عملی اجرا نمی‌گردد. پس از فعال شدن ورودی نوشتن/ خواندن عمل مورد اجرا را مشخص خواهد کرد.

توصیف حافظه در HDL

حافظه در Verilog HDL به وسیله آرایه‌ای از ثبات‌ها مدل‌سازی می‌شود. حافظه با کلمه کلید reg و با استفاده از آرایه دوبعدی اعلان می‌گردد. اولین عدد در آرایه تعداد بیت‌ها در یک کلمه و دومی تعداد کلمات در حافظه است. مثلاً یک حافظه 1024 کلمه‌ای با 16 بیت در هر کلمه به صورت زیر اعلان می‌شود.

```
reg[15:0] memword[0:1023];
```

این عبارت یک آرایه 1024 ثباتی را با 16 بیت در هر یک توصیف می‌کند. عدد داخل memword بیانگر تعداد کل کلمات حافظه بوده و برابر با آدرس حافظه است. مثلاً memword[512] به کلمه حافظه‌ای 16 بیتی در آدرس 512 اشاره می‌نماید. عمل یک واحد حافظه در مثال ۱-۷ HDL تشریح شده است. حافظه دارای 64 کلمه چهار بیتی است. دو ورودی کنترل وجود دارد: ReadWrite و Enable. خطوط DataIn و DataOut هر یک چهار بیتی‌اند. ورودی Address باید شش بیتی ($2^6 = 64$)

مثال ۱-۷، HDL

```
//Read and write operations of memory.
//Memory size is 64 words of 4 bits each.
 module memory (Enable,ReadWrite,Address,DataIn,DataOut);
    input  Enable,ReadWrite;
    input [3:0] DataIn;
    input [5:0] Address;
    output [3:0] DataOut;
    reg [3:0] DataOut;
    reg [3:0] Mem [0:63];          //64 x 4 memory
    always @ (Enable or ReadWrite)
      if (Enable)
      if (ReadWrite)
      DataOut = Mem[Address];  //Read
      else
      Mem[Address] = DataIn;   //Write
      else DataOut = 4'bz;          //High impedance state
endmodule
```

باشد، حافظه به صورت یک آرایه ثباتی دوبعدی با Mem اعلان می‌شود که آدرس 64 کلمه را مشخص می‌نماید. عملیات حافظه هنگامی شروع می‌شود که ورودی Enable فعال باشد. ورودی ReadWrite نوع عمل را مشخص می‌سازد. اگر ReadWrite = 1 باشد، حافظه عمل خواندن را با عبارت زیر نمادین می‌کند.

DataOut ← Mem [Address];

این عمل موجب می‌شود تا چهار بیت کلمه حافظه انتخابی مشخص شده با Address به خطوط DataOut منتقل شود. اگر ReadWrite = 0 باشد، حافظه عمل نوشتن را که با عبارت زیر نشان داده شده اجرا خواهد کرد.

Mem [Address] ← DataIn;

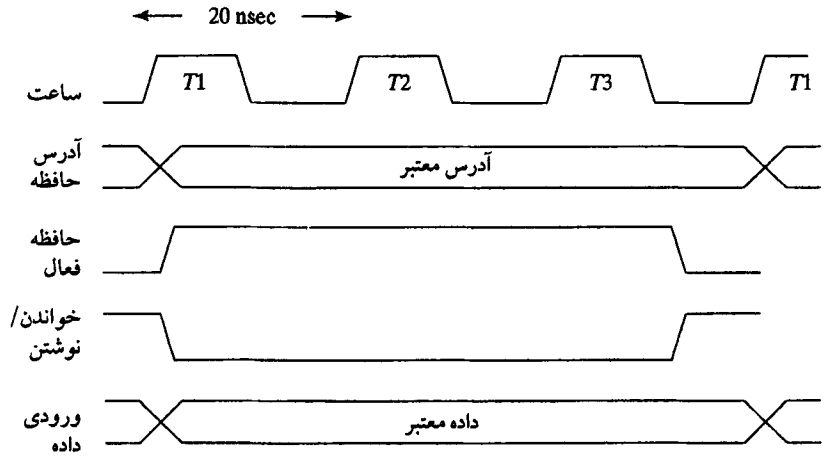
این عبارت سبب انتقال 4 بیت از خطوط DataIn به داخل کلمه حافظه انتخاب شده با Address می‌شود. وقتی که Enable=0 شود، حافظه از کار افتاده و فرض می‌شود که خروجی‌ها به حالت امپدانس بالا می‌روند. این حالت با کلمه کلیدی z مشخص شده و نشان می‌دهد که حافظه دارای خروجی‌های سه حالته است.

شکل موج‌های زمان‌بندی

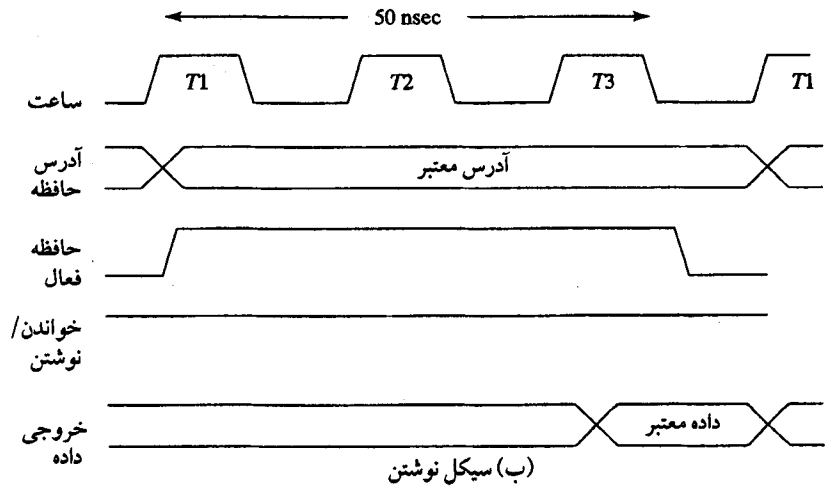
عملکرد یک واحد حافظه با یک وسیله خارجی مانند واحد مرکزی پردازش (CPU) کنترل می‌گردد. CPU معمولاً با ساعتش معرفی می‌شود. با این وجود، حافظه دارای ساعت درونی نیست. عمل خواندن و نوشتن نیز با خطوط کنترل مشخص می‌گردد. زمان دستیابی یک حافظه زمانی است که برای انتخاب یک کلمه و خواندن آن لازم است. زمان سیکل یک حافظه زمان لازم برای تکمیل نوشتن می‌باشد. CPU باید سیگنال‌های کنترل حافظه را تولید کند به نحوی که عملیات ساعت‌دار داخلی با عملیات خواندن و نوشتن همزمان شود. این بدان معنی است که زمان دستیابی و زمان سیکل یک حافظه باید در فاصله‌ای از زمان که مضربی از سیکل‌های ساعت CPU است قرار گیرند.

به عنوان مثال فرض کنید که یک CPU با یک ساعت 50MHz کار کند که در نتیجه پریود آن 20ns خواهد شد. اکنون تصور کنید که CPU با حافظه‌ای تبادل داده نماید که زمان دستیابی‌اش و زمان سیکل آن از 50ns تجاوز نکند. این بدان معنی است که سیکل نوشتن کلمه در حافظه باید در یک دوره 50ns قرار داشته باشد و نیز سیکل خواندن، داده انتخابی را در 50ns یا کمتر خارج کند. (البته دو عدد همیشه یکی نیستند). چون پریود سیکل CPU برابر 20ns است، لازم است دو ونیم و یا سه پالس ساعت برای هر تقاضای حافظه به کار گرفته شود.

زمانبندی حافظه که در شکل ۴-۷ دیده می‌شود برای یک CPU با ساعت 50MHz و حافظه‌ای با زمان سیکل حداکثر 50ns ترسیم شده است. سیکل نوشتن در بخش (الف) سه سیکل 20ns، T_1، T_2 و T_3 را نشان می‌دهد. برای عمل نوشتن، CPU آدرس و داده ورودی را برای حافظه فراهم می‌نماید. این کار در آغاز T_1 انجام می‌شود. (دو خط متقاطع در خطوط آدرس و داده، تغییر احتمالی در مقدار خطوط چندگانه را بیان می‌دارد.) پس از تثبیت سیگنال‌های آدرس، سیگنال‌های فعال‌ساز و نوشتن/ خواندن



(الف) سیکل خواندن



(ب) سیکل نوشتن

شکل ۴-۷. شکل موج‌های زمانبندی سیکل حافظه

فعال می‌گردند تا به این ترتیب داده‌ها در دیگر حافظه‌ها تخریب نگردند. سیگنال فعال‌ساز حافظه به سطح بالا و سیگنال خواندن/ نوشتن به سطح پایین سوئیچ می‌کنند تا عمل نوشتن انجام شود. دو سیگنال کنترل باید حداقل به مدت 50ns با ثُبات باقی بمانند. سیگنال‌های داده و آدرس باید برای مدت کوتاهی پس از غیرفعال شدن سیگنال‌های کنترل، بدون تغییر باقی بمانند. در پایان سومین سیکل ساعت، عمل نوشتن در حافظه تکمیل شده و CPU می‌تواند در سیکل T_1 بعدی دوباره به حافظه دست یابد.

سیکل خواندن در شکل ۴-۷ (ب) دارای آدرسی است که به وسیله CPU برای حافظه مهیا شده

است. سیگنال‌های فعال‌ساز حافظه و خواندن/ نوشتن برای خواندن باید در سطح بالا باشند. حافظه باید، داده کلمه انتخابی به وسیله آدرس را در کمتر از 50ns از لحظه فعال شدن حافظه، روی خطوط داده خروجی قرار دهد. CPU در حین گذر منفی T_3 ، داده را به یکی از ثبات‌های داخلی انتقال می‌دهد. سیکل T_1 بعدی برای یک تقاضای حافظه دیگر در اختیار است.

انواع حافظه‌ها

طرز دستیابی به یک سیستم حافظه به نوع قطعات به‌کار رفته بستگی دارد. در یک حافظه RAM ، مکان‌های حافظه را می‌توان در فضا مجزا از یکدیگر تصور کرد، که در آن هر کلمه یک مکان خاص را اشغال کرده است. در یک حافظه با دستیابی ترتیبی، اطلاعات ذخیره شده در محیط بلافاصله قابل دسترسی نیستند و بلکه پس از مدتی بدست می‌آیند. یک دیسک یا نوار مغناطیسی حافظه‌ای از این نوع است. هر مکان از حافظه از زیر هدهای خواندن و نوشتن به ترتیب عبور می‌کنند، ولی اطلاعات فقط وقتی خوانده می‌شود که کلمه مورد تقاضا فرا برسد. در یک حافظه RAM ، زمان دستیابی مستقل از مکان یک کلمه خاص بوده و همیشه کوتاه است. در حافظه با دستیابی ترتیبی، زمان دستیابی به یک کلمه به محل کلمه نسبت به محل هد خواندن وابسته است، بنابراین زمان دستیابی متغیر خواهد بود.

واحدهای مدار مجتمع RAM در دو نوع موجودند: نوع استاتیک و نوع دینامیک. RAM استاتیک (SRAM) اساساً از لچ‌های درونی تشکیل شده و اطلاعات دودویی را ذخیره می‌نمایند. اطلاعات ذخیره شده مادامی که تغذیه به واحد اعمال شده باشد معتبر خواهد ماند. RAM دینامیک (DRAM) اطلاعات دودویی را به فرم بارهای الکتریکی در خازن ذخیره می‌نماید. خازن‌ها به وسیله ترانزیستورهای MOS در داخل تراشه ساخته می‌شوند. بار ذخیره شده در خازن‌ها به تدریج تخلیه می‌گردد و بنابراین با تازه‌سازی حافظه دینامیکی باید خازن‌ها را بازسازی کرد. عمل تازه‌سازی هر چند میلی ثانیه یک بار با چرخش در کلمات برای بازسازی خازن‌ها انجام می‌شود. حافظه DRAM مصرف انرژی کمتر و ظرفیت ذخیره‌سازی بیشتری در تراشه حافظه دارد. SRAM کاربرد ساده‌تر و سیکل خواندن و نوشتن کوتاه‌تری را داراست.

واحدهای حافظه‌ای راکه به هنگام قطع برق اطلاعات خود را از دست می‌دهند حافظه فرّار می‌گویند. مدارهای مجتمع RAM ، چه استاتیک و چه دینامیک، از این رده محسوب می‌شوند زیرا سلول‌های دودویی برای حفظ اطلاعات نیاز به تغذیه بیرونی دارند. برعکس، یک حافظه غیر فرّار، مانند دیسک مغناطیسی، اطلاعات ذخیره شده خود را حتی بعد از حذف تغذیه حفظ می‌نماید. دلیل این است که داده ذخیره شده در عناصر مغناطیسی با جهت مغناطیسی شدن آن تعیین می‌گردد، که پس از حذف منبع تغذیه، باز هم باقی خواهند ماند. نوع دیگر حافظه غیر فرّار، حافظه فقط خواندنی ROM است. خاصیت غیر فرّاری در کامپیوترهای دیجیتال به هنگام خاموشی لازم می‌باشد. داده‌ها و برنامه‌هایی که تغییر نمی‌کنند در ROM ذخیره می‌شوند. برنامه‌های بزرگ معمولاً روی دیسک‌های مغناطیسی نگهداری می‌شود. وقتی که تغذیه روشن شود، کامپیوتر برنامه‌های ROM را می‌تواند استفاده کند. دیگر

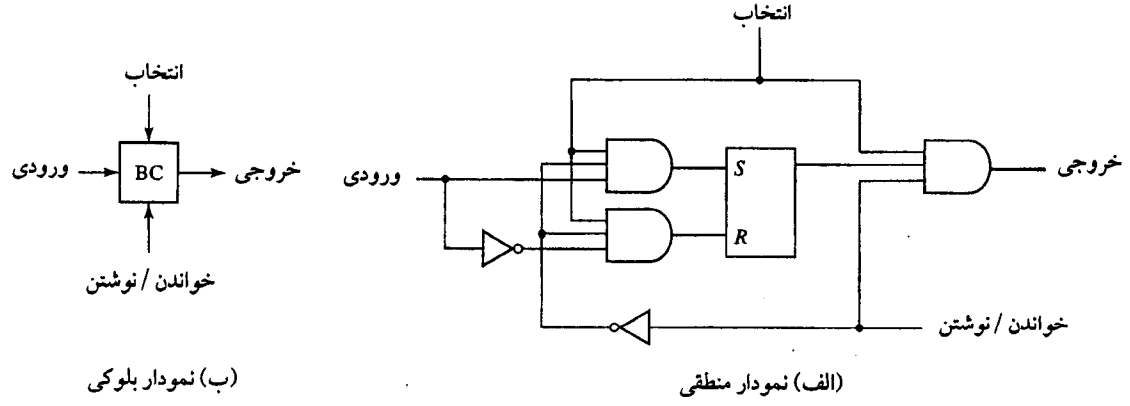
برنامه‌های موجود در یک دیسک مغناطیسی پس از آن به RAM کامپیوتر انتقال می‌یابد. قبل از خاموش کردن کامپیوتر اطلاعات موجود در RAM جهت حفظ و نگهداری به دیسک منتقل می‌گردد.

۷-۳ دیکدکردن حافظه

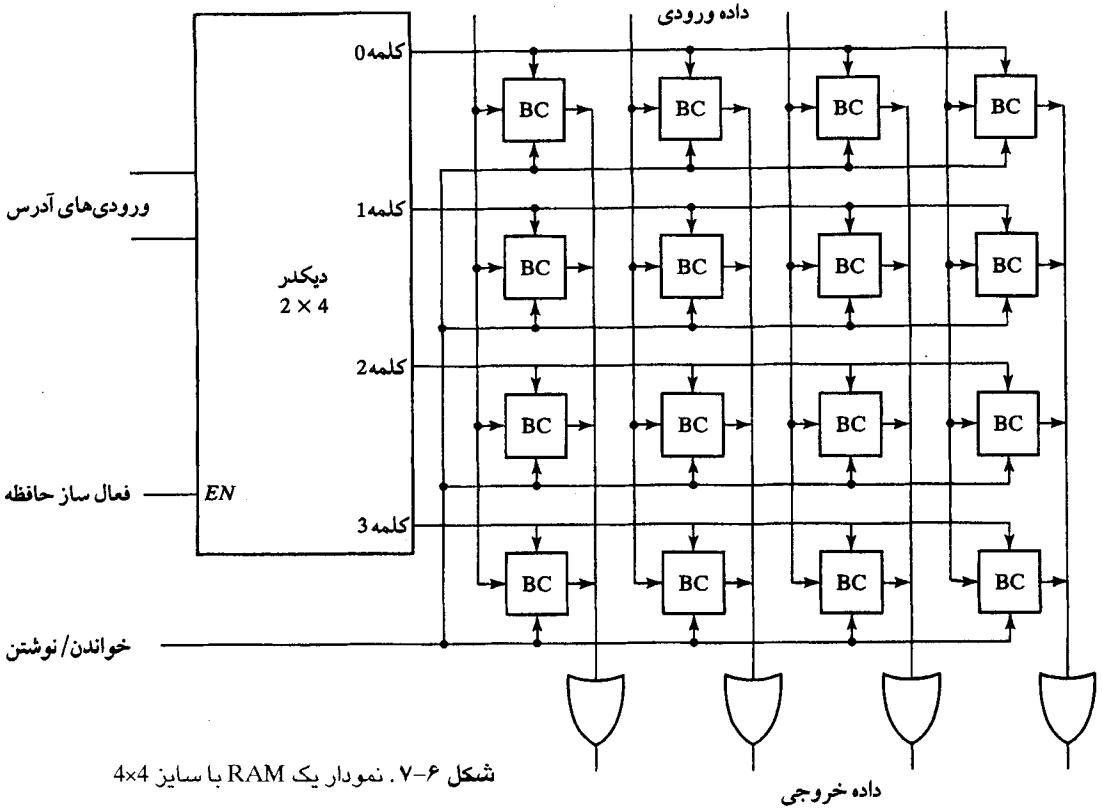
علاوه بر عناصر ذخیره‌سازی در یک واحد حافظه، برای انتخاب کلمه حافظه مشخص شده با آدرس ورودی، به یک دیکدر نیاز است. در این بخش ساختمان داخلی یک حافظه با دستیابی تصادفی را ارائه می‌کنیم و عملکرد دیکدر را نشان خواهیم داد. برای این که بتوان تمام حافظه را در یک نمودار جای داد واحد حافظه دارای 16 بیت حافظه با 4 بیت در هر کلمه فرض شده است. مثالی از آرایش دیکد کردن دو بعدی، برای درک بهتر حافظه‌های بزرگ‌تر آورده شده است. آنگاه مولتی پلکس کردن آدرس را که در مدارهای مجتمع DRAM به کار می‌رود نشان خواهیم داد.

ساختار درونی

ساختار درونی یک حافظه با دستیابی تصادفی m کلمه‌ای با n بیت در هر کلمه از یک مجموعه ذخیره‌سازی $m \times n$ و مدارهای دیکدکردن برای انتخاب هر کلمه تشکیل شده است. سلول ذخیره‌سازی دودویی بلوک ساختاری پایه واحد حافظه است. مدار معادل یک سلول دودویی که یک بیت از اطلاعات را ذخیره کند در شکل ۵-۷ نشان داده شده است. بخش ذخیره‌سازی آن نیز یک لچ SR به همراه گیت‌های مربوطه است. در واقع سلول یک مدار الکترونیکی ساخته شده از چهار تا شش ترانزیستور می‌باشد. با این وجود بهتر است که آن را با سمبل‌های منطقی مدل‌سازی نماییم. یک سلول ذخیره‌سازی دودویی باید خیلی کوچک باشد تا بتوان سلول‌های بیشتری را در یک فضای کوچک در تراشه جای داد. سلول دودویی یک بیت را در هر لچ داخلی ذخیره می‌نماید. ورودی انتخاب، سلول را برای خواندن یا نوشتن فعال می‌کند و ورودی خواندن/ نوشتن نوع عملکرد سلول را پس از انتخاب تعیین می‌نماید. یک 1 در ورودی خواندن/ نوشتن عمل خواندن را با ایجاد مسیری از لچ به پایانه بیرونی فراهم می‌نماید. یک 0 در ورودی خواندن/ نوشتن عمل نوشتن را با ایجاد مسیری از پایانه ورودی به لچ ممکن می‌سازد.



شکل ۵-۷. سلول حافظه



شکل ۶-۷. نمودار یک RAM با سایز 4×4

ساختار منطقی یک RAM کوچک در شکل ۶-۷ نشان داده شده است. این مدار چهار کلمه چهار بیت و جمعاً 16 سلول دودویی دارد. بلوک‌های کوچک برچسب خورده با BC، سلول دودویی را با سه ورودی و یک خروجی طبق شکل ۵-۷ (ب) نمایش می‌دهند. یک حافظه چهار کلمه‌ای به دو خط آدرس نیاز دارد. دو ورودی آدرس از یک دیکدر 4 × 2 برای انتخاب یکی از چهار کلمه عبور می‌نمایند. دیکدر با ورودی فعال‌ساز حافظه، فعال می‌شود. وقتی فعال‌ساز حافظه 0 باشد، همه خروجی‌های دیکدر 0 و هیچ یک از کلمات حافظه انتخاب نمی‌شود. با 1 شدن انتخاب حافظه بنا به مقدار دو خط آدرس، یکی از چهار کلمه حافظه انتخاب می‌گردد. پس از انتخاب کلمه، ورودی خواندن/ نوشتن عمل را معین می‌نماید. در حین خواندن، چهار بیت کلمه انتخابی از گیت‌های OR به پایانه‌های خروجی منتقل می‌شوند (توجه کنید که گیت‌های OR مطابق با قرارداد منطق آرایه‌ای شکل ۱-۷ ترسیم شده‌اند). در هنگام نوشتن، داده موجود در خطوط ورودی به چهار سلول دودویی کلمه انتخابی وارد می‌گردند. سلول‌های دودویی انتخاب نشده غیرفعال شده و مقدار قبلی آنها دست نخورده باقی می‌ماند. وقتی ورودی انتخاب حافظه وارده به دیکدر 0 شود، هیچ یک از کلمات انتخاب نمی‌شوند و محتوای همه

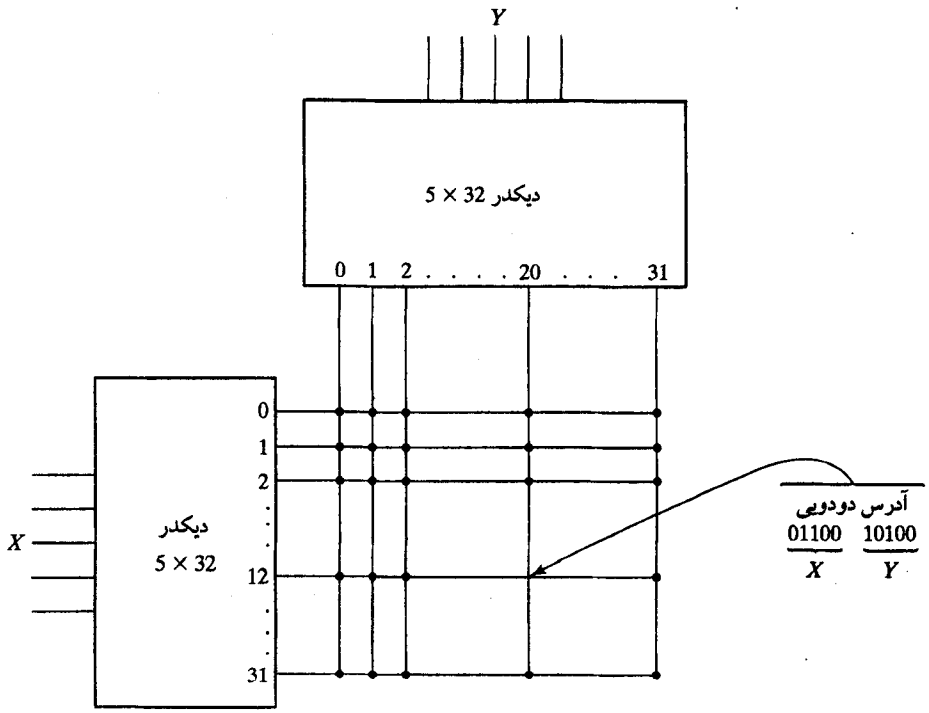
سلول‌ها مستقل از مقدار ورودی خواندن/ نوشتن بدون تغییر باقی می‌مانند.

حافظه‌های RAM تجاری می‌توانند تا هزاران کلمه در هر تراشه و از 1 تا 64 بیت در هر کلمه داشته باشند. ساختار منطقی یک حافظه با ظرفیت بالا توسعه مستقیمی از آرایش شکل ۶-۷ است. حافظه‌ای با 2^k کلمه n بیتی در هر کلمه نیاز به k خط آدرس دارد تا وارد دیکدر $k \times 2^k$ گردد. هر یک از خروجی‌های دیکدر یک کلمه n بیتی را برای نوشتن و یا خواندن انتخاب می‌نماید.

دیکد کردن متقارن

یک دیکدر با k ورودی و 2^k خروجی به 2^k گیت AND و k ورودی در هر گیت نیاز دارد. با انتخاب طرح دوبعدی و استفاده از دو دیکدر می‌توان تعداد کل گیت‌ها و تعداد ورودی‌های گیت‌ها را کاهش داد. ایده اولیه در دیکد کردن دوبعدی، آرایش سلول‌های حافظه در یک آرایه است که تا حد ممکن به یک مربع نزدیک باشد. در این آرایش، دو دیکدر با k/2 ورودی در عوض دیکدر k ورودی به کار می‌رود. یک دیکدر انتخاب سطر و دیگری انتخاب ستون را در آرایش ماتریسی دوبعدی به عهده دارد.

الگوی انتخاب دوبعدی برای یک حافظه 1k کلمه‌ای در شکل ۷-۷ نشان داده شده است. در عوض استفاده از یک دیکدر 1024 × 10، دو دیکدر 32 × 5 به کار رفته است. هنگام استفاده از دو دیکدر، 64 گیت AND و با 5 ورودی در هر کدام لازم می‌باشد. پنج بیت باارزش‌تر آدرس به ورودی X و پنج بیت



شکل ۷-۷. ساختار دیکد کردن دو بعدی برای حاففظه ۱K کلمه‌ای

کم‌ارزش‌تر به ورودی Y می‌رود. هر کلمه در آرایه حافظه با تلاقی یکی از 32 سطر، با یکی از 32 ستون از 1024 کلمه موجود، انتخاب می‌گردد. توجه کنید که هر تقاطع یک کلمه را با هر تعداد بیت نشان می‌دهد.

به عنوان مثال، کلمه‌ای با آدرس 404 را در نظر بگیرید. معادل دودویی 10 بیت عدد 404 برابر با 10100 01100 است. در نتیجه X = 01100 (12 دودویی) و Y = 10100 (دودویی 20) خواهد بود. کلمه n بیتی انتخابی در خروجی شماره 12 دیکدر X و خروجی شماره 20 دیکدر Y قرار دارد. همه بیت‌های کلمه برای خواندن یا نوشتن به کار می‌رود.

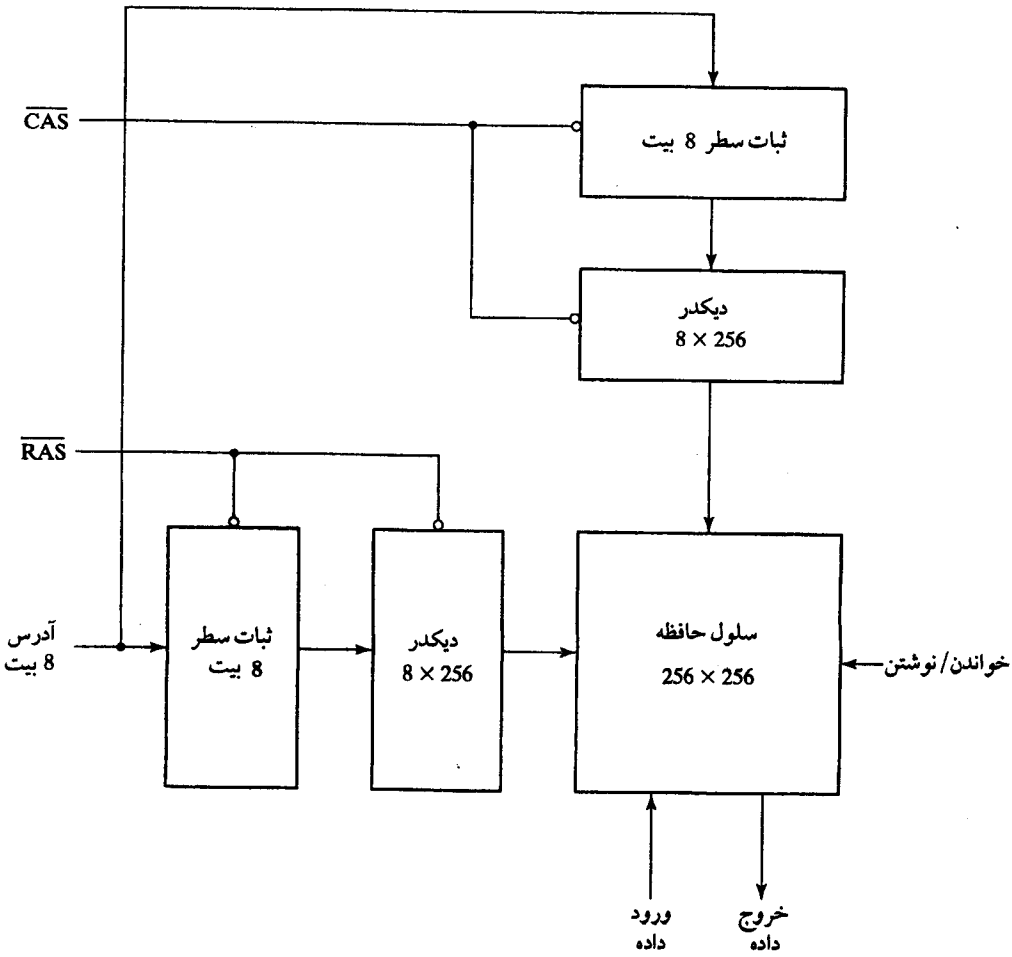
مولتی پلکس کردن آدرس

سلول حافظه SRAM که در شکل ۵-۷ مدل‌سازی شد معمولاً دارای شش ترانزیستور است. در ساخت حافظه‌های با ظرفیت بالاتر، لازم است تعداد ترانزیستورها را در هر سلول کاهش دهیم. سلول DRAM دارای یک ترانزیستور MOS و یک خازن است. بار ذخیره شده روی خازن با زمان تخلیه می‌شود و بنابراین حافظه به صورت پریودیک باید تازه‌سازی گردد. در DRAM‌ها به دلیل سادگی ساختار، چگالی سلول‌ها چهار برابر چگالی SRAM است. این قابلیت موجب می‌شود تا ظرفیت آن نسبت به حافظه SRAM چهار برابر گردد. قیمت هر بیت DRAM سه یا چهار برابر کمتر از SRAM می‌باشد. صرفه‌جویی دیگر به دلیل توان لازم کمتر در سلول‌های DRAM است. این مزایا موجب شده است تا DRAM در حافظه‌های بزرگ، تکنولوژی ارجح باشد. تراشه‌های DRAM با ظرفیت‌های 64K تا 256M بیت وجود دارند. بسیاری از DRAM‌ها سایز کلمه 1 بیتی دارند، بنابراین برای افزایش سایز کلمه باید چندین تراشه با هم ترکیب گردند.

به دلیل ظرفیت زیاد، دیکد کردن آدرس به صورت آرایه دوبعدی بوده و حافظه‌های بزرگتر حتی چندین آرایه دارند. برای کاهش تعداد پایه‌های بسته تراشه، طراحان از مولتی پلکس کردن آدرس استفاده می‌کنند که به وسیله یک مجموعه از پایه‌های ورودی آدرس عناصر آدرس را در خود جای دهند. در آرایه دوبعدی آدرس به دو بخش تقسیم شده و در زمان‌های مختلف بخش سطر آدرس ابتدا و سپس بخش ستون اعمال می‌گردند. چون برای هر دو بخش آدرس از یک مجموعه پایه استفاده می‌شود، سایز بسته به شدت کاهش می‌یابد.

برای تشریح ایده مولتی پلکس کردن از حافظه 64K کلمه‌ای استفاده شده است. نمودار آرایش دیکد کردن در شکل ۸-۷ دیده می‌شود. حافظه متشکل از آرایه دوبعدی سلول‌ها با 256 سطر در 256 ستون می‌باشد که جمعاً $64K = 2^{16} = 2^8 \times 2^8$ خواهد بود. تنها یک خط ورود داده، و یک کنترل خواندن نوشتن وجود دارد. هشت بیت ورودی آدرس و دو آگاه‌گر آدرس موجود است. آگاه‌گرهای آدرس برای فعال کردن ثبات‌های مربوطه‌شان در سطر و ستون در نظر گرفته شده‌اند. آگاه‌گر آدرس سطر (RAS)، ثبات 8 بیت سطر را فعال می‌کند و آگاه‌گر ستون CAS، ثبات 8 بیتی ستون را فعال می‌نماید. خط بار روی سمبل آگاه‌گر به معنی فعال شدن ثبات‌ها در سطح منفی سیگنال است.

16 بیت آدرس با استفاده از RAS و CAS طی دو مرحله به DRAM اعمال می‌شود. در آغاز هر دو



شکل ۸-۷. مولتی پلکس کردن آدرس برای یک 64K DRAM

آگاه‌گر در حالت 1 هستند. آدرس سطر 8 بیتی به ورودی‌های RAS اعمال شده و RAS = 0 می‌گردد. این موجب خواهد شد تا آدرس سطر در ثبات آدرس سطر بار شود. RAS دیکدر سطر را هم فعال می‌کند به نحوی که آدرس سطر دیکد شده و یکی از سطرهای آرایه انتخاب گردد. پس از گذشت زمان نشست انتخاب سطر، RAS به سطح 1 باز می‌گردد. سپس آدرس ستون 8 بیتی به ورودی‌های آدرس اعمال شده و CAS به 0 برده می‌شود. این موجب می‌گردد تا آدرس ستون در ثبات آدرس ستون بار شود و دیکدر ستون را فعال نماید. در این هنگام دو بخش آدرس در ثبات‌های مربوطه هستند، و دیکدرها آنها را رمزگشایی کرده‌اند تا یک سلول انتخاب شود، و آنگاه عمل خواندن یا نوشتن بر روی سلول قابل اجرا خواهد بود. قبل از آغاز اجرای عمل روی حافظه بعدی، CAS به 1 باز گردانده می‌شود.

۷-۴ تشخیص و تصحیح خطا

سطح پیچیدگی یک آرایه حافظه ممکن است هر از چندگاه خطاهایی را در ذخیره و بازیابی اطلاعات دودویی موجب شود. قابلیت اطمینان یک واحد حافظه را با استفاده از کدهای تشخیص و تصحیح خطا می‌توان بهبود بخشید. مرسوم‌ترین روش تشخیص خطا، بیت توازن است (بخش ۸-۳ ملاحظه شود). پس از خواندن کلمه از حافظه، توازن آن چک می‌شود. اگر توازن بیت‌های خوانده شده صحیح باشد کلمه داده پذیرفته خواهد شد. اگر حاصل توازن چک شده معکوس باشد، خطایی اعلام می‌شود ولی قابل اصلاح نیست.

یک کد تصحیح خطا، بیت‌های تست توازن چند بیتی تولید می‌کند که همراه با کلمه داده در حافظه ذخیره می‌گردد. هر بیت تست توازن مربوط به گروهی از بیت‌ها در کلمه داده است. وقتی که کلمه از حافظه بازخوانی می‌شود، بیت‌های توازن نیز از حافظه خوانده شده و با مجموعه جدیدی از بیت تست تولید شده که از داده خوانده شده بدست می‌آید مقایسه می‌گردد. اگر بیت‌های تست صحیح باشند، بیانگر عدم وقوع خطاست. اگر بیت‌های تست با توازن ذخیره شده همخوانی نداشته باشد، یک الگوی منحصر به فردی به نام نشانه (syndrome) تولید می‌کنند که می‌توان از آن برای تشخیص بیت خطا استفاده کرد. هر وقت یک خطا در یک بیت رخ دهد، به این معنی است که مقدار آن در حین خواندن و نوشتن از 1 به 0 و یا از 0 به 1 تغییر یافته است. اگر بیت خطای خاصی شناسایی شد، می‌توان با متمم‌کردن آن بیت خطا را اصلاح کرد.

کدهمینگ

یکی از رایج‌ترین کدهای تصحیح خطا که در حافظه‌های RAM به کار می‌رود به وسیله R.W.Hamming پیشنهاد گردید. در کد همینگ، k بیت توازن به کلمه داده n بیتی اضافه می‌شود تا یک کد جدید n + k بیتی بدست آید. مکان بیت‌ها از 1 تا n + k شماره‌گذاری می‌شود. مکان‌هایی که توانی از 2 هستند برای بیت‌های توازن رزرو شده‌اند. بیت‌های باقیمانده، بیت‌های داده هستند. کد می‌تواند با کلماتی با هر طول به کار برده می‌شود. قبل از ارائه مشخصات عمومی کد، عمل را با یک داده هشت بیتی تشریح می‌نماییم.

به عنوان مثال کلمه داده 8 بیتی 11000100 را در نظر بگیرید. برای کلمه هشت بیتی چهار بیت توازن را لحاظ می‌کنیم و دوازده بیت حاصل را به صورت زیر مرتب می‌نماییم.

موقعیت بیت	1	2	3	4	5	6	7	8	9	10	11	12
	P_1	P_2	1	P_4	1	0	0	P_8	0	1	0	0

چهار بیت توازن P_1، P_2، P_4 و P_8 به ترتیب در مکان‌های 1، 2، 4 و 8 قرار دارند. 8 بیت کلمه داده در بقیه مکان‌ها قرار گرفته‌اند. هر بیت توازن به طریق زیر محاسبه می‌شود.

$$P_1 = \text{XOR of bits } (3, 5, 7, 9, 11) = 1 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 0$$
$$P_2 = \text{XOR of bits } (3, 6, 7, 10, 11) = 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 = 0$$
$$P_4 = \text{XOR of bits } (5, 6, 7, 12) = 1 \oplus 0 \oplus 0 \oplus 0 = 1$$
$$P_8 = \text{XOR of bits } (9, 10, 11, 12) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$$

توجه داشته باشید که تابع XOR یک تابع فرد است. این تابع در ازاء تعداد فردی از 1 در متغیرها برابر 1 و در ازاء تعداد زوجی از 1 برابر 0 است. بنابراین، هر بیت توازن طوری تنظیم می‌شود که تعداد کل 1 ها در متغیرهای تست، از جمله بیت توازن همیشه زوج باشد.

کلمه داده هشت بیتی به همراه چهار بیت توازن به صورت یک کلمه مرکب 12 بیتی در حافظه ذخیره می‌گردد. با جایگزینی چهار بیت P در مکان‌های صحیح یک کلمه مرکب 12 بیتی بدست می‌آوریم.

موقعیت بیت :	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	1	1	1	0	0	1	0	1	0	0

وقتی که 12 بیت فوق‌الذکر از حافظه خوانده می‌شود، برای وجود خطاهای ممکن جستجو می‌گردد. توازن روی ترکیب بیت‌های فوق و بیت توازن چک شده و چهار بیت چک به طریق زیر ارزیابی می‌شود.

$$C_1 = \text{XOR of bits } (1, 3, 5, 7, 9, 11)$$
$$C_2 = \text{XOR of bits } (2, 3, 6, 7, 10, 11)$$
$$C_4 = \text{XOR of bits } (4, 5, 6, 7, 12)$$
$$C_8 = \text{XOR of bits } (8, 9, 10, 11, 12)$$

یک بیت چک 0 به معنی توازن زوج روی بیت‌های چک شده و یک 1 به معنی توازن فرد است. چون بیت‌ها با توازن زوج ذخیره شده بودند، نتیجه $C = C_8C_4C_2C_1 = 0000$ بیانگر عدم وجود خطاست. با این وجود اگر $C \neq 0$ باشد، آنگاه عدد دودویی 4 بیت تشکیل شده با بیت‌های چک یا تست، محل بیت خطا را مشخص خواهد کرد. مثلاً سه حالت زیر را در نظر بگیرید.

	موقعیت بیت :	1	2	3	4	5	6	7	8	9	10	11	12
بدون خطا		0	0	1	1	1	0	0	1	0	1	0	0
خطا در بیت 1		1	0	1	1	1	0	0	1	0	1	0	0
خطا در بیت 5		0	0	1	1	0	0	0	1	0	1	0	0

در حالت اول خطایی در کلمه 12 بیتی وجود ندارد. در حالت دوم خطایی در مکان شماره 1 وجود دارد زیرا از 0 به 1 تبدیل شده است. حالت سوم خطایی را در محل بیت 5 نشان می‌دهد که از 1 به 0 تبدیل شده است. با ارزیابی XOR بیت‌های مربوطه، چهار بیت چک را به طریق زیر محاسبه می‌کنیم:

	C_8	C_4	C_2	C_1
بدون خطا	0	0	0	0
خطا در بیت 1	0	0	0	1
خطا در بیت 5	0	1	0	1

بنابراین برای حالت بدون خطا $C = 0000$؛ با یک خطا در بیت 1، $C = 0001$؛ و با یک خطا در محل بیت 5، $C = 0101$ را داریم. عدد دودویی C، وقتی که برابر 0000 نباشد، محل بیت خطا را مشخص می‌کند. خطا را می‌توان با متمم کردن بیت مذکور تصحیح کرد. توجه کنید که خطا می‌تواند در

جدول ۲-۷. محدوده بیت‌های دادهٔ برای n بیت چک

محدوده های بیت داده ، n	تعداد بیت‌های چک ، k
2-4	3
5-11	4
12-26	5
27-57	6
58-120	7

یکی از کلمات داده و یا در یکی از بیت‌های توازن باشد.

کد همینگ را می‌توان برای کلمات داده‌ای با هر طول استفاده کرد. به طور کلی، این کد متشکل از k بیت چک، و n بیت داده بوده و مجموعاً $n + k$ بیت را شامل می‌شود. مقدار نشانه ، C ، متشکل از k بیت و در محدوده 2^k از 0 تا 2^k-1 می‌باشد. یکی از این مقادیر که معمولاً غیرصفر است، برای شناسایی عدم وجود خطا مورد استفاده قرار می‌گیرد، و 2^k-1 مقدار باقیمانده برای تعیین $n + k$ بیت خطا وارد می‌باشد. هر یک از این 2^k-1 مقدار را می‌توان برای توصیف منحصر وجود یک بیت خطا به کار برد. بنابراین محدوده 2^k-1 باید بزرگتر از $n + k$ باشد.

$$2^k - 1 \geq n + k$$

با حل n برحسب k داریم:

$$2^k - 1 - k \geq n$$

این رابطه فرمولی برای ایجاد تعداد بیت‌های داده‌ای که همراه با k بیت چک به کار می‌روند، را نشان می‌دهد. مثلاً وقتی $k = 3$ باشد، تعداد بیت‌های داده قابل استفاده برابر با $n \leq (2^3-1 - 3) = 4$ خواهد بود. برای $k = 4$ داریم $2^4-1 - 4 = 11$ و بنابراین $n \leq 11$ می‌باشد. کلمه داده ممکن است از 11 بیت کمتر باشد ولی باید حداقل 5 بیت داشته باشد، در غیر این صورت تنها 3 بیت چک نیاز خواهد بود. این خود وجود 4 بیت چک را در مثال داده 8 بیتی قبلی توجیه می‌نماید. محدوده n برای مقادیر مختلف k در جدول ۲-۷ لیست شده است.

گروه‌بندی بیت‌ها برای تولید توازن و چک کردن از لیستی از اعداد دودویی بین 0 تا 2^k-1 معین می‌شود. کم‌ارزش‌ترین بیت در اعداد دودویی 1 ، 3 ، 5 ، 7 و غیره برابر 1 است. دومین بیت باارزش‌تر در اعداد دودویی 2 ، 3 ، 6 ، 7 و غیره نیز برابر 1 است. با مقایسه این اعداد با مکان بیت‌های به کار رفته در تولید و چک بیت‌های توازن در کد همینگ رابطه‌ای بین گروه‌بندی بیت‌ها در یک کد و مکان بیت‌های 1 در رشته شمارش ملاحظه می‌شود. توجه کنید که هر گروه از بیت‌ها با عددی که توانی از 2 ، مانند 1 ، 2 ، 4 ، 8 ، 16 و غیره است آغاز می‌شود. این اعداد مکان بیت‌های توازن نیز هستند.

تصحیح تک خطا، تشخیص جفت خطا

کد همینگ فقط قادر است یک کد را شناسایی و تصحیح کند. خطاهای چندگانه قابل تشخیص نیستند. با افزودن یک بیت توازن دیگر به کلمه کد شده، کد همینگ را برای تصحیح یک خطا و تشخیص

دو خطا می‌توان به کار برد. اگر این بیت اضافی را لحاظ نماییم، آنگاه کد 12 بیتی قبلی تبدیل به $001110010100P_{13}$ خواهد شد که در آن P_{13} از XOR کردن 12 بیت دیگر ارزیابی می‌شود. این موجب می‌شود تا کد به صورت سیزده بیت 0011100101001 در آید (توازن زوج). وقتی که کد 13 بیت از حافظه خوانده شد، بیت‌های تست یا چک و نیز بیت P در تمام 13 بیت ارزیابی می‌گردند. اگر $P = 0$ باشد، توازن صحیح (توازن زوج) است، اما اگر $P = 1$ باشد، آنگاه توازن در 13 بیت غیرصحیح است (توازن فرد). چهار حالت زیر ممکن است اتفاق بیفتد.

اگر $C = 0$ و $P = 0$ باشد، خطا وجود ندارد.
اگر $C \neq 0$ و $P = 1$ باشد، یک خطا رخ داده که قابل اصلاح است.
اگر $C \neq 0$ و $P = 0$ باشد، یک جفت خطا رخ داده که قابل شناسایی است ولی قابل اصلاح نیست.
اگر $C = 0$ و $P = 1$ باشد، یک خطا در بیت P_{13} رخ داده است.

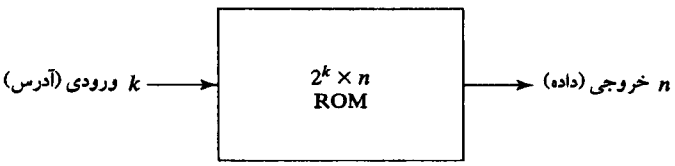
این روال ممکن است بیش از دو خطا را شناسایی کند ولی تصحیح آنها تضمین نمی‌شود.

مدارهای مجتمع از کد همینگ اصلاح شده برای تولید و تست بیت‌های توازن در اصلاح یک خطا و یا شناسایی دو خطا استفاده می‌کنند. کد اصلاح شده از آرایش توازن کاراکتری برخوردار است و در آن تعداد بیت‌های به کار رفته در محاسبه اعمال XOR بالانس شده است. نمونه‌ای از این آی‌سی که از کلمه داده 8 بیت و کلمه تست 5 بیت استفاده می‌نماید نوع 74637 است. مدارهای مجتمع دیگری هم برای کلمات 16 و 32 بیتی وجود دارند. این مدارها می‌توانند در کنار حافظه‌ها برای تصحیح یک خطا و یا تشخیص دو خطا در حین نوشتن و خواندن به کار روند.

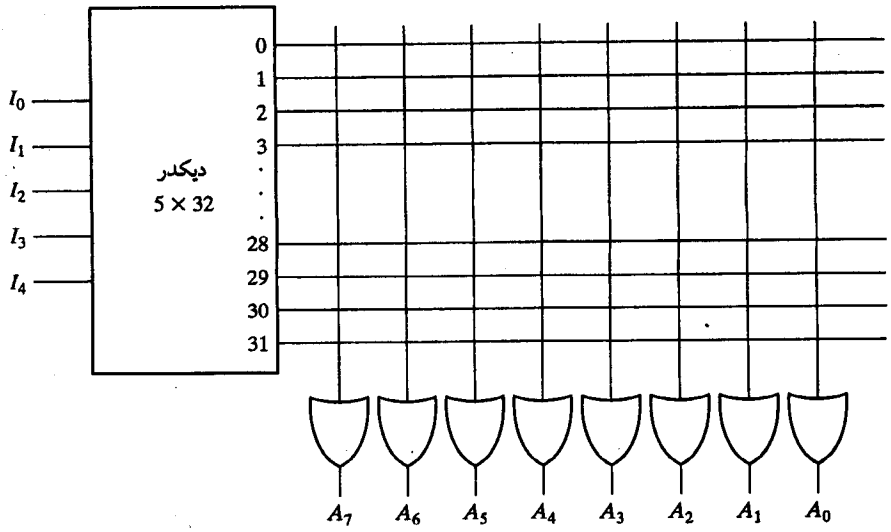
۷-۵ حافظه فقط خواندنی ROM

حافظه فقط خواندنی (ROM) اساساً یک وسیله ذخیره‌سازی است که در آن اطلاعات دودویی به طور دائم نگهداری می‌شود. اطلاعات داده باید توسط طراح مشخص شود و سپس برای ایجاد الگوی اتصالات داخلی به واحد حافظه وارد گردد. به محض ایجاد الگو، حتی با قطع و وصل منبع تغذیه، اطلاعات در آن باقی خواهد ماند.

نمودار بلوکی ROM در شکل ۹-۷ دیده می‌شود. این حافظه دارای k ورودی و n خروجی است. ورودی‌ها آدرس حافظه را مهیا می‌کنند و خروجی‌ها بیت‌های داده کلمه ذخیره شده انتخابی به وسیله آدرس را فراهم می‌نمایند. تعداد کلمات در حافظه با توجه به k بیت خط آدرس برابر 2^k کلمه است. توجه دارید که ROM فاقد ورودی داده است زیرا عمل نوشتن را انجام نمی‌دهد. تراشه‌های مدار مجتمع



شکل ۹-۷. نمودار بلوکی ROM



شکل ۱۰-۷. منطق درونی ROM 32×8

ROM دارای یک یا چند ورودی فعال‌ساز هستند و گاهی هم با خروجی سه حالته عرضه می‌گردند تاساخت آرایه‌های بزرگتر ROM را امکان‌پذیر سازند.

به عنوان مثال ROM 8 × 32 را در نظر بگیرید. واحد دارای 32کلمه 8 بیتی است. پنج خط ورودی اعداد دودویی 0 تا 31 را برای آدرس تشکیل می‌دهند. شکل ۱۰-۷ ساختار منطقی درونی ROM را نشان می‌دهد. پنج ورودی به وسیله دیکدر 32 × 5 به 32 خروجی جدا دیکد می‌شوند. هر یک از خروجی‌های دیکدر یک آدرس حافظه را نشان می‌دهند. 32 خروجی دیکدر به هر یک از هشت گیت OR متصل است. در نمودار، قرارداد منطق آرایه‌ای که برای مدارهای پیچیده به کار می‌رود ملاحظه می‌گردد. هر یک OR باید با 32 ورودی در نظر گرفته شود. هر خروجی دیکدر به یکی از ورودی‌های هر گیت OR وصل است. چون هر گیت OR دارای 32 اتصال درونی است. به طور کلی یک ROM $2^k \times n$ دارای دیکدر داخلی $2^k \times k$ و n گیت OR است. هر گیت OR دارای 2^k ورودی است که به هر یک از خروجی‌های دیکدر متصل می‌باشد.

تمام 256 تقاطع در شکل ۱۰-۷ قابل برنامه‌ریزی‌اند. یک اتصال قابل برنامه‌ریزی دو خط در عمل به معنی کلیدی است که می‌تواند به هر یک از دو حالت بسته (یعنی خط به هم وصلند) و باز (به معنی دو خط از هم جدا) تغییر وضع دهد. تقاطع قابل برنامه‌ریزی بین دو خط راگاهی نقطه تقاطع هم می‌گویند. برای پیاده‌سازی نقاط تقاطع وسایل فیزیکی متفاوتی وجود دارد. یکی از ساده‌ترین تکنولوژی‌ها از فیوز استفاده می‌کند که معمولاً دو نقطه را به هم وصل می‌نماید، و با اعمال یک پالس ولتاژ شدید به فیوز، اتصال باز یا سوزانده می‌شود.

ذخیره‌سازی درونی یک ROM با جدول درستی که محتوای کلمه را در هر آدرس نشان می‌دهد،

جدول ۳-۷. جدول درستی ROM (ناقص)

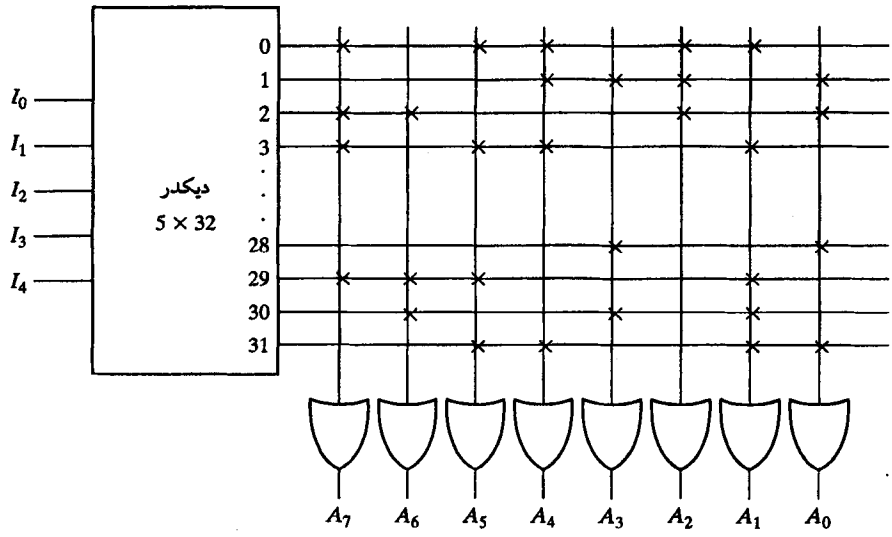
ورودی‌ها					خروجی‌ها							
I4	I3	I2	I1	I0	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮					⋮					
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

صورت می‌گیرد. مثلاً، محتوای یک ROM 8 × 32 ممکن است با یک جدول درستی مشابه جدول (۳-۷) مشخص می‌گردد. در جدول پنج ورودی که در زیر آنها همه 32 آدرس آمده دیده می‌شود. در هر آدرس یک کلمه 8 بیتی ذخیره شده است که در زیر ستون خروجی‌ها لیست شده است. جدول، تنها چهار کلمه اول و چهار کلمه آخر را در ROM نشان می‌دهد. جدول کامل حاوی 32 کلمه خواهد بود.

روش سخت‌افزاری برنامه‌ریزی ROM سوزاندن فیوزها طبق یک جدول درستی است. مثلاً برنامه‌ریزی ROM بر طبق جدول درستی (۳-۷) آرایش شکل ۱۱-۷ را نتیجه می‌دهد. هر 0 موجود در جدول به معنی عدم وجود اتصال و هر 1 مسیری را که به وسیله اتصال به دست می‌آید، نشان می‌دهد. به عنوان مثال، جدول کلمه 8 بیت 10110010 را در آدرس 3 نشان می‌دهد. چهار عدد 0 واقع در کلمه با سوزاندن فیوزهای بین خروجی 3 دیکدر و ورودی‌های گیت‌های OR مربوطه به خروجی‌های A_6 ، A_3 ، A_2 و A_0 ، برنامه‌ریزی می‌شود. چهار 1 در کلمه در نمودار با × علامت‌زنی شده‌اند تا اتصالی را در نمودار منطقی بیان نمایند. وقتی ورودی ROM برابر 00011 می‌باشد، همه خروجی‌های دیکدر 0اند به جز خروجی 3که در منطق 1 است. سیگنال معادل با منطق 1 در خروجی 3 دیکدر از میان اتصالات عبور کرده و به خروجی‌های A_7 ، A_5 ، A_4 و A_1 می‌رود. چهار خروجی دیگر در 0 باقی می‌مانند. نتیجه این است که کلمه ذخیره شده 10110010 به هشت خروجی داده ارسال می‌گردد.

پیاده‌سازی مدارهای ترکیبی

در بخش ۸-۴ دیدیم که یک دیکدر، 2^k مینترم را برای k متغیر ورودی تولید می‌کند. با نصب گیت‌های OR برای بدست آوردن جمع مینترم‌های توابع بول توانستیم هر مدار ترکیبی را تولید نماییم. ROM اساساً وسیله‌ای است که هر دو بخش دیکدر و گیت‌های OR را در خود دارد. با انتخاب اتصالات برای مینترم‌های تابع، خروجی‌های ROM برای نمایش متغیرهای خروجی تابع بول در مدار ترکیبی، قابل برنامه‌ریزی خواهد بود.



شکل ۱۱-۷. برنامه‌ریزی ROM بر طبق جدول ۳-۷

عملکرد درونی ROM را به دو طریق می‌توان تفسیر کرد. اول این که ROM حافظه‌ای است حاول الگوی ثابتی از کلمات ذخیره شده. تفسیر دوم این است که این واحد یک مدار ترکیبی را پیاده‌سازی می‌نماید. از این دیدگاه هر پایانه خروجی به طور جداگانه یک خروجی برای تابع بول تلقی می‌شود که برحسب جمع مینترم‌ها بیان گشته است. مثلاً ROM شکل ۱۱-۷ را می‌توان یک مدار ترکیبی با هشت خروجی تصور کرد که هر یک تابعی از پنج متغیر ورودی است. خروجی A_7 را می‌توان به صورت جمع مینترم‌های زیر نوشت:

$$A_7\ (I_4\ ,\ I_3\ ,\ I_2\ ,\ I_1\ ,\ I_0) = \Sigma(0,\ 2,\ 3,\ \ldots\ ,\ 29)$$

اتصالی که در شکل بالا × علامت زده شده است مینترمی را برای مجموع تولید می‌کند. دیگر نقاط تقاطع اتصال ندارند و بنابراین در جمع لحاظ نشده‌اند.

در عمل، وقتی یک مدار ترکیبی به وسیله ROM طراحی می‌شود، لزومی ندارد اتصال درونی گیت‌ها را نشان دهیم. آنچه که یک طراح باید انجام دهد این است که ROM را با شماره‌اش مشخص نماید و جدول درستی ROM را نیز مهیا سازد. جدول درستی همه اطلاعات لازم برای برنامه‌ریزی ROM را دارد. هیچگونه نمودار منطقی درونی برای تکمیل و همراهی جدول درستی مورد نیاز نیست.

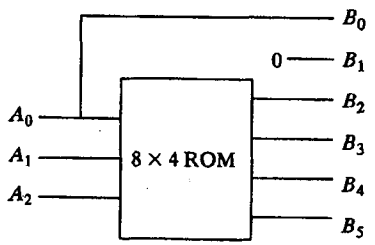
مثال ۱-۷

با استفاده از ROM یک مدار ترکیبی طراحی نمایید. مدار یک عدد 3 بیت را پذیرفته و یک عدد دودویی معادل با مرجع عدد ورودی تولید می‌نماید.

جدول ۴-۷. جدول درستی مدار مثال ۱-۷

ورودی‌ها			خروجی‌ها						
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	دهدهی
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

اولین قدم بدست آوردن جدول درستی مدار ترکیبی است. در بسیاری از موارد بدست آوردن جدول درستی کافی است. در بعضی از حالات هم جدول درستی کوچکتری را برای ROM با توجه به خواص متغیرهای خروجی تعریف می‌کنند. جدول ۴-۷ جدول درستی مدار ترکیبی است. برای لحاظ همه اعداد ممکن دودویی، سه ورودی و شش خروجی لازم است. با توجه به جدول ملاحظه می‌شود که خروجی B_0 همواره با ورودی A_0 برابر است، بنابراین نیازی به تولید B_0 توسط ROM نیست. به علاوه خروجی B_1 همواره 0 است، بنابراین خروجی یک ثابت شناخته شده است. در نتیجه لازم است فقط چهار خروجی را با ROM تولید نماییم، و دو خروجی دیگر به راحتی قابل حصول می‌باشند. به این ترتیب حداقل سایزی که ROM به آن نیاز دارد سه ورودی و چهار خروجی است که سه ورودی هشت کلمه را مشخص می‌کنند و بنابراین ROM باید دارای سایز 4 × 8 باشد. پیاده‌سازی ROM این تابع در شکل ۱۲-۷ مشاهده می‌شود که در آن سه ورودی مدار هشت کلمه چهار بیتی را معین می‌نمایند. جدول درستی شکل ۱۲-۷ (ب) حاوی اطلاعات لازم برای برنامه‌ریزی ROM است. نمودار بلوکی شکل ۱۲-۷ (الف) اتصالات مورد نیاز را برای مدار ترکیبی نشان می‌دهد.



(الف) نمودار بلوکی

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(ب) جدول درستی ROM

شکل ۱۲-۷. پیاده سازی ROM

انواع ROM

مسیرهای لازم در یک ROM به چهار روش مختلف برنامه‌ریزی می‌شوند. اولین روش که برنامه‌ریزی ماسک نامیده می‌شود به وسیله سازنده در آخرین مرحله ساخت قطعه صورت می‌گیرد. در

این روش، ساخت ROM مستلزم پر شدن جدول درستی مربوطه به وسیله سفارش دهنده است. این جدول ممکن است روی فرم مخصوصی که توسط سازنده ارائه می‌شود، و یا با قالب خاصی، روی خروجی‌های کامپیوتر ایجاد گردد. بر طبق جدول مشتری، سازنده ماسک مربوطه را برای تولید 1ها و 0ها برای مسیرها می‌سازد. این روش بسیار پرهزینه است زیرا سازنده مبلغ خاصی را برای ماسک کردن ROM از مشتری می‌گیرد. به همین دلیل، برنامه‌ریزی ماسک هنگامی اقتصادی است که تعداد زیادی از یک نوع ROM سفارش داده شود.

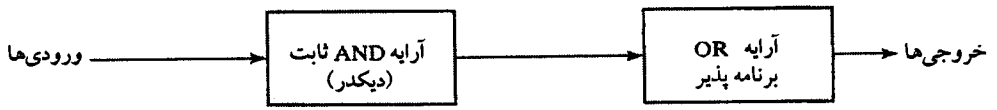
برای تعداد کم، نوع دوم آن به نام حافظه فقط خواندنی برنامه‌پذیر (PROM) اقتصادی‌تر است. هنگام سفارش، همه فیوزهای PROM دست نخورده و همه بیت‌های کلمه در منطق 1 هستند. سوزاندن فیوزها در PROM با استفاده از یک پالس ولتاژ قوی و از طریق پایه خاصی صورت می‌گیرد. فیوز سوخته منطق 0 و دست نخورده منطق 1 را ایجاد می‌نمایند. این امکانات به کاربر اجازه می‌دهد تا PROM را در آزمایشگاه برنامه‌ریزی کرده و به رابطه بین آدرس‌های ورودی و کلمات ذخیره شده دست یابد. دستگاه خاصی به نام برنامه‌نویس PROM برای انجام این کار به صورت تجاری در اختیار قرار دارد. در هر حال، همه رویه‌ها برای برنامه‌ریزی ROMها روال‌های سخت‌افزاری‌اند حتی اگر برنامه‌نویسی کلمات صورت گیرد.

روال برنامه‌نویسی یا برنامه‌ریزی سخت‌افزاری ROMها یا PROMها برگشت‌ناپذیرند و به محض انجام آن، الگویی ثابت و دائمی در آن ایجاد می‌شود که تغییرناپذیر است. به محض ایجاد الگوی بیتی، کل قطعه هنگام تغییر الگو دور ریخته می‌شود. سومین نوع ROM، به نام *PROM* پاک شونده یا EPROM است. می‌توان EPROM را دوباره به حالت اولیه‌اش در آورد، حتی اگر قبلاً برنامه‌نویسی شده باشد. وقتی EPROM را برای مدت معینی تحت تابش اشعه ماوراء بنفش قرار دهیم اشعه موج کوتاه مذکور گیت‌های شناور درونی که نقش اتصالات برنامه‌ریزی شده را دارند، تخلیه می‌کند. پس از پاک شدن، EPROM به حالت اولیه خود بازگشته و با مقادیر جدیدی قابل برنامه‌ریزی است.

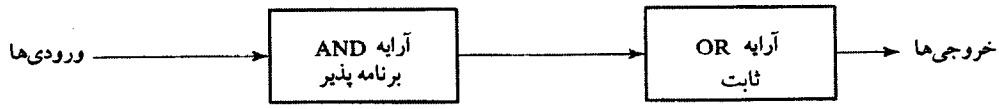
نوع چهارم ROM، *PROM* پاک شونده الکتریکی (EPROM یا EEPROM) است. این حافظه شبیه EPROM است با این تفاوت که اتصالات برنامه‌ریزی شده قبلی را به صورت الکتریکی در عوض اشعه ماوراء بنفش می‌توان پاک کرد. مزیت این کار این است که حافظه بدون نیاز به خارج کردن آن از سوکت پاک می‌شود.

PLD های ترکیبی

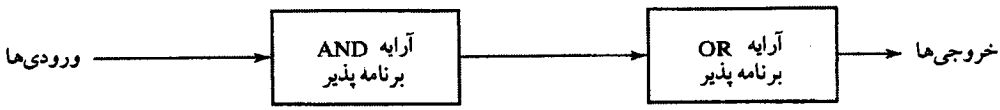
PROM یک وسیله منطقی برنامه‌پذیر ترکیبی در یک مدار مجتمع است که گیت‌های برنامه‌پذیر آن به دو بخش آرایه AND و آرایه OR تفکیک شده است تا جمع حاصلضرب‌ها با AND-OR پیاده‌سازی شود. سه نوع PLD ترکیبی وجود دارند و اختلاف آنها در ایجاد اتصالات برنامه‌پذیر در آرایه AND-OR است. شکل ۱۳-۷ آرایش سه PLD را نشان می‌دهد. حافظه فقط خواندنی برنامه‌پذیر (PROM) دارای آرایه AND ثابت به عنوان دیکدر است ولی آرایه OR آن برنامه‌پذیر است. گیت‌های



(الف) حافظه فقط خواندنی برنامه پذیر (PROM)



(ب) منطق آرایه‌ای برنامه پذیر (PAL)



(پ) آرایه منطقی برنامه پذیر (PLA)

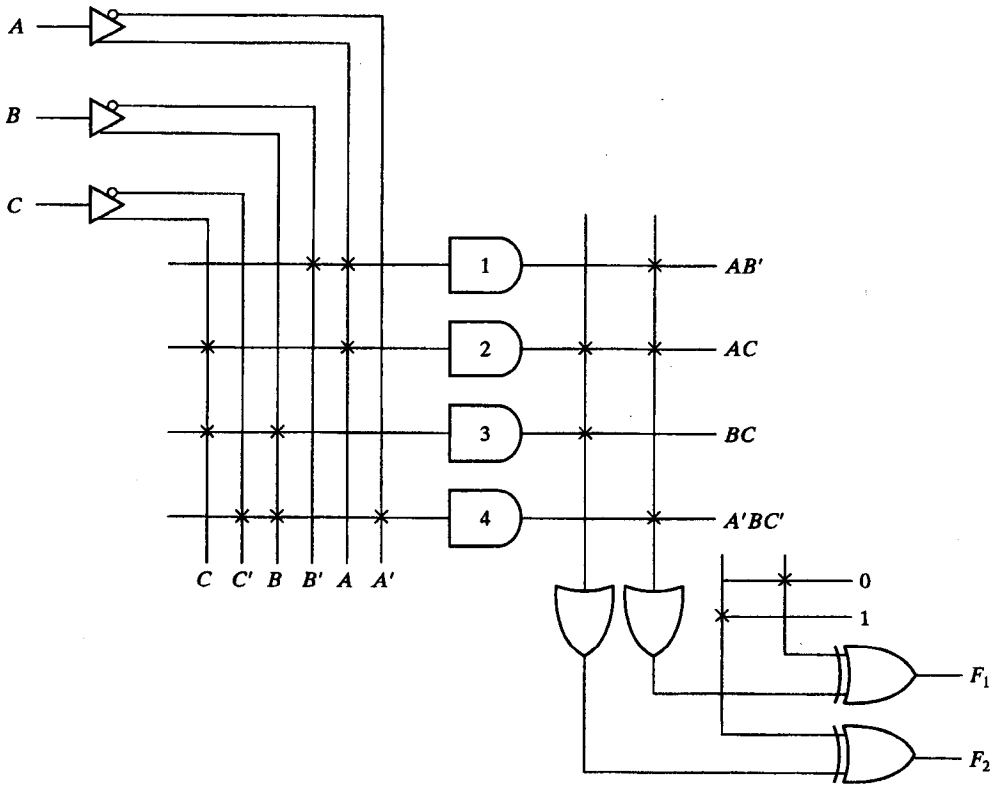
شکل ۱۳-۷. آرایش پایه سه نوع PDL

OR برنامه‌پذیر توابع بول را به صورت جمع حاصلضرب‌ها پیاده‌سازی می‌کنند. منطق آرایه برنامه‌پذیر (PAL) دارای آرایه AND برنامه‌پذیر و آرایه OR ثابت است. گیت‌های AND در تهیه جملات ضرب برای تابع بول برنامه‌ریزی می‌شوند که به صورت منطقی در هر گیت OR جمع شده‌اند. با انعطاف‌ترین PLD، آرایه منطقی برنامه‌پذیر (PLA) است که در آن هر دو آرایه AND و OR برنامه‌پذیرند. جملات ضرب در آرایه AND قابل استفاده مشترک با هر گیت OR برای پیاده‌سازی جمع حاصلضرب موردنظر است. نام‌های PAL و PLA از دو سازنده مختلف در حین ساخت وسایل منطقی برنامه‌پذیر متفاوت اعلان شد. پیاده‌سازی مدارهای ترکیبی با PROM در این بخش نشان داده شد. طراحی این مدارها با PLA و PAL در دو بخش بعد مطرح شده است.

۷-۶ آرایه منطقی برنامه‌پذیر PLA

آرایه منطقی برنامه‌پذیر (PLA) از نظر مفهوم شبیه PROM است. به جز این که PLA دیکد کامل متغیرها را فراهم نکرده و بنابراین همه مینترم‌ها را تولید نمی‌کند. در این مدار دیکدر با آرایه‌ای از گیت‌های AND جایگزین شده و می‌تواند برای تولید هر جمله ضرب متغیرهای ورودی برنامه‌ریزی شود. سپس جملات ضرب به گیت‌های OR وصل می‌شود تا جمع حاصلضرب‌ها برای توابع بول موردنظر تهیه شود.

منطق درونی یک PLA با سه ورودی و دو خروجی در شکل ۱۴-۷ دیده می‌شود. از لحاظ تجاری این قطعه به علت کوچکی مقرون به صرفه نیست، ولی در اینجا به منظور نمایش آرایش منطقی PLA



شکل ۱۴-۷ PAL با سه ورودی، چهار جمله ضرب و دو خروجی

نمونه مطرح شده است. نمودار، سمبل‌های گرافیکی آرایه‌ای را برای مدار مجتمع پیچیده به کار می‌برد. هر ورودی وارد یک بافر و یک وارونگر می‌شود که در نمودار هم با سمبلی مرکب نشان داده شده است. این سمبل دارای هر دو خروجی صحیح و متمم شده است. هر ورودی و متمم آن به ورودی‌های هر گیت AND وصل است. این اتصال با تقاطع خطوط عمودی و افقی انجام گردیده است. خروجی‌های گیت‌های AND به ورودی‌های هر گیت وصلند. خروجی گیت OR به یک گیت XOR رفته است. گیت‌های اخیر جهت دریافت 1 یا 0 برنامه‌ریزی شده‌اند. وقتی ورودی XOR به 1 وصل باشد خروجی معکوس می‌شود (چون $x \oplus 1 = x'$) هنگامی که ورودی XOR به 0 وصل باشد خروجی تغییر نمی‌کند، (چون $x \oplus 0 = x$) . توابع بول خاصی که در شکل ۱۴-۷ برای PLA پیاده‌سازی شده است عبارتند از:

$$F_1 = AB' + AC + A'BC'$$
$$F_2 = (AC + BC)'$$

جملات ضرب تولید شده در کنار خروجی هر گیت AND در نمودار نوشته شده است. جمله ضرب از

جدول ۵-۷. جدول برنامه ریزی PAL

	جمله ضرب	ورودی‌ها			خروجی‌ها (T)	(C)
		A	B	C	F_1	F_2
AB′	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A′BC′	4	0	1	0	1	–

روی نقاط تقاطع متصلی که با × علامت خورده معین می‌شود. خروجی یک گیت OR جمع منطقی جملات ضرب انتخابی را تهیه می‌کند. بسته به اتصال یکی از ورودی‌های گیت XOR ، فرم متمم یا غیرمتمم خروجی حاصل می‌گردد.

نقشه فیوز PLA به شکل جدول می‌تواند نشان داده شود. جدول برنامه‌ریزی مربوط به PLA شکل ۷-۱۴ در جدول ۵-۷ ملاحظه می‌گردد. جدول برنامه‌ریزی از سه بخش تشکیل شده است. اولین بخش جملات ضرب را به صورت عدد لیست کرده است. بخش دوم مسیرهای لازم بین ورودی‌ها و گیت‌های AND را مشخص می‌نماید. سومین بخش مسیرهای بین گیت‌های AND و OR را معین می‌کند. برای هر متغیر خروجی، از T (به معنی صحیح یا متمم نشده) و یا C (به معنی متمم) برای برنامه‌ریزی گیت XOR استفاده می‌نماییم. جملات ضرب لیست شده در سمت چپ بخشی از جدول نیستند، بلکه فقط به منظور داشتن مرجع لحاظ شده‌اند. برای هر جمله ضرب ورودی‌ها با 1 ، 0 یا – (خط تیره) علامت‌زنی شده‌اند. اگر متغیری در جمله ضرب متمم نشده ظاهر شود متغیر ورودی مربوطه با 1 علامت‌گذاری می‌شود. اگر متمم باشد آن را با 0 علامت می‌زنیم. اگر متغیر در جمله ضرب وجود نداشته باشد با علامت خط تیره علامت زده خواهد شد. مسیرهای بین ورودی‌ها و گیت‌های AND در جدول برنامه‌ریزی زیرستون "ورودی‌ها" ذکر شده‌اند. هر 1 از ستون ورودی اتصال را از متغیر ورودی به گیت AND نشان می‌دهد. هر 0 در ستون ورودی، اتصالی را از متمم متغیر به ورودی گیت AND نشان می‌دهد. خط تیره به معنی فیوز سوخته‌ای در هر دو ورودی متغیر و متمم آن است. ضمناً هر پایانه باز، به معنی منطق 1 فرض می‌شود.

مسیرهای بین گیت‌های AND و OR زیر ستون "خروجی‌ها" ذکر شده‌اند. متغیرهای خروجی برای جملات ضرب موجود در تابع با 1 علامت زنی می‌شوند. هر جمله ضربی که در ستون خروجی‌اش 1 دارد به مسیری از خروجی گیت AND به ورودی گیت OR احتیاج دارد. مکان‌هایی که با خط تیره علامت زده شده فیوزهای سوخته شونده را مشخص می‌کنند. فرض بر این است که پایانه‌های باز در ورودی گیت OR ، همچون 0 عمل کنند. بالاخره یک خروجی غیرمتمم، T ، لازم می‌دارد که دیگر ورودی‌های گیت XOR مربوطه به 0 وصل شوند، ولی خروجی متمم، C ، اتصال به 1 را بیانگر است.

سایز یک PLA با تعداد ورودی‌ها، جملات ضرب و تعداد خروجی‌ها معین می‌گردد. نوعی مدار

مجتمع PLA ممکن است 16 ورودی، 4 جمله ضرب و 8 خروجی داشته باشد. برای n ورودی، k جمله ضرب و m خروجی، منطق درونی PLA متشکل از n بافر- وارونگر، k گیت AND ، $k \times m$ اتصال بین آرایه AND و OR و m اتصال مربوط به گیت‌های XOR می‌باشد.

هنگام طراحی سیستم‌های دیجیتال با PLA ، نیازی به نمایش اتصالات درونی آن، همچون شکل ۷-۱۴ وجود ندارد. آنچه لازم است، یک جدول برنامه‌ریزی است که PLA به وسیله آن برنامه‌ریزی می‌گردد. همچون ROM ، PLA هم ممکن است به صورت ماسک برنامه‌پذیر، یا برنامه‌پذیر موردی باشد. با برنامه‌ریزی ماسک، مشتری جدول برنامه PLA خود را به سازنده ارائه می‌کند. این جدول به وسیله سازنده استفاده شده و PLA سفارشی که منطق درونی‌اش طبق سفارش مشتری است، تولید می‌گردد. نوع دوم PLA موجود، آرایه منطقی برنامه‌پذیر موردی (FPLA) نامیده می‌شود. FPLA می‌تواند با یک دستگاه برنامه‌نویس سخت‌افزاری تجاری به وسیله کاربر برنامه‌نویس گردد.

هنگام پیاده‌سازی یک مدار ترکیبی با PLA ، باید بررسی‌های دقیقی به عمل آید تا جملات ضرب جدا از هم کاهش یابد، زیرا PLA دارای تعداد گیت‌های محدودی است. این کار با حداقل کردن تعداد جملات در هر تابع بول امکان‌پذیر است. تعداد لیترال در یک جمله اهمیت ندارد زیرا همه متغیرهای ورودی به هر ترتیب در دسترسند. هر دو نوع تابع خروجی، یعنی متمم یا غیرمتمم باید مشخص گردد تا ببینیم کدام یک با تعداد کمتری جمله ضرب بیان می‌شود و کدام جمله نیز در دیگر توابع مشترک است.

مثال ۲-۷

دو تابع بول زیر را با PLA پیاده‌سازی کنید.

$$F_1(A, B, C) = \Sigma(0, 1, 2, 4)$$
$$F_2(A, B, C) = \Sigma(0, 5, 6, 7)$$

دو تابع در نقشه ۱۵-۷ ساده شده‌اند. هر دو فرم متمم و غیرمتمم تابع برحسب جمع حاصلضرب‌ها ساده شده‌اند. ترکیبی که حداقل جملات ضرب را می‌دهد عبارت است:

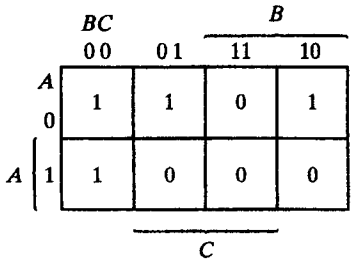
$$F_1 = (AB + AC + BC)'$$

و

$$F_2 = AB + AC + A'B'C'$$

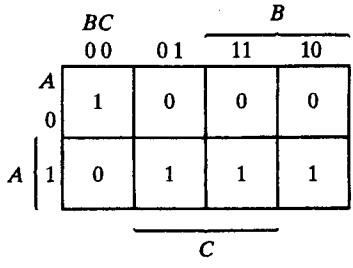
در این دو تابع چهار جمله ضرب مستقل وجود دارد که عبارتند از AB ، AC ، BC و $A'B'C'$. جدول برنامه‌ریزی PLA برای این ترکیب در شکل نشان داده شده است. توجه کنید که خروجی F_1 خروجی غیرمتمم است هر چند که روی حرف متمم (C) نوشته شده است. دلیل این است که F_1 با یک مدار AND-OR تولید شده و در خروجی گیت OR در دسترس است. گیت XOR ، تابع را متمم می‌کند تا خروجی صحیح F_1 بدست آید.

❑



$F_1 = A'B' + A'C' + B'C'$

$F_1 = (AB + AC + BC)'$



$F_2 = AB + AC + A'B'C'$

$F_2 = (A'C + A'B + AB'C')'$

جدول برنامه ریزی PLA

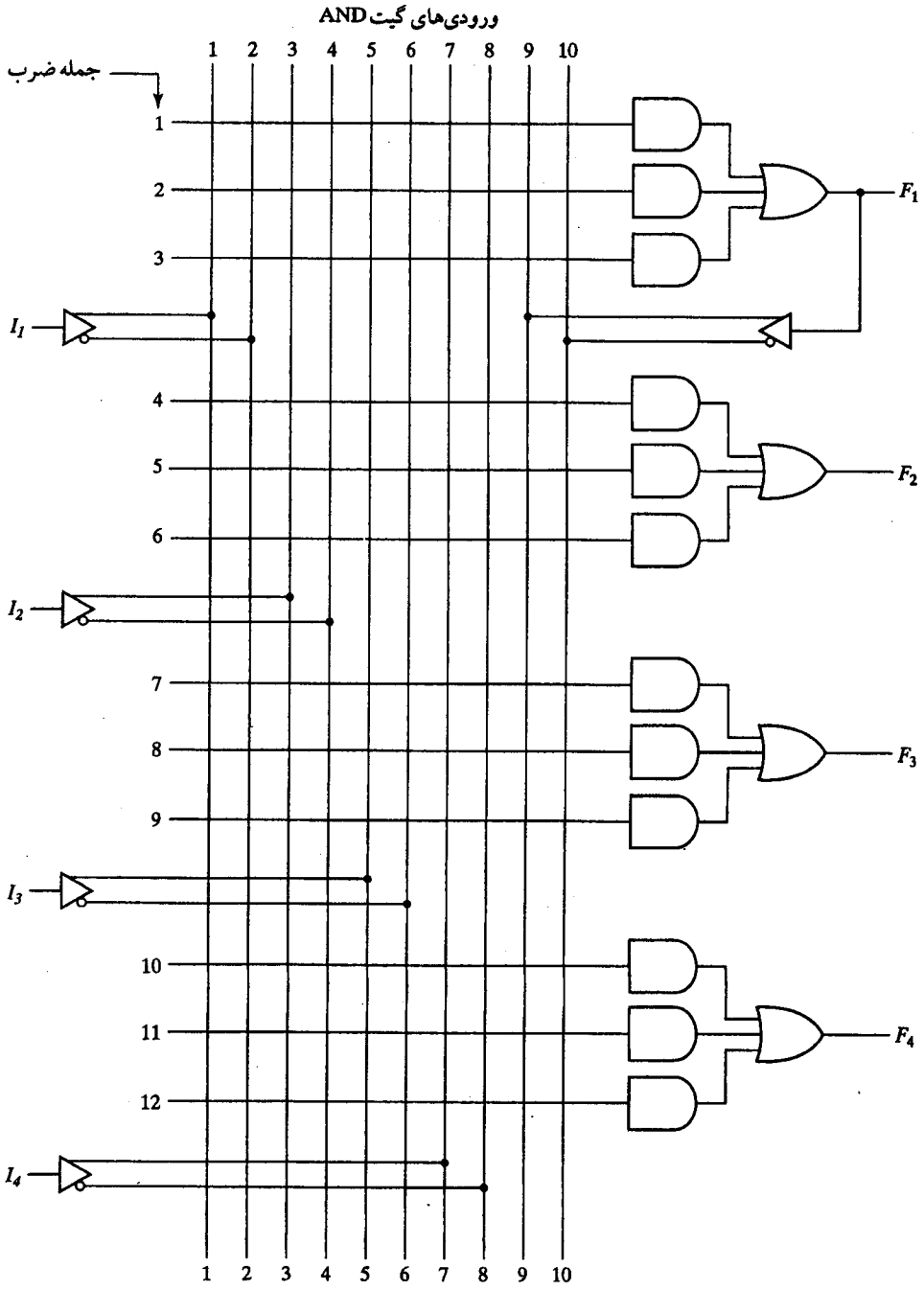
	جمله ضرب	ورودی‌ها A	B	C	خروجی‌ها (C) F_1	(T) F_2
AB	1	1	1	–	1	1
AC	2	1	–	1	1	1
BC	3	–	1	1	1	–
$A'B'C'$	4	0	0	0	–	1

شکل ۱۵-۷. حل مثال ۲-۷

مدار ترکیبی به کار رفته در مثال ۲-۷ برای پیاده‌سازی با یک PLA بسیار ساده است. این مثال صرفاً به خاطر تشریح اهداف ارائه شد. PLA های موجود معمولاً تعداد زیادی ورودی و جملات ضرب دارند. ساده‌سازی توابع با تعداد زیادی متغیر باید به کمک روال‌های کامپیوتری انجام شود. برنامه‌های طراحی CAD در این زمینه هر تابع و متمم آن را از نظر تعداد جملات ساده می‌کند. آنگاه برنامه همه توابع را به فرم معمول یا متمم با توجه به حداقل جملات ضرب انتخاب می‌کند. سپس جدول برنامه‌ریزی PLA تولید می‌گردد و نقشه فیوزها فراهم می‌شود. نقشه فیوزها به یک برنامه‌نویس که از یک روال سخت‌افزاری در سوزاندن استفاده می‌کند اعمال می‌گردد تا مدار موردنظر بدست آید.

۷-۷ منطق آرایه‌ای برنامه‌پذیر PAL

منطق آرایه‌ای برنامه‌پذیر (PAL) وسیله‌ای است با آرایه OR ثابت و آرایه AND برنامه‌پذیر. چون تنها گیت‌های AND برنامه‌پذیرند، PAL برای برنامه‌ریزی ساده‌تر است ولی به اندازه PLA انعطاف‌پذیر نیست. شکل ۱۶-۷ آرایش منطقی یک PAL نمونه را نشان می‌دهد. این مدار چهار ورودی و چهار خروجی دارد. هر ورودی به یک گیت بافر- وارونگر وصل است و هر خروجی با گیت OR ثابت تولید می‌گردد. در مدار چهار بخش موجود است که هر یک از آرایه AND-OR سه وجهی



شکل ۱۶-۷ . PAL با چهار ورودی، چهار خروجی AND-OR سه ردیفی

تشکیل شده است. این جمله به این دلیل بکار رفته است تا مشخص شود که سه گیت AND برنامه پذیر در هر بخش و نیز یک گیت OR ثابت وجود دارد. هر گیت AND دارای 10 اتصال ورودی برنامه‌پذیر است. این ویژگی با 10 خط عمودی که هر خط افقی را قطع می‌کند نشان داده شده است. خطوط افقی نمایشگر آرایش چند ورودی گیت AND می‌باشند. یکی از خروجی‌ها به گیت بافر- وارونگر وصل و سپس به دو ورودی گیت AND پسخورد شده است.

PAL های تجاری حامل گیت‌های بیشتری از آنچه در شکل ۱۶-۷ ملاحظه شد، هستند. بعضی از آنها تا هشت ورودی، هشت خروجی، هشت بخش هر یک با سه آرایه AND-OR سه وجهی دارند. پایانه‌های خروجی گاهی با بافرهای سه حالته و وارونگرها راه‌اندازی می‌شوند.

هنگام طراحی با PAL ، توابع بول، تا حدی که در هر بخش قابل پیاده‌سازی شدن باشند، باید ساده شوند. برخلاف PLA ، یک جمله ضرب نمی‌تواند میان دو یا چند گیت OR به اشتراک باشد. بنابراین، هر تابع بدون توجه به وجود اشتراک در جملات ضرب ساده می‌گردد. تعداد جملات ضرب در هر بخش ثابت است، و اگر تعداد جملات در تابع زیاد باشد، ممکن است برای پیاده‌سازی یک تابع بول از دو بخش استفاده شود.

به عنوان مثال از کاربری یک PAL در طراحی یک مدار ترکیبی، توابع بولی زیر را که برحسب مجموع مینترم‌ها داده شده است ملاحظه نمایید.

$$w(A, B, C, D) = \sum(2, 12, 13)$$
$$x(A, B, C, D) = \sum(7, 8, 9, 10, 11, 12, 13, 14, 15)$$
$$y(A, B, C, D) = \sum(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$$
$$z(A, B, C, D) = \sum(1, 2, 8, 12, 13)$$

با ساده کردن چهار تابع فوق به حداقل تعداد جملات، توابع بولی زیر نتیجه می‌شود

$$w = ABC' + A'B'CD'$$
$$x = A + BCD$$
$$y = A'B + CD + B'D'$$
$$z = ABC' + A'B'CD' + AC'D' + A'B'C'D$$
$$= w + AC'D' + A'B'C'D$$

توجه کنید که تابع z دارای چهار جمله ضرب است. جمع منطقی دوتای آن برابر w است. با به کارگیری w می‌توان تعداد جملات در z را از چهار به سه رساند.

جدول برنامه‌ریزی PAL مشابه PLA است با این تفاوت که فقط ورودی گیت‌های AND باید برنامه‌ریزی شوند. جدول ۶-۷ جدول برنامه‌ریزی PAL را برای چهار تابع بول نشان می‌دهد. جدول به چهار بخش با سه جمله ضرب در هر کدام تقسیم شده است تا با شکل ۱۶-۷ همخوانی داشته باشد. دو بخش اول تنها دو جمله ضرب لازم دارند تا تابع بول پیاده‌سازی شود. بخش آخر برای خروجی z چهار جمله ضرب لازم دارد. با استفاده از خروجی w ، می‌توانیم جملات را به سه برسانیم.

جدول ۶-۷. جدول برنامه ریزی PAL

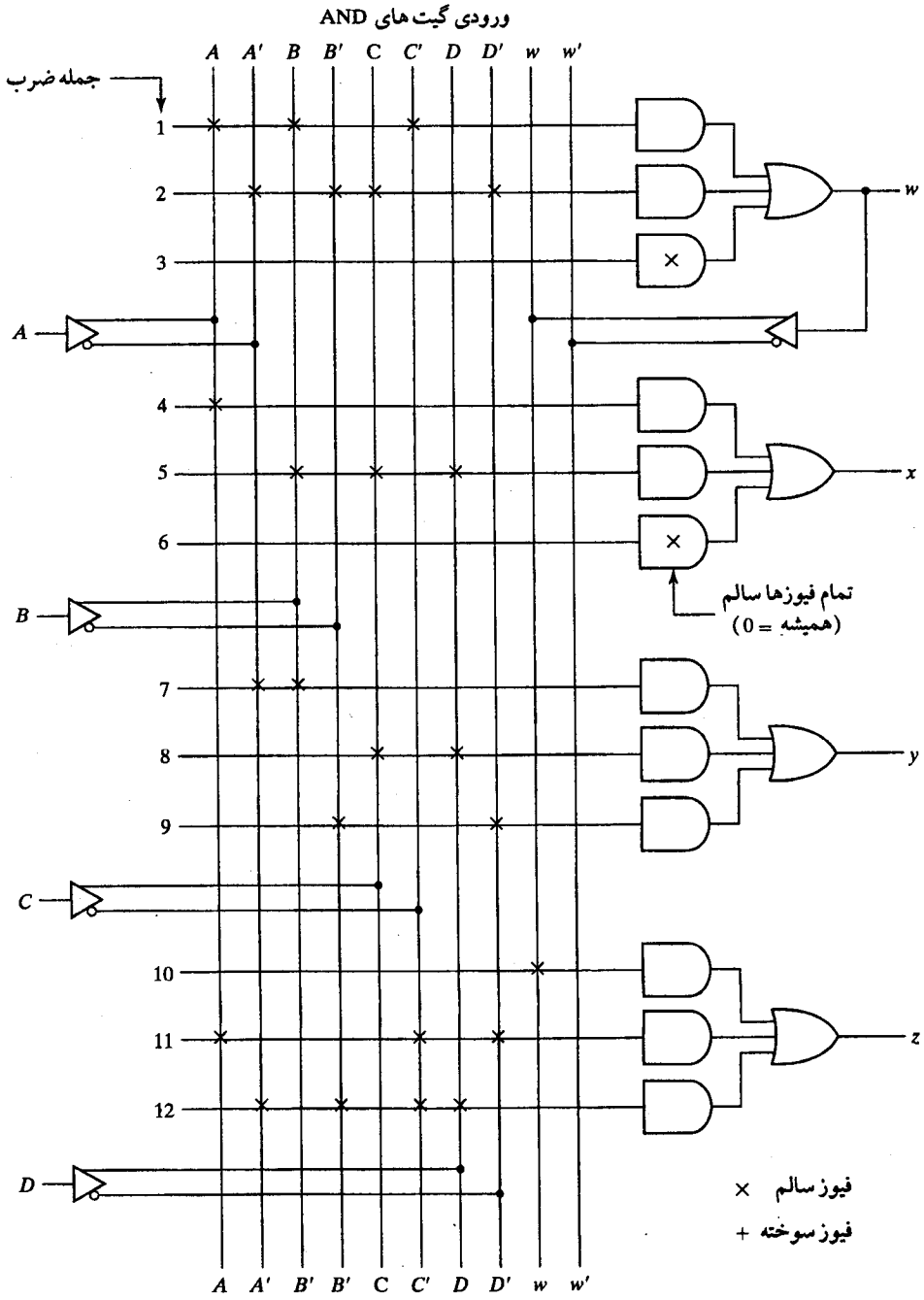
جمله حاصلضرب	ورودی‌های AND					خروجی‌ها
	A	B	C	D	W	
1	1	1	0	–	–	$w = ABC'$
2	0	0	1	0	–	$+ A'B'CD'$
3	–	–	–	–	–	
4	1	–	–	–	–	$x = A$
5	–	1	1	1	–	$+ BCD$
6	–	–	–	–	–	
7	0	1	–	–	–	$y = A'B$
8	–	–	1	1	–	$+ CD$
9	–	0	–	0	–	$+ B'D'$
10	–	–	–	–	1	$z = w$
11	1	–	0	0	–	$+ AC'D'$
12	0	0	0	1	–	$+ A'B'C'D$

نقشه فیوزها برای PAL طبق جدول برنامه‌ریزی در شکل ۷-۱۷ ملاحظه می‌شود. برای هر 1 یا 0 تقاطع را روی نمودار با فیوز سالم علامت می‌زنیم. برای هر خط تیره، نمودار با فیوز سوخته برای هر دو ورودی صحیح (غیرمتمم) و متمم علامت‌زنی می‌شود. اگر گیت AND به کار نرود، همه فیوزهای ورودی سالم رها می‌شوند. چون ورودی مربوطه هر دو فرم متمم و غیرمتمم هر متغیر ورودی را دریافت می‌کند، داریم $AA' = 0$ و خروجی گیت AND همیشه 0 است. طراحی PALها مانند همه PLDها با استفاده از تکنیک‌های کامپیوتری ساده شده است. سوزاندن فیوزهای داخلی یک روال سخت‌افزاری است که به کمک تجهیزات خاص انجام می‌گردد.

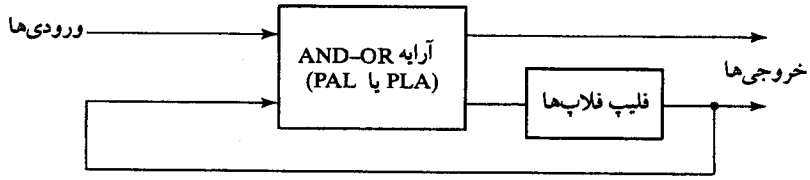
۷-۸ وسایل برنامه‌پذیر ترتیبی

سیستم‌های دیجیتال با استفاده از فلیپ فلاپ‌ها و گیت‌ها طراحی می‌شوند. چون PLD ترکیبی فقط از گیت‌های ساخته می‌شود، لازم است به هنگام استفاده از آنها در مدارهای ترتیبی، از فلیپ فلاپ‌های بیرونی استفاده شود. وسایل برنامه‌پذیر ترتیبی هر دو نوع مدارهای گیتی و فلیپ فلاپ را دارا هستند. به این ترتیب، مدار را برای اجرای انواع توابع ترتیبی می‌توان برنامه‌ریزی کرد. وسایل برنامه‌پذیر تجاری متعددی در دسترسند و هر وسیله هم تنوع خاص سازنده را دارد. منطق درونی این وسایل برای نمایش بسیار پیچیده است. بنابراین، سه نوع اصلی را بدون تشریح ساختار آن توصیف می‌کنیم.

۱- وسیله منطقی برنامه‌پذیر ترتیبی (SPLD) یا ساده
۲- وسایل منطقی برنامه‌پذیر پیچیده (CPLD)
۳- آرایه گیتی برنامه‌پذیر موردی (FPGA)



شکل ۱۷-۷. نقشه فیوز برای PAL طبق جدول (۶-۷)



شکل ۱۸-۷. وسیله منطقی ترتیبی برنامه پذیر

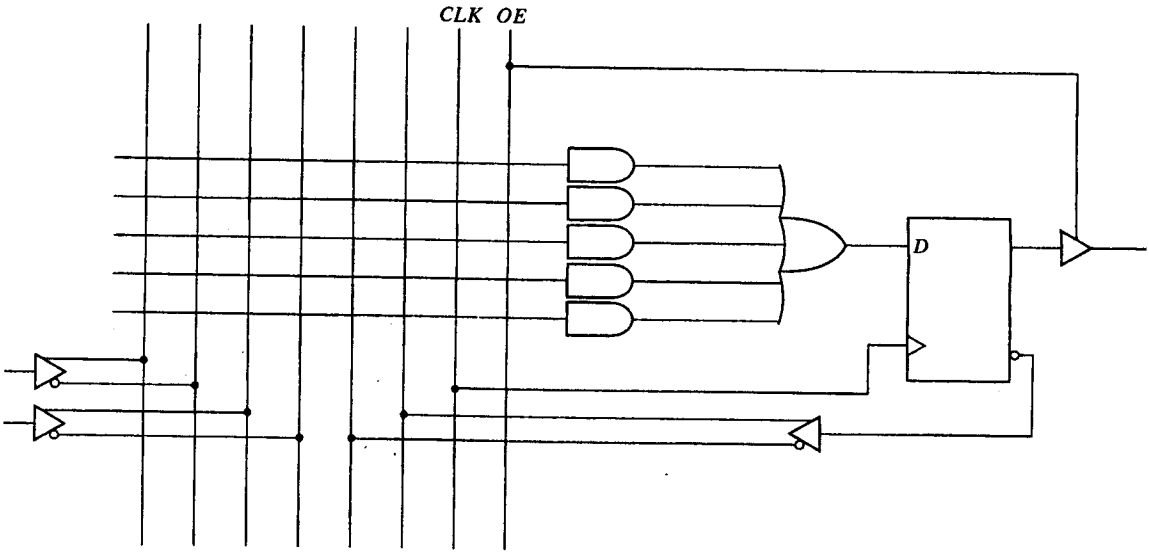
PLD ترتیبی را گاهی PLD ساده (SPLD) هم می‌گویند تا آن را از نوع PLD پیچیده تفکیک کنند. SPLD علاوه بر آرایه AND-OR حاوی فلیپ فلاپ‌ها در تراشه مدار مجتمع هم می‌باشد. نتیجه این ترکیب مدار ترکیبی شکل ۱۸-۷ است. در PLA یا PAL برای ایجاد یک ثبات، فلیپ فلاپ‌ها لحاظ شده‌اند. خروجی‌های مدار را از گیت‌های OR یا از خروجی‌های فلیپ فلاپ می‌توان دریافت کرد. مقداری اتصالات برنامه‌پذیر نیز برای ایجاد جملات ضرب حاصل از خروجی فلیپ فلاپ‌ها به کمک آرایه AND لحاظ شده است. فلیپ فلاپ‌ها می‌توانند از نوع D یا JK باشند.

اولین وسیله برنامه‌پذیری که پیاده‌سازی مدار ترتیبی را پشتیبانی می‌کند توالی‌گر منطقی برنامه‌پذیر موردی (FPLS) است. نمونه‌ای از FPLS حول یک PLA با چند خروجی که فلیپ فلاپ‌ها را راه می‌اندازند تشکیل شده است. به لحاظ این که فلیپ فلاپ‌ها می‌توانند به عنوان نوع D یا JK کار کنند، انعطاف‌پذیرند. FPLS از نظر تجارتی موفقیتی نداشت زیاد تعداد اتصالات برنامه‌پذیر آن فوق‌العاده زیاد بود. رایج‌ترین نوع SPLD، PAL ترکیبی همراه با فلیپ فلاپ‌های D است.

PAL حاوی فلیپ فلاپ را PAL ثبات‌دار گویند تا بدین وسیله مشخص شود که وسیله علاوه بر آرایه AND-OR دارای فلیپ فلاپ نیز هست. هر بخش از یک SPLD را ماکروسل گویند. ماکروسل مداری است که حاوی جمع حاصلضرب، تابع منطقی ترکیبی و یک فلیپ فلاپ اختیاری است. در اینجا فرض خواهیم کرد که جمع حاصلضرب AND-OR به کار رفته باشد ولی در عمل می‌تواند هر یک از پیاده‌سازی‌های دو سطحی، بخش ۷-۳، به کار گرفته شود.

شکل ۱۹-۷ نمودار منطقی یک ماکروسل را نشان می‌دهد. آرایه AND-OR مثل آنچه در PAL ترکیبی شکل ۱۶-۷ دیدیم می‌باشد. خروجی به وسیله یک فلیپ فلاپ D حساس به لبه راه‌اندازی می‌شود. فلیپ فلاپ به یک ورودی ساعت متصل است و حالتش با لبه ساعت تغییر می‌کند. خروجی فلیپ فلاپ به بافر سه حالته متصل است (یا وارونگر) که این بافر نیز با یک سیگنال فعال‌ساز خروجی به نام OE کنترل می‌شود. خروجی فلیپ فلاپ به یکی از ورودی‌های گیت‌های AND برنامه‌پذیر برای تهیه حالت فعلی مدار ترتیبی پسخورد شده است. نمونه‌ای از SPLD دارای 8 تا 10 ماکروسل در یک بسته آی‌سی است. همه فلیپ فلاپ‌ها به یک ورودی CLK مشترک وصلند و همه بافرهای سه حالته با ورودی OE کنترل می‌شوند. با برنامه‌ریزی ورودی‌های انتخاب می‌توان از مولتی پلکسرها برای انتخاب دو یا چهار مسیر مجزا، استفاده کرد.

علاوه بر آرایه AND، یک ماکروسل دارای امکانات برنامه‌ریزی دیگری نیز هست. انتخاب‌هایی از

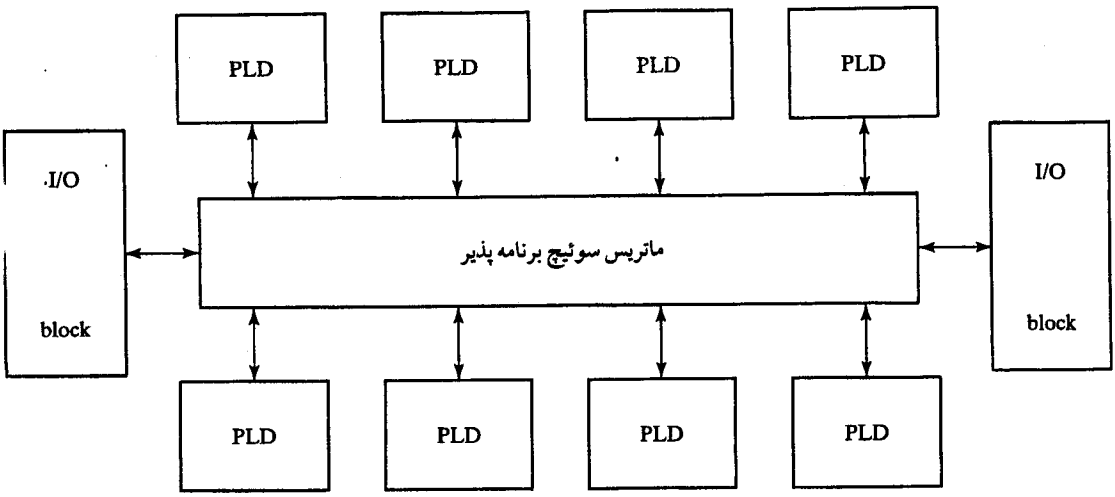


شکل ۱۹-۷. مدار منطقی ماکروسل‌پایه

این نوع امکانات شامل توانایی در استفاده یا نادیده گرفتن فلیپ فلاپ، انتخاب لبه ساعت، انتخاب پاک کردن و پیش تنظیم برای ثبات، و انتخاب خروجی صحیح یا متمم است. یک گیت XOR برای برنامه‌ریزی حالت صحیح/ متمم بکار رفته است.

طراحی یک سیستم دیجیتال با استفاده از PLD اغلب نیاز به اتصال چندین وسیله برای تکمیل ویژگی‌ها دارد. برای این نوع کاربرد، اقتصادی‌تر است از یک وسیله منطقی برنامه‌پذیر پیچیده (CPLD) استفاده شود. CPLD مجموعه‌ای از PLD های منفرد روی یک مدار مجتمع می‌باشد. ساختار اتصالات برنامه‌پذیر ارتباط PLD را به یکدیگر به همان طریق PLD های منفرد امکان پذیر می‌سازد.

شکل ۲۰-۷ آرایش کلی یک CPLD است. این مدار از چند PLD مرتبط به هم که از طریق یک ماتریس سوئیچ برنامه‌ریزی می‌شود تشکیل شده است. بلوک‌های ورودی/ خروجی (I/O) اتصالات به پایه‌های آی‌سی را مهیا می‌سازد. هر پایه I/O با یک بافر سه حالته راه‌اندازی می‌شود و می‌تواند به عنوان خروجی یا ورودی برنامه‌ریزی شود. ماتریس سوئیچ ورودی‌ها را از بلوک I/O دریافت می‌کند و آن را به سمت ماکروسل خاصی هدایت می‌نماید. به همین طریق، خروجی‌های انتخابی از ماکروسل‌ها به خروجی‌های موردنظر ارسال می‌شوند. معمولاً هر PLD حاوی 8 تا 16 ماکروسل است. ماکروسل‌های داخل هر PLD معمولاً به طور کامل وصلند. اگر ماکروسلی جملات ضرب به کار نرفته‌ای داشته باشد می‌توان از آنها در ماکروسل‌های مجاور استفاده کرد. در بعضی از موارد فلیپ فلاپ ماکروسل را می‌توان به عنوان D ، JK یا T برنامه‌ریزی کرد. سازندگان مختلف، راه‌های متفاوتی را برای معماری کلی CPLD ها به کار گرفته‌اند. زمینه‌های اختلاف عبارتند از PLD های درونی خاص (گاهی به آنها



شکل ۲۰-۷. آرایش کلی CPLD

بلوک‌های تابعی می‌گویند)، نوع ماکروسل، بلوک‌های I/O و ساختار اتصالات درونی برنامه‌پذیر. بهترین راه شناسایی وسیله خاص یک سازنده مطالعه مقالات سازنده است.

قطعه اصلی به کار رفته در طراحی VLSI آرایه گیتی است، یک آرایه گیتی از الگویی از گیت در یک ناحیه از سیلیکان ساخته شده و هزاران بار تکرار شده است تا تمام تراشه با گیت پوشش یابد. ابعاد آرایه از یک هزار تا یکصدهزار گیت بسته به تکنولوژی به کار رفته در یک تراشه می‌باشد. طراحی با آرایه‌های گیتی لازم می‌دارد تا مشتری الگوی موردنظر خود را برای سازنده فراهم کند. چند سطح اول فرآیند ساخت مشترک بوده و مستقل از تابع منطقی نهایی است. مراحل بعدی ساخت لازم می‌دارد تا اتصالات درونی گیت‌ها طبق مشخصات داده شده به وسیله طراح به هم وصل شوند.

آرایه گیتی برنامه‌پذیر موردی (FPGA) یک مدار VLSI است که قابل برنامه‌نویسی در محل کاربر است. بعضی از آنها آرایه‌ای متشکل از هزاران بلوک منطقی دارند، که به وسیله بلوک‌های ورودی و خروجی برنامه‌پذیر محصور شده‌اند و از طریق اتصالات قابل برنامه‌ریزی به هم متصل می‌گردند. آرایش‌های داخلی بسیار متنوعی در این گروه از وسایل وجود دارد. رفتار هر وسیله به مدار داخل بلوک و راندمان اتصالات برنامه‌ریزی شده بستگی دارد.

نوعی از بلوک منطقی FPGA از جداول نظاره، مولتی پلکسرها، گیت‌ها و فلیپ فلاپ‌ها ساخته شده است. جداول نظاره نوعی جدول درستی ذخیره شده در SRAM است و توابع مدار ترکیبی را برای بلوک منطقی فراهم می‌کند. این توابع از جدول درستی ذخیره شده در SRAM و شبیه به روش پیاده‌سازی مدارهای ترکیبی با ROM تحقق می‌یابند، بخش ۵-۷. مثلاً یک SRAM 2 × 16 می‌تواند جدول درستی یک مدار ترکیبی که چهار ورودی و دو خروجی دارد را ذخیره نماید. بخش منطق ترکیبی همراه با تعدادی مولتی پلکسرهای برنامه‌پذیر برای بدست آوردن معادلات ورودی به فلیپ فلاپ و

خروجی بلوک منطقی به کار می‌رود.

مزیت استفاده از RAM به جای ROM در ذخیره جدول درستی این است که جدول با نوشتن در حافظه قابل برنامه‌نویسی است. عیب آن هم این است که حافظه فرار بوده و هنگام قطع برق باید محتوای جدول نظاره دوباره بار شود. برنامه را می‌توان از کامپیوتر مرکزی یا از PROM های روی بورد بار کرد. برنامه تا برنامه‌ریزی مجدد FPGA و یا تا قطع برق در SRAM باقی می‌ماند. هر بار که برق وصل شود وسیله باید دوباره برنامه‌ریزی گردد. قابلیت برنامه‌ریزی مجدد FPGA کاربردهای زیادی را با پیاده‌سازی مدارهای منطقی مختلف در آن ممکن می‌سازد.

طراحی با PLD ، CPLD یا FPGA نیاز به ابزارهای قوی طراحی کامپیوتری CAD دارد تا روال طراحی آسان شود. ابزارهای متعددی مانند بسته‌های ورود شماتیک و زبان‌های توصیف سخت‌افزاری (HDL) مانند ABEL ، VHDL . Verilog موجودند. ابزارهای طراحی موجود بلوک‌های منطقی را مستقر، آرایش و به هم وصل می‌کنند تا با یک زبان سطح بالا در HDL تحقق یابد.

مسائل

۷-۱ واحدهای حافظه زیر با تعداد کلمات و تعداد بیت در هر کلمه مشخص می‌شود. در هر حالت چند خط آدرس و چند خط ورودی- خروجی داده وجود دارد؟
(الف) 16 × 4K ، (ب) 8 × 2G ، (پ) 32 × 16M ، (ت) 64 × 256K

۷-۲ تعداد بایت‌های ذخیره شده در حافظه‌های مسئله ۱-۷ را مشخص کنید.

۷-۳ شماره کلمه 723 حافظه در شکل ۳-۷، حاوی معادل دودویی 3451 است. آدرس 10 بیتی و محتوای حافظه 16 بیتی کلمه را لیست نمایید.

۷-۴ شکل موج زمانبندی سیکل حافظه برای اعمال نوشتن و خواندن را نشان دهید. ساعت CPU را 25MHz و سیکل حافظه را 60ns در نظر بگیرید.

۷-۵ یک برنامه تست برای حافظه توصیف شده در مثال ۱-۷ HDL بنویسید. برنامه تست عدد دودویی 7 را در آدرس 0 ، و 14 دودویی را در آدرس 60 ذخیره می‌کند. آنگاه دو آدرس برای اطمینان از ذخیره شدن خوانده می‌شوند.

۷-۶ حافظه RAM 4 × 4 شکل ۶-۷ را در یک نمودار بلوکی محصور کنید و تمام ورودی‌ها و خروجی‌ها را نشان دهید. با فرض خروجی‌های سه حالته، یک حافظه 8 × 8 با چهار RAM 4 × 4 بسازید.

۷-۷ یک حافظه 4 × 16K از یک روش دیکدکردن متقارن (coincident) با تقسیم دیکدر داخلی به دو بخش X و Y استفاده می‌کند.
(الف) سایز هر دیکدر چقدر است و چندگیت AND برای دیکد آدرس لازم است؟
(ب) وقتی آدرس ورودی معادل 6000 دودویی است خطوط انتخاب X و Y فعال شده را معین کنید.

۷-۸ (الف) چند تراشه RAM 8 × 32K برای تهیه یک حافظه با ظرفیت 256K لازم است؟

(ب) چند خط آدرس برای دستیابی به 256K بایت لازم است؟ چند خط از آنها به ورودی‌های آدرس همه تراشه‌ها متصل می‌شوند.

(پ) برای ورودی‌های انتخاب تراشه چند خط باید دیکدگردد.

۷-۹ یک تراشه DRAM از مولتی پلکس کردن دوبعدی استفاده می‌کند. این تراشه دارای 13 خط آدرس مشترک است که در آن آدرس سطر 1 بیت بیشتر است. ظرفیت حافظه چقدر است.

۷-۱۰ با فرض کلمه داده 8 بیت 01011011، یک کلمه مرکب 13 بیت برای کد همینگ تولید کنید تا یک خطا را اصلاح و دو خطا را تشخیص دهد.

۷-۱۱ کد همینگ 15 بیت را برای کلمه داده 11 بیت 11001001010 بدست آورید.

۷-۱۲ یک کد همینگ 12 بیت حاوی 8 بیت داده و 4 بیت توازن از حافظه خوانده شده است. اگر کلمه 12 بیت خوانده شده برابر زیر باشد کلمه داده 8 بیت اولیه نوشته در حافظه چیست:

(الف) 000011101010 (ب) 101110000110 (پ) 101111110100

۷-۱۳ چند بیت چک یا تست توازن باید در کلمه داده لحاظ شود تا برای موارد زیر یک خطا تصحیح و دو خطا تشخیص داده شود (الف) 16 بیت (ب) 32 بیت (پ) 48 بیت.

۷-۱۴ در صورت لزوم کد همینگ را برای چهار بیت D_3، D_5، D_6 و D_7 همراه با سه بیت توازن P_1، P_2 و P_4 فرموله کنید.

(الف) کلمه کد مرکب را برای کلمه داده 0010 بدست آورید.

(ب) سه بیت تست C_4، C_2 و C_1 را با فرض نبود خطا ارزیابی نمایید.

(پ) فرض کنید هنگام نوشتن در حافظه خطایی در بیت 5 رخ داده باشد. نشان دهید که چگونه خطا در این بیت شناسایی و اصلاح شده است.

(ت) بیت توازن P_8 را برای تشخیص دو بیت خطا در کد لحاظ کنید. فرض کنید که خطاها در بیت‌های P_2 و D_5 رخ دهد. نشان دهید که چگونه دو خطا تشخیص داده می‌شوند.

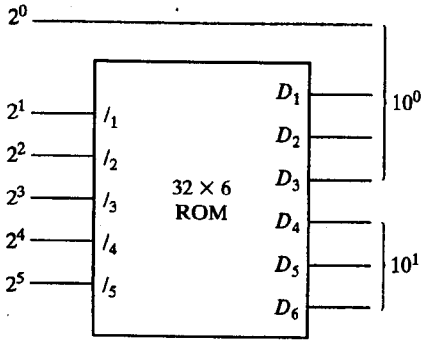
۷-۱۵ با فرض تراشه ROM 8 × 32 با یک ورودی فعال‌ساز، اتصالات بیرونی لازم برای ساخت یک ROM 8 × 128 را نشان دهید. از چهار تراشه و یک دیکدر استفاده کنید.

۷-۱۶ یک تراشه ROM با 8 × 4096 بیت دارای دو ورودی انتخاب تراشه بوده و با یک منبع تغذیه 5V کار می‌کند. برای بسته مدار مجتمع چند پایه لازم است؟ نمودار بلوکی را ترسیم کنید و همه ورودی‌ها و پایانه‌ها را در ROM علامت بزنید.

۷-۱۷ یک ROM 6 × 32 همراه با خط 2^0 مطابق شکل (م۷-۱۷)، یک عدد دودویی 6 بیت را به عدد BCD متناظرش تبدیل می‌کند. مثلاً 100001 (33 دهدهی به 0011 011 در BCD تبدیل می‌شود) جدول درستی ROM را مشخص نمایید.

۷-۱۸ سایز ROM (تعداد کلمات و تعداد بیت در هر کلمه) لازم که بتواند جدول درستی برای مدارهای ترکیبی زیر را در خود جای دهد چیست:

(الف) یک ضرب کننده که دو عدد 4 بیت را در هم ضرب کند.



شکل (م ۷-۱۷)

(ب) یک جمع- تفریق‌گر 4 بیت

(پ) یک مولتی پلکسر 2 به 1 چهارتایی با ورودی‌های فعال‌ساز و انتخاب مشترک

(ت) یک دیکدر BCD به هفت قسمتی با یک ورودی فعال‌ساز

۷-۱۹ جدول درستی یک ROM 4 × 8 را که توابع بولی زیر را پیاده‌سازی کند ایجاد کنید.

$$A(x, y, z) = \Sigma(1, 2, 4, 6)$$
$$B(x, y, z) = \Sigma(0, 1, 6, 7)$$
$$C(x, y, z) = \Sigma(2, 6)$$
$$D(x, y, z) = \Sigma(1, 2, 3, 5, 7)$$

اکنون با فرض ROM به عنوان یک حافظه، محتوای آدرس‌های 1 و 4 را معین نمایید.

۷-۲۰ جدول برنامه‌سازی PLA برای چهار تابع بولی لیست شده در مسئله ۷-۱۹ را ایجاد کنید. تعداد جملات ضرب را حداقل کنید.

۷-۲۱ جدول برنامه‌ریزی PLA را برای یک مدار ترکیبی که مربع 3 بیت را تولید کند مشخص کنید. تعداد جملات ضرب را حداقل نمایید. (شکل ۷-۱۲ برای پیاده‌سازی ROM معادل ملاحظه شود).

۷-۲۲ جدول برنامه‌ریزی PLA برای تبدیل BCD به افزونی 3 با توابع بولی ساده شده در شکل ۳-۴ را لیست کنید.

۷-۲۳ مسئله ۷-۲۲ را با PAL تکرار نمایید.

۷-۲۴ در زیر جدول درستی یک مدار ترکیبی 3 ورودی، 4 خروجی نشان داده شده است. جدول برنامه‌ریزی PAL را برای مدار نشان دهید و نقشه فیوز در نمودار PAL را مشابه شکل ۷-۱۷ علامت بزنید.

ورودی‌ها			خروجی‌ها			
x	y	z	A	B	C	D
0	0	0	0	1	0	0
0	0	1	1	1	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	0	0	1
1	1	0	1	1	1	0
1	1	1	0	1	1	1

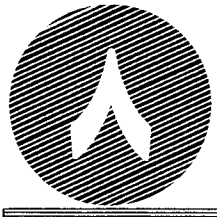
۷-۲۵ با به‌کارگیری ماکروسل ثباتی شکل ۱۹-۷، نقشه فیوز برای مدار ترتیبی حاوی دو ورودی x و y و یک فلیپ فلاپ A که با معادله D_A ورودی زیر تعریف شده را نشان دهید.

$$D_A = x \oplus y \oplus A$$

۷-۲۶ نمودار PAL شکل ۱۶-۷ را با لحاظ سه فلیپ فلاپ ساعت‌دار D بین گیت‌های OR و خروجی‌ها طبق ۱۹-۷ اصلاح کنید. نمودار باید با نمودار بلوکی مدار ترتیبی همخوانی داشته باشد. برای انجام این کار سه گیت بافر- وارونگر و سه خط عمودی برای خروجی‌های فلیپ فلاپ‌هایی که از طریق اتصالات برنامه‌پذیر به آرایه AND وصل می‌شوند لازم است. با به‌کارگیری نمودار PAL ثباتی اصلاح شده، نقشه فیوزی که سه شمارنده آرایه‌ای 3 بیت و یک نقلی خروجی را پیاده‌کند نشان دهید.

مراجع

1. TOCCI, R. J. and N. S. WIDMER. 2001. *Digital Systems Principles and Applications*, 8th ed. Upper Saddle River, NJ: Prentice Hall.
2. KITSON, B. 1984. *Programmable Array Logic Handbook*. Sunnyvale, CA: Advanced Micro Devices.
3. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices*, 3rd ed. Upper Saddle River, NJ: Prentice Hall.
4. NELSON, V. P., H. T. NAGLE, J. D. IRWIN, and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design*. Upper Saddle River, NJ: Prentice Hall.
5. HAMMING, R. W. 1950. Error Detecting and Error Correcting Codes. *Bell Syst. Tech. J.* 29: 147-160.
6. LIN, S., and D. J. COSTELLO, J R. 1983. *Error Control Coding*. Englewood Cliffs, NJ: Prentice-Hall.
7. 1988. *Programmable Logic Data Book*. Dallas: Texas Instruments.
8. TRIMBERGER, S. M. 1994. *Field Programmable Gate Array Technology*. Boston: Kluwer Academic Pub.
9. 1994. *The Programmable Logic Data Book*, 2nd ed. San Jose, CA: Xilinx, Inc.
10. 1986. *Memory Components Handbook*. Santa Clara, CA: Intel.



سطح انتقال ثباتی

۸-۱ نمایش در سطح انتقال ثباتی

یک سیستم دیجیتال مداری است ترتیبی که از فلیپ فلاپ‌ها و گیت‌ها ساخته می‌شود. مدارهای ترتیبی را می‌توان با جداول حالت فصل ۵ نمایش داد. نمایش سیستم‌های دیجیتال با یک جدول درستی اگر غیرممکن نباشد، مشکل هست، زیرا تعداد حالات بسیار زیاد خواهد بود. برای غلبه بر این مشکل، سیستم‌های دیجیتال با روش مدولی طراحی می‌شوند. کل سیستم به چند زیرسیستم مدولی تقسیم می‌شود، که هر یک کار معینی را انجام می‌دهد. مدول‌ها از وسایل دیجیتالی همچون، ثبات‌ها، دیکدرها، مولتی پلکسرها، عناصر محاسباتی و مدارکنترل ساخته می‌شوند. این مدول‌ها با مسیرهای داده یا همان پردازشگر و کنترل مشترک به هم وصل می‌شوند تا یک سیستم دیجیتال به وجود آید.

مدول‌های دیجیتال در بهترین فرم براساس ثبات‌ها و عملیاتی که روی داده‌های ذخیره شده در آنها انجام می‌شود تعریف می‌گردند. مثال‌هایی از عملیات ثباتی عبارتند از شیفت یا جابجایی، شمارش، پاک کردن و بار شدن است. فرض بر این است که ثبات‌ها عنصر مبنای سیستم دیجیتال باشند. جریان اطلاعات و پردازش روی داده‌های ذخیره شده را **عملیات انتقال ثباتی** (RTL) گویند. یک سیستم دیجیتال هنگامی در سطح انتقال ثباتی نمایش داده شده است که با سه عنصر زیر مشخص شده باشد.

۱- مجموعه‌ای از ثبات‌ها در سیستم

۲- عملیات انجام شده روی داده‌های ذخیره شده در ثبات‌ها

۳- کنترلی که رشته عملیات را در سیستم نظارت کند

ثبات مجموعه‌ای از فلیپ فلاپ‌هاست که اطلاعات دودویی را ذخیره می‌کند و قادر به انجام یک یا چند عمل جزیی است. یک ثبات می‌تواند اطلاعات جدیدی را بارکند، یا آن را به راست و چپ جابجا کند. یک شمارنده ثباتی است که یک عدد را یکی یکی افزایش می‌دهد. یک فلیپ فلاپ به تنهایی یک ثبات

1 بیت است، که می‌تواند به 1 نشانده شده یا به 0 پاک شود و یا این که متمم گردد. در واقع، فلیپ فلاپ‌ها و گیت‌های مربوط به آنها در هر مدار ترتیبی را می‌توان بنا به تعریف ثبات خواند.

عملیات اجرائی روی اطلاعات ذخیره شده در ثبات‌ها اعمالی ساده است که به طور موازی روی رشته‌ای از بیت‌ها در یک سیکل ساعت انجام می‌شود. نتیجه حاصل از یک عمل ممکن است جایگزین اطلاعات قبلی در یک ثبات شود. به همین ترتیب نتیجه ممکن است به ثبات دیگری رفته و داده قبلی را در ثبات خود بدون تغییر رها کند. مدارهای دیجیتالی که در فصل ۶ معرفی شدند ثبات‌هایی هستند که عملیاتی ابتدایی را انجام می‌دهند. یک شمارنده با بار شدن موازی قادر است افزایش و بار شدن را اجرا نماید. یک شیفت رجیستر دو طرفه، می‌تواند جابجایی به راست و چپ را انجام دهد. عامل کنترل که رشته عملیات را آغاز می‌نماید، متشکل از سیگنال‌های زمانبندی است و عملیات را به ترتیب و براساس روشی از پیش تعین شده اجرا می‌کند. شرایط خاصی که به نتایج عملیات قبلی وابسته می‌باشد ممکن است رشته عملیات بعدی را تعیین کند. خروجی‌های مدار کنترل متغیرهای دودویی می‌باشند و انواع عملیات را در ثبات آغاز می‌کنند.

انتقال اطلاعات از یک ثبات به ثباتی دیگر به صورت نمادین با استفاده از عملگر جایگزینی مشخص می‌شود. عبارت

$$R2 \leftarrow R1$$

بیانگر یک انتقال از ثبات R_1 به ثبات R_2 است. این عبارت جایگزین شدن محتوای R_2 را با R_1 بیان می‌نماید. بنا به تعریف محتوای ثبات R_1 پس از انتقال تغییری نمی‌کند. فلش بیان‌کننده انتقال و جهت آن است.

عبارتی که انتقال ثباتی را بیانگر است بر این دلالت دارد که مدارهایی از خروجی‌های ثبات مبدأ به ورودی‌های ثبات مقصد وجود دارند و نیز ثبات مقصد دارای قابلیت بار شدن موازی است. معمولاً هدف این نیست که انتقال با هر پالس ساعت رخ دهد، بلکه می‌باید تحت شرایط از پیش تعیین شده‌ای صورت گیرد. یک عبارت شرطی براساس if–then مانند زیر بیان می‌شود.

$$\text{If } (T1 = 1) \text{ then } (R2 \leftarrow R1)$$

که در آن T_1 یک سیگنال کنترل تولید شده در بخش کنترل است. توجه کنید که ساعت به عنوان یک متغیر در نظر گرفته نشده است. فرض بر این است که همه انتقال‌ها در یک گذر لبه ساعت رخ دهند. گرچه یک شرط کنترل ممکن است قبل از لبه ساعت، به وقوع بپیوندد، ولی تا گذر ساعت تکمیل نگردد، انتقال واقعی رخ نخواهد داد.

برای جدایی دو یا چند عمل که به طور همزمان اجرا شوند از ویرگول استفاده می‌کنیم. عبارت زیر را در نظر بگیرید.

$$\text{If } (T3 = 1) \text{ then } (R2 \leftarrow R1, R1 \leftarrow R2)$$

این عبارت تعویض محتوای دو ثبات را در لبه ساعت با شرایط $T_3 = 1$ بیان می‌کند. این گونه اعمال همزمان با ثبات‌هایی امکان‌پذیر است که فلیپ فلاپ‌های حساس به لبه دارند. دیگر انتقال‌های ثباتی

نمونه در زیر معرفی شده‌اند.

$$R1 \leftarrow R1 + R2$$
$$R3 \leftarrow R3 + 1$$
$$R4 \leftarrow \text{shr } R4$$
$$R5 \leftarrow 0$$

جمع با جمع‌کننده موازی دودویی انجام می‌شود، افزایش با یک شمارنده و جابجایی به کمک شیفت رجیستر صورت می‌گیرد. انواع عملیاتی که معمولاً در سیستم‌های دیجیتال با آن مواجه می‌شویم به چهار گروه زیر دسته‌بندی می‌شوند:

۱- عملیات انتقال که داده را از یک ثبات به ثبات دیگر منتقل می‌نماید.
۲- عملیات حسابی که محاسبات را روی داده‌های ثبات‌ها اجرا می‌نمایند.
۳- عملیات منطقی که اعمال بیتی داده‌های غیر عددی را در ثبات‌ها انجام می‌دهند.
۴- عملیات شیفت یا جابجایی که داده را در ثبات جابجا می‌کنند.

عملیات انتقال محتوای اطلاعاتی داده جابجا شده از ثبات مبدأ به ثبات مقصد را تغییر نمی‌دهد. سه نوع دیگر محتوای اطلاعاتی را در حین جابجایی عوض می‌کنند. علائم و نشانه‌های انتقال ثبات به کار رفته در عملیات انتقال داده استاندارد نیستند. در اینجا دو نوع از نشانه‌ها معرفی شده تا سیستم‌های دیجیتال در سطح انتقال ثباتی توصیف گردند. بخش بعدی سمبل‌های RTL به کار رفته در Verilog HDL را معرفی می‌کنند.

۲-۸ سطح انتقال ثباتی (RTL)

سیستم‌های دیجیتال در سطح انتقال ثباتی با استفاده از زبان توصیف سخت‌افزاری توصیف می‌گردند. در Verilog HDL ، توصیف‌های سطح انتقال ثباتی (RTL) ترکیبی از ساختار رفتاری و جریان داده را به کار می‌برند. انتقال‌های ثباتی با عبارات تخصیص اجرایی مشخص می‌شدند. توابع مدارهای ترکیبی با تخصیص مداوم یا عبارات تخصیص اجرایی مشخص شدند. سمبل به کار رفته برای یک انتقال یک علامت مساوی یا فلش است. همزمانی با ساعت با استفاده از عبارت always همراه با کنترل واقعه Posedge یا negedge حاصل می‌گردد. مثال‌های زیر راه‌های ممکن موجود برای یک انتقال در Verilog HDL را نشان می‌دهد.

```
assign  S = A + B;          Continuous assignment          تخصیص مداوم

always @ (A or B)           Procedural assignment (without a clock)
    S = A + B;                                          تخصیص اجرایی

always @ (posedge clock)    Blocking procedural assignment
  begin                                                 تخصیص اجرایی بلوکی
   RA = RA + RB;
   RD = RA;
  end
```

```
always @ (negedge clock)   Non-blocking procedural assignment
  begin                                    تخصیص اجرایی غیربلوکی
   RA <= RA + RB;
   RD <= RA;
  end
```

تخصیص‌های مداوم یا پیوسته برای مدارهای ترکیبی به کار می‌روند. عبارت تخصیص قبلی یک جمع دودویی، با ورودی‌های A و B و خروجی S را نشان می‌دهد. عملوند هدف در یک عبارت تخصیص (در اینجا S) نمی‌تواند یک ثبات باشد. خروجی‌های مدارهای ترکیبی نمی‌توانند با عبارت اجرایی ساعت‌دار به یک ثبات انتقال یابند. تخصیص‌های اجرایی غیرساعت‌دار در مثال دوم راه دیگری را برای مشخص کردن یک مدار ترکیبی نشان می‌دهد.

دو نوع تخصیص اجرایی وجود دارد: بلوکی و غیربلوکی. این دو با سمبل‌های مربوطه‌شان از یکدیگر تفکیک می‌شوند. تخصیص‌های بلوکی از سمبل (=) به عنوان عملگر انتقال و تخصیص‌های غیربلوکی از (= >) به عنوان عملگر استفاده می‌نمایند. عبارات تخصیص بلوکی به طور متوالی طبق لیست در بلوک ترتیبی اجرا می‌شوند. تخصیص‌های غیربلوکی عبارات را از سمت راست ارزیابی می‌کنند ولی تا تمام عبارات ارزیابی نشوند تخصیصی به سمت چپ اختصاص نمی‌دهند. دو مثال فوق را در نظر بگیرید. در تخصیص اجرایی بلوکی، اولین عبارت حاصل جمع را به RA و دومین عبارت مقدار جدید RA را به RD منتقل می‌کند. در پایان، هر دو RA و RD مقدار یکسانی دارند. در تخصیص اجرایی غیربلوکی، دو عمل به طور همزمان انجام می‌گردد بنابراین RD مقدار اولیه RA را دریافت خواهد کرد.

برای اطمینان از عملیات همزمان در طراحی RTL ، لازم است تخصیص‌های اجرایی غیربلوکی برای متغیرهایی به کار روند که عبارت always–clocked را دنبال می‌کنند. دلیل این است که از احتمال تضاد بین مدل طراحی و توصیف HDL جلوگیری شود. تخصیص غیربلوکی ظاهر شده در عبارت always–clocked رفتار یک مدار ترتیبی همزمان را به دقت مدل‌سازی می‌کند.

عملگرهای HDL

عملگرهای HDL –Verilog و سمبل‌های به کار رفته آنها در طراحی RTL در جدول ۱-۸ لیست شده‌اند. عملگرهای حسابی، منطقی و جابجایی برای توصیف عملیات انتقال ثباتی لازمند. عملگرهای منطقی و ارتباطی برای مشخص کردن شرایط کنترل مورد نیازند. عملیات حسابی با اعداد دودویی انجام می‌شوند. اعداد منفی به صورت متمم 2 معرفی می‌گردند. عملگر **modulus** باقیمانده و حاصل از تقسیم دو عدد را تولید می‌نماید. مثلاً باقیمانده 3 ÷ 14 برابر 2 خواهد بود.

دو نوع عملگر منطقی وجود دارد: بیتی و کاهشی. عملگرهای بیتی عمل منطقی بیت به بیت را روی دو عملوند انجام می‌دهند. آنها یک بیت را از یک عملوند بدست آورده و عمل را با بیت متناظرش در عملوند دیگر، اجرا می‌کنند. عملگرهای کاهشی اعمال را روی یک عملوند تک انجام می‌دهند. آنها عمل را بیت به بیت از راست به چپ اجرا می‌کنند و نتیجه 1 بیتی را بدست می‌آورند. مثلاً NOR (~1) با

جدول ۱-۸. عملگرهای Verilog HDL

نوع عملگر	سمبل	عمل اجرا شده
حسابی	+	جمع
	-	تفریق
	*	ضرب
	/	تقسیم
	%	مدولوس
منطقی	~	نفی (متمم)
بیتی	&	AND
or	|	OR
کاهش	^	OR (XOR)
منطقی	!	نفی
	&&	AND
	||	OR
جابجایی	>>	جابجایی به راست ست
	<<	جابجایی به چپ پ
	{ , }	ادغام
نسبی (ارتباطی)	>	بزرگتر از
	<	کوچکتر از
	==	تساوی
	!=	نامساوی
	>=	بزرگ تر از یا مساوی
	<=	کوچکتر از یا مساوی

عملوند 00101 نتیجه 0 و با عملوند 00000 نتیجه 1 را می‌دهد. از دستور **Negate** به عنوان یک عملگر کاهشی در اینجا استفاده نشده است. جدول درستی عملگرهای بیتی در جدول ۹-۴ بخش ۱۱-۴ ملاحظه شد.

عملیات منطقی و ارتباطی می‌توانند از متغیرها یا عبارات به عنوان عملوند استفاده کنند. این اعمال اصولاً برای تعیین حالات صحیح و غلط به کار می‌روند. این عملیات اگر شرایط صحیح باشد 1 و اگر غلط باشند، مقدار 0 را نتیجه می‌دهند. اگر شرط نامعین باشد x را خواهند داد. وقتی عملوند یک عدد است، اگر عدد برابر 0 باشد، 0 را تولید می‌نمایند و اگر غیرصفر باشد 1 را بدست می‌دهند. مثلاً اگر A = 1010 و B = 0000 باشد، آنگاه A به عنوان 1 (عدد غیرصفر است) و B به عنوان 0 تلقی می‌شوند. نتایج دیگر اعمال با این مقادیر عبارتند از:

```
A && B = 0
A || B = 1
!A = 0
!B = 1
(A > B) = 1
(A == B)= 0
```

عملگرهای جابجایی یک عملوند برداری را به دفعات مشخص شده‌ای به راست یا چپ جابجا می‌کنند. بیت‌های تخلیه شده با صفرها پر می‌گردند. مثلاً اگر R = 11010 باشد، عبارت

```
R = R >> 1;
```

R را یک بار به راست جابجا می‌کند. مقدار جدید R برابر 01101 است. عملگر ادغام مکانیزمی برای عملگرهای چندگانه ضمیمه فراهم می‌نماید. از این عملگرها می‌توان برای جابجایی و از جمله، بیت‌های انتقالی به مکان‌های خالی در شیفت رجیستر استفاده کرد. این نکته در مثال ۱-۶ HDL برای شیفت رجیستر نشان داده شده است.

عبارات حلقه‌زنی

Verilog HDL چهار نوع حلقه دارد که امکان اجرای مکرر عبارات اجرایی را میسر می‌سازد. این عملگرها عبارتند از: repeat ، forever ، while و for . همه عبارات حلقه‌زنی باید در بلوک initial یا always قرار گیرند.

حلقه repeat عبارات مربوطه را به دفعات مشخصی اجرا می‌کند. در زیر مثالی که قبلاً به کار رفته است ملاحظه می‌گردد:

```
initial
   begin
       clock = 1'b0;
       repeat (16)
       #5 clock = ~ clock;
   end
```

این بلوک هشت سیکل ساعت را با سیکل زمانی 10 واحد زمانی تولید می‌نماید.

حلقه forever اجرای مداوم یک عبارت اجرا را ایجاد می‌نماید. مثلاً حلقه زیر یک ساعت دائمی ایجاد می‌نماید:

```
initial
   begin
     clock = 1'b0;
     forever
     # clock = ~ clock;
   end
```

حلقه while اجرای یک عبارت یا بلوکی از عبارات را تکرار می‌نماید به شرطی که عبارتی صحت داشته باشد. در صورت غلط بودن شرط، عبارت هرگز اجرا نمی‌شود. در زیر استفاده از حلقه while نشان داده شده است.

```
integer count;
initial
   begin
     count = 0;
     while (count < 64)
     count = count + 1;
   end
```

مقدار شمارش از 0 تا 63 افزایش می‌یابد. حلقه در شماره 64 خارج می‌شود. هنگام برخورد با عبارات حلقه، بهتر است از داده نوع صحیح برای دستکاری کمیت‌ها استفاده شود. اعداد صحیح با کلمه کلیدی integer اعلان می‌شوند و در مثال قبل نیز مشاهده گردید. گرچه از کلمه کلیدی reg می‌توان برای متغیرها استفاده کرد، ولی بهتر است از متغیرهای صحیح در شمارش استفاده شود. متغیرهایی که از نوع داده reg هستند، به عنوان اعداد بی‌علامت ذخیره می‌شوند. آنهایی که از نوع integer هستند به صورت اعداد علامت‌دار و به فرم متمم 2 ذخیره می‌گردند. عرض پیش فرض یک عدد صحیح حداقل 32 بیت است.

حلقه for حاوی سه بخش است که با دو نقطه ویرگول از هم جدا شده‌اند.

- یک مقدار یا شرایط اولیه
- عبارتی برای تست پایان شرط
- عبارتی برای تغییر متغیر کنترل

در زیر مثالی برای حلقه for ارائه شده است.

```
for(i = 0; i < 8; i + 1)
procedural statements
```

عبارت حلقه، عبارات اجرایی را هشت بار تکرار می‌کند. متغیر کنترل i است و مقدار اولیه $i = 0$ می‌باشد، و حلقه مادامی که i کوچکتر از 8 باشد، تکرار خواهد شد. هر بار حلقه تکرار شود i یک واحد اضافه می‌شود.

توصیف یک دیکدر با 2 به 4 با استفاده از حلقه for در مثال ۱-۸ HDL نشان داده شده است. چون خروجی Y در یک عبارت اجرایی ارزیابی شده است، باید به صورت reg اعلان شود. متغیر کنترل برای حلقه یک عدد صحیح است. وقتی که حلقه گسترش یابد چهار حالت زیر را بدست خواهیم آورد (IN و Y دودویی و اندیس Y دهدهی است):

if $IN = 00$ then $Y(0) = 1$ else $Y(0) = 0$
if $IN = 01$ then $Y(1) = 1$ else $Y(1) = 0$
if $IN = 10$ then $Y(2) = 1$ else $Y(2) = 0$
if $IN = 11$ then $Y(3) = 1$ else $Y(3) = 0$

مثال ۱-۸، HDL

```
//description of 2x4 decoder
//using for loop statement
module decoder (IN, Y);
   input  [1:0] IN;     //Two binary inputs
   output [3:0] Y;     //Four binary outputs
   reg [3:0] Y;
   integer I;           //control variable for loop
   always @ (IN)
     for (I = 0; I <= 3; I = I + 1)
       if (IN == I) Y[I] = 1;
       else Y[I] = 0;
endmodule
```

سنتز منطقی

سنتز منطقی یک فرآیند اتوماتیک در انتقال یک توصیف زبان سطح بالا مانند HDL به یک netlist بهینه شده از گیت‌ها است که عملیات مشخص شده با کد منبع (source code) را اجرا می‌نمایند. تکنولوژی‌های موردنظر متعددی در پیاده‌سازی طرح وجود دارد. استفاده مؤثر از یک توصیف HDL لازم می‌دارد تا طراحان یک سبک خاص فروشنده را پیش گیرند تا به عنوان ابزار طراحی خاص مناسب باشد. نوع آی‌سی‌هایی که طرح را پیاده می‌کنند ممکن است مدار مجتمع از نوع کاربرد خاص (ASIC)، یک وسیله منطقی برنامه‌پذیر (PLD)، یا یک آرایه گیتی برنامه‌پذیر منطقی (FPGA) باشد.

ابزار طراحی منطقی برنامه‌هایی هستند که کد منبع، یک زبان توصیف سخت‌افزاری را تفسیر کرده و آن را به یک ساختار گیتی ترجمه یا بدل می‌کنند. طراحی‌های نوشته شده در HDL برای اهداف طراحی منطقی در سطح انتقال ثباتی می‌باشند. دلیل این است که ساختارهای HDL به کار رفته در توصیف RTL به راحتی قابل تبدیل به توصیف سطح گیتی می‌باشند.

عبارت assign برای توصیف مدارات ترکیبی به کار می‌رود. یک عبارت assign با توابع بولی به فرم مدار گیتی مربوطه تفسیر می‌گردد. یک عبارت با یک علامت بعلاوه (+) به صورت یک جمع دودویی با مدار جمع‌کننده کامل تفسیر می‌شود. عبارتی با علامت منها (−) به تفریق‌گری متشکل از جمع‌کننده کامل و گیت‌های XOR (شکل ۱۳-۴) تبدیل می‌گردد.

یک عبارت با عملگر شرطی مثل

```
assign Y = S ? I1 : I0;
```

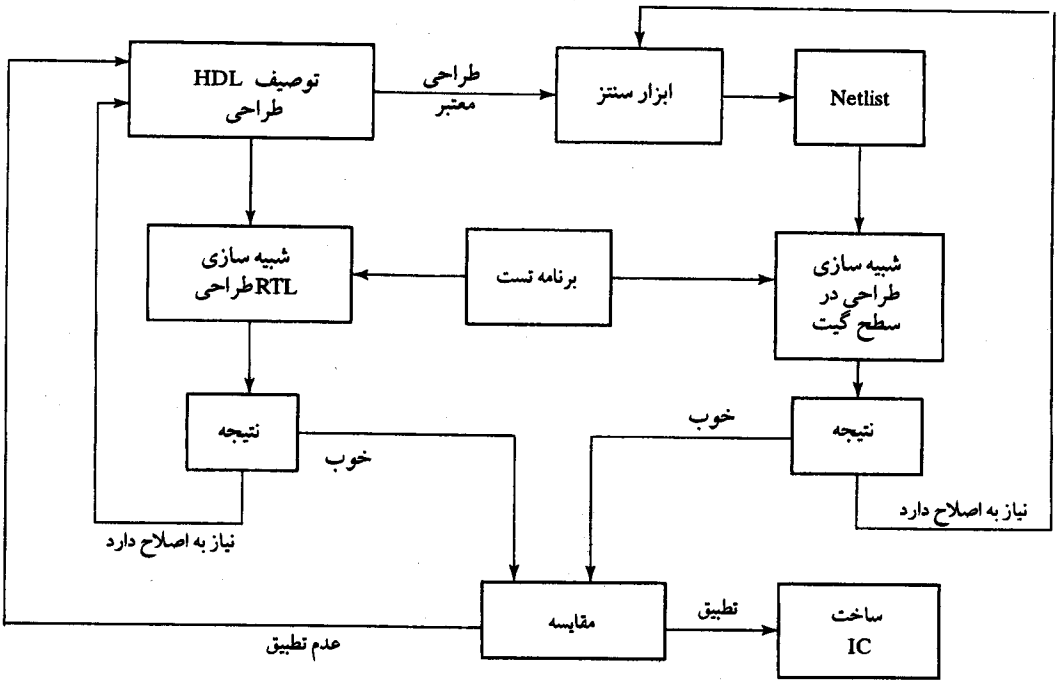
به یک مولتی پلکسر با ورودی کنترل S و ورودی‌های داده I_1 و I_0 برگردانده می‌شود. عبارت با عملگرهای شرطی چندگانه مولتی پلکسرهای بزرگتر را مشخص می‌نماید.

عبارت always ممکن است مداری ترکیبی یا ترتیبی را تداعی کند. برای مدارهای ترتیبی، کنترل واقعه باید لبه posedge یا negedge ساعت باشد، در غیر این صورت عبارت اجرایی یک مدار ترکیبی را مشخص خواهد کرد. مثلاً

```
always @ (I1 or I0 or S)
      if (S) Y = I1;
      else   Y = I0;
```

به یک مولتی پلکسر 2 به 1 ترجمه می‌گردد. عبارت case برای مولتی پلکسرهای بزرگتر به کار می‌رود. ساعت always @ posedge یا negedge مدار ترتیبی ساعت‌دار را مشخص می‌نماید. مدارهای مربوطه متشکل از فلیپ فلاپ‌های D و گیت‌هایی هستند که عملیات انتقال ثبات را پیاده می‌کنند. مثال‌هایی از اینگونه مدارها، ثبات‌ها و شمارنده‌ها می‌باشند. یک عبارت توصیف مدار ترتیبی، به یک مدار کنترل با فلیپ فلاپ D و گیت‌ها ترجمه می‌شود. بنابراین هر عبارت در یک توصیف RTL با سنتزگر تفسیر شده و به مدار گیتی و فلیپ فلاپ مربوطه تخصیص می‌یابد.

فلوچارت ساده شده فرآیند طراحی در شکل ۱-۸ نشان داده شده است. توصیف RTL طرح HDL شبیه‌سازی شده و برای اطمینان از صحت عملکرد چک می‌شود. برنامه تست سیگنال تحریک را برای



شکل ۱-۸. فرآیند شبیه سازی و سنتز HDL

شبیه‌سازی فراهم می‌نماید. اگر نتیجه شبیه‌سازی رضایت‌بخش نباشد، توصیف HDL اصلاح شده و دوباره چک می‌گردد. وقتی که نتایج شبیه‌سازی یک طرح معتبری را نشان دهد، توصیف RTL به سنتزگر (سیستم طراحی) منطقی اعمال می‌شود. ابزار سنتزگر، یک netlist معادل با توصیف سطح گیت طرح را تولید می‌کند. مدار سطح گیت، با همان محرک طرح RTL شبیه‌سازی می‌گردد. اگر تصحیحاتی لازم باشد، عمل شبیه‌سازی تا دستیابی به یک شبیه‌سازی مطلوب ادامه می‌یابد. نتایج دو شبیه‌سازی با هم مقایسه می‌گردند تا تطابق آنها مشاهده گردد. اگر به هم نزدیک نباشند طراح برای تغییر توصیف HDL بازگشته و هر خطایی را اصلاح می‌نماید. آنگاه دوباره توصیف به سنتزگر منطقی برده می‌شود تا یک توصیف سطح گیت جدید تولید گردد. به محض رضایت طراح از نتایج همه شبیه‌سازی‌ها، مدار قابل ساخت با یک مدار مجتمع است.

سنتزگر منطقی چندین مزیت را داراست. زمان کمتری در نوشتن یک توصیف HDL نیاز دارد و مدار سطح گیت آن مداری را بدست می‌دهد که از روی وارده‌های نمودار طرح آن قابل تهیه است. سادگی در تغییر توصیف امکان تغییر طراحی‌ها را عملی می‌سازد. تست اعتبار توصیف با شبیه‌سازی، ساده‌تر از ارزیابی نمونه سخت‌افزاری تولیدی است. بانک اطلاعاتی برای ساخت مدار مجتمع به طور خودکار با ابزار سنتزگر تولید می‌شود.

۸-۳ ماشین‌های حالت الگوریتمی (ASM)

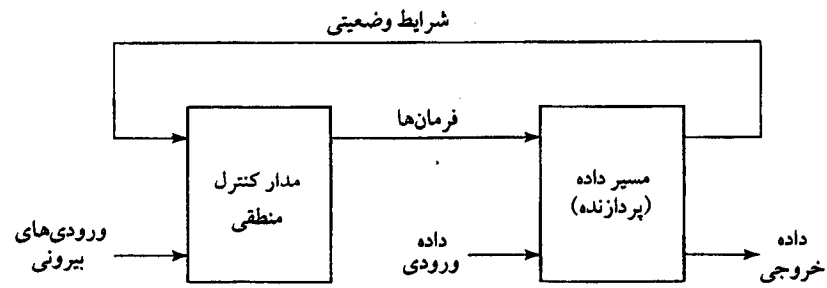
اطلاعات دودویی ذخیره شده در یک سیستم دیجیتال را می‌توان به دو صورت اطلاعات داده و کنترل دسته‌بندی کرد. داده عناصر گسسته‌ای از اطلاعات است که برای انجام اعمال حسابی، منطقی، جابجایی و دیگر عملیات پردازش داده دستکاری می‌شود. این عملیات با عناصر دیجیتالی چون جمع‌کننده‌ها، دیکدرها، مولتی پلکسرها، شمارنده‌ها و شیفت رجیسترها پیاده‌سازی می‌شوند. اطلاعات کنترلی، سیگنال‌های فرمانی که انواع عملیات را در بخش داده مدیریت می‌کنند تولید می‌نمایند تا کار پردازش داده موردنظر انجام شود. طراحی منطقی یک سیستم دیجیتال را می‌توان به دو بخش تقسیم کرد. یک بخش متعلق به مدارهای دیجیتال است که عملیات پردازش داده را انجام می‌دهند. بخش دیگر مربوط به طراحی مدارهای کنترلی است که عملیات مختلف و توالی آنها را مدیریت می‌نماید.

رابطه بین منطق کنترل و پردازش داده در یک سیستم دیجیتال در شکل ۲-۸ مشاهده می‌شود. مسیر پردازش داده، که به آن عموماً مسیر داده می‌گویند، برحسب نیازهای سیستم، داده را در ثبات دستکاری می‌کند. مدار کنترل یک رشته از فرامین را به مسیر داده صادر می‌نماید. مدار کنترل شرایط وضعیت را از مسیر داده دریافت کرده و برای تعیین رشته سیگنال‌های کنترل از آنها استفاده می‌کند.

مدار کنترلی که سیگنال‌هایی را برای توالی عملیات در مسیر داده تولید می‌کند یک مدار ترتیبی است که حالات داخلی‌اش فرامین کنترل را به سیستم دیکته می‌نماید. در هر زمان، حالت کنترل ترتیبی مجموعه‌ای از فرامین از پیش تعیین شده‌ای را آغاز می‌کند. بسته به شرایط وضعیت و دیگر ورودی‌های بیرونی، کنترل ترتیبی به حالت بعدی می‌رود تا اعمال دیگری را در مسیر داده آغاز کند و نیز حالت بعدی خود زیر سیستم کنترل را معین نماید.

وظایف رشته کنترل و مسیر داده سیستم دیجیتال با الگوریتم‌های سخت‌افزاری مشخص می‌گردد. یک الگوریتم متشکل است از تعداد معینی از مراحل اجرایی یا رویه‌ای، که چگونگی تهیه یک حل را برای یک مسئله با یک مقدار تجهیزات پیاده‌سازی می‌کند. ابتکاری‌ترین و مشکل‌ترین بخش از طراحی فرموله کردن الگوریتم‌های سخت‌افزاری برای رسیدن به اهداف است.

فلوچارت روشی مناسب برای مشخص کردن رشته مراحل اجرایی و مسیرهای تصمیم‌گیری در یک الگوریتم است. فلوچارت برای الگوریتم سخت‌افزاری جملات را به یک نمودار اطلاعاتی تبدیل



شکل ۲-۸. واکنش‌بین کنترل و مسیر داده

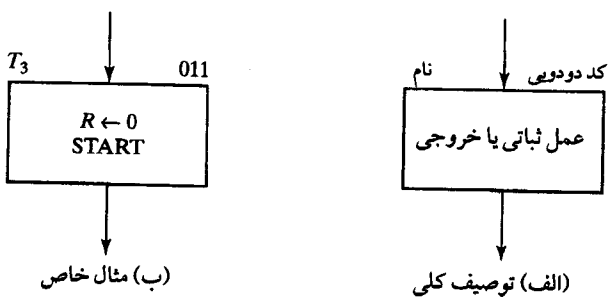
می‌کند که رشته عملیات آنها را به همراه شرایط لازم‌الاجرا، یک به یک می‌شمارد. یک فلوچارت خاص که فقط برای تعریف الگوریتم‌های سخت‌افزاری به وجود آید چارت ماشین حالت الگوریتمی (ASM) نام دارد. ماشین حالت نیز نام دیگری برای یک مدار ترتیبی است که ساختار مبنا برای یک سیستم دیجیتال می‌باشد.

چارت ASM ظاهراً همان چارت معمولی است، ولی تفسیر متفاوتی دارد. فلوچارت معمولی رشته مراحل اجرایی و مسیرهای تصمیم‌گیری را برای یک الگوریتم به ترتیب و بدون ملاحظه ارتباط زمانی آنها، توصیف می‌کند. چارت ASM رشته وقایع به همراه ارتباط زمانی بین حالات یک کنترل‌گر و وقایعی که ضمن رفتن از یک حالت به حالت بعدی رخ می‌دهند را توصیف می‌نماید. این چارت خصوصاً برای مشخص کردن دقیق رشته کنترل و عملیات مسیر داده (واحد پردازش) در یک سیستم دیجیتال انتخاب شده است، ضمن این که محدودیت‌های سخت‌افزار دیجیتال نیز لحاظ شده است.

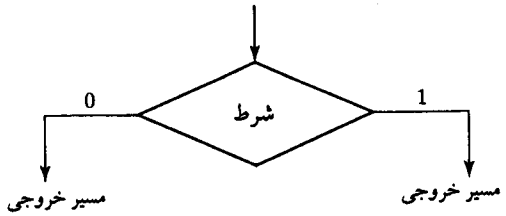
چارت ماشین حالت الگوریتمی

چارت ماشین حالت الگوریتمی (ASM) نوع خاصی از چارت است که برای توصیف عملیات ترتیبی یک سیستم دیجیتال مناسب می‌باشد. چارت از سه جزء مبنا ساخته شده است: جعبه حالت، جعبه تصمیم و جعبه شرط. یک حالت در رشته کنترل با جعبه حالت نشان داده می‌شود، شکل ۳-۸. جعبه حالت مستطیل شکل است و در داخل آن عملیات ثباتی یا نام سیگنال خروجی که کنترل تولید می‌کند نیز در این جعبه قرار دارند. به حالت نام سمبلیکی داده شده که در سمت چپ بالای جعبه نوشته می‌شود، کد تخصیص یافته به حالت در سمت راست بالا قرار می‌گیرد. شکل ۳-۸ (ب) مثال خاصی از یک جعبه حالت را نشان می‌دهد. حالت، نام سمبلیک T_3 را داشته و کد دودویی 011 به آن اختصاص یافته است. در داخل جعبه، عمل ثباتی $R \leftarrow 0$ نوشته شده و به معنی پاک شدن ثبات R به 0، در حالت T_3 است. نام START در داخل جعبه، می‌تواند مثلاً، یک سیگنال خروجی باشد که عمل خاصی را آغاز می‌نماید.

جعبه تصمیم تأثیر یک ورودی را روی زیر سیستم کنترل توصیف می‌کند. جعبه مربوط به آن طبق شکل ۴-۸ به فرم لوزی است که دو یا چند مسیر خروجی را داراست. شرط تست ورودی در داخل جعبه نوشته می‌شود. یکی از خروجی‌ها متعلق به صحت شرط و دیگری متعلق به عدم صحت آن است.



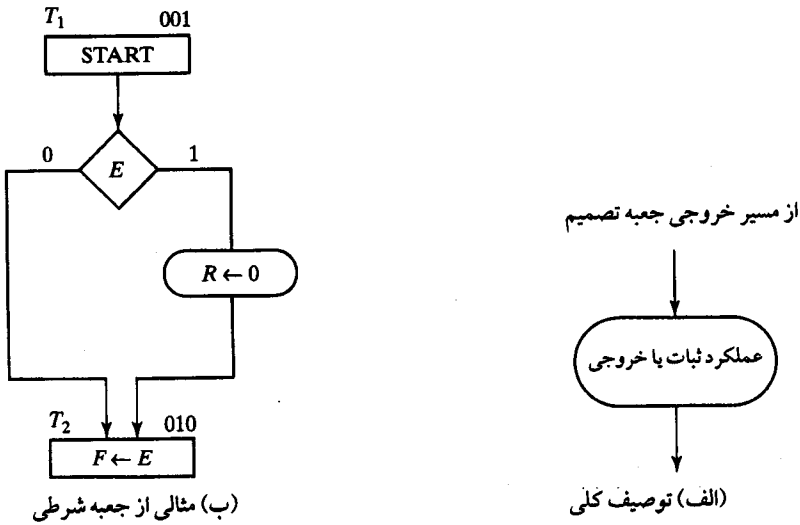
شکل ۳-۸. جعبه حالت



شکل ۴-۸. جعبه تصمیم

اگر به یک ورودی شرط مقدار دودویی اختصاص دهیم، دو مسیر با 1 و 0 مشخص می‌شوند.

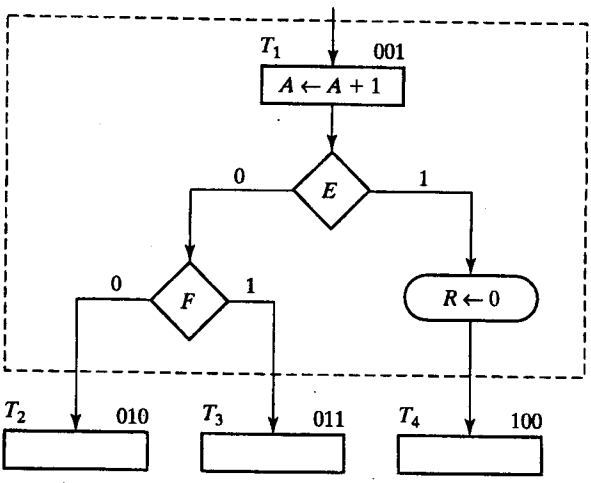
جعبه‌های حالت و تصمیم با توجه به فلوچارت‌های معمولی، آشنا به نظر می‌رسند. سومین عنصر یعنی جعبه شرطی تنها منحصر به چارت ASM است. شکل بیضی مانند جعبه شرطی در شکل ۵-۸ مشاهده می‌شود. گوشه‌های قوسی شکل، آن را از جعبه حالت متمایز می‌سازد. مسیر ورودی به جعبه شرطی باید از یک مسیر خروجی در جعبه تصمیم بیاید. عملیات ثباتی و خروجی‌های لیست شده در جعبه شرطی در طول یک حالت مفروض به شرط برآورده شدن شرط، صورت می‌پذیرد. شکل ۵-۸ (ب) مثالی را با یک جعبه شرطی نشان می‌دهد. کنترل یک سیگنال خروجی START را در حالت T_1 تولید می‌کند. کنترل حالت ورودی E را چک می‌کند. اگر $E = 1$ باشد، آنگاه $R = 0$ خواهد شد، در غیر این صورت R بدون تغییر می‌ماند. در هر دو صورت حالت بعدی T_2 خواهد بود.



شکل ۵-۸. جعبه شرطی

بلوک ASM

یک بلوک ASM ساختاری است متشکل از یک جعبه حالت و تمام جعبه‌های تصمیم و شرطی که به خروجی آن وصل شده‌اند. یک بلوک ASM دارای یک ورودی و تعداد نامشخصی از مسیرهای

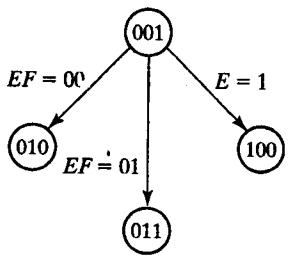


شکل ۶-۸. بلوک ASM

خروجی است که به وسیله ساختار جعبه‌های تصمیم نشان داده می‌شود. مثالی از یک بلوک ASM در شکل ۶-۸ ملاحظه می‌گردد. در ارتباط با حالت T_1 دو جعبه تصمیم و یک جعبه شرطی وجود دارد. خطوط نقطه‌چین حول تمام مجموعه، آن را از بقیه چارت جدا می‌سازد ولی معمولاً این عمل صورت نمی‌گیرد زیرا حالت ASM هر بلوک را منحصراً با توجه به ساختار آن توصیف می‌نماید. یک جعبه حالت بدون جعبه تصمیم و یا شرطی، یک بلوک ساده می‌باشد.

هر بلوک در چارت ASM یک حالت از سیستم را در یک پالس ساعت توصیف می‌نماید. عملیات درون جعبه‌های حالت و شرط در شکل ۶-۸ در یک پالس ساعت مشترک و در حالی که سیستم در T_1 است اجرا می‌گردند. همان پالس ساعت نیز کنترل‌گر سیستم را با توجه به مقادیر دودویی E و F به حالت‌های T_2 ، T_3 و یا T_4 خواهد برد.

چارت ASM خیلی شبیه به نمودار حالت است. هر بلوک حالت معادل با یک حالت در مدار ترتیبی است. جعبه تصمیم معادل با اطلاعات دودویی نوشته شده در امتداد فلشی است که دو حالت را در نمودار حالت بهم وصل می‌کند. در نتیجه گاهی بهتر است چارت را به نمودار حالت تبدیل نموده و سپس روش مدار ترتیبی را برای طراحی مدار کنترل دنبال نماییم. به منظور تشریح، چارت شکل ۶-۸ به صورت نمودار شکل ۷-۸ کشیده شده است. حالات با دوایری همراه با مقادیر دودویی‌شان که در



شکل ۷-۸ . نمودار حالت معادل با چارت ASM در شکل ۶-۸

داخل هر یک از دوایر نوشته شده، نشان داده شده‌اند. خطوط جهت‌دار شرایطی را نشان می‌دهند که حالت بعدی را معین می‌کنند. عملیات شرطی و غیرشرطی که باید اجرا شوند در نمودار حالت آورده نشده‌اند.

بررسی‌های زمانبندی

زمانبندی همه ثبات‌ها و فلیپ فلاپ‌ها به وسیله یک مدار ساعت اصلی کنترل می‌گردد. پالس‌های ساعت نه فقط به بخش مسیر داده بلکه به تمام فلیپ فلاپ‌های مدار کنترل نیز اعمال می‌گردند. ورودی‌ها نیز با پالس‌های ساعت همزمانند زیرا این ورودی‌ها، خروجی‌های مدار دیگری هستند که از همان پالس‌های ساعت استفاده می‌کنند. اگر سیگنال ورودی در یک زمان دلخواه و مستقل از پالس ساعت تغییر یابد آن را ورودی غیرهمزمان یا آسنکرون می‌نامیم. ورودی‌های غیرهمزمان ممکن است موجب بروز انواع مشکلات، طبق آنچه در فصل ۹ خواهیم دید، گردند. برای ساده‌کردن طراحی، فرض خواهیم‌کرد که تمام ورودی‌ها نسبت به پالس ساعت همزمان بوده و با لبه پالس ساعت تغییر حالت می‌دهند.

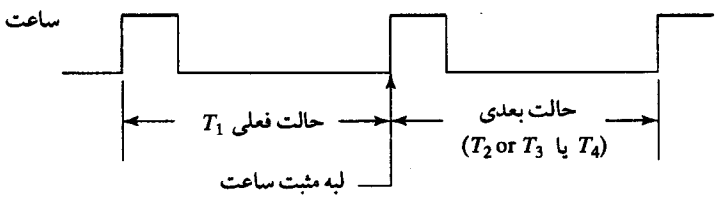
تفاوت عمده بین یک فلوچارت و یک ASM چارت در تفسیر زمانی روابط بین عملیات مختلف است. مثلاً اگر شکل ۶-۸ یک چارت معمولی باشد، عملیات لیست شده در آن یکی پس از دیگری با ترتیب زمانی دنبال می‌شود. با این فرض ابتدا ثبات A افزایش یافته و آنگاه E اضافه می‌گردد. اگر $E = 1$ باشد آنگاه ثبات R پاک شده و کنترل به حالت T_4 می‌رود در غیر این صورت با $E = 0$ در قدم بعدی F ارزیابی شده و به حالت T_2 یا T_3 خواهد رفت. برعکس یک چارت ASM کل بلوک را یک جا و به طور واحد در نظر می‌گیرد. تمام اعمالی که در داخل بلوک باشند به طور همزمان و در لحظه گذر لبه پالس ساعت و در لحظه انتقال سیستم از حالت T_1 به حالت بعدی اتفاق خواهند افتاد. این نکته به طور مصور در شکل ۸-۸ نشان داده شده است. فرض بر این است که لبه مثبت ساعت همه فلیپ فلاپ‌ها را تریگر کند. اولین انتقال مثبت ساعت مدار کنترل را به حالت T_1 می‌برد. ضمن قرارگرفتن سیستم در حالت T_1 مدار کنترل ورودی‌های F و E را چک کرده و سیگنال‌های مناسبی را بر طبق آنها تولید می‌نماید. عملیات زیر در لحظه گذر لبه پالس مثبت بعدی به طور همزمان رخ می‌دهد:

۱- ثبات A افزایش می‌یابد

۲- اگر $E = 1$ باشد ثبات R پاک می‌شود

۳- بسته به مقدار E و F، کنترل به حالت بعدی T_2 یا T_3 یا T_4 منتقل می‌گردد

توجه‌کنید که دو عمل در بخش مسیر داده و یک تغییر حالت در مدار کنترل به طور همزمان اتفاق می‌افتند.



شکل ۸-۸. انتقال بین حالات

۸-۴ مثال طراحی

اکنون اجزاء یک چارت ASM و نمایش انتقال ثبات را با مروری بر یک مثال طراحی نشان خواهیم داد. مثال را با مشخصات اولیه آغاز کرده و کار را با تشکیل یک ASM مناسب که از آن طراحی سخت‌افزار دیجیتال حاصل می‌شود، دنبال می‌کنیم.

می‌خواهیم سیستم دیجیتالی با دو فلیپ فلاپ E و F و یک شمارنده دودویی 4 بیت A طراحی کنیم. فلیپ فلاپ‌های A با A_4، A_3، A_2 و A_1 نام‌گذاری شده‌اند که در آن A_4 باارزش‌ترین بیت شمارش را نگه می‌دارد. یک سیگنال شروع S، عمل سیستم را با پاک کردن شمارنده A و فلیپ فلاپ F آغاز می‌کند. آنگاه شمارش با پالس ساعت بعدی شروع و تا پایان عملیات این افزایش ادامه دارد. بیت‌های A_3 و A_4 توالی عمل را معین می‌کنند.

- اگر $A_3 = 0$ باشد، $E = 0$ شده و شمارش ادامه می‌یابد.
- اگر $A_3 = 1$ باشد $E = 1$ شده؛ سپس اگر $A_4 = 0$ باشد شمارش ادامه می‌یابد، ولی اگر مقدار $A_4 = 1$ باشد در لبه پالس ساعت بعدی $F = 1$ خواهد بود.

سپس اگر S=0باشد، سیستم در حالت شروع باقی می‌ماند، ولی اگر S=1شود سیکل عمل تکرار می‌گردد.

چارت ASM

چارت ASM در شکل ۸-۹ دیده می‌شود. وقتی عملی اجرا نشود، سیستم در حالت اولیه T_0 بوده و منتظر سیگنال شروع S می‌باشد. وقتی ورودی S برابر 1 باشد، شمارنده A و فلیپ فلاپ F به 0 پاک می‌شود و کنترل‌گر به حالت T_1 می‌رود. توجه کنید که جعبه شرطی به دنبال جعبه تصمیم S آمده است. این بدان معنی است که اگر $S = 1$ باشد شمارنده و فلیپ فلاپ در T_0 پاک خواهند شد و در همان زمان کنترل به حالت T_1 خواهد رفت.

بلوک مربوط به حالت T_1 دارای دو جعبه تصمیم‌گیری و دو جعبه شرطی است. شمارنده با هر پالس ساعت افزایش می‌یابد. در همان زمان سه عمل دیگر رخ می‌دهد:

$E = 0$ شده و کنترل در T_1 می‌ماند $(A_3 = 0)$؛ یا

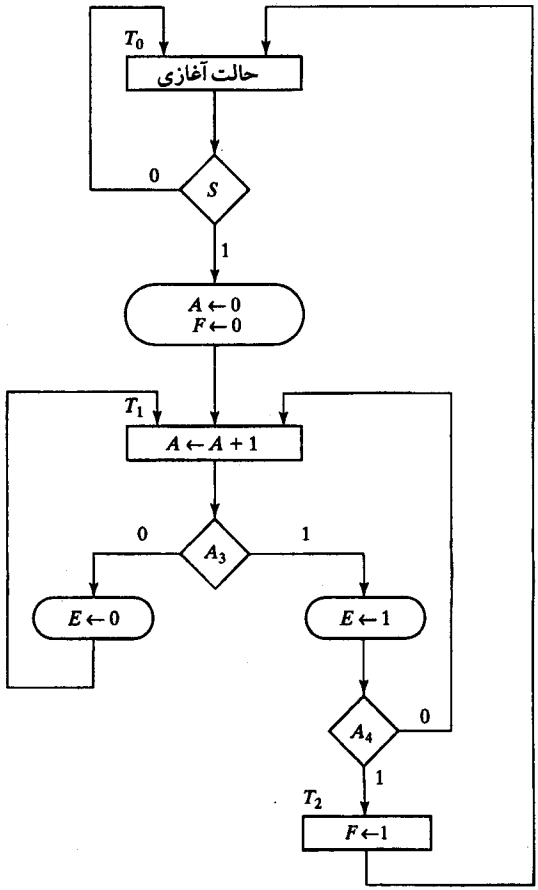
$E = 1$ شده و کنترل در T_1 باقی خواهد ماند $(A_3A_4 = 10)$؛ یا

$E = 1$ شده و کنترل به حالت T_2 می‌رود $(A_3A_4 = 11)$.

وقتی کنترل در حالت T_2 است، فلیپ فلاپ $F = 1$ شده و مدار به حالت اولیه T_0 باز می‌گردد.

چارت ASM از سه حالت و سه بلوک تشکیل شده است. بلوک متعلق به T_0 متشکل از جعبه حالت، یک جعبه تصمیم و یک جعبه شرطی است. بلوک T_2 فقط جعبه حالت دارد. کنترل منطقی یک ورودی بیرونی، S، و دو ورودی حالت A_3 و A_4 دارد.

در اینجا راه ترجمه یا برگرداندن یک توصیف کلمه‌ای را به چارت ASM نشان خواهیم داد. باید توجه داشت که مثال طراحی در چارت ASM کاربرد عملی ندارد و بسته به تفسیر ممکن است ساده و یا فرموله شود. با این وجود، پس از ایجاد چارت ASM، روش طراحی مدار بسیار ساده خواهد بود.



شکل ۹-۸. چارت ASM برای مثال طراحی

ترتیب زمانبندی

هر بلوک در چارت ASM بیانگر عملیاتی است که قرار است در طول یک پالس ساعت صورت گیرد. عملیات مشخص شده در داخل یک جعبه حالت و جعبه‌های شرطی در بخش مسیر داده انجام می‌شود. تغییر از یک حالت به حالت بعدی در مدار کنترل اتفاق می‌افتد. برای درک روابط زمانی موجود، عملیات را از ابتدای پالس شروع (استارت) تا بازگشت به حالت اولیه قدم به قدم پس از هر پالس ساعت لیست خواهیم کرد.

جدول ۲-۸ مقادیر دودویی شمارنده و دو فلیپ فلاپ را پس از هر پالس ساعت نشان می‌دهد. جدول به طور جداگانه وضعیت A_3 و A_4 را به همراه حالت فعلی کنترل‌گر نیز داراست. ما با حالت T_1 و درست پس از این که سیگنال ورودی S شمارنده و فلیپ فلاپ F را پاک می‌کند آغاز می‌کنیم. فرض بر

جدول ۲-۸. رشته‌ای از عملیات برای مثال طراحی

شمارنده				فلیپ فلاپ‌ها			
A_4	A_3	A_2	A_1	E	F	شرایط	حالت
0	0	0	0	1	0	$A_3 = 0, A_4 = 0$	T_1
0	0	0	1	0	0		
0	0	1	0	0	0		
0	0	1	1	0	0		
0	1	0	0	0	0	$A_3 = 1, A_4 = 0$	
0	1	0	1	1	0		
0	1	1	0	1	0		
0	1	1	1	1	0		
1	0	0	0	1	0	$A_3 = 0, A_4 = 1$	
1	0	0	1	0	0		
1	0	1	0	0	0		
1	0	1	1	0	0		
1	1	0	0	0	0	$A_3 = 1, A_4 = 1$	
1	1	0	1	1	0		T_2
1	1	0	1	1	1		T_0

این است که مقدار E برابر 1 باشد زیرا E در T_0 برابر 1 است (در انتهای جدول نشان داده شده است) و مقدار آن در گذر T_0 به T_1 عوض نمی‌شود. سیستم در طول 13 پالس ساعت بعدی در T_1 باقی می‌ماند. هر پالس شمارنده را افزایش داده و E را 0 یا 1 می‌کند. به ارتباط لحظه‌ای که $A_3 = 1$ و لحظه‌ای که $E = 1$ می‌شود توجه نمایید. وقتی $A = 0011$ است، پالس ساعت بعدی شمارنده را به 0100 افزایش می‌دهد، و در همان لحظه $A_3 = 0$ بوده و بنابراین $E = 0$ می‌گردد. پالس بعدی شمارنده را از 0100 به 0101 می‌برد، که در آن $A_3 = 1$ بوده و بنابراین $E = 1$ خواهد شد. به طور مشابه $E = 0$ می‌شود ولی نه هنگامی که شمارش از 0111 به 1000 می‌رود بلکه موقعی که از 1000 به 1001 برود و در این حالت فعلی شمارنده $A_3 = 0$ است.

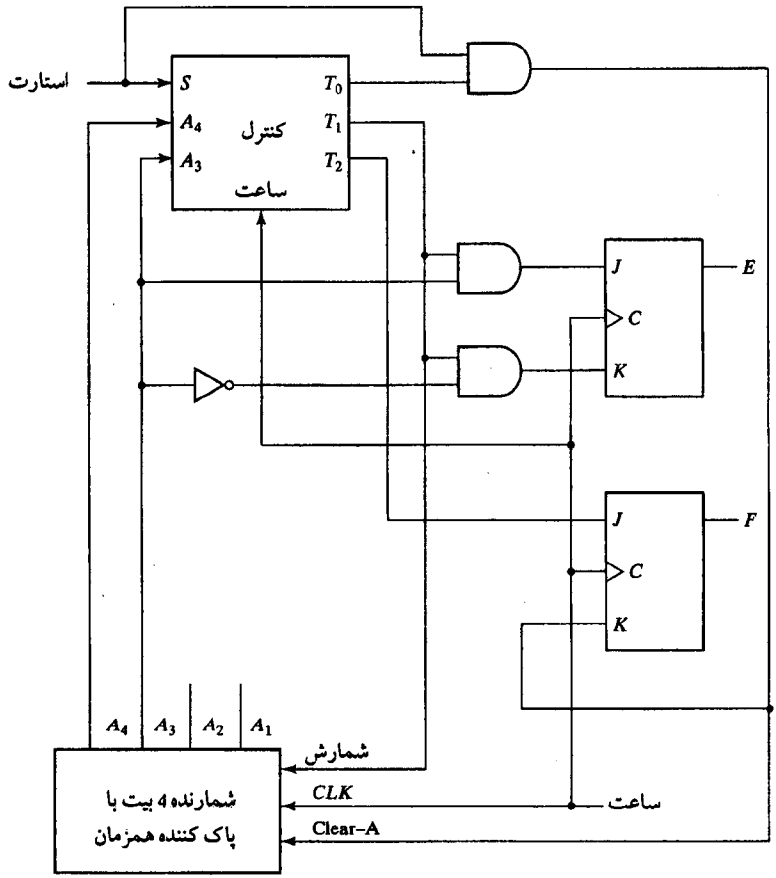
وقتی شمارش به 1100 برسد هر دو مقدار A_3 و A_4 برابر 1 خواهند بود. پالس ساعت بعدی A را یک واحد افزایش داده، E را برابر 1 نموده و کنترل را به حالت T_2 منتقل می‌کند. کنترل برای مدت یک پریود ساعت در T_2 می‌ماند. لبه ساعت متعلق به T_2 فلیپ فلاپ F را 1 می‌کند و کنترل به T_0 باز می‌گردد. سیستم مادامی که $S = 0$ باشد در T_0 باقی خواهد ماند.

با ملاحظه جدول ۲-۸، مشاهده می‌شود که عملیات انجام شده روی E به اندازه یک پالس ساعت به تعویق می‌افتد. این اختلاف بین یک چارت ASM و فلوچارت عادی است. اگر شکل ۹-۸ یک چارت معمولی می‌بود، می‌توانستیم فرض کنیم که ابتدا A افزایش یافته و مقدار افزایش یافته برای چک کردن وضعیت A_3 به کار برده می‌شود. عملیاتی که در سخت‌افزار دیجیتال رخ می‌دهند، همانطور که در بلوک ASM مشخص شده است در یک پالس ساعت بوده و مشابه با آنچه در فلوچارت معمولی بود به

صورت متوالی اتفاق نمی‌افتند. بنابراین مقدار مورد بررسی A_3 در جعبه تصمیم از وضعیت فعلی شمارنده و قبل از افزایش اختیار می‌شود. به این علت است که جعبه تصمیم متعلق به همان بلوکی است که T_1 در آن قرار دارد. مدارات دیجیتال در کنترل، سیگنال‌هایی را برای تمام عملیات مشخص شده در بلوک فعلی و قبل از ورود به بلوک بعدی تولید می‌کنند. لبه ساعت بعدی همه عملیات را در ثبات‌ها، فلیپ فلاپ‌ها، از جمله فلیپ فلاپ کنترل‌گر اجرا می‌کند و حالت بعدی را معین خواهد کرد.

طراحی مسیر داده

چارت ASM همه اطلاعات لازم را برای طراحی سیستم دیجیتال فراهم می‌سازد. نیازهای طراحی مسیر داده در داخل جعبه‌های حالت شرطی مشخص می‌گردد. مدار کنترل از جعبه‌های تصمیم و گذرهای حالت لازم بدست می‌آید. نموداری که سخت‌افزار مثال را نشان می‌دهد در شکل ۱۰-۸ ملاحظه می‌شود. زیر سیستم کنترل با ورودی‌ها و خروجی‌هایش نشان داده شده است. طراحی مشروح کنترل بعداً بررسی شده است. مسیر داده یا همان پردازشگر از شمارنده‌ای چهار بیتی، دو فلیپ فلاپ، و



شکل ۱۰-۸ . پردازشگر داده برای مثال طراحی

تعدادی گیت تشکیل شده است. شمارنده مشابه با آنچه در شکل ۱۲-۶ دیدیم می‌باشد، به جز این که برای پاک کردن همزمان نیاز به گیت‌های اضافی است. شمارنده هر بار که کنترل در T_1 است با هر پالس ساعت افزایش می‌یابد. واضح است که وقتی کنترل در حالت T_0 و کلید S در 1 باشد شمارنده پاک می‌شود. این عملکرد شرطی به یک گیت AND نیاز دارد تا هر دو شرط را تضمین کند. دو عملکرد شرطی دیگر از دو گیت AND دیگر برای نشاندن یا پاک کردن فلیپ فلاپ E استفاده می‌کند. فلیپ فلاپ F بدون شرط در T_2 به 1 نشانده می‌شود. توجه کنید که همه فلیپ فلاپ‌ها و ثبات‌ها از جمله فلیپ فلاپ‌های کنترل از ساعت مشترکی استفاده می‌کنند.

نمایش انتقالی ثباتی

یک سیستم دیجیتال، با مشخص کردن ثبات‌ها در سیستم، عملیات اجرایی و رشته کنترل، در سطح انتقال ثباتی (RTL) نمایش داده می‌شود. عملیات ثباتی و اطلاعات کنترلی را می‌توان با چارت ASM مشخص کرد. گاهی بهتر است تا مدار کنترل و عملیات ثباتی را از مسیر داده جدا کنیم. اطلاعات کنترل و عملیات انتقال ثباتی را می‌توان طبق شکل ۱۱-۸ به طور جداگانه نشان داد. نمودار حالت رشته حالت و عملیات ثباتی با علائم معرفی شده در بخش ۱-۸ نمایش داده می‌شوند. این نمایش راه دیگری نسبت به روش چارت ASM در شکل ۹-۸ می‌باشد. اطلاعات برای نمودار حالت مستقیماً از چارت ASM بدست می‌آید. نام حالت در هر جعبه حالت ذکر شده است. شرایطی که تغییر حالتی را سبب می‌شوند در داخل جعبه‌های تصمیم مشخص شده‌اند. خطوط جهت‌دار بین حالات و شرایط مربوط به آنها مسیر مشابهی را با چارت ASM دنبال می‌کنند. عملیات انتقال ثباتی برای هر یک از سه حالت به دنبال نام حالت و یک جفت نقطه (:) آمده است. زیرا این عملیات از جعبه‌های حالت مستطیلی و جعبه‌های حالت بیضی شکل در چارت ASM حاصل شده است.

جدول حالت

می‌توان نمودار حالت را به جدول حالت تبدیل کرد که از آن مدار ترتیبی کنترل‌گر قابل طراحی است. ابتدا باید مقادیری دودویی را به هر حالت در چارت ASM تخصیص دهیم. برای n فلیپ فلاپ در مدار

T_0: if $(S = 1)$ then $A \leftarrow 0, F \leftarrow 0$

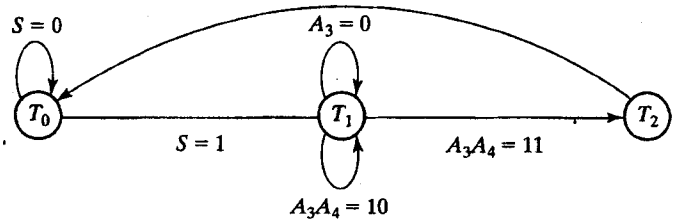
T_1: $A \leftarrow A + 1$

if $(A_3 = 1)$ then $E \leftarrow 1$

if $(A_3 = 0)$ then $E \leftarrow 0$

T_2: $F \leftarrow 1$

(ب) عملیات انتقال ثبات



(الف) نمودار حالت کنترل

شکل ۱۱-۸. توصیف سطح انتقال ثباتی

کنترل ترتیبی چارت ASM می‌تواند تا 2^n حالت را دارا باشد. چارتی با سه یا چهار حالت به یک مدار ترتیبی به دو فلیپ فلاپ نیاز دارد. با چهار تا هشت حالت، سه فلیپ فلاپ احتیاج است. هر ترکیبی از مقادیر فلیپ فلاپ‌ها یک عدد دودویی را برای یکی از حالات نشان می‌دهد.

یک جدول حالت برای یک کنترل‌گر، لیستی از حالات فعلی و مقادیر متناظرشان در حالت بعدی و خروجی‌هاست. در بسیاری از موارد، شرایط ورودی بی‌اهمیتی وجود دارند که باید لحاظ شوند، بنابراین پیشنهاد می‌شود تا جدول حالت تشکیل و این حالات در نظر گرفته شوند. ما مقادیر دودویی زیر را به سه حالت تخصیص می‌دهیم: $T_0 = 00$ ، $T_1 = 01$ ، $T_2 = 11$. حالت دودویی 10 به کار نرفته است و به عنوان حالت بی‌اهمیت تلقی می‌گردد. جدول حالت مربوط به نمودار حالت در جدول۳-۸ نشان داده شده است. دو فلیپ فلاپ مورد نیازند و با G_1 و G_0 نشانگذاری شده‌اند. سه ورودی و سه خروجی نیز وجود دارند. ورودی‌ها از شرایط درون جعبه‌های تصمیم اخذ می‌شوند. خروجی‌ها معادل با حالت فعلی کنترل‌اند. توجه کنید که در جدول سطری برای هر گذر ممکن بین حالات در نظر گرفته شده است. حالت اولیه 00 بسته به ورودی S به حالت 01 می‌رود یا در 00 می‌ماند. دو ورودی دیگر با علامت X مشخص شده‌اند و در تعیین حالت بعدی نقشی ندارند. ضمن قرار گرفتن سیستم در حالت دودویی 00، کنترل خروجی T_0 را تولید می‌کند تا عملیات ثباتی موردنظر آغاز گردد. گذر از حالت دودویی 01 به ورودی‌های A_3 و A_4 وابسته است. سیستم هنگامی به حالت دودویی 11 خواهد رفت که $A_3A_4 = 11$ باشد؛ در غیر این صورت در حالت دودویی 01 باقی خواهد ماند. بالاخره، حالت دودویی 11 مستقل از متغیرهای ورودی، در پالس ساعت بعدی به 00 خواهد رفت.

مدار کنترل

روال طراحی یک مدار ترتیبی در فصل ۵ با شروع از جدول حالت ملاحظه شد. اگر این روش به جدول ۳-۸ اعمال شود، لازم است از یک جدول پنج متغیره برای ساده کردن معادلات ورودی استفاده کنیم. دلیل این است که در زیر ستون حالت فعلی و ورودی، پنج متغیر وجود دارد. در عوض استفاده از نقشه برای ساده کردن معادلات ورودی، می‌توانیم آنها را با بررسی جدول حالت بدست آوریم. برای

جدول ۳-۸. جدول حالت برای کنترل شکل ۱۰-۸

سمبل حالت فعلی	حالت فعلی		ورودی‌ها			حالت بعدی		خروجی‌ها		
	G_1	G_0	S	A_3	A_4	G_1	G_0	T_0	T_1	T_2
T_0	0	0	0	X	X	0	0	1	0	0
T_0	0	0	1	X	X	0	1	1	0	0
T_1	0	1	X	0	X	0	1	0	1	0
T_1	0	1	X	1	0	0	1	0	1	0
T_1	0	1	X	1	1	1	1	0	1	0
T_2	1	1	X	X	X	0	0	0	0	1

طراحی مدار ترتیبی با فلیپ فلاپ‌های D، لازم است به ستون‌های حالت بعدی در جدول حالت رفته و همه شرایطی که توسط آنها هر فلیپ فلاپ به 1 می‌رود را مشخص نماییم. از جدول ۳-۸، توجه دارید که ستون حالت بعدی G_1 تنها یک 1 در پنجمین ستون دارد. ورودی D فلیپ فلاپ G_1 باید در حین حالت فعلی T_1 وقتی هر دو ورودی A_3 و A_4 برابر 1 هستند، برابر 1 گردد. این نکته با معادله ورودی فلیپ فلاپ D چنین نشان داده می‌شود.

$$D_{G1} = T_1 A_3 A_4$$

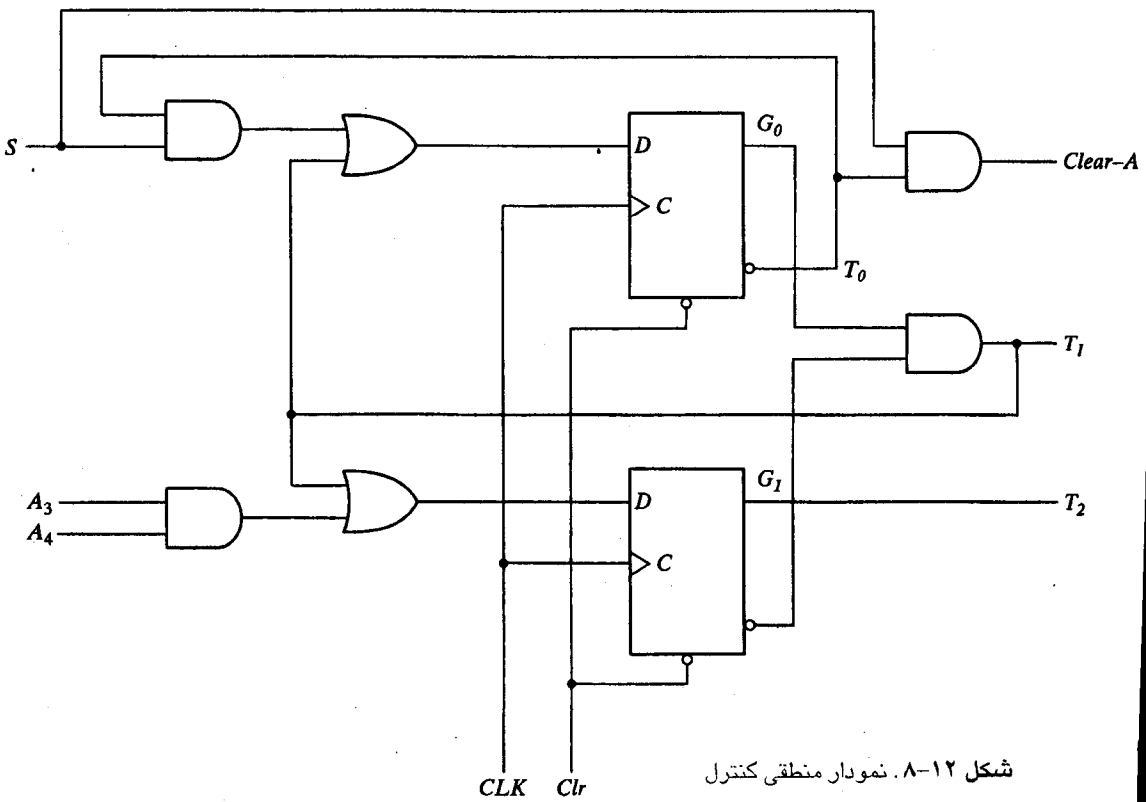
به طور مشابه ستون حالت بعدی G_0 دارای چهار 1 بوده و شرط 1 شدن آن چنین است.

$$D_{G0} = T_0 S + T_1$$

برای بدست آوردن سه تابع خروجی، از حالت بی‌استفاده 10 می‌توان استفاده کرد و معادلات خروجی زیر را بدست آورد.

$$T_0 = G_0'$$
$$T_1 = G_1' G_0$$
$$T_2 = G_1$$

نمودار منطقی کنترل در شکل ۱۲-۸ ملاحظه می‌گردد. این مدار ساختار درونی بلوک کنترل در شکل ۱۰-۸ را نشان می‌دهد که همراه با گیت AND که سیگنال clear A را تولید می‌کند، نشان می‌دهد.



شکل ۱۲-۸. نمودار منطقی کنترل

۸-۵ توصیف HDL مثال طراحی

در فصل‌های قبل، مثال توصیف HDL را برای مدارهای ترکیبی، ترتیبی و قطعات استانداردی چون مولتی پلکسرها و شمارنده‌ها و ثباتها نشان دادیم. اکنون در موقعیتی هستیم تا این اجزاء را در توصیف یک طرح خاص لحاظ کنیم. همانطور که قبلاً اشاره شد یک طرح می‌تواند در سطح ساختاری یا رفتاری باشد. توصیف‌های رفتاری را می‌توان بصورت سطح انتقال ثباتی یا سطح الگوریتمی خلاصه دسته‌بندی کرد. در نتیجه اکنون سه سطح طراحی را مطالعه می‌کنیم: توصیف ساختاری، توصیف RTL و توصیف رفتاری مبتنی بر الگوریتم.

توصیف ساختاری، پایین‌ترین و مشروح‌ترین سطح است. سیستم دیجیتال بر حسب اجزاء فیزیکی و اتصالات درونی‌شان مشخص می‌شوند. انواع اجزاء موجود عبارتند از گیت‌ها، فلیپ‌فلاپ‌ها و مدارهای استانداردی مثل مولتی‌پلکسرها و شمارنده‌ها. طراحی بطور سلسله مراتبی به واحدهای عملیاتی تفکیک می‌شود و هر واحد با مدول HDL توصیف می‌گردد. یک مدول سطح بالا کل سیستم را با ذکر همه مدول‌های سطح پایین‌تر ترکیب می‌کند.

توصیف RTL، سیستم دیجیتال را بر حسب ثبات‌ها، عملیات انجام شده، و کنترلی که ترتیب عملیات را کنترل می‌کند، مشخص می‌نماید. این نوع توصیف متشکل از عبارات اجرایی است که رابطه بین انواع عملیات طرح را بدون ارجاع به ساختار خاص معین می‌کند. توصیف RTL وجود آرایشی سخت افزاری را در میان ثبات‌ها تعیین می‌کند. این کار به طراح اجازه می‌دهد که طرحی را خلق کند که با عناصر دیجیتال استاندارد قابل ساخت است.

توصیف رفتاری مبتنی بر الگوریتم، خلاصه‌ترین سطح است. این روش کار طرح را به فرم یک الگوریتم اجرایی مثل زبان برنامه نویسی توصیف می‌نماید. در این روش هیچ اشاره‌ای به چگونگی پیاده شدن طرح با سخت افزار وجود ندارد. روش فوق الذکر، مناسبترین راه شبیه‌سازی سیستم‌های پیچیده برای تحقیق در مورد ایده‌های طرح است. توصیف‌های این سطح برای درک کاربرهای غیر فنی که زبان برنامه‌نویسی را بدانند در اختیار قرار می‌گیرد. بعضی نتایج ساخت در این سطح ممکن است قابل ترکیب و طراحی نباشد.

اکنون توصیف‌های RTL و ساختاری را با مثال طراحی بخش قبل پی‌می‌گیریم. یک توصیف مبتنی بر الگوریتم در بخش ۸-۸ تشریح شده است.

توصیف RTL

توصیف HDL یک طرح RTL را می‌توان به سه بخش تقسیم کرد. بخش اول ورودی‌ها، خروجی‌ها و ثبات‌ها را در طراحی تعریف می‌کند. دومین بخش رشته کنترل را مشخص می‌نماید. سومین بخش عملیات انتقال ثباتی و خروجی‌ها را فراهم می‌سازد. توصیف RTL مثال طراحی در مثال ۸-۲ HDL نشان داده شده است. به دنبال آن چارت ASM شکل ۸-۹ و ویژگی‌های انتقال ثباتی شکل ۸-۱۱ آمده است. ورودی‌ها عبارتند از S، CLK (ساعت، و Clr (پاک)). ورودی پاک کردن برای مقداردهی اولیه

```
//RTL description of design example (Fig.8-11)
module Example_RTL (S,CLK,Clr,E,F,A);
//Specify inputs and outputs
//See block diagram Fig. 8-10
   input S,CLK,Clr;
   output E,F;
   output [4:1] A;
//Specify system registers
   reg [4:1] A;                  //A register
   reg E, F;                     //E and F flip-flops
   reg [1:0] pstate, nstate;     //control register
//Encode the states
   parameter T0 = 2'b00, T1 = 2'b01, T2 = 2'b11;
//State transition for control logic
//See state diagram Fig. 8-11(a)
   always @(posedge CLK or negedge Clr)
      if (~Clr) pstate = T0;     //Initial state
      else pstate <= nstate;     //Clocked operations
   always @ (S or A or pstate)
       case (pstate)
         T0: if(S) nstate = T1; else nstate = T0;
         T1: if(A[3] & A[4]) nstate = T2; else nstate = T1;
         T2: nstate = T0;
       endcase
//Register transfer operations
//See list of operations Fig.8-11(b)
   always @(posedge CLK)
       case (pstate)
         T0: if(S)
               begin
                  A <= 4'b0000;
                  F <= 1'b0;
               end
         T1:
               begin
                  A <= A + 1'b1;
                  if (A[3]) E <= 1'b1;
                  else  E <= 1'b0;
               end
         T2:  F <= 1'b1;
      endcase
endmodule
```

حالت کنترل با T_0 مورد نیاز است. خروجی‌ها عبارتند از فلیپ فلاپ‌های E و F و ثبات A. ثبات کنترل با یک بردار 2 بیت که شرایط فعلی و بعدی را نگه می‌دارد مشخص شده است. به سه شرط کنترل نام سمبلیک داده شده و به مقادیر دودویی انکد شده است.

بخش بعدی توصیف RTL رشته کنترل را با دو عبارت always مشخص می‌نماید. اولین بلوک always دو عمل را مهیا می‌کند: ورودی Clr حالت فعلی را به T_0 مقداردهی اولیه می‌نماید، و لبه مثبت CLK (posedge) گذر حالت را با ساعت همگام یا همزمان می‌کند. دومین بلوک always شامل یک شرط case است که گذر از حالت فعلی به حالت بعدی را مشخص می‌کند. مثلاً اگر حالت فعلی T_0 و $S = 1$ باشد حالت بعدی T_1 خواهد بود. اگر $S = 0$ باشد، T_0 تغییر نخواهد کرد. تغییر از T_0 به T_1 (با $S = 1$) فقط در لبه مثبت CLK اجرا خواهد شد که با اولین عبارت always دیکته شده است.

سومین بخش توصیف RTL عمل انتقال ثباتی را در هر سه حالت کنترل معین می‌سازد. این بخش عملیات انتقال ثباتی لیست شده در شکل ۱۱-۸ را دنبال می‌نماید. توجه کنید که برای عملیات انتقال ثباتی از عبارات غیربلوکی (= >) استفاده شده است. این کاربری خصوصاً در حین حالت کنترل T_1 حیاتی است. در این حالت، A یک واحد افزایش می‌یابد و مقدار A[3] برای تعیین حالت E چک می‌شود. اگر از تخصیص بلوکی استفاده شود، باید دو عبارت یکی برای چک کردن E و سپس افزایش عبارت A به کار رود. با این وجود با استفاده از تخصیص‌های غیربلوکی، باید همزمانی لازم بدون توجه به رتبه عبارات لیست شده ایجاد شود.

تست توصیف طرح

رشته عملیات مثال طراحی در بخش قبل بررسی شد. جدول ۲-۸ مقادیر E و F را ضمن افزایش A نشان می‌دهد. در اینجا توصیه می‌شود برنامه تستی پیاده شود تا اعتبار توصیف HDL مشخص شود. برنامه تست (test bench) در مثال ۳-۸ HDL چنین مدولی را تهیه می‌نماید. (روال نوشتن یک برنامه تست در بخش ۱۱-۴ ملاحظه شد). مدول تست سیگنال‌هایی برای S، CLK، Clr تولید کرده و نتایج حاصل از ثبات‌های A، E و F را چک می‌کند. در آغاز سیگنال Clr در 0 قرار می‌گیرد تا کنترل آغاز گردد و نیز S و CLK برابر 0 می‌شوند. در $t = 5$، Clr با تنظیم آن در 1 غیرفعال می‌شود، با 1 کردن ورودی S، این ورودی فعال می‌شود، و ساعت برای 16 سیکل تکرار می‌گردد. عبارت $monitor مقادیر A، E و F را هر 10ns یک بار نشان می‌دهد. خروجی شبیه‌سازی که در مثال لیست شده، به صورت لگاریتم است. در آغاز، $t = 0$، مقدار ثبات‌ها نامعلومند بنابراین با سمبل x علامت‌زنی شده‌اند. اولین گذر لبه مثبت ساعت در $t = 0$، A و F را پاک می‌کند، ولی E را تغییر نمی‌دهد، پس E در این زمان نامعلوم است. بقیه جدول مثل جدول ۲-۸ است. توجه کنید که چون S در $t = 160$ هنوز برابر 1 است آخرین وارده در جدول نشان می‌دهد که A و F برابر 0 شده ولی E تغییری نکرده و در 1 باقی می‌ماند. این حالت در دومین گذر از T_0 به T_1 رخ می‌دهد.

```
//Test bench for design example
module test_design_example;
  reg S, CLK, Clr;
  wire [4:1] A;
  wire E, F;
//Instantiate design example
   Example_RTL dsexp (S,CLK,Clr,E,F,A);
    initial
      begin
           Clr = 0;
           S = 0;
           CLK = 0;
        #5 Clr = 1; S = 1;
           repeat (32)
           begin
            #5 CLK = ~ CLK;
           end
      end
    initial
     $monitor("A = %b E = %b F = %b time = %0d", A,E,F,$time);
endmodule

Simulation log:

A = xxxx E = x F = x time = 0
A = 0000 E = x F = 0 time = 10
A = 0001 E = 0 F = 0 time = 20
A = 0010 E = 0 F = 0 time = 30
A = 0011 E = 0 F = 0 time = 40
A = 0100 E = 0 F = 0 time = 50
A = 0101 E = 1 F = 0 time = 60
A = 0110 E = 1 F = 0 time = 70
A = 0111 E = 1 F = 0 time = 80
A = 1000 E = 1 F = 0 time = 90
A = 1001 E = 0 F = 0 time = 100
A = 1010 E = 0 F = 0 time = 110
A = 1011 E = 0 F = 0 time = 120
A = 1100 E = 0 F = 0 time = 130
A = 1101 E = 1 F = 0 time = 140
A = 1101 E = 1 F = 1 time = 150
A = 0000 E = 1 F = 0 time = 160
```

توصیف ساختاری

توصیف RTL یک طرح متشکل از عبارات اجرایی است که رفتار عملیاتی یک مدار دیجیتال را معین می‌کند. این نوع نمایش می‌تواند به وسیله ابزار سنتز (طراحی) HDL مورد استفاده قرار گیرد تا

مدار معادل سطح گیت طرح مشخص گردد. می‌توان طرح را برحسب ساختارش به جای عملکرد توصیف کرد. توصیف ساختاری طرح متشکل از ذکر اجزایی است که ساختار اتصالات میان آنها را تعریف می‌کند. از این لحاظ یک توصیف ساختاری با نمودار طرح یا نمودار بلوکی مدار برابر است.

نمودار بلوکی شکل ۱۰-۸ اطلاعات مورد نیاز توصیف ساختاری را تعریف می‌کند. برای سادگی مدار به سه بخش زیر تفکیک می‌شود.

۱- بلوک کنترل
۲- فلیپ فلاپ‌های E و F و گیت‌های مربوطه
۳- شمارنده با پاک کننده همزمان

توصیف سلسله مراتبی می‌تواند با ذکر تودرتو یا لانه‌ای سه بخش ایجاد شود.

مثال ۴-۸ HDL توصیف ساختاری مثال طراحی را نشان می‌دهد. این توصیف از شش مدول تشکیل شده است که با چهار بخش زیر قابل توصیف است.

۱- اولین مدول سه عنصر را ذکر می‌کند.
۲- دو مدول بعدی کنترل و فلیپ فلاپ D را توصیف می‌نمایند.
۳- دو مدول بعدی E و F و فلیپ فلاپ JK آنها را توصیف می‌کند.
۴- آخرین مدول شمارنده را توصیف می‌نماید.

اولین مدول ورودی‌ها و خروجی‌ها به مدار را اعلان می‌نماید. لیست پورت با لیست توصیف RTL یکی است. در اینجا خروجی‌ها به صورت reg اعلان نمی‌شوند، زیرا در مدول‌های سطح پایین‌تر با reg معرفی شده‌اند. مدول سطح بالا همه طرح را با ذکر سه جزء سطح پایین در برمی‌گیرد. بعضی از پورت‌ها در مدول‌های ذکر شده ورودی‌ها یا خروجی‌های مدارند. دیگر پورت‌ها ورودی‌های تولیدی از دیگر مدول‌ها یا خروجی‌های مورد لزوم دیگر مدول‌ها هستند. مثلاً مدول کنترل ct1 دارای ورودی‌های A[3] و A[4] است که از خروجی A[4 : 1] مدول counter ctr می‌آید. خروجی‌های T_1 و T_2 در مدول ct1 به عنوان ورودی در مدول EF استفاده شده است.

مدول کنترل مدار شکل ۱۲-۸ را توصیف می‌نماید. خروجی‌های دو فلیپ فلاپ G_1 و G_0 و ورودی‌های DG1 و DG0 نمی‌توانند به صورت reg اعلان شوند زیرا در عبارات تخصیص مداوم به کار رفته‌اند. پنج عبارت assign بخش ترکیبی مدار را مشخص می‌نمایند. دو معادله ورودی فلیپ فلاپ و سه معادله خروجی وجود دارد. خروجی‌های فلیپ فلاپ‌های G1 و G0 و معادلات ورودی DG1 و DG0 جایگزین خروجی Q و ورودی D دو فلیپ فلاپ‌های فوق‌الذکر شده‌اند. آنگاه فلیپ فلاپ D در مدول بعدی توصیف شده است. مدول EF الگوی یکسانی با دو فلیپ فلاپ JK را دنبال می‌کند. معادلات ورودی فلیپ فلاپ ابتدا بدست می‌آیند و فلیپ فلاپ JK با این مقادیر به عنوان ورودی‌ها ذکر می‌شود. آخرین مدول شمارنده را با پاک کننده همگام توصیف می‌نماید.

توصیف ساختاری با برنامه تست مثال ۳-۸ تست شد. تنها تغییر لازم جایگزینی ذکر مثال از

```
//Structural description of design example
//See block diagram Fig. 8-10
module Example_Structure (S,CLK,Clr,E,F,A);
   input S,CLK,Clr;
   output E,F;
   output [4:1] A;
//Instantiate control circuit
   control ctl (S,A[3],A[4],CLK,Clr,T2,T1,Clear);
//Instantiate E and F flip-flips
   E_F EF (T1,T2,Clear,CLK,A[3],E,F);
//Instantiate counter
   counter ctr (T1,Clear,CLK,A);
endmodule

//Control circuit (Fig. 8-12)
module control (Start,A3,A4,CLK,Clr,T2,T1,Clear);
   input Start,A3,A4,CLK,Clr;
   output T2,T1,Clear;
   wire G1,G0,DG1,DG0;
//Combinational circuit
   assign  DG1 = A3 & A4 & T1,
           DG0 = (Start & ~G0) | T1,
           T2  = G1,
           T1  = G0 & ~G1,
           Clear = Start & ~G0;
//Instantiate D flip-flop
   DFF G1F (G1,DG1,CLK,Clr),
       G0F (G0,DG0,CLK,Clr);
endmodule

//D flip-flop
module DFF (Q,D,CLK,Clr);
   input D,CLK,Clr;
   output Q;
   reg Q;
   always @ (posedge CLK or negedge Clr)
     if (~Clr) Q = 1'b0;
     else Q = D;
endmodule

//E and F flip-flops
module E_F (T1,T2,Clear,CLK,A3,E,F);
   input T1,T2,Clear,CLK,A3;
   output E,F;
   wire E,F,JE,KE,JF,KF;
```

```
//Combinational circuit
   assign JE = T1 & A3,
          KE = T1 & ~A3,
          JF = T2,
          KF = Clear;
//Instantiate JK flip-flop
   JKFF EF (E,JE,KE,CLK),
        FF (F,JF,KF,CLK);
endmodule

//JK flip-flop
module JKFF (Q,J,K,CLK);
   input J,K,CLK;
   output Q;
   reg Q;
   always @ (posedge CLK)
     case ({J,K})
       2'b00: Q = Q;
       2'b01: Q = 1'b0;
       2'b10: Q = 1'b1;
       2'b11: Q = ~Q;
     endcase
endmodule

//counter with synchronous clear
module counter (Count,Clear,CLK,A);
   input Count,Clear,CLK;
   output [4:1] A;
   reg [4:1] A;
   always @ (posedge CLK)
     if (Clear) A<= 4'b0000;
     else if (Count) A <= A + 1'b1;
     else A <= A;
endmodule
```

Example_RTL به Example_Structure است. نتیجه شبیه‌سازی برای توصیف ساختاری مثل شبیه‌سازی خروجی حاصل از توصیف RTL می‌باشد.

۶-۸ ضرب‌کننده دودویی

این بخش دومین طراحی را معرفی می‌کند. مدار یک الگوریتم سخت‌افزاری را برای ضرب کننده دودویی نمایش داده و آرایش ثبات را برای پیاده‌سازی آن پیشنهاد نموده و سپس برای نمایش طراحی مسیر داده و کنترل به کمک چارت ASM به پیش می‌رود.

سیستمی که معرفی خواهیم کرد دو عدد دودویی بی‌علامت را در هم ضرب می‌کند. در بخش ۶-۴،

یک الگوریتم سخت‌افزاری برای اجرای ضرب ایجاد شد که نتیجه‌اش یک مدار ترکیبی ضرب با چند جمع‌کننده و گیت‌های AND بود. برعکس در این بخش، الگوریتم سخت‌افزاری یک ضرب کننده ترتیبی که فقط از یک جمع کننده و یک شیفت رجیستر ساخته شده را بدست می‌دهد.

ضرب دو عدد دودویی با کاغذ و قلم با جمع و شیفت متوالی انجام می‌گردد. این فرآیند با یک مثال عددی بهتر تشریح می‌شود. اکنون ضرب دو عدد دودویی 10111 و 10011 را در نظر بگیرید:

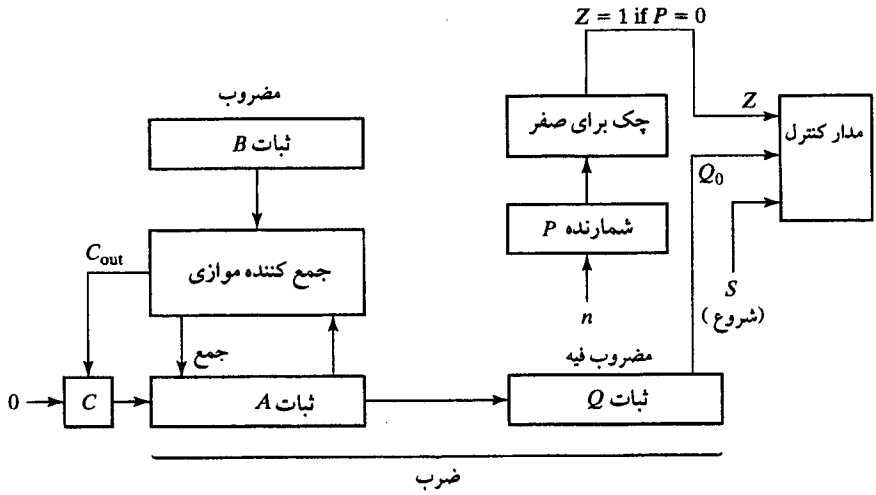
مضروب	10111	23
مضروب فیه	10011	19
	10111	
	10111	
	00000	
	00000	
	10111	
حاصلضرب	110110101	437

فرآیند شامل نگاه به بیت‌های متوالی مضروب‌فیه از سمت کم‌ارزش‌ترین بیت است. اگر بیت مضروب فیه 1 است، مضروب در پایین کپی می‌شود؛ در غیر این صورت 0 کپی می‌گردد. اعدادی که در خطوط متوالی کپی می‌شوند نسبت به قبل یک مکان به چپ جابجا می‌شوند. بالاخره اعداد با هم جمع شده و حاصل جمع آنها، حاصلضرب را تشکیل خواهد داد. حاصلضرب حاصل از ضرب دو عدد n بیتی می‌تواند تا 2n بیت فضا اشغال نماید.

وقتی که روال ضرب با سخت‌افزار دیجیتال پیاده شود، بهتر است تا فرآیند کمی تغییر کند. ابتدا در عوض تهیه مدار دیجیتال برای ذخیره و جمع همزمان به تعداد 1ها در مضروب فیه، بهتر است مداری فراهم گردد که فقط جمع دو عدد دودویی محاسبه شده و به طور متوالی حاصلضرب‌های جزیی حاصل در یک ثبات انبار شود. دوم، در عوض جابجایی مضروب به چپ حاصلضرب جزیی به راست جابجا می‌شود. این موجب می‌شود تا مضروب و حاصلضرب جزیی در کنار هم قرار گیرند. سوم، وقتی بیت مورد نظری در مضروب فیه 0 است، نیاز به جمع 0ها به حاصلضرب جزیی نیست زیرا مقدار حاصل را تغییر نمی‌دهد.

آرایش ثبات

نمودار بلوکی برای مضروب‌فیه دودویی در شکل ۱۳-۸ نشان داده شده است. مضروب در ثبات B ، مضروب فیه در ثبات Q و حاصلضرب جزیی در ثبات A ایجاد و در A و Q ذخیره می‌گردند. یک جمع کننده موازی محتوای ثبات B را به ثبات A اضافه می‌کند. فلیپ فلاپ C ، رقم نقلی را پس از جمع ذخیره می‌نماید. شمارنده P در آغاز با یک عدد که برابر تعداد 1ها در مضروب فیه است پر می‌شود. این شمارنده پس از هر بار تشکیل حاصلضرب جزیی یک واحد کم می‌شود. وقتی محتوای شمارنده به صفر برسد، حاصل ضرب کل در جفت ثبات A و B تشکیل شده است و بنابراین فرآیند متوقف می‌گردد.

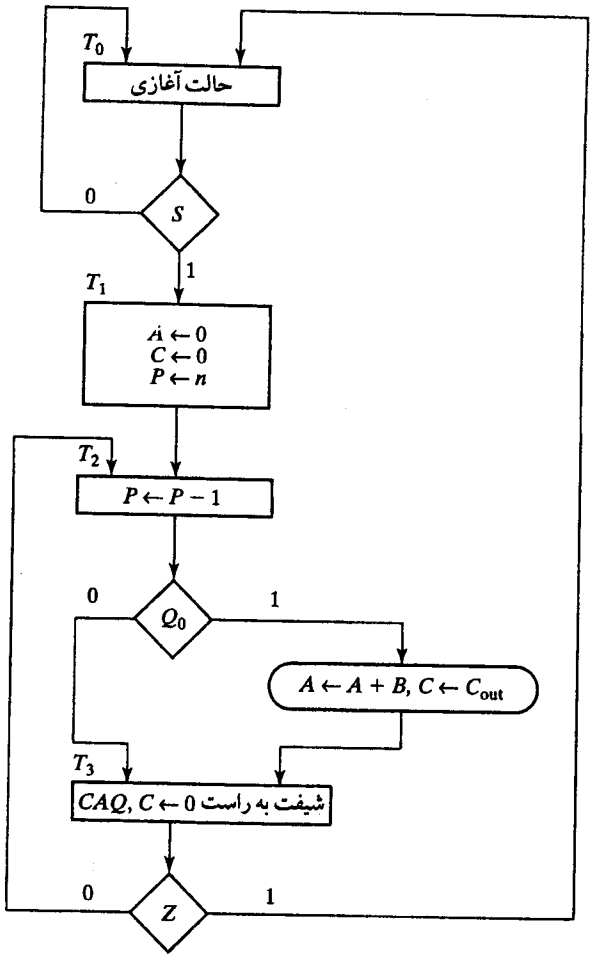


شکل ۱۳-۸. نمودار بلوکی ضرب کننده دودویی

مدار کنترل تا سیگنال شروع S برابر 1 شود در حالت اولیه باقی می‌ماند. آنگاه سیستم عمل ضرب را انجام می‌دهد. حاصل جمع A و B، n بیت باارزش‌تر حاصلضرب جزیی را تولید می‌کند که به A منتقل می‌گردد. نقلی خروجی از جمع، چه 0 و چه 1 به C منتقل می‌گردد. هر دو مقدار ضرب جزیی در A و مضروب فیه در Q به راست جابجا می‌شوند. بیت کم‌ارزش‌تر A به باارزش‌ترین بیت در Q منتقل می‌گردد؛ نقلی از C به باارزش‌ترین مکان در A می‌رود؛ و 0 به C وارد می‌شود. پس از عمل جابجایی به راست، یک بیت از حاصلضرب جزیی وارد Q می‌گردد ضمن این که بیت‌های مضروب فیه در Q یک مکان به راست می‌روند. به این ترتیب کم‌ارزش‌ترین بیت ثبات Q که به آن Q_0 می‌گوییم بیتی از مضروب فیه را نگه می‌دارد که در مرحله بعدی برای ادامه کار باید مورد وارسی قرار گیرد. مدار کنترل، جمع کردن یا نکردن را برحسب این بیت ورودی معین خواهد کرد. این مدار سیگنال Z را هم از مداری که شمارنده P را برای وجود صفر چک می‌کند دریافت می‌نماید. Q_0 و Z ورودی‌های وضعیت واحد کنترل هستند. ورودی شروع S یک ورودی بیرونی به کنترل است. خروجی‌های مدار کنترل عملیات لازم را در ثبات‌ها فعال می‌کنند.

چارت ASM

چارت ASM برای ضرب کننده دودویی در شکل ۱۴-۸ ملاحظه می‌شود. در آغاز مضروب در B و مضروب فیه در Q است. مادامی که مدار در حالت شروع و $S = 0$ است، هیچ اتفاقی رخ نمی‌دهد و سیستم در حالت اولیه T_0 باقی خواهد ماند. فرآیند ضرب با $S = 1$ شروع شده و کنترل به حالت T_1 می‌رود. ثبات A و فلیپ فلاپ C به 0 پاک می‌شوند و شمارنده دفعات، با عدد n مقداردهی می‌شود که n تعداد بیت‌ها در مضروب فیه است. آنگاه سیستم به T_2 می‌رود. بیت مضروب فیه در Q_0 چک می‌شود



شکل ۱۴-۸. چارت ASM برای ضرب دودویی

و اگر برابر 1 بود، مضروب در B به حاصلضرب جزیی در A اضافه می‌شود. رقم نقلی حاصل از جمع به C منتقل می‌گردد. حاصلضرب جزیی در A و C اگر $Q_0 = 0$ باشد، دست نخورده باقی می‌مانند. شمارنده P بدون توجه به مقدار Q_0 یک واحد کم می‌شود. در هر حال حالت بعدی T_3 است. ثبات‌های A، C و Q در یک ثبات مرکب CAQ تصور خواهند شد و محتوای آن یک بار به راست جابجا می‌شود تا حاصلضرب جزیی جدیدی حاصل گردد. این عمل شیفت در فلوچارت به فرم عبارت زیر نمایش شده است

$$\text{Shift right } CAQ, C \leftarrow 0$$

بااستفاده از سمبل‌های ثباتی جداگانه، عمل جابجایی با عملیات ثباتی زیر توصیف می‌گردد

$$A \leftarrow \text{shr } A, A_{n-1} \leftarrow C$$
$$Q \leftarrow \text{shr } Q, Q_{n-1} \leftarrow A_0$$
$$C \leftarrow 0$$

هر دو ثبات A و Q به راست جابجا شده‌اند. سمت چپ‌ترین بیت A که با A_{n-1} نشان داده می‌شود رقم نقلی C را دریافت می‌کند. سمت چپ‌ترین بیت Q، یا Q_{n-1} سمت راست‌ترین بیت A یعنی A_0 را دریافت می‌نماید و C به 0 بازنشانی می‌گردد. به بیان دیگر این یک جابجایی طویل در ثبات مرکب CAQ است با یک 0 که از ورودی سریال متصل به C وارد آن می‌شود.

مقدار موجود در شمارنده P پس از تشکیل هر حاصلضرب جزیی چک می‌شود. اگر محتوای P صفر نباشد بیت وضعیت Z برابر 0 بوده و فرآیند برای تشکیل یک حاصلضرب جزیی تکرار می‌گردد. فرآیند با رسیدن شمارنده P به 0 پایان یافته و در این حال ورودی کنترل برابر 1 خواهد بود. توجه کنید که حاصلضرب جزیی تشکیل شده در A به اندازه یک بیت در هر بار به Q منتقل می‌شود و نهایتاً جایگزین مضروب فیه خواهد شد. حاصلضرب نهایی در A و Q خواهد بود که A بیت‌های باارزش‌تر و Q بیت‌های کم‌ارزش‌تر حاصل ضرب است.

مثال عددی قبلی در جدول ۴-۸ برای شفافیت روند ضرب تکرار شده است. روال مراحل ذکر شده در چارت ASM را دنبال می‌کند.

نوع ثبات‌های لازم برای زیر سیستم پردازش داده از عملیات ثباتی لیست شده در چارت ASM بدست می‌آید. ثبات A یک شیفت رجیستر با امکان بار شدن موازی برای پذیرش جمع از جمع‌کننده است و باید قابلیت پاک شدن همزمان را داشته باشد تا ثبات به 0 بازنشانی شود. ثبات Q هم یک شیفت رجیستر است. شمارنده P یک پایین شمار دودویی با امکان بار شدن موازی یک ثابت دودویی است. فلیپ‌فلاپ

جدول ۴-۸. مثال عددی ضرب دودویی

	C	A	Q	P
مضروب B = 10111				
مضروب فیه در Q	0	00000	10011	101
$Q_0 = 1$: با B جمع کن		10111		
ضرب جزئی اول	0	10111		100
جابجایی به راست CAQ	0	01011	11001	
$Q_0 = 1$: با B جمع کن		10111		
ضرب جزئی دوم	1	00010		011
جابجایی به راست CAQ	0	10001	01100	
$Q_0 = 0$ جابجایی به راست CAQ	0	01000	10110	010
$Q_0 = 0$ جابجایی به راست CAQ	0	00100	01011	001
$Q_0 = 1$: با B جمع کن		10111		
پنجمین ضرب جزئی	0	11011		
جابجایی به راست CAQ	0	01101	10101	000
حاصلضرب نهایی در AQ = 0110110101				

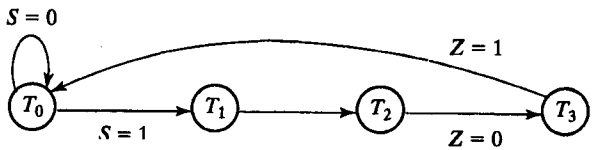
C باید طوری طراحی شود تا نقلی ورودی و پاک شدن همزمان را بپذیرد. ثبات‌های B و Q دارای قابلیت بار شدن موازی هستند تا مضروب و مضروب فیه را قبل از شروع فرآیند ضرب، دریافت نمایند.

۸-۷ مدارکنترل

طراحی یک سیستم دیجیتال را می‌توان به دو بخش تفکیک کرد: طراحی انتقال ثباتی در مسیر داده و طراحی مدارکنترل. طراحی مدارکنترل یک طراحی مدار ترتیبی است. در این صورت بهتر است نمودار کنترل ترتیبی را بدست آوریم. یک چارت ASM مشابه یک نمودار حالت است. بلوک‌های مستطیلی که جعبه‌های حالت را تداعی می‌کنند حالات مدار ترتیبی هستند. بلوک‌های لوزی شکلی که جعبه تصمیم‌گیری را تداعی می‌نمایند، شرایط را برای گذر به حالت بعدی در نمودار حالت مشخص می‌کنند. به عنوان مثال، نمودار حالت کنترل برای ضرب کننده دودویی که در بخش قبل بدست آمد. در شکل ۸-۱۵ نشان داده شده است. اطلاعات نمودار مستقیماً از چارت ASM شکل ۸-۱۴ حاصل شده است. چهار حالت T_0 تا T_3 از جعبه‌های حالت مستطیلی بدست آمده‌اند. ورودی‌های S و Z از جعبه‌های تصمیم لوزی اخذ شده‌اند. عملیات انتقال ثباتی برای هر چهار حالت در زیر نمودار حالت آمده است. آنها از حالت متناظر و جعبه‌های شرطی در چارت ASM فراهم شده‌اند.

به هنگام پیاده‌سازی یک مدارکنترل دو دیدگاه مورد بحث وجود دارد. یکی ایجاد رشته حالات موردنظر و تهیه سیگنال‌های کنترل عملیات ثباتی است. رشته حالات در نمودار حالت مشخص می‌گردد. سیگنال‌های کنترل عملیات در ثبات‌ها در عبارات انتقال ثبات مشخص می‌شوند. در ضرب کننده، این سیگنال‌ها T_1 (برای پاک کردن A و C و بارکردن یک عدد در P)، T_2 (برای کاهش P)، T_3 (برای شیفت رجیستر CAQ)، و Q_0 جهت افزودن B به A و یا نیفزودن آن می‌باشد. نمودار بلوکی کنترل در شکل ۸-۱۶ دیده می‌شود. ورودی‌های مدار کنترل ترتیبی S و Z هستند و طبق نمودار حالت خروجی‌ها نیز T_0 ، T_1 ، T_2 و T_3 می‌باشند. گیت AND مولد $L = T_2Q_0$ برای بارکردن حاصل جمع در ثبات A به شرط $Q_0 = 1$ در حالت T_2 است.

یکی از مراحل با اهمیت در طراحی، تخصیص کد دودویی به حالات است. ساده‌ترین تخصیص،



T_0: Initial state

T_1: $A \leftarrow 0, C \leftarrow 0, P \leftarrow n$

T_2: $P \leftarrow P - 1$

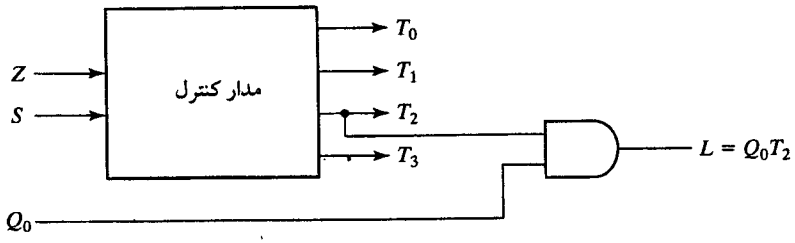
if $(Q_0) = 1$ then $(A \leftarrow A + B, C \leftarrow C_{out})$

T_3: shift right CAQ, $C \leftarrow 0$

(ب) عملیات انتقال ثباتی

(الف) نمودار حالت

شکل ۸-۱۵. مشخصات کنترل برای ضرب دودویی



شکل ۱۶-۸. نمودار بلوکی کنترل

اختصاص مستقیم اعداد دودویی، طبق جدول ۵-۸ می‌باشد. تخصیص مشابه دیگر، کدگری است که در آن هنگام رفتن از یک عدد به بعدی، فقط یک بیت تغییر می‌نماید. تخصیص حالتی که اغلب در طراحی کنترل به کار می‌رود تخصیص 1 _ بارز است. این تخصیص به تعداد حالات مدار از بیت استفاده می‌کند. در هر لحظه از زمان، تنها یک بیت برابر 1 است (1 بارز) در حالی که دیگر بیت‌ها در 0 نگهداشته می‌شوند. این نوع تخصیص برای هر حالت از یک فلیپ فلاپ استفاده می‌کند.

چون کنترل یک مدار ترتیبی است، می‌توان طبق مطالب فصل ۵، آن را با روش مدار ترتیبی طراحی کرد. با این وجود در بسیاری از موارد این روش عملاً پیچیده است زیرا این گونه مدارهای کنترل دارای تعداد زیادی حالات و ورودی هستند. در نتیجه، لازم است از روش‌های خاصی برای طراحی کنترل استفاده شود. اکنون دو نمونه از این گونه مدارها را ارائه می‌کنیم. یکی از آنها از یک ثبات و دیکدر و دیگری از فلیپ فلاپ برای هر حالت استفاده می‌کند.

ثبات و دیکدر

همانطور که از نام این روش پیداست، روش ثبات و دیکدر از یک ثبات برای کنترل حالات و یک دیکدر برای تهیه یک خروجی متعلق به هر حالت استفاده می‌کند. ثباتی با n فلیپ فلاپ، می‌تواند 2^n حالت داشته و دیکدر n به 2^n نیز دارای 2^n خروجی است. یک ثبات اساساً مداری با n فلیپ فلاپ همراه با گیت‌های متعلقه برای تأثیر بر گذر حالت است.

نمودار حالت کنترل برای ضرب دودویی دارای چهار حالت و دو ورودی است. برای پیاده‌سازی آن با یک ثبات و دیکدر، لازم است دو فلیپ فلاپ برای ثبات و یک دیکدر 2 به 4 انتخاب کنیم. گرچه مثال

جدول ۵-۸. تخصیص حالت برای کنترل

حالت	دودویی	کدگری	یک بارز
T_0	00	00	0001
T_1	01	01	0010
T_2	10	11	0100
T_3	11	10	1000

جدول ۶-۸. جدول حالت برای مدار کنترل

حالت فعلی		ورودی		حالت بعدی		خروجی‌ها			
G_1	G_0	S	Z	G_1	G_0	T_0	T_1	T_2	T_3
0	0	0	X	0	0	1	0	0	0
0	0	1	X	0	1	1	0	0	0
0	1	X	X	1	0	0	1	0	0
1	0	X	X	1	1	0	0	1	0
1	1	X	0	1	0	0	0	0	1
1	1	X	1	0	0	0	0	0	1

تقریباً ساده است، ولی روال را می‌توان به مدارهای پیچیده‌تر هم تعمیم داد.

جدول حالت کنترل ترتیبی در جدول ۶-۸ نشان داده شده است. این جدول مستقیماً از نمودار شکل ۱۵-۸ (الف) حاصل شده است. در اینجا دو فلیپ فلاپ را G_1 و G_0 می‌نامیم و حالات 00، 01، 10 و 11 را به ترتیب T_0، T_1، T_2، T_3 می‌خوانیم. توجه کنید که ستون ورودی‌ها دارای وارده‌های بی‌اهمیت‌اند. این حالات بی‌اهمیت متعلق به مواردی است که ورودی برای تعیین حالت بعدی به‌کار نرود. خروجی‌های مدار کنترل با نام حالات معین شده‌اند. حالت 1 برای هر متغیر خروجی خاص در هر زمان از مقدار حالت فعلی‌اش حاصل می‌گردد. بنابراین وقتی حالت فعلی $G_1G_0 = 00$ است، خروجی T_0 باید 1 و دیگر خروجی‌ها باید 0 باشند. چون خروجی‌ها فقط تابعی از حالت فعلی هستند، می‌توان آنها را با مدار دیکدری که دارای دو ورودی G_1 و G_0 و چهار خروجی T_0، T_1، T_2 و T_3 می‌باشند، تولید کرد.

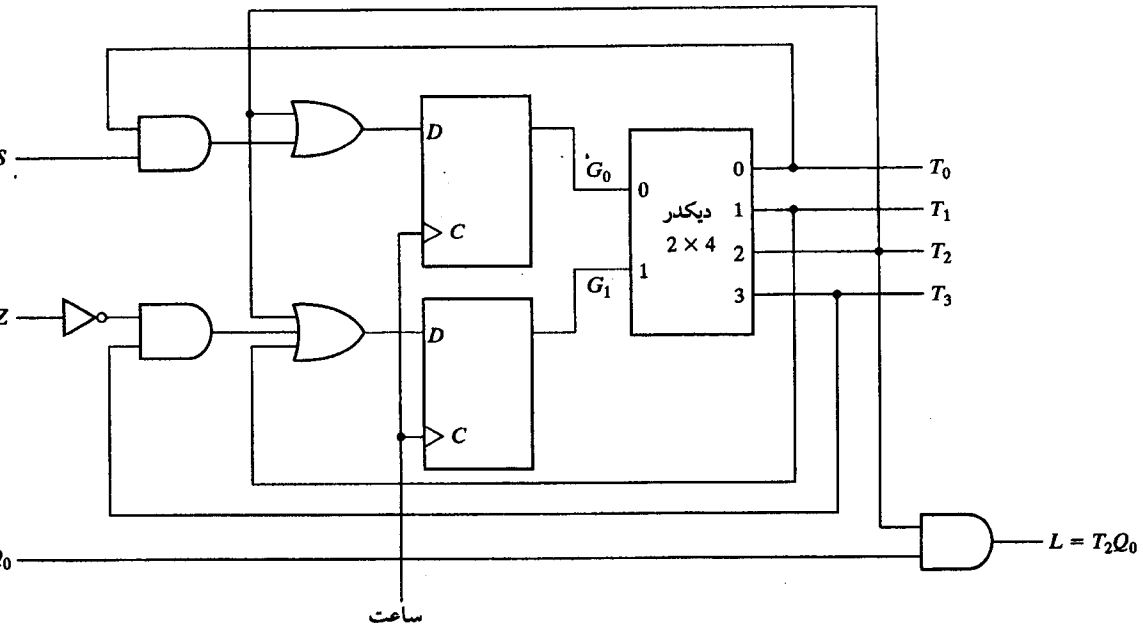
با استفاده از روش کلاسیک فصل ۵ می‌توان مدار ترتیبی را از جدول حالت طراحی کرد. این مثال دارای تعداد کمی حالت و ورودی است و بنابراین می‌توان از نقشه کارنو برای ساده کردن تابع بول استفاده نمود. در بسیاری از کاربردهای مدار کنترل، تعداد ورودی‌ها و حالات بسیار زیادتر است. در تهیه معادلات ورودی ساده شده برای فلیپ فلاپ‌ها، به کارگیری روش کلاسیک کار زیادی را می‌طلبد. چنانچه فرض کنیم خروجی‌های دیکدر برای استفاده در طراحی در دسترسند طراحی بسیار ساده‌تر خواهد شد. در عوض به کارگیری نقشه برای ساده کردن معادلات فلیپ فلاپ‌ها، می‌توانیم آنها را مستقیماً با وارسی جدول حالت بدست آوریم. مثلاً، با توجه به شرایط حالت بعدی در جدول حالت، در می‌یابیم که حالت بعدی G_1 به شرطی 1 می‌باشد که حالت فعلی T_1 یا T_2 و یا T_3 همراه با $Z=0$ باشد. این شرایط را می‌توان با معادله زیر بیان کرد.

$$D_{G1} = T_1 + T_2 + T_3Z'$$

که در آن D_{G1} ورودی D فلیپ فلاپ G_1 است. به طور مشابه ورودی D مربوط به G_0 برابر است با

$$D_{G0} = T_0S + T_2$$

البته با روش بررسی جدول حالت نمی‌توان مطمئن شد که توابع بول به بهترین وجه ساده شده‌اند. به این دلیل، توصیه می‌شود برای اطمینان از صحت معادلات، تحلیل‌های لازم صورت گیرد، تا بدین وسیله گذر حالات لازم تولید شود.



شکل ۸-۱۷ . نمودار منطقی کنترل برای ضرب دودویی با استفاده از یک ثبات و دیکدر

نمودار منطقی مدار کنترل در شکل ۸-۱۷ ترسیم شده است. این مدار متشکل از یک ثبات با دو فلیپ فلاپ G_1 و G_0 و یک دیکدر 4 × 2 است. خروجی‌های دیکدر برای تولید ورودی‌ها به مدار حالت بعدی و خروجی‌های کنترل به کار رفته‌اند. خروجی‌های کنترل‌گر باید برای فعال کردن عملیات ثباتی لازم به مسیر داده (بخش پردازش) متصل گردند.

یک فلیپ فلاپ برای هر حالت

روش ممکن دیگر در طراحی مدار کنترل استفاده از تخصیص 1 بارز است که یک مدار ترتیبی با یک فلیپ فلاپ را نتیجه می‌دهد. هر بار تنها یکی از فلیپ فلاپ‌ها در 1 و بقیه در وضعیت 0 هستند. انتشار یک 1 از یک فلیپ فلاپ به دیگری براساس کنترل مدار تصمیم‌گیری صورت می‌گیرد. در این آرایش هر فلیپ فلاپ یک حالت را نمایش می‌دهد و فقط هنگامی که بیت کنترل به آن منتقل شود وجود دارد.

این روش بیشترین تعداد فلیپ فلاپ را بکار می‌برد. مثلاً یک مدار ترتیبی با 12 حالت به حداقل چهار فلیپ فلاپ نیاز دارد. با روش یک فلیپ فلاپ برای هر حالت، مدار نیاز به 12 فلیپ فلاپ خواهد داشت. در نگاه اول چنین به نظر می‌رسد که به علت وجود تعداد زیادی فلیپ فلاپ، هزینه این سیستم افزایش خواهد یافت. ولی مدار مزایایی را داراست که نمی‌توان آنها را نادیده گرفت. یکی از این مزایا این است که مدار را با وارسی چارت ASM یا نمودار حالت می‌توان طراحی کرد. اگر فلیپ فلاپ‌های نوع D به کار گرفته شود نیازی به جداول حالت و تحریک نیست. دیگر مزایای قابل ذکر، صرفه‌جویی در

تلاش‌هایی است که در طراحی صورت می‌گیرد، افزایش سادگی اعمال، و احتمالاً کاهش در تعداد کل گیت‌ها به دلیل عدم نیاز به دیکدر است.

روال طراحی را می‌توان با مدار کنترل نمودار حالت شکل ۱۵-۸ (الف) نشان داد. چون در نمودار حالت چهار حالت وجود دارد، چهار فلیپ فلاپ D اختیار کرده و خروجی آنها را T_0 ، T_1 ، T_2 و T_3 می‌نامیم. معادلات ورودی برای 1کردن هر فلیپ فلاپ از حالت فعلی و شرایط ورودی در روی خطوط جهت‌داری که به حالت بعدی می‌رود مشخص می‌گردد. مثلاً فلیپ فلاپ T_0 هنگامی با لبه پالس ساعت بعدی 1 می‌شود که مدار در حالت فعلی T_3 و ورودی Z برابر 1 یا اگر مدار در حالت فعلی T_0 است $S = 0$ باشد. این شرایط با معادله ورودی زیر نشان داده می‌شود.

$$D_{T0} = T_0S' + T_3Z$$

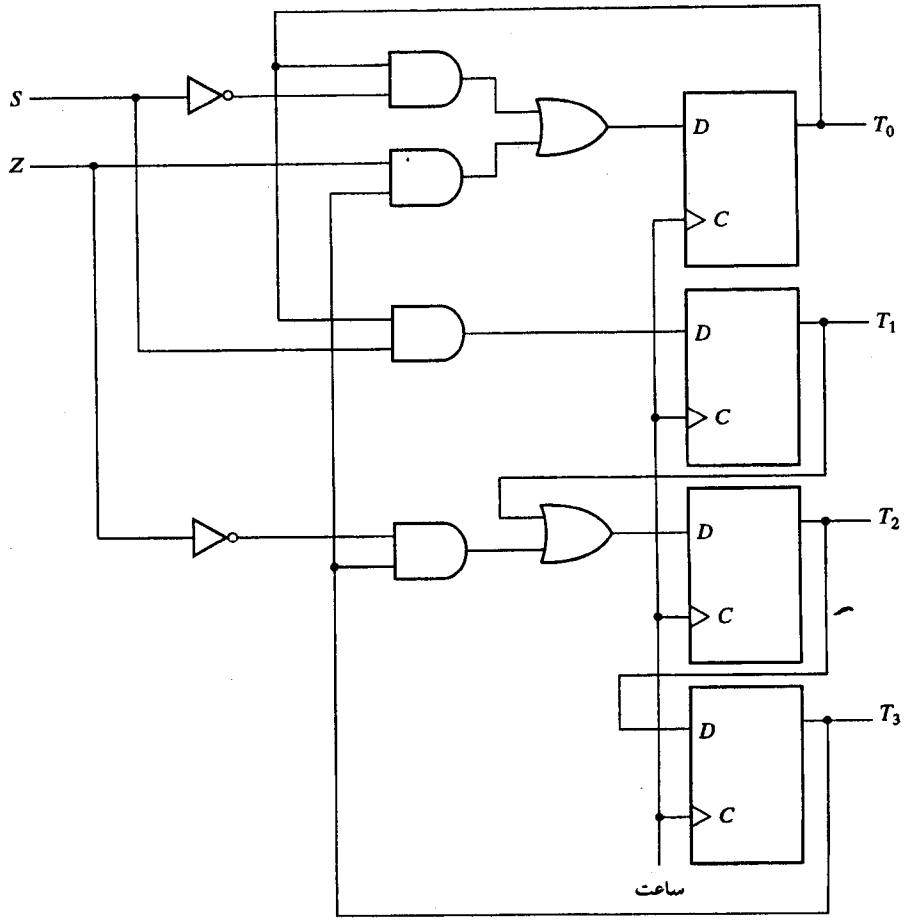
که در آن D_{T0} ورودی D فلیپ فلاپ T_0 است. در واقع، شرط 1کردن یک فلیپ فلاپ مستقیماً از نمودار حالت و از خطوط جهت‌دار متصل به فلیپ فلاپ مربوطه که با حالت بعدی فلیپ فلاپ AND شده بدست می‌آید. اگر تعداد خطوطی که به یک حالت می‌روند بیش از یکی باشد، همه حالات باید OR شوند. با استفاده از این روال برای سه فلیپ فلاپ دیگر معادلات ورودی چنین خواهند بود.

$$D_{T1} = T_0S$$
$$D_{T2} = T_1 + T_3Z'$$
$$D_{T3} = T_2$$

نمودار منطقی کنترل‌گر در شکل ۱۸-۸ دیده می‌شود. این شکل دارای چهار فلیپ فلاپ D از T_0 تا T_3 و گیت‌های مربوطه است که با معادلات ورودی نشان داده شده‌اند. در ابتدا T_0 باید در 1 نشانده شود. همه فلیپ فلاپ‌های دیگر باید 0 گردند، به این ترتیب فلیپ فلاپی که حالت آغازین را نشان می‌دهد فعال می‌گردد. این کار را می‌توان با به کارگیری یک پیش تنظیم غیرهمزمان روی فلیپ فلاپ T_0 و یک پاک کردن غیرهمزمان برای دیگر فلیپ فلاپ‌ها تحقق بخشید. به محض این که مدار شروع به کار کند، کنترل‌گر یک فلیپ فلاپ بر هر حالت شروع به انتشار حالت موجود خواهد کرد. در هر لبه پالس ساعت تنها یک فلیپ فلاپ در 1 و بقیه در 0 خواهند بود، زیرا ورودی آنها 0 است.

۸-۸ توصیف HDL ضرب‌کننده دودویی

دومین مثال برای توصیف HDL یک طرح RTL، در مثال ۵-۸ نشان داده شده است. مثال مربوط به طراحی ضرب کننده دودویی بخش ۶-۸ می‌باشد. می‌توان توصیف را به پنج بخش که هر بخش با توضیحاتی همراه است تقسیم کرد. بخش اول همه ورودی‌ها و خروجی‌ها را طبق شکل ۱۳-۸ لیست می‌کند. ورودی‌های داده به B و Q و خروجی‌های حاصل از A و Q بردارهای 5 بیتی هستند. دلیل انتخاب 5 بیت برای این مثال این است که بتوانیم نتایج را با مثال عددی جدول ۴-۸ مقایسه کنیم. بخش دوم همه ثبات‌ها از جمله ثبات کنترل، همراه با انکد کردن چهار حالت را اعلان می‌کند. بخش سوم مداری ترکیبی را با عبارت **assign** مشخص می‌نماید. در ضرب کننده دو مدار ترکیبی وجود دارد. یکی



شکل ۱۸-۸. نمودار منطقی کنترل با استفاده از یک فلیپ فلاپ در هر مرحله

از آنها خروجی شمارنده را برای یافتن صفر خروجی وارسی می‌کند؛ و دیگری یک جمع‌کننده موازی است. مبدأ جمع کننده به عنوان بخشی از عملگر انتقال ثباتی که مضروب را با حاصلضرب جزیی جمع می‌کند در نظر گرفته خواهد شد. شناسایی صفر با یک گیت NOR پنج ورودی انجام می‌گردد.

بخش چهارم گذر حالت را برای کنترل توصیف کرده و نمودار حالت شکل ۱۵-۸ (الف) را دنبال می‌نماید. آخرین بخش توصیف عملیات انتقال ثباتی برای بخش مسیر داده است. به دنبال آن اعمال لیست شده در چارت ASM شکل ۱۵-۸ (ب) آمده است. در چارت ASM ورودی‌های داده لیست نشده‌اند، ولی در اینجا برای چک کردن عمل مدار باید لحاظ شوند. در حین حالت آغازین، مضروب به ثبات B انتقال می‌یابد. مضروب فیه در زمان T_1 منتقل می‌شود. در T_2، مضروب به حاصلضرب جزیی اضافه می‌شود به شرطی که Q[0] = 1 باشد. در T_3، حاصلضرب جزیی به راست جابجا می‌شود.

```
//RTL description of binary multiplier
//Block diagram Fig. 8-13 and ASM chart Fig. 8-14
//n = 5 to compare with Table 8-4
module mltp(S,CLK,Clr,Binput,Qinput,C,A,Q,P);
   input S,CLK,Clr;
   input [4:0] Binput,Qinput;          //Data inputs
   output C;
   output [4:0] A,Q;
   output [2:0] P;
//System registers
   reg C;
   reg [4:0] A,Q,B;
   reg [2:0] P;
   reg [1:0] pstate, nstate;           //control register
   parameter T0=2'b00, T1=2'b01, T2=2'b10, T3=2'b11;
//Combinational circuit
   wire Z;
   assign Z = ~|P;                     //Check for zero
//State transition for control
//See state diagram Fig. 8-15(a)
   always @(negedge CLK or negedge Clr)
     if (~Clr) pstate = T0;
     else pstate <= nstate;
   always @(S or Z or pstate)
     case (pstate)
       T0: if (S) nstate = T1; else nstate = T0;
       T1: nstate = T2;
       T2: nstate = T3;
       T3: if (Z) nstate = T0;
           else   nstate = T2;
     endcase
//Register transfer operations
//See register operation Fig. 8-15(b)
   always @(negedge CLK)
     case (pstate)
       T0: B <= Binput;                //Input multiplicand
       T1: begin
             A <= 5'b00000;
             C <= 1'b0;
             P <= 3'b101;             //Initialize counter to n=5
             Q <= Qinput;              //Input multiplier
           end
       T2: begin
             P <= P - 3'b001;          //Decrement counter
             if (Q[0])
             {C,A} <= A + B;           //Add multiplicand
           end
       T3: begin
             C <= 1'b0;                //Clear C
             A <= {C,A[4:1]};          //Shift right A
             Q <= {A[0],Q[4:1]};       //Shift right Q
           end
     endcase
endmodule
```

مثال ۶-۸ HDL یک برنامه تست را برای تست ضرب کننده نشان می‌دهد. ورودی‌ها عبارتند از: مضروب و مضروب فیه در B و Q . خروجی حاصلضرب در A و Q قرار دارد. ما مقدار نقلی C و شمارنده P را هم چک می‌کنیم. ورودی‌های دودویی 5 بیتی همان ورودی‌های به کار رفته در جدول ۴-۸ هستند. دومین عبارت آغازین 13 پالس ساعت را 10 واحد زمانی برای هر کدام فراهم می‌نماید. بررسی گذر حالات نشان می‌دهدکه سیکل‌های T_2 و T_3 ، در حلقه 5 بار تکرار می‌شوند (برای هر حاصلضرب جزیی یک بار). این خود 10 پالس ساعت را لازم دارد. به هنگام Z = 1 ، سه سیکل ساعت دیگر هم برای T_0 ، T_1 و بازگشت به T_0 لازم است.

نتایج محاسبات با استفاده از تکلیف سیستم $strobe نمایش داده شده است. این تکلیف مشابه تکلیف‌های $display و $monitor تشریح شده در بخش ۱۱-۴ می‌باشد. تکلیف $strobe یک مکانیزم همزمانی را برای اطمینان از نمایش داده پس از اجرای همه عبارات در یک مرحله از زمان فراهم می‌کند. این مکانیزم در مدارهای ترتیبی همزمانی که در آنها هر مرحله زمانی با لبه ساعت آغاز می‌شود اهمیت دارد. استفاده از $strobe پس از عبارت always @ negedge CLK به این معنی است که صفحه نمایش مقادیر سیگنال را پس از لبه منفی ساعت نشان خواهد داد.

مدول test_mltp در مثال ۶-۸ HDL مدول mltp در مثال ۵-۸ را ذکر می‌نماید. هنگام شبیه‌سازی با Verilog HDL هر دو مدول باید لحاظ شوند. نتیجه این شبیه‌سازی اعدادی مشابه با جدول ۴-۸ را نمایش خواهد داد.

مثال ۶-۸ HDL

```
//Testing binary multiplier
module test_mltp;
//Inputs for multiplier
   reg S,CLK,Clr;
   reg [4:0] Binput,Qinput;
//Data for display
   wire C;
   wire [4:0] A,Q;
   wire [2:0] P;
//Instantiate multiplier
   mltp mp(S,CLK,Clr,Binput,Qinput,C,A,Q,P);
   initial
     begin
          S=0; CLK=0; Clr=0;
      #5  S=1; Clr=1;
          Binput = 5'b10111;
          Qinput = 5'b10011;
      #15 S = 0;
     end
```

```
   initial
     begin
         repeat (26)
      #5 CLK = ~CLK;
     end
//Display computations and compare with Table 8-4
   always @(negedge CLK)
     $strobe ("C=%b  A=%b  Q=%b  P=%b  time=%0d",C,A,Q,P,$time);
endmodule
```

```
Simulation log:

C=x  A=xxxxx  Q=xxxxx  P=xxx  time=10
C=0  A=00000  Q=10011  P=101  time=20
C=0  A=10111  Q=10011  P=100  time=30
C=0  A=01011  Q=11001  P=100  time=40
C=1  A=00010  Q=11001  P=011  time=50
C=0  A=10001  Q=01100  P=011  time=60
C=0  A=10001  Q=01100  P=010  time=70
C=0  A=01000  Q=10110  P=010  time=80
C=0  A=01000  Q=10110  P=001  time=90
C=0  A=00100  Q=01011  P=001  time=100
C=0  A=11011  Q=01011  P=000  time=110
C=0  A=01101  Q=10101  P=000  time=120
C=0  A=01101  Q=10101  P=000  time=130
```

توصیف رفتاری ضرب کننده

می‌توان یک ضرب کننده دودویی را در سطح رفتاری مانند مثال ۸-۷ HDL هم توصیف کرد. دو ورودی به عنوان بردارهای 8 بیت و خروجی به عنوان بردار 16 بیت اعلان شده‌اند. در اینجا سخت‌افزار خاصی مانند آنچه در توصیف‌های RTL یا ساختاری ملاحظه شد، مورد نیاز نیست. اجرا فقط با یک عبارت ضرب صورت می‌گیرد. در این حال، اگر نرم‌افزار سنتز (طراحی) طراحی شده باشد، توصیف ممکن است برای یک مدار ترکیبی ضرب ایجاد گردد. به طور کلی، توصیف‌های مبتنی بر الگوریتم همیشه قابل سنتز یا ترکیب نیستند.

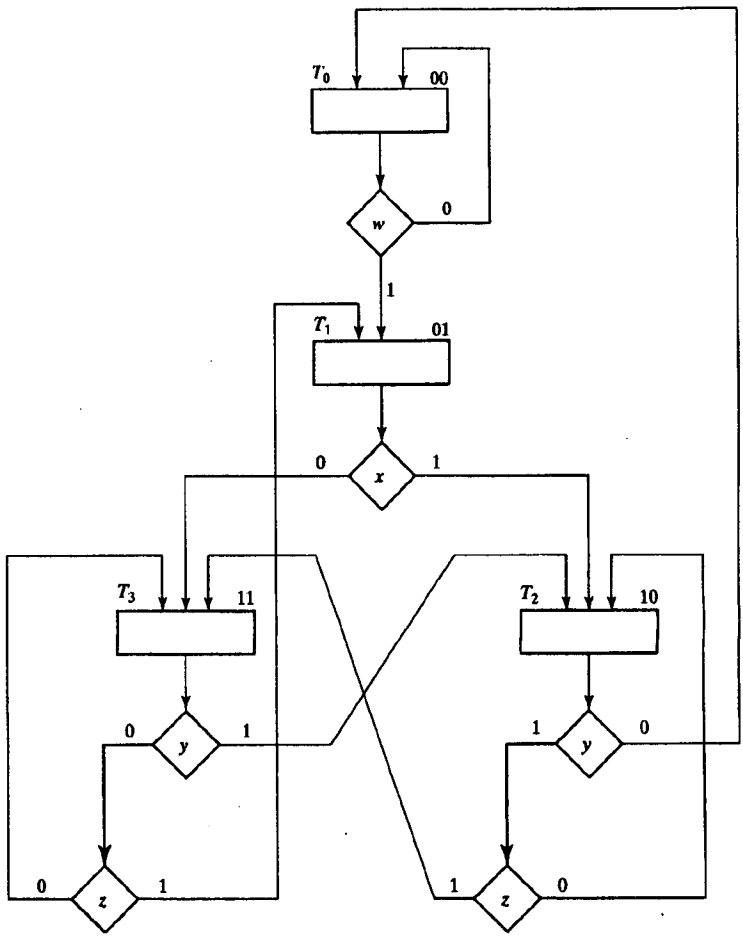
مثال ۸-۷ HDL

```
//Behavioral description of multiplier (n = 8)
module Mult (A,B,Q);
   input [7:0] B,Q;
   output [15:0] A;
   reg [15:0] A;
   always @ (B or Q)
     A = B * Q;
endmodule
```

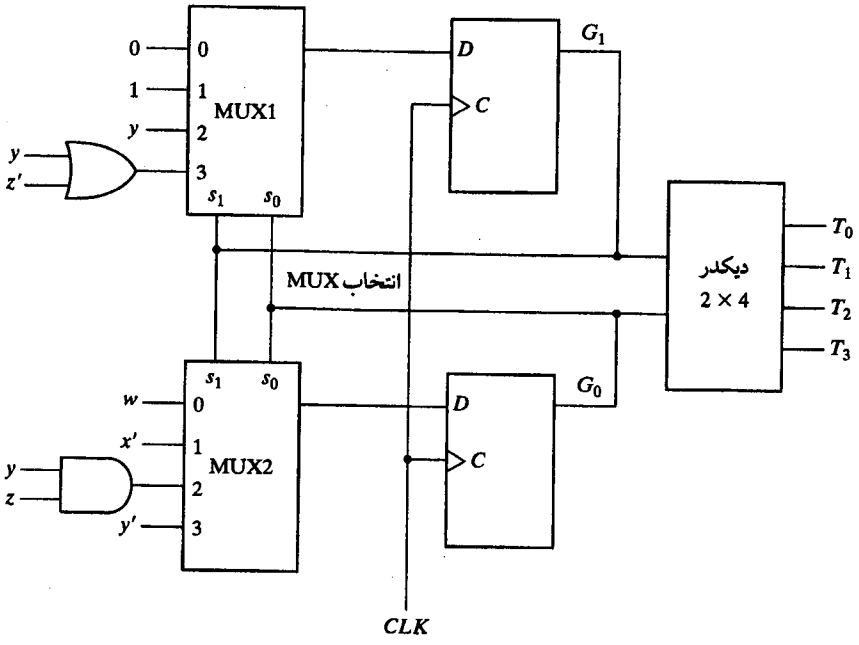
۸-۹ طراحی با مولتی پلکسر

کنترل ثبات و دیکدر از سه عنصر تشکیل شده بود: فلیپ فلاپ‌ها که مقدار حالت دودویی را نگه می‌دارند، دیکدر که خروجی‌های کنترل را فراهم می‌کند، و گیت‌ها که حالت بعدی و سیگنال‌های خروجی را تولید می‌نمایند. در بخش ۱۰-۴ نشان داده شده که یک مدار ترکیبی را می‌توان به جای گیت با مولتی پلکسر پیاده‌سازی کرد. از جایگزینی گیت‌ها با مولتی پلکسرها الگوی منظمی از سه طبقه عناصر حاصل می‌گردد. اولین سطح متشکل از مولتی پلکسرهایی است که حالت بعدی ثبات را معین می‌کنند. دومین سطح حاوی ثباتی است که حالت دودویی فعلی را نگه می‌دارد. سومین سطح دیکدری دارد که خروجی جداگانه‌ای را برای هر حالت کنترل فراهم می‌کند. این سه عنصر سلول‌های استاندارد از پیش تعریف شده در بسیاری از مدارهای مجتمع می‌باشد.

به عنوان مثال، چارت ASM شکل ۱۹-۸ را ملاحظه کنید. این چارت از چهار حالت و چهار ورودی کنترل تشکیل شده است. جعبه‌های حالت خالی در اینجا گذاشته شده‌اند زیرا ما فقط علاقمند به



شکل ۱۹-۸ . مثالی از چارت ASM با چهار ورودی کنترل



شکل ۲۰-۸. پیاده سازی کنترل با مولتی پلکسر

رشته کنترلی هستیم که مستقل از عملیات ثباتی است. تخصیص دودویی برای هر حالت در گوشه سمت راست بالای جعبه‌های حالت مشخص شده است. جعبه‌های تصمیم، گذرهای حالت را به صورت تابعی از چهار ورودی کنترل w، x، y و z مشخص می‌نمایند. پیاده‌سازی کنترل سه طبقه‌ای در شکل ۲۰-۸ نشان داده شده است. این مدار شامل مولتی پلکسرهای MUX 1، MUX 2 با یک ثبات متشکل از دو فلیپ فلاپ G_0 ، G_1 و یک دیکدر با چهار خروجی است. خروجی‌های ثبات به ورودی‌های دیکدر و نیز ورودی‌های انتخاب مولتی پلکسرها وصل شده است. به این ترتیب حالت فعلی ثبات برای انتخاب یکی از ورودی‌های هر مولتی پلکسر به کار برده می‌شود. آنگاه خروجی‌های مولتی پلکسرها به ورودی‌های D فلیپ فلاپ‌های G_1 و G_0 وصل می‌شود. هدف از کاربرد مولتی پلکسرها تهیه ورودی برای فلیپ فلاپ مربوطه‌اش می‌باشد به نحوی که برابر با مقدار دودویی حالت بعدی باشد. ورودی مولتی پلکسرها از جعبه‌های تصمیم‌گیری و گذر حالت موجود در چارت ASM حاصل می‌گردد. مثلاً حالت 00 بسته به مقدار ورودی w در 00 می‌ماند و یا به 01 می‌رود. چون حالت بعدی G_1 در هر حال 0 است یک 0 در ورودی شماره 0 مولتی پلکسر 1 قرار می‌دهیم. اگر $w = 0$ باشد حالت بعدی G_0 برابر 0 و اگر $Q = 1$ باشد $G_0 = 1$ خواهد بود. چون حالت بعدی G_0 برابر w است، ورودی کنترل w را به ورودی 0 از MUX 2 وصل می‌نماییم. معنی این کار این است که وقتی ورودی‌های انتخاب مولتی‌پلکسرها برابر حالت فعلی 00 است، خروجی مولتی پلکسرها مقداری را به دودویی تولید می‌کنند که در پالس ساعت بعدی به ثبات منتقل می‌گردد.

جدول ۸-۷ شرایط ورودی مولتی پلکسر

حالت فعلی		حالت بعدی		شرایط ورودی	ورودی ها	
G_1	G_0	G_1	G_0		MUX1	MUX2
0	0	0	0	w'		
0	0	0	1	w	0	w
0	1	1	0	x		
0	1	1	1	x'	1	x'
1	0	0	0	y'		
1	0	1	0	yz'	$yz' + yz = y$	yz
1	0	1	1	yz		
1	1	0	1	$y'z$		
1	1	1	0	y		
1	1	1	1	$y'z'$	$y + y'z' = y + z'$	$y'z + y'z' = y'$

برای ایجاد امکان ارزیابی ورودی‌های مولتی پلکسر جدولی را آماده می‌کنیم تا شرایط ورودی را برای هر گذر ممکن در چارت ASM نشان دهد. جدول ۷-۸ این اطلاعات را برای چارت ASM شکل ۱۹-۸ ارائه می‌کند. از حالت فعلی 00 یا 01 دو گذر و از حالت فعلی 10 یا 11 سه گذر یا انتقال وجود دارد. این حالات در جدول با خطی افقی از یکدیگر جدا شده‌اند. شرایط ورودی در جدول از جعبه‌های تصمیم در چارت ASM بدست آمده‌اند. مثلاً در شکل ۱۹-۸ می‌بینیم که اگر $x = 1$ باشد حالت فعلی 01 به 10 خواهد رفت و اگر $x = 0$ باشد حالت بعدی 11 است. در جدول این شرایط را به ترتیب با x و x' علامت می‌زنیم. دو ستون زیر "ورودی‌های مولتی پلکسر" در جدول مقادیر ورودی را که باید به MUX 1 و MUX 2 اعمال شوند مشخص می‌کنند. ورودی مولتی پلکسر برای هر حالت فعلی از شرایط ورودی وقتی که حالت بعدی فلیپ فلاپ برابر 1 است بدست می‌آید. بنابراین بعد از حالت فعلی 01، حالت بعدی G_1 همواره برابر با 1 و حالت بعدی G_2 برابر با متمم x است. بنابراین هنگام حالت فعلی 01 ثبات، ورودی MUX 1 به 1 و MUX 2 به x' وصل می‌شوند. به عنوان مثالی دیگر، پس از حالت فعلی 10، حالت بعدی G_1 باید برابر 1 باشد به شرطی که شرایط ورودی yz' و یا yz برابر قرار باشد. اگر این دو تابع بولی با یکدیگر OR شوند، همانطور که در جدول دیده می‌شود تنها متغیر y باقی خواهد ماند. اگر شرایط $yz = 11$ در ورودی برقرار باشد، حالت بعدی G_0 برابر 1 خواهد بود. اگر حالت بعدی G_1 پس از حالت مفروضی در 0 باقی بماند، ما یک 0 در ورودی مولتی پلکسر طبق آنچه در حالت فعلی 00 برای مولتی پلکسر MUX 1 نشان داده شده قرار می‌دهیم. همانطور که در حالت فعلی 00 برای MUX 2 دیده شد، اگر حالت بعدی G_1 همواره 1 باشد، در ورودی مولتی پلکسر، 1 قرار می‌دهیم. این کار برای حالت فعلی 01 برای MUX 1 نشان داده شده است. سایر عناصر جدول برای MUX 1 و MUX 2 به طور مشابه حاصل می‌شوند. آنگاه ورودی‌های مولتی پلکسرها از جدول، در پیاده‌سازی کنترل شکل ۲۰-۸ به کار می‌رود. توجه کنید که اگر حالت بعدی یک فلیپ فلاپ تابعی از دو یا چند متغیر باشد، ممکن است مولتی پلکسر به یک یا چند گیت در ورودی‌اش نیاز داشته باشد. در غیر این صورت ورودی مولتی پلکسر برابر با متغیر کنترل، یا متمم آن، یا 0 و یا 1 است.

مثال طراحی - شمارش تعداد 1 ها در یک ثبات

ما پیاده‌سازی یک کنترل مولتی پلکسری را با یک مثال نشان خواهیم داد. مثال، فرموله شدن چارت ASM و پیاده‌سازی زیرسیستم مسیر داده (پردازشگر) را نشان خواهد داد.

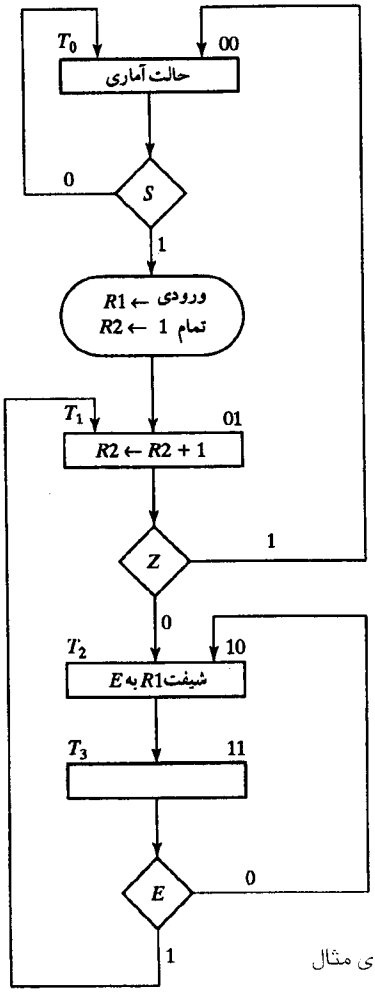
سیستم دیجیتالی که قرار است طراحی شود متشکل از دو ثبات R_1 و R_2 و یک فلیپ فلاپ E می‌باشد. سیستم تعداد 1های عدد بار شده در ثبات R_1 را می‌شمارد و ثبات R_2 را با تعداد شمارش شده پر می‌کند. به عنوان مثال، اگر عدد دودویی بار شده در ثبات R_1 برابر 10111001 باشد، مدار 5 عدد 1 را در R_1 شمرده و ثبات R_2 را با عدد دودویی 101 پر خواهد کرد. این عمل هر بار با جابجایی هر بیت از R_1 به فلیپ فلاپ E صورت می‌گیرد. مقدار E بعداً به وسیله کنترل چک شده و هر دفعه که برابر با 1 باشد ثبات R_2 یک واحد افزایش می‌یابد.

کنترل از یک ورودی بیرونی S برای شروع عملیات و دو ورودی وضعیت E و Z از پردازشگر استفاده می‌کند. E خروجی فلیپ فلاپ و Z خروجی مداری است که محتوای ثبات R_1 را برای یافتن حالت تمام 0 چک می‌کند. وقتی $R_1 = 0$ باشد خروجی $Z = 1$ تولید خواهد شد.

چارت ASM برای مدار در شکل ۲۱-۸ نشان داده شده است. عدد دودویی در R_1 بار می‌شود و R_2 با مقدار تمام 1 پر می‌شود. توجه کنید که عدد تمام 1 در یک ثبات وقتی افزایش یابد حالت تمام 0 را تولید می‌نماید. در حالت T_1 ، ثبات R_2 افزایش می‌یابد و محتوای R_1 مورد وارسی قرار می‌گیرد. اگر محتوا 0 باشد، آنگاه $Z = 1$ شده و نشان می‌دهد که در ثبات R_1 ، 1 ، وجود ندارد، بنابراین عملیات پایان یافته و محتوای R_2 برابر 0 خواهد بود. اگر محتوای R_1 غیرصفر باشد، آنگاه $Z = 0$ شده و به این معنی است که تعدادی 1 در ثبات ذخیره شده است. آنگاه عدد موجود در R_1 جابجا شده و سمت چپ‌ترین بیت آن به E منتقل می‌گردد. این عمل به دفعاتی که برای انتقال 1 به E لازم باشد تکرار می‌گردد. برای هر 1 یافت شده در E ثبات R_2 یک واحد افزایش یافته و ثبات R_1 برای یافتن 1های بیشتر وارسی می‌شود. حلقه بزرگتر آن قدر تکرار می‌گردد تا تمام 1های موجود در R_1 شمرده شوند. دقت کنید که جعبه حالت T_3 دارای عملیات ثباتی نیست، ولی بلوک متعلق به آن حاوی جعبه تصمیم برای E است. و نیز توجه کنید که ورودی سریال به شیفت رجیستر R_1 باید معادل 0 باشد زیرا نمی‌خواهیم 1های بیرونی وارد R_1 شوند.

نمودار بلوکی مدار در شکل ۲۲-۸ نشان داده شده است. کنترل سه ورودی و چهار خروجی دارد. فقط سه خروجی توسط پردازشگر یا مسیر داده استفاده شده است. ثبات R_1 یک شیفت رجیستر است. ثبات R_2 یک شمارنده با امکان بار شدن موازی می‌باشد. برای پرهیز از پیچیدگی نمودار ساعت نشان داده نشده، ولی باید به دو ثبات، فلیپ فلاپ E و فلیپ فلاپ‌های کنترل اعمال گردد.

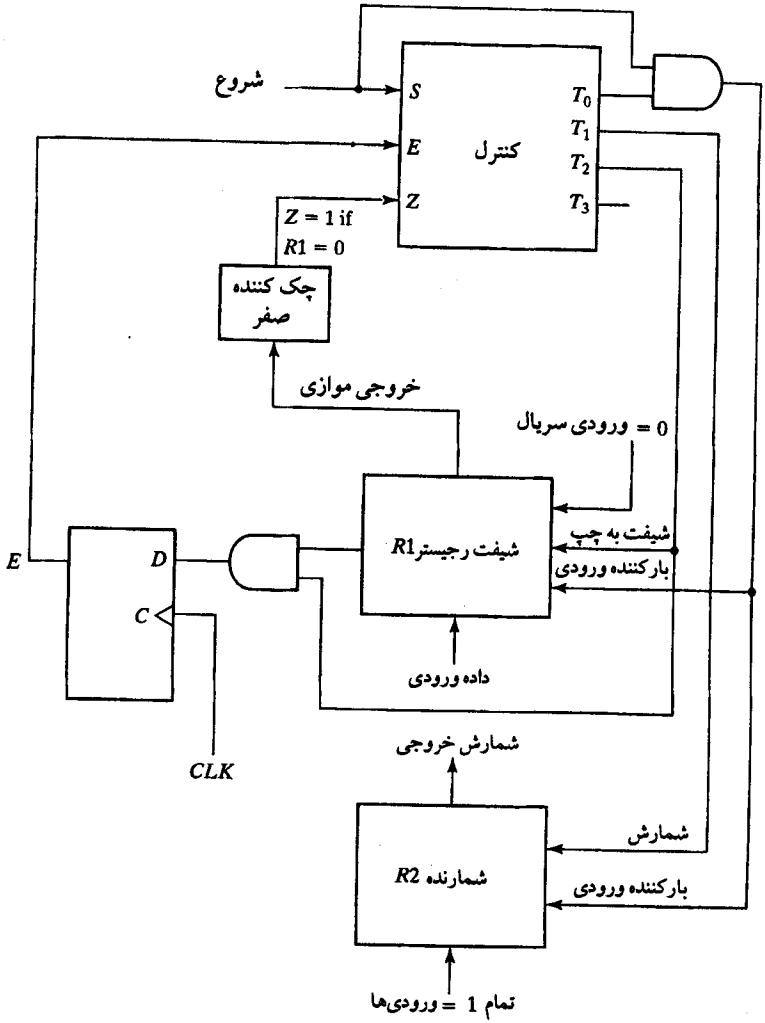
شرایط ورودی مولتی پلکسر برای کنترل از جدول ۸-۸ معین می‌گردد. این شرایط برای هر گذر حالت دودویی ممکن از جدول ASM فراهم می‌شود. به چهار حالت چهار مقدار 00 تا 11 تخصیص یافته است. گذر از حالت فعلی 00 به S وابسته است. گذر از 01 به Z و گذر از حالت فعلی 11 به E بستگی دارد. حالت فعلی 10 بدون شرط در پالس ساعت بعدی به 11 می‌رود. مقادیر زیر ستون‌های



شکل ۸-۲۱. چارت ASM برای مثال

جدول ۸-۸. شرایط ورودی مولتی پلکسر برای مثال طراحی

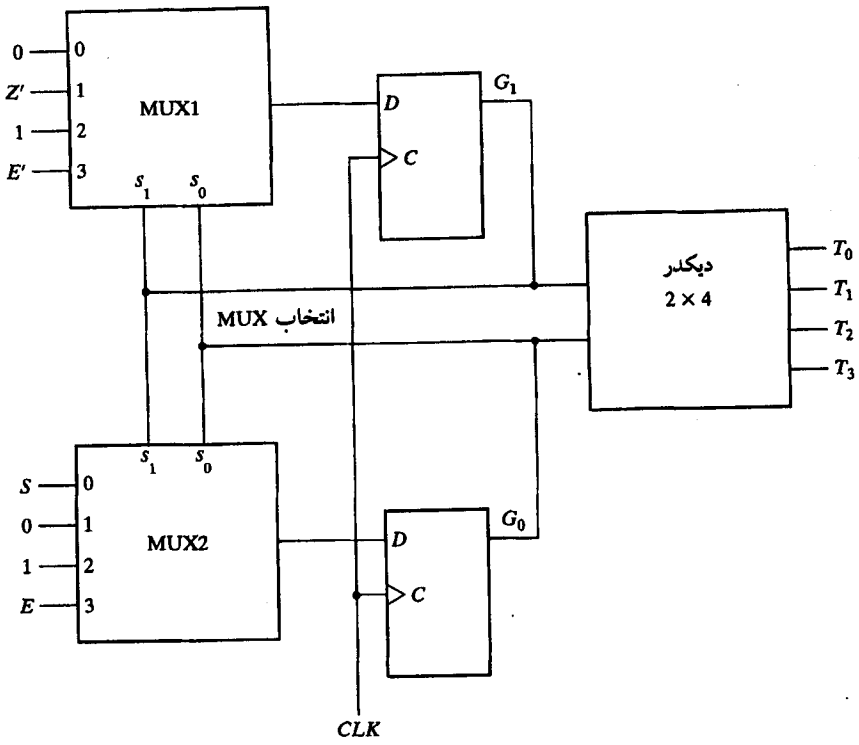
حالت فعلی		حالت بعدی		شرایط ورودی	ورودی‌های مولتی پلکسر	
G_1	G_0	G_1	G_0		MUX1	MUX2
0	0	0	0	S'		
0	0	0	1	S	0	S
0	1	1	0	Z		
0	1	1	1	Z'	Z'	0
1	0	1	1	None	1	1
1	1	1	0	E'		
1	1	0	1	E	E'	E



شکل ۲۲-۸. نمودار بلوکی شمارش 1ها

MUX 1 و MUX 2 در جدول به ترتیب از شرایط بولی ورودی برای حالت بعدی G_1 و G_0 فراهم می‌گردد.

پیاده سازی کنترل مثال طراحی در شکل ۲۳-۸ ملاحظه می‌شود. این مدار یک پیاده سازی سه سطحی است که در آن مولتی‌پلکسرها در سطح اول قرار گرفته‌اند. ورودی‌ها به مولتی پلکسرها از جدول۸-۸ بدست آمده‌اند.



شکل ۲۳-۸ . پیاده سازی کنترل در مثال طراحی

مسائل

۸-۱ به زبان خود عملیات انتقال ثباتی زیر را توصیف نمایید.

(الف) $R2 \leftarrow R2 + 1, R1 \leftarrow R2$

(ب) $R3 \leftarrow R3 - 1$

(پ) $\text{If } (T_1 = 1) \text{ then } (R0 \leftarrow R1) \text{ else if } (T_2 = 1) \text{ then } (R0 \leftarrow R2)$

۸-۲ بخش آغازین یک چارت ASM را رسم کنید. دو سیگنال کنترل x و y وجود دارد. اگر xy = 10 باشد، ثبات R یک واحد اضافه می‌شود و کنترل به حالت دوم می‌رود. اگر xy = 01، شد، ثبات R صفر شده و کنترل از حالت آغازین به حالت سوم می‌رود. در غیر این صورت، کنترل در حالت آغازین می‌ماند.

۸-۳ چارت‌های ASM را برای گذرهای حالت زیر بنویسید:

(الف) اگر x = 0 باشد، کنترل از T_1 به T_2 می‌رود؛ اگر x = 1 باشد یک حالت شرطی تولید کرده و از T_1 به T_2 می‌رود.

(ب) اگر $z = 1$ باشد، کنترل از T_1 به T_2 و سپس به T_3 می‌رود؛ اگر $x = 0$ باشد، کنترل از T_1 به T_3 می‌رود.

(پ) با شروع از T_1؛ آنگاه اگر $xy = 00$ باشد به T_2 برود؛ اگر $xy = 01$ باشد به T_3 برود؛ اگر $xy = 10$ باشد به T_3 برود؛ اگر $xy = 10$ باشد به T_1 برود؛ در غیر این صورت به T_3 برود.

۸-۴ هشت مسیر خروجی از بلوک ASM که از جعبه‌های تصمیم‌گیری بیرون می‌آیند را رسم کنید تا هشت حالت دودویی ممکن برای سه متغیر x و y و z را چک نماید.

۸-۵ تفاوت یک چارت ASM را از چارت معمولی بیان کنید. این تفاوت را با به کارگیری شکل ۵-۸ نشان دهید.

۸-۶ برای یک سیستم دیجیتال یک چارت ASM بنا کنید تا تعداد افراد داخل یک اطاق را بشمارد. افراد از یک درب وارد می‌شوند و هر وقت نور فتوسلی به وسیله فردی قطع شود خروجی آن تغییری از 1 به 0 در سیگنال x ایجاد می‌نماید. افراد از درب دومی اطاق را ترک می‌کنند و خروجی فتوسل مشابهی، y، از 1 به 0 تغییر می‌یابد. مدار شامل یک بالا- پایین شمار است و نمایشگر آن تعداد افراد داخل اطاق را نشان می‌دهد.

۸-۷ یک چارت ASM برای یک مدار با دو ثبات 8 بیت RA و RB که دو عدد دودویی بی‌علامت را دریافت می‌کنند و تفریق زیر را انجام می‌دهند رسم نمایید.

$$RA \leftarrow RA - RB$$

از روش بخش ۵-۱ برای تفریق استفاده کنید و اگر جواب منفی بود یک فلیپ فلاپ قرض را به 1 بنشانید.

۸-۸ یک مدرا دیجیتال با سه ثبات 16 بیتی AR، BR و CR طراحی کنید تا عملیات زیر را انجام دهد.

(الف) دو عدد علامت‌دار 16 بیتی (به متمم 2) را به AR و BR منتقل نماید.

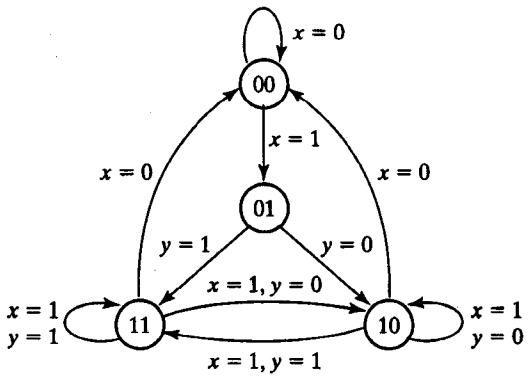
(ب) اگر عدد درون AR منفی است، عدد موجود در AR را بر 2 تقسیم کرده و نتیجه را به CR ببرید.

(پ) اگر عدد در AR مثبت ولی غیرصفر است، عدد درون BR را در 2 ضرب و نتیجه را به CR منتقل کنید.

(ت) اگر عدد درون AR صفر است ثبات CR را صفر نمایید.

۸-۹ کنترل نمودار شکل ۱۱-۸(الف) را با استفاده از یک فلیپ فلاپ بر هر حالت (تخصیص 1 بارز) طراحی کنید.

۸-۱۰ نمودار حالت واحد کنترلی در شکل (م ۱۰-۸) نشان داده شده است. این نمودار چهار حالت و دو ورودی x و y دارد. چارت ASM معادل آن را رسم نمایید و درون جعبه‌های حالت را خالی کنید.



شکل (م ۱۰-۸). نمودار حالت کنترل برای مسائل ۱۰-۸ و ۱۱-۸

۸-۱۱ کنترلی طراحی کنیدکه نمودار حالتش شکل (م ۸-۱۰) باشد. از فلیپ فلاپ D استفاده نمایید.

۸-۱۲ فرض کنید R_1 در شکل ۸-۲۲ یک شیفت رجیستر 4 بیت مطابق شکل ۷-۶ باشد. نشان دهید که چگونه ورودی‌های جابجایی و بار شدن به ورودی‌های S_1 و S_0 شیفت رجیستر وصل می‌شوند.

۸-۱۳ یک شمارنده 4 بیت با ورودی همزمان پاک شدن که در شکل ۸-۱۰ نشان داده شده است طراحی نمایید.

۸-۱۴ یک مدار دیجیتال که دو عدد دودویی را با روش جمع مکرّر در هم ضرب کند طراحی نمایید. مثلاً برای ضرب 5×4، سیستم ضرب را با چهار بار جمع مضروب انجام می‌دهد: یعنی $5 + 5 + 5 + 5 = 20$. اکنون تصور کنید که مضروب در BR، مضروب فیه در AR و حاصلضرب در ثبات PR باشد. یک مدار جمع کننده محتوای BR را با PR جمع می‌کند. یک مدار تشخیص صفر، پس از هر بار کاهش AR، Z را چک می‌کند.

۸-۱۵ ثابت کنید که ضرب دو عدد n بیتی حاصلضربی با طول کمتر یا برابر 2n بیت تولید می‌کند.

۸-۱۶ در شکل ۸-۱۳، ثبات Q مضروب فیه و B مضروب را نگه می‌دارد. هر عدد را 16 بیتی تصور کنید.

(الف) در حاصلضرب، چند بیت انتظار دارید و در کجا در دسترس است؟

(ب) شمارنده P چند بیت است، و عدد اولیه بار شده در آن چند است.

(پ) مداری طراحی کنید که صفر موجود در P را شناسایی نماید.

۸-۱۷ محتوای ثبات‌های A، C و P را مشابه جدول ۸-۴ در حین عمل ضرب برای دو عدد 1111 (مضروب) و 10101 (مضروب فیه)، لیست کنید.

۸-۱۸ زمان عمل ضرب دودویی بخش ۸-۶ را بدست آورید. فرض کنید که ثبات Q دارای n بیت بوده و سیکل ساعت هم t نانو ثانیه است.

۸-۱۹ مدار کنترل یک ضرب کننده دودویی با نمودار حالت شکل ۸-۱۵ را طراحی کنید. از مولتی پلکسر، دیکدر و یک ثبات استفاده نمایید.

۸-۲۰ چارت ASM شکل (م ۸-۲۰) را ملاحظه کنید. عملیات ثباتی مشخص نشده است زیرا ما فقط به طراحی مدار کنترل علاقمندیم.

(الف) نمودار حالت معادل را رسم نمایید.

(ب) کنترل را با یک فلیپ فلاپ در هر حالت طراحی کنید.

(پ) جدول حالت را برای کنترل لیست نمایید.

(ت) کنترل را با سه فلیپ فلاپ D، یک دیکدر و گیت طراحی کنید.

(ث) جدولی را بدست آورید که شرایط ورودی مولتی پلکسر را برای کنترل نشان دهد.

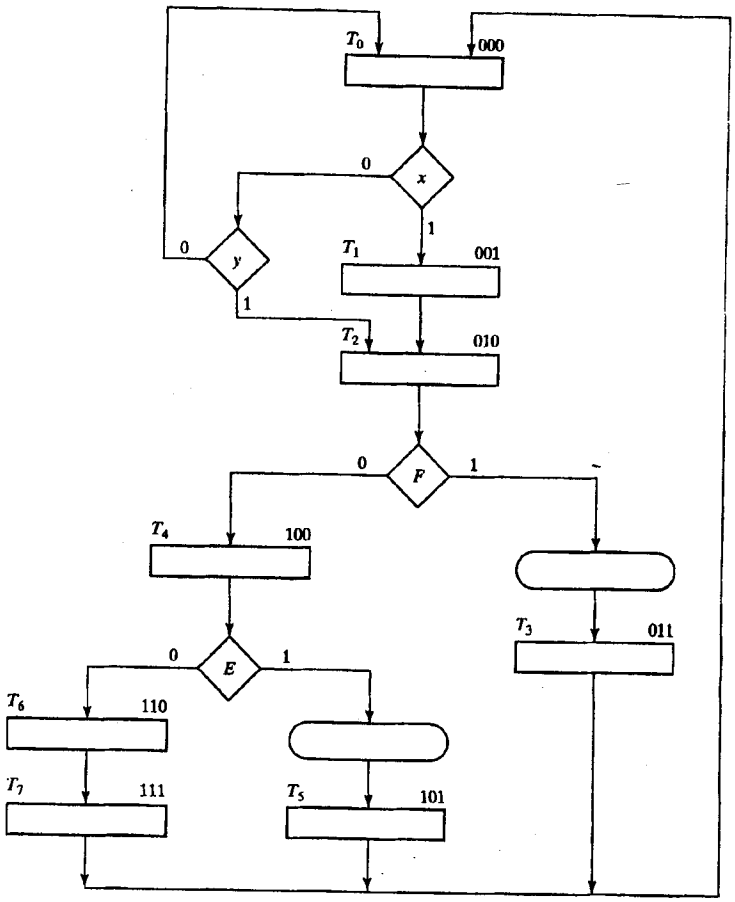
(ج) کنترل را با سه مولتی پلکسر، یک ثبات با سه فلیپ فلاپ و یک دیکدر 8×3 طراحی کنید.

۸-۲۱ در هر بلوک HDL مقدار E چقدر است. فرض کنید RA = 1 است.

(الف)
```
RA = RA - 1;
if (RA == 0) E = 1;
else E = 0;
```

(ب)
```
RA <= RA - 1;
if (RA == 0) E <= 1;
else E <= 0;
```

۸-۲۲ با به کارگیری عملگرهای Verilog HDL در جدول ۸-۱، عملیات زیر را ارزیابی کنید. فرض شود



شکل (م ۲۰-۸)

$A = 4'b0110$ ، $B = 4'b0010$ و $C = 4'b0000$ باشد.

```
A * B; A + B; A - B; ~C; A & B; A | B; A ^ B; & A; ~|C; A || B;
A && C; |A; A < B; A > B; A != B;
```

۲۳-۸ بلوک always زیر را ملاحظه کنید.

```
always @ (posedge CLK)
if (T1) R1 <= R1 + R2;
else if (T2) R1 <= R1 + 1;
else R1 <= R1;
```

با استفاده از یک شمارنده 4 بیت با بار شدن موازی برای R1 (طبق شکل ۱۴-۶) و یک جمع کننده 4 بیت، نمودار بلوکی نشان دهنده اتصالات اجزاء و سیگنال‌های کنترل را برای یک طراحی، رسم نمایید.

۲۴-۸ یک عبارت case چند سطحی اغلب به وسیله یک طراح منطقی به صورت مولتی پلکسرهای سخت‌افزاری برگردانده می‌شود. بلوک case زیر را چگونه به سخت‌افزار برمی‌گردانید؟ هر ثبات را 8 بیت تصور کنید.

```
case (state)
T0:  R4 = R0;
T1:  R4 = R1;
T2:  R4 = R2;
T3:  R4 = R3;
endcase
```

۸-۲۵ مداری طراحی کنید که تعداد 1 ها را در یک ثبات همچون بخش ۸-۹ بشمارد. چارت ASM برای این مدار در شکل ۸-۲۱ و نمودار بلوکی در شکل ۸-۲۲ دیده می‌شود.

(الف) توصیف HDL سطح انتقال ثباتی مدار را بنویسید.

(ب) مدار کنترل را با استفاده از یک فلیپ فلاپ در هر حالت (تخصیص 1 بارز) طراحی کنید. معادلات ورودی را برای هر چهار فلیپ فلاپ بنویسید.

(پ) توصیف ساختاری HDL مدار را با استفاده از کنترل طراحی شده در بخش (ب) و نمودار بلوکی شکل ۸-۲۲ بنویسید.

(ت) یک برنامه تست برای تست مدار بنویسید. برای اطمینان از صحت عملکرد مدار در هر دو برنامه ساختاری و RTL، آن را شبیه‌سازی کنید.

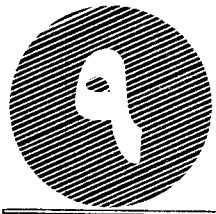
۸-۲۶ توصیف ساختاری HDL ضرب کننده طراحی شده در بخش ۸-۶ را بنویسید. نمودار بلوکی شکل ۸-۱۳ و مدار کنترل شکل ۸-۱۷ را به کار ببرید. طرح را شبیه‌سازی کنید و عملکرد آن را با استفاده از برنامه تست مثال ۸-۶ چک کنید.

۸-۲۷ جمع دو عدد دودویی علامت‌دار در نمایش مقدار - علامت از قوانین حساب معمولی تبعیت می‌کند. اگر هر دو عدد علامت یکسانی داشته باشند، دو مقدار با هم جمع می‌شوند و علامت هم یکی از علائم است. اگر دو عدد علامت مخالف داشته باشند مقدار کوچکتر از بزرگتر کسر می‌شود و نتیجه با علامت عدد بزرگتر خواهد بود. سمت چپ‌ترین بیت علامت و 7 بیت دیگر مقدار را نگاه می‌دارند.

۸-۲۸ توصیف HDL مدار مسئله ۸-۱۴ را بنویسید.

مراجع

1. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis.* SunSoft Press (A Prentice Hall Title).
2. BHASKER, J. 1997. *A Verilog HDL Primer.* Allentown, PA: Star Galaxy Press.
3. BHASKER, J. 1998. *Verilog HDL Synthesis.* Allentown, PA: Star Galaxy Press.
4. THOMAS, D. E., and P. R. MOORBY. 1998. *The Verilog Hardware Description Language* 4th ed. Boston: Kluwer Academic Publishers.
5. CILETTI, M. D. 1999. *Modeling, Synthesis, and Rapid Prototyping with Verilog HDL.* Upper Saddle River, NJ: Prentice Hall.
6. ARNOLD, M. G. 1999. *Verilog Digital Computer Design.* Upper Saddle River, NJ: Prentice Hall.
7. SMITH, D. J. 1996. *HDL Chip Design.* Madison, AL: Doone Publications.



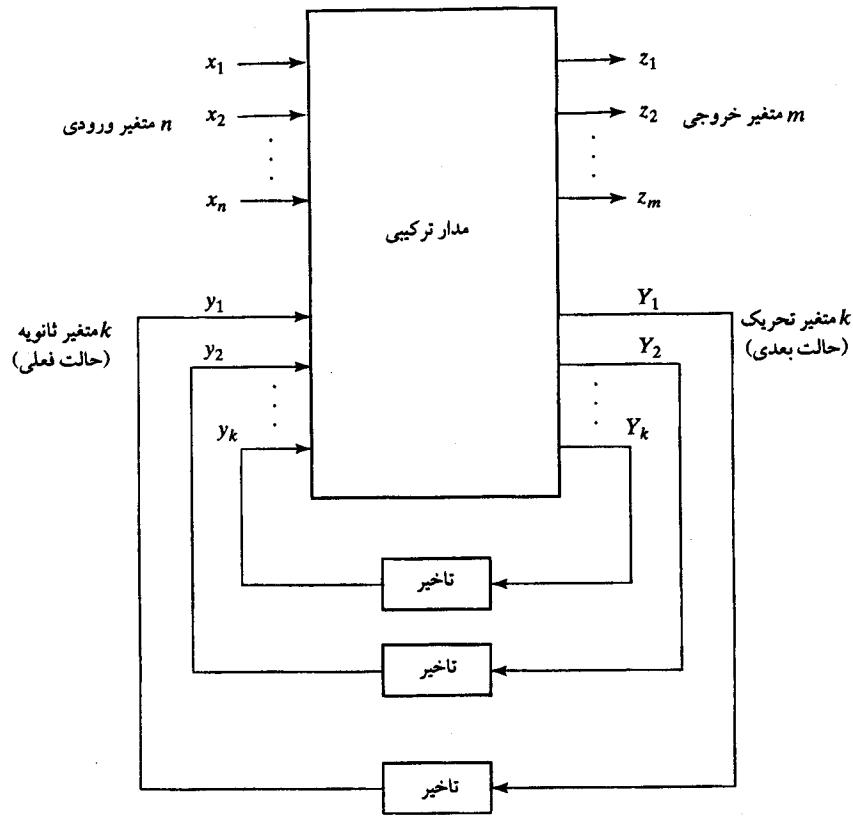
مدارهای ترتیبی غیرهمزمان

۹-۱ مقدمه

یک مدار ترتیبی با دنباله‌ای از ورودی‌ها، خروجی‌های وابسته به زمان و حالات داخلی‌اش مشخص می‌شود. در مدارهای ترتیبی همزمان (سنکرون) تغییر حالت داخلی به وسیلهٔ اعمال پالس‌های ساعت زمانبندی شده صورت می‌گیرد. مدارهای ترتیبی غیرهمزمان (آسنکرون) از پالس‌های ساعت استفاده نمی‌کنند. در این گونه مدارها تغییر حالت به هنگام تغییر در ورودی‌ها رخ می‌دهد. عناصر حافظه در مدارهای ترتیبی همزمان، فلیپ فلاپ‌های ساعت‌دار می‌باشند. عناصر حافظه در مدارهای ترتیبی غیرهمزمان، فلیپ فلاپ‌های بدون ساعت و یا عناصر تأخیر زمانی هستند. توانایی حفظ اطلاعات در یک قطعه تولیدکننده تأخیر زمانی از این حقیقت ناشی می‌شود که این قطعه برای انتشار یک سیگنال از داخل گیت‌های دیجیتال، زمان محدودی را نیاز دارد. یک مدار ترتیبی غیرهمزمان در بسیاری از موارد شبیه به یک مدار ترکیبی پسخوردی (فیدبک) عمل می‌کند.

به دلیل مسائل و مشکلات زمانی مربوط به مسیر پسخورد، طراحی مدارهای ترتیبی غیرهمزمان، بسیار مشکل‌تر از طراحی مدارهای همزمان است. در یک سیستم همزمان که به درستی طراحی شده باشد مشکلات زمانی با راه‌اندازی همزمان (تریگر) کلیه فلیپ‌فلاپ‌ها با لبه پالس از بین می‌رود. تغییر از یک حالت و رفتن به حالت بعدی، طی زمان کوتاه تغییر وضعیت پالس، که همان گذر پالس می‌باشد، صورت می‌گیرد. نظر به این که مدارهای غیرهمزمان از پالس ساعت استفاده نمی‌کنند، تغییر حالت سیستم بلافاصله پس از هر تغییر در ورودی رخ می‌دهد. در این گونه سیستم‌ها باید مراقب بود که با توجه به وجود پسخورد، هر حالت جدید، مدار را در یک وضعیت پایدار نگهدارد.

مدارهای ترتیبی غیرهمزمان در کاربردهای متنوعی مورد استفاده قرار می‌گیرند. از این مدارها هنگامی استفاده می‌شود که سرعت عملیات مدّنظر است، و به ویژه در حالت‌هایی که سیستم دیجیتال



شکل ۱-۹ . نمودار بلوکی یک مدار ترتیبی همزمان

باید به سرعت و بد ون انتظار برای پالس ساعت عکس‌العمل نشان دهد، این مدارها مورد توجه‌انـد. مدارهای غیرهمزمان در سیستم‌های کوچکی که فقط نیاز به تعداد کمی قطعه دارند و هزینه تهیه یک مدار جداگانه برای تولید پالس ساعت مقرون به صرفه نـیست بسـیار اقـتصادی‌انـد. ایـن مـدارهـا در کاربردهایی که سیگنال‌های ورودی به سیستم ممکن است در هر لحظه و مستقل از یک ساعت داخلی تغییر کنند، مفید می‌باشند. اتصال دو واحد که هر یک دارای پالس ساعت مستقل خودش است باید به وسیله مدارهای غیرهمزمان صورت گیرد. طراحان دستگاه‌های دیجیتال اغلب یک سیستم مختلط تولید می‌کنند به نحوی که بعضی از قسمتهای دستگاه همزمان دارای ویژگی‌های یک مدار غیرهمزمان است. آگاهی از عملکرد یک مدار ترتیبی غیرهمزمان می‌تواند در اطمینان از صحت عملکرد آن مفید باشد.

شکل ۱-۹ نمودار بلوکی یک مدار ترتیبی غیرهمزمان را نشان می‌دهد. این مدار شامل یک بخش ترکیبی و عناصر تأخیری برای تشکیل حلقه‌های تأخیر است. مدار دارای n مـتغیر ورودی، m مـتغیر خروجی، و k حالت داخلی است. عناصر تأخیر زمانی را در مدارهـای تـرتیبی مـی‌توان بـه صـورت حافظه‌های کوتاه‌مدت تجسم کرد. در یک مدار گیتی، تأخیر انتشار موجود در مسیر ورودی به خروجی

به اندازه کافی تأخیر در حلقه پسخورد تولید می‌کند. بنابراین هیچ عنصر تأخیر زمانی در مسیر پسخورد لحاظ نمی‌شود. معمولاً متغیرهای حالت فعلی و حالت بعدی در مدارهای ترتیبی غیرهمزمان به ترتیب، متغیرهای ثانوی و متغیرهای تحریک نامیده می‌شوند. متغیرهای تحریک را نباید با متغیرهای جدول تحریک به کار رفته در طراحی مدارهای ترتیبی ساعت دار اشتباه کرد.

وقتی که مقدار یک متغیر ورودی تغییر می‌کند، متغیرهای ثانوی y به طور آنی تغییر نمی‌کنند. انتشار سیگنال از ورودی و از طریق مدار ترکیبی به متغیرهای تحریک Y برای تولید حالت بعدی جدید، به مقدار معینی زمان نیاز دارد. این مقادیر از طریق عناصر تأخیر انتشار می‌یابند تا تبدیل به حالت فعلی جدیدی برای متغیرهای ثانوی شوند. به اختلاف بین yها و Yها توجه کنید. در حالت پایدار، آنها یکی هستند، ولی در حین گذر یکسان نمی‌شوند. برای مقدار مفروضی از متغیرهای ورودی، اگر مدار مذکور به یک حالت با ثبات یا پایدار برسد به نحوی که $y_i = Y_i$ با $i = 1, 2, ..., k$ باشد، سیستم موردنظر پایدار است، در غیر این صورت به طور مداوم در حال گذر پیوسته‌ای بوده و ناپایدار خواهد بود. در مدارهای غیرهمزمان درک این مطلب که گذر از یک حالت پایدار به حالتی دیگر فقط در پاسخ به یک تغییر در متغیرهای ورودی است حائز اهمیت می‌باشد. برعکس در مدارهای همزمان انتقال حالت در پاسخ به اعمال یک پالس ساعت صورت می‌گیرد.

به منظور اطمینان از یک عملکرد صحیح، باید اجازه داد تا مدار ترتیبی غیرهمزمان قبل از هر تغییر جدید در ورودی‌هایش، به یک حالت با ثبات و پایداری برسد. به دلیل وجود تأخیر در سیم‌ها و گیت‌های مدار، داشتن دو تغییر همزمان و یا بیشتر غیرممکن است و حتی به این که کدام اول رخ داده است اطمینان چندانی نیست. بنابراین تصور تغییر همزمان دو یا چند متغیر ورودی منع می‌شود. این قید به این معنی است که در هر لحظه از زمان فقط یک متغیر ورودی مجاز به تغییر است و زمان بین دو تغییر هم باید از زمان رسیدن مدار به یک حالت پایدار طولانی‌تر باشد. این نوع عملیات به عنوان عملیات اساسی خوانده می‌شوند. در عملیات اساسی فرض بر این است که سیگنال‌های ورودی یک به یک و هنگام پایداری مدار تغییر می‌کنند.

۲-۹ روش تحلیل

تحلیل مدارهای ترتیبی غیرهمزمان به معنی تهیه جدول یا نموداری است که رشته حالات داخلی و خروجی‌ها را به صورت تابعی از تغییرات متغیرهای ورودی توصیف نماید. اگر یک مدار ترتیبی غیرهمزمان دارای یک یا چند حلقه پسخورد باشد و یا از فلیپ فلاپ‌هایی که به پالس ساعت حساس نیستند تشکیل شده باشد نمودار منطقی آن را نشان خواهد داد. در این بخش رفتار مدارهای ترتیبی غیرهمزمانی که دارای مسیرهای پسخورد هستند و از فلیپ‌فلاپ‌ها استفاده نمی‌کنند را مورد بررسی قرار می‌دهیم. به فلیپ فلاپ‌های بدون ساعت لچ هم می‌گویند و استفاده از آنها در مدارهای ترتیبی غیرهمزمان در بخش بعد تشریح خواهد شد.

روش تحلیل همراه با سه مثال خاص ارائه می‌شود. مثال اول جدول گذر را معرفی می‌کند. دومین

مثال جدول روند را معرفی می‌نماید. در مثال سوم ثُبات مدارهای ترتیبی غیرهمزمان مورد تفحص قرار می‌گیرد.

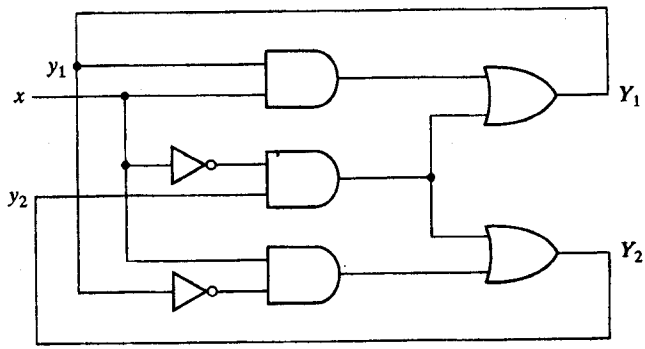
جدول کذر

مثالی از یک مدار ترتیبی غیرهمزمان که فقط با گیت‌ها ساخته شده باشد در شکل ۲-۹ دیده می‌شود. این نمودار به وضوح دو حلقه پسخورد را از خروجی گیت‌های OR به ورودی گیت‌های AND نشان می‌دهد. مدار شامل یک متغیر ورودی x و دو حالت داخلی است. حالات داخلی شامل دو متغیر تحریک Y_1 و Y_2 و دو متغیر ثانویه y_1 و y_2 می‌باشند. تأخیر مربوط به هر حلقه پسخورد از تأخیر انتشار بین هر ورودی y و خروجی Y مربوط به آن بدست می‌آید. هر گیت منطقی در مسیر، تأخیر انتشاری حدود 2 تا 10ns را تولید می‌کند. سیم‌هایی که سیگنال‌های الکترونیکی را هدایت می‌کنند تقریباً یک نانو ثانیه تأخیر را در هر فوت ایجاد می‌نمایند. بنابراین وقتی که مدار ترکیبی و سیم‌های مسیر پسخورد تأخیر کافی را به وجود می‌آورند، نیازی به عناصر اضافی تولید کننده تأخیر نیست.

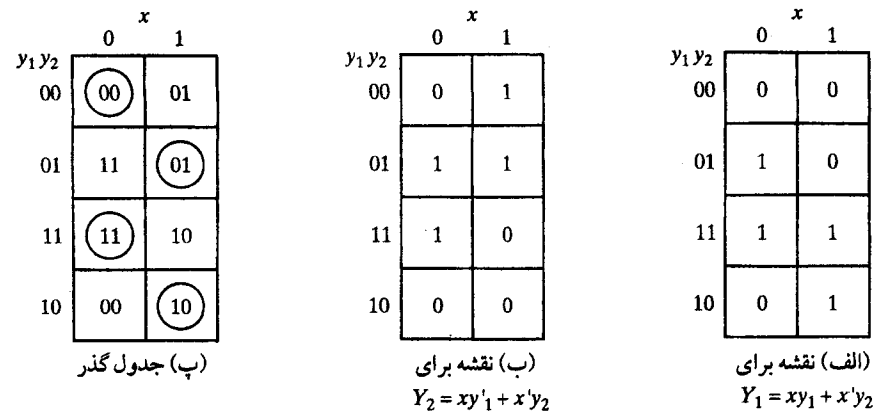
تحلیل یک مدار با ملاحظه متغیرهای تحریک به عنوان خروجی‌ها و متغیرهای ثانویه به عنوان ورودی‌ها شروع می‌شود. آنگاه عبارات بولی را برای متغیرهای تحریک به صورت تابعی از ورودی‌ها و متغیرهای ثانویه بدست می‌آوریم. این عبارات به راحتی از نمودار منطقی بدست می‌آیند.

$$Y_1 = xy_1 + x'y_2$$
$$Y_2 = xy_1' + x'y_2$$

قدم بعد ترسیم توابع Y_1 و Y_2، طبق شکل ۳-۹ (الف) و (ب) در یک نقشه است. مقادیر دودویی انکد شده متغیرهای y برای نام‌گذاری سطرها و متغیر ورودی x برای ستون‌ها به کار رفته‌اند. این آرایش با کمی اختلاف از نقشه سه متغیره‌ای که در فصل‌های قبل به کار رفت، نتیجه می‌شود. با این وجود، علیرغم اعتبار نقشه، این نوع آرایش هنگام بررسی مدارهای ترتیبی غیرهمزمان مناسب‌تر است. توجه کنید که متغیرهای متعلق به مربع‌های مربوطه در امتداد اضلاع نقشه، آن طور که در فصل‌های قبل دیدیم، ذکر نشده‌اند.



شکل ۲-۹ . مثالی از مدار ترکیبی غیرهمزمان



شکل ۳-۹. نقشه‌ها و جدول گذر مدار شکل ۲-۹

جدول گذر شکل ۳-۹ (پ) با ترکیب مقادیر دودویی مربع‌های مربوطه، در نقشه‌های فوق حاصل می‌شود. این جدول در داخل هر مربع مقدار $Y = Y_1Y_2$ را نشان می‌دهد. اولین بیت Y از مقدار Y_1 و دومین بیت از Y_2 در همان مربع بدست می‌آید. برای داشتن یک حالت پایدار، مقدار Y باید برابر با $y = y_1y_2$ باشد. وارده‌هایی از جدول که در آنها $Y = y$ است با دایره محصور شده‌اند تا حالات پایدار مشخص شوند. وارده‌هایی که با دایره محصور نشده‌اند، حالت‌های ناپایدارند.

اکنون به تأثیر یک تغییر در متغیر ورودی توجه کنید. مربع مربوط به $x = 0$ و $y = 00$ در جدول گذر نشان می‌دهد که $Y = 00$ است. چون Y حالت بعدی y را نشان می‌دهد، این حالت یک حالت پایدار است. اگر هنگام $y = 00$، x از 0 به 1 تغییر کند، مدار مقدار Y را به 01 تغییر خواهد داد. این وضعیت یک حالت موقت ناپایدار را نشان می‌دهد زیرا Y با مقدار فعلی y برابر نیست. آنچه که بعد از این اتفاق می‌افتد این است که به محض انتشار سیگنال برای تولید $Y = 01$، مسیر پسخورد موجب می‌شود تا $y = 01$ شود. این مطلب با یک گذر یا انتقال از اولین سطر $(y = 00)$ به سطر دوم که در آن $y = 01$ است، نشان داده می‌شود. اکنون $Y = y$ است و مدار با $x = 1$ به یک حالت پایدار رسیده است. به طور کلی اگر یک ورودی مدار را به حالت ناپایداری بکشاند، ضمن ثابت ماندن x، مقدار y آنقدر تغییر می‌کند تا به یک حالت باثبات (محصور در دایره) برسد. با استفاده از این تحلیل برای بقیه مربعات جدول گذر، در می‌یابیم که مدار شبیه حالات 00، 01، 11، 10 را وقتی x متناوباً 0 و 1 شود تولید خواهد کرد.

به اختلاف بین یک مدار ترتیبی همزمان و غیرهمزمان توجه نمایید. در یک سیستم همزمان حالت فعلی کلاً با مقادیر فیلپ فلاپ‌ها مشخص می‌شود و اگر ساعت غیرفعال باشد در ازاء تغییر ورودی، این حالت تغییری نمی‌کند. در یک مدار غیرهمزمان، حالت داخلی بلافاصله پس از تغییر در ورودی، عوض می‌شود. به این علت گاهی بهتر است حالت داخلی را با مقدار ورودی ترکیب کرده و آن را حالت کلی مدار بنامیم. مداری که جدول گذر آن در شکل ۳-۹ (پ) نشان داده شده است چهار حالت کلی پایدار $y_1y_2x = 000$، 011، 110 و 101 و چهار حالت ناپایدار 001، 010، 111 و 100 را داراست.

جدول ۱-۹. جدول حالات مدار شکل ۲-۹

حالت فعلی		حالت بعدی			
		$x = 0$		$x = 1$	
0	0	0	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	1	1	1	1	0

جدول گذر مدارهای ترتیبی غیرهمزمان مشابه با جدول حالت در مدارهای ترتیبی همزمان است. اگر متغیرهای ثانویه را به عنوان حالت فعلی و متغیرهای تحریک را حالت بعدی تصور کنیم، جدول حالت ۱-۹ را بدست خواهیم آورد. این جدول اطلاعات مشابهی را همچون جدول گذر فراهم می‌کند. در حالت غیرهمزمان یک محدودیتی وجود دارد که در حالت همزمان دیده نمی‌شود. در جدول گذر غیرهمزمان، معمولاً حداقل یک وارده حالت بعدی مثل مقدار حالت فعلی در هر سطر است. در غیر این صورت همه حالات کلی در آن سطر بی‌ثبات خواهند بود.

روش تهیه جدول گذر از نمودار یک مدار ترتیبی غیرهمزمان به طریق زیر است:

۱- همه حلقه‌های پسخورد را در مدار معین کنید.

۲- خروجی هر حلقه را با متغیر Y_i و ورودی مربوطه‌اش را با y_i مشخص نمایید به طوری که $i = 1, 2, ..., k$ باشد که در آن k تعداد حلقه‌های پسخورد در مدار است.

۳- توابع بولی همه Yها را به عنوان تابعی از ورودی‌های بیرونی و yها بدست آورید.

۴- با استفاده از متغیرهای y برای سطرها و ورودی‌های بیرونی برای ستون‌ها، هر تابع Y را در یک نقشه ترسیم کنید.

۵- همه نقشه‌ها را در یک جدول با هم ترکیب نمایید به نحوی که مقدار هر $Y = Y_1Y_2, ..., Y_k$ را در هر مربع نشان دهد.

۶- مقداری از Y در هر مربع که با مقدار $y = y_1y_2, ..., y_k$ در همان ردیف برابر است رادر دایره محصور کنید.

پس از تکمیل جدول گذر، رفتار مدار با مشاهده گذر حالت به عنوان تابعی از تغییرات در متغیرهای ورودی قابل تحلیل است.

جدول روند

در حین طراحی مدارهای ترتیبی غیرهمزمان، بهتر است بدون ذکر مقادیر دودویی، حالت‌ها را با سمبل‌های حرفی نامگذاری کنیم. چنین جدولی را جدول روند نامند. جدول روند مشابه با جدول گذر است با این تفاوت که حالات داخلی به جای اعداد دودویی با حروف نشان داده شده‌اند. جدول روند شامل مقادیر خروجی مدار برای هر حالت پایدار نیز هست.

	x = 0	x = 1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

(الف) چهار حالت با یک ورودی

	x_1x_2 = 00	01	11	10
a	(a), 0	(a), 0	(a), 0	b, 0
b	a, 0	a, 0	(b), 1	(b), 0

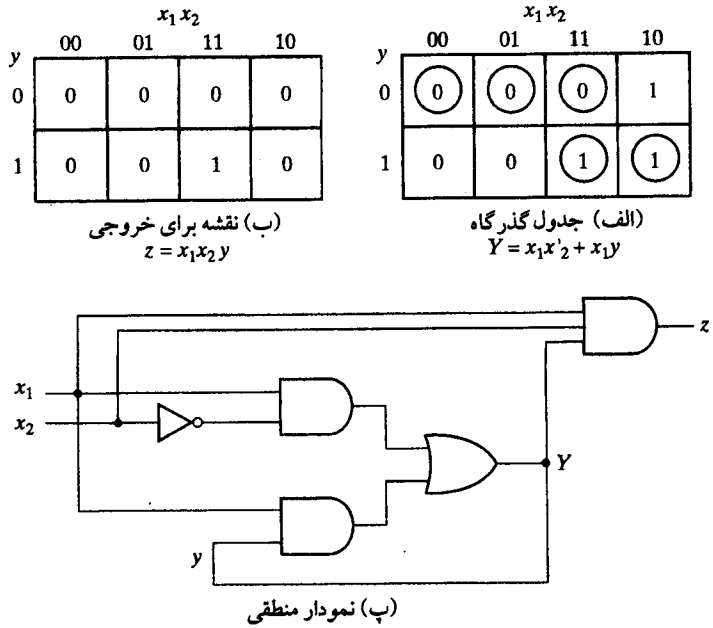
(ب) دو حالت با دو ورودی و یک خروجی

شکل ۴-۹ . مثالهایی از جدولهای روند

چند مثال از جدول روند در شکل ۴-۹ ملاحظه می‌شود. در شکل ۴-۹ (الف) چهار حالت a، b، c و d را ملاحظه می‌کنید. اگر مقادیر $a = 00$، $b = 01$، $c = 11$ و $d = 10$ را به حالات آن تخصیص دهیم شکل ۳-۹ (پ) نتیجه خواهد شد. جدول شکل ۴-۹ (الف) را جدول روند اصلی می‌نامند زیرا در هر سطر فقط دارای یک حالت پایدار می‌باشد. شکل ۴-۹ (پ) جدول روندی را نشان می‌دهد که در همان سطر بیش از یک حالت پایدار را داراست. این جدول دو حالت a و b، دو ورودی x و y و یک خروجی z دارد. مقدار دودویی متغیر خروجی در داخل مربع و مجاور سمبل حالت مدار مشخص شده و با یک ویرگول تفکیک شده است. از روی جدول روند، رفتار زیر قابل استنباط است. اگر $x_1 = 0$ باشد، مدار در حالت a است. اگر x_1 به 1 برود و x_2 در 0 باشد، مدار به حالت b می‌رود. با $x_1x_2 = 11$، مدار ممکن است در حالت a یا حالت b باشد. اگر در حالت a قرار داشته باشد، خروجی 0 و اگر در حالت b باشد، خروجی 1 است. اگر ورودی‌ها از 10 به 11 تغییر کنند حالت b حفظ می‌شود. اگر ورودی‌ها از 01 به 11 بروند مدار در حالت a باقی خواهد ماند. به خاطر دارید که در حالت اساسی، دو متغیر ورودی به طور همزمان نمی‌توانند تغییر کنند و بنابراین، اجازه تغییر 00 به 11 را نخواهیم داد.

برای بدست آوردن مدار توصیف شده با یک جدول روند، لازم است به هر حالت مقدار دودویی منحصر به فردی نسبت دهیم. این تخصیص، جدول روند را به جدول گذر تبدیل خواهد کرد و از آن می‌توان نمودار منطقی را بدست آورد. این کار برای جدول روند شکل ۴-۹ (ب) در شکل ۵-۹ نشان داده شده است، 0 دودویی را به حالت a، 1 دودویی را به حالت b نسبت می‌دهیم. نتیجه این کار جدول گذر شکل ۵-۹ (الف) خواهد بود. نقشه خروجی شکل ۵-۹ (ب) مستقیماً از مقادیر خروجی در جدول روند فراهم شده است. تابع تحریک Y و تابع خروجی z به وسیله دو نقشه ساده شده‌اند. نمودار منطقی هم در شکل ۵-۹ (پ) ملاحظه می‌گردد.

این مثال روال تهیه نمودار منطقی را از یک جدول روند نشان می‌دهد. البته نباید فراموش کرد که روال کار همیشه به این سادگی نیست. در تخصیص مقادیر حالت و خروجی‌های حالات ناپایدار مشکلات متعددی وجود دارد. این مشکلات به تفصیل در بخش‌هایی که به دنبال می‌آیند بحث شده‌اند.

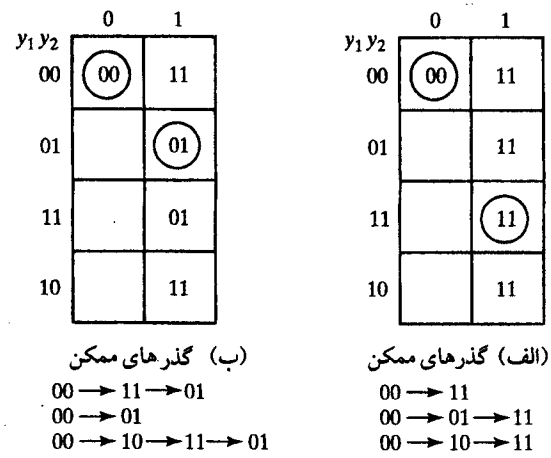


شکل ۵-۹. تعیین مدار مشخص شده بوسیله جدول روند شکل (۴-۹ ب)

وضعیت‌های رقابتی

در یک مدار ترتیبی غیرهمزمان اگر دو یا چند متغیر حالت دودویی در پاسخ به یک متغیر در یک ورودی تغییر کنند، یک وضعیت رقابتی وجود خواهد داشت. هنگامی که مدار با تأخیرهای نامساوی مواجه شود، وضعیت رقابتی موجب می‌گردد تا متغیرهای حالت به طور ناخواسته‌ای تغییر کنند. مثلاً اگر قرار باشد متغیرهای حالت از 00 به 11 تغییر نمایند، تفاوت تأخیرها ممکن است سبب شود تا متغیر اول زودتر از دومی تغییر کند و در نتیجه متغیرهای حالت از 00 به 10 و سپس به 11 بروند. اگر متغیر حالت دوم سریع‌تر از اولی تغییر نماید متغیرهای حالت از 00 به 01 و سپس به 11 تغییر خواهند نمود. بنابراین می‌توان نحوه تغییر متغیرهای حالت را از قبل پیش‌بینی کرد. اگر حالت پایدار نهایی که مدار به آن می‌رسد، مستقل از ترتیب متغیرهای حالت باشد، رقابت مذکور را رقابت غیربحرانی نامند. ولی اگر حالات پایدار نهایی دو یا چند حالت متفاوت باشند و این تفاوت به ترتیب تغییر متغیرهای حالت وابسته باشد، در این صورت رقابت بحرانی خواهد بود. اگر عملیات صحیحی از مدار موردنظر باشد باید از رقابت‌های بحرانی پرهیز شود.

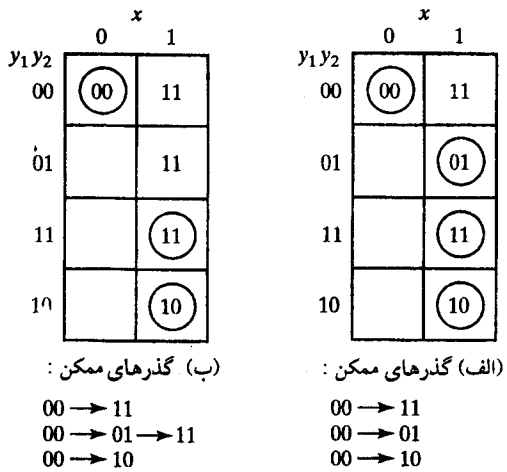
دو مثال شکل ۶-۹ رقابت‌های غیربحرانی را تشریح می‌کنند. ما با حالت پایدار کل $y_1y_2\ x = 000$ آغاز می‌کنیم و سپس ورودی را از 0 به 1 تغییر می‌دهیم. متغیرهای حالت باید از 00 به 11 تغییر کنند که گویای وجود یک وضعیت رقابتی است. گذرهای لیست شده در زیر هر جدول سه امکان تغییر



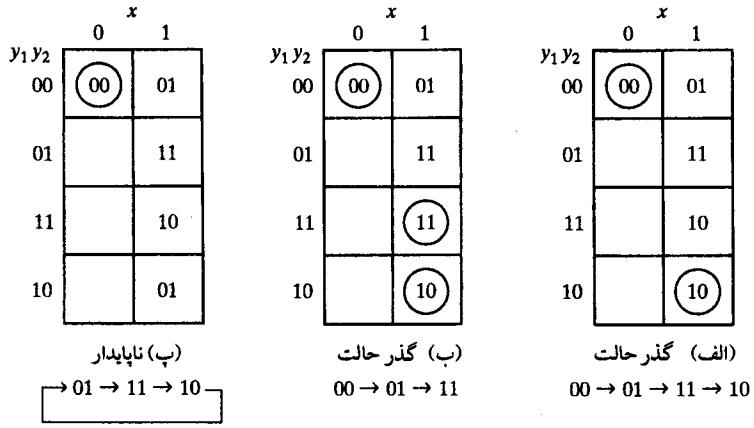
شکل ۶-۹ . مثالهایی از رقابتهای غیربحرانی

متغیرهای حالت را نشان می‌دهند. این گذرها یا انتقال‌ها می‌توانند به طور همزمان از 00 به 11 بروند و یا به ترتیبی از 00 به 10 و سپس به 11 بروند. در هر صورت، حالت پایدار نهایی یکسان و از نوع وضعیت رقابتی غیربحرانی است. در (الف) حالت کلی نهایی y_1y_2 x = 111 و در (ب) برابر با 011 می‌باشد.

جدول گذر شکل ۷-۹ رقابت‌های بحرانی را نشان می‌دهند. ما در اینجا هم با حالت کلی y_1y_2 x = 000 آغاز می‌کنیم و سپس ورودی را از 0 به 1 تغییر خواهیم داد. متغیرهای حالت باید از 00 به 11 تغییر نمایند. اگر آنها به طور همزمان عوض شوند، حالت پایدار کل نهایی 111 خواهد بود. در جدول گذر (الف)، اگر به دلیل تأخیر انتشار نابرابر Y_2 قبل از Y_1 عوض شود، آنگاه مدار به حالت پایدار کل 011 رفته و در آن باقی می‌ماند. از طرف دیگر، اگر ابتدا Y_1 تغییر کند حالت داخلی 10 شده و مدار در حالت پایدار کل 101 باقی خواهد ماند. از این رو، رقابت بحرانی است زیرا مدار، بسته به ترتیب تغییر



شکل ۷-۹ . مثالهایی از رقابتهای بحرانی



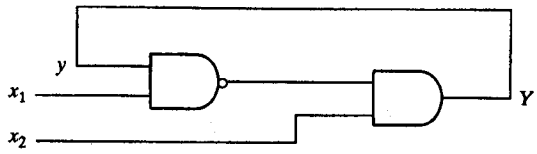
شکل ۸-۹ . مثالهای از سیکل

متغیرهای حالت به حالات پایدار متفاوتی می‌رود. جدول گذر شکل ۷-۹ (ب) یک نمونه رقابت بحرانی دیگری را نشان می‌دهد، که در آن دو گذر ممکن یک حالت پایدار کلی را نتیجه می‌دهند، ولی گذر سوم به حالت کلی دیگری می‌رود. با انتساب صحیح دودویی به متغیرهای حالت می‌توان از وضعیت رقابتی پرهیز کرد. اعداد دودویی باید طوری به متغیرهای حالت تخصیص یابند که در هنگام یک گذر در جدول حالت هر بار تنها یک متغیر حالت عوض شود. موضوع تخصیص حالت بدون رقابت در بخش ۶-۹ مورد بحث قرار گرفته است.

با هدایت مدار به حالات بی‌ثبات میانی و با یک تغییر حالت منحصر می‌توان از رقابت اجتناب کرد. وقتی که مدار از میان یک رشته حالات بی‌ثبات عبور می‌کند، گوییم مدار دارای چرخه یا سیکل است. شکل ۸-۹ وقوع چرخه‌ها را نشان می‌دهد. در اینجا هم با $y_1y_2 = 00$ آغاز می‌کنیم و سپس ورودی را از 0 به 1 تغییر می‌دهیم. جدول گذر بخش (الف) دنباله منحصر به فردی را نشان می‌دهد که به حالت کلی 101 ختم می‌گردد. جدول (پ) نشان می‌دهد که هر چند متغیرهای حالت از 00 به 11 تغییر می‌کنند، ولی گذر منحصر به فردی را از 00 به 01 و سپس به 11 فراهم می‌نمایند. وقتی که از چرخه برای یک مدار استفاده می‌شود باید دقت کرد تا به یک حالت پایدار منتهی گردد. اگر یک چرخه به یک حالت پایدار ختم نشود، مدار از یک حالت ناپایدار به حالت ناپایدار دیگری رفته و کل سیستم ناپایدار خواهد بود. این مطلب در شکل ۸-۹ (پ) و نیز مثال زیر نشان داده شده است.

ملاحظات مربوط به پایداری

به دلیل وجود پسخورد موجود در مدارهای ترتیبی غیرهمزمان، باید مراقب بود تا مدار ناپایدار نشود. یک وضعیت ناپایدار موجب می‌گردد تا مدار بین حالات ناپایدار نوسان کند. روش تحلیل جدول گذر می‌تواند در تشخیص وقوع بی‌ثباتی مفید باشد.



(الف) نمودار منطقی

y \ x_1x_2	00	01	11	10
0	(0)	1	1	(0)
1	0	(1)	0	0

(ب) جدول گذر

شکل ۹-۹. مثالی از مدار ناپایدار

به عنوان مثال مدار شکل ۹-۹ (الف) را ملاحظه کنید. تابع تحریک برابر است با

$$Y = (x_1y)'x_2 = (x_1' + y')x_2 = x_1'x_2 + x_2y'$$

جدول گذر برای مدار در شکل ۹-۹ (ب) نشان داده شده است. مقادیری از Y که با y برابر است در دایره محصور شده و حالات پایدار را نشان می‌دهند. وارده‌های محصور نشده وضعیت‌های ناپایدارند. توجه کنید که ستون 11 دارای حالت پایداری نیست. این بدان معنی است که با تثبیت x_1x_2 در 11، مقادیر Y و y هرگز یکسان نمی‌شوند. اگر $y = 0$ باشد آنگاه $Y = 1$ می‌شود، که این خود نیز موجب گذر به سطر دوم جدول با $y = 1$ و $Y = 0$ خواهد شد. این وضعیت مجدداً بازگشت به سطر اول را سبب می‌گردد. در نتیجه تا وقتی که ورودی 11 باشد متغیر حالت به میزان نامتناهی بین 0 و 1 نوسان خواهد کرد.

وضعیت ناپایدار را می‌توان مستقیماً از نمودار منطقی شناسایی کرد. اجازه بدهید تا $x_1 = 1$، $x_2 = 1$ و $y = 1$ باشد. خروجی گیت NAND برابر 0، خروجی گیت AND برابر 0 و بنابراین $Y = 0$ خواهد شد و در نتیجه $Y \neq y$ است. اگر $y = 0$ باشد، خروجی گیت NAND برابر 1 و خروجی گیت AND برابر 1 می‌گردد و بنابراین $Y \neq y$ خواهد شد. اگر فرض کنیم که هر گیت دارای تأخیر در انتشار 5ns باشد (همراه با تأخیر سیم‌ها) در می‌یابیم که Y در 10ns برابر 0 و در 10ns بعد در 1 قرار خواهد داشت. این وضعیت باعث تولید یک موج مربعی با پریود 20ns خواهد شد. فرکانس نوسان عکس پریود بوده و برابر 50MHz می‌گردد. ناپایداری در مدارهای ترتیبی غیرهمزمان نامطلوب است و باید از آن پرهیز کرد، مگر این که مدار به منظور تولید موج مربعی طراحی شود.

۹-۳ مدارهای لچ‌دار

به لحاظ تسلسل، مدارهای ترتیبی غیرهمزمان قبل از مدارهای همزمان شناخته شده و مورد استفاده بوده‌اند. اولین مدار دیجیتال کاربردی با رله‌ها که با عملیات غیرهمزمان سازگارترند، ساخته شد. به این

دلیل روش سنتی آرایش مدار غیرهمزمان با قطعاتی انجام می‌شد تا به هم متصل و یک یا چند مدار پسخورد را به وجود آورند. هنگامی که مدارهای دیجیتال با اجزاء الکترونیک ساخته می‌شوند، استفاده از لچ SR (بخش ۲-۵) به عنوان عنصر حافظه آسان‌تر است. به کارگیری لچ‌های SR در مدارهای ترتیبی الگوی منظمی در نمودارهای منطقی ایجاد می‌نماید که در آن بخش حافظه کلاً قابل رویت است. در این بخش عملکرد لچ SR با تکنیک بخش قبل تحلیل خواهد شد. آنگاه روالی را برای پیاده‌سازی مدارهای ترتیبی غیرهمزمان با لچ SR نشان خواهیم داد.

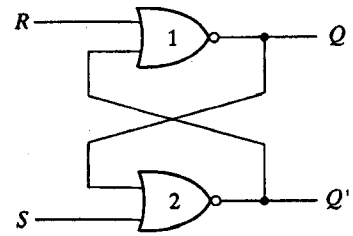
لچ SR

لچ SR مداری دیجیتال با دو ورودی S و R و دو گیت NOR متقاطع یا دو گیت NAND متقاطع است. مدار ساخته شده با گیت NOR در شکل ۱۰-۹ دیده می‌شود. این مدار و جدول درستی‌اش از شکل ۳-۵ بدست آمده است. به منظور تحلیل مدار با استفاده از روش جدول گذر، آن را دوباره در شکل ۱۰-۹ (پ) ترسیم کرده‌ایم تا مسیر پسخورد از خروجی گیت 1 به ورودی گیت 2 ملاحظه شود. خروجی Q معادل با متغیر تحریک Y و متغیر ثانویه y است. تابع بول خروجی برابر است با

$$Y = [(S + y)' + R]' = (S + y)R' = SR' + R'y$$

با تهیه جدول Y طبق شکل ۱۰-۹ (ت) جدول گذر را برای مدار بدست خواهیم آورد.

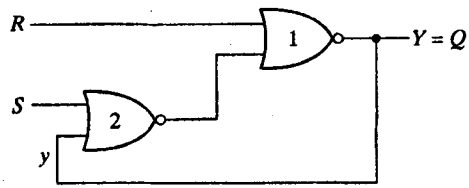
اکنون می‌توانیم رفتار لچ SR را با استفاده از جدول گذر بررسی کنیم. با SR = 10، خروجی



(الف) مدار با کوپل متقاطع

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(پس از SR = 10)
0	1	0	1	
0	0	0	1	(پس از SR = 01)
1	1	0	0	

(ب) جدول درستی



(پ) مدار با نمایش فیدبک

y \ SR	00	01	11	10
0	(0)	(0)	(0)	1
1	(1)	0	0	(1)

$Y = SR' + R'y$
$Y = S + R'y$ when $SR = 0$

(ت) جدول گذر

شکل ۱۰-۹. لچ SR با گیتهای NOR

$Q = Y = 1$ خواهد شد و در این حال گوییم لچ به 1 نشانده شده است. اگر S را به 0 بازگردانیم مدار در همان حال باقی می‌ماند. با $SR = 01$، خروجی $Q = Y = 0$ شده و در این حال گوییم لچ به 0 بازنشانی شده است. با تغییر R به 0 مدار در همان حالت بازنشانی باقی خواهد ماند. این حالات در جدول درستی هم لیست شده‌اند. وقتی که S و R هر دو در 1 باشند مدار مشکل خواهد داشت. با توجه به جدول درستی ملاحظه می‌شود که هر دو خروجی Q و Q' برابر 0اند. این حالت قانون متمم بودن این دو خروجی را زیر سؤال می‌برد. به علاوه با توجه به جدول گذر می‌بینیم که رفتن از $SR = 11$ به $SR = 00$ نتیجه غیرقابل پیش‌بینی را تولید می‌کند. اگر S ابتدا به 0 برود، خروجی در 0 باقی می‌ماند ولی اگر R زودتر به 0 برود خروجی به 1 خواهد رفت. در یک عملکرد عادی باید مطمئن باشیم که مقدار 1 به طور همزمان به S و R اعمال نمی‌شود. این شرط را می‌توانیم با تابع بولی $SR = 0$ نشان دهیم که بیان می‌دارد AND (ضرب منطقی) S و R همیشه باید 0 باشد.

اگر به تابع تحریک بازگردیم، در می‌یابیم که اگر عبارت بولی SR' و SR را OR کنیم نتیجه تک متغیر S خواهد بود.

$$SR' + SR = S(R' + R) = S$$

از این رابطه نتیجه می‌شود که اگر $SR = 0$ باشد $S = SR'$ خواهد بود. بنابراین تابع تحریک قبلی حاصل

$$Y = SR' + R'y$$

را می‌توان به صورت زیر بیان کرد

$$Y = S + R'y \qquad \text{وقتی } SR = 0$$

برای تحلیل یک مدار با لچ SR، ابتدا باید شرط بولی $SR = 0$ را چک کنیم. آنگاه تابع تحریک کاهش یافته را برای تحلیل مدار به کار خواهیم برد. با این وجود، اگر S و R را به طور همزمان برابر 1 بیابیم، آنگاه لازم است از همان تابع تحریک اولیه و ساده نشده استفاده کنیم.

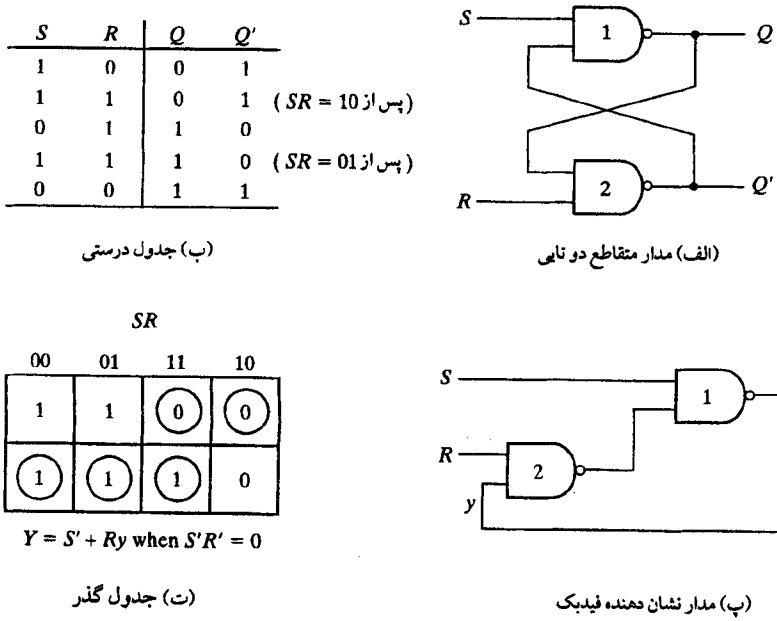
تحلیل لچ SR با گیت‌های NAND در شکل ۱۱-۹ انجام شده است. ورودی‌های لچ SR به طور معمول در 1 هستند، مگر این که حالت لچ را بخواهیم عوض کنیم. اِعمال 0 به R موجب می‌شود تا Q به 0 برود و به این ترتیب لچ در حالت بازنشانی قرار خواهد گرفت. پس از بازگشت R به 1، تغییر S به 0 سبب نشاندن لچ خواهد شد. حالتی که در اینجا باید از آن پرهیز کرد 0 شدن همزمان S و R است. ممانعت از این حالت با $S'R' = 0$ میسر است. تابع تحریک برای مدار عبارت است از

$$Y = [S(Ry)']' = S' + Ry$$

از مقایسه این رابطه با تابع تحریک لچ NOR ملاحظه می‌کنیم که S با S' و R' با R جایگزین شده است. از این رو متغیر ورودی برای لچ NAND، متمم ورودی‌های لچ NOR می‌باشند. به این دلیل لچ NAND را گاهی لچ $S'R'$ هم می‌خوانند (یا لچ $\bar{S}-\bar{R}$).

مثال تحلیل

مدارهای ترتیبی غیرهمزمان را با لچ‌های SR، با یا بدون مسیرهای پسخورد بیرونی هم می‌توان



S	R	Q	Q′	
1	0	0	1	
1	1	0	1	(پس از SR = 10)
0	1	1	0	
1	1	1	0	(پس از SR = 01)
0	0	1	1	

(ب) جدول درستی

(الف) مدار متقاطع دو تایی

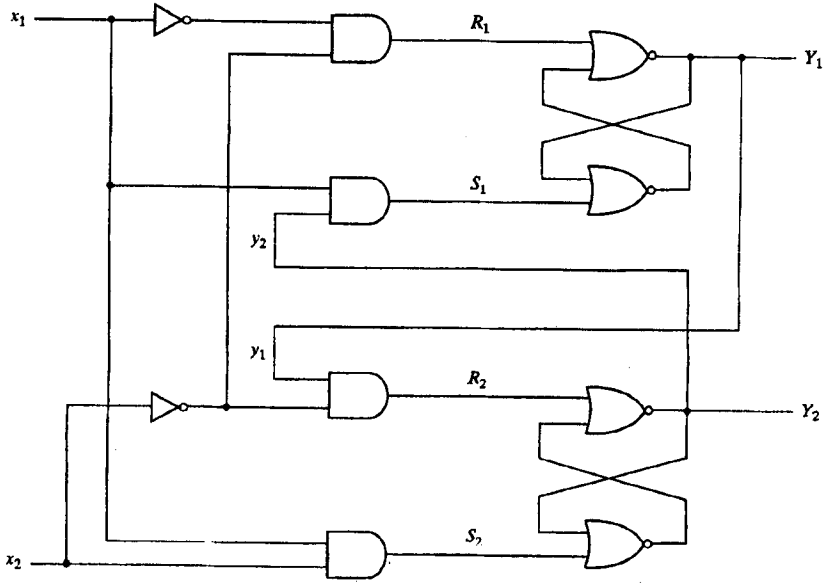
(ت) جدول گذر

(پ) مدار نشان دهنده فیدبک

شکل ۱۱-۹ . لچ SR با گیتهای NAND

ساخت. البته، همیشه مسیر حلقه پسخوردی در خود لچ وجود دارد. تحلیل مداری که با لچ ساخته شده باشد با مثالی نشان داده خواهد شد. با توجه به این مثال، می‌توان روال را تعمیم داده قدم‌های مؤثر بعدی را برای طراحی دیگر مدارهای مشابه برداشت.

مدار شکل ۱۲-۹ دارای دو لچ SR با خروجی‌های Y_1 و Y_2 است. دو ورودی x_1 و x_2 و دو حلقه



شکل ۱۲-۹ . مثالی از یک مدار با لچهای SR

پسخورد بیرونی وجود دارند که متغیرهای ثانویه y_1، y_2 از آنها اخذ شده‌اند. توجه دارید که این مدار شبیه به یک مدار ترتیبی معمولی است که در آن لچ‌ها همچون فیلپ فلاپ‌های بدون ساعت عمل می‌کنند. برای تحلیل مدار، ابتدا توابع بول را برای ورودی‌های S و R هر لچ بدست می‌آوریم.

$$S_1 = x_1y_2 \qquad S_2 = x_1x_2$$
$$R_1 = x_1'x_2' \qquad R_2 = x_2'y_1$$

آنگاه شرط $SR = 0$ را برای اطمینان از عملکرد مدار بررسی می‌کنیم:

$$S_1R_1 = x_1y_2x_1'x_2' = 0$$
$$S_2R_2 = x_1x_2x_2'y_1 = 0$$

نتیجه 0 است زیرا $x_1x'_1 = x_2x'_2 = 0$ می‌باشد.

قدم بعدی بدست آوردن جدول گذر مدار است. به یاد دارید که جدول گذر، مقدار Y را به صورت تابعی از y و x بدست می‌دهد. توابع تحریک از رابطه $Y = S + R'y$ حاصل می‌گردند.

$$Y_1 = S_1 + R_1'y_1 = x_1y_2 + (x_1 + x_2)y_1 = x_1y_2 + x_1y_1 + x_2y_1$$
$$Y_2 = S_2 + R_2'y_2 = x_1x_2 + (x_2 + y_1')y_2 = x_1x_2 + x_2y_2 + y_1'y_2$$

اکنون یک نقشه مرکب برای $Y = Y_1Y_2$ بدست می‌آوریم. همانطور که در شکل ۱۳-۹ ملاحظه می‌شود متغیرهای y به سطرهای نقشه و متغیر x به ستون‌ها تخصیص یافته‌اند. توابع بول Y_1 و Y_2 که در فوق بیان شدند برای ترسیم نقشه مرکب استفاده می‌شوند. در هر سطر وارده‌هایی از Y که دارای مقداری برابر y هستند در دایره محصور شده و حالات پایداری را نمایش می‌دهند. با بررسی جدول گذر، نتیجه می‌گیریم که مدار با ثبات است. اگر مدار در آغاز در $y_1y_2\ x_1x_2 = 1101$ باشد و x_2 از 1 به 0 تغییر کند، یک وضعیت رقابت بحرانی به وجود خواهد آمد. اگر Y_1 قبل از Y_2 به 0 برود، مدار به جای 0000 به 0100 خواهد رفت. با این وجود، با تأخیرهای تقریباً مساوی در گیت‌ها و لچ‌ها، به نظر نمی‌رسد که چنین وضعیت نامطلوبی رخ دهد.

روال تحلیل یک مدار ترتیبی غیرهمزمان با لچ‌های SR را می‌توان به صورت زیر خلاصه کرد.

y_1y_2 \ x_1x_2	00	01	11	10
00	(00)	(00)	01	(00)
01	(01)	(01)	11	11
11	00	(11)	(11)	10
10	00	(10)	11	(10)

شکل ۱۳-۹. جدول گذر مدار (۱۲-۹)

۱- خروجی هر لچ را با Y_i و مسیر فیدبک خارجی‌اش را (اگر وجود داشته باشد) با y_i مشخص کنید که در آن $i = 1, 2, ..., k$ می‌باشد.
۲- توابع بولی را برای ورودی‌های S_i و R_i هر لچ بدست آورید.
۳- برای هر لچ NOR، $SR = 0$ و برای هر لچ NAND، $S'R' = 0$ را چک کنید. اگر این شرایط صادق نبودند، ممکن است مدار به درستی کار نکند.
۴- رابطه $Y = S + R'y$ را برای هر لچ NOR، یا $Y = S' + Ry$ را برای هر لچ NAND ارزیابی کنید.
۵- نقشه‌ای برای y‌ها در سطر و ورودی‌های x در ستون بسازید.
۶- مقدار $Y = Y_1Y_2, ..., Y_k$ را در نقشه پیاده کنید.
۷- همه حالات با ثبات را که در آن $Y = y$ است در دایره محصور نمایید. نقشه حاصل همان جدول گذر است.

جدول تحریک لچ

جدول گذر لچ SR برای تحلیل و تعریف عملکرد لچ بسیار مفید است. این جدول وقتی متغیر ثانویه y و ورودی‌های S و R معلومند متغیر Y را مشخص می‌کند. در حین روند پیاده‌سازی، جدول گذر مدار معلوم می‌باشد و ما علاقمند به یافتن مقادیر S و R هستیم. به این دلیل به جدولی نیاز داریم تا ورودی‌های S و R را برای هر گذر ممکن از y به Y لیست نماید. چنین سیستمی را **جدول تحریک** گوییم.

جدول تحریک لچ SR در شکل ۱۴-۹ (ب) ملاحظه می‌شود. دو ستون اول چهار گذر ممکن از y به Y را نشان می‌دهد. دو ستون بعدی مقادیر ورودی لازمی را مشخص می‌کنند که گذر را موجب می‌شوند. مثلاً برای گذر از $y = 0$ به $Y = 1$، لازم است ورودی $S = 1$ و ورودی $R = 0$ باشد. این مطلب در سطر دوم جدول گذر نشان داده شده است.

شرایط ورودی لازم برای هر چهار گذر در جدول تحریک را مستقیماً از جدول گذر لچ شکل ۱۴-۹ (ت) و پس از حذف حالت ناپایدار $SR = 11$ می‌توان بدست آورد. مثلاً، برای تغییر از $y = 0$ به $Y = 0$، جدول گذر نشان می‌دهد که SR می‌تواند 00 یا 01 باشد. این بدان معنی است که باید $S = 1$ ولی R می‌تواند 0 یا 1 باشد. بنابراین اولین سطر در جدول تحریک نشان می‌دهد که $S = 0$ و $R = X$ است که X حالت بی‌اهمیت بوده و می‌تواند 0 یا 1 باشد.

مثال پیاده‌سازی

پیاده‌سازی یک مدار ترتیبی با لچ‌های SR روشی برای تهیه نمودار منطقی از یک جدول گذر می‌باشد. در این روش لازم است تا تابع بول را برای ورودی‌های S و R هر لچ معین نماییم. سپس نمودار منطقی با رسم لچ‌های SR و گیت‌های منطقی که توابع S و R را پیاده‌سازی می‌کنند، بدست می‌آید. برای تشریح این روش، پیاده‌سازی مثال شکل (۵-۹) را تکرار خواهیم کرد. مدار خروجی تغییر نمی‌کند و بنابراین دوباره تکرار نمی‌شود.

y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	1

(ب) جدول تحریک لچ

x_1x_2

y	00	01	11	10
0	(0)	(0)	(0)	1
1	0	0	(1)	(1)

(الف) جدول گذر

$Y = x_1x'_2 + x_1y$

x_1x_2

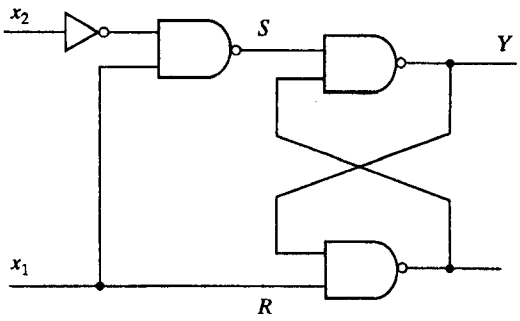
y	00	01	11	10
0	X	X	X	0
1	1	1	0	0

(ت) نقشه برای $R = x'_1$

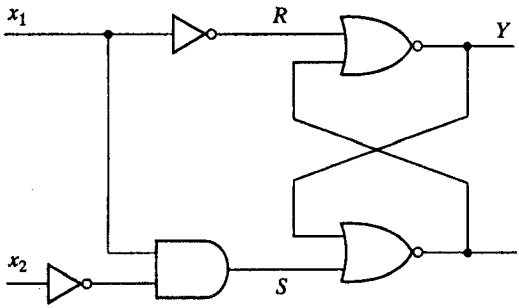
x_1x_2

y	00	01	11	10
0	0	0	0	1
1	0	0	X	X

(پ) نقشه برای $S = x_1x'_2$



(ج) مدار لچ با NAND



(ث) مدار لچ با NOR

شکل ۱۴-۹ . بدست آوردن مداری با لچ، از یک جدول انتقال

جدول گذر شکل ۵-۹ (الف) مجدداً در شکل ۱۴-۹ (الف) آمده است. با توجه به اطلاعات مفروض در جدول گذر و شرایط موجود در جدول تحریک لچ در شکل ۱۴-۹ (پ)، می‌توانیم نقشه‌های ورودی‌های S و R لچ را طبق شکل ۱۴-۹ (پ) و (ت) بدست آوریم. مثلاً، مربع سطر دوم و ستون سوم ($yx_1x_2 = 111$) در شکل ۱۴-۹ (الف) به گذر $y = 1$ به $Y = 1$ نیاز دارد. جدول تحریک برای این تغییر $S = X$ و $R = 0$ را مشخص می‌کند. بنابراین مربع مربوطه در نقشه S با X و در نقشه R با 0 علامت زده شده است. دیگر مربع‌ها همگی به همین روش با مقادیر مربوطه پر شده‌اند. آنگاه نقشه‌ها برای بدست آوردن توابع بول ساده شده به کار می‌روند.

$$S = x_1x'_2 \qquad \text{و} \qquad R = x'_1$$

نمودار منطقی از یک لچ SR و گیت‌های مربوطه برای پیاده‌سازی توابع بولی S و R تشکیل شده است. وقتی که لچ NOR به کار رود مدار مطابق شکل ۱۴-۹ (ث) است. با لچ NAND باید مقادیر متمم شده را برای S و R به کار ببریم.

$$S = (x_1 x_2')' \qquad \text{و} \qquad R = x_1$$

این مدار در شکل ۱۴-۹ (ج) نشان داده شده است.

اکنون می‌توان روش کلی پیاده‌سازی مدار با لچ‌های SR با استفاده از یک جدول گذر را به صورت زیر خلاصه کرد.

۱- با فرض داشتن جدول گذر که تابع تحریک $Y = Y_1 Y_2, \ldots, Y_k$ را مشخص کند، جفت نقشه‌هایی برای S_i و R_i با $i = 1, 2, \ldots, k$ بدست آورید. این کار با استفاده از شرایط مشخص شده در جدول تحریک شکل ۱۴-۹ (پ) انجام می‌شود.

۲- توابع بول ساده شده را برای هر S_i و R_i بدست آورید. دقت کنید که S_i و R_i در مربع‌های متناظر برابر 1 نشوند.

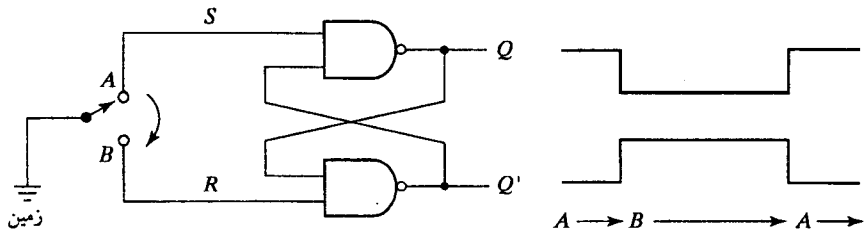
۳- نمودار منطقی را با استفاده از k لچ همراه با گیت‌های لازم برای تولید توابع بول S و R رسم کنید. برای لچ‌های NOR، از توابع بول S و R بدست آمده در مرحله ۲ استفاده کنید.

مثال مفید دیگری از پیاده‌سازی لچ را می‌توان در بخش ۷-۹ در ارتباط با شکل ۳۸-۹ بدست آورد.

مدار نوسان‌گیر

اطلاعات دودویی ورودی در یک سیستم دیجیتال را می‌توان بصورت دستی با سوئیچ‌های مکانیکی، تولید کرد. یک وضعیت سوئیچ، ولتاژی معادل با منطق 1، و دیگری ولتاژ دوم معادل منطق 0 را فراهم می‌سازد. از سوئیچ‌های مکانیکی برای شروع، ختم یا بازنشانی سیستم دیجیتال هم استفاده می‌شود. هنگام تست مدارهای دیجیتال در آزمایشگاه، سیگنال‌های ورودی معمولاً از سوئیچ‌ها می‌آیند. خصیصه معمول یک سوئیچ مکانیکی این است که وقتی اهرم آن از یک سو به سوی دیگر پرتاب می‌شود، اتصال سوئیچ قبل از استقرار نهایی، چندین بار نوسان می‌کند. در نوعی از یک سوئیچ ممکن است این نوسانات چندین میلی ثانیه طول بکشند. نوسانات مذکور ممکن است به دفعات موجب تغییر سیگنال منطقی بین 1 و 0 گردند.

مدار نوسان‌گیر، وظیفه حذف رشته پالس‌هایی را به عهده دارد که از نوسان اتصال مکانیکی حاصل می‌شود و بدین وسیله یک گذر یکنواختی برای سیگنال از 0 به 1 و از 1 به 0 فراهم می‌سازد. یک چنین مداری، طبق شکل ۱۵-۹، از یک سوئیچ دو حالته یک قطبی متصل به لچ SR تشکیل شده است. مرکز اتصال به زمین وصل است تا سیگنالی معادل 0 منطقی تولید کند. وقتی یکی از دو اتصال A یا B از طریق سوئیچ به زمین وصل نشده باشد یک سیگنال 1 منطقی تولید می‌کند. گاهی هر یک از این اتصالات از طریق یک مقاومت به یک ولتاژ ثابت متصل می‌شوند تا بدین وسیله یک سیگنال 1 منطقی ثابت و مناسبی تولید گردد. وقتی که سوئیچ، مطابق شکل از حالت A به حالت B پرتاب و سپس باز



شکل ۱۵-۹. مدار نوسان گیر

می‌گردد، لچ یک تک پالس منفی برای Q و مثبت برای Q' تولید می‌نماید. سوئیچ معمولاً از نوع فشاری است که حالت ثابت آن در وضعیت A قرار دارد. هنگام فشردن کلید، اتصال به B رفته و وقتی رها شود به وضعیت A باز می‌گردد.

عملکرد مدار نوسان‌گیر مطابق زیر است. هنگام اتصال سوئیچ به نقطه A، ما شرایط $S = 0$، $R = 1$ و $Q = 1$ و $Q' = 0$ را داریم، شکل ۱۱-۹ (ب). هنگام اتصال آن به نقطه B، اتصال به زمین سبب رفتن R به 0 و S به 1 می‌گردد زیرا اتصال A باز است. این وضعیت سبب رفتن خروجی Q به 0 و Q' به 1 می‌شود. سوئیچ پس از اولین اتصال به B چندین بار نوسان می‌کند، ولی باید به منظور داشتن یک عملکرد صحیح تصور کنیم که این بازگشت آنقدرها نیست تا به نقطه A برسد. خروجی لچ به علت باقی ماندن Q' در 1 (و Q در 0) بی‌تغییر خواهد ماند زیرا در ازاء 0 یا 1 بودن R (اتصال با زمین و عدم اتصال با زمین) Q' در 1 باقی می‌ماند. وقتی که سوئیچ به وضعیت A بازگردد، S برابر 0 و Q به 1 برمی‌گردد. مجدداً خروجی یک گذر صافی را علاوه بر وجود نوسان در وضعیت A، از خود به نمایش می‌گذارد.

۹-۴ روش طراحی

طراحی یک مدار غیر همزمان با صورت مسئله آغاز و با نمودار منطقی پایان می‌یابد. برای بهینه‌سازی پیچیدگی مدار و تولید یک مدار با ثبات و بدون رقابت تعدادی مراحل طراحی وجود دارد. بطور خلاصه مراحل طراحی بقرار زیرند:

از مشخصات طراحی، یک جدول روند اصلی بدست می‌آوریم. این جدول به حداقل حالات کاهش می‌یابد. آنگاه به حالات مقادیری دودویی تخصیص یافته و از آن جدول گذر بدست می‌آید. بالاخره از جدول گذر، نمودار منطقی مدار ترکیبی همراه با پسخوردی که در آن لچ‌های SR قرار دارند حاصل می‌گردد. فرآیند طراحی با ارائه یک مثال نمایش داده می‌شود. پس از حل مثال، درک مراحل طراحی ساده‌تر خواهد بود. بعضی از مراحل روش‌های مستدل‌تری را لازم دارند که در بخش‌های بعد مفصلاً تشریح شده است.

مثال طراحی

می‌خواهیم یک مدار لچ گیت را با دو ورودی G و D و یک خروجی Q بسازیم. اطلاعات دودویی حاضر در ورودی D وقتی که G برابر 1 باشد به خروجی Q منتقل می‌گردد. مادامی که $G=1$ است

خروجی Q ورودی D را دنبال خواهد کرد. وقتی که D به 0 برود اطلاعات موجود در ورودی D پس از گذر ساعت در خروجی Q باقی می‌ماند. لچ گیت‌دار عنصر حافظه‌ای است که مقدار D را وقتی $G = 1$ است پذیرفته و این مقدار را پس از رفتن G به 0 حفظ می‌نماید. به محض $G = 0$، هر تغییر در D مقدار خروجی Q را عوض نمی‌کند.

جدول روند اصلی

همانطور که قبلاً تعریف شد، یک جدول روند اصلی جدولی است با تنها یک حالت کلی پایدار در هر سطر. به یاد می‌آورید که یک حالت کل از ترکیب حالت داخلی با ورودی ایجاد می‌شود. اگر ابتدا جدولی را با همه حالت‌های کلی سیستم تشکیل دهیم، بدست آوردن جدول روند اصلی ساده می‌گردد. این کار برای لچ گیت‌دار در شکل ۲-۹ نشان داده شده است. هر سطر از جدول بیانگر یک حالت کلی است، که متشکل از یک حرف برای بیان حالت داخلی و ترکیبی از ورودی برای D و G است. خروجی Q هم برای هر حالت کلی نشان داده شده است. ما با دو حالت کلی که $G = 1$ دارند آغاز می‌کنیم. با توجه به ویژگی‌های طراحی، می‌دانیم که اگر $DG = 01$ باشد $Q = 0$ و اگر $DG = 11$ باشد، $Q = 1$ است زیرا با $G = 1$، D باید با Q برابر باشد. ما این حالات را به a و b تخصیص می‌دهیم. وقتی که G به 0 برود، خروجی به آخرین مقدار D بستگی دارد. بنابراین اگر، گذر DG از 01 به 00 و به 10 رخ دهد، آنگاه Q باید در 0 باقی بماند زیرا در زمان گذر G از 1 به 0، D برابر 0 است. اگر گذر DG از 11 به 10 به 00 باشد، آنگاه Q در 1 باقی خواهد ماند. این اطلاعات شش حالت کلی را مطابق جدول نتیجه می‌دهد. توجه کنید که گذر همزمان دو متغیر ورودی، مانند گذر از 01 به 10 یا از 11 به 00 دو عملکرد حالت اساسی مجاز نیست.

جدول روند اصلی برای لچ گیتی در شکل ۱۶-۹ دیده می‌شود. این جدول برای هر حالت دارای یک سطر و برای ترکیب ورودی یک ستون دارد. ابتدا یک مربع در هر سطر متعلق به حالت پایدار در آن سطر

	DG = 00	01	11	10
a	*c*, –	(*a*), 0	*b*, –	–, –
b	–, –	*a*, –	(*b*), 1	*e*, –
c	(*c*), 0	*a*, –	–, –	*d*, –
d	*c*, –	–, –	*b*, –	(*d*), 0
e	*f*, –	–, –	*b*, –	(*e*), 1
f	(*f*), 1	*a*, –	–, –	*e*, –

شکل ۱۶-۹. جدول روند اصلی

جدول ۲-۹ . حالتهای کلی لچ گیت‌دار

حالت	ورودی		خروجی	توضیحات
	D	G	Q	
a	0	1	0	$G = 1$ زیرا $D = Q$
b	1	1	1	$G = 1$ زیرا $D = Q$
c	0	0	0	پس از حالت a یا d
d	1	0	0	پس از حالت c
e	1	0	1	پس از حالت b یا f
f	0	0	1	پس از حالت e

را پر می‌کنیم. این وارده‌ها از جدول ۲-۹ معین می‌شوند. مثلاً وقتی ورودی 01 باشد حالت a با ثبات و خروجی 0 است. این اطلاعات در جدول روند، سطر اول ستون دوم وارد می‌شود. به طور مشابه، پنج حالت دیگر، همراه با خروجی آنها در ستون‌های مربوطه وارد می‌گردند.

سپس، چون تغییر همزمان دو ورودی مجاز نیست، می‌توانیم در سطری که نسبت به متغیرهای ورودی حالت پایدار در دو یا بیشتر متغیر اختلاف دارد خط تیره بگذاریم. مثلاً، اولین سطر در جدول روند حالت پایداری با ورودی 01 را نشان می‌دهد. چون در هر لحظه از زمان تنها یک متغیر مجاز به تغییر است، ورودی می‌تواند به 00 یا 11 تغییر نماید. بنابراین در سطر a ستون 10 دو خط تیره وارد می‌کنیم. این کار نهایتاً به معنی حالت بی‌اهمیت برای حالت بعدی در این مربع است. به دنبال این عمل، دومین مربع در هر سطر جدول روند پر خواهد شد.

آنگاه باید در هر سطر دو مربع دیگر را تعیین تکلیف کنیم. اظهار نظر موجود در جدول ۲-۹ می‌تواند در بدست آوردن اطلاعات لازم به ما کمک کند. مثلاً حالت c مربوط به ورودی 00 است و پس از یک تغییر ورودی از حالت a یا d فرا می‌رسد. به این دلیل یک حالت c ناپایدار در ستون 00 و سطرهای a و d از جدول روند نشان داده شده است. خروجی با خط تیره نشان داده شده است تا حالت بی‌اهمیت را بیانگر باشد. تفسیر این است که اگر مدار در حالت پایدار a باشد و ورودی از 01 به 00 برود، مدار ابتدا به یک حالت ناپایدار بعدی c می‌رود که در این صورت حالت فعلی a به c تبدیل می‌شود و در نتیجه گذری به سطر سوم ستون اول جدول روند را خواهیم داشت. مقادیر حالات ناپایدار دیگر که به طریقی مشابه بدست می‌آیند با خط تیره مشخص شده و به معنی حالات بی‌اهمیت تلقی می‌گردند. تخصیص مقادیر واقعی به خروجی‌ها بعد و پس از تکمیل یک مثال مورد بحث قرار خواهد گرفت.

کاهش جدول روند اصلی

یک جدول روند اصلی تنها یک حالت پایدار در هر سطر دارد. اگر دو یا چند حالت در یک سطر قرار گیرند جدول به تعداد کمتری از سطرها کاهش می‌یابد. جمع کردن حالات پایدار در یک سطر به زیر یک ستون را ادغام می‌گویند. ادغام کردن تعدادی از حالات پایدار واقع در یک سطر به این معنی است که متغیر حالت دودویی که نهایتاً به سطر ادغام شده تخصیص می‌یابد، با تغییر متغیرهای ورودی تغییر

DG	00	01	11	10
a	c, –	(a), 0	b, –	–, –
c	(c), 0	a, –	–, –	d, –
d	c, –	–, –	b, –	(d), 0

DG	00	01	11	10
b	–, –	a, –	(b), 1	e, –
e	f, –	–, –	b, –	(e), 1
f	(f), 1	a, –	–, –	e, –

(الف) حالت های منتخب برای ادغام

DG	00	01	11	10
a, c, d	(c), 0	(a), 0	b, –	(d), 0
b, e, f	(f), 1	a, –	(b), 1	(e), 1

DG	00	01	11	10
a	(a), 0	(a), 0	b, –	(a), 0
b	(b), 1	a, –	(b), 1	(b), 1

(ب) جدول کاهش یافته (دو انتخاب دیگر)

شکل ۱۷-۹ . کاهش جدول روند اصلی

نخواهد کرد. دلیل این است که در یک جدول روند اصلی، متغیر حالت در ازاء هر تغییر ورودی، تغییر می‌کند، ولی در یک جدول روند کاهش یافته، اگر حالت با ثبات بعدی با قبلی در یک سطر قرار داشته باشند، متغیر حالت تغییر نخواهد کرد.

روال کاهش یک جدول روند در بخش بعدی داده شده است. برای تکمیل مثال طراحی بدون پیگیری یک روال مشخص، فرآیند ادغام را با به کارگیری قواعد ساده شده ادغام اعمال خواهیم کرد. در جدول روند می‌توان دو یا چند سطر را در یک سطر ادغام کرد به شرطی که حالات و خروجی در هر یک از ستون‌ها در تضاد نباشند، اگر یک سمبل حالت و وارده‌های بی‌اهمیت در یک ستون ظاهر شوند، حالت در سطر ادغام شده لیست می‌گردد. به علاوه اگر حالت در یکی از سطرها در دایره محصور شده باشد، در سطر ادغام شده هم محصور می‌شود. مقدار خروجی هم در هر حالت پایدار در سطر ادغام شده لحاظ می‌گردد.

اکنون این قواعد را به جدول روند شکل ۱۶-۹ اعمال می‌کنیم. برای این که ببینیم چگونه این کار صورت می‌گیرد، جدول روند به دو بخش و هر بخش به سه سطر، طبق شکل ۱۷-۹ (الف) تقسیم می‌شود. هر بخش سه حالت با ثبات را نشان می‌دهد که قابل ادغامند زیرا هیچ وارده متضادی در هر یک از چهار ستون ملاحظه نمی‌شود. ستون اول حالت c را در همه سطرها و 0 و خط تیره را برای خروجی نشان می‌دهد. چون هر خط تیره بیانگر حالت بی‌اهمیت است، می‌تواند مربوط به هر حالت یا خروجی باشد. دو خط تیره در ستون اول را می‌توان به عنوان خروجی 0 تلقی کرد تا هر سه سطر مشابه با سطری شود که در آن حالت با ثبات c و خروجی 0 است. دومین ستون نشان می‌دهد که خطوط تیره را می‌توان

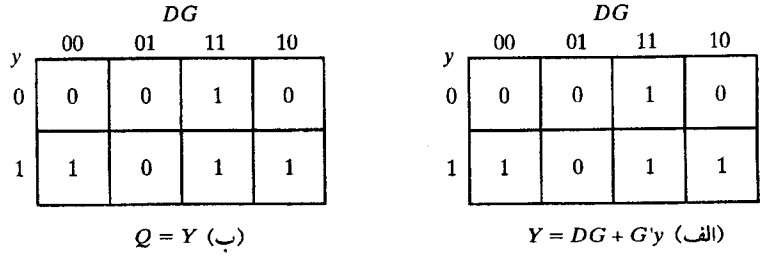
به حالت پایدار a با خروجی 0 تخصیص داد. توجه کنید که اگر حالت در یکی از سطرها محصور باشد، در سطر ادغام شده نیز محصور خواهد بود. به طور مشابه ستون سوم را می‌توان به یک حالت بی‌ثبات b و خروجی بی‌اهمیت ادغام نمود و چهارمین ستون به حالت پایدار d و خروجی 0 ادغام می‌شود. بنابراین سه سطر a، c و d را می‌توان به صورت یک سطر با سه حالت پایدار و یک حالت ناپایدار ادغام کرد. سطر اول شکل ۱۷-۹ (ب) این ادغام را نشان می‌دهد. سطر دوم جدول ادغام شده، از ادغام سطرهای b ، e و f در جدول روند بدست می‌آید. برای ترسیم جدول کاهش یافته دو راه وجود دارد. سمبل‌های حرفی برای حالات را می‌توان حفظ کرد تا رابطه بین جداول روند اصلی و کاهش یافته مشخص شود. روش دیگر این است که برای جداول روند اصلی و کاهش یافته برای همه حالات پایدار در سطرهای ادغام شده سمبل‌های حرفی مشترکی تعریف کنیم. بنابراین حالات c و d با حالت a و حالات e و f با حالت b جایگزین می‌شوند. هر دو روش در شکل ۱۷-۹ (ب) مشاهده می‌شود.

جدول گذر و نمودار منطقی

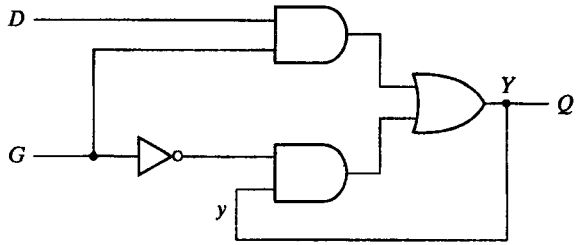
برای بدست آوردن مداری که با جدول روند کاهش یافته تعریف شده باشد، لازم است به هر حالت مقدار دودویی جداگانه‌ای اختصاص یابد. این تخصیص جدول روند را به جدول گذر مبدل می‌سازد. در حالت کلی، برای مصون بودن مدار از رقابت‌های بحرانی، تخصیص حالت دودویی باید صورت گیرد. مسئله تخصیص حالت در مدارهای ترتیبی غیرهمزمان در راه حل‌های آن در بخش ۶-۹ مورد بحث قرار گرفته است. خوشبختانه در جدول روند دو سطری حالت رقابت وجود ندارد، و بنابراین، می‌توانیم طراحی لچ گیت‌دار را قبل از بخش ۶-۹ مطالعه کنیم. با تخصیص 0 به حالت a و 1 به حالت b در جدول روند کاهش یافته شکل ۱۷-۹ (ب)، جدول گذر شکل ۱۸-۹ (الف) بدست می‌آید. در واقع جدول گذر نقشه‌ای برای متغیر تحریک Y است. آنگاه تابع بول ساده شده برای Y از نقشه به صورت زیر بدست می‌آید.

$$Y = DG + G'y$$

در جدول روند ساده شده نهایی دو خروجی بی‌اهمیت وجود دارد. اگر مقادیر خروجی را مطابق شکل ۱۸-۹ (ب) اختیار کنیم، ممکن است خروجی Q را برابر تابع تحریک Y نمود. اگر دیگر مقادیر ممکن را به خروجی بی‌اهمیت نسبت دهیم، خروجی Q را برابر y خواهیم کرد. در هر یک از دو حال،



شکل ۱۸-۹. جدول گذر و نقشهٔ خروجی لچ گیت دار



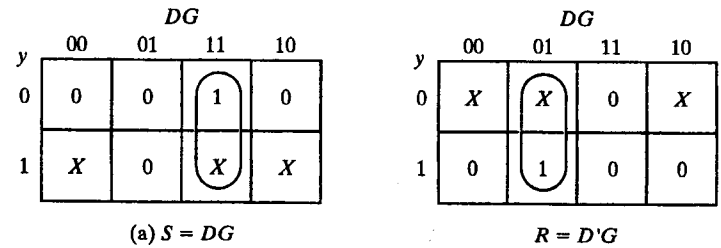
شکل ۱۹-۹. نمودار منطقی لچ گیت‌دار

نمودار منطقی لچ گیت‌دار مطابق شکل ۱۹-۹ خواهد بود.

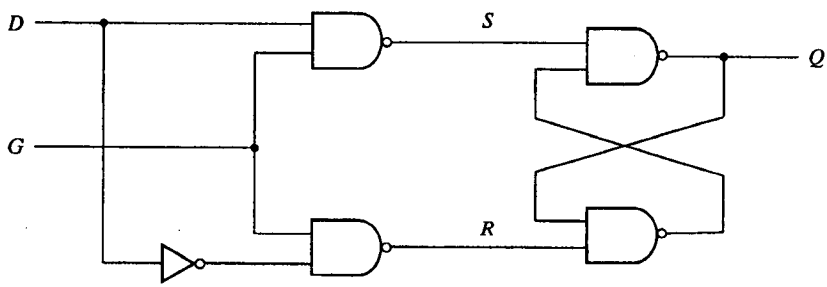
نمودار را با لچ SR هم می‌توان پیاده‌سازی کرد. با توجه به روال بخش ۳-۹ ابتدا توابع بول را برای S و R، طبق شکل ۲۰-۹ (الف) بدست می‌آوریم. نمودار منطقی با گیت‌های NAND در شکل ۲۰-۹ (ب) دیده می‌شود. توجه کنید که لچ گیت‌دار یک لچ D حساس به سطح است که در شکل ۶-۵ بخش ۲-۵ معرفی شد.

تخصیص خروجی‌ها به حالت ناپایدار

حالات پایدار در یک جدول روند دارای مقادیر خروجی خاص مربوط به خود هستند. حالات ناپایدار

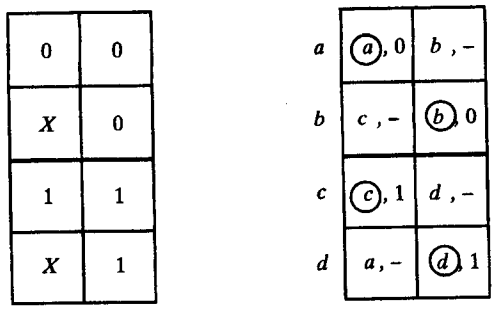


(الف) نقشه‌های S و R



(ب) نمودار منطقی

شکل ۲۰-۹. مدار با لچ SR



(الف) جدول روند **(ب) تخصیص خروجی**

شکل ۲۱-۹. انتساب مقادیر خروجی به حالتهای ناپایدار

دارای وارده‌های نامشخص می‌باشند که با خطوط تیره نشان داده شده‌اند. مقادیر خروجی برای حالات ناپایدار باید طوری انتخاب شوند که حین گذر بین دو حالت، خروجی‌های لحظه‌ای غلط را تولید نکنند. این بدان معنی است که در نتیجه یک گذر بین دو حالت قرار نیست که یک متغیر خروجی تغییر نماید. در این صورت یک حالت بی‌ثبات بین دو حالت پایدار باید خروجی یکسانی با خروجی حالات پایدار داشته باشد. به عنوان مثال جدول حالت شکل ۲۱-۹ (الف) را در نظر بگیرید. گذر از حالت پایدار a به حالت پایدار b از طریق حالت ناپایدار b صورت می‌گیرد. اگر خروجی مربوط به حالت بی‌ثبات b برابر 1 باشد، آنگاه یک پالس کوتاه لحظه‌ای ضمن جابجایی مدار از خروجی 0 در حالت a به خروجی 1 برای حالت بی‌ثبات b رفته و ضمن رفتن به حالت پایدار b دوباره به 0 باز می‌گردد. بنابراین خروجی مربوط به حالت ناپایدار b باید 0 در نظر گرفته شود تا خروجی لحظه‌ای غلط تولید نشود. اگر یک متغیر خروجی به دلیل تغییر حالت، عوض شود، آنگاه به این متغیر یک حالت بی‌اهمیت اختصاص می‌یابد. مثلاً گذر از حالت پایدار b به حالت پایدار c در شکل ۲۱-۹ (الف)، خروجی را از 0 به 1 تغییر مقدار می‌دهد. اگر برای حالت بی‌ثبات c خروجی 0 در نظر گرفته شود، آنگاه متغیر خروجی تا انتهای گذر تغییر نخواهد کرد. اگر یک 1 وارد کنیم، تغییر در آغاز گذر رخ می‌دهد. چون فرقی نمی‌کند که تغییر خروجی چه زمانی اتفاق بیفتد بنابراین برای خروجی حالت ناپایدار c یک X می‌گذاریم. شکل ۲۱-۹ (ب) تخصیص خروجی را برای جدول روند نشان می‌دهد. این جدول چهار حالت ممکن را در تغییر خروجی نمایش می‌دهد. روال تخصیص مقادیر به خروجی‌های مربوط به حالات بی‌ثبات در زیر خلاصه شده است.

۱- به متغیر خروجی که مربوط به حالت ناپایدار بین دو حالت پایدار با متغیر خروجی 0اند مقدار 0 را اختصاص دهید.

۲- به متغیر خروجی که مربوط به حالت ناپایدار بین دو حالت پایدار با متغیر خروجی 1اند، مقدار 1 را منتسب کنید.

۳- به متغیر خروجی که مربوط به حالت ناپایدار بین دو حالت پایدار با متغیرهای متفاوتند (0 و 1 یا 1 و 0) مقدار بی‌اهمیت را نسبت دهید.

خلاصه روش طراحی

طراحی مدارهای ترتیبی غیرهمزمان با استفاده از روش تشریح شده در مثال قبل قابل اجراست. بعضی از مراحل طراحی نیاز به تشریح بیشتری دارند که در بخش‌های زیر توضیح داده شده است. مراحل عبارتند از:

۱- از مشخصات طراحی مفروض، یک جدول روند اصلی بدست آورید. این مشکل‌ترین بخش طراحی است زیرا باید در آن تجربه و تبحر را برای رسیدن به یک تفسیر صحیح از مشخصات مسئله به کار برد.

۲- با ادغام سطرها در جدول روند اصلی، جدول روند کاهش یافته را بدست آورید. یک روال مستدل برای ادغام سطرها در بخش ۵-۹ ارائه شده است.

۳- به هر سطر جدول روند کاهش یافته متغیرهای حالت دودویی را نسبت دهید تا جدول گذر حاصل شود، روش تخصیص حالتی که رقابت‌های بحرانی احتمالی را حذف می‌کند در بخش ۶-۹ داده شده است.

۴- به خطوط تیره مربوط به حالات ناپایدار مقادیر خروجی را نسبت دهید تا نقشه خروجی تهیه شود. این کار قبلاً توضیح داده شد.

۵- توابع بولی تحریک و متغیرهای خروجی را ساده کنید و نمودار منطقی آنها را طبق بخش ۲-۹ رسم نمایید. نمودار منطقی را با لچ‌های SR طبق بخش ۲-۹ و نیز انتهای بخش ۷-۹ می‌توان ترسیم کرد.

۵-۹ کاهش جداول حالت و روند

روال کاهش تعداد حالات داخلی در یک مدار ترتیبی غیرهمزمان، روش به کار رفته در مدارهای همزمان را به خاطر می‌آورد. الگوریتم کاهش حالت یک جدول حالت کامل در بخش ۶-۵ داده شد. ما این الگوریتم را مرور کرده و آن را به روش کاهش حالتی که از یک جدول ایجاب استفاده می‌کند اعمال می‌نماییم. سپس الگوریتم و جدول ایجاب اصلاح می‌شود تا کاهش حالت مربوط به جداول حالت غیرکامل را پوشش دهد. این الگوریتم اصلاح شده برای تشریح روال کاهش جدول روند مدارهای ترتیبی غیرهمزمان به کار خواهد رفت.

جدول ایجاب

روال کاهش حالت جداول حالت کامل بر این اساس استوار است که اگر دو حالت در جدول حالت معادل باشند می‌توان آنها را ترکیب کرده و با یک حالت نشان داد. دو حالت را معادل گوییم اگر به ازاء هر ورودی ممکن، دقیقاً خروجی یکسانی تولید کنند و به حالت بعدی یکسان و یا حالات بعدی معادلی هم بروند. جدول ۶-۶ مثالی از حالات معادلی را نشان می‌دهد که دارای خروجی و حالات بعدی

جدول ۳-۹. جدول حالات برای نشان دادن حالتهای معادل

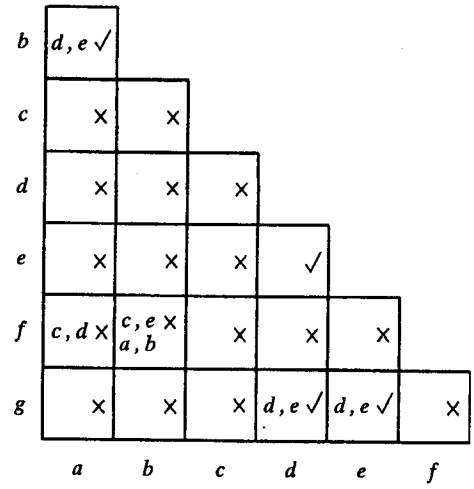
حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	*c*	*b*	0	1
b	*d*	*a*	0	1
c	*a*	*d*	1	0
d	*b*	*d*	1	0

یکسانی در قبال هر ترکیبی از ورودی‌هاست. مواردی وجود دارد که جفت حالت، حالات بعدی یکسانی ندارند، ولی کم و بیش به حالت بعدی معادلی می‌روند. مثلاً جدول حالت ۳-۹ را ملاحظه نمایید. حالات فعلی a و b به ازاء ورودی‌های یکسان، خروجی یکسانی دارند. حالت بعدی آنها برای $x = 0$ برابر c و d و بازاء $x=1$ برابر a و b است. اگر بتوانیم نشان دهیم که جفت حالت (d و c) معادلند، جفت حالت (b و a) هم معادل خواهند بود، زیرا آنها حالت بعدی یکسان یا معادلی خواهند داشت. وقتی چنین رابطه‌ای برقرار باشد، گوییم زوج (b و a) زوج (d و c) را موجب است. با توجه به دو سطر آخر جدول ۳-۹، می‌بینیم که جفت حالت (d و c) موجب (b و a) است. ویژگی حالات معادل این است که اگر (d و c) موجب (b و a) و (b و a) نیز (d و c) را موجب شوند، آنگاه جفت حالات معادلند؛ یعنی a و b معادلند و نیز c و d معادلند. در نتیجه چهار سطر جدول ۳-۹ را می‌توان با ترکیب a و b به یک حالت و c و d را به دومین حالت کاهش داد.

بررسی جفت حالات برای یافتن معادل‌های احتمالی در یک جدول که در آن تعداد قابل توجهی حالت وجود دارد، به طور سیستماتیک توسط جدول ایجاب قابل انجام است. جدول ایجاب چارتی متشکل از خانه‌های مربعی شکل است که هر یک متعلق به یک جفت حالت بوده و فضایی را برای لیست کردن هر حالت ممکن فراهم می‌سازد. با به کارگیری صحیح جدول، می‌توان همه جفت حالات معادل را پیدا کرد. جدول حالت ۴-۹، برای تشریح این روال به کار گرفته خواهد شد. جدول ایجاب در شکل ۲۲-۹ ملاحظه می‌شود. در ضلع سمت چپ عمودی همه حالاتی که در جدول حالت تعریف

جدول ۴-۹. جدول حالتی که قرار است کاهش یابد

حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	*d*	6	0	0
b	*e*	*a*	0	0
c	*g*	*f*	0	1
d	*a*	*d*	1	0
e	*a*	*d*	1	0
f	*c*	*b*	0	0
g	*a*	*e*	1	0



شکل ۲۲-۹. جدول ایجاب

شده‌اند، به جز اولی، لیست شده‌اند. در ضلع پایین جدول هم متغیرها به جز آخری لیست شده‌اند. نتیجه این کار نمایش همه ترکیبات ممکن از دو حالت با مربعی است که در محل تقاطع یک سطر و یک ستون واقع است و در آنها معادل‌ها تست خواهند شد.

در مربعات مربوط به دو حالتی که معادل نیستند علامت ضربدر (×) زده شده است، در حالی که معادل‌ها علامت تأیید (✓) خورده‌اند. بعضی از مربعات وارده‌هایی از حالات موجب دارند که باید بیشتر مورد بررسی قرار گیرند تا معین شود آیا معادلند یا خیر. روال قدم به قدم در پر کردن مربعات به شرح زیر است. ابتدا ضربدری در مربع جفت حالتی قرار می‌دهیم که خروجی‌های آنها در قبال هر ورودی یکی نیست. در این راستا، حالت c خروجی متفاوتی با دیگر حالات دارد و به این علت علامت ضربدر در دو مربع سطر c و چهار مربع ستون c، زده می‌شود. به همین ترتیب نُه مربع دیگر در این جدول وجود دارند که ضربدر زده شده‌اند.

سپس در مربع‌های باقیمانده جفت حالاتی را وارد می‌کنیم که بوسیله جفت حالات بیانگر مربع موجب شده‌اند. این کار را از مربع فوقانی در سمت چپ به پایین انجام می‌دهیم. و سپس کار را با ستون بعدی ادامه داده و به سمت راست پیش می‌رویم. با توجه به جدول حالات، می‌بینیم که معادل بودن (b و a) دلالت بر معادل بودن (e و d) دارد. بنابراین (e و d) در مربعی که به وسیله ستون a و سطر b تعریف می‌شود ثبت می‌گردد. به همین ترتیب کار را تا تکمیل شدن همه جدول ادامه می‌دهیم. دقت کنید که حالات (e و d) با هم معادلند زیرا به حالت بعدی یکسانی رفته و خروجی برابری دارند. در نتیجه یک علامت تأیید (✓) در مربعی می‌گذاریم که به وسیله ستون d و سطر e تعریف می‌شود. این کار بدان معنی است که دو حالت معادلند و از هر جفت حالت موجب مستقل می‌باشند.

قدم بعدی ساختن مسیرهای متوالی در جدول است که برای تعیین مربع‌هایی که باید (×) بخورند به

کار می‌روند. هر مربعی از جدول که حداقل حاوی یک جفت موجب نا معادل است ضربدر می‌خورد. مثلاً مربعی که به وسیله a و f تعریف می‌شود، بعد از c و d ضربدر خورده است زیرا جفت (d و c) مربعی را تعریف می‌کنند که حاوی یک ضربدر است. این روال تا جایی ادامه می‌یابد که دیگر مربعی برای ضربدر زدن وجود نداشته باشد. بالاخره در همه مربعاتی که ضربدر نخورده‌اند، علامت تأیید زده می‌شود. این مربعات جفت حالات معادل را تعریف می‌کنند. در این مثال حالات معادل عبارتند از

$$(a, b) \quad (d, e) \quad (d, g) \quad (e, g)$$

اکنون جفت حالات را به گروه حالات معادل بزرگتری تبدیل می‌نماییم. سه جفت آخر را می‌توان به یک گروه حالت معادل سه‌تایی (d, e, g) تبدیل کرد زیرا هر یک از حالات در گروه با دو حالت دیگر معادل است. آخرین بخش حالات متشکل از حالات معادلی است که از جدول ایجاب همراه با حالات باقیمانده در جدول حالت بدست می‌آیند و با دیگر حالات معادل نیستند، یعنی

$$(a, b) \quad (c) \quad (d, e, g) \quad (f)$$

این بدان معنی است که جدول ۴-۹ قابل کاهش از هفت حالت به چهار حالت است که هر حالت متعلق به یکی از پرانتزهاست. جدول کاهش یافته از جایگزینی b با a و حالات e و g با d حاصل می‌گردد. این جدول در شکل ۵-۹ مشاهده می‌شود.

ادغام جدول روند

مواردی وجود دارد که در آن جدول حالت برای یک مدار ترتیبی به طور غیرکامل یا ناقص بیان شده است. این مورد هنگامی پیش می‌آید که بعضی از ترکیبات ورودی‌ها یا رشته‌های ورودی به دلیل محدودیت‌ها هرگز رخ نمی‌دهند. در یک چنین حالتی، حالات بعدی و خروجی‌هایی که به شرط وقوع ورودی‌ها می‌باید رخ می‌دادند، هرگز اتفاق نمی‌افتند و بنابراین به عنوان حالات بی‌اهمیت تلقی می‌شوند. هر چند که مدارهای ترتیبی همزمان گاهی با جداول حالت ناقص نمایش داده می‌شوند، ولی، در اینجا علاقمند به بررسی مدارهای ترتیبی غیرهمزمانی هستیم که جدول روند اصلی آنها همیشه به صورت ناقص تعریف شده است.

حالات ناقص را می‌توان برای کاهش حالات جدول روند با یکدیگر ترکیب کرد. چنین حالاتی را

جدول ۵-۹. جدول حالات کاهش یافته

حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
f	c	a	0	0

نمی‌توانیم معادل بخوانیم زیرا تعریف قبلی معادل لازم می‌دارد تا همه خروجی‌ها و حالات بعدی برای همه ورودی‌ها مشخص باشند. در عوض دو حالت ناقصی که بتوانند با یکدیگر ترکیب شوند سازگار خوانده می‌شوند. دو حالت هنگامی سازگارند که برای هر ورودی ممکن خروجی یکسان و حالت بعدی آنها به محض اینکه مشخص شدند سازگار باشند. همه حالات بی‌اهمیتی که با خط تیره معین شده‌اند حالات نامشخص را بیان می‌کنند و هنگام جستجوی حالات سازگار تأثیری ندارند.

فرآیندی که باید برای یافتن گروه‌های سازگار در راستای ادغام یک جدول روند اعمال گردد قابل تفکیک به سه مرحله زیر است:

۱- با استفاده از جدول ایجاب همه جفت‌های سازگار را معین کنید.

۲- با استفاده از نمودار ادغام، سازگاری‌های ماکزیمال را بدست آورید.

۳- مجموعه‌ای از سازگارهای مینیمال را پیدا کنید که کلیه حالات را دربرگیرد و نیز مجموعه‌ای بسته باشد.

آنگاه این مجموعه مینیمال برای ادغام سطرهای جدول روند مورد استفاده قرار می‌گیرد. اکنون با استفاده از روند اولیه مثال بخش قبل، سه مرحله فوق را تشریح خواهیم کرد.

زوج‌های سازگار

روال یافتن زوج‌های سازگار در شکل ۲۳-۹ تشریح شده است. جدول روند اصلی در (الف) همانند شکل ۶-۹ می‌باشد. وارده‌ها در هر مربع، حالت بعدی و خروجی را بیان می‌کند. خطوط تیره (–) حالات یا خروجی‌های نامشخص را نمایش می‌دهند. جدول ایجاب، همچون موردی که در یافتن

	00	01	11	10
a	c, –	(a), 0	b, –	–, –
b	–, –	a, –	(b), 1	e, –
c	(c), 0	a, –	–, –	d, –
d	c, –	–, –	b, –	(d), 0
e	f, –	–, –	b, –	(e), 1
f	(f), 1	a, –	–, –	e, –

(الف) جدول روند اصلی

	a	b	c	d	e
b	✓				
c	✓	d, e ×			
d	✓	d, e ×	✓		
e	c, f ×	✓	d, e × c, f ×	×	
f	c, f ×	✓	×	d, e × c, f ×	✓

(ب) جدول ایجاب

شکل ۲۳-۹. جدولهای روند و ایجاب

حالات معادل در وضعیت کامل به کار رفت، برای بدست آوردن حالات سازگار نیز مورد استفاده است. تنها تفاوت این است که وقتی سطرها را مقایسه می‌کنیم، آزادیم تا خطوط تیره را برای تطبیق هر حالت موردنظر تنظیم کنیم.

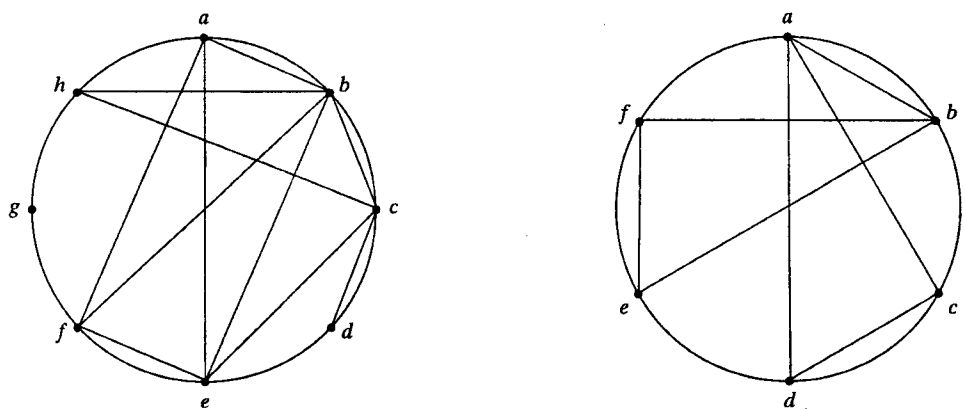
دو حالت را سازگار گوییم اگر در هر ستون از سطر مربوطه در جدول روند، حالات یکسان یا سازگار وجود داشته و تضاد و تناقضی در مقادیر خروجی نباشد. مثلاً سطرهای a و b در جدول روند سازگارند ولی سطرهای a و f فقط اگر c و f سازگار باشند، سازگار خواهند بود. معهذا سطرهای c و f سازگار نیستند زیرا در ستون اول دارای خروجی‌های متفاوتی می‌باشند. این اطلاعات در جدول ایجاب ثبت می‌شود. علامت تأیید (✓) مربعی را مشخص می‌کند که جفت حالات آن سازگارند. حالاتی که سازگار نباشند با (×) علامت خورده‌اند. مربع‌های باقیمانده با زوج‌های موجب پر می‌شوند و بررسی بیشتری را نیاز دارند.

جدول ایجاب اولیه به محض پر شدن، مجدداً مرور می‌گردد تا حالات موجبی که ناسازگارند ضربدر بخورند. مربع‌های باقیمانده که دارای علامت (✓) می‌باشند جفت‌های سازگار را مشخص می‌نمایند. در مثال شکل (۲۳-۹) زوج‌های سازگار عبارتند از:

$$(a,b)\quad (a,c)\quad (a,d)\quad (b,e)\quad (b,f)\quad (c,d)\quad (e,f)$$

سازگارهای ماکزیمال

پس از یافتن زوج‌های سازگار، پیدا کردن بزرگترین مجموعه‌ها از مجموعه‌های سازگار، قدم بعدی است. سازگار ماکزیمال گروهی از سازگارها هستند که حاوی همه ترکیبات ممکن از حالات سازگار باشند. سازگار ماکزیمال را از نمودار ادغام طبق شکل ۲۴-۹، می‌توان بدست آورد. نمودار ادغام گرافی است که در آن هر حالت با یک نقطه بر روی یک دایره مشخص می‌شود. بین دو نقطه که یک زوج سازگار



(الف) سازگارهای ماکزیمال
$(a,b,)\ (a,c,d)\ (b,e,f)$

(ب) سازگارهای ماکزیمال
$(a,b,e,f)\ (b,c,h)\ (c,d)\ (g)$

شکل ۲۴-۹ . نمودارهای ادغام

را تشکیل دهند یک خط رسم می‌شود. کلیه سازگارهای ممکن را می‌توان از روی نمودار ادغام با مشاهده شکل‌های هندسی که در آنها حالات به یکدیگر متصل هستند، بدست آورد. یک نقطه جدا مانده که به دیگر نقاط وصل نباشد حالتی است که با هیچ حالت دیگر سازگار نیست. یک خط، نشان دهنده یک زوج سازگار است. یک مثلث سه حالت سازگار را مشخص می‌کند. در نمودار ادغام، n حالت سازگار با یک کثیر الاضلاع n ضلعی که کلیه قطرهای آن وصل باشند، مشخص می‌شوند.

نمودار ادغام شکل ۲۴-۹ (الف) از روی لیست جفت‌های سازگار مشتق از جدول ایجاب شکل ۲۳-۹ بدست می‌آید. در این شکل هفت خط مستقیم وجود دارد که نقاط را به هم وصل می‌کنند و هر یک متعلق به یک زوج سازگار می‌باشند. الگوی هندسی حاصل از این خطوط شامل دو مثلث (a, c, d) و (b, e, f) و یک خط (a, b) می‌باشد. سازگارهای ماکزیمال عبارتند از:

$$(a, b) \quad (a, c, d) \quad (b, e, f)$$

شکل ۲۴-۹ (پ) نمودار ادغام یک جدول روند هشت حالته را نشان می‌دهد. در این شکل الگوی هندسی شامل مستطیلی است که قطرهایش رسم شده است تا چهار حالت سازگار (a, b, e, f)، یک مثلث (b, c, h)، یک خط (c, d) و یک حالت منفرد g که با هیچ حالت دیگری سازگار نیست، به وجود آید. سازگارهای ماکزیمال عبارتند از:

$$(a, b, e, f) \quad (b, c, h) \quad (c, d) \quad (g)$$

مجموعه سازگار ماکزیمال می‌تواند برای ادغام جدول روند استفاده شود به این ترتیب که در جدول کاهش یافته به هر عضو مجموعه سطری منتسب می‌گردد. در بسیاری از موارد سازگارهای ماکزیمال لزوماً مجموعه سازگارهای مینیمال را تشکیل نمی‌دهند. غالباً تهیه مجموعه کوچکتری از سازگارها که شرط ادغام سطرها را دارا باشند، امکان‌پذیر است.

شرط پوششی بسته

شرطی که برای ادغام سطر وجود دارد این است که مجموعه سازگارهای منتخب باید همه حالات را پوشش داده و بسته باشند. مجموعه‌ای تمام حالات را پوشش می‌دهد که همه حالات جدول اولیه را دارا باشد. شرط بسته بودن هنگامی برقرار است که حالات موجب وجود نداشته باشد و یا این حالات در داخل خود مجموعه باشند. یک مجموعه بسته از سازگارها که شامل همه حالات باشد، مجموعه پوششی بسته نامیده می‌شود. شرط پوششی بسته، با دو مثال زیر توضیح داده شده‌اند.

به سازگارهای ماکزیمال شکل ۲۴-۹ (الف) توجه کنید. اگر (a, b) را حذف کنیم، دو مجموعه سازگار را خواهیم داشت.

$$(a, c, d) \quad (b, e, f)$$

همه حالات ششگانه جدول روند شکل ۲۳-۹ در این مجموعه وجود دارند. این خود حالت پوششی را می‌رساند. با توجه به جدول ایجاب شکل ۲۳-۹ (ب) حالات موجبی برای (a, c) ؛ (a, d) ؛ (c, d)؛ (b, e) ؛ (b, f) و (e, f) وجود ندارند، بنابراین شرط بسته بودن برقرار است. به این ترتیب جدول روند

اصلی قابل ادغام به دو سطر و هر یک برای یکی از سازگارها خواهد بود. جزئیات ساختن جدول کاهش یافته برای این مثال خاص در بخش قبل انجام و در شکل ۱۷-۹ (ب) نشان داده شد.

مثال دوم از یک جدول روند اصلی است (در اینجا نشان داده نشده است) که جدول ایجاب آن در شکل ۵-۹ (الف) دیده می‌شود. زوج سازگارهای حاصل از جدول ایجاب عبارتند از:

$$(a, b)\ (a, d)\ (b, c)\ (c, d)\ (c, e)\ (d, e)$$

با توجه به نمودار ادغام شکل ۲۵-۹ (ب)، سازگارهای ماکزیمال را معین می‌کنیم:

$$(a, b) \quad (a, d) \quad (b, c) \quad (c, d, e)$$

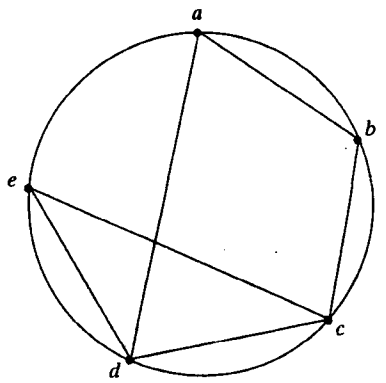
اگر دو سازگار

$$(a, b) \quad (c, d, e)$$

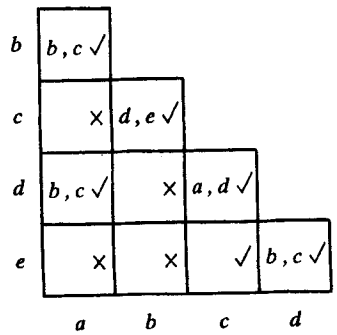
را انتخاب کنیم در مجموعه مزبور هر پنج حالت جدول اصلی را می‌پوشاند. شرط بسته بودن را به کمک جدول بسته بودن شکل ۲۵-۹ (پ) می‌توان چک کرد. جفت‌های موجب که برای هر سازگار لیست شده مستقیماً از جدول ایجاب حاصل شده‌اند. حالات موجب برای (a, b)، حالات (b, c) می‌باشند. ولی (b, c) در مجموعه انتخابی (a, b) (c, d, e) وجود ندارد، در نتیجه مجموعه سازگارها بسته نیست. مجموعه‌ای از سازگارها که شرط پوششی بسته را اغناء می‌کنند عبارتند از:

$$(a, d) \quad (b, c) \quad (c, d, e)$$

این یک مجموعه پوششی است زیرا دارای هر پنج حالت می‌باشد. توجه کنید هر حالت می‌تواند بیش از



(ب) نمودار ادغام



(الف) جدول ایجاب

سازگارها	(a, b)	(a, d)	(b, c)	(c, d, e)
حالات موجب	(b, c)	(b, c)	(d, e)	$(a, d,)$ $(b, c,)$

(پ) جدول بسته بودن

شکل ۲۵-۹. انتخاب مجموعه‌ای از سازگارها

یک بار تولید شود. مجموعه شرط بسته بودن را هم دارد زیرا حالات موجب (a, d) و (b, c) (d, e) می‌باشند که در مجموعه لحاظ شده‌اند. جدول روند اصلی (که در اینجا نشان داده نشده است) را می‌توان با ادغام سطرهای a و d؛ b و c؛ c و d با e از 5 سطر به 3 سطر کاهش داد. توجه کنید انتخاب دیگری که سازگارهای پوششی بسته را شامل می‌شود عبارت است از (a, b) (d, e) (b, c). به طور کلی، ممکن است هنگام کاهش جدول روند اصلی بیش از یک راه ادغام وجود داشته باشد.

۶-۹ تخصیص حالت فاقد رقابت

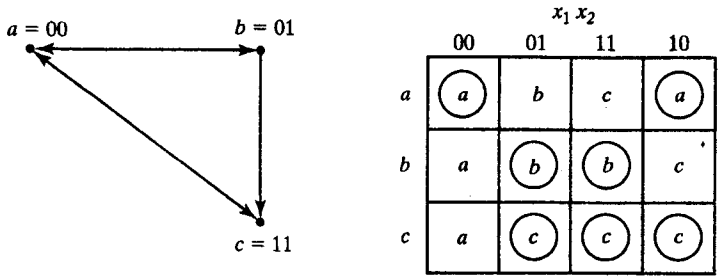
به محض این که یک جدول روند کاهش یافته برای یک مدار ترتیبی غیرهمزمان بدست آمد، قدم بعدی در طراحی، تخصیص مقادیر دودویی به هر حالت پایدار است. این تخصیص موجب تبدیل جدول روند به جدول گذر معادلش می‌شود. هدف اصلی در انتخاب یک تخصیص دودویی صحیح ممانعت از رقابت‌های بحرانی است. موضوع رقابت‌های بحرانی در بخش ۲-۹ به همراه شکل ۷-۹ توضیح داده شد.

از رقابت‌های بحرانی می‌توان اجتناب کرد به این ترتیب که تخصیص حالت دودویی طوری صورت گیرد که در هر لحظه از زمان که گذر حالتی در جدول روند رخ می‌دهد تنها یک متغیر عوض شود. برای حصول به این تفکر، لازم است تا بین حالاتی که گذر اتفاق می‌افتد تخصیص‌های همجوار را نسبت دهیم. دو مقدار دودویی را همجوار گوییم اگر فقط در یک متغیر با هم اختلاف داشته باشند. مثلاً 010 و 011 مجاورند زیرا فقط در بیت سوم با هم اختلاف دارند.

برای اطمینان از داشتن یک گذر فاقد رقابت‌های بحرانی، لازم است هر گذر ممکن بین دو حالت پایدار تست شده و اطمینان حاصل کنیم که هر بار یک متغیر حالت دودویی تغییر می‌کند. البته این کار خسته کننده است، به خصوص اگر سطرها و ستون‌های جدول متعدد باشند. برای ساده کردن موضوع، ما روال تخصیص حالت دودویی را با چند مثالی که فقط سه یا چهار سطر دارند توضیح خواهیم داد. این مثال‌ها روالی کلی را که باید دنبال شوند نشان می‌دهند تا یک تخصیص حالت فاقد رقابت تضمین شود. آنگاه می‌توان روال را به جداول روندی با هر تعداد سطر و ستون تعمیم داد.

مثال جدول روند سه سطری

انتساب یک متغیر دودویی به یک جدول روند دو سطری مشکل رقابت بحرانی را دربرندارد. یک جدول روند سه سطری به تخصیص دو متغیر نیاز دارد. اگر تخصیص مقادیر دودویی به حالات پایدار به درستی انجام نشود ممکن است سبب بروز رقابت‌های بحرانی گردد. مثلاً به جدول روند کاهش یافته شکل ۲۶-۹ (الف) توجه کنید. به منظور سادگی خروجی‌ها از جدول حذف شده‌اند. بررسی سطر a نشان می‌دهد که از حالت a به b در ستون 01 و از حالت a به c در ستون 11 گذری وجود دارد. طبق شکل ۲۶-۹ (ب) این اطلاعات به نمودار گذر انتقال می‌یابد. خطوط مستقیم بین a و b، و نیز بین a و c دو انتقال فوق را نشان می‌دهند. به طور مشابه گذر از دو سطر دیگر نیز با خطوط جهت‌دار در نمودار گذر نمایش داده شده است. نمودار گذر، نمایشی مصور از کلیه انتقال‌های موجود بین سطرهاست.



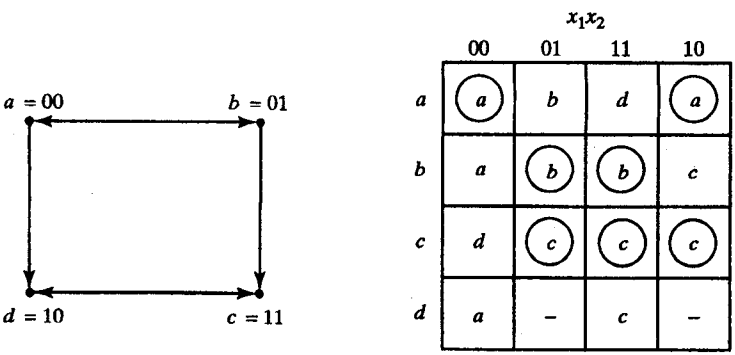
(ب) نمودار انتقال

(الف) جدول روند

شکل ۲۶-۹ . مثال جدول روند سه سطری

برای اجتناب از رقابت‌های بحرانی باید تخصیص حالتی دودویی را بیابیم که در هر گذر حالت آن تنها یک متغیر عوض شود. تلاشی در راستای یافتن چنین تخصیصی در نمودار گذر نشان داده شده است. به حالت a، مقدار دودویی 00 و به حالت c مقدار دودویی 11 منتسب شده است. این تخصیص به هنگام گذر از a به c یک رقابت بحرانی ایجاد خواهد کرد زیرا در متغیرهای حالت دودویی دو تغییر وجود دارد. توجه کنید که گذر از c به a هم موجب بروز رقابت می‌گردد ولی غیر بحرانی است.

با افزودن یک سطر اضافی به جدول روند می‌توانیم یک تخصیص فاقد رقابت را بدست آوریم. استفاده از یک سطر اضافی چهارم تعداد متغیرهای حالت را افزایش نمی‌دهد، بلکه سیکل‌هایی را بین دو حالت پایدار به وجود خواهد آورد. به جدول روند اصلاح شده در شکل ۲۷-۹ توجه کنید. سه سطر اول همان وضعیت جدول سه سطری اصلی را دارند. به سطر چهارم، که با d نام‌گذاری شده و مجاور به a و c است 10 تخصیص یافته است. اکنون گذر از a به c باید از طریق d انجام شود، به این صورت که متغیرهای دودویی از $a = 00$ به $d = 10$ و سپس به $c = 11$ تغییر می‌نمایند. و بنابراین حالت رقابت پرهیز می‌شود. این کار با تغییر سطر a، ستون 11 به d و سطر d، ستون 11 به c انجام می‌شود به همین ترتیب گذر از c به a علیرغم وجود رقابت غیربحرانی در ستون 00 از طریق حالت ناپایدار d انجام می‌گردد.



(ب) نمودار گذر

(الف) جدول روند

شکل ۲۷-۹ . جدول روند با یک سطر اضافی

	x_1x_2 00	01	11	10
$a = 00$	(00)	01	10	(00)
$b = 01$	00	(01)	(01)	11
$c = 11$	10	(11)	(11)	(11)
$d = 10$	00	–	11	–

شکل ۲۸-۹. جدول گذر

جدول گذر مربوط به جدول روند همراه با تخصیص حالت دودویی در شکل ۲۸-۹ نشان داده شده است. دو خط تیره در سطر d بیانگر حالات نامشخصی هستند که می‌توانند به عنوان حالات بی‌اهمیت تلقی شوند. البته باید مراقب بود که تخصیص 10 به این مربعات صورت نگیرد تا بدین وسیله از ایجاد یک حالت پایدار ناخواسته در سطر چهارم جلوگیری به عمل آید.

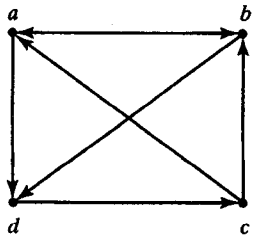
این مثال، استفاده از یک سطر اضافی در جدول روند را به منظور بدست آوردن یک انتساب فاقد رقابت نشان داد. این سطر اضافی به هیچ حالت پایدار خاصی منسوب نشد بلکه در عوض برای تبدیل یک رقابت بحرانی به سیکل مورد استفاده قرار گرفت که از میان حالات همجوار عبور می‌نمود. گاهی ممکن است وجود یک سطر اضافی برای جلوگیری از رقابت‌های بحرانی کافی نباشد و مجبور شویم دو یا چند سطر را به جدول روند اضافه کنیم. این کار در مثال بعدی نشان داده شده است.

مثال جدول روند چهار سطری

یک جدول روند با چهار سطر به حداقل دو متغیر نیاز دارد. هر چند دستیابی به یک تخصیص فاقد رقابت گاهی فقط با دو متغیر حالت دودویی امکان‌پذیر است، ولی در بسیاری از موارد نیاز به سطرهای اضافی برای اجتناب از رقابت‌های بحرانی، استفاده از سه متغیر حالت دودویی را دیکته می‌کند. به عنوان مثال جدول روند و نمودار گذر مربوطه‌اش را در شکل ۲۹-۹ ملاحظه نمایید. اگر گذری در جهت

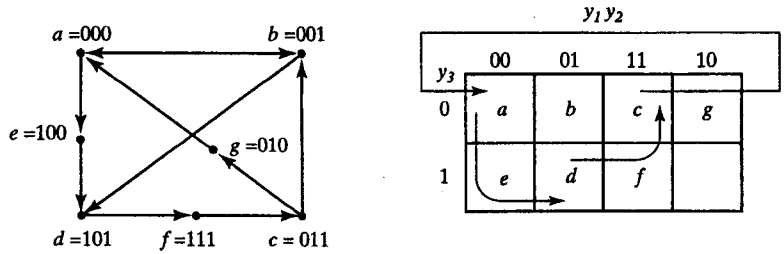
	00	01	11	10
a	b	(a)	d	(a)
b	(b)	d	(b)	a
c	(c)	a	b	(c)
d	c	(d)	(d)	c

(الف) جدول روند



(ب) نمودار گذر

شکل ۲۹-۹. مثال جدول روند چهار سطری



(ب) نمودار گذر (الف) تخصیص یا انتساب دودویی

شکل ۳۰-۹ . انتخاب سطرهای اضافی برای جدول روند

قطب (از b به d یا از c به a) وجود نداشته باشد یافتن یک تخصیص همجوار برای چهار گذر باقیمانده امکان‌پذیر است. با یک یا دو گذر قطری، برای تخصیص دو متغیر دودویی که نیاز همجواری را برآورده سازد، راهی وجود ندارد. بنابراین حداقل سه متغیر حالت دودویی لازم است.

شکل ۳۰-۹ یک نقشه تخصیص حالت را نشان می‌دهد که برای هر جدول روند چهار سطری مناسب است. حالات a، b، c و d حالات اصلی‌اند و حالات e، f و g حالات اضافی‌اند. حالاتی که در مربعات همجوار واقعند، تخصیص‌های مجاور خواهند داشت. عدد دودویی 001 به حالت b منتسب شده و همجوار با سه حالت اصلی دیگر است. گذر از a به d باید از طریق حالت اضافی e صورت گیرد تا چرخه یا سیکلی ایجاد شود به نحوی که هر بار تنها یک متغیر دودویی عوض شود. به طور مشابه گذر از c به a از طریق g انجام می‌گردد و گذر از d به c هم از f هدایت می‌شود. با توجه به تخصیص مربوط به نقشه، می‌توان طبق شکل ۳۱-۹ جدول چهار سطری را به جدول هفت سطری فاقد رقابت گسترش داد.

	00	01	11	10
000 = a	b	(a)	e	(a)
001 = b	(b)	d	(b)	a
011 = c	(c)	g	b	(c)
010 = g	–	a	–	–
110 –	–	–	–	–
111 = f	c	–	–	c
101 = d	f	(d)	(d)	f
100 = e	–	–	d	–

شکل ۳۱-۹ . انتساب حالات برای اصلاح جدول روند

توجه کنید که با وجود بودن هفت سطر، تنها چهار حالت پایدار وجود دارد. حالاتی که در سه سطر بدون دایره مشخص شده‌اند صرفاً برای ایجاد یک گذر بدون رقابت بین حالت‌های پایدار آمده‌اند.

مثال فوق روش انتخاب سطرهای اضافی در یک جدول روند برای دستیابی به یک تخصیص بی‌رقابت را نشان داد. یک نقشه انتساب حالات شبیه به نقشه شکل ۳۰-۹ (الف) می‌تواند در اغلب موارد مفید باشد. گاهی می‌توان از وارده‌های نامعین در جدول روند سود برد. در عوض اضافه کردن سطرها به جدول، ممکن است بتوان با هدایت بعضی از گذرهای حالات از طریق وارده‌های بی‌اهمیت، رقابت‌های بحرانی را حذف نمود. تخصیص واقعی تا رسیدن به یک تخصیص رضایت‌بخش فاقد رقابت با روش سعی و خطا اجرا می‌شود.

روش چند سطری

روشی که با اضافه کردن سطرهای اضافی برای دستیابی به یک گذر فاقد رقابت در جدول روند در دو مثال قبل تشریح شد را گاهی روش سطر مشترک می‌گویند. برای این دستیابی روش دومی نیز وجود دارد که البته به اندازه روش سطر مشترک کارایی ندارد ولی کاربرد آن آسانتر است. این روش، روش چند سطری نامیده می‌شود. در تخصیص چند سطری، هر حالت از جدول روند اولیه با دو یا چند ترکیب از متغیرهای حالت جایگزین می‌گردد. نقشه تخصیص حالت شکل ۳۲-۹ (الف) چنین انتسابی را نشان می‌دهد که قابل استفاده در جدول روند چهار سطری است. برای هر حالت پایدار دو متغیر حالت دودویی وجود دارد که هر یک متمم منطقی دیگری است. مثلاً حالت اولیه a با دو حالت معادل $a_1 = 000$ و $a_2 = 111$ جایگزین شده است. مقادیر خروجی که در اینجا نشان داده نشده‌اند، باید در a_1

	00	01	11	10
$000 = a_1$	b_1	(a_1)	d_1	(a_1)
$111 = a_2$	b_2	(a_2)	d_2	(a_2)
$001 = b_1$	(b_1)	d_2	(b_1)	a_1
$110 = b_2$	(b_2)	d_1	(b_2)	a_2
$011 = c_1$	(c_1)	a_2	b_1	(c_1)
$100 = c_2$	(c_2)	a_1	b_2	(c_2)
$010 = d_1$	c_1	(d_1)	(d_1)	c_1
$101 = d_2$	c_2	(d_2)	(d_2)	c_2

(ب) جدول روند

y_2y_3

y_1	00	01	11	10
0	a_1	b_1	c_1	d_1
1	c_2	d_2	a_2	b_2

(الف) انتساب دودویی

شکل ۳۲-۹. انتساب چند سطری

و a_2 یکسان باشند. توجه کنید که a_1 مجاور به b_1، c_2 و d_1، و a_2 مجاور به c_1، b_2 و d_2 و به طور مشابه هر حالت مجاور به سه حالت با حروف متفاوت است. رفتار مدار در قبال حالت داخلی a_1 و یا a_2 یکسان است و به همین ترتیب دیگر حالات نیز چنین خواصی دارند.

شکل ۳۲-۹ (ب) یک تخصیص چند سطری را برای جدول روند اولیه شکل ۲۹-۹ (الف) نشان می‌دهد. جدول گسترش یافته، از جایگزینی هر سطر از جدول روند اصلی با دو سطر به وجود می‌آیند. مثلاً سطر b با سطرهای b_1 و b_2 جایگزین شده و حالت پایدار b در هر دو سطر b_1 و b_2 در ستون‌های 00 و 11 آورده شده است. پس از این که همه حالت‌های باثبات وارد شدند، حالات ناپایدار با مراجعه به تخصیص‌های مشخص شده در نقشه قسمت (الف) پر می‌شوند. هنگام انتخاب حالت بعدی برای یک حالت فعلی، حالتی که مجاور حالت فعلی است انتخاب می‌شود. در جدول اولیه، حالات بعدی b برای ورودی‌های 10 و 01 به ترتیب a و d هستند. در جدول گسترش یافته حالات بعدی b_1 عبارتند از a_1 و d_2 زیرا این حالات مجاور b_1 می‌باشند. به طور مشابه حالات بعدی b_2 هم a_2 و d_1 می‌باشند زیرا این حالات با b_2 مجاورند.

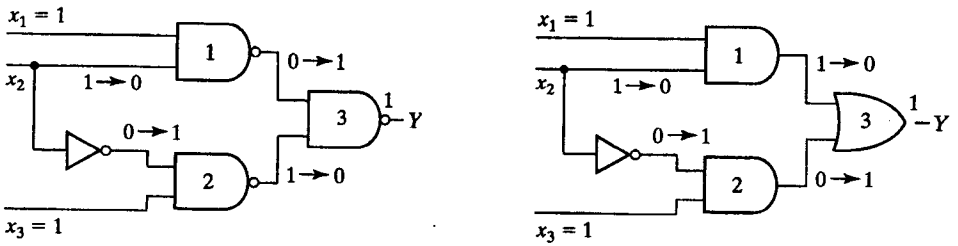
در تخصیص چند سطری، تغییر از یک حالت پایدار به حالت پایدار دیگر، همواره موجب تغییر فقط یک متغیر حالت دودویی می‌شود. هر حالت با ثبات دارای دو انتساب دودویی با خروجی‌های دقیقاً یکسان است. در هر لحظه از زمان تنها یکی از تخصیص‌ها به کار گرفته می‌شود. مثلاً اگر با حالت a_1 و ورودی 01 شروع کنیم و سپس ورودی را به 11، 01 و 00 تغییر دهیم و پس از آن به 01 بازگردیم، رشته حالات داخلی عبارت از a_1، d_1، c_1 و a_2 خواهد بود. اگر چه مدار با حالت a_1 شروع و در حالت a_2 خاتمه می‌یابد، ولی از دید رابطه ورودی- خروجی دو حالت a_1 و a_2 معادل a در جدول روند اصلی‌اند.

۹-۷ هزارد (HAZARD)

هنگام طراحی مدارهای ترتیبی غیرهمزمان، برای دستیابی به عملکردهای صحیح، باید محدودیت‌ها و احتیاط‌های لازم را در نظر گرفت. مدار باید در حالت اساسی کار کند، یعنی در هر لحظه فقط یک ورودی عوض شود و فاقد رقابت‌های بحرانی باشد. به علاوه، یک پدیده دیگر به نام هزارد hazard یا خروجی‌های تصادفی هم وجود دارد که ممکن است موجب اشکال در کار مدار گردد. هزاردها گذرهای سوئیچینگ ناخواسته‌ای هستند که ممکن است در خروجی مدارها ظاهر شوند زیرا مسیرهای مختلف در مدار دارای تأخیرهای انتشار متفاوت می‌باشند. هزاردها در مدارهای ترکیبی رخ می‌دهند و سبب تولید مقدار خروجی غلط موقت می‌گردند. هنگام وقوع این وضعیت احتمال گذر به یک حالت پایدار غلط وجود دارد. بنابراین لازم است مدار به لحاظ وقوع هزاردها مورد بررسی قرار گیرد تا مشخص شود آیا آنها موجب یک عملکرد نادرست می‌شوند یا خیر. برای حذف تأثیر آنها باید اقداماتی را به عمل آورد.

هزاردها در مدارهای ترکیبی

هزارد یا "خروجی تصادفی" وضعیتی است که در آن یک تغییر آنی در خروجی ایجاد کند در زمانی



(الف) مدار AND-OR

(ب) مدار NAND

شکل ۳۳-۹. مدارهایی با هزارد

که نباید ایجاد کند. مدار شکل ۳۳-۹ (الف) رخداد یک هزارد را نشان می‌دهد. فرض کنید هر سه ورودی در ابتدا برابر 1 هستند. این وضعیت موجب می‌شود تا خروجی گیت شماره 1 برابر 1، خروجی گیت شماره 2 برابر 0 و بنابراین خروجی مدار برابر 1 گردد. اکنون فرض کنید x_2 از 1 به 0 برود. خروجی گیت شماره 1 به 0 و خروجی گیت شماره 2 به 1 تغییر خواهد کرد ولی خروجی کل در همان 1 باقی خواهد ماند. با این وجود اگر تأخیر انتشار گیت وارونگر لحاظ شود، خروجی ممکن است برای یک لحظه به 0 برود. تأخیر انتشار در وارونگر ممکن است سبب شود تا قبل از این که خروجی گیت 2 به 1 تغییر کند، خروجی گیت 1 برابر 0 شود. در این لحظه هر دو ورودی گیت 3 برابر 0 هستند و این باعث می‌شود تا برای یک لحظه کوتاه، یعنی در مدت زمان تأخیر انتشار سیگنال ورودی x_2 در وارونگر، خروجی مدار 0 شود.

مدار شکل ۳۳-۹ (ب)، پیاده‌سازی NAND همان تابع بول است. این مدار نیز به دلیلی مشابه دارای هزارد است. چون گیت‌های 1 و 2 از نوع NAND هستند، خروجی آنها متمم گیت‌های AND نظیرشان است. وقتی x_2 از 1 به 0 می‌رود، هر دو ورودی گیت 3 ممکن است برابر 1 گردند و بنابراین تولید یک تغییر لحظه‌ای 0 را در وضعیت 1 بنمایند.

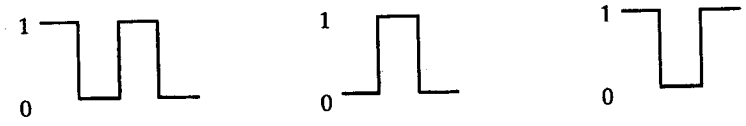
دو مدار شکل ۳۳-۹ تابع بول را به صورت جمع حاصلضرب‌ها پیاده‌سازی کرده‌اند:

$$Y = x_1x_2 + x_2'x_3$$

این نوع پیاده‌سازی ممکن است موجب رفتن خروجی به 0 به هنگام 1 بودن گردد. اگر مدار به صورت ضرب حاصل جمع‌ها پیاده‌سازی شود (بخش ۵-۳)، یعنی

$$Y = (x_1 + x_2')(x_2 + x_3)$$

آنگاه خروجی ممکن است به طور لحظه‌ای به هنگام 0 بودن به 1 برود. حالت اول را هزارد −1 ایستا و دومی را هزارد 0 ایستا می‌نامند. سومین نوع هزارد که هزارد پویا نام دارد موجب می‌شود تا خروجی به جای تغییر از 0 به 1 و یا 1 به 0، سه بار یا بیشتر تغییر حالت دهد. شکل ۳۴-۹ هر سه نوع هزارد را نشان می‌دهد. وقتی مداری به فرم جمع حاصلضرب‌ها یا گیت‌های AND−OR یا با NAND پیاده‌سازی می‌شود، حذف هزارد 1 ایستا عدم وقوع هزارد 0 ایستا و هزارد پویا را تضمین می‌کند.

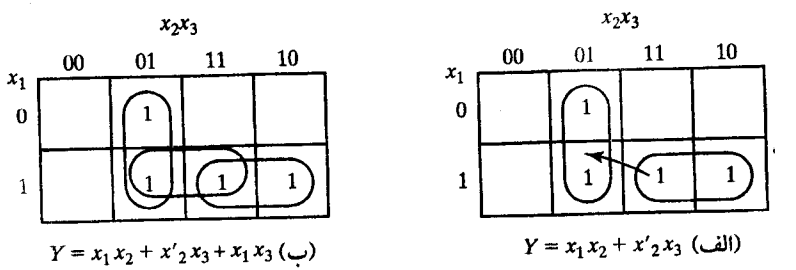


(الف) هزارد 1 ایستا (ب)هزارد 0ایستا (پ) هزارد پویا

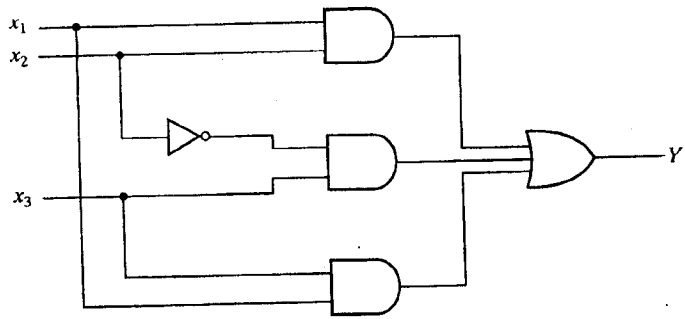
شکل ۳۴-۹. انواع هزارد

رخداد خروجی‌های هزارد را می‌توان با بررسی نقشه مدار آن آشکار کرد. برای تشریح موضوع، نقشه شکل ۳۵-۹ (الف) را ملاحظه نمایید که ترسیمی از تابع پیاده‌سازی شده در شکل ۳۳-۹ است. تغییر x_2 از 1 به 0 مدار را از مینترم 111 به مینترم 101 می‌برد. در اینجا هزارد وجود دارد زیرا متغیر ورودی، جمله حاصلضرب متفاوتی که دو مینترم را می‌پوشاند تولید خواهد کرد. مینترم 111 با جمله ضرب پیاده شده درگیت 1، و مینترم 101 به وسیله جمله ضرب پیاده شده درگیت 2 از شکل ۳۳-۹ پوشش یافته است. هر وقت مدار بخواهد از یک جمله ضرب به جمله ضرب دیگری برود، وقتی که هیچ یک از خروجی‌ها 1 نیستند، موجب تولید خروجی لحظه‌ای ناخواسته 0 می‌گردد.

چاره کار برای حذف هزارد پوشش دادن دو مینترم با جمله ضرب دیگری است که هر دو را در برگیرد. این کار در شکل ۳۵-۹ (ب) نشان داده شده است و در آن دو مینترمی که هزارد را تولید می‌کنند در یک جمله ضرب با یکدیگر ترکیب شده‌اند. مدار فاقد هزارد حاصل از این آرایش در شکل ۳۶-۹ دیده می‌شود. گیت اضافی در مدار جمله حاصلضرب x_1x_3 تولید می‌کند. به طور کلی هزاردها را در مدارهای ترکیبی می‌توان حذف کرد، به این ترتیب که دو مینترمی را که موجب هزارد می‌شوند با جمله ضرب مشترک در هر دو پوشش دهیم. لازمه حذف هزارد افزودن گیت‌های اضافی به مدار است.



شکل ۳۵-۹. نقشه‌های نشان دهندهٔ هزارد و حذف آن



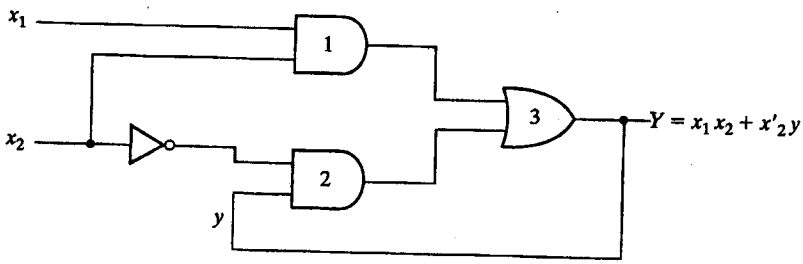
شکل ۳۶-۹. مدار فاقد هزارد

هزارد در مدارهای ترتیبی

در طراحی مدارهای ترکیبی معمولی که به همراه مدارهای ترتیبی همگام می‌آیند، چون سیگنال‌های لحظه‌ای خطا کلاً مشکل ساز نمی‌باشند، هزاردها مورد توجه نیستند. با این وجود اگر یک سیگنال لحظه‌ای نادرست در مدار ترتیبی غیرهمزمان پسخورد شود ممکن است سیستم را به حالت پایدار غلطی هدایت نماید. این مطلب در مثال شکل ۳۷-۹ نشان داده شده است. اگر مدار در حالت پایدارکلی $yx_1x_2 = 111$ باشد و ورودی x_2 از 1 به 0 تغییر نماید حالت پایدارکلی بعدی 110 خواهد بود. با این وجود، به علت هزارد، خروجی Y ممکن است یک لحظه به 0 برود. اگر قبل از این که خروجی وارونگر 1 شود، این سیگنال غلط به گیت 2 پسخورد گردد، خروجی گیت 2 در 0 مانده و مدار به حالت پایدارکل ناصحیح 010 خواهد رفت. این عملکرد نادرست را می‌توان با اضافه کردن یک گیت، طبق شکل ۳۶-۹، حذف کرد.

پیاده‌سازی با لچ SR

روشی دیگر برای اجتناب از هزارد در مدارهای ترتیبی غیرهمزمان، پیاده‌سازی با لچ‌های SR است. اعمال یک سیگنال آنی 0 به ورودی‌های S یا R در یک لچ NOR، تأثیری بر حالت مدار ندارد. به همین ترتیب اعمال یک سیگنال آنی 1 به ورودی‌های S و R روی حالت لچ تأثیر نخواهد داشت. در شکل ۳۳-۹ (ب) دیدیم که اگر هر دو ورودی گیت 3، 1 شوند، جمله جمع حاصلضرب‌های دو طبقه که با گیت‌های NAND پیاده‌سازی شده ممکن است هزارد 1 ایستا داشته و برای یک لحظه خروجی مدار را از 1 به 0 ببرد. اما اگر گیت 3 قسمتی از لچ باشد، سیگنال لحظه 1، اثری روی خروجی نخواهد داشت زیرا ورودی سومی از سمت متمم شده به گیت مذکور می‌آید و چون برابر 0 است خروجی در 1 باقی



(الف) نمودار منطقی

y \ x_1x_2	00	01	11	10
0			1	
1	1		1	1

(پ) نقشه برای Y

y \ x_1x_2	00	01	11	10
0	0	0	1	0
1	1	0	1	1

(ب) جدول گذر

شکل ۳۷-۹. هزارد در مدار ترتیبی همزمان

خواهد ماند. برای روشن شدن مطلب، لچ SR ساخته شده از NAND را با توابع بول زیر برای S و R در نظر بگیرید:

$$S = AB + CD$$
$$R = A'C$$

چون این یک لچ NAND است، باید متمم مقادیر را به ورودی‌ها اعمال کنیم:

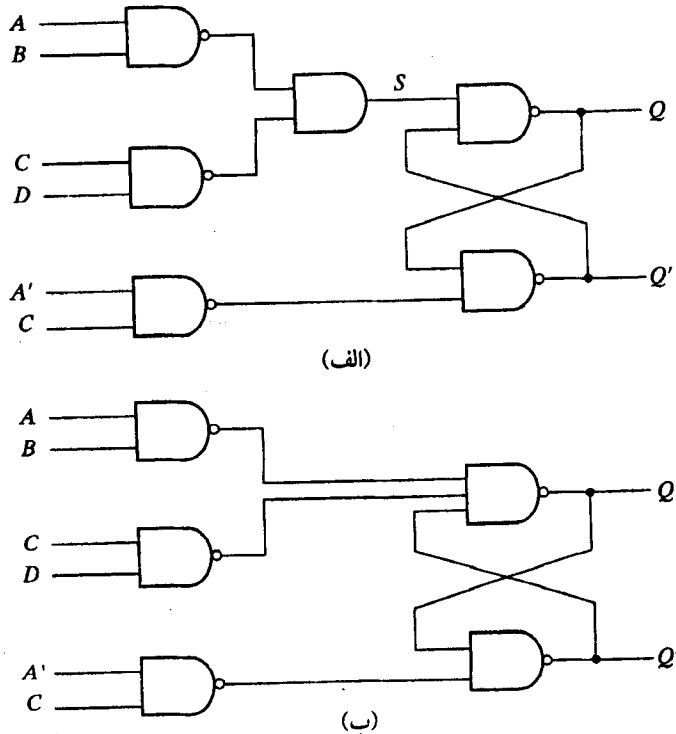
$$S = (AB + CD)' = (AB)'(CD)'$$
$$R = (A'C)'$$

این پیاده‌سازی در شکل ۳۸-۹ (الف) دیده می‌شود. S به کمک دو گیت NAND و یک گیت AND تولید می‌شود. تابع بول برای خروجی Q برابرست با

$$Q = (Q'S)' = [Q'(AB)'(CD)']'$$

این تابع در شکل ۳۸-۹ (ب) با گیت‌های NAND دو طبقه تولید شده است. اگر خروجی Q برابر 1 باشد، آنگاه Q′ برابر 0 است. اگر دو ورودی از سه ورودی یک لحظه به 1 بروند، گیت NAND مربوط به خروجی Q در 1 باقی خواهد ماند زیرا Q′ در 0 باقی می‌ماند.

شکل ۳۸-۹ (ب) نمونه مداری را نشان می‌دهد که در ساخت مدارهای ترتیبی غیرهمزمان به کار



شکل ۳۸-۹. یاده سازی لچ

می‌رود. دو گیت NAND تشکیل دهنده لچ معمولاً دو ورودی‌اند. با این وجود اگر توابع S و R حاوی دو جمله ضرب یا بیشتر در نمایش جمع حاصلضرب‌ها باشند، آنگاه گیت NAND متعلق به لچ SR سه یا چند ورودی خواهد داشت. به این ترتیب دو جمله در عبارت جمع حاصلضرب‌های اولیه عبارتند از AB و CD که هر یک با گیت NAND پیاده‌سازی شده‌اند و خروجی آنها به ورودی NAND لچ اعمال شده است. بنابراین هر متغیر حالت یک مدار دو طبقه با گیت‌های NAND است که هر جمله ضرب را در عبارت بولی اولیه S و R پیاده‌سازی می‌کند. طبقه دوم، اتصال تقاطعی لچ SR را با ورودی‌هایی که از خروجی هر گیت NAND در اولین طبقه می‌آیند، تشکیل می‌دهد.

هزاردهای اساسی

آنچه را که تاکنون بررسی نمودیم هزاردهای ایستا و پویا بودند. نوع دیگری از هزاردها در مدارهای ترتیبی غیرهمزمان وجود دارند که به آن هزارد اساسی می‌گویند. هزارد اساسی توسط تأخیرهای نابرابر که در طی دو یا چند مسیر از یک ورودی سرچشمه می‌گیرند، تولید می‌گردد. تأخیری بیش از حد در یک وارونگر در مقایسه با تأخیر ناشی از پسخورد می‌تواند عامل تولید چنین هزاردهایی باشد. هزاردهای اساسی را نمی‌توان همچون هزاردهای ایستا با افزودن گیت اصلاح کرد. مشکل مربوط به آنها را می‌توان با تنظیم تأخیر در مسیر مربوطه حل کرد. برای اجتناب از هزاردهای اساسی، هر حلقه پسخورد باید با دقت خاص مورد بررسی قرار گیرد و اطمینان حاصل شود که مسیر پسخورد به اندازه کافی در مقایسه با تأخیرهای دیگر سیگنال‌های واصله از پایانه‌های ورودی، طولانی است. به نظر می‌رسد این مشکل با توجه به وابستگی‌اش به مدار خاص و تأخیرهای متفاوت در مسیرهای مختلف، راه‌حل خاص در هر مدار را دارد.

۸-۹ مثال طراحی

اکنون در موقعیتی هستیم تا یک مثال طراحی، برای مدار ترتیبی غیرهمزمان را به طور کامل بررسی کنیم. این مثال می‌تواند به عنوان مرجعی برای طراحی مدارهای مشابه دیگر مورد استفاده قرار گیرد. ما روش طراحی را با اجرای مراحل پیشنهادی زیر که در انتهای بخش ۴-۹ لیست شده نشان خواهیم داد.

۱- بیان ویژگی‌ها یا مشخصات طرح
۲- ایجاد جدول روند اصلی
۳- کاهش جدول روند با ادغام سطرها
۴- تخصیص حالت دودویی فاقد رقابت
۵- تهیه جدول گذر و نقشه خروجی
۶- تهیه نمودار منطقی به کمک لچ‌های SR

ویژگی‌های طرح

می‌خواهیم یک فیلپ فلاپ T که در لبه منفی تریگر شود را طراحی کنیم. مدار دارای دو ورودی، T،

جدول ۶-۹. مشخصات حالتهای کلی

حالت	ورودی‌ها		خروجی	توضیحات
	T	C	Q	
a	1	1	0	خروجی اولیه 0 است
b	1	0	1	بعد از حالت a
c	1	1	1	خروجی اولیه 1 است
d	1	0	0	بعد از حالت c
e	0	0	0	بعد از حالات d یا f
f	0	1	0	بعد از حالات e یا a
g	0	0	1	بعداز حالات b یا h
h	0	1	1	بعد از حالات g یا c

C و یک خروجی Q است. اگر $T = 1$ باشد و پالس ساعت C از 1 به 0 تغییر کند (تریگر با لبه منفی)، خروجی متمم می‌شود. در غیر این صورت تحت هر وضعیت ورودی دیگر، خروجی بدون تغییر باقی می‌ماند. اگرچه این مدار می‌تواند به عنوان یک فیلپ فلاپ در مدارهای ترتیبی ساعت‌دار مورد استفاده قرار گیرد، ولی طراحی درونی آن یک مسئله غیرهمزمان است.

جدول روند اصلی

اگر در ابتدا جدولی از کلیه حالات ممکن در مدار تهیه کنیم، ایجاد جدول روند اولیه می‌تواند ساده‌تر شود. این کار در جدول ۶-۹ نشان داده شده است. ما با حالت $TC = 11$ شروع و آن را به حالت a منتسب می‌کنیم. وقتی که C از 1 به 0 می‌رود ولی T در 1 ثابت است، مدار به حالت b رفته و خروجی Q از 0 به 1 متمم می‌شود. تغییر دیگر در خروجی وقتی رخ می‌دهد که مدار از حالت c به حالت d برود. در این حالت $T = 1$، C از 1 به 0 و خروجی از 1 به 0 متمم می‌شود. چهار حالت دیگر در جدول خروجی را عوض نمی‌کنند زیرا T برابر 0 است. اگر Q از ابتدا 0 باشد، در 0 باقی می‌ماند، و اگر در 1 باشد حتی با تغییر ورودی ساعت در همان 1 باقی خواهد ماند. این اطلاعات شش حالت کلی را نتیجه می‌دهند. توجه داشته باشید که گذر همزمان دو متغیر ورودی مثل 01 به 10 به حساب نیامده‌اند زیرا آنها شرایط عملکرد مد اساسی را زیر سؤال می‌برند.

جدول روند اصلی در شکل ۳۹-۹ دیده می‌شود. اطلاعات جدول روند را می‌توان مستقیماً از شرایط لیست شده در جدول ۶-۹ بدست آورد. ما ابتدا یک مربع مربوط به هر حالت با ثبات در هر سطر را، طبق جدول، پر می‌کنیم. سپس در مربع‌هایی که ورود آنها با ورودی‌های متناظر در حالت پایدار دو اختلاف دارد خط تیره می‌گذاریم. آنگاه شرایط بی‌ثباتی با استفاده از اطلاعات لیست شده در زیر ستون توضیحات در جدول ۶-۹ معین می‌شود.

ادغام جدول روند

برای ادغام سطرهای جدول روند اصلی، ابتدا همه جفت حالات سازگار بدست می‌آیند. این کار با به

	TC 00	01	11	10
a	$-,-$	$f,-$	$(a),0$	$b,-$
b	$g,-$	$-,-$	$c,-$	$(b),1$
c	$-,-$	$h,-$	$(c),1$	$d,-$
d	$e,-$	$-,-$	$a,-$	$(d),0$
e	$(e),0$	$f,-$	$-,-$	$d,-$
f	$e,-$	$(f),0$	$a,-$	$-,-$
g	$(g),1$	$h,-$	$-,-$	$b,-$
h	$g,-$	$(h),1$	$c,-$	$-,-$

شکل ۳۹-۹ . جدول روند اصلی

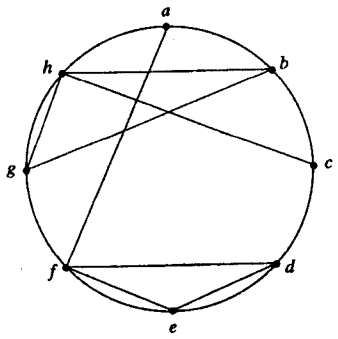
کارگیری جدول ایجاب شکل ۴۰-۹ انجام می‌شود. مربعاتی که دارای علامت تأیید (✓) هستند جفت‌های سازگار را تعریف می‌کنند:

$$(a,f)\quad (b,g)\quad (b,h)\quad (c,h)\quad (d,e)\quad (d,f)\quad (e,f)\quad (g,h)$$

سازگارهای ماکزیمال از روی نمودار ادغام شکل ۴۱-۹ بدست می‌آیند. الگوهای هندسی موجود در

	a	b	c	d	e	f	g
b	$a,c\times$						
c	$\times$	$b,d\times$					
d	$b,d\times$	$\times$	$a,c\times$				
e	$b,d\times$	$e,g\times$ $b,d\times$	$f,h\times$	✓			
f	✓	$e,g\times$ $a,c\times$	$f,h\times$ $a,c\times$	✓	✓		
g	$f,h\times$	✓	$b,d\times$	$e,g\times$ $b,d\times$	$\times$	$e,g\times$ $f,h\times$	
h	$f,h\times$ $a,c\times$	✓	✓	$d,e\times$ $c,f\times$	$e,g\times$ $f,h\times$	$\times$	✓

شکل ۴۰-۹ . جدول ایجاب



شکل ۴۱-۹. نمودار ادغام

نمودار مذکور شامل دو مثلث و دو خط راست است. مجموعه سازگار ماکزیمال عبارت است از:

$$(a, f) \quad (b, g, h) \quad (c, h) \quad (d, e, f)$$

در این مثال خاص، مجموعه سازگارهای مینیمال، همان مجموعه سازگارهای ماکزیمال است. توجه کنید که شرط بسته بودن برقرار است زیرا هر چند h و f تکرار شده‌اند، ولی مجموعه فوق شامل هر هشت حالت اصلی جدول روند می‌باشد. شرط پوششی بودن نیز بر آورده شده است زیرا کلیه زوج‌های سازگار، طبق جدول ایجاب، دارای حالات موجب نیستند.

جدول کاهشی یافته در شکل ۴۲-۹ دیده می‌شود. شکل (الف) سمبل‌های حالت اولیه را داراست ولی سطرهای مربوطه را ادغام نموده است. مثلاً، حالت a و f سازگارند و به یک سطر کاهش یافته‌اند ولی سمبل‌های حرفی اولیه در آن حفظ شده است. به طور مشابه مجموعه سازگار دیگر حالات برای ادغام جدول روند در چهار سطر به کار رفته‌اند و هشت سمبل اولیه را نگهداشته‌اند. روش دیگر ترسیم جدول روند ادغام شده است که در قسمت (ب) شکل ملاحظه می‌شود. ما در اینجا به کلیه حالت‌های پایدار در سطر ادغام شده یک سمبل حرفی مشترک نسبت داده‌ایم. بنابراین، سمبل f با a، و g و h با b جایگزین و به همین ترتیب در سطرهای بعدی عمل شده است. دومین جدول به وضوح یک جدول روند چهار حالته را فقط با چهار سمبل حرفی نشان می‌دهد.

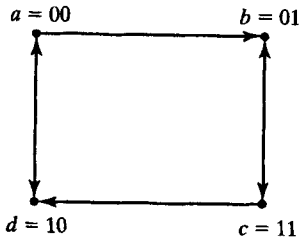
	TC 00	01	11	10
a, f	e, –	(f), 0	(a), 0	b, –
b, g, h	(g), 1	(h), 1	c, –	(b), 1
c, h	g, 1	(h), 1	(c), 1	d, –
d, e, f	(e), 0	(f), 0	a, –	(d), 0

(الف)

	TC 00	01	11	10
a	d, –	(a), 0	(a), 0	(b), –
b	(b), 1	(b), 1	c, –	(b), 1
c	b, –	(c), 1	(c), 1	d, –
d	(d), 0	(d), 0	a, –	(d), 0

(ب)

شکل ۴۲-۹. جدول روند کاهش یافته



شکل ۴۳-۹ . نمودار گذر

تخصیص حالت و جدول گذر

قدم بعدی در طراحی یافتن تخصیص بی‌رقابت برای چهار حالت پایدار موجود در جدول روند کاهش یافته است. برای یافتن یک تخصیص همجوار مناسب، ما نمودار گذر آن را طبق شکل ۴۳-۹ رسم می‌کنیم. در این مثال چون خطوط قطری وجود ندارد بدون نیاز به حالات اضافی می‌توان یک تخصیص همجوار مناسب بدست آورد.

با جایگزینی تخصیص دودویی نمودار گذر در جدول روند کاهش یافته، جدول گذر شکل ۴۴-۹ بدست می‌آید. نقشه خروجی از جدول روند کاهش یافته حاصل می‌شود. خطوط تیره در بخش خروجی مقادیر تخصیصی بر طبق قواعد برپا شده در بخش ۴-۹ است.

نمودار منطقی

مداری که باید طراحی شود دارای دو متغیر حالت Y_1 و Y_2 و یک خروجی Q است. نقشه خروجی شکل ۴۴-۹ نشان می‌دهد که Q برابر با متغیر حالت y_2 می‌باشد. پیاده‌سازی مدار به دو لچ SR نیاز دارد که هر کدام متعلق به یک متغیر حالت است. نقشه ورودی‌های S و R دو لچ در شکل ۴۵-۹ آورده

TC

y_1y_2	00	01	11	10
$a = 00$	10	(00)	(00)	01
$b = 01$	(01)	(01)	11	(01)
$c = 11$	01	(11)	(11)	10
$d = 10$	(10)	(10)	00	(10)

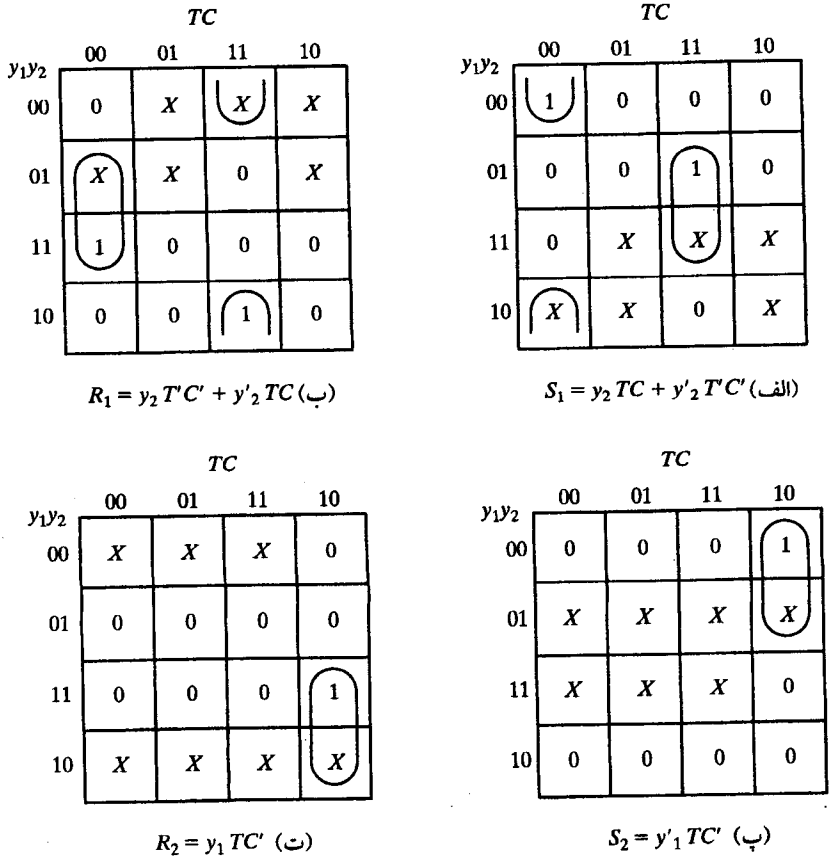
(الف) جدول گذر

TC

y_1y_2	00	01	11	10
00	0	0	0	X
01	1	1	1	1
11	1	1	1	X
10	0	0	0	0

(ب) نقشه خروجی $Q = y_2$

شکل ۴۴-۹ . جدول گذر و نقشهٔ خروجی

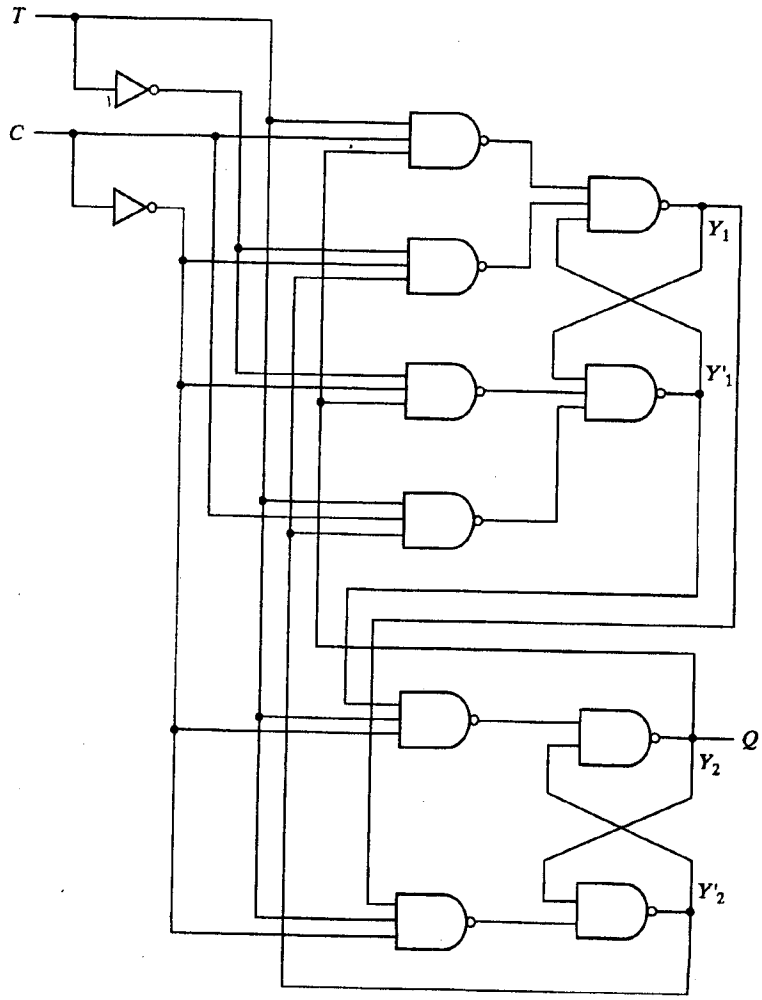


شکل ۴۵-۹ . نقشه‌های ورودی‌های لچ

شده‌اند. نقشه‌ها از اطلاعات موجود در جدول گذر و با استفاده از شرایط مشخص شده در جدول تحریک لچ در شکل ۱۴-۹ (ب) بدست آمده‌اند.

نمودار منطقی مدار در شکل ۴۶-۹ رسم شده است. در اینجا دو لچ NAND با دو یا سه ورودی در هر گیت به کار رفته است. این پیاده‌سازی، مطابق با الگوی ایجاد شده در بخش ۷-۹ همراه با شکل ۳۸-۹ (ب) انجام شده است.

این مثال، پیچیدگی موجود در طراحی مدارهای ترتیبی همگام را نشان می‌دهد. برای بدست آوردن نمودار نهایی مدار می‌بایست ده نمودار تهیه کنیم. اگر چه بیشتر مدارهای دیجیتال همزمان هستند اما گاهی با عملیات غیرهمزمان هم مواجه می‌شویم. خصوصیات مبنای ارائه شده در این فصل برای درک کامل نحوه عملکرد داخلی مدارهای دیجیتال ضروری می‌باشند.

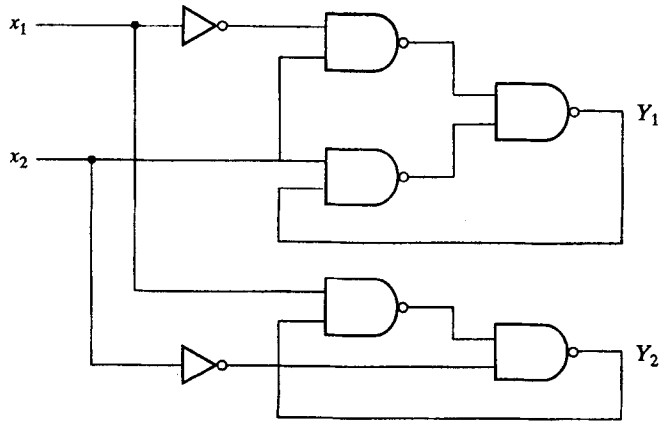


شکل ۴۶-۹. نمودار منطقی یک فلیپ فلاپ T که در لبهٔ منفی تریگر می‌شود.

مسائل

۹-۱ (الف) تفاوت بین مدارهای ترتیبی همزمان و غیرهمزمان را شرح دهید.
(ب) عملیات مد اساسی را تعریف کنید.
(پ) تفاوت بین حالات پایدار و ناپایدار را شرح دهید.
(ت) تفاوت بین یک حالت داخلی و یک حالت کلی چیست؟

۹ـ۲ جدول گذر مدار ترتیبی غیرهمزمان شکل (م ۹-۲) را بدست آورید. رشته حالات داخلی Y_1Y_2 را برای رشته ورودی‌های زیر مشخص کنید. x_1x_2 : 00, 10, 11, 01, 11, 10, 00



شکل (م ۹-۲)

۹ـ۳ یک مدار ترتیبی غیرهمزمان با توابع تحریک و خروجی زیر تعریف شده است:

$$Y = x_1x_2' + (x_1 + x_2')y$$
$$z = y$$

(الف) نمودار منطقی مدار را بکشید.
(ب) جدول گذر و نقشه خروجی را بکشید.
(پ) جدول روند دو حالتی را تهیه کنید.
(ت) رفتار مدار را شرح دهید.

۹ـ۴ یک مدار ترتیبی غیرهمزمان دارای دو حالت داخلی و یک خروجی است توابع تحریک و خروجی توصیف کننده مدار به این شرح هستند.

$$Y_1 = x_1x_2 + x_1y_2' + x_2'y_1$$
$$Y_2 = x_2 + x_1y_1'y_2 + x_1'y_1$$
$$z = x_2 + y_1$$

(الف) نمودار منطقی مدار را رسم کنید.
(ب) جدول گذر و نقشه خروجی را بدست آورید.
(پ) جدول روند را برای مدار بدست آورید.

	x_1x_2 00	01	11	10
a	(a), 0	b , –	c , –	(a), 1
b	a , –	(b), 0	(b), 0	c, –
c	a , –	b , –	(c), 1	(c), 0

شکل (م ۹-۵)

۹-۵ جدول روند شکل (م ۵-۹) را با تخصیص مقادیر دودویی $a = 00$، $b = 11$ و $c = 01$ به جدول گذر تبدیل کنید.

(الف) به چهارمین حالت اضافی مقادیری تخصیص دهید تا از رقابت بحرانی اجتناب شود.

(ب) به حالات بی‌اهمیت خروجی‌هایی تخصیص دهید تا از خروجی‌های نادرست لحظه‌ای جلوگیری شود.

(پ) نمودار منطقی مدار را بدست آورید.

۹-۶ جدول (م ۶-۹) را بررسی کرده و همه شرایط رقابت را معین نمایید و بگویید بحرانی‌اند یا غیربحرانی. آیا چرخه یا سیکلی هم در جدول وجود دارد یا خیر.

y_1y_2 \ x_1x_2	00	01	11	10
00	10	(00)	11	10
01	(01)	00	10	10
11	01	00	(11)	(11)
10	11	00	(10)	(10)

شکل (م ۶-۹)

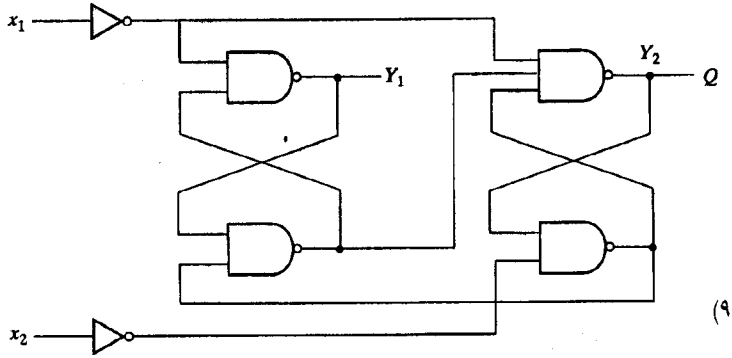
۹-۷ لچ SR را همراه با کنترل در شکل ۵-۵ تحلیل نمایید. جدول گذر را بدست آورده و نشان دهید که وقتی هر سه ورودی 1 باشند مدار بی‌ثبات یا ناپایدار است.

۹-۸ نمودار شکل ۵-۵ (الف) را برای تبدیل به یک لچ نوع JK اصلاح کنید. برای انجام این کار دو مسیر پسخورد از خروجی‌ها به ورودی‌ها وصل نمایید. نشان دهید که وقتی $J = K = 1$ و $C = 1$ است مدار ناپایدار می‌باشد.

۹-۹ برای مدار ترتیبی غیرهمزمان شکل (م ۹-۹)،

(الف) توابع بول را برای خروجی‌های Y_1، Y_2 دو لچ SR بدست آورید. دقت کنیدکه ورودی S دومین لچ $x'_1y'_1$ می‌باشد.

(ب) جدول گذر و نقشه خروجی مدار را بدست آورید.



شکل (م ۹-۹)

۹-۱۰ مدار تعریف شده در مسئله ۹-۳ را با لچ SR نوع NOR پیاده‌سازی کنید. پیاده‌سازی را با لچ SR نوع NAND هم تکرار نمایید.

۹-۱۱ مدار تعریف شده در مسئله ۹-۴ را با لچ‌های SR نوع NAND پیاده‌سازی نمایید.

۹-۱۲ یک جدول روند اصلی برای یک مدار با دو ورودی x_1 و x_2 و دو خروجی z_1 و z_2 بدست آورید به نحوی که چهار شرط زیر برقرار باشد:

(الف) وقتی $x_1x_2 = 00$ است، خروجی $z_1z_2 = 00$ باشد.

(ب) اگر $x_1 = 1$ و x_2 از 0 به 1 تغییر نماید، خروجی $z_1z_2 = 01$ باشد.

(پ) اگر $x_2 = 1$ و x_1 از 0 به 1 عوض شود دو خروجی $z_1z_2 = 10$ خواهند بود.

۹-۱۳ یک چراغ ترافیک در تقاطع یک جاده با راه‌آهن نصب شده است. این چراغ با دو کلید که روی ریل به فاصله یک مایلی از نقطه تقاطع نصب‌اند کنترل می‌شود. وقتی قطار روی کلیدی واقع می‌شود، آن کلید وصل می‌گردد، در غیر این صورت قطع خواهد شد. در یک مایلی قطار از تقاطع، چراغ از سبز (منطق 0) به قرمز (منطق 1) تبدیل می‌گردد. وقتی که قطار یک مایل از تقاطع دور شد لامپ به رنگ سبز باز می‌گردد. توجه کنید که طول قطار کم‌تر از 2 مایل است.

(الف) جدول روند اصلی را برای مدار بدست آورید.

(ب) نشان دهید که جدول روند قابل کاهش به 4 سطر است.

۹-۱۴ می‌خواهیم یک مدار ترتیبی با دو ورودی x_1 و x_2 و یک خروجی z طراحی کنیم. ابتدا هر دو ورودی برابر 0 اند. وقتی x_1 یا x_2 برابر 1 شوند، $z = 1$ می‌گردد. اگر دومین ورودی هم 1 شود، خروجی به 0 تغییر می‌کند. با بازگشت مدار به حالت اولیه، خروجی در 0 باقی می‌ماند.

(الف) برای مدار یک جدول روند اصلی بدست آورید و نشان دهید که این جدول قابل کاهش به شکل (م ۹-۱۴) است.

(ب) طراحی مدار را تکمیل نمایید.

	00	01	11	10
a	(a), 0	(a), 1	b, –	(a), 1
b	a, –	(b), 0	(b), 0	(b), 0

شکل (م ۹-۱۴)

۹-۱۵ مقادیر خروجی را به حالات بی‌اهمیت در جدول روند شکل (م ۹-۱۵) طوری تخصیص دهید که از

	00	01	11	10
a	(a), 0	b, –	–, –	d, –
b	a, –	(b), 1	(b), 1	c, –
c	b, –	–, –	b, –	(c), 0
d	c, –	(d), 1	c, –	(d), 1

(الف)

	00	01	11	10
a	(a), 0	b, –	b, –	(a), 0
b	a, –	(b), 0	(b), 1	c, –
c	b, –	d, –	(c), 1	(c), 1
d	(d), 0	(d), 1	c, –	a, –

(ب)

شکل (م ۹-۱۵)

پالس‌های گذرا در خروجی اجتناب شود.

۱۶ - ۹ با استفاده از روش جدول ایجاب، نشان دهید که جدول حالت ۷-۵ بیش از این کاهش نمی‌یابد.

۱۷ - ۹ تعداد حالات جدول حالت مسئله ۱۲-۵ را کاهش دهید. از جدول ایجاب استفاده کنید.

۱۸ - ۹ هر یک از جدول روند اصلی در شکل (م ۱۸-۹) را ادغام کنید. این کار را به طریق زیر اجرا نمایید.

(الف) همه جفت‌های سازگار را به کمک جدول ایجاب بدست آورید.

(ب) سازگارهای ماکزیمال را با نمودار ادغام بدست آورید.

(پ) مجموعه مینیمالی از سازگارهایی که همه حالات را پوشش داده و بسته باشند، پیدا کنید.

	00	01	11	10
a	(a), 1	f, –	–, –	e, –
b	c, –	–, –	j, –	(b), 0
c	(c), 0	d, –	–, –	b, –
d	c, –	(d), 0	g, –	–, –
e	a, –	–, –	g, –	(e), 1
f	a, –	(f), 1	g, –	–, –
g	–, –	d, –	(g), 0	k, –
h	(h), 0	d, –	–, –	k, –
j	–, –	f, –	(j), 1	b, –
k	a, –	–, –	j, 1	(k), 0

(ب)

	00	01	11	10
a	(a), 0	b, –	–, –	e, –
b	a, –	(b), 0	c, –	–, –
c	–, –	d, –	(c), 0	h, –
d	a, –	(d), 1	–, –	–, –
e	a, –	–, –	f, –	(e), 0
f	–, –	g, –	(f), 0	h, –
g	a, –	(g), 0	–, –	–, –
h	a, –	–, –	–, –	(h), 0

(الف)

شکل (م ۱۸-۹)

۱۹ - ۹ (الف) یک انتساب حالت دودویی برای جدول روند کاهش یافته شکل (م ۱۹-۹) بدست آورید. از شرایط بحرانی دوری نمایید.

x_1x_2

	00	01	11	10
a	(a), 0	(a), 1	b, –	d, –
b	a, –	(b), 0	(b), 0	c, –
c	a, –	–, –	d, –	(c), 0
d	a, –	a, –	(d), 1	(d), 1

شکل (م ۱۹-۹)

۹-۲۰ یک تخصیص حالت فاقد رقابت بحرانی برای جدول روند شکل (م ۹-۲۰) پیدا کنید.

	00	01	11	10
a	(*a*)	*d*	(*a*)	*c*
b	*a*	(*b*)	(*b*)	*d*
c	*d*	(*c*)	*b*	(*c*)
d	(*d*)	(*d*)	*e*	(*d*)
e	*f*	*c*	(*e*)	*c*
f	(*f*)	*b*	*a*	(*f*)

شکل (م ۹-۲۰)

۹-۲۱ جدول روند شکل (م ۹-۲۱) را ملاحظه نمایید.

(الف) نمودار گذر را بدست آورده و نشان دهید که برای یک تخصیص حالت دودویی فاقد رقابت سه متغیر لازم است.

(ب) باکمک تخصیص روش چندسطری جدول روند گسترش یافته را طبق شکل (۳۲-۹الف) بدست آورید.

۹-۲۲ مداری پیدا کنید که فاقد هزارد ایستا یا استاتیک بوده و تابع بولی زیر را پیاده‌سازی نمایید.

$$F(A, B, C, D) = \Sigma(0, 2, 6, 7, 8, 10, 12)$$

۹-۲۳ نمودار منطقی عبارت ضرب حاصل جمع‌های زیر را رسم کنید.

$$Y = (x_1 + x_2')(x_2 + x_3)$$

نشان دهید که وقتی x_1 و x_3 هر دو برابر 0 اند و x_2 از 0 به 1 می‌رود یک هزارد 0 ایستا تولید می‌شود. با افزودن یک گیت OR راهی برای حذف آن بدست آورید.

۹-۲۴ عبارات بولی یک لچ SR به قرار زیراند:

$$S = x_1'x_2'x_3 + x_1x_2x_3$$
$$R = x_1x_2' + x_2x_3'$$

نمودار مدار را با استفاده از حداقل گیت‌های NAND مشخص کنید.

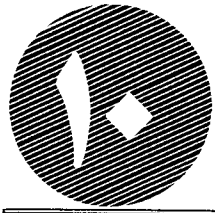
۹-۲۵ طراحی مدار مسئله ۹-۱۳ را تکمیل نمایید.

مراجع

1. Hill, F. J., and G. R. Peterson. 1981. *Introduction to Switching Theory and Logical Design,* 3rd Ed. New York: John Wiley.

2. Kohavi, Z. 1978. *Switching and Automata Theory,* 2nd Ed. New York: McGraw-Hill.

3. McCluskey, E. J. 1986. *Logic Design Principles.* Englewood Cliffs, NJ: Prentice-Hall.



مدارهای مجتمع دیجیتال

۱۰-۱ مقدمه

در بخش ۸-۲ مدار مجتمع (IC) و خانواده‌های منطقی دیجیتال معرفی شدند. این فصل مدارهای الکترونیک را در هر مدار مجتمع منطقی مورد بررسی قرار داده عملکرد الکتریکی آنها را مورد تجزیه و تحلیل قرار می‌دهد. در ادامه بحث اطلاعات اولیه‌ای در مورد مدارهای الکتریکی لازم است.

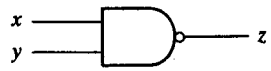
خانواده‌های مدارهای مجتمع مورد مطالعه به قرار زیرند:

منطق مقاومت ترانزیستور	RTL
منطق دیود ترانزیستور	DTL
منطق ترانزیستور- ترانزیستور	TTL
منطق کوپلاژامیتر	ECL
فلز- اکسید نیمه‌هادی	MOS
فلز- اکسید نیمه‌هادی متمم (یامکمل)	CMOS

دو خانواده اول، RTL و DTL دارای ارزش تاریخی‌اند زیرا در حال حاضر در طراحی سیستم‌های دیجیتال به کار نمی‌روند. RTL اولین خانواده تجاری بود که به صورت گسترده‌ای مورد استفاده قرار گرفت و چون نقطه شروع خوبی برای تشریح اعمال گیت‌های منطقی می‌باشد از آن استفاده شده است. مدارهای DTL با TTL جایگزین شدند. در واقع TTL فرم اصلاح شده گیت DTL است. طرز کار گیت TTL پس از تحلیل گیت DTL قابل فهم‌تر است. TTL، ECL و CMOS تعداد زیادی از مدارهای SSI، MSI، LSI و VLSI را شامل می‌شوند.

مدار پایه در هر یک از خانواده‌ها، گیت NAND یا NOR است. این مدار پایه بلوک ساختار اصلی است که دیگر اجزاء دیجیتال پیچیده‌تر از آن بدست می‌آیند. هر خانواده از مدارهای مجتمع دارای برگه

ورودی ها		خروجی
x	y	z
L	L	H
L	H	H
H	L	H
H	H	L



شکل ۱-۱۰ . گیت NAND منطق مثبت

اطلاعاتی است که همه مدارهای مجتمع را لیست می‌نماید. اختلاف بین توابع منطقی حاصل از هر خانواده منطقی به اندازه اختلاف در مشخصات یک گیت پایه نیست.

گیت‌های NAND و NOR معمولاً به وسیله توابع بولی تعریف می‌شوند و این توابع عباراتی از متغیرهای دودویی را پیاده‌سازی می‌کنند. هنگام تحلیل این گیت‌ها به عنوان مدارهای الکترونیک، لازم است رابطه ورودی- خروجی آنها برحسب ولتاژ سطح بالا که با H نشان داده می‌شود و سطح پایین که با L نمایش داده می‌شود، مورد تفحص قرار گیرد. همانطور که در بخش ۸-۲ ذکر شد تخصیص 1 به H، سیستم منطق مثبت و تخصیص 1 به L، سیستم منطقی منفی را تعریف می‌کند. جدول درستی رفتار گیت NAND برحسب H و L با منطق مثبت در شکل ۱-۱۰ ملاحظه می‌شود. توجه دارید، مادامی که یک یا چند ورودی در سطح پایین است، خروجی در سطح بالاست. رفتار یک گیت NAND با منطق مثبت در مقابل یک سیگنال بالا و پایین به طریق زیر بیان می‌شود:

اگر هر یک از ورودی‌های گیت NAND در سطح پایین باشد، خروجی در سطح منطقی بالاست.

اگر همه ورودی‌های گیت NAND بالا باشد، خروجی پایین است.

جدول درستی یک گیت NOR با منطق مثبت در شکل ۲-۱۰ دیده می‌شود.

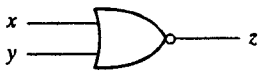
خروجی گیت NOR هنگامی پایین است که یک یا چند ورودی آن بالا باشد. خروجی هنگامی بالاست که همه ورودی‌ها پایین باشد. رفتار یک گیت NOR با منطق مثبت برحسب سطح سیگنال‌ها را چنین می‌توان بیان کرد:

اگر هر یک از ورودی‌های گیت NOR در سطح بالا باشد، خروجی پایین است.

اگر همه ورودی‌های گیت NOR در سطح پایین باشند خروجی بالاست.

این عبارات را برای NAND و NOR باید به خاطر سپرد زیرا از آنها در طول تحلیل گیت‌های الکترونیک در این فصل استفاده خواهد شد.

ورودی		خروجی
x	y	z
L	L	H
L	H	L
H	L	L
H	H	L



شکل ۲-۱۰ . گیت NOR منطق منفی

یک ترانزیستور پیوندی دو قطبی (BJT) می‌تواند از نوع *npn* یا *pnp* باشد. برعکس ترانزیستورهای اثر میدان (FET) را تک قطبی می‌گویند. عملکرد ترانزیستور دو قطبی براساس جریان دو نوع حامل بار یعنی الکترون و حفره است. ترانزیستور تک قطبی به جریان یک نوع بار که ممکن است الکترون (کانال n) یا حفره‌ها (کانال P) باشد وابسته است. چهار خانواده اول یعنی RTL، DTL، TTL و ECL از ترانزیستورهای دو قطبی استفاده می‌کنند. دو خانواده آخر، یعنی MOS و CMOS، نوعی از ترانزیستور تک قطبی به نام ترانزیستور اثر میدان فلز -اکسید- نیمه‌هادی را به کار می‌برند و به طور خلاصه به آن MOSFET یا خلاصه‌تر MOS گویند.

در این فصل، رایج‌ترین مشخصه‌های خانواده‌های منطقی را با هم مقایسه می‌کنیم. آنگاه خواص ترانزیستور را توصیف کرده و گیت‌های پایه را در خانواده‌های منطقی تحلیل می‌نماییم. سپس عملکرد ترانزیستور MOS را توضیح داده و گیت‌های پایه دو خانواده را معرفی خواهیم کرد.

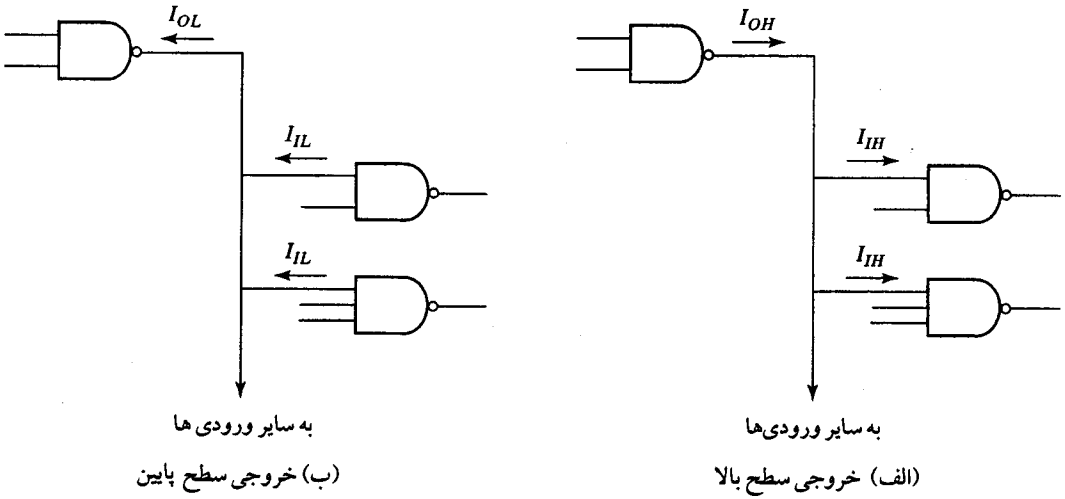
۲-۱۰ مشخصات ویژه

مشخصات خانواده‌های منطقی دیجیتال معمولاً با تحلیل مدار گیت پایه در هر خانواده با هم مقایسه می‌شوند. مهمترین پارامترهای مورد ارزیابی و مقایسه عبارتند از گنجایش خروجی، توان مصرفی، تأخیر انتشار و حد پارازیتی. ما ابتدا خصوصیات این پارامترها را شرح داده و سپس آنها را برای مقایسه خانواده‌های منطقی مجتمع به کار خواهیم برد.

گنجایش خروجی

گنجایش خروجی یک گیت نشان دهنده تعداد بارهای استانداردی است که بدون تخریب عملکرد معمولی آن به خروجی گیت وصل شود. معمولاً بار استاندارد به عنوان جریان لازم به وسیله گیت بعدی از همان خانواده تعریف می‌گردد. گاهی اوقات جمله بار شدن به جای گنجایش خروجی به کار می‌رود. جمله فوق به این علت به کار رفته که خروجی یک گیت مقدار محدودی از جریان را تهیه می‌نماید و بیش از آن قادر به ادامه کار صحیح نمی‌باشد. در این حالت گوییم خروجی فرابار شده است. معمولاً خروجی یک گیت به ورودی گیت دیگر وصل می‌شود. هر ورودی مقدار معینی جریان را از خروجی مصرف می‌کند در نتیجه هر اتصال اضافی به مقدار بار گیت منبع خواهد افزود. قوانین بار شدن حداکثر بار مجاز را برای خروجی هر گیت از یک خانواده تعریف می‌کنند. تجاوز از مقدار بار حداکثر ممکن است سبب آسیب دیدگی شود زیرا مدار قادر به تحویل توان تقاضا شده نمی‌باشد. گنجایش خروجی تعداد حداکثر ورودی‌هایی است که به یک خروجی قابل اتصال می‌باشد و با یک عدد بیان می‌شود.

گنجایش خروجی را با توجه به جریان موجود در خروجی یک گیت و مقدار جریان لازم برای ورودی محاسبه می‌کنند. به اتصالات شکل ۳-۱۰ توجه نمایید. خروجی یک گیت به یک یا چند ورودی گیت دیگر وصل شده است. خروجی گیت در شکل ۳-۱۰ (الف) در سطح بالای ولتاژ قرار دارد. این خروجی یک منبع جریان I_{OH} را برای همه ورودی‌های گیت‌های دیگر فراهم می‌نماید. هر ورودی



شکل ۳-۱۰ . محاسبه گنجایش خروجی

برای انجام کار صحیح نیاز به یک جریان I_{IH} دارد. به طور مشابه در شکل ۳-۱۰ (ب) خروجی گیت در سطح پایین ولتاژ است. در این حالت جریان فروکشی از تمام ورودی‌های گیت‌ها به خروجی گیت منبع به وجود خواهد آمد. هر ورودی گیت جریان I_{IL} خود را فراهم می‌سازد. گنجایش خروجی یک گیت از نسبت کوچکتر I_{OL}/I_{IL} یا I_{OH}/I_{IH} محاسبه می‌شود (هر کدام که کوچکتر باشد). مثلاً گیت‌های TTL استاندارد دارای مقادیر زیر برای جریان‌ها هستند:

$$I_{OH} = 400\ \mu A$$
$$I_{IH} = 40\ \mu A$$
$$I_{OL} = 16\ mA$$
$$I_{IL} = 1.6\ mA$$

در این حالت هر دو نسبت، یک عدد را بدست می‌دهند.

$$\frac{400\ \mu A}{40\ \mu A} = \frac{16\ mA}{1.6\ mA} = 10$$

بنابراین گنجایش خروجی TTL استاندارد 10 است. این بدان معنی است که خروجی یک گیت TTL را نمی‌توان به بیش از 10 ورودی از گیت‌های همان خانواده منطقی وصل کرد. در غیر این صورت گیت قادر نخواهد بود جریان لازم برای ورودی‌های متصل به خود تولید و یا فروکش کند.

توان مصرفی

هر مدار الکترونیک برای کار به توان معینی نیاز دارد. توان مصرفی پارامتری است که برحسب میلی وات (mW) بیان شده و مقدار توان لازم هر گیت را نشان می‌دهد. عددی که این پارامتر را نشان می‌دهد

به معنی توان انتقال یافته از گیت دیگر نیست، بلکه به معنی توان منتقله از منبع تغذیه متصل به همان گیت است. یک مدار مجتمع با چهار گیت چهار برابر توانی را لازم دارد که یک گیت مصرف می‌کند.

توان مصرفی در هر گیت از ولتاژ تغذیه V_{CC} و جریان کشیده شده به وسیله مدار I_{CC} محاسبه می‌گردد. در واقع توان حاصلضرب $V_{CC} \times I_{CC}$ است. جریان کشیده شده از منبع تغذیه به حالت منطقی گیت بستگی دارد. این جریان وقتی که خروجی گیت در سطح بالای ولتاژ است I_{CCH} نامیده می‌شود. در سطح پایین ولتاژ، جریان، I_{CCL} است. جریان متوسط برابر است با

$$I_{CC}(\text{avg}) = \frac{I_{CCH} + I_{CCL}}{2}$$

و برای محاسبه توان مصرفی به کار می‌رود:

$$P_D(\text{avg}) = I_{CC}(\text{avg}) \times V_{CC}$$

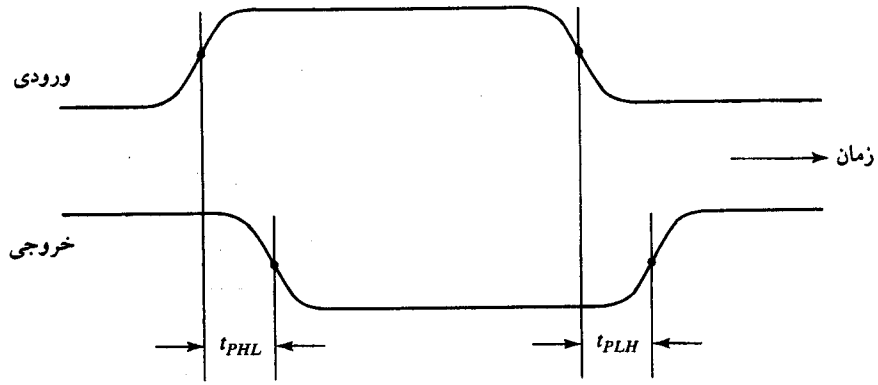
مثلاً یک گیت TTL NAND از یک ولتاژ تغذیه $V_{CC} = 5\text{V}$ استفاده می‌کند و جریان فروکش آن $I_{CCH} = 1\text{mA}$ و $I_{CCL} = 3\text{mA}$ است. متوسط جریان $\frac{3+1}{2} = 2\text{mA}$ خواهد بود. توان متوسط مصرفی $5 \times 2 = 10\text{mW}$ خواهد شد. مدار مجتمعی که دارای چهار گیت است جمعاً $10 \times 4 = 40\text{mW}$ توان لازم دارد. در یک سیستم دیجیتال نوعی، تعداد قابل توجهی مدار مجتمع وجود دارد و توان مصرفی به وسیله هر آی‌سی باید مورد ملاحظه قرار گیرد. توان کل مصرفی در یک سیستم جمع کل توان‌های مصرفی در همه مدارهای مجتمع است.

تأخیر انتشار

تأخیر انتشار یک گیت، متوسط تأخیر انتشار زمانی است که طی آن تغییر حالت سیگنال در ورودی، به خروجی منتقل می‌گردد. انتشار سیگنال‌ها از ورودی به خروجی یک گیت مدت زمان معینی طول می‌کشد. این فاصله زمانی همان تأخیر انتشار است. این زمان برحسب نانو ثانیه محاسبه می‌شود. هر نانو ثانیه 10^{-9} ثانیه است.

سیگنال‌ها حین انتقال از ورودی‌ها به خروجی‌های یک مدار دیجیتال، از یک سری گیت‌ها می‌گذرند. مجموع تأخیرهای انتشار از گیت‌ها برابر با کل تأخیر در مدار است. هنگامی که سرعت عمل اهمیت دارد، هر گیت باید تأخیر انتشار کوتاهی داشته و مدار دیجیتال هم باید حداقل تعداد گیت‌ها را بین ورودی و خروجی دارا باشد.

زمان تأخیر انتشار متوسط هر گیت با توجه به شکل موج‌های ورودی و خروجی، مطابق شکل ۴-۱۰ محاسبه می‌گردد. زمان تأخیر سیگنال بین ورودی و خروجی به هنگام تغییر خروجی از 1 به 0 را t_{PHL} نامند. بطور مشابه وقتی خروجی از 0 به 1 می‌رود، تأخیر را t_{PLH} خوانند. رسم بر این است که تا فاصله زمانی بین 50 درصد از تغییر حالت در ورودی و خروجی اندازه‌گیری شود. به طور کلی این دو تأخیر با یکدیگر برابر نیستند و هر دو با تغییر شرایط بار تغییر می‌کنند. زمان متوسط تأخیر انتشار به عنوان متوسط دو تأخیر در نظر گرفته می‌شود.



شکل ۴-۱۰ . اندازه گیری تأخیر زمانی

به عنوان مثال، تأخیرها در یک گیت TTL استاندارد عبارتند از $t_{PHL} = 7ns$ ، $t_{PLH} = 11ns$. این کمیت‌ها در کتابچه‌های اطلاعات TTL وارده شده و با بار 400Ω و ظرفیت خازنی 15PF در نظر گرفته شده است. زمان متوسط تأخیر انتشار گیت TTL برابر $9ns = 2/(7 + 11)$ می‌باشد.

تحت شرایطی خاص، دانستن حداکثر زمان تأخیر یک گیت از مقدار متوسط آن اهمیت بیشتری دارد. در کتابچه‌های اطلاعات، تأخیرهای انتشار برای یک گیت NAND برابر با $t_{PHL}=15ns$ و $t_{PLH} = 22ns$ اعلان شده است. وقتی که سرعت از اهمیت بالاتری برخوردار است باید تأخیر ماکزیمم را در نظر گرفت تا از عملکرد صحیح اطمینان حاصل شود.

سیگنال‌های ورودی در بسیاری از مدارهای دیجیتال، به طور همزمان به بیش از یک گیت اعمال می‌گردند. همه گیت‌های متصل به ورودی‌های بیرونی، سطح منطقی اول را تشکیل می‌دهند. گیتی که حداقل یک ورودی از خروجی گیت‌های سطح اول دریافت کند، در سطح دوم قرار دارد و به همین ترتیب گیت‌های سطوح بالاتر می‌باشند. تأخیر انتشار کل مدار برابرست با تأخیر انتشار یک گیت ضرب در تعداد سطوح منطقی مدار. بنابراین کاهش در تعداد سطوح منطقی موجب کاهش زمان تأخیر سیگنال و بنابراین سریع‌تر شدن آن خواهد شد. در مواردی که سرعت عمل اهمیت دارد کاهش تأخیر انتشار در مدارها مهمتر از کاهش تعداد کل گیت‌هاست.

حد پارازیت

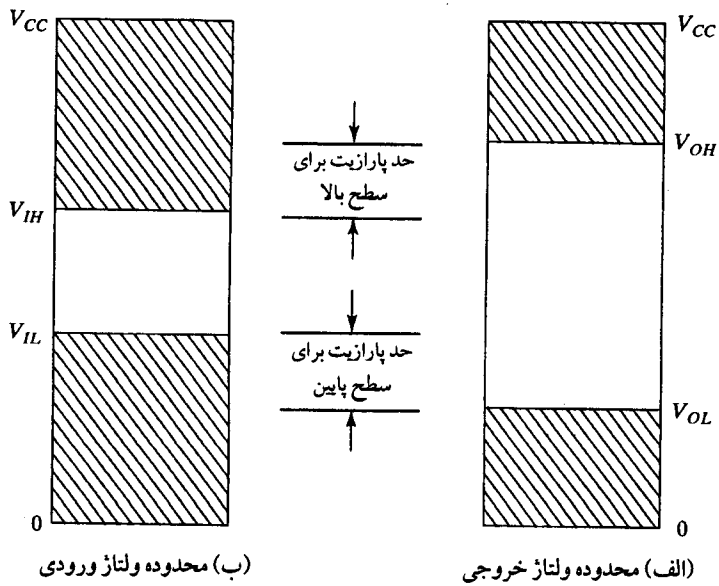
در صنعت و دیگر فعالیت‌های مشابه، سیگنال‌های الکتریکی قادرند ولتاژهای القایی نامطلوبی را در سیستم‌های رابط بین مدارهای منطقی ایجاد کنند. این سیگنال‌های ناخواسته را پارازیت می‌نامند. دو نوع سیگنال قابل بررسی وجود دارد. یکی پارازیت DC است که به علت **انحراف** سطوح ولتاژ یک سیگنال ایجاد می‌شود و دیگری پارازیت AC است که ممکن است توسط تغییر سطح سیگنال‌های دیگر تولید گردد. بنابراین پارازیت کلمه‌ای است که برای درک وجود یک سیگنال ناخواسته بر روی یک

سیگنال معمول در مدار به کار می‌رود. حد پارازیت حداکثر ولتاژ پارازیتی است که به سیگنال ورودی اضافه شده ولی سبب تغییر ناخواسته‌ای در خروجی مدار نمی‌گردد. توان کار یک مدار در یک محیط پارازیت‌دار از اهمیت ویژه‌ای برخوردار است. حد پارازیت برحسب ولت بیان شده و بیانگر حداکثر ولتاژی است که به وسیله گیت قابل تحمل است.

حد پارازیت با آگاهی از سیگنال ولتاژ موجود در خروجی و سیگنال ولتاژ در ورودی گیت قابل محاسبه است. شکل ۵-۱۰ سیگنال‌های مورد استفاده در حد پارازیت را نشان می‌دهد. بخش (الف) محدوده ولتاژهای خروجی که در یک گیت نمونه رخ می‌دهد، را نمایش می‌دهد. هر ولتاژ خروجی واقع در بین V_{CC} و V_{OH} به عنوان حالت سطح بالا و هر ولتاژ بین 0 و V_{OL} در خروجی گیت به عنوان سطح پایین تلقی می‌گردد. ولتاژهای بین V_{OL} و V_{OH} نامعین هستند و تحت شرایط عادی به جز به هنگام گذر بین دو سطح ظاهر نمی‌گردند. محدوده دو سطح ولتاژ قابل تشخیص به وسیله ورودی گیت، در شکل ۵-۱۰ (ب) دیده می‌شوند. برای جبران هر سیگنال پارازیت، مدار باید طوری طراحی شود که V_{IL} بزرگتر از V_{OL} و V_{IH} کمتر از V_{OH} باشند. حد پارازیت تفاضل کوچکتر دو مقدار $V_{OH}-V_{IH}$ یا $V_{IL}-V_{OL}$ است.

همانگونه که در شکل ۵-۱۰ تشریح شد، V_{OL} حداکثر ولتاژی است که خروجی می‌تواند در سطح پایین دارا باشد. مدار قادر است تا هر سیگنال پارازیت کمتر از حد پارازیت $(V_{IL}-V_{OL})$ را تحمل کند زیرا ورودی سیگنال را در سطح پایین تشخیص خواهد داد. هر سیگنال بزرگتر از V_{OL} به علاوه حد پارازیت، ورودی را به ناحیه نامعین می‌فرستد و ممکن است سبب بروز خطا در خروجی گردد. به طریق مشابه، یک پارازیت منفی بزرگتر از $V_{OH}-V_{IH}$، ولتاژ ورودی را به ناحیه نامعین خواهد فرستاد.

پارامترهای حد پارازیت در گیت NAND، TTL استاندارد V_{OH}=2.4V، V_{OL}=0.4V، V_{IH} = 2V



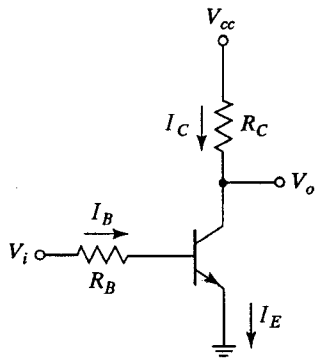
شکل ۵-۱۰ . سیگنال‌ها برای ارزیابی حد پارازیت

و V_{IL}=0.8V می‌باشند. حد پارازیت سطح بالا 0.4V = 2 − 2.4 و حد پایین آن 0.4V = 0.4 − 0.8 است. در این حالت هر دو مقدار یکی هستند.

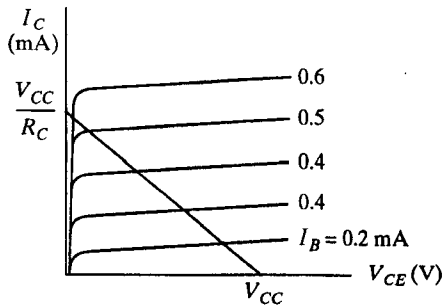
۳-۱۰ مشخصه‌های ترانزیستور دو قطبی

این بخش به مروری بر ترانزیستور دو قطبی به کار رفته در مدارهای دیجیتال اختصاص یافته است. این اطلاعات برای تحلیل مدار پایه مورد استفاده در چهار خانواده منطقی دو قطبی به کار می‌رود. ترانزیستورهای دو قطبی ممکن است از نوع *npn* یا *pnp* باشند. به علاوه، آنها از مواد نیمه هادی سیلیکان یا ژرمانیوم ساخته شده‌اند. معمولاً ترانزیستورهای مدار مجتمع (IC) با سیلیکان ساخته شده و اغلب از نوع *npn* هستند.

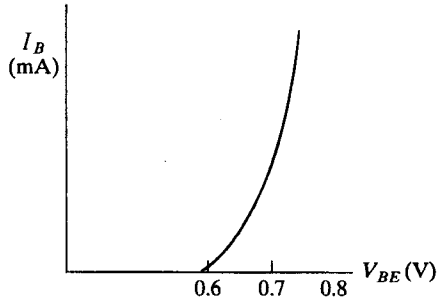
اطلاعات اولیه مورد نیاز جهت تحلیل مدارهای دیجیتال را می‌توان با بررسی منحنی‌های ترانزیستور امیتر مشترک نوع سیلیکان با پیوند *npn* شکل ۶-۱۰، بدست آورد. مدار (الف) یک وارونگر ساده با دو مقاومت و یک ترانزیستور است. جریان I_C از مقاومت R_C و کلکتور ترانزیستور عبور می‌کند. جریان I_B از مقاومت R_B و بیس ترانزیستور می‌گذرد. امیتر به زمین وصل است و جریان آن $I_E = I_C + I_B$ می‌باشد. ولتاژ تغذیه بین V_{CC} و زمین قرار دارد. ورودی هم بین V_i و زمین است، و بالاخره خروجی بین V_0 و زمین می‌باشد.



(الف) مدار وارونگر



(پ) مشخصه کلکتور ترانزیستور



(ب) مشخصه بیس ترانزیستور

شکل ۶-۱۰ . ترانزیستور *npn* سیلیکان

ما جریان مثبت مدار را مطابق شکل تصور می‌کنیم. این جهت‌ها معمولاً به یک ترانزیستور *npn* تعلق دارد. جریان امیتر I_E وقتی مثبت فرض می‌شود که از ترانزیستور خارج شود. سمبل V_{CE} به معنی افت ولتاژ دو سر کلکتور و امیتر بوده و همیشه مثبت تلقی می‌گردد. به همین ترتیب افت ولتاژ دو سر پیوند بیس- امیتر است. این پیوند با V_{BE} مثبت، به طور مستقیم و با V_{BE} منفی، به طور معکوس تغذیه می‌شود.

مشخصه گرافیکی بیس- امیتر در شکل ۶-۱۰ (ب) نشان داده شده است که عبارت است از نمودار V_{BE} در برابر I_B. اگر ولتاژ بیس- امیتر کمتر از 0.6V باشد، گوییم ترانزیستور در حالت قطع است. هنگام تغذیه مستقیم با ولتاژی بزرگتر از 0.6V، ترانزیستور هدایت کرده و I_B به سرعت افزایش می‌یابد ضمن آن که V_{BE} تغییر کمی دارد. ولتاژ V_{BE} دو سر یک ترانزیستور در حال هدایت به ندرت از 0.8V تجاوز می‌کند.

مشخصه‌های گرافیکی کلکتور- امیتر، همراه با خط بار، در شکل ۶-۱۰ (پ) آمده است. وقتی V_{BE} کمتر از 0.6V است، ترانزیستور با $I_B = 0$ در حالت قطع است و در این هنگام جریان ناچیزی از کلکتور عبور می‌کند. بنابراین مدار کلکتور- امیتر همچون یک مدار باز عمل می‌نماید. در ناحیه فعال ولتاژ کلکتور، V_{CE}، می‌تواند در هر جایی بین 0.8V تا V_{CC} باشد. در این ناحیه جریان کلکتور I_C را می‌توان با رابطه تقریبی $I_B h_{FE}$ محاسبه کرد، که در آن h_{FE} پارامتری از ترانزیستور به نام بهره جریان dc است. جریان حداکثر کلکتور به I_B وابسته نیست، بلکه به مدار بیرونی متصل به کلکتور بستگی دارد. دلیل این است که V_{CE} همیشه مثبت و کمترین مقدار آن 0V است. مثلاً در وارونگری که ملاحظه شد، حداکثر جریان با $V_{CE} = 0$ برابر با $I_C = V_{CC}/R_C$ می‌باشد.

قبلاً بیان شد که در ناحیه فعال $I_C = h_{FE} I_B$ است. پارامتر h_{FE} در محدوده نقطه کار ترانزیستور تغییرات وسیعی دارد، ولی باز هم برای اهداف تحلیلی استفاده از مقدار متوسط آن مفید است. در یک بازه عمل نمونه، h_{FE} حدود 50 است، ولی تحت شرایط می‌تواند تا $h_{FE} I_B$ از I_C بزرگتر گردد. اگر چنین حالتی فرا برسد در ناحیه اشباع خواهیم بود. بنابراین، شرط اشباع از رابطه زیر بدست می‌آید.

$$I_B \geq \frac{I_{CS}}{h_{FE}}$$

که در آن I_{CS} حداکثر جریان کلکتور در هنگام اشباع است. V_{CE} در ناحیه اشباع دقیقاً صفر نیست بلکه حدود 0.2 می‌باشد.

اطلاعات پایه برای تحلیل مدارهای دیجیتال ترانزیستوری دو قطبی در جدول ۱-۱۰ لیست شده است. در ناحیه قطع، V_{BE} کمتر از 0.6V است، V_{CE} به صورت یک مدار باز عمل می‌کند، و هر دو جریان قابل چشم‌پوشی‌اند. در ناحیه فعال، V_{BE} حدود 0.7V است، V_{CE} ممکن است در محدوده

جدول ۱-۱۰. پارامترهای یک نمونه ترانزیستور *npn* سیلیکان

ناحیه	V_{BE} (V)	V_{CE} (V)	رابطه جریان
قطع	< 0.6	مدار باز	$I_B = I_C = 0$
فعال	0.6–0.7	> 0.8	$I_C = h_{FE} I_B$
اشباع	0.7–0.8	0.2	$I_B \geq I_{CS}/h_{FE}$

وسیعی تغییر کند، و I_C به صورت تابعی از I_B قابل محاسبه است. در ناحیه اشباع، V_{BE} به سختی قابل تغییر است، ولی V_{CE} به 0.2V افت می‌کند. جریان بیس باید به حد کافی بزرگ باشد یا در نامساوی فوق صدق کند. برای سادگی تحلیل $V_{BE} = 0.7\text{V}$ را برای حالت اشباع یا هدایت ترانزیستور فرض خواهیم کرد.

تحلیل مدارهای دیجیتال ممکن است با استفاده از روشهای از پیش تعیین شده زیر باشد: برای هر ترانزیستور در مدار ببینید آیا $V_{BE} = 0.6\text{V}$ است یا خیر. اگر این طور باشد، ترانزیستور قطع است و مدار کلکتور به امیتر به صورت یک مدار باز در نظر گرفته می‌شود. اگر V_{BE} بزرگتر از 0.6V باشد، ترانزیستور ممکن است در ناحیه اشباع یا فعال باشد. جریان بیس را با فرض $V_{BE} = 0.7\text{V}$، محاسبه نمایید. سپس حداکثر میزان ممکن جریان کلکتور I_{CS} را با فرض $V_{CE} = 0.2\text{V}$ محاسبه کنید. این محاسبات باید برحسب ولتاژ و مقادیر مقاومت‌ها باشند. سپس اگر جریان بیس به اندازه کافی بزرگ بود تا رابطه $I_B \geq I_{CS}/h_{FE}$ برقرار باشد، آنگاه ترانزیستور در ناحیه اشباع با $V_{CE} = 0.2\text{V}$ خواهد بود. به هر حال، اگر جریان بیس کمتر و رابطه فوق برقرار نباشد، ترانزیستور در ناحیه فعال است و جریان I_C را از رابطه $I_C = h_{FE} I_B$ بدست خواهیم آورد.

برای اثبات این مطلب، مدار وارونگر شکل ۶-۱۰ (الف) را با پارامترهای زیر در نظر بگیرید.

$R_C = 1\,k\Omega$	$V_{CC} = 5\,\text{V}$	(ولتاژ تغذیه)
$R_B = 22\,\text{k}\Omega$	$H = 5\,\text{V}$	(ولتاژ سطح بالا)
$h_{FE} = 50$	$L = 0.2\,\text{V}$	(ولتاژ سطح پایین)

با ولتاژ ورودی $V_i = L = 0.2\text{V}$ داریم $V_{BE} < 0.6\text{V}$ و ترانزیستور خاموش است. مدار کلکتور-امیتر همچون یک مدار باز عمل می‌نماید، به نحوی که $V_0 = 5\text{V} = H$.

با یک ولتاژ ورودی $V_i = H = 5\text{V}$، نتیجه می‌گیریم که $V_{BE} > 0.6\text{V}$. با فرض $V_{BE} = 0.7$، جریان بیس را محاسبه می‌کنیم:

$$I_B = \frac{V_i - V_{BE}}{R_B} = \frac{5 - 0.7}{22\,\text{k}\Omega} = 0.195\,\text{mA}$$

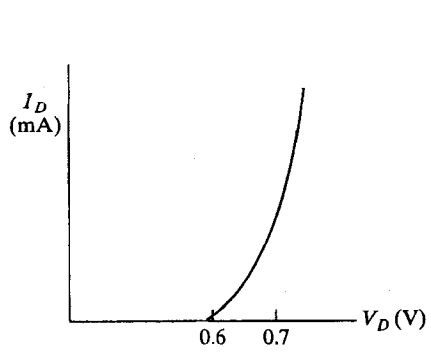
حداکثر جریان کلکتور با فرض $V_{CE} = 0.2$ برابر است با

$$I_{CS} = \frac{V_{CC} - V_{CE}}{R_C} = \frac{5 - 0.2}{1\,\text{k}\Omega} = 4.8\,\text{mA}$$

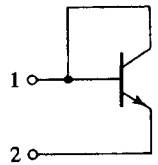
سپس وجود حالت اشباع را از رابطه زیر چک می‌نماییم

$$0.195 = I_B \geq \frac{I_{CS}}{h_{FE}} = \frac{4.8}{50} = 0.096\,\text{mA}$$

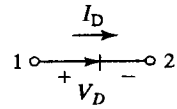
و در می‌یابیم که نامساوی برقرار است، زیرا $0.096 < 0.195$ بوده و نتیجه می‌گیریم که ترانزیستور در حالت اشباع است و ولتاژ خروجی $V_0 = V_{CE} = 0.2\text{V} = L$ می‌باشد. بنابراین مدار به صورت یک وارونگر عمل می‌کند.



(پ) مشخصه دیود



(الف) ترانزیستوری که جهت استفاده بعنوان یک دیود بکار رفته است



(ب) سمبل گرافیکی دیود

شکل ۷-۱۰ . سمبل و مشخصه دیود سیلیکان

روال توصیف شده فوق، به طور گسترده‌ای در تحلیل مدارهای بخش‌های بعدی استفاده خواهد شد. این کار با تحلیل کیفی، یعنی بدون نوشتن معادلات عددی صورت خواهد گرفت. تحلیل کمی و محاسبات خاص به عنوان تمرین در بخش مسائل انتهای فصل ارائه شده است.

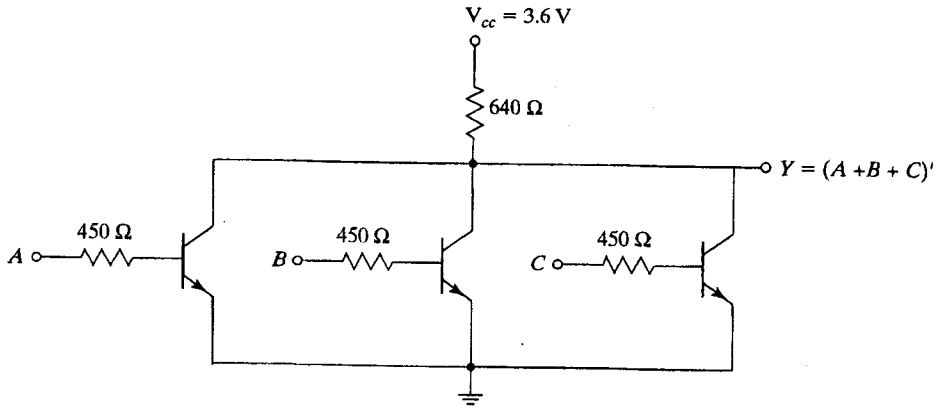
مواردی وجود دارد که نه تنها ترانزیستور بلکه دیودها نیز در مدارهای دیجیتال به کار می‌روند. یک دیود مدار مجتمع (IC) معمولاً از یک ترانزیستور که کلکتور آن به بیس وصل شده ساخته می‌شود، شکل ۷-۱۰ (الف). سمبل گرافیکی به کار رفته برای یک دیود در شکل ۷-۱۰ (ب) ملاحظه می‌شود. دیود اساساً همانند پیوند بیس- امیتر یک ترانزیستور کار می‌کند. مشخصه‌های گرافیکی آن که در شکل ۷-۱۰ (پ) دیده می‌شود، مشابه مشخصه‌های بیس امیتر ترانزیستور است. بنابراین می‌توان نتیجه گرفت که وقتی ولتاژ مستقیم آن، V_D، کمتر از 0.6V باشد دیود قطع بوده و هدایت نخواهد کرد. هنگام هدایت دیود، جریان I_D در جهتی که در شکل ۷-۱۰ (ب) نشان داده شده، جاری شده و V_D در حدود 0.7V ثابت خواهد ماند. برای محدود کردن جریان در دیود باید مقاومتی را با آن سری کرد، زیرا ولتاژ در حدود چیزی کمتر از یک ولت ثابت می‌ماند.

۴-۱۰ مدارهای RTL و DTL

گیت پایه RTL

مدار پایه خانواده منطقی دیجیتال RTL گیت NOR است که در شکل ۸-۱۰ نشان داده شده است. هر ورودی این خانواده با یک مقاومت و یک ترانزیستور مرتبط است. کلکتور همه ترانزیستورها در خروجی به هم وصل شده‌اند. سطوح منطقی برای مدار برابر 0.2V برای سطح پایین و از 1 الی 3.6V برای سطح بالاست.

تحلیل گیت RTL بسیار ساده است و بحث مطرح شده بخش قبل را دنبال می‌کند. اگر هر یک از ورودی‌های گیت RTL بالا باشد، ترانزیستور مربوط به آن ورودی به حالت اشباع کشانده می‌شود. این



شکل ۸-۱۰ . گیت NOR پایه RTL

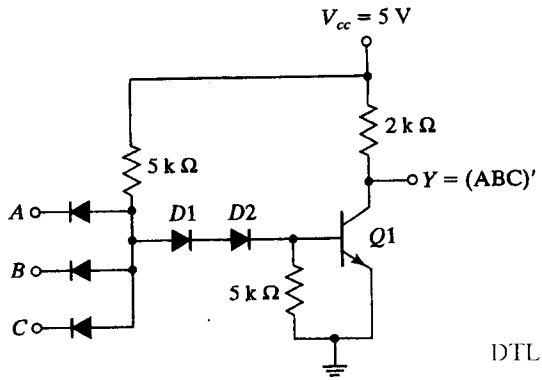
وضعیت موجب می‌گردد تا خروجی بدون توجه به حالت دیگر ترانزیستورها به سطح پایین منطقی برود. اگر همه ورودی‌ها در 0.2V باشند، همه آنها خاموش خواهند بود، زیرا در همه آنها $V_{BE}<0.6V$ است. این وضعیت موجب می‌شود تا خروجی مدار به سطح منطقی بالا و به سمت ولتاژ تغذیه V_{CC} برود. این حالات، وضعیت‌های بیان شده گیت NOR در شکل ۲-۱۰ را تأیید می‌کند. توجه داشته باشید که حد پارازیت برای سیگنال سطح پایین $0.6 - 0.2 = 0.4V$ است.

گنجایش خروجی گیت RTL وقتی در ولتاژ خروجی بالاست، محدود نمی‌باشد. با بار شدن خروجی توسط ورودی گیت‌های دیگر، جریان بیشتری مصرف می‌شود. این جریان باید از مقاومت 640Ω بگذرد. یک محاسبه ساده (رجوع به مسئله ۲-۱۰) نشان می‌دهد که اگر h_{FE} به 20 تنزل کند، ولتاژ خروجی برای گنجایش 5 ، به 1V می‌رسد. هر ولتاژ کمتر از 1V در خروجی قادر نخواهد بود تا ترانزیستور بعدی را آن طور که باید به اشباع ببرد. توان تلف شده در گیت RTL حدود 12mW و متوسط تأخیر انتشار حدود 25ns می‌باشد.

گیت پایه DTL

مدار پایه خانواده منطقی دیجیتال DTL گیت NAND شکل ۹-۱۰ است. هر ورودی متصل به یک دیود است. دیودها و مقاومت 5KΩ تشکیل یک گیت AND را می‌دهند. ترانزیستور نقش تقویت کننده جریان را به عهده دارد. ضمن این که سیگنال را معکوس هم می‌کند. سطوح ولتاژ عبارتند از 0.2V برای سطح منطقی پایین و بین 4 الی 5 ولت برای سطح منطقی بالا.

تحلیل گیت DTL باید مطابق شرایط گیت NAND موجود در شکل (۱-۱۰) باشد. اگر هر یک از ورودی‌های گیت در سطح پایین 0.2V باشد، دیود ورودی مربوطه، جریان را از V_{CC} و از طریق مقاومت 5KΩ به گره ورودی هدایت می‌نماید. ولتاژ در نقطه P برابر با ولتاژ ورودی 0.2V به علاوه افت ولتاژ 0.7V در دو سر دیود است که جمعاً 0.9V می‌شود. برای این که ترانزیستور شروع به هدایت کند ولتاژ در نقطه



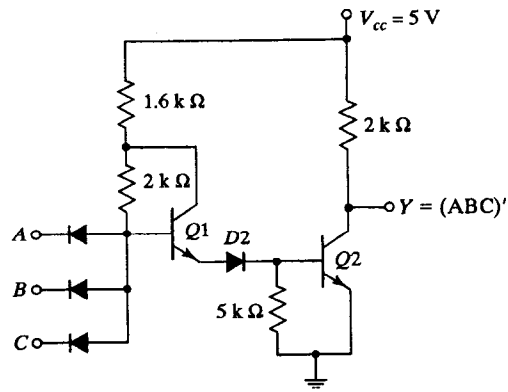
شکل ۹-۱۰. گیت NAND پایه DTL

P باید بیشتر از یک افت V_{BE} در $Q1$ بعلاوه دو افت دیود در دیودهای $D1$ و $D2$ یا $3 \times 0.6 = 1.8V$ باشد. از آنجایی که ولتاژ در نقطه P به وسیله دیود ورودی در 0.9V ثابت می‌ماند، ترانزیستور خاموش شده و ولتاژ خروجی در سطح بالای 5 ولت خواهد بود.

اگر همه ورودی‌های گیت در سطح بالا باشند، ترانزیستور به ناحیه اشباع کشانده می‌شود. اکنون ولتاژ در P برابر V_{BE} بعلاوه دو افت ولتاژ دو سر دیودهای $D1$ و $D2$ یا $3 \times 0.7 = 2.1V$ است. چون همه ورودی‌ها در ولتاژ سطح بالای 5V هستند و $V_P = 2.1V$ است، دیودهای ورودی تغذیه معکوس شده و خاموش می‌مانند. جریان بیس برابر با تفاضل جریان‌های عبوری از دو مقاومت 5KΩ می‌باشد و برای اشباع شدن ترانزیستور کافی است (رجوع به مسئله ۳-۱۰). با اشباع شدن ترانزیستور، خروجی به $V_{CE} = 0.2V$ افت می‌کند که سطح پایین منطقی برای گیت است.

توان تلف شده در یک گیت DTL حدود 12mW و متوسط تأخیر انتشار 30ns است. حد پارازیت حدود 1V و گنجایش خروجی تا 8 امکان‌پذیر است. گنجایش خروجی گیت DTL با حداکثر جریان اشباع در کلکتور ترانزیستور محدود شده است (رجوع به مسئله ۴-۱۰).

اگر یکی از دیودهای موجود در بیس را با یک ترانزیستور عوض کنیم، گنجایش خروجی گیت DTL افزایش می‌یابد، شکل ۱۰-۱۰. وقتی که خروجی $Q2$ اشباع شود، ترانزیستور $Q1$ در ناحیه فعال



شکل ۱۰-۱۰. گیت تصحیح شده DTL

نگهداری می‌شود. در نتیجه، مدار اصلاح شده می‌تواند جریان بیشتری را به پایه ترانزیستور خروجی اعمال کند. اکنون ترانزیستور خروجی می‌تواند مقدار بیشتری جریان کلکتور را قبل از رفتن به اشباع دارا باشد. بخشی از جریان کلکتور از دیودهای هادی موجود در گیت‌های بار شده به هنگام اشباع $Q2$ می‌آید. بنابراین افزایش مجاز جریان اشباع شده کلکتور اجازه می‌دهد تا بارهای بیشتری به خروجی وصل شوند که به معنی افزایش گنجایش خروجی است.

۵-۱۰ منطق ترانزیستور- ترانزیستور TTL

گیت TTL پایه اولیه، تنها فرم تکمیل شده گیت DTL بود. با پیشرفت تکنولوژی TTL، اصلاحات اضافی تا حدی افزایش یافتند که این خانوادگی منطقی به طور گسترده در طراحی سیستم‌های دیجیتال به کار گرفته شد. در حال حاضر چندین نوع یا سری از خانواده TTL وجود دارد. نام و مشخصات هشت نوع TTL در جدول ۲-۱۰ ملاحظه می‌شود. آی‌سی‌های TTL تجاری با شماره‌ای که اول آن با 74 شروع می‌شود مشخص می‌گردند و بدنبال آن نام سری آمده است. مثال‌هایی از آن عبارت است از 7404، 74S86 و 74ALS161. گنجایش خروجی، توان تلف شده و تأخیر در انتشار در بخش ۲-۱۰ تعریف شدند. حاصلضرب سرعت- توان پارامتر مهم دیگری در مقایسه انواع سری‌های TTL است. این حاصلضرب تأخیر انتشار و مصرف بوده و برحسب پیکوژول اندازه‌گیری می‌شود. مقادیر کمتر برای این پارامتر مطلوب‌تر است زیرا به این معنی است که می‌توان بدون صرف انرژی به تأخیر انتشار مورد نظری دسترسی پیدا کرد.

اولین نوع از خانواده TTL، نوع TTL استاندارد بود. سپس این گیت پایه با مقادیر متفاوتی از مقاومت طراحی گردید تا گیت‌هایی با توان مصرفی کم یا با سرعت بالاتر تولید شوند. تأخیر انتشار یک

جدول ۲-۱۰. انواع TTL و مشخصات آنها

نام سری TTL	پیشوند	گنجایش خروجی	توان مصرفی (mW)	تاخیر انتشار (ns)	حاصلضرب توان سرعت (pJ)
استاندارد	74	10	10	9	90
توان پایین	74L	20	1	33	33
سرعت بالا	74H	10	22	6	132
شوتکی	74S	10	19	3	57
شوتکی توان پایین	74LS	20	2	9.5	19
شوتکی پیشرفته	74AS	40	10	1.5	15
شوتکی پیشرفته توان پایین	74ALS	20	1	4	4
سریع	74F	20	4	3	12

ترانزیستور در حالت اشباع عمدتاً به دو عامل وابسته است: زمان ذخیره‌سازی و ثابت‌های زمانی RC. کاهش زمان ذخیره‌سازی موجب کم شدن تأخیر در انتشار می‌گردد. البته ماحصل این کار، اتلاف توان بیشتری است، زیرا مقاومت کمتر جریان بیشتری را از مبنع می‌کشد. سرعت گیت متناسب با عکس تأخیر انتشار است.

درگیت TTL با توان پایین مقادیر مقاومت، بیشتر از مقادیر مشابه درگیت نوع استاندارد می‌باشند تا بدین ترتیب توان مصرفی کاهش یابد ولی تأخیر انتشار افزایش یافته است. درگیت TTL سریع، مقادیر مقاومت کاهش یافته تا زمان تأخیر کم شود، ولی توان مصرفی افزایش یافته است. گیت TTL شوتکی اصلاح بعدی در تکنولوژی بود. تأثیر ترانزیستور شوتکی، کاهش زمان تأخیر ذخیره‌سازی با ممانعت از اشباع شدن ترانزیستور بود. این سری، سرعت عمل را بدون افزودن توان مصرفی بالا می‌برد. نوع TTL شوتکی با توان پایین، مقداری از سرعت را فدای کاهش توان تلف شده نموده است. از نظر تأخیر، این نوع مشابه با نوع استاندارد است ولی توان مصرفی آن $\frac{1}{5}$ می‌باشد. اصلاحات منجر به ساخت سری شوتکی پیشرفته شده است. این نمونه دارای تأخیر انتشار کمتر و نیز توان مصرفی پایین‌تر نسبت به نوع استاندارد است. نوع شوتکی پیشرفته دارای حاصلضرب سرعت- توان کوچکتر بوده و کاراترین نوع است. خانواده TTL سریع بهترین انتخاب برای طراحی‌های سریع است.

همه سری‌های TTL به صورت SSI و نیز فرم‌های پیچیده‌تر MSI و LSI در دسترسند. تفاوت میان سری‌های TTL در منطق دیجیتال آنها نیست، بلکه ساختار داخلی گیت پایه NAND آنها با هم اختلاف دارد. در هر حال گیت‌های TTL در همه سری‌ها دارای سه نوع آرایش خروجی‌اند:

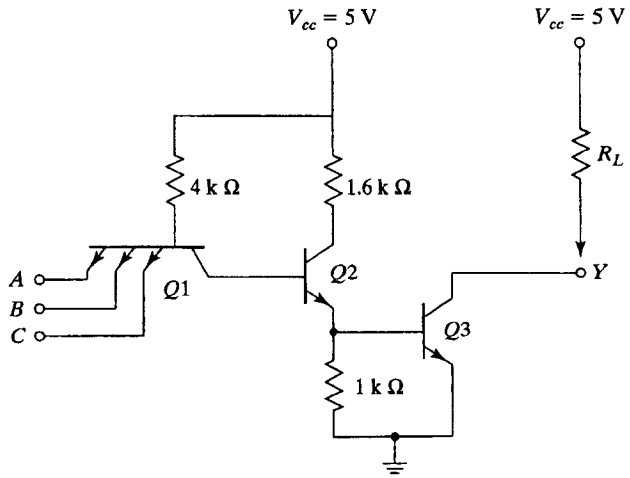
۱- خروجی کلکتور باز
۲- خروجی تاتم پل (totem - pole)
۳- خروجی سه حالته

این نوع خروجی همراه با توصیف گیت TTL پایه آنها مورد بررسی قرار خواهد گرفت.

گیت‌با خروجی کلکتور باز

گیت پایه TTL در شکل ۱۱-۱۰، نوع اصلاح شده گیت DTL است. به امیتر چندین ورودی متصل شده است. این امیترها اغلب مشابه ورودی دیودها در گیت DTL عمل می‌کنند زیرا با بیس مشترک خود، یک پیوند pn را تشکیل می‌دهند. پیوند بیس- کلکتور $Q1$ که پیوند دیودی pn دیگری است نقش $D1$ درگیت DTL را بازی می‌کند. (شکل ۵-۱۰ ملاحظه شود). تراتزیستور $Q2$ جایگزین دومین دیود، $D2$، در گیت DTL شده است. خروجی گیت TTL از کلکتور باز $Q3$ گرفته می‌شود. یک مقاومت متصل به V_{CC} باید از بیرون به بسته IC وصل شود تا هنگام خاموشی $Q3$ خروجی را به سطح بالای ولتاژ، بالا بکشد، در غیر این صورت خروجی به صورت مدار باز عمل خواهد کرد. دلیل قرار ندادن مقاومت در درون مدار بعد بحث خواهد شد.

دو سطح ولتاژ گیت TTL عبارتند از 0.2V برای سطح منطقی پایین و 2.4 تا 5V برای سطح بالا،



شکل ۱۱-۱۰. گیت TTL کلکتور باز

مدار پایه یک گیت NAND است. اگر هر یک از ورودی‌ها پایین باشد، پیوند بیس- امیتر مربوطه در $Q1$ تغذیه مستقیم می‌گردد. ولتاژ در بیس $Q1$ برابر با ولتاژ ورودی 0.2V بعلاوه افت ولتاژ 0.7 و یا مجموعاً 0.9V می‌باشد. برای این که $Q3$ شروع به هدایت کند، مسیر $Q1$ تا $Q3$ باید بر افت ولتاژ یک دیود در پیوند pn بیس- کلکتور $Q1$ و دو افت V_{BE} در $Q2$ و $Q3$ یا $3 \times 0.6 = 1.8V$ غلبه نماید. چون بیس $Q1$ به وسیله سیگنال ورودی در 0.9V نگهداری می‌شود، خروجی ترانزیستور نمی‌تواند هدایت کند و قطع می‌شود. خروجی به شرطی به سطح بالا می‌رود که یک مقاومت خارجی بین خروجی و V_{CC} قرار گیرد (اگر مقاومت به کار نرود به صورت مدار باز در خواهد آمد).

اگر همه ورودی‌ها در سطح بالا باشند، هر دو ترانزیستور $Q2$ و $Q3$ هدایت کرده و به اشباع می‌روند. ولتاژ بیس $Q1$ با ولتاژ دو سر پیوند pn بیس- کلکتور بعلاوه دو افت ولتاژ V_{BE} در $Q2$ و $Q3$ یا حدود $0.7 \times 3 = 2.1V$، برابر است. چون همه ورودی‌ها در سطح منطقی و بزرگتر از 2.4V می‌باشند، پیوندهای بیس- امیتر $Q1$ همگی معکوس تغذیه شده‌اند. هنگام اشباع شدن ترانزیستور $Q3$ (به شرطی که مسیر جریان وجود داشته باشد) ولتاژ خروجی به 0.2V می‌رود. این حالت عملکرد NAND را برای گیت تصدیق می‌کند.

در این تحلیل، گفتیم که پیوند کلکتور- بیس ترانزیستور $Q1$ مانند یک پیوند pn در دیود عمل می‌نماید. این مطلب در شرایط سکون صحت دارد. با این وجود، در حین گذر به حالت خاموشی، $Q1$ واکنش نشان داده و زمان تأخیر انتشار را کاهش می‌دهد. وقتی همه ورودی‌ها در سطح بالا باشند و آنگاه یکی از ورودی‌ها به سطح پایین برده شود، هر دو ترانزیستور $Q2$ و $Q3$ شروع به خاموش شدن می‌کنند. در این هنگام، پیوند کلکتور $Q1$ معکوس شده و امیتر مستقیم می‌شود، و ترانزیستور $Q1$ برای یک لحظه به ناحیه فعال برده می‌شود. جریان کلکتور $Q1$ از بیس $Q2$ می‌آید و به سرعت بار ذخیره شده در اشباع قبلی $Q2$ را از بین می‌برد. این وضعیت موجب کاهش زمان ذخیره‌سازی مدار در مقایسه با ورودی

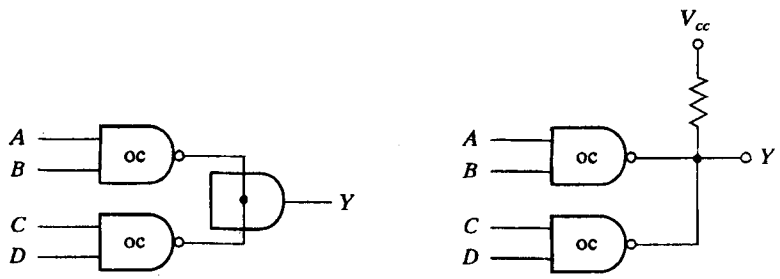
نوع DTL می‌گردد. نتیجه حاصل کاهش زمان خاموشی گیت خواهد بود.

گیت کلکتور باز TTL وقتی که متصل به ورودی گیت‌های TTL دیگر است بدون مقاومت خارجی هم عمل خواهد کرد. با این وجود به علت حد کم پارازیت این کار توصیه نمی‌شود. در غیاب مقاومت خارجی، وقتی $Q3$ خاموش است، خروجی گیت مدار باز خواهد بود. برای یک ورودی گیت TTL ، مدار باز همچون ورودی سطح بالا عمل می‌کند (ولی مقدار کمی پارازیت قادر است وضعیت را عوض کند). وقتی $Q3$ هدایت کند از طریق V_{CC} ، مقاومت $4K\Omega$ و پیوند بیس- امیتر گیت بار دارای مسیر جریانی خواهد شد.

گیت‌های کلکتور باز در سه نوع کاربرد مهم استفاده می‌شوند: راه‌اندازی یک لامپ یارله، اجرای منطقی سیمی، و ایجاد یک گذرگاه مشترک. یک خروجی کلکتور باز می‌تواند یک لامپ واقع در خروجی‌اش را از طریق مقاومت محدودگر روشن کند. وقتی خروجی پایین است، ترانزیستور اشباع شده $Q3$ مسیری برای روشن کردن لامپ ایجاد می‌نماید. وقتی که ترانزیستور خاموش شود، لامپ هم خاموش می‌شود زیرا مسیری برای جریان وجود ندارد.

اگر خروجی چندین گیت TTL کلکتور باز با یک مقاومت به هم گره بخورند، یک منطق AND - سیمی ایجاد می‌شود. به خاطر می‌آورید که یک تابع AND با منطق مثبت هنگامی تولید خروجی سطح بالا می‌کند که همه ورودی‌ها در سطح بالا باشند؛ در غیر این صورت خروجی در سطح پایین است. با اتصال خروجی گیت‌های کلکتور باز، خروجی مشترک فقط هنگامی در سطح بالاست که همه ترانزیستورهای خروجی خاموش (سطح بالا) باشند. اگر یک ترانزیستور خروجی هدایت کند، به ناچار خروجی کل به سطح پایین برده می‌شود.

منطق سیمی با گیت‌های TTL کلکتور باز در شکل ۱۲-۱۰ دیده می‌شود. سیم‌بندی فیزیکی در (الف) چگونگی اتصال به یک مقاومت مشترک را نشان می‌دهد. سمبل گرافیکی برای یک چنین اتصالی در (پ) مشاهده می‌شود. تابع AND تشکیل شده از اتصال دو خروجی به یکدیگر را تابع AND سیمی می‌گویند. گیت AND با عبور خطوط از مرکز گیت ترسیم می‌گردد تا به این ترتیب از گیت معمولی AND تمیز داده شود. گیت AND سیمی یک گیت فیزیکی نیست و تنها علامتی برای معرفی کردن تابع حاصل از اتصال می‌باشد. تابع بولی که از مدار شکل ۱۲-۱۰ بدست می‌آید عمل AND دو خروجی



(الف) اتصال فیزیکی (ب) سمبل گرافیکی منطق سیمی

شکل ۱۲-۱۰ . AND سیمی دو گیت کلکتور باز $Y = (AB + CD)'$

حاصل از دو گیت NAND است:

$$Y = (AB)' \cdot (CD)' = (AB + CD)'$$

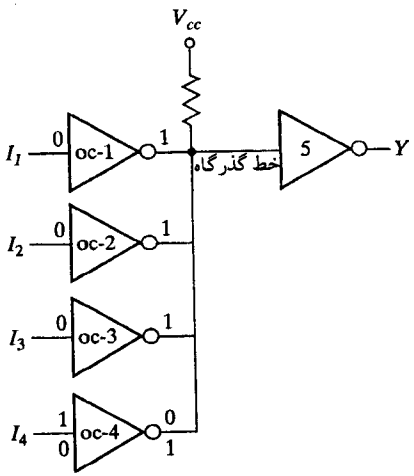
معمولاً عبارت دوم ترجیح داده می‌شود زیرا یک تابع AND– OR– INVERT است (بخش ۷-۳).

گیت‌های کلکتور باز می‌توانند جهت تشکیل یک گذرگاه مشترک به یکدیگر متصل شوند. در هر زمان باید خروجی‌های همه گیت‌های متصل به گذرگاه، به جز یکی، در سطح بالا نگهداری شوند. بسته به این که بخواهیم یک 1 و یا یک 0 به گذرگاه منتقل شود گیت مربوطه در سطح بالا یا پایین خواهد بود. از مدارهای کنترل، برای انتخاب گیت خاصی که گذرگاه را در هر زمان راه می‌اندازد، باید استفاده کرد.

شکل ۱۳-۱۰ اتصال چهار منبع متصل به یک خط گذرگاه مشترک را نشان می‌دهد. هر یک از چهار ورودی یک وارونگر کلکتور باز را راه می‌اندازد، و خروجی وارونگرها برای ایجاد یک خط گذرگاه مشترک، به هم وصل شده‌اند. شکل مذکور نشان می‌دهد که سه ورودی 0 می‌باشند که تولید 1 را روی گذرگاه خواهند نمود. اکنون چهارمین ورودی I_4 قادر است اطلاعات را از طریق خط گذرگاه مشترک به وارونگر 5 منتقل نماید. به خاطر می‌آورید که عمل AND با منطق سیمی انجام شد. اگر I_4=1 باشد خروجی گیت 4 برابر 0 و عمل AND سیمی یک 0 تولید می‌کند. اگر I_4=0 باشد خروجی گیت شماره 4 برابر 1 و عمل AND سیمی یک 1 تولید می‌نماید. بنابراین اگر همه خروجی‌های دیگر در 1 نگهداری شوند، گیت انتخابی قادر خواهد بود تا مقدارش را از طریق گذرگاه ارسال نماید. مقدار ارسال شده متمم I_4 است، و وارونگر 5 در سمت گیرنده به راحتی خواهد توانست این سیگنال را معکوس کرده و $Y=I_4$ را تولید نماید.

خروجی تاتم پل

امپدانس خروجی گیت معمولاً بار مقاومتی بعلاوه بار خازنی است. بار خازنی متشکل از ظرفیت ترانزیستور خروجی، ظرفیت گیت‌های گنجایش خروجی، و ظرفیت هر سیم‌بندی پراکنده است. وقتی

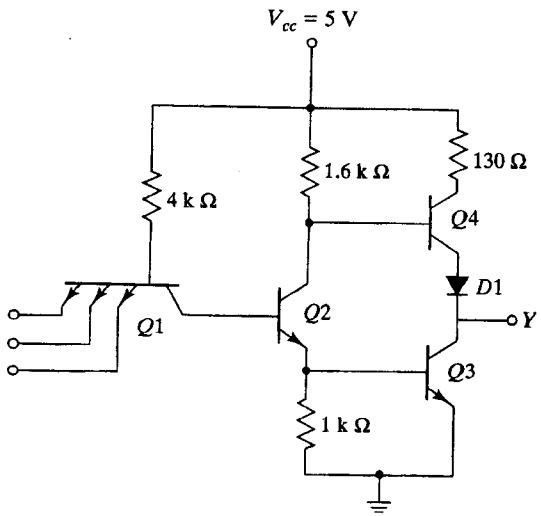


شکل ۱۳-۱۰ . گیت‌های کلکتور باز تشکیل دهنده یک گذرگاه مشترک

خروجی از حالت پایین به حالت بالا تغییر پیدا می‌کند، ترانزیستور خروجی گیت، از اشباع به حالت قطع می‌رود و ظرفیت بارکل C به صورت نمایی از سطح ولتاژ پایین به بالا با ثابت زمانی RC شارژ می‌گردد. برای گیت کلکتور باز، R یک مقاومت خارجی است و با R_L نشان داده می‌شود. برای یک مقدار عملی نمونه $C = 15PF$ و $R_L = 4K\Omega$، تأخیر زمانی گیت TTL کلکتور باز در حین زمان خاموشی 35ns است. با جایگزینی یک مدار بالاکش فعال به جای مقاومت بالاکش R_L، تأخیر انتشار به 10ns کاهش می‌یابد. این آرایش که در شکل ۱۴-۱۰ آمده است به دلیل نشستن ترانزیستور $Q4$ بر روی $Q3$ تاتم پل نامیده شده است.

گیت TTL با خروجی تاتم پل مشابه گیت کلکتور باز است. و فقط در ترانزیستور $Q4$ و دیود $D1$ تفاوت دارد. وقتی که خروجی Y در حالت منطقی پایین است، $Q2$ و $Q3$ مثل گیت کلکتور باز به اشباع رانده می‌شوند. ولتاژ در کلکتور $Q2$ برابر است با $V_{BE}(Q3) + V_{CE}(Q2)$ یا $0.7 + 0.2 = 0.9V$. خروجی $Y = V_{CE}(Q3) = 0.2V$ است. ترانزیستور $Q4$ خاموش خواهد بود زیرا بیس آن باید یک افت V_{BE} بعلاوه یک افت دیود یا $2 \times 0.6 = 1.2V$ باشد تا هدایت آغاز شود. چون کلکتور $Q2$ به بیس $Q4$ وصل است، در عوض 1.2V، برابر 0.9V می‌باشد، و در نتیجه $Q4$ قطع یا خاموش خواهد بود. دلیل استقرار دیود در مدار، یک افت دیود در مسیر خروجی است تا به هنگام اشباع $Q3$، ترانزیستور $Q4$ قطع باقی بماند.

وقتی که خروجی به سطح بالا تغییر وضعیت دهد، چون یکی از ورودی‌ها به سطح پایین می‌افتد، ترانزیستورهای $Q2$ و $Q3$ به حالت خاموش می‌روند. با این وجود، خروجی برای یک لحظه پایین باقی می‌ماند، زیرا ولتاژ دو سر بار خازنی بلافاصله تغییر نمی‌کند. به محض خاموش شدن $Q2$، $Q4$ هدایت می‌نماید زیرا بیس آن از طریق مقاومت $1.6K\Omega$ به V_{CC} وصل می‌شود. جریان لازم برای شارژ بار خازنی سبب می‌شود تا $Q4$ برای یک لحظه اشباع گردد، ولتاژ خروجی با ثابت زمانی RC بالا می‌رود. اما R در این حالت معادل 130Ω، بعلاوه مقاومت اشباع $Q4$، بعلاوه مقاومت دیود و در مجموع 150Ω



شکل ۱۴-۱۰. گیت TTL با خروجی تاتم پل

است. این مقدار R خیلی کوچکتر از مقاومت بالاکش غیرفعال به کار رفته در مدار کلکتور باز است. در نتیجه، گذر از سطح پایین به سطح بالا خیلی سریع‌تر است.

با شارژ بار خازنی، ولتاژ خروجی بالا می‌رود و جریان در $Q4$ کاهش می‌یابد و ترانزیستور را به ناحیه فعال می‌برد. بنابراین برخلاف دیگر ترانزیستورها، $Q4$ در حالت سکون در ناحیه فعال خواهد بود. در این صورت مقدار نهایی ولتاژ خروجی برابر است با 5V منهای افت ولتاژ V_{BE} در $Q4$، منهای افت دو سر دیود $D1$ و در حدود 3.6V خواهد بود. ترانزیستور $Q3$ به سرعت خاموش می‌شود، ولی در اوایل زمان گذر هر دو ترانزیستور روشن هستند و جریان ماکزیممی از منبع تغذیه کشیده می‌شود. این جریان ناگهانی در سیستم توزیع منبع تغذیه تولید پارازیت می‌کند. وقتی تغییر حالت تکرار شود، تغییرات لحظه‌ای جریان موجب بالا رفتن جریان از منبع شده و بدین ترتیب توان مصرف مدار افزایش می‌یابد.

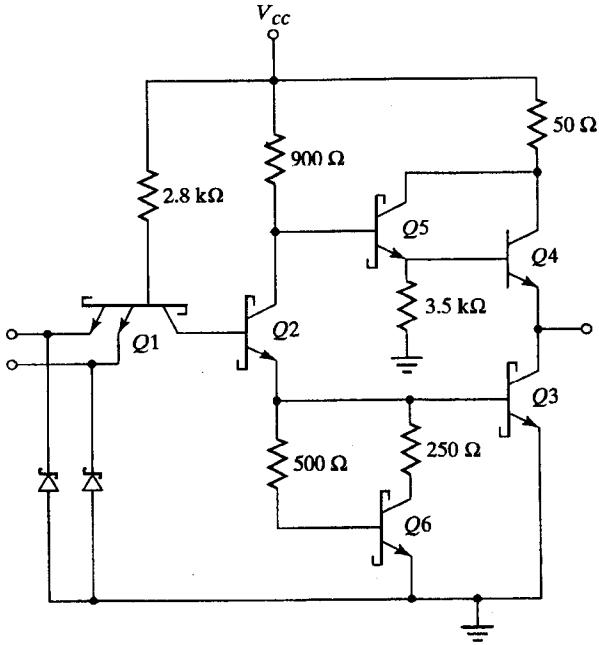
اتصال منطق سیمی با مدارهای تاتم پل مجاز نیست. وقتی دو خروجی تاتم پل به هم پیوسته شوند و خروجی یکی از آنها بالا و دومی پایین باشد، جریان زیادکشیده شده سبب می‌گردد تا گرمای تولید شده از آن به ترانزیستورهای مدار آسیب برساند (مسئله ۷-۱۰). بعضی از گیت‌های TTL طوری طراحی شده‌اند تا این مقدار جریان را تحمل کند. در هر صورت جریان کلکتور در گیت پایین ممکن است تا آن حد بالا باشد که ترانزیستور را به ناحیه فعال برده و یک ولتاژ خروجی بزرگتر از 0.8V در محل اتصال سیمی به وجود آورد که از نظر سیگنال دودویی برای گیت‌های TTL مجاز نیست.

گیت TTL شوتکی

همانطور که قبلاً اشاره شد، کاهش زمان ذخیره‌سازی موجب کاهش زمان تأخیر انتشار می‌گردد. این بدان علت است که زمان لازم برای خروج ترانزیستور از حالت اشباع موجب تأخیر سوئیچینگ ترانزیستور در رفتن از حالت روشن به حالت خاموش می‌گردد. اشباع را می‌توان با استقرار یک دیود شوتکی بین بیس و کلکتور هر ترانزیستور اشباع شده در مدار برطرف کرد. دیود شوتکی از اتصال یک فلز به یک نیمه هادی ساخته می‌شود، در حالی که دیود معمولی از پیوند مواد نیمه هادی نوع p و n به وجود می‌آید. ولتاژ دو سر دیود شوتکی در حال هدایت تنها 0.4V در مقابل 0.7V در دیود معمولی است. وجود دیود شوتکی بین بیس و کلکتور مانع رفتن ترانزیستور به اشباع می‌گردد. ترانزیستور حاصل، **ترانزیستور شوتکی** خوانده می‌شود. استفاده از دیود شوتکی در یک گیت TTL، تأخیر انتشار را بدون افزایش توان تلف شده، کاهش می‌دهد.

گیت TTL شوتکی در شکل ۱۵-۱۰ مشاهده می‌گردد. به سمبل خاص به کار رفته برای ترانزیستورها و دیودهای شوتکی توجه نمایید. همانطور که ملاحظه می‌شود همه ترانزیستورها به جز $Q4$ از نوع شوتکی‌اند زیرا این ترانزیستور به اشباع نمی‌رود، بلکه در ناحیه فعال باقی می‌ماند. همچنین توجه کنید که مقادیر مقاومت کاهش یافته‌اند تا تأخیر انتشار کم شود.

علاوه بر استفاده از ترانزیستورهای شوتکی و پایین آمدن مقاومت‌ها، در مدار شکل ۱۵-۱۰ دو تصحیح دیگر لحاظ شده است که در گیت استاندارد شکل ۱۴-۱۰ وجود ندارد. در این شکل دو



شکل ۱۵-۱۰. گیت TTL شوتکی

ترانزیستور $Q5$ و $Q6$ اضافه شده است و در بین هر پایانه ورودی و زمین یک دیود شوتکی قرار دارد. در مدار تاتم پل دیودی وجود ندارد. با این وجود، ترکیب جدید $Q5$ و $Q4$ هنگام پایین بودن خروجی، دو افت V_{BE} تولید می‌نماید تا از هدایت $Q4$ ممانعت شود. این ترکیب یک جفت امیتر فالوور را به نام جفت دارلینگتون به وجود می‌آورد. جفت دارلینگتون بهره بسیار بالا و مقاومت خیلی کم را فراهم می‌سازد. این دقیقاً همان چیزی است که هنگام تغییر خروجی مورد نیاز است، و در نتیجه زمان تأخیر انتشار کاهش می‌یابد.

دیودهایی که در هر ورودی نشان داده شده‌اند هرگونه افزایش در خطوط ورودی را برش می‌دهند. تحت شرایط سوئیچنگ، خطوط سیگنال، القایی به نظر می‌رسند؛ این خاصیت به همراه ظرفیت پارازیتی مدار سبب می‌شود تا سیگنال‌ها نوسان یا چرخش کنند. وقتی خروجی یک گیت از سطح بالا به پایین سوئیچ می‌کند، موج نوسانی می‌تواند 2 تا 3 ولت از زمین منفی‌تر شود. با توجه به این که دیودهای متصل به زمین ولتاژ منفی بیش از 0.4V را هدایت می‌نماید به بریده شدن این ورودی‌ها کمک می‌کنند. وقتی منفی شدن محدود گردد، نوسان مثبت نیز کاهش می‌یابد. کارایی دیودهای برش‌گر در محدود کردن اثرات خط آن قدر بالاست که در تمام نسل‌های گیت‌های TTL از آنها استفاده می‌شود.

مقاومت امیتر $Q2$ در شکل ۱۴-۱۰ به وسیله مداری متشکل از ترانزیستور $Q6$ و دو مقاومت در شکل ۱۵-۱۰ جایگزین شده است. نقش این مدار کاهش جهش‌های جریان قطع بحث شده قبلی است. تحلیل این مدار که به کاهش زمان انتشار گیت کمک می‌کند آن قدر پیچیده است که نمی‌توان آن را در این بحث مختصر ارائه نمود.

گیت‌سه حالته

همانطورکه قبلاً ذکر شد، خروجی‌های دوگیت TTL با ساختار تاتم پل نمی‌توانند مثل خروجی‌های کلکتور باز به هم وصل شوند. با این وجود نوع خاصی از تاتم پل وجود دارد که سیم‌بندی خروجی‌ها را برای ایجاد یک سیستم گذرگاه مشترک ممکن می‌سازد. هنگامی که یک خروجی تاتم پل دارای چنین خصوصیتی است، آن را گیت سه حالته می‌خوانند.

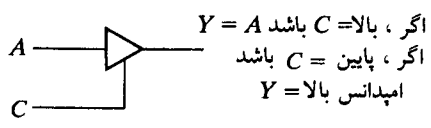
یک گیت سه حالته، سه وضعیت خروجی را به نمایش می‌گذارد: (۱) یک حالت سطح پایین به هنگام روشن شدن ترانزیستور پایینی و خاموش ماندن ترانزیستور بالایی در تاتم پل؛ (۲) یک حالت سطح بالا به هنگام روشن بودن ترانزیستور بالایی و خاموش ماندن ترانزیستور پایینی؛ (۳) حالت سوم هنگامی است که هر دو ترانزیستور در تاتم پل خاموش باشند. حالت اخیر مدار باز یا حالت امپدانس بالا را در خروجی ایجاد می‌نماید و اجازه می‌دهد تا چندین خروجی به طور مستقیم به هم متصل شوند. وجود گیت‌های سه حالته در گذرگاه‌ها نیاز به گیت‌های کلکتور باز در مدار را از بین می‌برد.

شکل ۱۶-۱۰ (الف) سمبل گرافیکی یک گیت بافر سه حالته را نشان می‌دهد. وقتی که ورودی کنترل C بالا باشد، گیت فعال شده و مانند یک بافر عمل نموده و خروجی‌اش برابر مقدار دودویی ورودی خواهد بود. اگر ورودی کنترل در سطح منطقی پایین باشد، خروجی به فرم مدار باز بوده و امپدانس بالایی را بدون توجه به مقدار ورودی A به نمایش می‌گذارد (سومین حالت). برخی از گیت‌های سه حالته، امپدانس بالا را به هنگام بالا بودن ورودی کنترل تولید می‌کنند. این مطلب به فرم سمبلیک در شکل ۱۶-۱۰ (ب) نشان داده شده است. در این شکل دو دایره کوچک، یکی برای خروجی وارونگر، و دیگری برای فعال شدن گیت در ازاء ورودی سطح پایین C ملاحظه می‌شود.

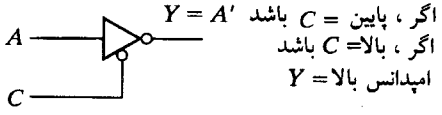
مدار یک وارونگر سه حالته در شکل ۱۶-۱۰ (پ) نشان داده شده است. ترانزیستورهای $Q6$، $Q7$ و $Q8$ مربوط به بخش کنترل، مداری مشابه با گیت کلکتور باز را تشکیل می‌دهند. ترانزیستورهای $Q1-Q5$ مربوط به بخش ورودی یک مدار از نوع تاتم پل را نمایش می‌دهند. دو مدار از طریق دیود $D1$ به هم مرتبط شده‌اند. همچون مدار کلکتور باز، وقتی که ورودی C در سطح پایین باشد ترانزیستور $Q8$ خاموش می‌شود. این حالت مانع هدایت دیود $D1$ می‌گردد. بعلاوه، امیتر $Q1$ که به $Q8$ وصل است دارای مسیر هدایت نخواهد بود. تحت این شرایط ترانزیستور $Q8$، اثری روی عمل گیت نداشته و خروجی در Y تنها به داده ورودی در A وابسته است.

وقتی که ورودی کنترل در سطح بالا باشد، ترانزیستور $Q8$ روشن می‌شود و جریان از V_{CC} به دیود $D1$ موجب می‌شود تا این ترانزیستور اشباع گردد. اکنون ولتاژ در بیس $Q5$ برابر با ولتاژ دو سر ترانزیستور اشباع شده $Q8$، بعلاوه یک افت دیود، یعنی 0.9V خواهد بود. این ولتاژ $Q5$ و $Q4$ را خاموش می‌کند، زیرا کمتر از دو افت V_{BE} است. در همین زمان ورودی پایین به یکی از امیترهای $Q1$ موجب خاموش شدن $Q3$ (و $Q2$) می‌گردد. بنابراین هر دو $Q3$ و $Q4$ در تاتم پل خاموش و خروجی مدار همچون مدار باز با امپدانس خیلی بالا عمل خواهد کرد.

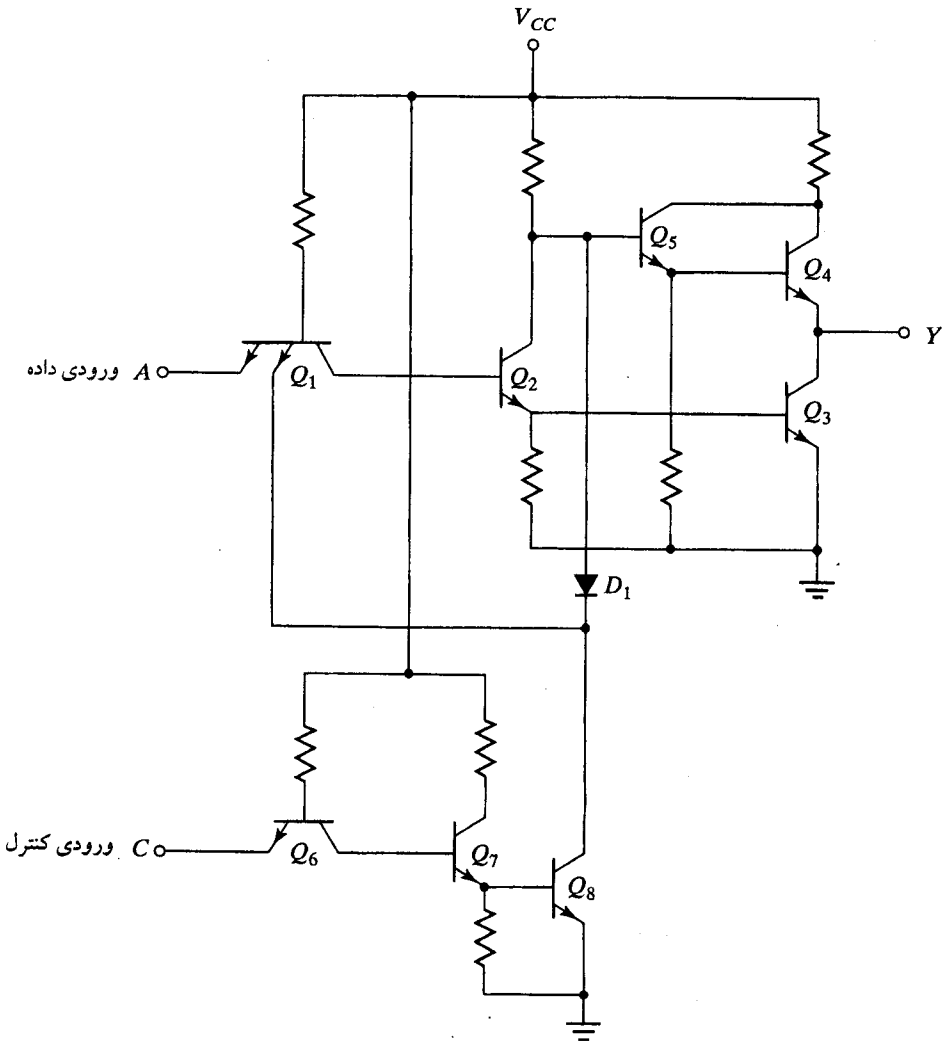
یک گذرگاه سه حالته با وصل کردن چندین سیم به چند خروجی سه حالته حاصل می‌شود. در هر



(الف) گیت بافر سه حالته



(ب) گیت وارونگر سه حالته



(پ) نمودار مدار برای وارونگر شکل (ب)

شکل ۱۶-۱۰. گیت TTL سه گانه

لحظه فقط یکی از ورودی‌های کنترل فعال شده و خروجی مربوط به آن به گذرگاه مشترک وصل می‌شود ضمن این که همه دیگر خروجی‌ها در حالت امپدانس بالا هستند. تک گیتی که در امپدانس بالا نیست قادر به انتقال اطلاعات از گذرگاه مشترک است. توجه دارید که به جز یک خروجی، همه آنها در حالت سوم هستند، در غیر این صورت وضعیت نابهنجار اتصال دو خروجی تاتم پل را خواهیم داشت.

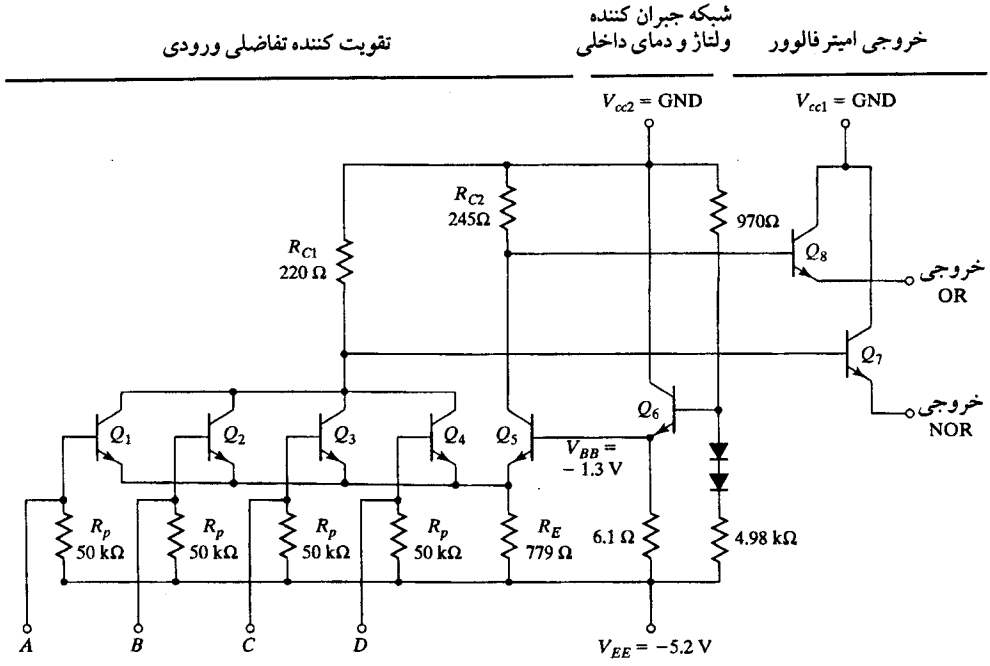
ویژگی خاص بسیاری از گیت‌های سه حالته این است که تأخیر فعال‌ساز خروجی طولانی‌تر از غیرفعال‌ساز آن است. اگر یک مدار کنترل، گیتی را فعال و همزمان با آن دیگری را غیرفعال کند، گیت فعال نشده قبل از فعال شده به حالت امپدانس بالا خواهد رفت. این خصوصیت مانع فعال شدن همزمان دو گیت می‌گردد.

در گیت سه حالته جریان نشتی کمی در وضعیت امپدانس بالا وجود دارد. معهذا این جریان به قدری کوچک است که می‌توان 100 عدد خروجی سه‌حالته را برای تشکیل یک گذرگاه سه‌حالته به هم متصل کرد.

۱۰-۶ منطق کوپلاژ امیتر ECL

منطق کوپلاژ امیتر ECL ، یک خانواده منطقی دیجیتال غیراشباع است. چون ترانزیستورها اشباع نمی‌شوند، ممکن است به تأخیرهای زمانی حدود 1–2ns دست یافت. این خانواده منطقی دارای پایین‌ترین تأخیر انتشار در مقایسه با دیگر خانواده‌هاست و اغلب در سیستم‌هایی که سرعت بالایی دارند استفاده می‌شود. با این وجود حد پارازیت و توان مصرفی آن بدترین است.

نمونه‌ای از یک مدار پایه خانواده ECL در شکل ۱۰-۱۷ ملاحظه می‌شود. خروجی‌ها، هر دو تابع OR و NOR را در اختیار می‌گذارند. هر ورودی به پایه یک ترانزیستور متصل است. دو سطح ولتاژ



شکل ۱۰-۱۷. گیت پایه ECL

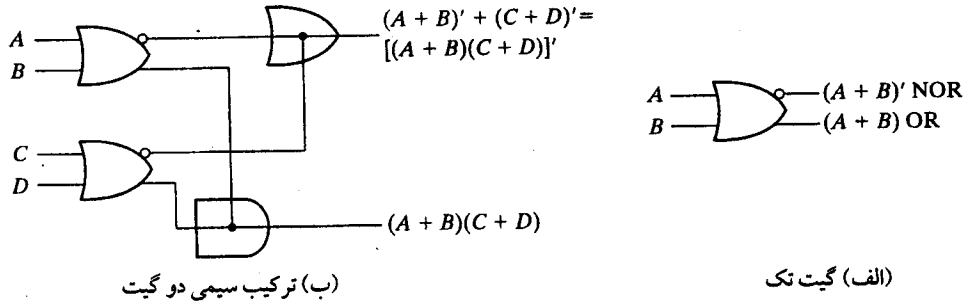
عبارتند از 0.8V− برای سطح بالا و حدود 1.8V− برای سطح پایین. مدار شامل تقویت کننده تفاضلی، یک جبران‌ساز دما و ولتاژ تغذیه، و یک خروجی امیتر فالوور (کلکتور مشترک) است. خروجی‌های امیتر برای راه افتادن جریان نیاز به مقاومت پایین کش دارند. این نیاز، از مقاومت ورودی، R_P، مربوط به گیت دیگر مشابه یا یک مقاومت بیرونی متصل به یک منبع ولتاژ منفی فراهم می‌گردد.

مدار داخلی جبران کننده ولتاژ- دما، ولتاژ مرجعی را برای تقویت کننده تفاضلی فراهم می‌سازد. ولتاژ تغذیه V_{BB} در 1.3V− تنظیم می‌گردد، که وسط محدوده دو سطح منطقی است. دیودها در تقسیم کننده ولتاژ به همراه $Q6$ مداری را تشکیل می‌دهند که علیرغم تغییر دما یا ولتاژ منبع، مقدار V_{BB} ثابت می‌ماند. هر یک از ورودی‌های منبع می‌تواند به عنوان زمین مورد استفاده قرار گیرد. با این وجود استفاده از نقطه V_{CC} به عنوان زمین و V_{EE} در 5.2V− بهترین حد پارازیت حاصل می‌شود.

اگر هر یک از ورودی‌ها در گیت ECL بالا برود، ترانزیستور مربوطه روشن و $Q5$ خاموش می‌شود. یک ورودی 0.8V− ، ترانزیستور را روشن کرده و سبب قرار گرفتن 1.6V− روی امیتر همه ترانزیستورها می‌گردد (افت V_{BE} در ترانزیستورهای ECL برابر 0.8V است). چون $V_{BB} = -1.3V$ است ولتاژ بیس $Q5$ تنها 0.3V مثبت‌تر از امیتر می‌باشد. با توجه به این که V_{BE} ترانزیستور $Q5$ به حداقل 0.6V برای شروع هدایت نیاز دارد بنابراین $Q5$ خاموش خواهد بود. جریان در مقاومت R_{C2} به بیس $Q8$ می‌رود (به شرط این که مقاومت باری وجود داشته باشد). این جریان به قدری کوچک است که فقط افت ولتاژ ناچیزی در دو سر R_{C2} به وجود می‌آید. خروجی OR گیت به اندازه یک افت V_{BE} کمتر از زمین، یعنی 0.8V− ، که سطح منطقی بالاست، می‌باشد. جریان در R_{C1} و ترانزیستور روشن سبب افت ولتاژ 1V کمتر از زمین می‌شود (مسئله ۹-۱۰). خروجی NOR یک افت V_{BE} زیر این سطح، یعنی 1.8V− است که سطح پایین منطقی می‌باشد.

اگر همه ورودی‌ها در سطح پایین باشند، همه ترانزیستورهای ورودی خاموش و $Q5$ روشن می‌گردد. ولتاژ در نقطه مشترک امیترها افتی برابر با یک V_{BE} زیر V_{BB} یعنی 1.2V− خواهد داشت. چون بیس مربوط به هر ورودی در سطح پایین 1.8V− است، هر پیوند بیس- امیتر تنها دارای 0.3V بوده و تمام ترانزیستورهای ورودی خاموش خواهند بود. جریان R_{C2} و $Q5$ سبب افت ولتاژی حدود 1V شده، و بنابراین خروجی OR یک V_{BE} به زیر این مقدار افت می‌نماید که مقدار آن 1.8V− بوده و سطح پایین منطقی خواهد بود. جریان در R_{C1} ناچیز بوده و خروجی NOR یک V_{BE} کمتر از زمین یعنی 0.8V− می‌شود که سطح بالای منطقی است. این وضعیت‌ها عملیات OR و NOR مدار را تأیید می‌کنند.

تأخیر انتشار گیت ECL ، 2ns است و توان تلف شده آن 25mW می‌باشد. این کمیت‌ها، حاصلضرب سرعت- توان 50 را بدست می‌دهند که برابر با حاصلضرب مشابه در TTL شوتکی است. حد پارازیت حدود 0.3V است ولی به خوبی گیت TTL نیست. امکان گنجایش خروجی زیادی به دلیل وجود امپدانس ورودی بالای تقویت کننده تفاضلی و امپدانس خروجی کم در امیتر فالوور، در گیت ECL وجود دارد. به دلیل سرعت قابل توجه سیگنال‌ها، سیم‌های بیرونی مانند خطوط انتقال عمل می‌نمایند. به جز مواردی که طول سیم حدود چند سانتی‌متر است، خروجی‌های ECL باید از کابل‌های



شکل ۱۸-۱۰ . سمبل‌های گرافیکی گیت‌های ECL

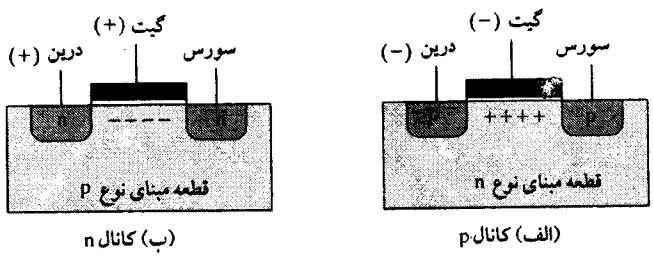
کواکسیال (هم محور) استفاده کنند و در انتهای آنها نیز باید یک مقاومت نصب شود تا از انعکاس جلوگیری نماید.

سمبل گرافیکی گیت ECL در شکل ۱۸-۱۰ (الف) ملاحظه می‌شود. دو خروجی در دسترسند: یکی برای تابع OR و دیگری تابع NOR. خروجی‌های دو یا چند گیت ECL را می‌توان به هم وصل کرد تا منطق سیمی بوجود آید. طبق شکل ۱۸-۱۰ (ب) اتصال سیمی دو خروجی NOR یک اتصال OR به وجود می‌آورد. در بعضی از آی‌سی‌های ECL اتصال درونی خروجی‌های OR تولید AND سیمی می‌کند (بعضی به آن AND – dot یا AND نقطه‌ای می‌گویند). در مواردی که از گیت‌های ECL برای ایجاد AND–INVERT – OR و توابع AND–OR استفاده می‌شود، این خاصیت به‌کار می‌رود.

۱۰-۷ فلز-اکسید-نیمه‌هادی (MOS)

ترانزیستور اثر میدان (FET) نوعی ترانزیستور تک قطبی است، زیرا عملکرد آن فقط به جریان یک نوع حامل وابسته است. دو نوع ترانزیستور اثر میدان وجود دارد: ترانزیستور اثر میدان پیوندی، JFET و فلز-اکسید-نیمه‌هادی، MOS. نوع اول در مدارهای خطی و دومی در مدارهای دیجیتال به کار می‌رود. در مقایسه با ترانزیستورهای دو قطبی، ترانزیستورهای MOS سطح کمتری را در ساخت اشغال می‌کنند.

ساختار ترانزیستور مبنای MOS در شکل ۱۹-۱۰ دیده می‌شود. MOS کانال p متشکل از قطعه بدنه‌ای است که از ماده سیلیکان نوع n با ناخالصی کم می‌باشد. دو ناحیه‌ای که ناخالصی p آنها بالاست سورس و درین را تشکیل می‌دهند. ناحیه بین دو بخش نوع p نقش کانال را به عهده دارد. گیت



شکل ۱۹-۱۰ . ساختار پایه ترانزیستور MOS

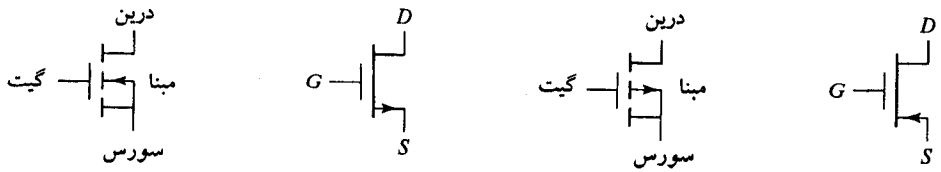
یک صفحه فلزی است که توسط یک دی‌الکتریک از جنس دی اکسید سیلیکان از کانال جدا شده است. یک ولتاژ منفی (نسبت به بدنه) در گیت موجب القاء میدان الکتریکی در کانال شده و حامل‌های نوع p را جذب می‌نماید. با افزایش مقدار ولتاژ منفی در گیت، ناحیه زیر گیت حامل‌های مثبت بیشتری را در خود جای داده و هدایت افزایش می‌یابد و جریان از سورس به درین جاری شده و افت ولتاژهای بین این دو پایانه به وجود می‌آید.

چهار نوع ساختار برای MOS وجود دارد. بسته به این که حامل‌های اکثریت از نوع حفره یا الکترون باشند، کانال از نوع p و یا n خواهد بود. اگر به کانال p مقدار کمی ناخالص تزریق شود، کانال در ولتاژ گیت صفر ولت هدایت خواهد کرد و در این حال گوییم عملکرد در مد تهی است. در این مد، جریان جاری خواهد بود مگر این که کانال به وسیله میدان حاصل از گیت تهی شود. اگر ناحیه زیر گیت ابتدا بدون بار باشد، کانال باید با میدان گیت القاء شود تا جریان بتواند جاری گردد. بنابراین جریان کانال با ولتاژ گیت افزایش می‌یابد که در این صورت گوییم قطعه در مد افزایشی یا پیشرفته کار می‌کند.

سورس پایانه‌ای است که از آن حامل‌های اکثریت وارد کانال میله‌ای شکل می‌شوند. درین پایانه‌ای است که از آن طریق حامل‌های اکثریت کانال را ترک می‌نمایند. در یک MOS کانال p، پایانه سورس به بدنه و درین به ولتاژ منفی متصل می‌گردند. وقتی که ولتاژ گیت بالاتر از ولتاژ آستانه V_T (حدود $-2V$) باشد، جریانی درکانال برقرار نیست و مسیر درین به سورس مانند یک مدار باز است. هنگامی که ولتاژ گیت به اندازه کافی منفی‌تر از V_T باشد، کانال ایجاد شده و حامل‌های نوع p از سورس به درین خواهند رفت. حامل‌های نوع p مثبتند و به جریان مثبت از سورس به درین وابسته‌اند.

در MOS کانال n، پایانه سورس به بدنه و ولتاژ مثبت به درین وصل است. وقتی که ولتاژ گیت به قدر کافی مثبت‌تر از V_T برای تشکیل کانال باشد، حامل‌های نوع n از سورس به درین می‌روند. حامل‌های n منفی‌اند، و متعلق به جریان مثبت از درین به سورس است. ولتاژ آستانه ممکن است بسته به فرآیند به کار رفته بین 1 الی 4V تغییر نماید.

سمبل‌های گرافیکی برای ترانزیستورهای MOS در شکل ۲۰-۱۰ دیده می‌شود. سمبل نوع افزایشی به صورت خط منقطع بین سورس و درین می‌باشد. در این سمبل بدنه به سورس وصل شده است. ما از سمبل دیگری که در آن از بدنه چشم‌پوشی شده است نیز استفاده خواهیم کرد. در این سمبل فلش در پایانه سورس برای نمایش جهت جریان مثبت به کار گرفته شده است. یادآوری می‌شود که جهت جریان در کانال p از سورس به درین و در کانال n از درین به سورس می‌باشد.



(الف) کانال p (ب) کانال n

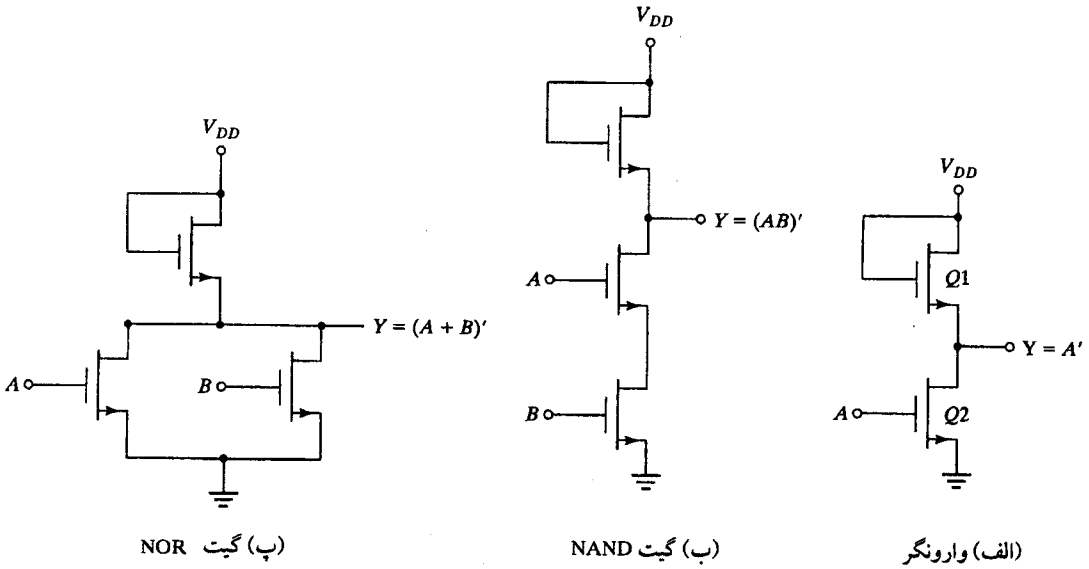
شکل ۲۰-۱۰ . علائم ترانزیستورهای MOS

به دلیل تقارن ساختار سورس و درین، ترانزیستور MOS قادر است به صورت یک قطعه دو جهته کار کند. گرچه در حالت معمولی جهت حرکت حامل‌ها از سورس به درین است. مواردی وجود دارد که بهتر است حامل‌ها از درین به سورس بروند (مسئله ۱۲-۱۰ ملاحظه شود).

یکی از مزایای قطعه MOS این است که نه تنها به عنوان ترانزیستور به کار می‌رود بلکه به عنوان مقاومت نیز مورد استفاده قرار می‌گیرد. با تغذیه دائم پایانه گیت می‌توان از MOS مقاومت ساخت. سپس نسبت به ولتاژ سورس- درین به جریان کانال، مقدار مقاومت را تعیین می‌کند. مقادیر مختلف مقاومت را می‌توان به هنگام ساخت با تثبیت طول کانال و عرض قطعه MOS تعیین کرد.

سه مدار منطقی که از MOS استفاده می‌کنند در شکل (۲۱-۱۰) نشان داده شده است. برای MOS کانال n ولتاژ تغذیه V_{DD} مثبت است (حدود 5V) تا جریان مثبتی را از درین به سورس به وجود آورد. دو سطح ولتاژ منطقی تابعی از ولتاژ آستانه V_T هستند. سطح پایین از صفر تا V_T و سطح بالا از V_T تا V_{DD} است. گیت‌های کانال n معمولاً از منطق مثبت استفاده می‌کنند. مدارهای MOS کانال p ولتاژ منفی را برای V_{DD} به کار می‌برند تاجریان مثبت از سورس به درین جاری شود. دو سطح ولتاژ هر دو منفی و در بالا و پایین ولتاژ آستانه V_T قرار دارند. گیت‌های کانال p معمولاً منطق منفی را به کار می‌برند.

مدار وارونگری که در شکل ۲۱-۱۰ (الف) ملاحظه می‌شود از MOS استفاده کرده است. $Q1$ به عنوان مقاومت بار و $Q2$ به عنوان یک قطعه فعال عمل می‌نمایند. گیت مقاومت بار MOS به V_{DD} وصل است و بنابراین همیشه آن را روشن نگه می‌دارد. وقتی که ولتاژ ورودی پایین است (زیرا V_T)، $Q2$ خاموش می‌شود. چون $Q1$ همیشه روشن است، ولتاژ خروجی حدود V_{DD} خواهد بود. هنگامی که ولتاژ ورودی بالاست (بیش از V_T)، $Q2$ روشن می‌گردد. جریان از V_{DD} از طریق مقاومت بار $Q1$ به $Q2$



(پ) گیت NOR (ب) گیت NAND (الف) وارونگر

شکل ۲۱-۱۰ . مدارهای منطقی MOS کانال n

می‌رود. ابعاد دو قطعه MOS باید طوری باشد که هنگام هدایت مقاومت $Q2$ خیلی کوچکتر از مقاومت $Q1$ باشد تا ولتاژ خروجی Y در کمتر از V_T نگهداری شود.

گیت NAND که در شکل ۲۱-۱۰ (ب) دیده می‌شود از ترانزیستورهای سری استفاده می‌کند. هر دو ورودی A و B برای همه ترانزیستورها باید در سطح بالا باشند تا هدایت نمایند و بنابراین خروجی به پایین برود. اگر هر یک از دو ورودی در سطح پایین باشد، ترانزیستور متعلق به آن خاموش شده و خروجی بالا می‌رود. مجدداً یک مقاومت سری که با دو وسیله MOS فعال تشکیل می‌شود باید خیلی کمتر از مقاومت بار MOS در مدار باشد، گیت NOR شکل ۲۱-۱۰ (پ) از ترانزیستورهای موازی استفاده می‌کند. اگر هر یک از ورودی‌ها بالا باشد ترانزیستور مربوطه روشن و خروجی پایین خواهد رفت. اگر همه ورودی‌ها پایین باشند، همه ترانزیستورهای فعال خاموش شده و خروجی به سطح بالا خواهد رفت.

۱۰-۸ MOS متمم (CMOS)

مدارهای MOS متمم از این مزیت سود می‌برند که هر دو نوع قطعه کانال p و n بر روی یک بدنه قابل ساخت‌اند. مدارهای CMOS از دو نوع قطعه MOS در درون به هم متصلند تا توابع منطقی را ایجاد کنند. مدار پایه، وارونگر است، که از یک ترانزیستور کانال p و یک ترانزیستور کانال n طبق شکل ۲۲-۱۰ (الف) ساخته می‌شود. پایانه سورس قطعه کانال p در V_{DD} و پایانه سورس قطعه کانال n به زمین وصل است. مقدار V_{DD} می‌تواند در هر جا بین 3+ تا 18V+ باشد. دو سطح ولتاژ عبارتند از 0V برای سطح پایین و V_{DD} برای سطح بالا که معمولاً 5V است.

برای درک عملکرد وارونگر، باید رفتار ترانزیستور MOS را از دو بخش قبل مرور کنیم:

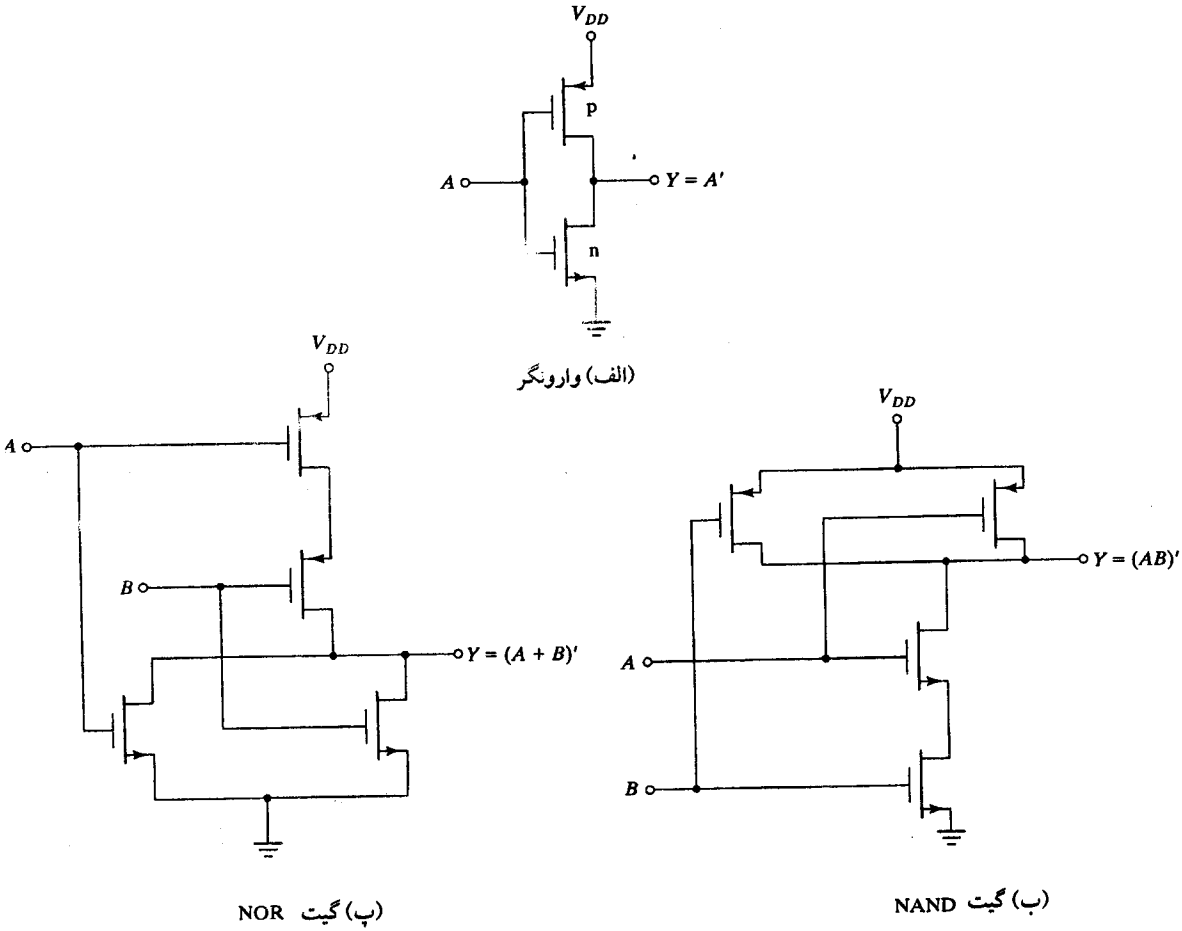
۱- MOS کانال n وقتی هدایت می‌کند که ولتاژ گیت- سورس مثبت باشد.

۲- MOS کانال p هنگامی هدایت می‌نماید که ولتاژ گیت- سورس منفی است.

۳- هر یک از دو نوع در ازاء ولتاژ صفر بین گیت- سورس، خاموش است.

اکنون به طرز کار وارونگر توجه کنید. وقتی ورودی پایین است، هر دو گیت در پتانسیل صفر هستند. ورودی نسبت به سورس قطعه کانال p در $V_{DD}-$ و نسبت به سورس قطعه کانال n در 0V است. نتیجه این که قطعه کانال p روشن و قطعه کانال n خاموش خواهد بود. تحت این شرایط یک مسیر با امپدانس کم از V_{DD} به خروجی و مسیری با امپدانس خیلی زیاد از خروجی به زمین برقرار می‌گردد. بنابراین، ولتاژ خروجی تحت شرایط بار معمولی به سمت ولتاژ بالای V_{DD} میل خواهد کرد. وقتی که ورودی بالاست، هر دو گیت در V_{DD} بوده و وضعیت معکوس می‌گردد. قطعه کانال p خاموش و کانال n روشن خواهد شد. در نتیجه خروجی به سمت 0V می‌رود.

دو گیت پایه دیگر CMOS در شکل ۲۲-۱۰ ملاحظه می‌گردد. یک گیت NAND دو ورودی از دو بخش موازی نوع p و دو بخش سری نوع n طبق شکل ۲۲-۱۰ (ب) تشکیل شده است. اگر همه ورودی‌ها بالا باشند، هر دو ترانزیستور کانال p خاموش و هر دو ترانزیستور کانال n روشن خواهند بود. خروجی دارای امپدانس کمی نسبت به زمین بوده و سطح پایین را تولید می‌نماید. اگر هر یک از

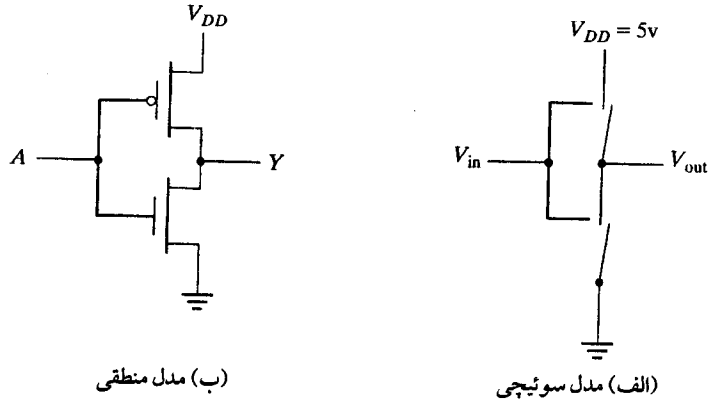


شکل ۲۲-۱۰ . مدارهای منطقی CMOS

ورودی‌ها پایین باشد، ترانزیستور کانال n مربوط به آن خاموش و ترانزیستور کانال p روشن می‌شود. خروجی به V_{DD} متصل و به سطح بالا می‌رود. گیت‌های NAND چند ورودی را می‌توان به ترتیب با موازی و سری کردن تعدادی مساوی از ترانزیستورهای نوع p و نوع n طبق شکل ۲۲-۱۰ (ب) ساخت.

یک گیت NOR دو ورودی از دو واحد نوع n موازی و دو واحد p سری، مطابق شکل ۲۲-۱۰ (پ) تشکیل می‌شود. وقتی همه ورودی‌ها پایین هستند، واحدهای کانال p روشن و هر دو واحد کانال n خاموش‌اند. خروجی به V_{DD} وصل شده و به سطح بالا می‌رود. اگر هر یک از ورودی‌ها بالا برود، ترانزیستور کانال p خاموش و ترانزیستور کانال n روشن می‌شود. این موجب می‌شود تا خروجی به زمین متصل و سطح پایین در آن تشکیل گردد.

ترانزیستورهای MOS را می‌توان به عنوان سوئیچ‌هایی که وصل می‌کنند و یا باز هستند، تصور کرد.



شکل ۲۳-۱۰. وارونگر CMOS

به عنوان مثال وارونگر MOS را می‌توان طبق شکل ۲۳-۱۰ (الف) متشکل از دو سوئیچ تصور کرد. اعمال ولتاژ پایین سبب می‌شود تا سوئیچ بالایی (p) بسته شده و ولتاژ بالایی را به خروجی متصل نماید. اعمال ولتاژ بالا به ورودی موجب می‌گردد تا سوئیچ پایینی (n) بسته شده و خروجی را به زمین وصل کند. بنابراین خروجی V_{out} متمم ورودی V_{in} است. کاربردهای تجاری اغلب از سمبل‌های گرافیکی دیگر برای ترانزیستور استفاده می‌کنند تا بر رفتار سوئیچی آن تأکید شود. از فلشی که جهت جریان را نشان می‌دهد صرف‌نظر شده است. در عوض، ورودی گیت ترانزیستور کانال p با یک حباب وارونگر در پایانه گیت رسم شده است تا مشخص شود که با ولتاژ فعال می‌شود. مدار وارونگر با این سمبل در شکل ۲۳-۱۰ (ب) دوباره رسم شده است. یک 0 در ورودی موجب می‌شود تا ترانزیستور فوقانی هدایت کرده و خروجی به منطق 1 برود. یک 1 در ورودی ترانزیستور تحتانی را روشن کرده و خروجی به 0 منطقی وصل می‌شود.

مشخصه‌های COMS

وقتی که یک مدار منطقی CMOS در حالت استاتیک یا سکون است توان مصرفی‌اش خیلی کم است. دلیل این است که وقتی حالت مدار تغییر نکند، همیشه یک ترانزیستور در مسیر خاموش است. در نتیجه یک گیت CMOS نوعی دارای توان مصرفی استاتیکی حدود 0.01mW است. با این وجود وقتی که مدار با فرکانس 1MHz تغییر حالت دهد توان مصرفی به حدود 1mW و در 10MHz به 5mW می‌رسد.

مدار منطقی CMOS با یک منبع بین 18V-3 و معمولاً 5V تغذیه می‌شود. راه‌اندازی CMOS با منبع تغذیه بالاتر کاهش زمان تأخیر انتشار و تصحیح حد پارازیت را به دنبال دارد ولی توان مصرفی افزایش می‌یابد. زمان تأخیر در انتشار برای $V_{DD} = 5V$ بین 5 تا 20ns است که این محدوده به نوع CMOS به کار رفته وابسته است. حد پارازیت معمولاً حدود 40 درصد ولتاژ منبع تغذیه می‌باشد. گنجایش خروجی گیت‌های CMOS وقتی در فرکانس 1MHz کار کند حدود 30 می‌باشد. گنجایش

خروجی با افزایش فرکانس کار، کاهش می‌یابد.

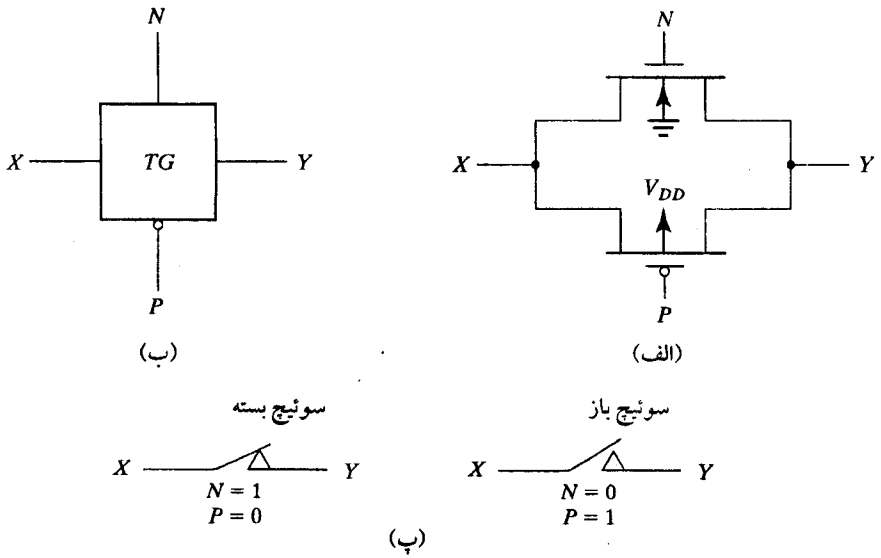
چندین سری از خانواده دیجیتال CMOS وجود دارد و سری 74C با وسایل TTL از نظر پایه و کار سازگار است و دارای شماره یکسانی نیز هستند. مثلاً آی‌سی CMOS نوع 74C04 دارای شش وارونگر با آرایش پایه مشابه نوع TTL، 7404 می‌باشد. سری CMOS سرعت بالا 74HC، اصلاحی بر سری 74C است که در آن سرعت سوئیچینگ 10 برابر شده است. سری 74HCT از نظر الکتریکی سازگار با آی‌سی‌های TTL است. این بدان معنی است که این نوع مدار قابل اتصال به ورودی‌ها و خروجی‌های آی‌سی‌های TTL می‌باشد بدون آن که به مدارهای واسط اضافی نیاز باشد. نوع جدیدتر CMOS، نوع سریع 74VHC و سازگار با آن، 74VHCT می‌باشد.

فرآیند ساخت CMOS ساده‌تر از TTL بوده و چگالی بسته‌بندی بیشتری را هم داراست، یعنی مدارهای بیشتری در سطح مفروضی از سیلیکان قابل پیاده‌سازی بوده و بنابراین فاکتور قیمت بر تابع آن کاهش می‌یابد. این خاصیت، همراه با توان مصرفی کم، حد پارازیت خوب و زمان تأخیر انتشار منطقی آن، CMOS را به یک خانواده منطقی مورد توجه تبدیل نموده است.

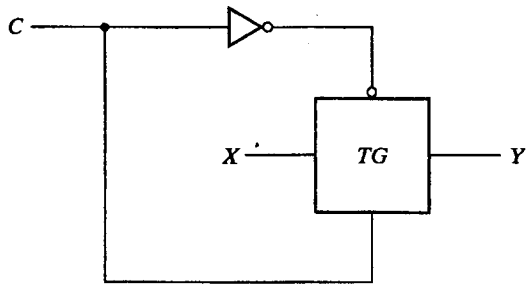
۱۰-۹ مدارهای گیت انتقال CMOS

نوع خاصی از مدارهای CMOS که در دیگر خانواده‌های منطقی موجود نیست، گیت انتقال می‌باشد. این مدار اساساً یک سوئیچ الکترونیک است که به وسیله یک سطح ورودی منطقی کنترل می‌گردد. این مدار برای ساده کردن ساخت انواع قطعات دیجیتال که با تکنولوژی CMOS ساخته می‌شوند به کار می‌رود.

شکل ۲۴-۱۰ (الف) مدار پایه گیت انتقال را نشان می‌دهد. این مدار شامل یک ترانزیستور کانال n و یک ترانزیستور MOS کانال p است که با یکدیگر موازیند.



شکل ۲۴-۱۰ . گیت انتقال (TG)

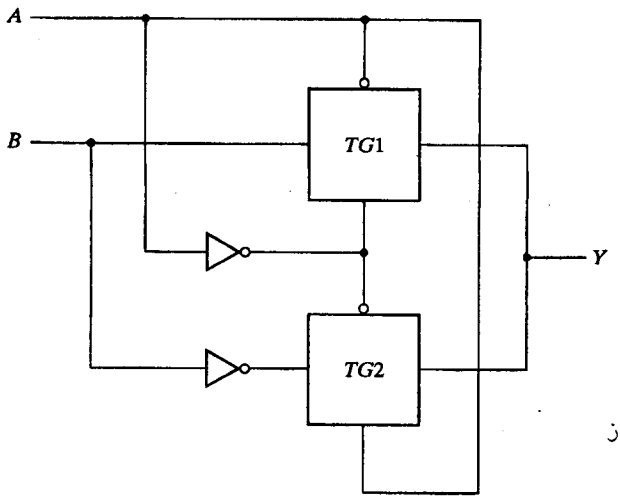


شکل ۲۵-۱۰ . سویچ دو طرفه

بدنه کانال n به زمین و بدنه کانال p به V_{DD} وصل است. وقتی گیت N در V_{DD} و گیت p به زمین متصل است، هر دو ترانزیستور هدایت کرده و یک مسیر بسته بین ورودی X و خروجی Y به وجود می‌آید. وقتی N به زمین است و گیت p به V_{DD} وصل می‌باشد، هر دو ترانزیستور خاموش و بین X و Y یک مدار باز وجود خواهد داشت. شکل ۲۴-۱۰ (ب) نمودار بلوکی گیت انتقال را نشان می‌دهد. توجه کنید که پایانه گیت کانال p با علامت نفی علامت‌گذاری شده است. شکل ۲۴-۱۰ (ب) رفتار سوئیچ را برحسب تخصیص منطق مثبت نشان می‌دهد که در آن V_{DD} معادل با منطق 1 و زمین نیز منطق 0 است.

گیت انتقال معمولاً به یک وارونگر وصل می‌گردد، شکل ۲۵-۱۰. این نوع آرایش را سوئیچ دو طرفه می‌نامند. ورودی کنترل C مستقیماً به گیت کانال n و معکوس آن به گیت کانال p وصل است. وقتی $C = 1$ است، سوئیچ بسته شده و مسیری بین X و Y به وجود می‌آورد. اگر $C = 0$ باشد، سوئیچ باز شده و مسیر بین X و Y را قطع می‌کند. با گیت انتقال می‌توان انواع مدارها را ساخت. برای نمایش کارآیی آن به عنوان یک قطعه در خانواده CMOS ، مدار سه مثال را نشان می‌دهیم.

گیت OR انحصاری (XOR) را می‌توان با گیت انتقال و دو وارونگر ساخت، شکل ۲۶-۱۰ .

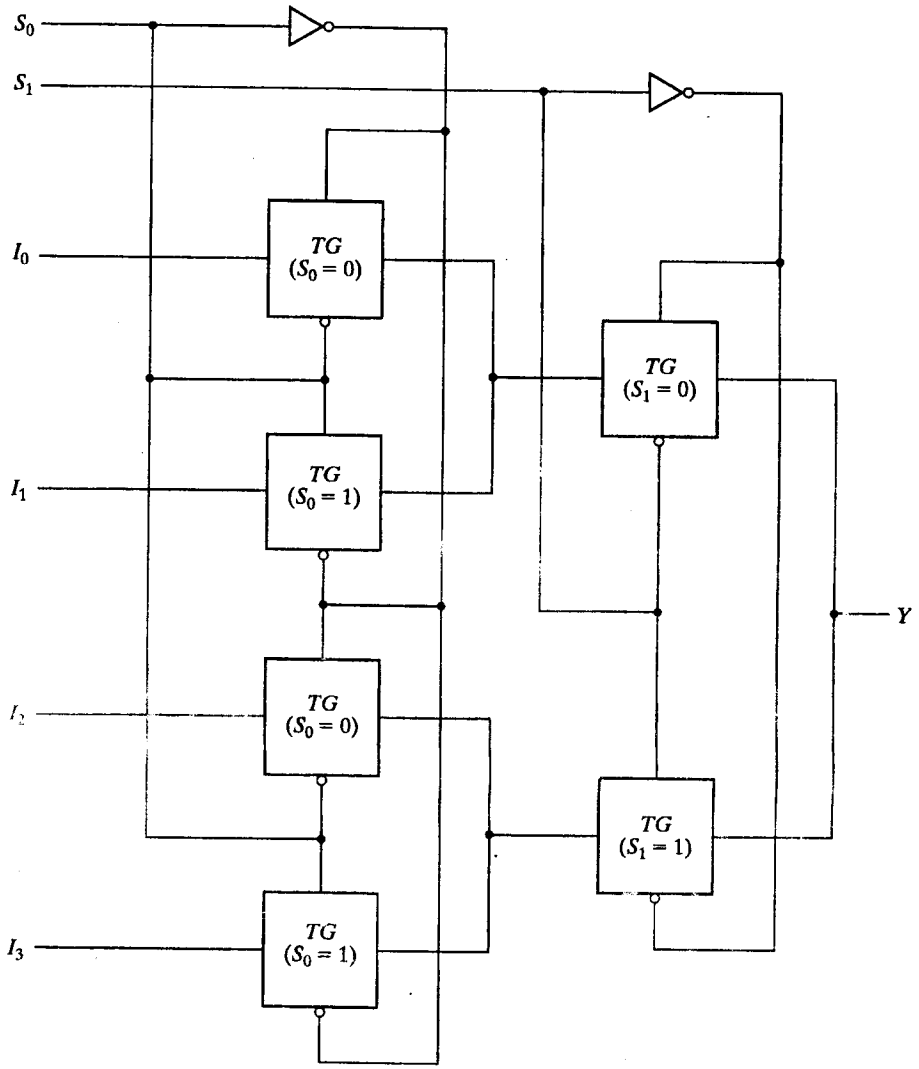


A	B	TG1	TG2	Y
0	0	بسته	باز	0
0	1	بسته	باز	1
1	0	باز	بسته	1
1	1	باز	بسته	0

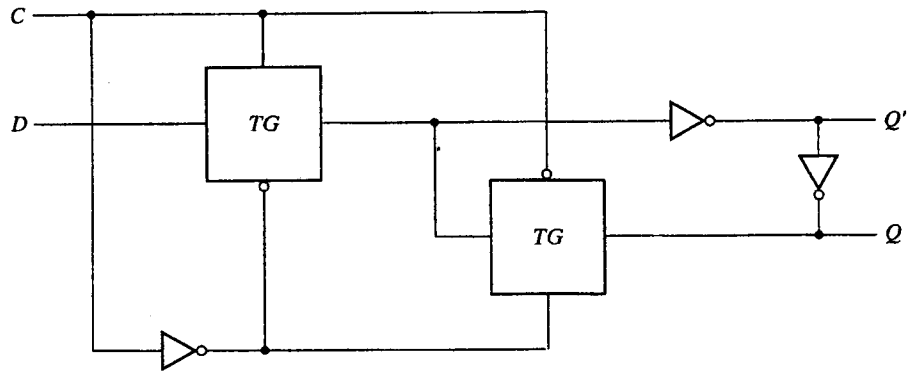
شکل ۲۶-۱۰ . XOR ساخته شده از گیت‌های انتقال

ورودی A مسیرها را در گیت‌های انتقال کنترل می‌کند و ورودی B از طریق گیت‌ها به خروجی Y وصل می‌باشد. وقتی ورودی A برابر 0 است، گیت انتقال $TG1$ بسته و خروجی Y برابر با ورودی B می‌گردد. هنگامی که ورودی A در 1 است، گیت $TG2$ بسته و خروجی برابر با متمم ورودی B خواهد بود. این حالات جدول درستی XOR را مطابق با شکل ۲۶-۱۰ نتیجه می‌دهند.

مدار دیگری که با گیت انتقال قابل ساخت است، مولتی پلکسر می‌باشد. یک مولتی پلکسر 4 به 1 با گیت‌های انتقال در شکل ۲۷-۱۰ ملاحظه می‌شود. مدار TG، به هنگام 1 بودن کنترل‌های عمودی در ورودی‌های بدون حباب، و 0 در ورودی‌های حباب‌دار، یک مسیر انتقال بین ورودی‌های افقی و خروجی ایجاد می‌نماید. با تغییر قطبیت در ورودی‌های کنترل مسیر قطع و مدار مانند یک سوئیچ باز



شکل ۲۷-۱۰ . مولتی پلکسر با گیت‌های انتقال



شکل ۲۸-۱۰ . لچ D گیت شده با گیت‌های انتقال

عمل خواهد کرد. دو ورودی انتخاب S_1 و S_0، مسیر انتقال را در مدار TG کنترل می‌نمایند. در داخل هر مربع شرط بسته بودن سوئیچ گیت انتقال نوشته شده است. بنابراین اگر $S_0 = 0$ و $S_1 = 0$ باشد، مسیر بسته‌ای از ورودی I_0 به خروجی Y که از طریق دو گیت TG که با $S_0 = 0$ و $S_1 = 0$ مشخص شده‌اند، ایجاد می‌شود. سه ورودی دیگر به وسیله سایر مدارهای TG از خروجی جدا شده‌اند.

فلیپ فلاپ D حساس به سطح که معمولاً به آن لچ D گیتی می‌گویند، به وسیله گیت‌های انتقال قابل ساخت است، شکل ۲۸-۱۰. ورودی C دو گیت انتقال را کنترل می‌نماید. وقتی $C = 1$ است، TG متصل به ورودی D مسیر بسته‌ای دارد و دیگری که به خروجی Q متصل است مسیر بازی ایجاد می‌نماید. این وضعیت دو مدار معادلی، از ورودی D از طریق دو وارونگر به خروجی به وجود می‌آورد. بنابراین مادامی که C فعال بماند خروجی، ورودی را دنبال خواهد کرد. اگر $C = 0$ باشد، اولین TG ورودی D را از مدار قطع و دومین TG مسیر بسته‌ای را بین دو وارونگر در خروجی ایجاد می‌کند. بنابراین مقدار موجود در ورودی D در لحظه انتقال C از 1 به 0 در خروجی Q حفظ خواهد شد.

می‌توان یک فلیپ فلاپ حاکم- تابع را هم با دو مدار طبق شکل ۲۸-۱۰ ساخت. اولین مدار حاکم و دومی تابع است. بنابراین یک فلیپ فلاپ D حاکم- تابع را می‌توان با چهار گیت انتقال و شش وارونگر ایجاد کرد.

۱۰-۱۰ مدل‌سازی سطح- سوئیچ با HDL

CMOS ، خانواده منطقی دیجیتال حاکم در مدارهای مجتمع است. بنا به تعریف، CMOS اتصال متمم یا مکمل یک ترانزیستور NMOS با یک ترانزیستور PMOS می‌باشد. ترانزیستورهای MOS را می‌توان به عنوان سوئیچ در نظر گرفت که در حالت بسته یا باز قرار دارند. با تعریف اتصالات بین سوئیچ‌های MOS ، طراح می‌تواند یک مدار ساخته شده با CMOS را تعریف نماید. این نوع تعریف را در Verilog HDL ، مدل‌سازی سطح- سوئیچ می‌نامند.

دو نوع سوئیچ MOS در Verilog HDL با دو کلمه کلیدی nmos و pmos تعریف شده‌اند. این سوئیچ‌ها با سه پایانه ترانزیستور طبق شکل ۲۰-۱۰ ذکر می‌شوند.

```
nmos (drain, source, gate);
pmos (drain, source, gate);
```

سوئیچ‌ها به عنوان Primitive ملاحظه می‌شوند بنابراین استفاده از instance name اختیاری است.

به هنگام طراحی مدارهای MOS، اتصال به یک منبع تغذیه (V_{DD}) و به زمین باید مشخص گردد. منبع تغذیه و زمین با کلمات کلیدی **Supply1** و **Supply0** تعریف می‌شوند. این کلمات مطابق عبارات زیر مشخص می‌گردند.

```
supply1 PWR;
supply0 GRD;
```

منابع نوع **Supply1** معادل V_{DD} بوده و دارای مقدار 1 منطقی‌اند. منابع نوع **Supply0** معادل زمین بوده و دارای مقدار 0 منطقی‌اند.

وارونگر CMOS شکل ۲۲-۱۰ (الف) در مثال ۱-۱۰ HDL توصیف شده است. ورودی، خروجی و دو منبع تغذیه ابتدا اعلان شده‌اند. مدول یک ترانزیستور PMOS و NMOS را ذکر می‌نماید. خروجی Y برای هر دو ترانزیستور مشترک است که به درین آنها متصل می‌باشد. ورودی نیز برای هر دو ترانزیستور در پایانه‌های گیت مشترک است. پایانه سورس ترانزیستور PMOS به PWR و پایانه سورس ترانزیستور NMOS به GRD وصل است.

دومین مدول در مثال ۲-۱۰ مدار CMOS NAND شکل ۲۲-۱۰ (ب) را توصیف می‌نماید. در این شکل دو ترانزیستور PMOS وجود دارد که به طور موازی به هم وصل و به PWR متصل شده‌اند. دو ترانزیستور NMOS دیگر به طور سری به هم وصلند و پایانه W1 برای آنها مشترک است. درین NMOS اول به خروجی و سورس NMOS دوم به GRD متصل می‌باشد.

گیت انتقال

گیت انتقال در Verilog HDL با کلمه کلیدی **cmos** ذکر می‌شود. این گیت دارای یک خروجی،

مثال ۱-۱۰، HDL

```
----------------------------------------
//CMOS inverter Fig. 10-22 (a)
module inverter (Y,A);
   input A;
   output Y;
   supply1  PWR;
   supply0  GRD;
   pmos (Y,PWR,A);       //(Drain,source,gate)
   nmos (Y,GRD,A);       //(Drain,source,gate)
endmodule
----------------------------------------
```

مثال ۲-۱۰ HDL

```
//CMOS 2-input NAND Fig. 10-22(b)
module NAND2 (Y,A,B);
   input A,B;
   output Y;
   supply1  PWR;
   supply0  GRD;
   wire W1;                 //terminal between two nmos
   pmos (Y,PWR,A);          //source connected to Vdd
   pmos (Y,PWR,B);          // parallel connection
   nmos (Y,W1,A);           // serial connection
   nmos (W1,GRD,B);         // source connected to ground
endmodule
```

ورودی و دو سیگنال کنترل مطابق شکل ۲۴-۱۰ است. به آن سوئیچ **cmos** می‌گویند. کد مربوط به آن مطابق زیر است.

```
cmos(output,input,ncontrol,pcontrol); //general description
cmos(Y,X,N,P);  //transmission gate of Fig. 10-24(b)
```

ncontrol و pcontrol متمم یکدیگرند. سوئیچ **cmos** به منابع تغذیه نیازی ندارد چون V_{DD} و زمین به بدنه ترانزیستورهای MOS وصلند. گیت‌های انتقال در ساخت مولتی پلکسرها و فلیپ فلاپ‌ها با مدارهای CMOS مورد توجه‌اند.

مثال ۳-۱۰ HDL توصیف یک مدار با سوئیچ‌های **cmos** را نشان می‌دهد. مدار XOR شکل ۲۶-۱۰ دارای دو گیت انتقال و دو وارونگر است. دو وارونگر با مدول وارونگر CMOS ذکر شده‌اند. دو سوئیچ **cmos** بدون instance name ذکر شده‌اند زیرا به عنوان primitive در نظر گرفته شده‌اند. یک مدول تست برای تست عملکرد مدار لحاظ شده است. با اعمال همه ترکیبات ممکن دو ورودی، نتیجه حاصل از شبیه‌ساز عملکرد مدار XOR تصدیق می‌شود. خروجی شبیه‌سازی به قرار زیر است:

$$\begin{array}{lll} A=0 & B=0 & Y=0 \\ A=0 & B=1 & Y=1 \\ A=1 & B=0 & Y=1 \\ A=1 & B=1 & Y=0 \end{array}$$

```
//XOR with CMOS switchs Fig. 10-25
module SXOR (A,B,Y);
   input A,B;
   output Y;
   wire Anot, Bnot;
//instantiate inverter
   inverter v1 (Anot,A);
   inverter v2 (Bnot,B);
//instantiate cmos switch
   cmos (Y,B,Anot,A);      //(output,input,ncontrol,pcontrol)
   cmos (Y,Bnot,A,Anot);
endmodule


//CMOS inverter Fig. 10-22(a)
module inverter (Y,A);
   input A;
   output Y;
   supply1  PWR;
   supply0  GRD;
   pmos (Y,PWR,A);         //(Drain,source,gate)
   nmos (Y,GRD,A);         //(Drain,source,gate)
endmodule


//Stimulus to test SXOR
module test_SXOR;
   reg A,B;
   wire Y;
//Instantiate SXOR
   SXOR X1 (A,B,Y);
//Apply truth table
   initial
     begin
          A=1'b0; B=1'b0;
       #5 A=1'b0; B=1'b1;
       #5 A=1'b1; B=1'b0;
       #5 A=1'b1; B=1'b1;
     end
//display results
   initial
    $monitor ("A =%b  B= %b  Y =%b",A,B,Y);
endmodule
```

مسائل

۱۰-۱ در زیر مشخصات گیت‌های NAND دو ورودی چهارتایی TTL 74S00 شوتکی ارائه شده است. گنجایش خروجی، توان تلف شده، تأخیر انتشار و حد پارازیت گیت NAND شوتکی را محاسبه کنید.

پارامتر	نام	مقدار
V_{CC}	منبع تغذیه	5 V
I_{CCH}	جریان تغذیه سطح بالا (چهار گیت)	10 mA
I_{CCL}	جریان تغذیه سطح پایین (چهار گیت)	20 mA
V_{OH}	ولتاژ خروجی سطح بالا (حداقل)	2.7 V
V_{OL}	ولتاژ خروجی سطح پایین (حداکثر)	0.5 V
V_{IH}	ولتاژ ورودی سطح بالا (حداقل)	2 V
V_{IL}	ولتاژ ورودی سطح پایین (حداکثر)	0.8 V
I_{OH}	جریان خروجی سطح بالا (حداقل)	1 mA
I_{OL}	جریان خروجی سطح پایین (حداکثر)	20 mA
I_{IH}	جریان ورودی سطح بالا (حداقل)	0.05 mA
I_{IL}	جریان ورودی سطح پایین (حداکثر)	2 mA
t_{PLH}	تاخیر پایین به بالا	3 ns
t_{PHL}	تاخیر بالا به پایین	3 ns

۱۰-۲ (الف) ولتاژ سطح بالای گیت RTL را برای گنجایش خروجی 5 محاسبه نمایید.

(ب) حداقل ولتاژ لازم برای راندن ترانزیستور RTL به اشباع $h_{FE} = 20$ را معین نمایید.

(پ) با نتایج (الف) و (ب) حد پارازیت گیت RTL را وقتی ورودی بالا و گنجایش خروجی 5 است معین کنید.

۱۰-۳ نشان دهید که ترانزیستور خروجی گیت DTL شکل ۹-۱۰، وقتی همه ورودی‌ها در سطح بالا باشند، به اشباع می‌رود. $h_{FE} = 20$ می‌باشد.

۱۰-۴ خروجی Y گیت DTL که در شکل ۹-۱۰ ملاحظه می‌شود را به N ورودی گیت‌های مشابه دیگر وصل کنید. فرض شود که ترانزیستور خروجی اشباع شده و جریان بیس آن 0.44mA است. $h_{FE} = 20$ فرض می‌گردد.

(الف) جریان مقاومت 2KΩ را محاسبه کنید.

(ب) جریان وارده از هر ورودی متصل به گیت را محاسبه نمایید.

(پ) جریان کلکتور کل را در ترانزیستور خروجی به صورت تابعی از N حساب کنید.

(ت) مقداری از N را بدست آورید که ترانزیستور را در اشباع نگهدارد.

(ث) گنجایش خروجی گیت چند است؟

۱۰-۵ اجازه بدهید تا ورودی‌های گیت TTL کلکتور باز شکل ۱۱-۱۰ در سطح بالا، 3V باشند.

(الف) ولتاژهای بیس، کلکتور و امیتر همه ترانزیستورها را معین نمایید.

(ب) حداقل h_{FE} لازم برای اشباع شدن ترانزیستور را معین کنید.

(پ) جریان بیس $Q3$ را حساب کنید.

(ت) فرض شود که h_{FE} حداقل برای $Q3$ برابر 6.18 است. حداکثر جریان قابل تحمل به وسیله کلکتور برای اطمینان از اشباع $Q3$ چقدر است؟

(ث) حداقل مقدار R_L برای اشباع $Q3$ چقدر است؟

۱۰-۶ (الف) با به‌کارگیری ترانزیستورهای خروجی واقعی دو گیت TTL کلکتور باز، نشان دهید. (با جدول درستی) که وقتی به یک مقاومت بیرونی و V_{CC} وصل شده است، اتصال سیمی آن تولید تابع AND می‌کند.

(ب) ثابت کنید که دو وارونگر TTL کلکتور باز وقتی به هم وصل شوند تولید تابع NOR می‌نمایند.

۱۰-۷ در بخش ۵-۱۰ بیان شد که خروجی‌های تاتم- پل برای ساخت منطق سیمی نباید به هم گره بخورند. برای دیدن علت این منع، دو نمونه از آن را به هم وصل کرده و اجازه بدهید خروجی یکی در سطح بالا و خروجی گیت دیگر در حالت پایین باشد. نشان دهید که جریان (در این حالت مجموع جریان‌های کلکتور و بیس ترانزیستور اشباع $Q4$ در شکل ۴-۱۰) حدود 32mA است. این مقدار را با جریان بار پیشنهادی در سطح بالای 0.4mA مقایسه نمایید.

۱۰-۸ برای شرایط زیر، ترانزیستورهای خاموش و آنهایی که در حال هدایتند را برای گیت TTL سه حالته شکل ۱۶-۱۰ (پ) مشخص نمایید. (برای $Q1$ و $Q6$ لازم است حالات را در پیوندهای بیس- امیتر و بیس- کلکتور به طور جداگانه حساب کنید.).

(الف) وقتی C پایین و A نیز در سطح پایین است.

(ب) وقتی C پایین و A در سطح بالاست.

(پ) وقتی C در سطح بالاست.

در هر حالت خروجی در چه وضعیتی است.

۱۰-۹ جریان امیتر I_E را در دو سر R_E در گیت ECL شکل ۱۷-۱۰، در شرایط زیر بدست آورید.

(الف) حداقل یک ورودی در سطح بالا، $-0.8V$ ، است.

(ب) همه ورودی‌ها در $-1.8V$ هستند.

(پ) اکنون فرض کنید که $I_C = I_E$ است. افت ولتاژ دو سر مقاومت کلکتور در هر حالت را محاسبه کنید و نشان دهید که مقدار آن حدود 1V است.

۱۰-۱۰ حد پارازیت گیت ECL را محاسبه کنید.

۱۰-۱۱ با استفاده از خروجی‌های NOR دو گیت ECL ، نشان دهید که وقتی با هم به یک مقاومت بیرونی و ولتاژ تغذیه منفی وصل شوند، اتصال سیمی تولید تابع OR می‌نماید.

۱۰-۱۲ ترانزیستور MOS دو طرفه است، یعنی جریان می‌تواند از سورس به درین یا از درین به سورس جاری شود. با استفاده از این خصوصیت، مداری بدست آورید که تابع بولی زیر را پیاده‌سازی کند.

$$Y = (AB + CD + AED + CEB)'$$

از شش ترانزیستور MOS استفاده نمایید.

۱۰-۱۳ (الف) مدار گیت NAND چهار ورودی را با ترانزیستورهای CMOS نشان دهید.

(ب) کار خود را برای یک گیت NOR چهار ورودی تکرار نمایید.

۱۰-۱۴ یک مدار XNOR با دو وارونگر و دو گیت انتقال بسازید.

۱۰-۱۵ یک مولتی پلکسر 8 به 1 خط را با گیت‌های انتقال و وارونگر بسازید.

۱۰-۱۶ نمودار منطقی فلیپ فلاپ D حاکم- تابع را با استفاده از گیت‌های انتقال و وارونگر رسم نمایید.

۱۰-۱۷ یک برنامه تست که مدار NAND مثال ۱۰-۲ HDL را تست کند بنویسید. شبیه‌سازی باید صحت جدول درستی گیت را تحقیق نماید.

مراجع

1. TOCCI, R. J. and N. S. WIDMER. 2001. *Digital Systems Principles and Applications,* 8th ed. Upper Saddle River, NJ: Prentice Hall.

2. WESTE, N. E. and K. ESHRAGHIAN. 1993. *Principles of CMOS VLSI design: A System Perspective,* 2nd ed. Reading, MA: Addison-Wesley.

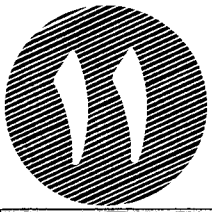
3. WAKERLY, J. F. 2000. *Digital Design: Principles and Practices*, 3rd ed. Upper Saddle River, NJ: Prentice Hall.

4. HODGES, D. A., and H. G. JACKSON. 1988. *Analysis and Design of Digital Integrated Circuits,* 2nd ed. New York: McGraw-Hill.

5. 1988. *The TTL Logic Data Book.* Dallas: Texas Instruments.

6. 1994. *CMOS Logic Data Book.* Dallas: Texas Intruments.

7. CILETTI, M. D. 1999. *Modeling, Synthesis, and Rapid Prototyping with Verilog HDL.* Upper Saddle River, NJ: Prentice Hall.



تمرینات آزمایشگاهی

۱۱-۰ مقدمه‌ای بر آزمایش‌ها

این فصل ۱۸ تمرین آزمایشگاهی را برای مدارهای دیجیتال و طراحی منطقی ارائه می‌دهد.این آزمایش‌ها همراه مطالب کتاب تجربه کافی را برای دانشجو فراهم می‌نمایند. مدارهای دیجیتال با استفاده از مدارهای مجتمع (IC) که به راحتی در آزمایشگاه قابل نصب روی بردبورد هستند ساخته می‌شوند. ترتیب ارائه آزمایش‌ها منطبق بر مطالب کتاب است. آخرین بخش چند قسمت تکمیلی همراه با پیشنهادات برای استفاده از Verilog HDL ، شبیه‌سازی و تست مدارهای دیجیتال را در بردارد.

بردبور منطقی مورد استفاده مناسب باید تجهیزات زیر را داشته باشد.

۱- لامپ‌های نشانگر LED

۲- سوئیچ‌های دگر وضع برای تهیه منطق 1 و 0

۳- پالس‌ساز همراه با کلید فشاری و مدار نوسان‌گیر برای تولید تک پالس‌ها

۴- مولد پالس ساعت با حداقل دو فرکانس- یک فرکانس پایین در حد یک پالس بر ثانیه برای مشاهده تغییرات در سیگنال‌های دیجیتال و فرکانس بالاتر برای مشاهده امواج در یک اسیلوسکوپ

۵- منبع تغذیه 5V

۶- سوکت ردیفی برای نصب آی‌سی‌ها

۷- سیم‌های محکم و یک جفت سیم‌چین برای بریدن سیم‌ها

وسایل آموزشی مدارهای دیجیتال که تجهیزات لازم را دارند از چندین سازنده قابل خریداری است. یک سیستم آموزشی دیجیتال حاوی لامپ‌های LED ، کلیدهای دگر وضع، پالس‌سازها، یک ساعت متغیر، منبع تغذیه و سوکت نواری آی‌سی است. بعضی از آزمایش‌ها ممکن است کلیدها، لامپ‌ها یا سوکت‌های اضافی لازم داشته باشند. همچنین ممکن است بردبوردها همراه با سوکت‌های بدون لحیم و

کلیدهای نصبی و لامپ‌های اضافی هم مورد نیاز باشند.

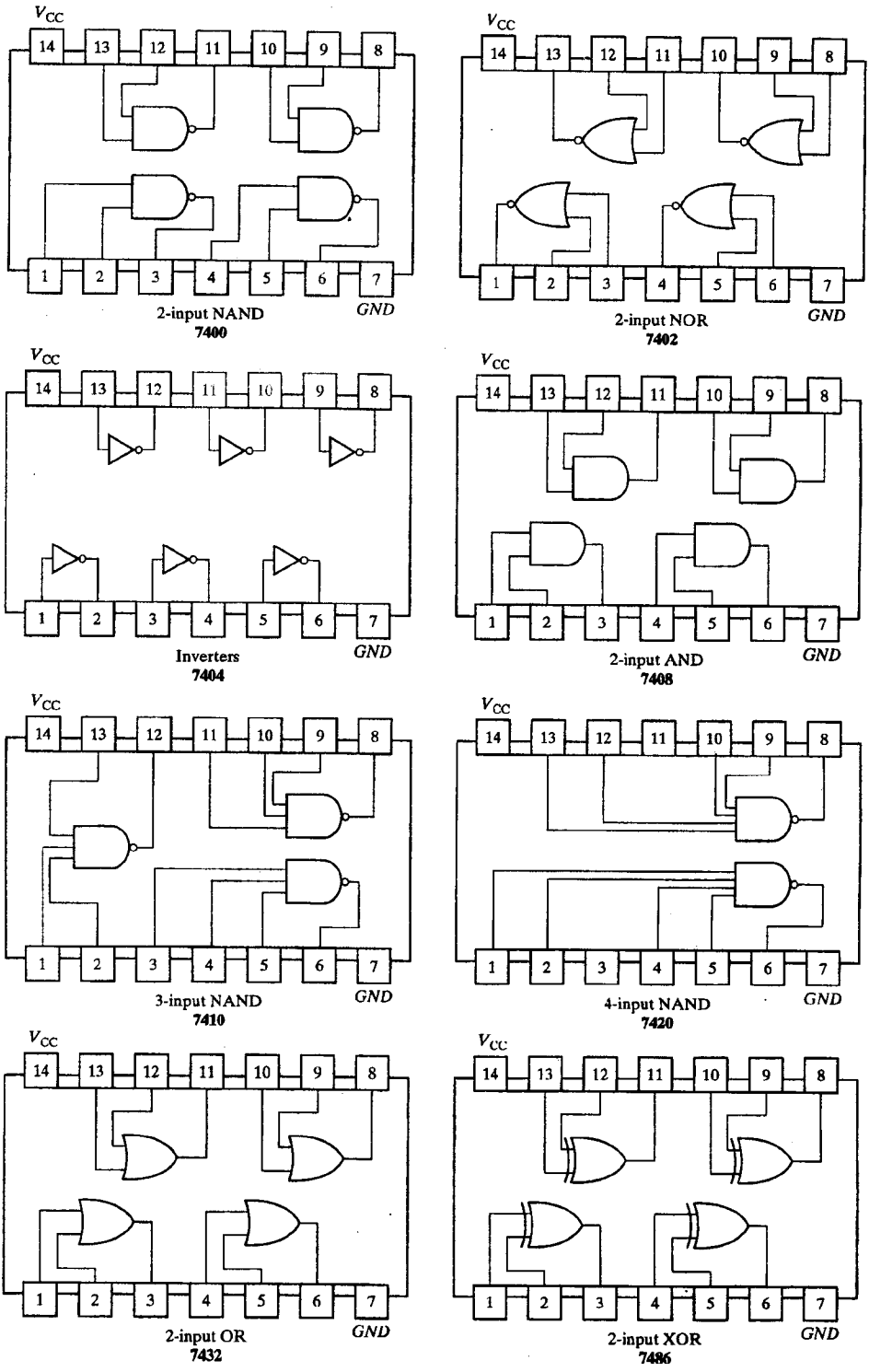
امکانات اضافی دیگر لازم عبارتند از اسیلوسکوپ دو کاناله (برای آزمایش‌های 1، 2، 8 و 15)، یک پروب منطقی برای عیب‌یابی و تعدادی IC. IC های لازم عبارتند از TTL سری 7400 یا CMOS.

مدارهای مجتمع مورد استفاده در آزمایش‌ها را به صورت مدارهای مجتمع با فشردگی کم (SSI) یا مدارهای مجتمع با فشردگی متوسط (MSI) می‌توان دسته‌بندی کرد. مدارهای SSI حاوی گیت‌های جدا از هم و مدارهای MSI توابع دیجیتال خاصی را تولید می‌نمایند. هشت آی‌سی گیت SSI به کار رفته در آزمایشها در شکل ۱-۱۱ ملاحظه می‌شود. آنها شامل گیت‌های NAND، NOR، AND، OR و XOR دو ورودی، NOT و گیت NAND چهار ورودی‌اند. نام‌گذاری پایه‌ها برای گیت‌ها در نمودار مشخص شده است. پایه‌ها از 1 تا 14 شماره‌گذاری شده‌اند. پایه 14 با V_{CC} و پایه 7 با GND (زمین) علامت‌گذاری شده است. این پایه‌ها مربوط به تغذیه‌اند که باید برای دستیابی به یک عملکرد صحیح به منبع تغذیه 5V وصل شوند. هر آی‌سی با شماره‌ای مشخص شناسایی می‌شود؛ مثلاً، گیت‌های NAND دو ورودی در آی‌سی شماره 7400 یافت می‌شوند.

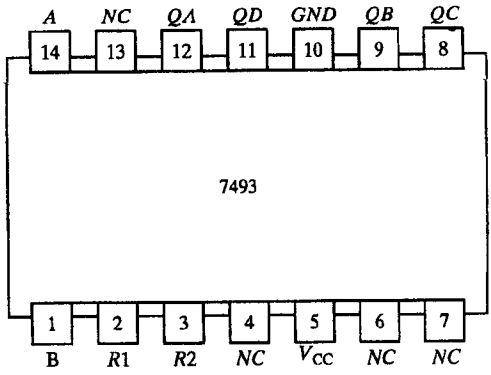
توضیحات بیشتر برای مدارهای MSI را می‌توان در کتب اطلاعات سازندگان یافت. بهترین راه کسب تجربه در مورد مدارهای MSI تجاری مطالعه توصیف آنها در این گونه منابع است که اطلاعات کاملی را در مورد مشخصه‌های درونی، بیرونی و الکتریکی آنها در اختیار می‌گذارند. سازندگان مختلف، کتابچه‌های اطلاعات متعدد را برای سری 7400 چاپ کرده‌اند. مدارهای MSI مورد نیاز آزمایشات به هنگام کاربردشان توصیف خواهند شد. عملکرد مدار با مراجعه به مدارات مشابه در فصول قبل توضیح داده می‌شود. اطلاعات داده شده درباره مدارهای MSI در این فصل برای اجرای صحیح آزمایشات کفایت می‌کنند. با این وجود مراجعه به برگه‌های اطلاعاتی همواره مفید خواهد بود، زیرا توصیف بیشتری برای مدارها در اختیار می‌گذارند.

اکنون به نمایش ارائه مدارهای MSI می‌پردازیم. این کار برای شمارنده موج گونه 7493 با مثال خاصی صورت گرفته است. این IC در آزمایش 1 و آزمایش‌های بعدی برای تولید رشته اعداد دودویی هنگام تحقیق صحت عملکرد مدارهای ترکیبی مورد استفاده می‌گیرد.

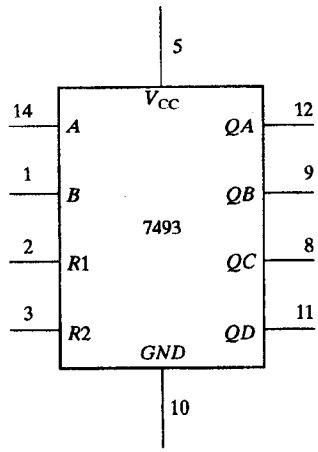
اطلاعات حاصل از برگه‌ها و کتب اطلاعات برای 7493 در شکل ۲-۱۱ (الف) و (ب) نشان داده شده است. بخش (الف) نمودار مدار منطقی داخلی و اتصالات آن به پایه‌های خارجی را نشان می‌دهد. به همه ورودی‌ها حرف سمبلیکی نسبت داده شده و نیز به پایه‌ها شماره تخصیص یافته است. بخش (ب) نمای فیزیکی IC همراه با نام سیگنال‌های مربوط به 14 پایه شمارنده را نشان می‌دهد. پایه‌هایی که به کار نرفته‌اند با NC علامت‌گذاری شده‌اند. آی‌سی بر روی سوکتی سوار شده و سیم‌ها از طریق پایه‌های سوکت به پایه‌های آن مرتبط شده‌اند. به هنگام رسم نمودار در این فصل، IC را به فرم نمودار بلوکی شکل ۲-۱۱ (پ) نشان خواهیم داد. شماره IC یعنی 7493 در داخل بلوک نوشته شده است. تمام پایه‌های ورودی در سمت چپ و پایه‌های خروجی در سمت راست بلوک نوشته شده‌اند. سمبل‌های حرفی سیگنال مانند A، $R1$ و QA هم در داخل بلوک مشخص شده‌اند، و پایه‌های مربوطه مانند 14، 2، 12 در امتداد



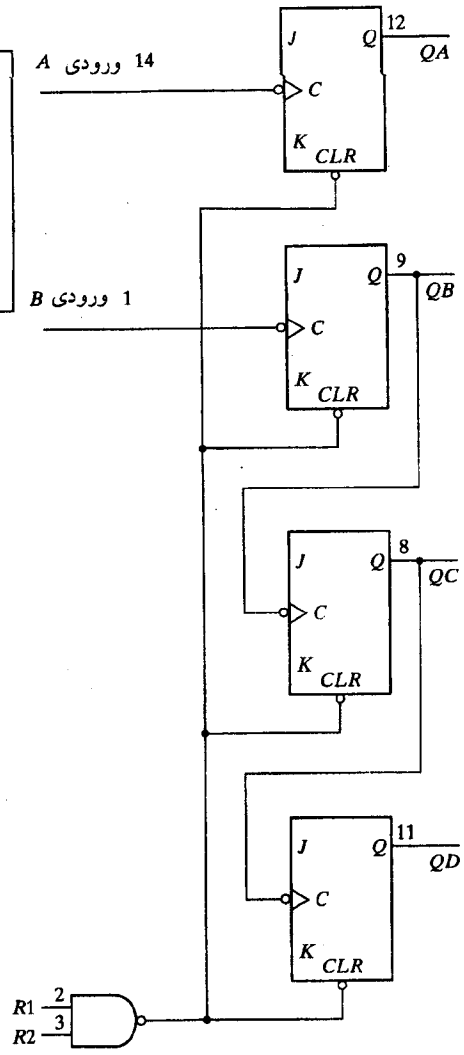
شکل ۱-۱۱ . گیت‌های دیجیتال در بسته‌های IC با شماره مشخصه و شماره پایه‌ها



(ب) نمای فیزیکی (NC : یعنی قطع)



(پ) نمودار بلوکی



(الف) نمودار داخلی مدار

شکل ۲-۱۱ . آی سی شمارنده موج‌گونه 7493

خطوط بیرونی نوشته شده‌اند. V_{CC} و GND پایانه‌های تغذیه بوده و به پایه‌های 5 و 10 وصل می‌گردند. برای جای دادن تمام پایانه‌های ورودی و خروجی ممکن است سایز بلوک تغییر نماید. ورودی‌ها یا خروجی هم گاهی جهت سهولت در بالا و یا در پایین بلوک قرار داده می‌شوند.

طرز کار مدار مشابه شمارنده موج‌گونه شکل ۸-۶ (الف) همراه با پاک کننده غیرهمزمان CLR برای هر فلیپ فلاپ است. وقتی ورودی‌های $R1$ یا $R2$ یا هر دو در منطق 0 هستند، همه پاک کننده‌های غیرهمزمان در منطق 1 بوده و غیرفعال می‌باشند. برای پاک کردن هر چهار فلیپ فلاپ، باید خروجی

گیت NAND برابر 0 باشد. این کار با قرار دادن $R1$ و $R2$ در 1 منطقی انجام می‌گردد (حدود 5 ولت). توجه کنید که ورودی‌های J و K به جایی متصل نیستند. این از ویژگی‌های مدار TTL است که یک ورودی بدون اتصال با بیرون، دارای اثر یک سیگنال معادل منطقی 1 است. همچنین توجه کنید که خروجی QA از داخل به ورودی B متصل نیست.

آی‌سی 7493 می‌تواند با استفاده از ورودی B و فلیپ فلاپ‌های QB، QC و QD کار کند. و نیز با به کارگیری ورودی A، اگر خروجی QA به ورودی B وصل شده باشد، این مدار به عنوان یک شمارنده 4 بیت کار خواهد کرد. بنابراین، برای راه‌اندازی به صورت یک شمارنده چهار بیتی، لازم است یک اتصال بیرونی بین پایه 12 و 1 ایجاد شود. ورودی‌های بازنشانی، $R1$ و $R2$، باید در نقاط 2 و 3 به زمین وصل شوند. پایه‌های 5 و 10 باید به تغذیه 5V متصل گردند. پالس‌های ورودی باید به ورودی A در پایه 14 اعمال گردند، و بالاخره چهار خروجی فلیپ فلاپ QA، QB، QC و QD باید به ترتیب از پایه‌های 12، 9، 8 و 11 اخذ شوند، که QA کم‌ارزش‌ترین بیت می‌باشد.

شکل ۲-۱۱ (پ) روش نمایش گرافیکی همه مدارهای MSI را در این IC نشان می‌دهد. برای هر IC فقط یک نمودار بلوکی مشابه با این شکل به کار خواهد رفت. سمبل‌های حرفی برای ورودی‌ها و خروجی‌ها در نمودار بلوکی مطابق سمبل‌های به کار رفته در کتب اطلاعات آن است. با ارجاع به نمودارهای منطقی فصول قبل، عملکرد مدار تشریح خواهد شد. این عملکرد با جدول درستی یا یک جدول تابع مشخص خواهد شد.

دیگر سمبل‌های گرافیکی ممکن برای آی‌سی‌ها در فصل ۱۲ ارائه شده است. این سمبل‌های گرافیکی استاندارد به وسیله انستیتو مهندسان برق و الکترونیک IEEE 91-1984 مورد تصویب قرار گرفته است. سمبل‌های گرافیکی گیت‌های SSI شکل مستطیلی دارند، شکل ۱-۱۲. سمبل گرافیکی استاندارد آی‌سی 7493 را در شکل ۱۳-۱۲ ملاحظه خواهید کرد. این سمبل می‌تواند جایگزین سمبل به کار رفته در شکل ۲-۱۱ (پ) گردد. سمبل گرافیکی سایر آی‌سی‌های به کار رفته در آزمایشات در فصل ۱۲ ارائه شده‌اند. این سمبل‌ها در صورت لزوم برای ترسیم نمودارهای شماتیک مدارهای منطقی، به کار خواهند رفت.

جدول ۱-۱۱ آی‌سی‌های مورد لزوم برای آزمایشات را به همراه شماره شکل‌های مربوطه لیست کرده است. به علاوه، جدول، شماره شکل‌های فصل ۱۲ را نیز لیست نموده است که در آنها سمبل‌های استاندارد معادل نشان داده شده‌اند.

بقیه فصل شامل ۱۹ قسمت است. هجده بخش اول، هجده آزمایش سخت‌افزاری را نشان می‌دهد که به مدارهای مجتمع دیجیتال نیاز دارند. بخش ۱۹-۱۱ آزمایش‌های شبیه‌سازی HDL را شامل می‌شود که به یک شبیه‌ساز و کامپایلر Verilog HDL نیاز دارند.

۱-۱۱ اعداد دودویی و دهدهی

این آزمایش شمارش یک رشته اعداد دودویی و نمایش اعداد BCD را نشان می‌دهد. آزمایش مذکور به منظور آشنایی با بردبورد به کار رفته در آزمایشگاه و نیز آشنایی دانشجویان با اسیلوسکوپ

جدول ۱-۱۱ . مدارات مجتمع مورد نیاز برای آزمایشات

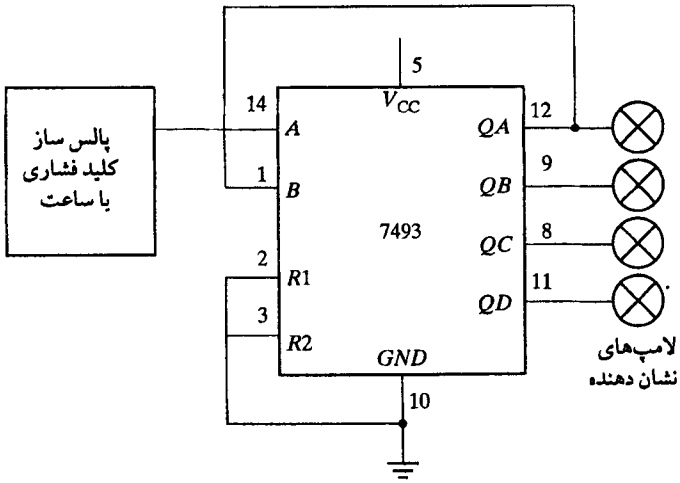
شمارهٔ آی‌سی	توضیح	سمبل‌های گرافیکی در فصل ۱۱	سمبل‌های گرافیکی در فصل ۱۲
	انواع گیت	شکل ۱-۱۱	شکل ۱-۱۲
7447	دیکدر BCD به هفت قسمتی	شکل ۸-۱۱	
7474	دو فلیپ فلاپ نوع D	شکل ۱۳-۱۱	شکل ۹-۱۲ (ب)
7476	دو فلیپ فلاپ JK	شکل ۱۲-۱۱	شکل ۹-۱۲ (الف)
7483	جمع کننده دودویی چهار بیت	شکل ۱۰-۱۱	شکل ۲-۱۲
7493	شمارنده موج گونه چهار بیت	شکل ۲-۱۱	شکل ۱۳-۱۲
74151	مولتی پلکسر 8x1	شکل ۹-۱۱	شکل ۷-۱۲ (الف)
74155	دیکدر 3x8	شکل ۷-۱۱	شکل ۶-۱۲
74157	چهار مولتی پلکسر 2x1	شکل ۱۷-۱۱	شکل ۷-۱۲ (ب)
74161	شمارنده چهار بیت سنکرون	شکل ۱۵-۱۱	شکل ۱۴-۱۲
74189	حافظه RAM، 16x4	شکل ۱۸-۱۱	شکل ۱۵-۱۲
74194	شیفت رجیستر چهار بیت	شکل ۱۹-۱۱	شکل ۱۲-۱۲
74195	شیفت رجیستر دو جهته چهار بیت	شکل ۱۶-۱۱	شکل ۱۱-۱۲
7730	نمایش هفت قسمتی LED	شکل ۸-۱۱	
72555	تایمر دقیق	شکل ۲۱-۱۱	

اشعه کاتودیک است. مطالب مرجع مورد نیاز در حین کار را می‌توانید از بخش ۲-۱ برای اعداد دودویی و بخش ۷-۱ برای اعداد BCD بدست آورید.

شمارش دودویی

آی‌سی 7493 مطابق شکل ۲-۱۱ از چهار فلیپ فلاپ تشکیل شده است. سلول‌ها می‌توانند برای شمارش دودویی یا BCD به هم متصل شوند. با سیم‌بندی بیرونی، مطابق شکل ۳-۱۱، آن را به عنوان شمارنده دودویی آماده نمایید. این کار با اتصال از پایه 12 (خروجی QA) به پایه 1 (ورودی B) انجام می‌شود. ورودی A در پایه 14 به پالس‌سازی وصل می‌شود تا تک پالس‌هایی تولید نماید. دو ورودی بازنشانی، $R1$ و $R2$، به زمین متصل می‌گردند. چهار خروجی به چهار لامپ مرتبط شده‌اند و در آن بیت کم‌ارزش‌تر از QA به سمت راست‌ترین لامپ متصل شده است. از اتصال تغذیه 5V و زمین به IC فراموش ننمایید. حین اجرای اتصالات، منبع تغذیه باید خاموش باشد.

اکنون منبع تغذیه را روشن کنید و چهار لامپ را ملاحظه نمایید. با هر پالس تولیدی به وسیله دکمه فشاری روی پالس‌ساز، عدد 4 بیتی خروجی یک واحد افزایش می‌یابد. شمارش به 15 رفته و سپس به 0 باز می‌گردد. ورودی شمارنده در پایه 14 را از پالس‌ساز جدا کنید و به یک مولد پالس ساعت ببندید تا



شکل ۳-۱۱ . شمارنده دودویی

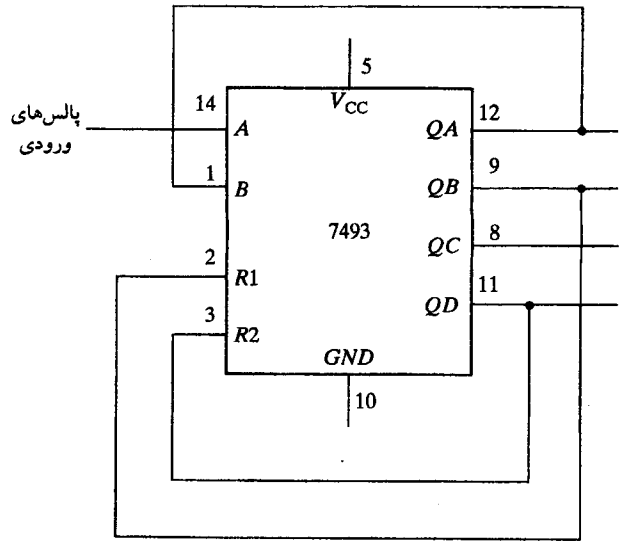
رشته‌ای از پالس‌ها با فرکانسی حدود یک هرتز به شمارنده منتقل گردد. این کار موجب شمارش خودکار خواهد شد. به یاد داشته باشید که شمارنده دودویی در آزمایشات بعدی برای تهیه سیگنال‌های دودویی ورودی در تست مدارهای ترکیبی به کار خواهد رفت.

نمایش اسیلوسکوپ

فرکانس ساعت را به 10KHz یا بالاتر افزایش دهید و خروجی آن را به یک اسیلوسکوپ دو کاناله متصل نمایید. خروجی ساعت را روی اسیلوسکوپ مشاهده کرده و آن را رسم کنید. خروجی QA را به یک کانال و خروجی مولد پالس ساعت را به کانال دوم وصل نمایید. هر بار که پالس ساعت از 1 به 0 برود QA متمم می‌گردد. همچنین توجه کنید که فرکانس ساعت در خروجی اولین فلیپ فلاپ نصف فرکانس ساعت در ورودی است. هر فلیپ فلاپ به نوبه خود، فرکانس دریافتی خود را نصف می‌نماید. بنابراین شمارنده چهار بیت، فرکانس دریافتی را در خروجی QD بر 16 تقسیم خواهد کرد. یک نمودار زمانبندی برای نمایش ارتباط پالس ساعت و چهار خروجی شمارنده تهیه کنید. دقت نمایید در نمودار حداقل 16 پالس ساعت رسم شود. روال پیشروی آزمایش با اسیلوسکوپ بدین شرح است. ابتدا پالس‌های ساعت QA را مشاهده و شکل موج آنها را ثبت نمایید. سپس QA و QB ملاحظه و ثبت شود و به دنبال آن، اعمال را برای QB با QC و QC با QD تکرار گردد. نتیجه نهایی، رابطه زمانی ساعت و چهار خروجی، در یک نمودار مرکب با حداقل 16 پالس ساعت خواهد بود.

شمارش BCD

نمایش BCD از اعداد دودویی 0000 تا 1001 برای نمایش ارقام دهدهی استفاده می‌کند. آی‌سی



شکل ۴-۱۱. شمارنده BCD

7493 با توجه به اتصالات شکل ۴-۱۱ قادر است به عنوان یک شمارنده BCD عمل کند. خروجی‌های QB و QD به دو ورودی بازنشانی $R1$ و $R2$ وصلند. وقتی هر دو ورودی $R1$ و $R2$ برابر 1 باشند، هر چهار سلول شمارنده، جدا از پالس‌های ورودی، به 0 پاک می‌شوند. شمارنده از 0 شروع به کار کرده و با هر پالس ورودی، 1 واحد افزایش می‌یابد تا به 1001 برسد. پالس بعدی خروجی را بـه 1010 تـغییر می‌دهد، و در آن QB و QD برابر 1 هستند. این خروجی لحظه نمی‌تواند دوامی داشته باشد، زیرا هر چهار سلول بلافاصله پاک شده و در نتیجه خروجی‌ها 0000 خواهند شد. بنابراین پس از عـدد 1001 خروجی به 0000 رفته و به این ترتیب یک شمارش BCD تولید شده است.

IC را برای یک شمارش BCD سیم‌بندی نمایید. ورودی‌ها را به یک مولد پالس و خروجی‌ها را به لامپ‌ها وصل کنید. ببینید آیا شمارش از 0000 تا 1001 انجام می‌شود؟

ورودی را از پالس‌ساز جدا کرده و آن را به مولد ساعت ببندید. پالس ساعت و چهار خـروجی را روی اسیلوسکوپ ملاحظه نمایید. یک نمودار زمانبندی دقیقی که نشان دهنده ارتباط زمانی بین پالس ساعت و چهار خروجی باشد بدست آورید. مطمئن شوید که حداقل ده پالس ساعت به صورت نمودار مرکب روی صفحه اسیلوسکوپ وجود داشته باشد.

الگوی خروجی

وقتی که شمارش پالس‌ها در یک شمارنده BCD ادامه یابد، شمارنده رشته 0000 تا 1001 را شمرده و به 0000 باز می‌گردد. این بدان معنی است که هر بیت در چهار خروجی الگوی ثابتی از 1ها و 0ها را تولید می‌نماید و پس از هر ده پالس عمل تکرار می‌گردد. الگوی شمارش از روی جدول اعداد 0000 تا

1001 قابل پیش‌بینی است. لیست نشان می‌دهد که خروجی QA، که کم‌ارزش‌ترین بیت است، الگوی یک در میانی از 0 تا 1 تولید می‌نماید. خروجی QD که باارزش‌ترین بیت می‌باشد هشت 0 و دو 1 تولید می‌کند. برای دو خروجی دیگر الگوی موجود را بدست آورده و سپس هر چهار الگو را روی اسیلوسکوپ چک نمایید. این عمل با یک اسیلوسکوپ دو کانال و اتصال پالس ساعت به یکی، و یکی از خروجی‌ها به کانال دوم میسر است. الگوی 1 ها و 0 ها برای هر خروجی با مشاهده سطوح خروجی و تغییرات آن در امتداد محور عمودی و در جایی که پالس ساعت از 1 به 0 تغییر سطح می‌دهد، بدست می‌آید.

سایر شمارش‌ها

آی‌سی 7493 می‌تواند برای شمارش از 0 تا هر عدد نهایی موردنظر سیم‌بندی شود. این کار با وصل یک یا دو خروجی به ورودی‌های بازنشانی، یعنی $R1$ و $R2$ صورت می‌گیرد. بنابراین اگر $R1$ به جای QB در شکل ۴-۱۱، به QA وصل شود، شمارش حاصل از 0000 تا 1000 خواهد بود، که یکی کمتر از 1001 ($QD = 1$ و $QA = 1$) است.

با استفاده از اطلاعات خود در مورد چگونگی تأثیر $R1$ و $R2$ روی شمارش نهایی، آی‌سی 7493 را برای شمارش از 0000 تا شماره‌های نهایی زیر سیم‌بندی نمایید.

(الف) 0101

(ب) 0111

(پ) 1011

هر مدار را بسته و رشته شمارش آن را با اعمال پالس‌هایی از پالس‌ساز روی لامپ‌ها تحقیق کنید. اگر شمارش اولیه از عدد نهایی بزرگتر باشد، اعمال پالس ورودی را تا پاک شدن خروجی‌ها به 0 ادامه دهید.

۲-۱۱ گیت‌های منطقی دیجیتال

در این آزمایش رفتار منطقی انواع گیت‌های آی‌سی را تحقیق خواهید کرد که عبارتند از:

7400 - چهار گیت NAND دو ورودی

7402 - چهار گیت NOR دو ورودی

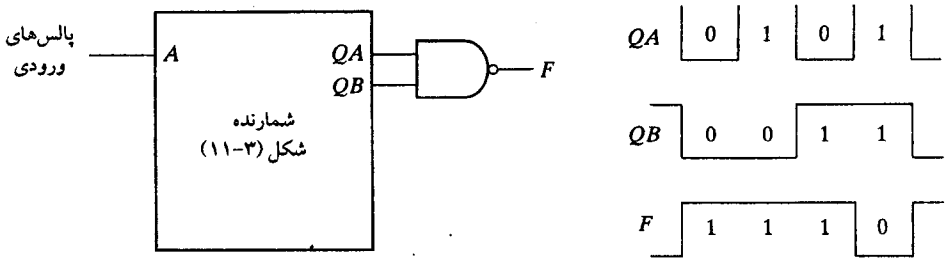
7404 - شش گیت NOT

7408 - چهار گیت AND دو ورودی

7432 - چهار گیت OR دو ورودی

7486 - چهار گیت XOR دو ورودی

تخصیص پایه‌ها به انواع گیت‌ها در شکل ۱-۱۱ دیده می‌شود. در برگه اطلاعات آی‌سی برای اعلان چهارتایی بودن گیت از کلمه "Quadruple" استفاده می‌گردد. گیت‌های منطقی دیجیتال و مشخصه‌های آنها در بخش ۸-۲ بحث شده‌اند. پیاده‌سازی با NAND در بخش ۶-۳ بحث شد.



شکل ۵-۱۱ . شکل موج‌های گیت NAND

جداول درستی

از هر آی‌سی لیست شده فوق یک گیت انتخاب کنید و جدول درستی آن را بدست آورید. جدول درستی با اتصال ورودی‌های گیت به کلیدها و خروجی به یک لامپ بدست می‌آید. نتایج حاصل را با جدول درستی شکل ۵-۲ مقایسه نمایید.

شکل موج‌ها

برای هر گیت لیست شده فوق، رابطه شکل موج ورودی- خروجی را بدست آورید. شکل موج‌ها را روی اسیلوسکوپ مشاهده کنید. از دو خروجی پایین رتبه شمارنده دودویی (شکل ۳-۱۱) برای تهیه ورودی به گیت استفاده نمایید. به عنوان مثال، مدار و امواج گیت NAND در شکل ۵-۱۱ تشریح شده است. اسیلوسکوپ این شکل موج را تکرار خواهد کرد، ولی شما باید بخش غیرتکراری آن را ثبت نمایید.

تأخیر انتشار

هر شش وارونگر داخل آی‌سی 7404 را به طور متوالی به هم ببندید. خروجی با ورودی یکسان خواهد بود و تنها اختلاف تأخیری است که سیگنال برای انتشار در شش وارونگر نیاز دارد. پالس‌های ساعت را به ورودی اولین وارونگر اعمال نمایید. با به کارگیری اسیلوسکوپ، زمان تأخیر ورودی اولین و خروجی ششمین وارونگر به هنگام پایین آمدن یا بالا رفتن را معین کنید. این عمل با یک اسیلوسکوپ دو کانال و اتصال پالس ساعت به یکی از کانال‌ها و خروجی ششمین وارونگر به کانال دیگر، صورت می‌گیرد. پیچ مربوط به مبنای زمان (time base) اسیلوسکوپ را روی پایین‌ترین بخش قرار دهید. زمان صعود و نزول پالس باید روی صفحه اسیلوسکوپ ظاهر گردد. برای بدست آوردن متوسط تأخیر انتشار در هر گیت، تأخیر کل را بر شش تقسیم نمایید.

گیت NAND یونیورسال

با استفاده از یک آی‌سی 7400 ، مداری با مشخصات زیر بسازید.

(الف) یک وارونگر، NOT
(ب) یک AND دو ورودی
(پ) یک OR دو ورودی
(ت) یک NOR دو ورودی
(ث) یک XOR دو ورودی (شکل ۳۲-۳)

در هر حال، صحت عملکرد مدار خود را با مقایسه با جدول درستی تحقیق می‌نماید.

مدار NAND

با به کارگیری یک آی‌سی 7400، با گیت‌های NAND تابع بولی زیر را پیاده‌سازی کنید.

$$F = AB + CD$$

۱- نمودار مدار را رسم نمایید.
۲- جدول درستی تابع F را به صورت تابعی از چهار ورودی بدست آورید.
۳- مدار را وصل و صحت جدول درستی را تحقیق کنید.
۴- الگوهای 0 و 1 را برای F ضمن تغییر ورودی‌های A، B، C و D از 0 تا 15 دودویی ثبت نمایید.
۵- هر چهار خروجی شمارنده دودویی شکل ۳-۱۱ را به چهار ورودی مدار NAND وصل کنید. پالس‌های ساعت ورودی را به یک کانال، خروجی F را به کانال دیگر اسیلوسکوپ دو کاناله وصل نمایید. الگوهای 1 و 0 را برای F پس از هر پالس ساعت ثبت کرده و با الگوهای ثبت شده مرحله ۴ مقایسه نمایید.

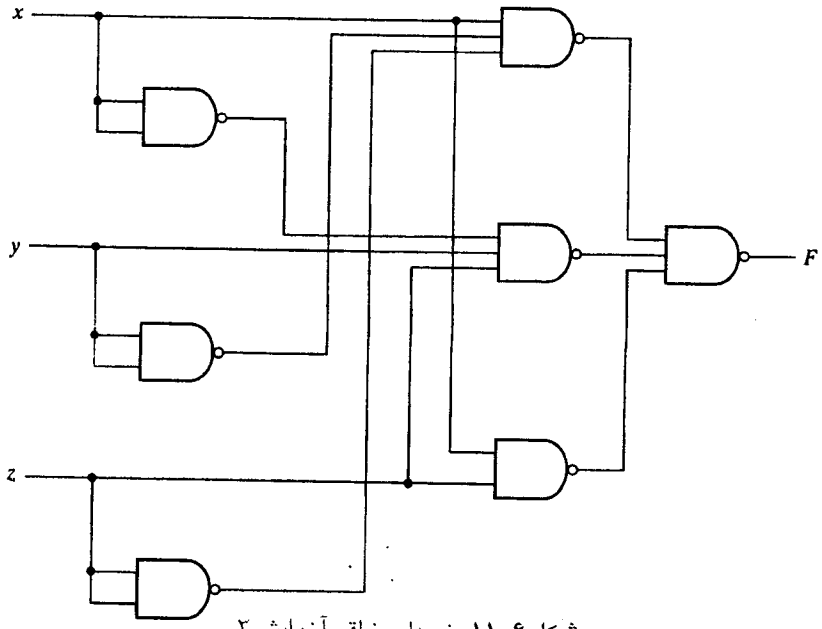
۳-۱۱ ساده‌سازی توابع بول

این آزمایش رابطه بین یک تابع بول و نمودار منطقی مربوطه‌اش را نشان می‌دهد. توابع بولی طبق بحث فصل ۳ با روش نقشه ساده می‌شوند. نمودارهای منطقی طبق بحث بخش ۶-۳ با گیت‌های NAND رسم خواهند شد.

آی‌سی‌های مورد استفاده برای نمودار منطقی باید از نوع گیت‌های NAND شکل ۱-۱۱ باشند:

گیت NAND دو ورودی 7400
گیت NOT (NAND یک ورودی) 7404
گیت NAND سه ورودی 7410
گیت NAND چهار ورودی 7420

اگر یکی از ورودی‌های گیت NAND مورد استفاده قرارنگیرد نباید آن را آزاد رها کرد بلکه باید به ورودی دیگری که استفاده شده وصل گردد. مثلاً اگر مدار وارونگری لازم دارد و دو گیت اضافی در آی‌سی 7400 موجود است، آنگاه هر دو ورودی گیت باید به هم وصل شوند تا برای وارونگر یک ورودی فراهم شود.



شکل ۶-۱۱ . نمودار منطقی آزمایش ۳

نمودار منطقی

این بخش از آزمایش با نمودار منطقی مفروضی، آغاز می‌شود و ما از آن برای کاهش تعداد گیت و احتمالاً تعداد آی‌سی استفاده خواهیم کرد. نمودار منطقی شکل ۶-۱۱ به دو آی‌سی 7400 و 7410 نیاز دارد. توجه کنید که برای داشتن وارونگرها، ورودی‌های x و y و z از سه گیت باقیمانده را استفاده نمایید. اگر وارونگرها از 7404 استفاده شوند مدار نیاز به سه آی‌سی خواهد داشت. همچنین توجه نمایید که در ترسیم مدار SSI، مشابه با آنچه در MSI دیدیم، گیت‌ها به صورت بلوک کشیده نمی‌شوند.

به تمام ورودی‌ها و خروجی‌های گیت‌ها، شماره پایه تخصیص داده و ورودی‌های x، y و z را به سه کلید و خروجی F را به یک لامپ وصل نمایید. با جدول درستی حاصل، مدار را تست کنید.

تابع بول مدار را بدست آورده و آن را با استفاده از روش نقشه ساده کنید. مدار ساده شده را بدون اتصال به مدار اصلی بسازید. هر دو مدار را با اعمال ورودی یکسان و خروجی‌های جداگانه تست نمایید. برای هر هشت ترکیب ورودی‌ها، نشان دهید که دو مدار خروجی‌های یکسانی دارند. این مطلب ثابت خواهد کرد که مدار ساده شده با مدار اصلی یکسان است.

توابع بول

با فرض داشتن توابع بول زیر، آنها را با روش نقشه ساده کنید.

$$F_1(A, B, C, D) = (0, 1, 4, 5, 8, 9, 10, 12, 13)$$

$$F_2(A, B, C, D) = (3, 5, 7, 8, 10, 11, 13, 15)$$

با توجه به چهار ورودی A، B، C و D و دو خروجی F_1 و F_2 یک نمودار منطقی مشترک بدست آورید. دو تابع را با حداقل تعداد آی‌سی‌های NAND پیاده‌سازی نمایید.

اگر جمله‌ای در هر دو تابع مشترک است، آن را دوباره تکرار نکنید. در صورت امکان از آی‌سی‌های موجود به جای وارونگر استفاده نمایید. مدار را وصل و طرز کار آن را تست کنید. جدول درستی F_1 و F_2 باید با مینترم‌های ذکر شده در تابع مطابقت داشته باشد.

متمم

تابع زیر را روی نقشه کارنو پیاده کنید.

$$F = A'D + BD + B'C + AB'D$$

1های نقشه را برای بدست آوردن فرم جمع حاصلضرب‌های تابع ساده شده با هم ترکیب کنید. سپس 0ها را با هم ترکیب کنید تا F' برحسب جمع حاصلضرب‌ها بدست آید. هر دو تابع F و F' را با NAND پیاده‌سازی کرده و دو مدار را به کلیدهای یکسان ولی لامپ‌های جدا وصل نمایید. جدول درستی هر مدار را در آزمایشگاه بدست آورده و نشان دهید که متمم یکدیگرند.

۴-۱۱ مدارهای ترکیبی

در این آزمایش، شما چهار مدار ترکیبی را طراحی و خواهید ساخت. دو مدار اول با گیت‌های NAND، سومی با XOR و چهارمی با دیکدر و گیت‌های NAND ساخته خواهند شد. مطالبی راجع به مولد توازن را می‌توانید در بخش ۸-۳ بیابید. پیاده‌سازی با دیکدر در بخش ۸-۴ ملاحظه شد.

مثال طراحی

مداری ترکیبی با چهار ورودی A، B، C و D و یک خروجی F طراحی نمایید. وقتی $A = 1$ باشد F برابر 1 است به شرطی که $B = 0$ باشد، یا وقتی $B = 1$ است و C یا D برابر 1 باشند. در غیر این صورت خروجی 0 است.

۱- جدول درستی مدار را بدست آورید.
۲- تابع خروجی را ساده کنید.
۳- نمودار منطقی مدار را با گیت‌های NAND و حداقل تعداد IC رسم نمایید.
۴- مدار را بسازید و آن را با توجه به شرایط فوق تست کنید.

منطق اکثریت

منطق اکثریت مداری است که اگر اکثریت ورودی‌ها 1 باشند، خروجی آن 1 است. در غیر این صورت خروجی 0 خواهد بود. با گیت‌های NAND یک مدار اکثریت سه ورودی بسازید و حداقل IC را بکار ببرید.

مولد توازن

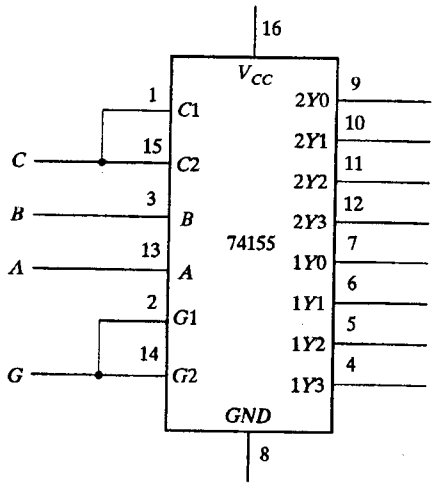
یک مدار مولد بیت توازن را با چهار بیت پیام طراحی کرده، ساخته و تست نمایید. با افزودن یک گیت XOR اضافی، مدار را طوری تکمیل نمایید که بتواند بیت توازن فرد را تولید کند.

پیاده‌سازی با دیکدر

یک مدار ترکیبی سه ورودی x، y و z و سه خروجی F_1، F_2 و F_3 دارد. تابع بولی ساده شده مدار به قرار زیر است.

$$F_1 = xz + x'y'z'$$
$$F_2 = x'y + xy'z'$$
$$F_3 = xy + x'y'z$$

مدار ترکیبی را با یک آی‌سی دیکدر 74155 و گیت‌های NAND بیرونی پیاده‌سازی نمایید.
نمودار بلوکی دیکدر و جدول درستی آن در شکل ۷-۱۱ ملاحظه می‌شود. 74155 می‌تواند به



جدول درستی

ورودی‌ها				خروجی‌ها							
G	C	B	A	2Y0	2Y1	2Y2	2Y3	1Y0	1Y1	1Y2	1Y3
1	X	X	X	1	1	1	1	1	1	1	1
0	0	0	0	0	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1
0	1	1	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	1	1	0

شکل ۷-۱۱ . آی سی 74155 که بصورت دیکدر 3×8 بسته شده است.

عنوان یک جفت دیکدر 4 × 2 یا یک دیکدر 8 × 3 در مدار قرار گیرد. زمانی که دیکدر 8 × 3 نیاز باشد، ورودی‌های C_1 و C_2 و ورودی‌های G_1 و G_2، طبق شکل نمودار منطقی، باید به هم وصل شوند. عملکرد مدار مشابه شکل ۱۸-۴ خواهد بود. G یک ورودی فعال‌ساز است و هنگام عمل صحیح باید به 0 وصل شود. هشت خروجی با سمبل‌های موجود در کتب اطلاعاتی علامت‌گذاری شده‌اند. 74155 از گیت‌های NAND استفاده می‌کند، بنابراین خروجی‌های انتخابی به 0 رفته و بقیه در 1 خواهند ماند. پیاده‌سازی با دیکدر در شکل ۲۱-۴ دیده می‌شود، با این تفاوت که وقتی 74155 به کار رود گیت‌های OR با NAND جایگزین می‌شوند.

۱۱-۵ مبدل‌های کد

تبدیل از یک کد دودویی به نوع دیگر در سیستم‌های دیجیتال بسیار رایج است. در این آزمایش شما سه مدار ترکیبی مبدل را طراحی خواهید کرد. تبدیل کد در بخش ۳-۴ مورد بحث قرار گرفت.

کدگری به دودویی

مداری ترکیبی با چهار ورودی و چهار خروجی بسازید تا یک کد چهار بیت گری (جدول ۶-۱) را به معادل دودویی چهار بیت تبدیل کند. مدار را با گیت‌های OR پیاده‌سازی نمایید (این کار را با آی‌سی 7486 انجام دهید). مدار را به چهار کلید و چهار چراغ وصل کرده صحت عملکرد را تحقیق نمایید.

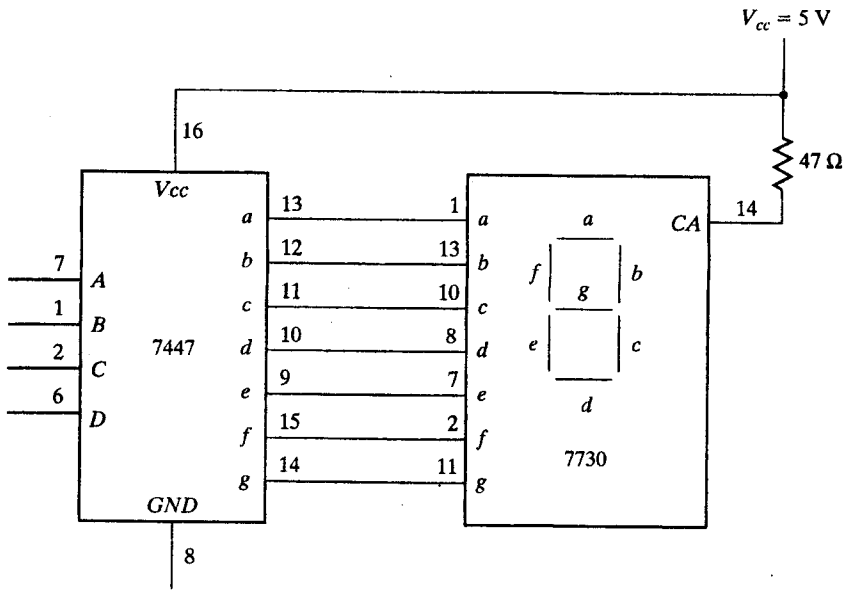
متمم 9

یک مدار ترکیبی با چهار خط ورودی و چهار خط خروجی طراحی کنید که در آن ورودی‌ها ارقام دهدهی را به BCD و خروجی‌ها متمم 9 ورودی را تولید نمایند. پنجمین خروجی، وجود خطا را در BCD ورودی نشان می‌دهد. این خروجی هنگامی 1 است که چهار ورودی یک ترکیب غیرمجاز از کد BCD را دارا باشند. هر یک از گیت‌های لیست ۱-۱۱ را به کار ببرید ولی تعداد آی‌سی‌ها حداقل باشد.

نمایشگر هفت قسمتی

یک نمایشگر هفت قسمتی یا هفت قطعه‌ای برای نمایش هر یک از ارقام دهدهی 0 تا 9 به کار می‌رود. معمولاً ارقام دیجیتال دهدهی به صورت BCD در دسترس است. یک دیکدر BCD به هفت قسمتی عدد دیجیتال را به BCD دریافت و کد هفت قسمتی مربوط به آن را تولید می‌نماید. این مطلب در مسئله ۹-۴ نشان داده شد.

شکل ۸-۱۱ اتصالات لازم بین دیکدر و نمایشگر را نشان می‌دهد. آی‌سی 7447 یک دیکدر/ درایور BCD به هفت قسمتی است. این آی‌سی دارای چهار ورودی برای یک رقم BCD است. ورودی D باارزش‌ترین و A کم‌ارزش‌ترین است. رقم چهار بیتی BCD به کد هفت قسمتی با خروجی‌های a تا g



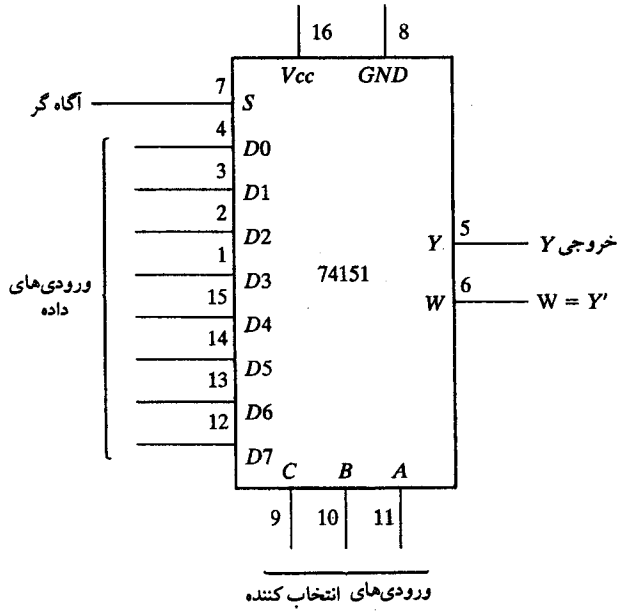
شکل ۸-۱۱ . دیکدر BCD به هفت قسمتی (7447) و نمایشگر هفت قسمتی (7730)

تبدیل می‌شود. خروجی‌های 7447 به ورودی‌های نمایشگر هفت قسمتی 7730 یا معادل آن اعمال می‌گردند. این آی‌سی دارای هفت قطعه LED در قسمت بالایی بسته می‌باشد. ورودی در پایه 14 آنود مشترک (CA) برای همه LED‌هاست. برای ایجاد جریان مناسب به قطعه انتخاب شده در LED یک مقاومت 47Ω به V_{CC} متصل است. سایر نمایشگرهای LED معادل ممکن است دارای پایانه‌های آنود اضافی بوده و به مقاومت‌های متفاوتی نیاز داشته باشند.

مدار شکل (۸-۱۱) را برپا کنید. چهار بیت BCD را از طریق چهار کلید به مدار اعمال نموده و نتیجه را برای ارقام 0 تا 9 روی نمایشگر ملاحظه نمایید. ورودی‌های 1010 تا 1111 در BCD دارای مفهوم نیستند. بسته به نوع دیکدر این مقادیر ممکن است به صورت الگوهای نامفهوم یا تاریک نشان داده شوند. برای شش حالت غیرمجاز الگوهای مزبور را ملاحظه و ثبت کنید.

۱۱-۶ طراحی با مولتی پلکسر

در این آزمایش شما یک مدار ترکیبی را طراحی نموده و آن را طبق آنچه در بخش ۱۰-۴ ملاحظه شد، با مولتی پلکسر پیاده‌سازی خواهید کرد. مولتی‌پلکسر مورد نظر، آی‌سی 74151 است که در شکل۹-۱۱ ملاحظه می‌شود. ساختار داخلی 74151 شبیه نمودار شکل ۲۵-۴ است، با این تفاوت که در عوض چهار ورودی، هشت ورودی وجود دارد. هشت ورودی با D_0 تا D_7 نام‌گذاری شده‌اند. سه خط انتخاب C ، B و A، ورودی خاصی را که قرار است مولتی پلکس شوند و به خروجی اعمال گردند، انتخاب می‌نمایند. یک



جدول تابع

آگاه گر	انتخاب			خروجی
S	C	B	A	Y
1	X	X	X	0
0	0	0	0	$D0$
0	0	0	1	$D1$
0	0	1	0	$D2$
0	0	1	1	$D3$
0	1	0	0	$D4$
0	1	0	1	$D5$
0	1	1	0	$D6$
0	1	1	1	$D7$

شکل ۹-۱۱ . آی سی مولتی پلکسر 1×8، نوع 74151

کنترل آگاه‌گر S نقش سیگنال فعال‌ساز را به عهده دارد. جدول تابع مقدار خروجی Y را براساس خطوط انتخاب مشخص می‌کند. خروجی W متمم Y است. به منظور صحت عمل، ورودی آگاه‌گر S باید به زمین متصل گردد.

مشخصات طراحی

یک شرکت کوچک دارای 10 سهم و هر سهم دارای یک رأی در جلسه سهامداران است. 10 سهم موجود بین چهار سهامدار به صورت زیر تقسیم شده است.

آقای W : یک سهم
آقای X : دو سهم
آقای Y : سه سهم
آقای Z : چهار سهم

هر یک از این افراد دارای کلیدی است که به هنگام رأی مثبت آن را می‌بندد و در رأی منفی آن را باز می‌کند.

لازم است مداری طراحی شود که تعداد کل سهام رأی مثبت را در هر رأی‌گیری نشان دهد. از یک نمایشگر هفت قسمتی و یک دیکدر، مثل شکل ۸-۱۱، برای نمایش عدد موردنظر استفاده نمایید. اگر در یک نوبت رأی‌گیری همه آرا منفی بود، نمایشگر باید تاریک باشد. (توجه کنید که عدد دودویی 15 به 7447 همه قطعات را تاریک می‌کند. اگر هر 10 سهام‌دار رأی مثبت دهند، نماشگر باید 0 را نشان دهد. در غیر این حالات، نمایشگر عددی دهدهی معادل با تعداد سهامی که رأی مثبت داده‌اند را نشان می‌دهد. از چهار مولتی پلکسر برای طراحی مدار ترکیبی استفاده نمایید، تا ورودی را از سوئیچ‌های سهامداران به ارقام BCD برای 7447 تبدیل کند. از 5V برای منطق 1 استفاده نکنید. خروجی یک وارونگری که ورودی‌اش به زمین وصل است را به کار بگیرید.

۱۱-۷ جمع و تفریق‌گر

در این آزمایش، شما انواع مدارهای جمع و تفریق‌گر را ساخته و تست خواهید کرد. سپس مدار تفریق‌گر برای مقایسه اندازه نسبی دو عدد به کار خواهد رفت. جمع‌کننده‌ها در بخش ۳-۴ مورد بحث قرار گرفتند. تفریق‌گر با متمم 2 در بخش ۶-۱ ملاحظه شد. یک جمع- تفریق‌گر در شکل ۱۳-۴ نشان داده شده، و مقایسه دو عدد هم در بخش ۷-۴ توضیح داده شد.

نیم جمع‌کننده

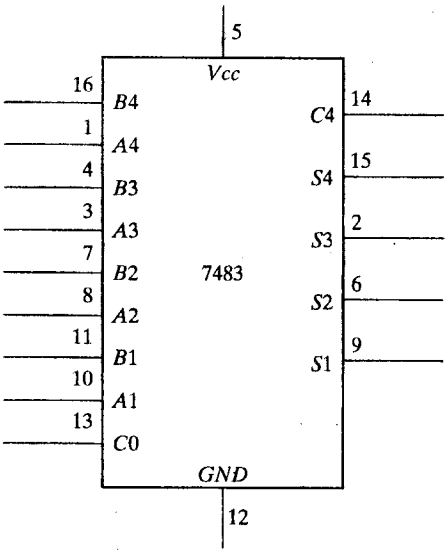
یک مدار نیم جمع کننده را با استفاده از یک گیت XOR و دو گیت NAND ، طراحی، ساخته و تست نمایید.

جمع کننده کامل

یک مدار جمع کننده کامل را با استفاده از دو آی‌سی 7486 و 7400 ، طراحی ، ساخته و تست نمایید.

جمع‌کننده موازی

آی‌سی 7483 یک جمع‌کننده موازی 4 بیت است. تخصیص پایه در شکل ۱۰-۱۱ نشان داده شده است. دو عدد دودویی 4 بیت $A1$ تا $A4$ و $B1$ تا $B4$ هستند. جمع چهار بیت از $S1$ تا $S4$ حاصل می‌شود. $C0$ نقلی ورودی و $C4$ نقلی خروجی است.



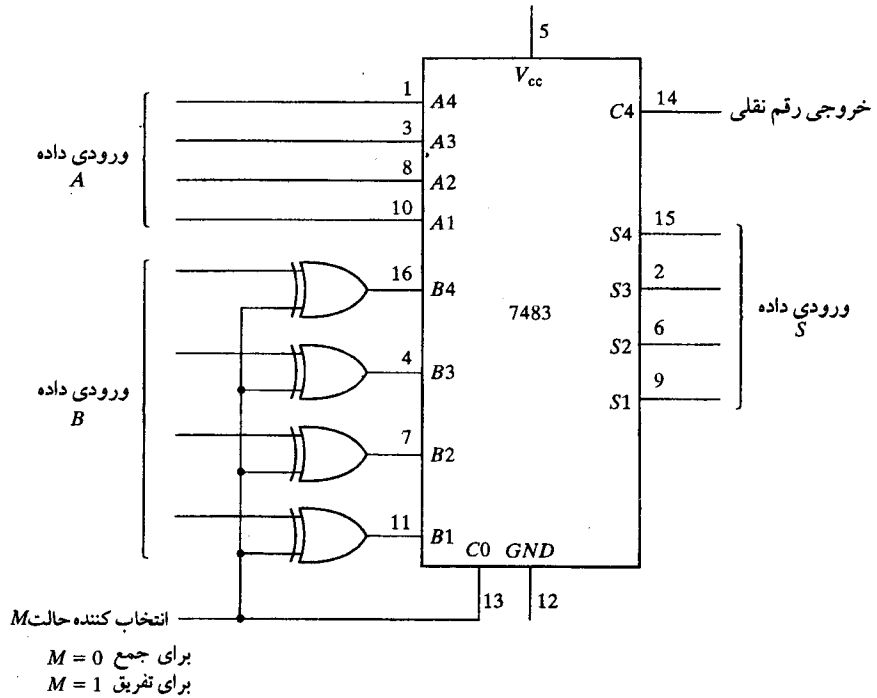
شکل ۱۰-۱۱ . آی سی جمع کننده دودویی چهار بیت 7483

با اتصال پایانه‌های منبع تغذیه و زمین، جمع‌کننده دودویی 4 بیت 7483 را تست نمایید. سپس چهار ورودی A را به عدد ثابت دودویی 1001 و ورودی‌های B و ورودی نقلی را به پنج کلید دگر وضع‌وصل کنید. پنج خروجی به پنج لامپ متصل‌اند. جمع چند عدد دیگر را هم اجرا نموده و ببینید آیا جمع خروجی و نقلی خروجی مقادیر صحیحی را ارائه می‌دهند یا خیر. نشان دهید که وقتی رقم نقلی ورودی 1 است، به خروجی یک واحد اضافه می‌گردد.

جمع- تفریق‌گر

تفریق دو عدد دودویی می‌تواند با استفاده از متمم 2 مفروق و جمع آن با مفروق منه بدست آید. متمم 2 را می‌توان با جمع متمم 1 با عدد 1 بدست آورد. برای اجرای $A-B$، ما چهار بیت از B را متمم نموده و آن را به چهار بیت A اضافه می‌کنیم، آنگاه یک واحد نیز از طریق رقم نقلی به مجموع آن دو می‌افزاییم. این کار در شکل ۱۱-۱۱ نشان داده شده است. چهار گیت XOR بیت‌های B را وقتی حالت $M = 1$ برقرار است متمم می‌نمایند. (زیرا $x \oplus 1 = x'$) و اگر $M = 0$ باشد بیت‌های B دست نخورده باقی می‌ماند (زیرا $x \oplus 0 = x$). بنابراین وقتی $M = 1$ است، ورودی نقلی $C0$ هم 1 بوده و خروجی حاصل جمع برابر است با A بعلاوه متمم 2 عدد B. وقتی $M = 0$ باشد، نقلی ورودی هم 0 و حاصل جمع تولید شده $A + B$ خواهد بود.

مدار جمع- تفریق‌گر را ببندید و آن را آزمایش کنید. چهار ورودی A را به عدد ثابت دودویی 1001 و B را به چهار سوئیچ وصل نمایید. اعمال زیر را اجرا و مقادیر خروجی جمع و نقلی خروجی $C4$ را ثبت کنید.



شکل ۱۱-۱۱. جمع کننده - تفریق‌گر چهار بیت

9 + 5	9 − 5
9 + 9	9 − 9
9 + 15	9 − 15

نشان دهید که ضمن عمل جمع، اگر حاصل جمع از 15 بیشتر شود، نقلی خروجی 1 خواهد شد. و نیز نشان دهید که وقتی $A \geq B$ باشد، عمل تفریق جواب صحیح $A - B$ را تولید می‌کند و رقم نقلی $C4 = 1$ است. ولی وقتی $A < B$ باشد، تفریق متمم 2 را برای $A - B$ را تولید کرده و نقلی خروجی هم 0 است.

مقایسه گر مقدار

مقایسه دو عدد عملی است که بزرگتر بودن، کوچکی و یا مساوی بودن یک عدد را نسبت به دیگری مشخص می‌کند. مقایسه دو عدد A و B ابتدا با تفریق $A - B$، طبق شکل ۱۱-۱۱ صورت می‌گیرد. اگر خروجی S برابر صفر باشد، می‌فهمیم که $A = B$ است. نقلی خروجی بیانگر نسبت مقادیر است: وقتی $C4 = 1$ باشد، داریم $A \geq B$. اگر $C4 = 0$ باشد $A < B$ است و اگر $C4 = 1$ و $S \neq 0$ باشد $A > B$ خواهد بود.

برای انجام مقایسه منطقی لازم است تا مدار شکل ۱۱-۱۱ اصلاح گردد. این عمل با مداری ترکیبی

که پنج ورودی $S1$ تا $S4$ و $C5$ و سه خروجی x و y و z را دارد صورت می‌گیرد به طوری که

اگر $A = B$ باشد	$x = 1$ است	$(S = 0000)$
اگر $A < B$ باشد	$y = 1$ است	$(C4 = 0)$
اگر $A > B$ باشد	$z = 1$ است	$(C4 = 1$ و $S \neq 0000)$

مدار ترکیبی می‌تواند با دو آی‌سی 7404 و 7408 پیاده‌سازی شود.

مدار مقایسه‌گر را بسازید و عمل آن را تست نمایید. حداقل دو مجموعه اعداد را برای A و B به کار برده و هر خروجی x و y و z را چک کنید.

۸-۱۱ فلیپ فلاپ‌ها

در این آزمایش، شما انواع لچ‌ها و فلیپ فلاپ‌ها را ساخته و طرز کار آنها را تست و بررسی خواهید کرد. ساختمان داخلی انواع لچ‌ها و فلیپ فلاپ‌ها در بخش‌های ۲-۵ و ۳-۵ ارائه شد.

لچ SR

با دو گیت NAND متقاطع یک لچ SR بسازید. دو ورودی را به سوئیچ‌ها و دو خروجی را به لامپ‌ها وصل نمایید. دو سوئیچ را در منطق 1 قرار دهید و سپس هر کدام را جداگانه 0 نمایید و بعد به 1 بازگردانید. جدول تابع مدار را بدست آورید.

لچ D

یک لچ D با گیت‌های NAND (فقط یک آی‌سی 7400) بسازید و صحت جدول تابع را تحقیق نمایید.

فلیپ فلاپ حاکم-تابع

با استفاده از دو لچ D و یک وارونگر یک فلیپ فلاپ D حاکم- تابع بسازید. ورودی D را به کلید و ورودی ساعت را به پالس ساز وصل کنید. خروجی لچ حاکم را به یک لامپ و خروجی لچ تابع را به لامپ دیگر وصل نمایید. مقدار ورودی را برابر با متمم خروجی انتخاب کنید. دکمه فشاری را روی پالس ساز بفشارید و سپس آن را رها نمایید تا یک پالس تولید شود. مشاهده خواهید کرد که تغییر حاکم در هنگام مثبت شدن پالس و تغییر تابع موقع منفی شدن آن رخ می‌دهد. کار را چند بار تکرار نمایید و ضمن انجام کار لامپ‌ها را مشاهده کنید. رشته انتقال را از ورودی به حاکم و از حاکم به تابع منفی توضیح دهید.

ورودی ساعت را از پالس ساز جدا نمایید و آن را به یک مولد ساعت وصل کنید. متمم خروجی را به ورودی D باز گردانید. این کار موجب می‌شود تا خروجی فلیپ فلاپ با هر پالس ساعت متمم شود. با استفاده از یک اسیلوسکوپ دو کاناله، امواج ساعت و خروجی‌های حاکم- تابع را مشاهده نمایید. تحقیق کنید که تأخیر بین خروجی‌های حاکم و تابع برابر نیمه مثبت یک سیکل ساعت است. یک نمودار زمانبندی برای نمایش رابطه موج ساعت و خروجی‌های حاکم و تابع بدست آورید.

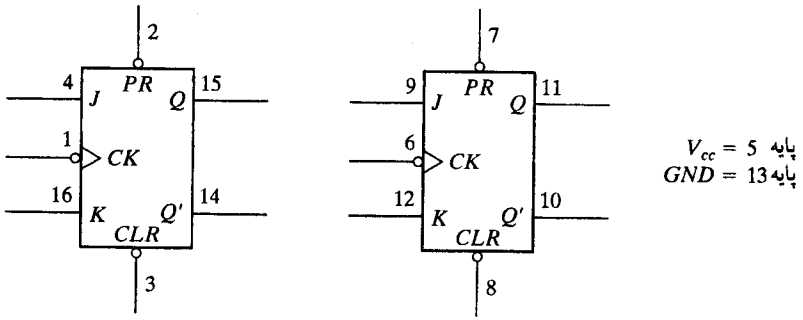
فلیپ فلاپ حساس به لبه

یک فلیپ فلاپ نوع D حساس به لبه مثبت با شش گیت NAND بسازید. ورودی ساعت را به پالس ساز و ورودی D را به کلید دگر وضع، و خروجی Q را به لامپ متصل نمایید. در ورودی D متمم Q را ایجاد کنید. نشان دهید که خروجی Q فقط در پاسخ به یک گذر مثبت پالس ساعت تغییر می‌نماید. تحقیق کنید که خروجی Q در ازاء 1 بودن پالس ساعت، گذر منفی، و یا 0 بودن آن تغییری نمی‌کند. در تمام مدت سعی شود ورودی D متمم Q باشد.

ورودی را از پالس ساز قطع نمایید و آن را به مولد ساعت وصل کنید. متمم خروجی، Q' را به ورودی D وصل نمایید. این سبب می‌شود تا خروجی با هر گذر مثبت پالس ساعت متمم گردد. با استفاده از اسیلوسکوپ دو کاناله رابطه زمانی بین ورودی ساعت و خروجی Q را مشاهده و ثبت نمایید. نشان دهید که خروجی در قبال یک گذر مثبت تغییر خواهد کرد.

فلیپ فلاپ‌های آی‌سی

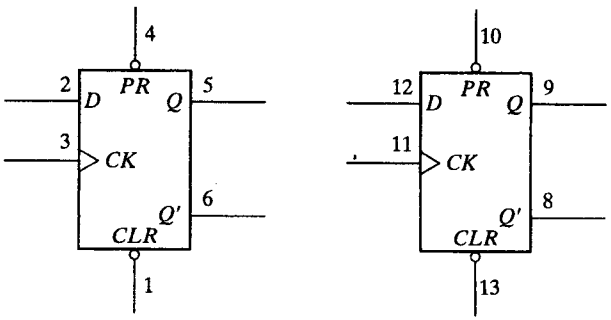
آی‌سی 7476 متشکل از فلیپ فلاپ‌های حاکم- تابع JK با امکان پیش تنظیم و پاک کردن است. تخصیص پایه برای هر فلیپ فلاپ در شکل ۱۲-۱۱ ملاحظه می‌شود. سه سطر اول جدول عملکرد ورودی‌های پیش تنظیم غیرهمزمان و پاک کردن را نشان می‌دهد. این ورودی‌ها همچون لچ SR نوع



جدول تابع

ورودی ها					خروجی‌ها	
PR	CLR	CK	J	K	Q	Q'
0	1	X	X	X	1	0
1	0	X	X	X	0	1
0	0	X	X	X	1	1
1	1	⊓	0	0	بلاتغییر	
1	1	⊓	0	1	0	1
1	1	⊓	1	0	1	0
1	1	⊓	1	1	متمم	

شکل ۱۲-۱۱. آی سی فلیپ فلاپ دوتایی تابع - متبوع JK نوع 7476



جدول تابع

ورودی‌ها				خروجی‌ها	
PR	*CLR*	*CK*	*D*	*Q*	*Q'*
0	1	*X*	*X*	1	0
1	0	*X*	*X*	0	1
0	0	*X*	*X*	1	1
1	1	↑	0	0	1
1	1	↑	1	1	0
1	1	0	*X*	بلاتغییر	

شکل ۱۳-۱۱ . آی سی فلیپ فلاپ D حساس به نوع لبهٔ مثبت

NAND عمل کرده و مستقل از ساعت یا ورودی‌های J و K است (Xها حالات بی‌اهمیت هستند). چهار وارده آخر در جدول تابع بیانگر اثر پالس ساعت بوده و در این حال هر دو ورودی پیش تنظیم و پاک کردن در منطق 1 هستند. مقدار پالس ساعت به صورت تک پالس نشان داده شده است. گذر مثبت پالس فلیپ فلاپ حاکم و گذر منفی آن فلیپ فلاپ تابع و نیز خروجی مدار را تغییر می‌دهد. با $J = K = 0$، خروجی عوض نمی‌شود. فلیپ فلاپ 7476 را بررسی کرده و جدول تابع را تحقیق کنید.

آی‌سی نوع 7474 از دو فلیپ فلاپ حساس به لبه مثبت همراه با ورودی پاک کردن و پیش تنظیم تشکیل شده است. تخصیص پایه‌ها در شکل ۱۳-۱۱ دیده می‌شود. جدول تابع اَعمال پاک کردن پیش تنظیم و ساعت را مشخص می‌کند. ساعت به همراه یک فلش رو به بالا به منظور مشخص کردن حساسیت فلیپ‌فلاپ به لبه مثبت است. طرز کار یکی از فلیپ فلاپ‌ها و جدول تابع آن را تحقیق کنید.

۹-۱۱ مدارهای ترتیبی

در این آزمایش، شما سه مدار ترتیبی همزمان را طراحی، ساخته و تست خواهید نمود. آی‌سی نوع 7476 (شکل ۱۲-۱۱) یا 7474 (شکل ۱۳-۱۱) را به کار ببرید. هر کدام که تعداد کل آی‌سی‌ها را حداقل کند، انتخاب نمایید. مطالب مربوط به مدارهای ترتیبی همزمان در بخش ۷-۵ ملاحظه شد.

بالا- پایین‌شمار با فعال‌ساز

یک شمارنده 2 بیت بالا یا پایین‌شمار را طراحی، ساخته و تست کنید. یک ورودی فعال‌ساز E مشخص کننده فعال شدن یا نشدن شمارنده می‌باشد. اگر $E = 0$ باشد، شمارنده غیرفعال شده و علیرغم اعمال پالس‌های ساعت به فلیپ فلاپ‌ها، در حال فعلی خود باقی خواهد ماند. اگر $E = 1$ باشد، شمارنده فعال شده و دومین ورودی، x، جهت شمارش را معین خواهد کرد. اگر $x = 1$ باشد مدار رو به بالا و با رشته 00، 01، 10 و 11 شمرده و شمارش تکرار خواهد شد. اگر $x = 0$ گردد، مدار رو به پایین شمرده و رشته شمارش 11، 10، 01 و 00 خواهد بود و سپس تکرار می‌شود. از E برای غیرفعال کردن ساعت استفاده نکنید، مدار ترتیبی را با E و x به عنوان ورودی‌ها بسازید.

نمودار حالت

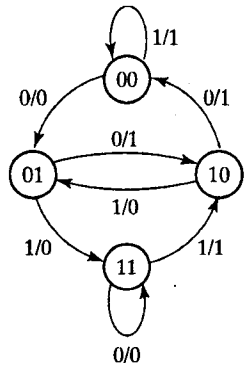
یک مدار ترتیبی که نمودار حالتش در شکل ۱۴-۱۱ دیده می‌شود را طراحی، ساخته و تست نمایید. دو فلیپ فلاپ را A و B و ورودی را x و خروجی را y بخوانید.

خروجی فلیپ فلاپ کم‌ارزش‌تر B را به ورودی x وصل کرده و رشته حالات و خروجی که بر اثر اعمال پالس ساعت رخ می‌دهند را تخمین بزنید. گذر حالت و خروجی را با تست مدار تحقیق نمایید.

طراحی شمارنده

شمارنده‌ای طراحی کنید که وارد رشته حالات دودویی زیر شود: 0، 1، 2، 3، 6، 7، 10، 11، 12، 13، 14 و 15 و دو مرتبه بازگشت به 0 و تکرار. توجه کنید که حالات دودویی 4، 5، 8 و 9 به کار نرفته است. شمارنده باید خود آغازگر باشد، یعنی اگر مدار از هر یک از چهار حالت نامعتبر شروع به کار کند، پالس‌های شمارش مدار را به یکی از حالات معتبر خواهد برد تا شمارش ادامه یابد.

عملکرد مدار را برای تست رشته شمارش موردنظر بررسی نمایید. آیا مدار خود آغاز است. این عمل با مقداردهی اولیه و بردن مدار به یکی از حالات به کار نرفته، با استفاده از پیش تنظیم و ورودی پاک و سپس اعمال پالس برای ردیابی رسیدن شمارنده تا رسیدن به یکی از حالات معتبر صورت می‌گیرد.



شکل ۱۴-۱۱. نمودار حالت آزمایش ۹

۱۰-۱۱ شمارنده‌ها

در این آزمایش، شما انواع مدارهای شمارنده همزمان و موج‌گونه را ساخته و تست خواهید کرد. شمارنده‌های موج‌گونه در بخش۳-۶ ملاحظه شدند و شمارنده‌های همزمان در بخش ۴-۶ مشاهده گردیدند.

شمارنده موج‌گونه

یک شمارنده موج‌گونه دودویی 4 بیت را با استفاده از 7476 بسازید (شکل ۱۲-۱۱). همه ورودی‌های غیرهمزمان پاک کردن و پیش تنظیم را به منطق 1 ببرید. ورودی شمارش پالس را به یک پالس ساز وصل کرده و عملکرد صحیح شمارنده را چک نمایید.

مدار را طوری اصلاح کنید که به عوض رو به بالا، رو به پایین بشمارد. نشان دهید که هر پالس ورودی شمارنده را یک واحد کم می‌کند.

شمارنده همزمان

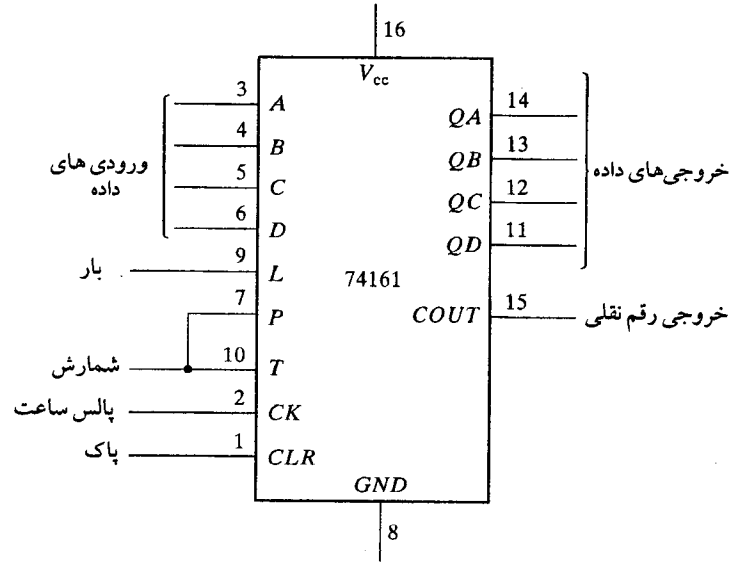
یک شمارنده دودویی 4 بیت همزمان ساخته و عملکرد آن را چک کنید. دو آی‌سی 7476 و یک آی‌سی 7408 به کار ببرید.

شمارنده دهدهی

یک شمارنده BCD همزمان که از 0000 تا 1001 بشمارد طراحی نمایید. از دو آی‌سی 7476 و یک آی‌سی 7408 استفاده کنید. از نظر شمارش صحیح شمارنده را تست کنید. آیا خود آغاز است. این عمل با شروع شمارش از هر یک از حالات به کار نرفته و با استفاده از امکان پیش تنظیم و پاک کردن صورت می‌گیرد. در صورت خود آغاز بودن، اعمال پالس‌ها به شمارنده باید آن را به یک حالت معتبر ببرد.

شمارنده دودویی با بار شدن موازی

آی‌سی 74161 شمارنده چهار بیت همزمان با بار شدن و پاک شدن غیرهمزمان است. مدار داخلی آن مشابه شکل ۱۴-۶ می‌باشد. تخصیص پایه به ورودی‌ها و خروجی‌ها در شکل ۱۵-۱۱ ملاحظه می‌گردد. وقتی که بار شدن فعال شود چهار داده ورود داده به داخل فلیپ فلاپ‌های درونی Q_A تا Q_D می‌روند که در آن Q_D باارزش‌ترین بیت است. در این آی‌سی دو ورودی فعال‌ساز شمارش P و T وجود دارد. برای انجام شمارش هر دو ورودی باید برابر 1 باشند. جدول تابع با یک استثناء مشابه جدول ۶-۶ است: ورودی بارکردن در 74161 با 0 فعال می‌شود. برای بارکردن داده باید ورودی پاک کردن در 1 و ورودی بار باید در 0 باشد. دو ورودی شمارش حالت بی‌اهمیت را داشته و می‌توانند 1 و یا 0 باشند. فلیپ فلاپ‌های داخلی باگذر مثبت پالس ساعت عمل می‌کنند. مدار به شرطی به صورت یک شمارنده



جدول تابع

تابع	شمارش	بار	ساعت	پاک کننده
پاک کردن خروجی به 0	X	X	X	0
بار کردن داده ورودی	X	0	↑	1
شمارش تا عدد دودویی بعدی	1	1	↑	1
عدم تغییر در خروجی	0	1	↑	1

شکل ۱۵-۱۱ . آی سی شمارنده دودویی 74161 با بار شدن موازی

کار می‌کند که ورودی بار و هر دو ورودی شمارش P و T در 1 باشند. اگر هر یک از P یا T برابر 0 شوند، خروجی تغییری نخواهد کرد. آزمایشی را در تأیید جدول تابع برای آی‌سی 74161 انجام دهید.

نشان دهید که چگونه می‌توان 74161 همراه با دو گیت NAND دو ورودی به عنوان یک شمارنده BCD همزمان ساخت تا از 0000 تا 1001 بشمارد. ورودی پاک کردن را به کار نبرید. با استفاده از یک گیت NAND حالت 1001 را چک کنید و سپس تمام شمارنده را با 0 پر کنید.

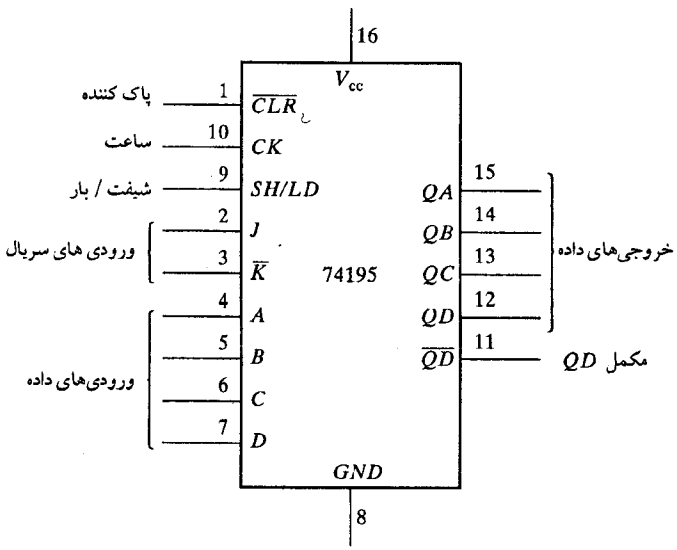
۱۱-۱۱ شیفت رجیستر

در این آزمایش، عملکرد یک شیفت رجیستر (ثبات جابجایی) را تحقیق خواهید کرد. آی‌سی مورد استفاده، شیفت رجیستر 74195 با امکان بار شدن موازی است. شیفت رجیستر در بخش ۲-۶ مورد بحث قرار گرفت.

آی‌سی شیفت رجیستر

آی‌سی 74165 یک شیفت رجیستر 4 بیت با بار شدن موازی و پاک شدن غیرهمزمان است. تخصیص پایه‌ها به ورودی‌ها و خروجی‌ها در شکل ۱۶-۱۱ نشان داده شده است. تک خط کنترل که با SH/LD (بار/ جابجایی) علامت خورده است عملکرد همزمان آن را معین می‌نماید. وقتی $SH/LD=0$ است، ورودی کنترل در حالت بارکردن بوده و داده ورودی چهار بیتی وارد چهار فلیپ فلاپ QA تا QD می‌شوند. اگر $SH/LD = 1$ گردد ورودی کنترل در حالت شیفت است و اطلاعات در ثبات به راست و از QA به سمت QD جابجا می‌شود. ورودی سریال به QA در حین جابجایی از ورودی‌های J و $\overline{K}$ معین می‌گردد. این دو ورودی مانند J و متمم K در فلیپ فلاپ JK عمل می‌کنند. وقتی هر دو J و $\overline{K}$ برابر 0 باشند، فلیپ فلاپ QA پس از جابجایی به 0 پاک می‌شوند. اگر هر دو ورودی برابر 1 باشند، QA پس از جابجایی برابر 1 می‌گردد. دو حالت باقیمانده برای J و $\overline{K}$ متمم یا حالت بی‌تغییر را در خروجی فلیپ فلاپ QA، پس از جابجایی به وجود می‌آورند.

جدول تابع برای 74195، حالت عمل ثبات را نشان می‌دهد. وقتی که ورودی پاک‌کردن به 0 می‌رود، فلیپ فلاپ‌ها به طور غیرهمزمان، یعنی بدون نیاز به پالس ساعت 0 می‌شوند. اعمال همزمان با لبه مثبت



جدول تابع

تابع	ورودی سریال	$\overline{K}$	J	ساعت	شیفت/بار	پاک‌کننده
پاک کننده آسنکرون	X	X	X	X	X	0
عدم تغییر در خروجی	X	X	X	0	X	1
بار شدن داده و رودی	X	X	X	↑	0	1
شیفت از QA به سمت QD، $QA = 0$	0	0	0	↑	1	1
شیفت از QA به سمت QD، $QA = 1$	1	1	1	↑	1	1

شکل ۱۶-۱۱ . آی سی شیفت رجیستر 74195 با بار شدن موازی

پالس ساعت انجام می‌شوند. برای بارکردن داده ورودی، SH/LD باید 0 شود و لبه بالارونده پالس ساعت باید رخ دهد. برای جابجایی به راست، باید $SH/LD = 1$ باشد. برای ایجاد ورودی سریال J و $\overline{K}$ باید به هم وصل شوند.

آزمایشی انجام دهید که عملکرد آی‌سی 74195 را تأیید کند. نشان دهید که این آی‌سی همه عملیات جدول تابع را اجرا می‌کند. دو حالت $J\overline{K} = 01$ و 10 را هم در جدول تابع خود لحاظ نمایید.

شمارنده حلقوی

یک شمارنده حلقوی، یک شیفت رجیستر دوار است که در آن سیگنال از خروجی سریال QD به ورودی سریال باز می‌گردد. برای ایجاد ورودی سریال ورودی‌های J و $\overline{K}$ را به هم وصل نمایید. از ورودی بار برای پیش تنظیم شمارنده حلقوی با مقدار اولیه 1000 استفاده کنید. ضمن دوران تک بیت 1 حالت ثبات را پس از هر پالس ساعت چک نمایید.

یک شمارنده حلقوی دنباله چرخان از خروجی متمم QD برای ورودی سریال استفاده می‌نماید. شمارنده حلقوی دنباله چرخان را با 0000 پیش تنظیم کنید و رشته حالات حاصل از شیفت رجیستر را پیش‌بینی نمایید. پیش‌بینی خود را با مشاهده رشته حالات پس از هر جابجایی تصدیق نمایید.

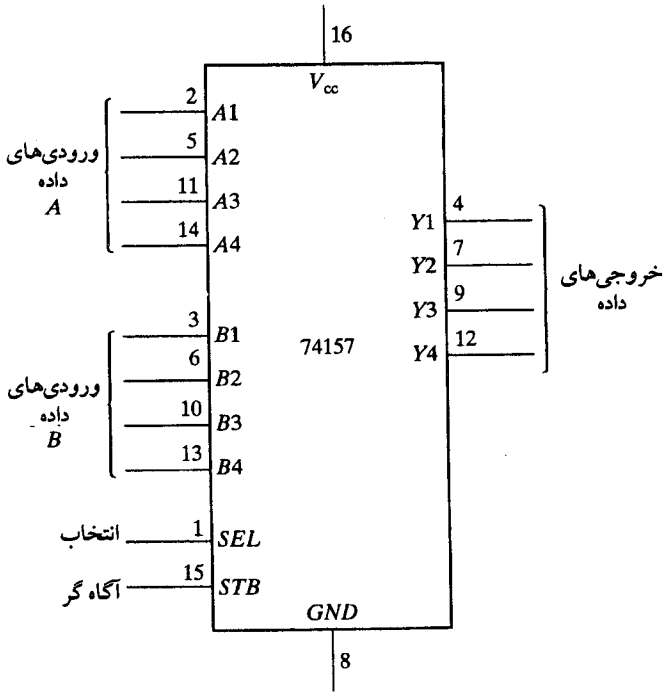
شیفت رجیستر پسخوردی

یک شیفت رجیستر پسخوردی، نوعی شیفت رجیستر است که در آن ورودی سریال به یک تابع خروجی از ثبات معینی وصل است. ورودی یک شیفت رجیستر پسخوردی را به XOR خروجی‌های QC و QD وصل کنید. رشته حالات ثبات را با $ABCD = 1000$ پیش‌بینی نمایید. صحت پیش‌بینی خود را با مشاهده رشته حالات پس از هر پالس ساعت تحقیق کنید.

شیفت رجیستر دو طرفه

آی‌سی 74195 فقط قادر است عمل جابجایی را به راست و از QA به سمت QD، انجام دهد. می‌توان برای تبدیل آن به یک شیفت رجیستر دو طرفه، از حالت بارکردن استفاده کرده و عمل جابجایی به چپ را (از QD به QA) انجام داد. این کار با اتصال خروجی هر فلیپ فلاپ به ورودی فلیپ فلاپ سمت چپ آن و استفاده از ورودی SH/LD به عنوان کنترل جابجایی، صورت می‌گیرد. ورودی D برای عمل جابجایی به چپ به عنوان ورودی سریال عمل خواهد کرد.

شیفت رجیستر 74195 را به عنوان یک جابجا کننده دو طرفه (بدون بار موازی) ببندید. ورودی سریال شیفت رجیستر را به کلید قطع و وصل متصل کنید. با اتصال خروجی سریال QA به ورودی سریال D یک شمارنده حلقوی ایجاد نمایید. ثبات را پاک کرده و سپس عمل آن را با اعمال 1 از سوئیچ ورودی چک کنید. سه بار دیگر جابجایی به راست را تکرار نموده و بعد از آن با ورودی کنترل جابجایی به چپ، یک بار چرخش به چپ را انجام دهید. عدد 1 باید ضمن جابجایی قابل رؤیت باشد.



جدول تابع

خروجی‌های داده Y	انتخاب	آگاه گر
تمام 0	X	1
انتخاب ورودی‌های داده A	0	0
انتخاب ورودی‌های داده B	1	0

شکل ۱۷-۱۱ . آی سی مولتی پلکسر 1×2، 74157 چهار تایی

شیفت رجیستر دو طرفه با بار شدن موازی

می‌توان آی‌سی 74195 را به کمک یک مولتی پلکسر تبدیل به یک شیفت رجیستر دو طرفه با بار شدن موازی نمود. برای این منظور از یک 74157 استفاده خواهیم کرد. این آی‌سی یک مولتی پلکسر چهارتایی دو خط به یک خط است که مدار داخلی‌اش در شکل ۲۶-۴ ملاحظه می‌شود. تخصیص پایه به ورودی‌ها و خروجی‌های 74157 در شکل ۱۷-۱۱ ملاحظه می‌گردد. توجه کنید که ورودی فعال‌ساز را در 74157، آگاه گر می‌نامند.

یک ثبات جابجایی با بار موازی دو طرفه را با 74195 و مولتی پلکسر 74157 بسازید. مدار باید قادر به انجام اعمال زیر باشد:

۱- ورودی پاک کردن همزمان داشته باشد.
۲- جابجایی به راست انجام دهد.
۳- جابجایی به چپ انجام دهد.
۴- قابلیت بار شدن موازی را داشته باشد.
۵- پاک شدن همزمان داشته باشد.

جدولی برای پنج عمل فوق به صورت تابعی از ورودی‌های پاک کردن، ساعت، جابجایی و بارکردن در 74195 و ورودی آگاه‌گر و انتخاب در 74157 تشکیل دهید. مدار را وصل نمایید و صحت جدول را تحقیق کنید. از حالت بار موازی برای تهیه یک مقدار اولیه در ثبات استفاده کنید و آنگاه خروجی سریال را به دو ورودی سریال وصل کنید تا اطلاعات ضمن جابجایی از دست نرود.

۱۲-۱۱ جمع سریال

در این آزمایش شما یک مدار جمع- تفریق‌گر سریال را ساخته و تست خواهید کرد. جمع سریال دو عدد دودویی را می‌توانید طبق بحث بخش ۲-۶ با شیفت رجیستر و تمام جمع‌کننده انجام دهید.

جمع‌کننده سریال

با شروع از شکل ۶-۶، یک جمع‌کننده سریال 4 بیت با استفاده از آی‌سی‌های 74195 (دو عدد)، 7408، 7486 و 7476 طراحی و پیاده‌سازی کنید. برای ثبات B امکاناتی فراهم کنید تا بتوانید داده موازی را از چهار کلید قطع و وصل بپذیرید و ورودی سریال را نیز به زمین وصل نمایید به نحوی که 0ها در حین جمع وارد ثبات B شوند. کلید دیگری لازم است تا به وسیله آن مشخص شود که آیا ثبات B اطلاعات موازی دریافت کند یا ضمن جمع عمل جابجایی انجام دهد.

تست جمع‌کننده

برای تست جمع‌کننده سریال، جمع دودویی $26 = 15 + 6 + 5$ را اجرا نمایید. این کار ابتدا با پاک کردن ثبات‌ها و فلیپ فلاپ نقلی انجام می‌گردد. مقدار 0101 را به طور موازی در ثبات B وارد کنید. چهار پالس برای جمع سریال B با A اعمال نمایید و تحقیق کنید که جواب نهایی در A برابر 0101 است. (توجه کنید که پالس ساعت برای 7476 باید طبق شکل ۱۲-۱۱ باشد. 0110 را به صورت موازی در B بارکنید و به صورت سریال با A جمع نمایید. وجود حاصل جمع صحیحی را در A تحقیق کنید. مقدار 1111 را به B وارد و آن را با A جمع کنید. آیا مقدار داخل A برابر 1010 و فلیپ فلاپ رقم نقلی نیز 1 شده است. ثبات‌ها و فلیپ فلاپ‌ها را پاک کرده و برای اطمینان از عملکرد جمع کننده سریال خود چند عدد دیگر را به کار ببرید.

جمع- تفریق‌گر سریال

اگر روال بخش ۲-۶ را برای طراحی یک تفریق‌گر سریال ($A - B$) دنبال کنیم در می‌یابیم که

خروجی تفاضل با خروجی جمع یکی است، ولی ورودی J و K از فلیپ فلاپ قرض باید متمم QD باشد (در 74195 موجود است). با استفاده از دو گیت XOR از 7486، جمع‌کننده سریال را به جمع-تفریق‌گر سریال با حالت‌کنترل M تبدیل نمایید. وقتی $M = 0$ است، مدار جمع $A + B$ را انجام می‌دهد. وقتی $M=1$ است، مدار تفریق $A-B$ را انجام داده و فلیپ فلاپ در عوض نقلی، قرض را نگه خواهد داشت.

بخش جمع را با تکرار اعمال پیشنهادی فوق تست کنید و اطمینان حاصل نمایید که اصلاح تغییری در عملکرد ایجاد نمی‌کند. بخش تفریق سریال را با اجرای اعمال $7- = 13 - 5 - 4 - 15$ تست نمایید. عدد 15 دودویی می‌تواند به ثبات A به این صورت انتقال یابد که ابتدا آن را با 0 پاک کرده و سپس با عدد 15 در B جمع کنید. در حین تفریق نتایج حاصل در هر مرحله را چک کنید. توجه داشته باشید که $7-$ به صورت متمم دو عدد 7 با قرض 1 از فلیپ فلاپ بدست می‌آید.

۱۳-۱۱ واحد حافظه

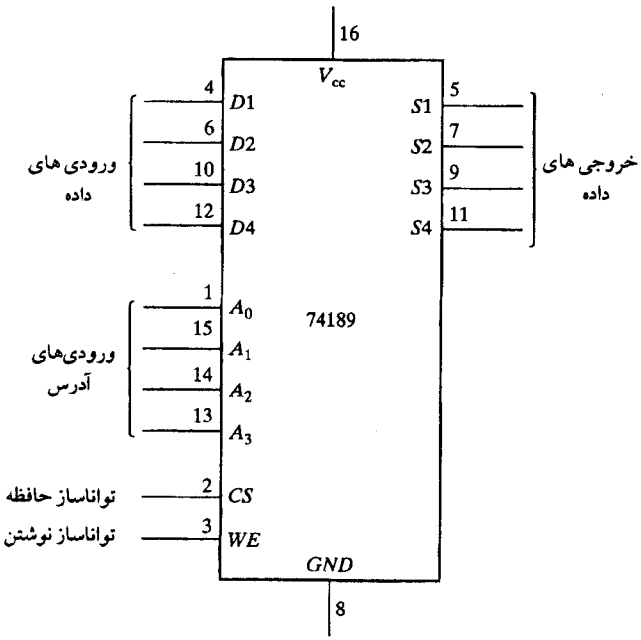
در این آزمایش رفتار یک حافظه RAM و قابلیت ذخیره‌سازی آن را تست خواهید کرد. از RAM برای شبیه‌سازی یک ROM استفاده خواهد شد. سپس شبیه‌ساز ROM، طبق بحث بخش ۵-۷ برای پیاده‌سازی مدارهای ترکیبی به کار می‌رود. واحد حافظه در بخش‌های ۲-۷ و ۳-۷ بحث گردید.

آی‌سی RAM

آی‌سی 74189 یک حافظه RAM با سایز 4×16 است. مدار درونی آن شبیه مدار RAM 4×4 در شکل ۶-۷ می‌باشد. تخصیص پایه به ورودی‌ها و خروجی‌ها در شکل ۱۸-۱۱ دیده می‌شود. چهار خط ورودی آدرس یکی از 16 کلمه در حافظه را انتخاب می‌نماید. کم‌ارزش‌ترین بیت آدرس $A0$ و باارزش‌ترین آن $A3$ است. ورودی انتخاب تراشه (CS) باید 0 باشد تا حافظه فعال گردد. اگر $CS = 1$ باشد، حافظه غیرفعال شده و هر چهار خروجی در حالت امپدانس بالا خواهد ماند. فعال‌ساز نوشتن (WE) نوع عمل را مطابق جدول تابع معین می‌نماید. وقتی $WE = 0$ باشد، عمل نوشتن اجرا می‌شود. این کار به معنی انتقال یک عدد دودویی از ورودی‌های داده به کلمه موردنظر حافظه است. عمل خواندن هنگامی انجام می‌گردد که $WE = 1$ باشد. این کار مقدار متمم کلمه ذخیره شده را به خطوط داده خروجی منتقل می‌سازد. حافظه برای ایجاد گسترش دارای خروجی‌های سه حالته است.

تست RAM

چون خروجی‌های 74189 مقادیر متمم تولید می‌نمایند، باید چهار وارونگر برای تغییر خروجی‌ها به مقدار معمولی، به مدار اضافه کنیم. تست حافظه RAM پس از اتصال‌های زیر امکان‌پذیر است: ورودی‌های آدرس را به یک شمارنده دودویی 7493 متصل نمایید (شکل ۳-۱۱). چهار ورودی داده را به کلیدهای قطع و وصل و چهار خروجی داده را به وارونگرهای 7404 متصل سازید. چهار لامپ برای



جدول تابع

CS	WE	عملکرد	خروجی‌های داده
0	0	نوشتن	امپدانس بالا
0	1	خواندن	مکمل کلمه منتخب
1	X	ناتوان	امپدانس بالا

شکل ۱۸-۱۱. آی سی RAM، 4×16 از نوع 7489

آدرس و چهار لامپ دیگر برای خروجی‌های وارونگر فراهم کنید. ورودی *CS* را به زمین و *WE* را به کلید قطع و وصل یا یک پالس ساز وصل نمایید. چند کلمه در حافظه ذخیره کنید و سپس آنها را بخوانید و مطمئن شوید که عمل خواندن و نوشتن به درستی انجام می‌گردد. هنگام استفاده از *WE* دقت لازم را مبذول نمایید. همواره ورودی *WE* را در حالت خواندن قرار دهید، مگر این که بخواهید در حافظه بنویسید. روش صحیح نوشتن این است که ابتدا آدرس را در شمارنده و ورودی‌ها را در چهار کلید قطع و وصل ایجاد کنیم. برای ذخیره کلمه در حافظه، کلید *WE* را در موقعیت نوشتن قرار داده و سپس آن را به حالت خواندن باز گردانید. دقت کنید که آدرس یا ورودی‌ها را وقتی *WE* در حالت نوشتن است، تغییر ندهید.

شبیه‌ساز ROM

یک شبیه‌ساز ROM از یک RAM که فقط در حالت خواندن کار کند، بدست می‌آید. ابتدا با استقرار حافظه در حالت نوشتن الگوهای 1 و 0 وارد RAM می‌شوند. شبیه‌سازی با حالت خواندن و انتخاب خطوط

آدرس به عنوان ورودی به ROM انجام می‌گردد. سپس ROM می‌تواند برای پیاده‌سازی مدار ترکیبی بکار رود.

با شبیه‌ساز ROM یک مدار ترکیبی پیاده‌سازی کنید تا طبق جدول ۶-۱ یک عدد دودویی چهار بیت را به کد گری تبدیل نماید. این کار به صورت زیر صورت می‌گیرد: جدول درستی مبدل کد را به دست آورید. با تنظیم ورودی‌های آدرس به مقادیر دودویی در جدول درستی و ورودی‌های داده با مقادیر کد گری مربوطه در 7489 ذخیره نمایید. پس از نوشتن هر 16 وارده جدول در حافظه، شبیه‌ساز ROM با $WE = 1$ ایجاد می‌گردد. با اعمال ورودی‌ها به خطوط آدرس، صحت خروجی‌ها در خطوط خروجی داده را تحقیق نمایید.

گسترش حافظه

واحد حافظه را با استفاده از دو آی‌سی 74189 به RAM 4 × 32 تبدیل کنید. ورودی‌های CS را برای انتخاب یکی از دو آی‌سی به کار ببرید. توجه کنید که چون خروجی‌ها سه حالته‌اند، می‌توانید جفت پایانه‌ها را به هم گره بزنید تا یک OR منطقی بین دو آی‌سی به وجود آید. مدار خود را با به‌کارگیری آن به عنوان یک ROM شبیه‌سازی شده که یک عدد 3 بیت را با عدد 2 بیت دیگر جمع می‌کند تا حاصل جمع 4 بیتی بدست آید. تست کنید مثلاً اگر ورودی به ROM، 10110 باشد، آنگاه خروجی برابر است با 0111 = 10 + 101. سه بیت اول ورودی عدد 5 و دو بیت آخر 2 را نمایش می‌دهند و خروجی نیز 7 است. از شمارنده برای ایجاد چهار بیت آدرس و از یک کلید نیز برای تولید بیت پنجم استفاده کنید.

۱۱-۱۴ هندبال لامپی

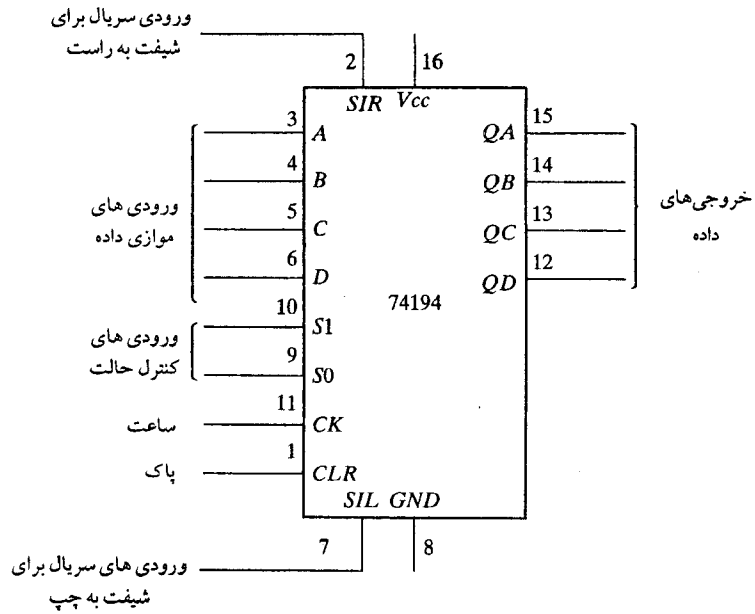
در این آزمایش، شما یک بازی الکترونیک به نام هندبال را با به کارگیری یک لامپ روشن برای شبیه‌سازی توپ استفاده خواهید کرد. این پروژه کاربرد دو طرفه یک شیفت رجیستر با بار شدن موازی را نشان می‌دهد. همچنین عمل ورودی غیرهمزمان فلیپ فلاپها هم نشان داده شده است. ما ابتدا آی‌سی مورد نیاز این آزمایش را معرفی کرده و سپس نمودار منطقی بازی هندبال لامپی را ارائه می‌کنیم.

آی‌سی 74194

این یک شیفت رجیستر دو طرفه با بار شدن موازی است. مدار درونی آن شبیه شکل ۷-۶ است. تخصیص پایه به ورودی‌ها و خروجی‌ها در شکل ۱۹-۱۱ نشان داده شده است. دو ورودی کنترل حالت نوع عمل را طبق جدول تابع مشخص می‌کند.

نمودار منطقی

نمودار منطقی هندبال لامپی الکترونیکی در شکل ۲۰-۱۱ مشاهده می‌شود. این مدار از دو آی‌سی



جدول تابع

تابع	حالت S0	حالت S1	ساعت	پاک
پاک شدن خروجی	X	X	X	0
عدم تغییر در خروجی	0	0	↑	1
شیفت به راست در جهت QA به QA	1	0	↑	1
شیفت به چپ در جهت QD به QD	0	1	↑	1
ورودی بار شدن موازی داده	1	1	↑	1

QD. SIR

QA. SIL

شکل ۱۹-۱۱ . آی سی شیفت رجیستر دو طرفه با بار شدن موازی نوع 74194

74194 یک آی‌سی فلیپ فلاپ D دوگانه 7474 و سه گیت 7400 ، 7404 و 7408 ساخته شده است. توپ با شبیه‌سازی لامپ متحرکی که از طریق شیفت رجیستر دو طرفه به چپ و راست می‌رود، حرکت می‌کند. سرعت حرکت لامپ با فرکانس ساعت معین می‌گردد. مدار ابتدا با کلید بازنشانی مقداردهی اولیه می‌گردد. کلید شروع، بازی را با استقرار توپ (یکی از لامپها) در منتهی‌الیه سمت راست آغاز می‌نماید. بازیکن باید دکمه فشاری پالس ساز را برای آغاز بازی بفشارد تا توپ به سمت چپ برود. لامپ روشن آنقدر به چپ منتقل می‌شود تا به سمت چپ‌ترین مکان برسد (دیوار) ، آنگاه توپ با معکوس شدن جهت جابجایی لامپ به بازیکن برمی‌گردد. وقتی که دوباره در سمت راست‌ترین مکان

قرار گرفت بازیکن باید کلید را بفشارد تا جهت جابجایی عوض شود. اگر بازیکن کلید را زودتر یا دیرتر از موعد مقرر بفشارد توپ ناپدید شده و لامپ خاموش می‌گردد. با روشن و خاموش کردن کلید شروع می‌توان بازی را دوباره آغاز کرد. در حین بازی کلید شروع باید باز (در حالت 1) باشد.

تحلیل مدار

قبل از اتصال مدار، نمودار منطقی را تحلیل کنید تا از درک عملکرد آن مطمئن شوید. بخصوص سعی کنید به سؤالات زیر پاسخ دهید.

۱- وظیفه کلید بازنشانی چیست؟

۲- توضیح دهید که چگونه نور در سمت راست‌ترین مکان با اتصال کلید شروع به زمین، روشن می‌گردد. چرا قبل از آغاز بازی باید کلید شروع را روی منطق 1 قرار داد.

۳- دو ورودی کنترل $S0$ و $S1$ در هنگام حرکت توپ چه حالاتی دارند.

۴- اگر ضمن حرکت توپ به چپ، کلید پالس‌ساز را فشار دهیم چه اتفاقی در ورودی‌های کنترل حالت به وجود می‌آید؟ اگر توپ در حال حرکت به راست باشد و عمل فوق تکرار گردد چه اتفاقی خواهد افتاد؟

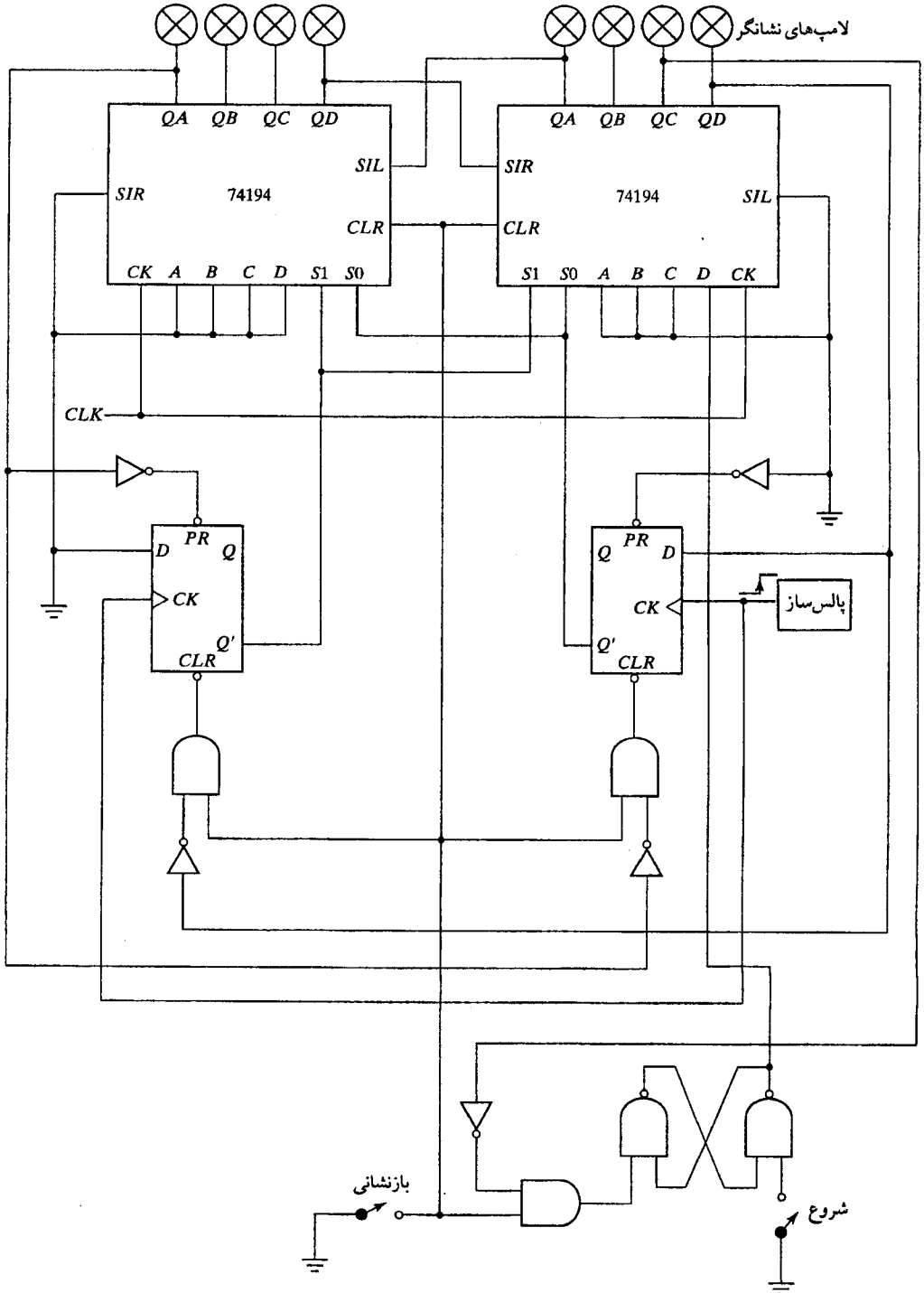
۵- فرض کنید که توپ به منتهی‌الیه سمت راست رسیده باشد ولی هنوز کلید پالس‌ساز فشرده نشده‌باشد. حالت ورودی‌های کنترل حالت، اگر پالس‌ساز فشار داده‌شود چه می‌شود؟ اگر فشار داده نشود چه می‌شود؟

انجام بازی

مدار شکل ۲۰-۱۱ را ببندید. با مدار برای اطمینان از صحت کارش بازی کنید. توجه شود که پالس ساز باید یک گذر مثبت تولید کند و نیز هر دو کلید بازنشانی و شروع در حین بازی باید باز شوند (در منطق 1). ابتدا با فرکانس ساعت پایینی شروع کرده و سپس برای داشتن یک بازی هیجان‌آور آن را بالا ببرید.

شمارش تعداد باخت‌ها

مداری طراحی کنید که تعداد دفعات باخت بازیکن را در حین بازی نگهدارد. از یک دیکدر BCD به هفت قسمتی و نمایشگر هفت قسمتی مثل شکل ۸-۱۱ استفاده کنید و شمارش 0 تا 9 را نمایش دهید. شمارش با شمارنده دهدهی با استفاده از 7493 به عنوان شمارنده موج‌گونه دهدهی یا 74161 و یک گیت NAND به عنوان یک شمارنده دهدهی همزمان صورت بگیرد. وقتی مدار در حالت بازنشانی است، شمارنده باید 0 را نشان دهد. هر بار که توپ ناپدید شود و لامپ خاموش گردد، نمایشگر به اندازه 1 واحد افزایش خواهد یافت. اگر لامپ در حین بازی روشن بماند مقدار نمایشگر نباید تغییر یابد. طرح نهایی باید یک مدار شمارش خودکار باشد و نمایش دهدهی آن به طور خودکار در ازاء هر باخت و خاموش‌شدن لامپ، یک واحد افزایش یابد.



شکل ۲۰-۱۱ . نمودار منطقی هندبال لامپی

لامپ پینگ پونگ TM

مدار شکل ۲۰-۱۱ را طوری اصلاح کنید که یک بازی پینگ پونگ لامپی بدست آید. در این بازی دو نفر می‌توانند شرکت کنند و هر بازیکن پالس‌ساز خاص خود را داراست. بازیکنی که پالس‌ساز سمت راست را در اختیار دارد، توپ را از سمت راست‌ترین مکان باز می‌گرداند و بازیکن سمت چپ هم به همین ترتیب توپ را از سمت چپ برمی‌گرداند. تنها اصلاح لازم برای بازی پینگ پونگ پالس ساز دوم و تغییر چند سیم می‌باشد.

۱۵-۱۱ مولد پالس ساعت

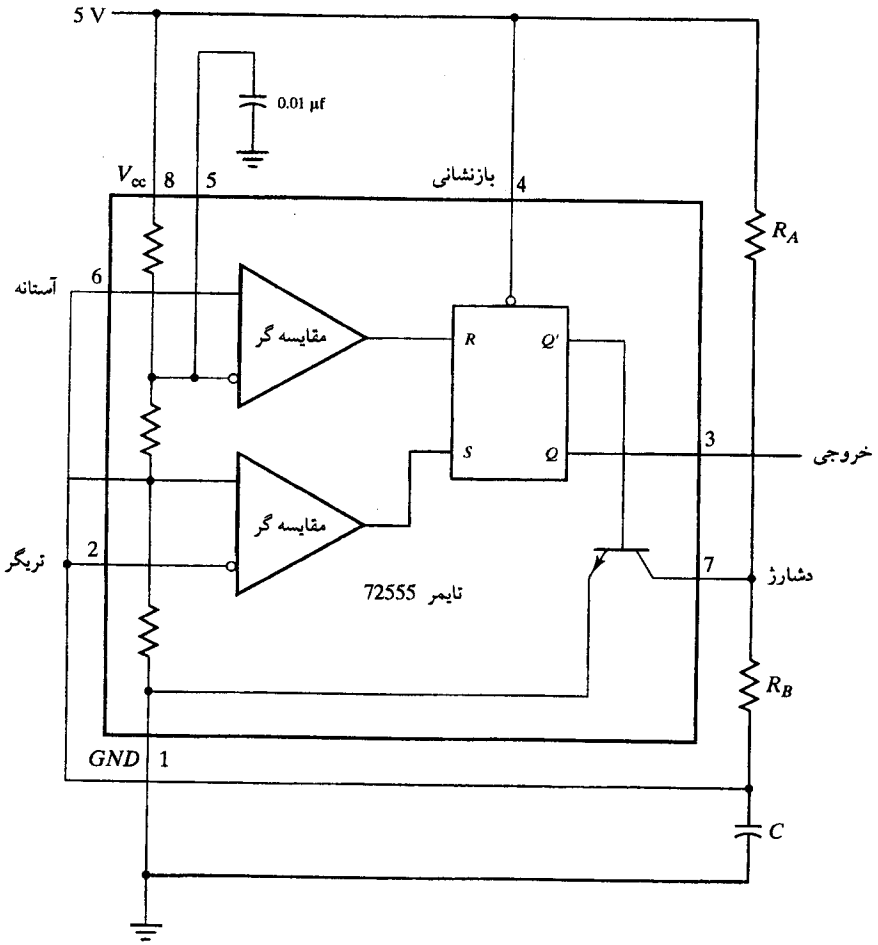
در این آزمایش، شما از یک آی‌سی تایمر استفاده کرده و برای تولید پالس‌های ساعت در فرکانس مفروضی آن را وصل خواهید کرد. این مدار به دو مقاومت و دو خازن بیرونی نیاز دارد. اسیلوسکوپ اشعه کاتودیک برای مشاهده امواج و اندازه‌گیری فرکانس به کار خواهد رفت.

آی‌سی تایمر

آی‌سی 72555 (یا 555) یک مدار تایمر دقیق است که داخل آن در شکل ۲۱-۱۱ ملاحظه می‌شود. (مقاومت‌های R_A و R_B و دو خازن بخشی از آی‌سی نیستند). این مدار حاوی دو مقایسه‌گر، یک فلیپ فلاپ و یک مقاومت درونی است. تقسیم ولتاژ $V_{CC} = 5V$ به وسیله سه مقاومت درونی به زمین وصل است تا $\frac{1}{3}$ و $\frac{2}{3}$ مقدار V_{CC} (3.3V و 1.7V) در ورودی مقایسه‌گرها در اختیار قرار گیرد. وقتی که ورودی آستانه در پایه 6 به بیش از 3.3V برسد، مقایسه‌گر بالایی فلیپ فلاپ را به 0 بازنشانی کرده و بنابراین خروجی به عدد 0V خواهد رفت. وقتی که ورودی راه‌اندازی در پایه 2 به زیر 1.7V برود، مقایسه‌گر پایینی فلیپ فلاپ را به 1 نشانده و خروجی ترانزیستور به حدود 5V خواهد رسید. وقتی که خروجی پایین است، Q' در سطح بالاست و پیوند بیس- امیتر ترانزیستور تغذیه مستقیم می‌گردد. وقتی که خروجی بالاست، Q' پایین و ترانزیستور خاموش می‌گردد (بخش ۲-۱۰). مدار تایمر می‌تواند تأخیر زمانی دقیقی را با مدار خارجی RC تولید کند. در این آزمایش تایمر در حالت آستابل برای تولید پالس ساعت به کار می‌رود.

عملکرد مدار

شکل ۲۱-۱۱ اتصالات بیرونی لازم را برای کار در حالت آستابل نشان می‌دهد. خازن C به هنگام خاموشی ترانزیستور شارژ شده و از طریق مقاومت‌های R_B به هنگام روشنی ترانزیستور دشارژ می‌گردد. وقتی که ولتاژ شارژ دو سر خازن C به 3.3V برسد، ورودی آستانه در پایه، موجب بازنشانی فلیپ فلاپ و در نتیجه روشنی ترانزیستور می‌گردد. وقتی دشارژ ولتاژ به 1.7V برسد، ورودی تریگر در پایه 2 موجب 1 شدن فلیپ فلاپ و خاموش شدن ترانزیستور می‌شود. بنابراین ولتاژ مرتباً بین دو سطح



شکل ۲۱-۱۱. آی سی تایمر 72555 که بعنوان مولد پالس ساعت بسته شده است.

در خروجی فلیپ فلاپ تغییر خواهد کرد. خروجی برای مدتی برابر با زمان شارژ در سطح بالا باقی می‌ماند. این دوره زمانی از معادله زیر بدست می‌آید.

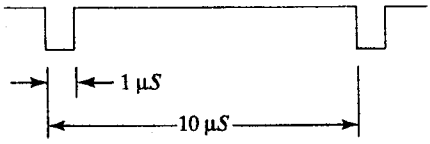
$$t_H = 0.693(R_A + R_B)C$$

خروجی برای مدت زمان دشارژ در سطح پایین باقی می‌ماند. این زمان از رابطه زیر محاسبه می‌گردد.

$$t_L = 0.693R_BC$$

مولد پالس ساعت

با داشتن خازن C به ظرفیت $0.001\mu F$، مقادیر R_A و R_B را برای تولید پالس‌های ساعت



شکل ۲۲-۱۱ . شکل موج مولد ساعت

شکل ۲۲-۱۱ محاسبه نمایید. عرض پالس $1\mu s$ در سطح پایین است، و با فرکانس 100KHz تکرار می‌شود. مدار را وصل و خروجی را روی اسیلوسکوپ چک نمایید.

خروجی دو سر خازن C را مشاهده نمایید و دو سطح آن را ثبت کنید و نشان دهید که دو سطح آن بین دو مقدار تریگر و آستانه است.

شکل موج در کلکتور ترانزیستور در پایه 7 را ملاحظه نمایید و همه اطلاعات مربوطه را ثبت کنید. شکل موج را با تحلیل عمل مدار توضیح دهید.

برای تولید مولد پالس فرکانس متغیر یک مقاومت متغیر (پتانسیومتر) را با R_A سری کنید. دوره سطح پایین همان $1\mu s$ باقی می‌ماند. فرکانس بین 20 تا 100KHz خواهد بود.

پالس‌های سطح پایین را به کمک یک وارونگر به پالس‌های سطح بالا تبدیل کنید. در این صورت پالس‌های $1\mu s$ با فرکانس متغیر خواهید داشت.

۱۱-۱۶ جمع‌کننده موازی و انباره

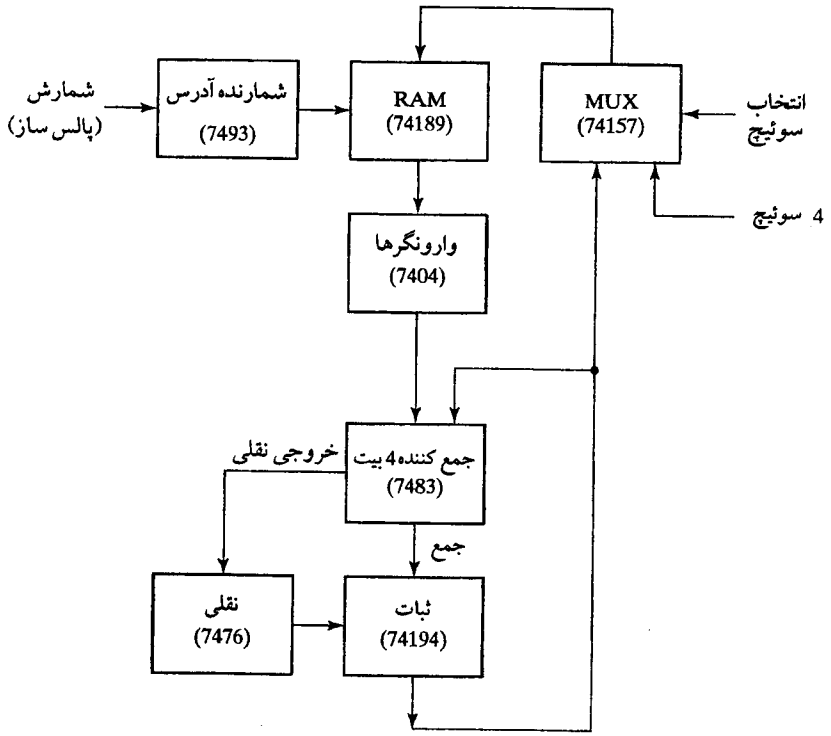
در این آزمایش، شما یک جمع‌کننده موازی چهار بیت، که جمع حاصل از آن می‌تواند در یک ثبات بار شود را خواهید ساخت. مجموعه اعدادی که باید جمع شوند در حافظه RAM ذخیره خواهند شد.

نمودار بلوکی

مدار RAM را از حافظه آزمایش بخش ۱۳-۱۱ انتخاب کرده و برای ساخت مدار از یک جمع‌کننده 4 بیت، یک شیفت رجیستر 4 بیت با بار شدن موازی، یک فلیپ فلاپ نقلی و یک مولتی پلکسر استفاده نمایید. نمودار بلوکی و IC های به‌کار رفته در شکل ۲۳-۱۱ مشاهده می‌گردد. اطلاعات حاصل از چهار کلید، یا از چهار بیت داده موجود در خروجی یک ثبات می‌تواند در یک RAM نوشته شود. انتخاب یکی از دو منبع اطلاعات به وسیله مولتی پلکسر انجام می‌گردد. داده در RAM می‌تواند با محتویات ثبات جمع شده و حاصل جمع به ثبات بازگردانده شود.

کنترل ثبات

سوئیچ‌های قطع و وصلی برای کنترل ثبات 74197 و فلیپ فلاپ نقلی 7476 مطابق زیر فراهم آورید.



شکل ۲۳-۱۱ . نمودار بلوکی جمع کننده موازی برای آزمایش ۱۶

(الف) با توجه به اعمال پالس ساعت شرایط بار شدن (LOAD) برای انتقال حاصل جمع به ثبات و خروجی نقلی به فلیپ فلاپ را فراهم آورید.

(ب) با توجه به اعمال پالس ساعت، شرایط جابجایی (SHIFT) به راست به همراه انتقال فلیپ فلاپ نقلی به سمت چپ‌ترین مکان ثبات را فراهم آورید. محتوای فلیپ فلاپ نقلی نباید در حین جابجایی تغییر کند.

(پ) حالت بی‌تغییری راکه محتوای ثبات و فیلپ فلاپ حتی با اعمال پالس ساعت تغییر نکنند را ایجاد نمایید.

مدار رقم نقلی

برای اطمینان از مشخصات فوق، مداری بین خروجی نقلی از جمع کننده و ورودی‌های J و K فلیپ فلاپ 7476 تهیه نمایید به طوری که خروجی رقم نقلی تحت شرایط بار شدن به فلیپ فلاپ (0 یا 1) ضمن اعمال پالس ساعت به آن انتقال یابد. اگر شرط بار شدن غیرفعال یا شرط جابجایی فعال گردد، فلیپ فلاپ رقم نقلی نباید تغییر یابد.

تست مدار

اعداد زیر را در RAM ذخیره کرده و سپس آنها را یک به یک با محتوای ثبات جمع کنید. کار را با ثبات و فلیپ فلاپ پاک شده آغاز نمایید. مقدار خروجی ثبات و رقم نقلی را پس از هر جمع حدس زده و صحت نتایج خود را تحقیق نمایید.

0110 + 1110 + 1101 + 0101 + 0011

عملکرد مدار

ثبات و رقم نقلی را پاک نموده و اعداد چهار بیت زیر را در آدرس‌های ذکر شده در RAM ذخیره نمایید.

آدرس	محتوا
0	0110
3	1110
6	1101
9	0101
12	0011

اکنون چهار عمل زیر را اجرا کنید.

۱- محتوای آدرس‌های ثبات با شرط LOAD را با محتوای ثبات جمع کنید.
۲- حاصل جمع را در آدرس 1 حافظه RAM ذخیره نمایید.
۳- محتویات ثبات و رقم نقلی را با شرط SHIFT به راست جابجا نمایید.
۴- محتوای جابجا شده ثبات را در آدرس 2 حافظه RAM ذخیره کنید.

محتوای سه مکان اول RAM را مطابق زیر تست نمایید.

آدرس	محتوا
0	0110
1	0110
2	0011

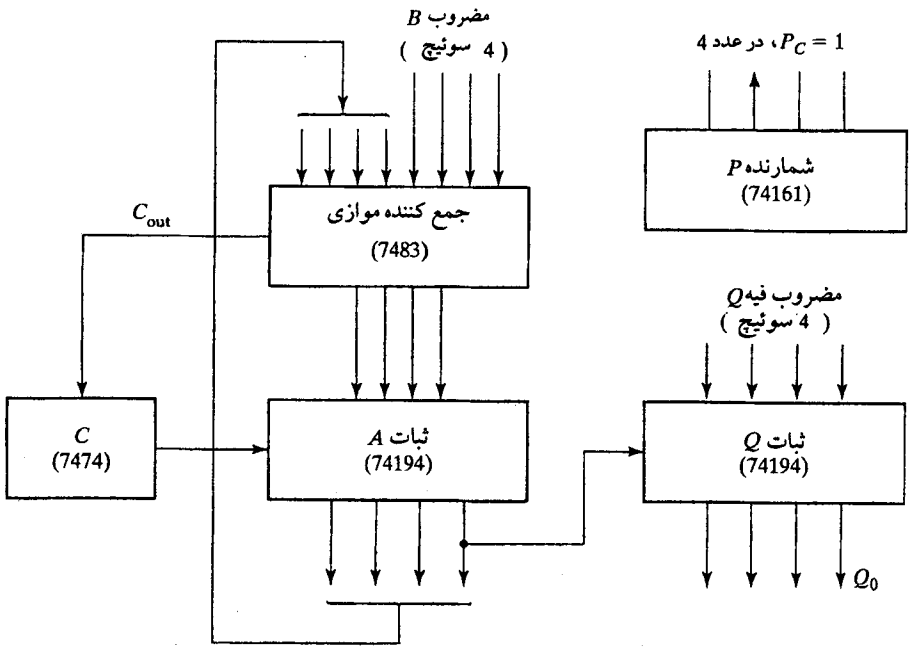
عملیات فوق را برای چهار عدد دودویی دیگر در RAM تکرار کنید. آدرس‌های 4، 7، 10 و 13 را برای ذخیره حاصل جمع موجود در ثبات در مرحله ۲ به کار ببرید. آدرس‌های 5، 8، 11 و 14 را برای ذخیره مقدار جابجا شده در ثبات در مرحله ۴ استفاده نمایید. محتوای آدرس‌های 0 تا 14 از RAM را تخمین بزنید و برای اطمینان از نتایج خود، آنها را تست کنید.

۱۷-۱۱ ضرب‌کننده دودویی

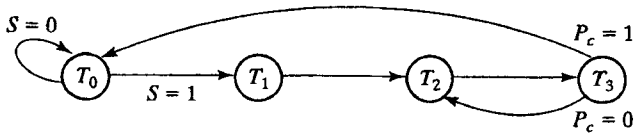
در این آزمایش، شما یک مدار ضرب‌کننده دو عدد 4 بیت بی‌علامت برای تولید حاصلضرب 8 بیت طراحی و خواهید ساخت. الگوریتم ضرب دو عدد دودویی در بخش ۶-۸ ملاحظه گردید.

نمودار بلوکی

نمودار بلوکی ضرب کننده دودویی با آی‌سی‌های پیشنهادی به کار رفته در شکل ۲۴-۱۱ (الف) نشان داده شده است. مضروب B در عوض یک ثبات، از چهار کلید حاصل می‌شود. مضروب فیه Q از چهار کلید دیگر بدست می‌آید. حاصلضرب با هشت لامپ مشخص می‌گردد. شمارنده P با 0 مقداردهی اولیه شده و با هر بار ایجاد حاصلضرب جزیی، یک واحد افزایش می‌یابد. وقتی که شمارنده به عدد چهار برسد، خروجی P_c برابر 1 شده و ضرب خاتمه می‌یابد.



(الف) نمودار پردازشگر داده



T_1: $A \leftarrow 0, C \leftarrow 0, P \leftarrow 0, Q \leftarrow$ مضروب فیه

T_2: $P \leftarrow P + 1$

T_2Q_0: $A \leftarrow A + B, C \leftarrow C_{out}$

T_3: $CAQ, C \leftarrow 0$ جابجایی به راست

(ب) نمودار حالت کنترل

شکل ۲۴-۱۱ . مدار ضرب دودویی

کنترل ثبات‌ها

چارت ASM برای ضرب کننده دودویی در شکل ۱۴-۸ نشان می‌دهد که سه ثبات و فلیپ فلاپ نقلی به وسیله سیگنال‌های T_1 ، T_2 و T_3 کنترل می‌شوند. یک سیگنال کنترل اضافی دیگر که به Q_1 بستگی دارد، جمع را در ثابت A و رقم نقلی را در فلیپ فلاپ C بار می‌نماید. Q_1 کم‌ارزش‌ترین بیت ثبات Q است. نمودار حالت کنترل و اعمال انجام شده در هر حالت در شکل ۲۴-۱۱ (ب) لیست شده است. T_2Q_0 با یک گیت AND که ورودی‌هایش T_2 و Q_0 است تولید می‌گردد. توجه کنید که فلیپ فلاپ نقلی C با هر پالس ساعت، به جز هنگام انتقال رقم نقلی به داخل آن، قابل پاک شدن است.

مثال ضرب کننده

قبل از اتصال مدار، مطمئن شوید که عملکرد مدار ضرب کننده را فهمیده‌اید. برای انجام این کار، جدولی مشابه جدول ۴-۸ بسازید، که در آن مضروب $B = 1111$ و مضروب فیه $Q = 1011$ باشد. در توضیحات سمت چپ جدول، بگویید کدام یک از متغیرهای حالت T_1 ، T_2 و T_3 ، در هر حالت فعال شده است. حالات باید با T_1 شروع و سپس T_2 و T_3 چهار بار تکرار می‌گردند.

طراحی پردازشگر داده

نمودار مشروحی برای بخش پردازشگر (مسیر داده) رسم کرده و همه اتصالات پایه‌های آی‌سی را نشان دهید. سیگنال‌های کنترل T_1 ، T_2 و T_3 را با سه کلید تولید نمایید و آنها را برای تهیه اعمال کنترلی موردنظر در انواع ثبات‌ها، به کار ببرید. مدار را وصل کرده و عملکرد صحیح هر قطعه را چک کنید. با قرار داشتن سه متغیر کنترل در 0 ، کلیدهای مضروب را در 1111 و مضروب فیه را در 1011 تنظیم کنید. متغیرهای کنترل را به صورت دستی یا به وسیله کلیدها طبق نمودار حالت شکل ۲۴-۱۱ (پ) تولید کنید. در هر حالت کنترل یک تک پالس اعمال نموده و خروجی ثبات‌های A و Q و مقادیر C و P_e را مشاهده نمایید. این مقادیر را با اعداد مثال عددی خود مقایسه و صحت عملکرد مدار را تحقیق نمایید. توجه داشته باشید که آی‌سی 74161 دارای فلیپ فلاپ حاکم- تابع است. برای به راه‌اندازی دستی، لازم است تک پالس‌های ساعت یک پالس منفی باشد.

طراحی کنترل

مدار کنترل را با توجه به نمودار حالت طراحی کنید. می‌توانید از هر یک از روش‌های پیاده‌سازی کنترل بخش ۷-۸ استفاده نمایید.

روشی را انتخاب کنید که تعداد آی‌سی‌ها حداقل باشد. صحت عملکرد مدار کنترل را قبل از اتصال به پردازشگر داده تحقیق نمایید.

تست ضرب کننده

خروجی‌های مدار کنترل را به پردازشگر داده وصل و عملکرد کلی مدار را با تکرار مراحل ضرب 1111 در 1011 ، بررسی نمایید. اکنون تک پالس‌های ساعت حالات کنترل را باید به ترتیب به پیش ببرند (کلیدهای دستی حذف شوند). سیگنال شروع S می‌تواند ضمن قرار داشتن کنترل در T_0 ، با یک کلید در وضعیت 1 تولید گردد.

سیگنال شروع S را به وسیله یک پالس‌ساز یا هر نوع پالس کوتاه دیگر تولید نموده و ضرب کننده را به وسیله پالس‌های ساعت حاصل از مولد پالس ساعت به کار اندازید. با فشار دادن S ، پالس‌ساز، عمل ضرب را تا پایان دوباره تکرار خواهد کرد، و حاصلضرب در ثبات‌های A و Q ایجاد خواهد شد. دقت کنید که S را به 0 بازگردانده و سپس کلیدها را با دو عدد چهار بیتی دیگر تنظیم نموده‌اید و آنگاه S را مجدداً فشار دهید. حاصلضرب جدید باید در خروجی‌ها ظاهر شود. ضرب با چند عدد دیگر را برای اطمینان از صحت عملکرد مدار تکرار نمایید.

۱۸-۱۱ مدارهای ترتیبی غیرهمزمان

در این آزمایش شما مدارهای ترتیبی غیرهمزمان را طراحی و خواهید ساخت. این مدارها در فصل ۹ بررسی شدند.

مثال تحلیل

تحلیل مدارهای ترتیبی غیرهمزمان با لچ‌های SR در بخش ۳-۹ آورده شد. مدار شکل (م۹-۹) را در بخش مسائل با ترسیم جدول گذر و نقشه خروجی مدار تحلیل کنید. از روی جدول گذر و نقشه خروجی، معین کنید؛ (الف) وقتی ورودی x_1 بدون توجه به مقدار ورودی x_2، برابر 1 باشد، چه در خروجی Q رخ می‌دهد؛ (ب) وقتی $x_2 = 1$ و $x_1 = 0$ باشد در Q چه اتفاقی می‌افتد. (پ) وقتی دو ورودی به 0 باز گردند چه می‌شود؟

مدار را ببندید و نشان دهید که عملکرد آن با تحلیل مطابقت دارد.

مثال طراحی

مدار فلیپ فلاپ D حساس به لبه مثبت در شکل ۱۰-۵ نشان داده شد. مدار فلیپ فلاپ T حساس به لبه منفی در شکل ۴۶-۹ نشان داده شد. با استفاده از روند ۶ مرحله‌ای در بخش ۸-۹ یک فلیپ فلاپ D ، طراحی، ساخته و تست نمایید به نحوی که در هر دو لبه منفی و مثبت پالس ساعت تریگر شود. مدار دارای ورودی‌های D و C و یک خروجی Q است. پس از تغییر C از 0 به 1 مقدار خروجی با ورودی D برابر می‌شود. اگر C در 1 باقی بماند، خروجی بدون توجه به D بدون تغییر خواهد ماند. در گذر بعدی پالس ساعت یعنی، به هنگام تغییر C از 1 به 0 ، خروجی دوباره به مقدار D اصلاح می‌گردد. مجدداً هنگامی که $C = 0$ باشد خروجی بدون تغییر است.

۱۹-۱۱ آزمایش‌های شبیه‌سازی VERILOG HDL

بعضی از آزمایش‌های سخت‌افزاری بررسی شده در این فصل را می‌توان با یک روال نرم‌افزاری تکمیل کرد، که در آن از زبان توصیف نرم‌افزاری Verilog (HDL) استفاده شده است. برای تکمیل، کامپایلر و شبیه‌ساز Verilog مورد نیاز است. در زیر پیشنهادات شبیه‌سایز و تست بعضی از مدارهای به کار رفته و آزمایشات آمده است.

تکمیل آزمایش ۲ (بخش ۲-۱۱)

انواع گیت‌های منطقی و تأخیر انتشار آنها در آزمایش سخت‌افزاری معرفی شد. در بخش ۹-۳، مدار ساده‌ای با تأخیر گیت مورد بررسی قرار گرفت. به عنوان مقدمه‌ای بر برنامه آزمایشگاهی Verilog مدار توصیف شده در مثال ۳-۳ HDL را کامپایل کرده و شبیه‌ساز را به کار اندازید تا صحت امواج شکل ۳-۳۸ تحقیق گردد.

به مدار XOR شکل ۳-۳۲ (الف) تأخیرهای زیر را نسبت دهید: 10ns برای یک وارونگر، 20ns برای یک گیت AND، و 30ns برای یک گیت OR. ورودی مدار از $xy = 00$ به $xy = 01$ می‌رود.

(الف) خروجی هر گیت را بین $t = 0$ تا $t = 50ns$ معین کنید.

(ب) توصیف HDL مدار از جمله تأخیرها را بنویسید.

(پ) مدول محرک (مشابه مثال ۳-۳ HDL) را بنویسید و برای بررسی پاسخ بخش (الف) آن را شبیه‌سازی نمایید.

تکمیل آزمایش ۴ (بخش ۴-۱۱)

عملکرد یک مدار ترتیبی با تست خروجی و مقایسه آن با جدول درستی مدار تحقیق شد. مثال ۱۰-۴ HDL (بخش ۱۱-۴) روال تهیه جدول درستی به کمک شبیه‌سازی را برای آن نشان می‌دهد. برای دستیابی و آشنایی با این روال، مثال ۱۰-۴ HDL را کامپایل و شبیه‌سازی نمایید و سپس جدول درستی خروجی را چک کنید.

در آزمایش ۴، شما مدار منطقی اکثریت را طراحی کردید. توصیف سطح گیت HDL را برای مدار منطقی اکثریت بنویسید و نیز جدول درستی را هم نمایش دهید. مدار را کامپایل و شبیه‌سازی کرده و پاسخ خروجی را چک نمایید.

تکمیل آزمایش ۵ (بخش ۵-۱۱)

این آزمایش در مورد تبدیل کد ارائه شده است. یک مبدل BCD به افزونی 3 در بخش ۳-۴ طراحی شد. نتایج طرح را برای تست آن با نتایج حاصل از یک شبیه‌ساز HDL به کار ببرید.

(الف) توصیف سطح گیت HDL را برای مدار شکل ۴-۴ بنویسید.

(ب) توصیف روند داده را با استفاده از عبارات بولی شکل ۳-۴ بنویسید.
(پ) توصیف رفتاری HDL یک مبدل BCD به افزونی 3 را بنویسید.
(ت) یک برنامه تست برای شبیه‌سازی و تست مدار مبدل BCD به افزونی 3 بنویسید تا جدول درستی را تصدیق نماید. هر سه مدار را چک نمایید.

تکمیل آزمایش ۷ (بخش ۷-۱۱)

در این آزمایش یک جمع- تفریق‌گر ساخته شد. این مدار در بخش ۴-۴ هم ساخته شده بود.
(الف) یک توصیف رفتاری HDL برای جمع کننده 4 بیت 7483 بنویسید.
(ب) یک توصیف رفتاری برای جمع- تفریق‌گر مدار شکل ۱۱-۱۱ بنویسید.
(پ) یک توصیف سلسله مراتبی برای جمع- تفریق‌گر 4 بیت شکل ۱۳-۴ بنویسید (شامل ۷). این کار با ذکر نوع اصلاح شده جمع کننده 4 بیت در مثال ۲-۴ HDL امکان‌پذیر است.
(ت) یک برنامه تست HDL برای شبیه‌سازی و تست مدارهای بخش (پ) بنویسید.مقادیری که سبب سرریز می‌گردند، $V = 1$، را چک کنید.

تکمیل آزمایش ۸ (بخش ۸-۱۱)

فلیپ فلاپ D حساس به لبه در شکل ۱۳-۱۱ ملاحظه شد. این فلیپ فلاپ دارای ورودی‌های پیش تنظیم و پاک کردن است.
(الف) با استفاده از هر دو خروجی یک توصیف رفتاری HDL برای فلیپ فلاپ 7474D بنویسید (توجه شود که وقتی پیش تنظیم برابر 0 باشد، $Q = 1$ می‌شود، و وقتی پیش تنظیم برابر 1 و پاک کردن هم 0 باشد، $Q = 0$ خواهد شد. بنابراین پیش تنظیم بر پاک کردن اولویت دارد).
(ب) یک توصیف رفتاری HDL ، با استفاده از هر دو خروجی برای فلیپ فلاپ 7474 بنویسید. دومین خروجی را Q−not نام‌گذاری کنید و توجه داشته باشید این، همیشه متمم Q نیست. (وقتی پیش تنظیم و پاک کردن هر دو 0 باشند، Q و Q−not به 1 می‌روند.

تکمیل آزمایش ۹ (بخش ۹-۱۱)

در آزمایش سخت‌افزاری، از شما خواسته شد تا یک مدار ترتیبی با نمودار حالت شکل ۱۴-۱۱ طراحی و تست کنید. این مدار به نام مدار ترتیبی مدول میلی خوانده می‌شود و مشابه با توصیف HDL مثال ۵-۵ (بخش ۵-۵) است.
(الف) توصیف HDL نمودار حالت شکل ۱۴-۱۱ را بنویسید.
(ب) توصیف ساختاری HDL مدار ترتیبی حاصل از طرح را بنویسید (این مشابه مثال ۷-۵ HDL در بخش ۵-۵ است).

(پ) شکل ۲۴-۱۱ (ب) (بخش ۱۷-۱۱) یک نمودار حالت کنترلی را نشان می‌دهد.

توصیف HDL نمودار حالت را با استفاده از تخصیص دودویی 1 - بارز (جدول ۹-۵ در بخش ۶-۵ ملاحظه شود)، به همراه سه خروجی T_1، T_2 و T_3 بنویسید.

تکمیل آزمایش ۱۰ (بخش ۱۰-۱۱)

آی‌سی شمارنده همزمان با بار شدن موازی 74161، در شکل ۱۵-۱۱ ملاحظه شد. با دو اختلاف، این شکل مشابه مثال ۳-۶ HDL در بخش ۶-۶ است. ورودی بارکردن وقتی 0 شود فعال می‌گردد و دو ورودی P و T وجود دارند که شمارش را کنترل می‌کنند. توصیف HDL آی‌سی 74161 را بنویسید.

تکمیل آزمایش ۱۱ (بخش ۱۱-۱۱)

یک شیفت رجیستر دو طرفه با بار شدن موازی با استفاده از آی‌سی‌های نوع 74195 و 74157 طراحی گردید.

(الف) توصیف HDL شیفت رجیستر 74195 را بنویسید. فرض کنید که ورودی‌های J و $\bar{K}$ به هم وصل شده‌اند تا ورودی سریال را به وجود آورند.

(ب) توصیف HDL مولتی پلکسر 74157 را بنویسید.

(پ) توصیف HDL شیفت رجیستر چهار بیت طراحی شده در آزمایش را بدست آورید.

(۱) توصیف ساختاری را با ذکر دو آی‌سی بنویسید و اتصالات داخلی آنها را مشخص نمایید، و

(۲) توصیف رفتاری مدار را با استفاده از جدول تابعی که در آزمایش بدست آمده، بنویسید.

تکمیل آزمایش ۱۳ (بخش ۱۳-۱۱)

این آزمایش عملکرد حافظه RAM را بررسی می‌کند. روش توصیف یک حافظه در HDL در بخش ۲-۷ به همراه مثال ۱-۷ HDL توضیح داده شد.

(الف) توصیف HDL یک حافظه 74189 RAM شکل ۱۸-۱۱ را بنویسید.

(ب) عملکرد حافظه را با نوشتن برنامه محرک که عدد 3 را در آدرس 0 و عدد دودویی 1 را در آدرس 14 بنویسد تست‌کنید. آنگاه اعداد ذخیره‌شده را از دو آدرس بخوانید و ببینید آیا صحیح ذخیره شده‌اند.

تکمیل آزمایش ۱۴ (بخش ۱۴-۱۱)

توصیف رفتاری HDL شیفت رجیستر دو طرفه 74194 را با بارشدن موازی مطابق شکل ۱۹-۱۱ بنویسید.

تکمیل آزمایش ۱۶ (بخش ۱۶-۱۱)

یک جمع‌کننده با ثبات انباره خود و یک واحد حافظه در شکل ۲۳-۱۱ ملاحظه شد. توصیف

ساختاری مدار مشخص شده در شکل را بنویسید. توصیف ساختاری این مدار را می‌توان با ذکر قطعات مختلف بدست آورد. به عنوان مثالی از یک توصیف ساختاری به مثال ۴-۸ در بخش ۵-۸ مراجعه شود. ابتدا لازم است توصیف رفتاری هر دو قطعه نوشته شود. شمارنده 74161 را به عوض 7493 به کار ببرید و فلیپ فلاپ 7474 را به جای فلیپ فلاپ *JK* 7476 استفاده کنید. نمودار بلوکی قطعات مختلف را از جدول ۱-۱۱ تهیه کنید.

تکمیل آزمایش ۱۷ (بخش ۱۷-۱۱)

نمودار بلوکی یک ضرب کننده دودویی 4 بیت در شکل ۲۴-۱۱ ملاحظه می‌شود. ضرب کننده به دو صورت توصیف می‌گردد: (۱) با عبارات سطح انتقالی ثباتی لیست شده در بخش (ب) شکل یا (۲) با استفاده از نمودار بلوکی در بخش (الف) آن. توصیف ضرب کننده برحسب قالب سطح انتقال ثباتی (RTL) در مثال ۵-۸ HDL (بخش ۵-۸) انجام گرفت. در این آزمایش از اجزاء مدار مجتمع مشخص شده در نمودار بلوکی استفاده خواهیم کرد تا توصیف ساختاری آزمایش ضرب کننده دودویی را بنویسیم. توصیف ساختاری با استفاده از توصیف مدول هر قطعه و سپس ذکر آنها برای نمایش ارتباط درونی آنها بدست می‌آید. (بخش ۵-۸ ملاحظه گردد). توصیف HDL قطعات را از آزمایش‌های قبلی هم می‌توان بدست آورد. 7483 در حل آزمایش ۷ (الف)، 7474 در آزمایش ۸ (الف)، 74161 با آزمایش ۱۰، 74194 با آزمایش ۱۴ و توصیف کنترل از حل آزمایش ۹ (پ) بدست می‌آید.

۱۲ سمبل‌های گرافیکی استاندارد

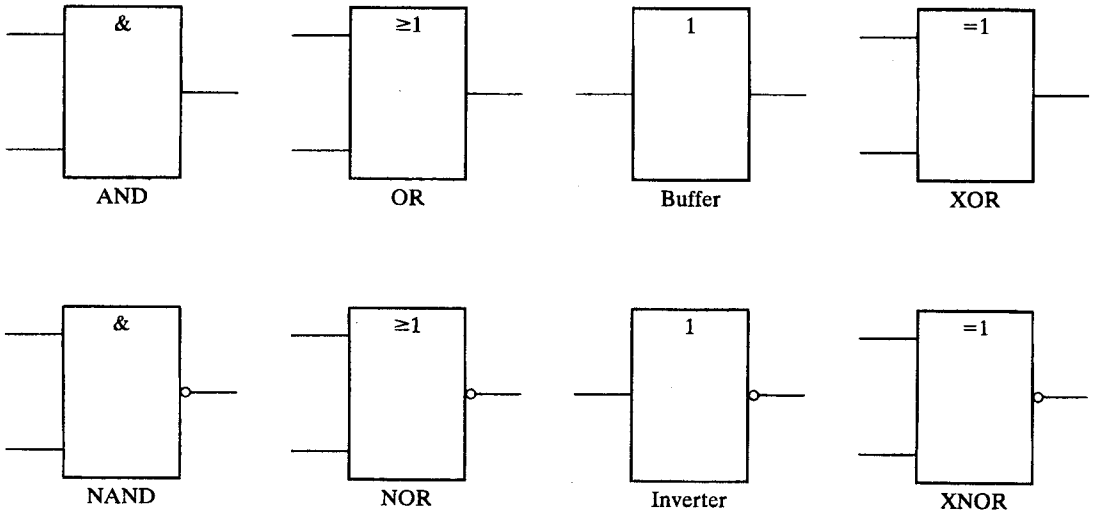
۱۲-۱ سمبل‌های مستطیلی شکل

قطعات دیجیتال مانند گیت‌ها، دیکدرها، مولتی پلکسرها و ثبات‌ها به صورت مدارهای مجتمع در بازار موجود بوده و به مدارات SSI و MSI دسته‌بندی می‌شوند. سمبل‌های گرافیکی استاندارد برای این قطعات و قطعات دیگر چنان تهیه شده است که کاربر با سمبل تخصیص یافته منحصر به فردش به راحتی آن را تشخیص می‌دهد. این استاندارد که ANSI/IEEE Std 91-1984 خوانده می‌شود به وسیله صنایع، دولت‌ها و سازمان‌های حرفه‌ای تأیید شده و سازگار با استانداردهای جهانی است.

این استاندارد، یک طرح مستطیلی شکل را برای نمایش هر تابع منطقی خاص به کار می‌برد. درداخل شکل، یک سمبل توصیفی کلی که بیانگر عمل قطعه است، وجود دارد. به عنوان مثال، سمبل توصیف کلی برای مولتی پلکسر MUX است. سایز طرح اختیاری است و می‌تواند مربع یا مستطیلی شکل با نسبت طول- عرض دلخواه باشد. خطوط ورودی در سمت چپ و خروجی‌ها در سمت راست قرار می‌گیرند. اگر جهت جریان سیگنال معکوس گردد، باید با فلش معین شود.

سمبل‌های مستطیلی برای گیت‌های SSI در شکل ۱-۱۲ نشان داده شده‌اند. سمبل مصوب برای گیت AND یک امپرسند (&) است. گیت OR سمبل مصوب بزرگتر یا مساوی 1 (1 ≥) را داراست، و به این معنی است که برای فعال بودن خروجی حداقل یک ورودی فعال لازم است. سمبل گیت بافر 1 است، و به این معنی است که در آن تنها 1 ورودی وجود دارد سمبل XOR این واقعیت را بیان می‌کند که برای فعال بودن خروجی، یک ورودی باید فعال باشد. لحاظ سمبل‌های نفی دایروی در خروجی، گیت‌ها را به مقادیر متمم تبدیل می‌نمایند. گرچه به گیت‌ها سمبل‌های مستطیلی شکل پیشنهاد شده است، ولی این استاندارد دیگر شکل‌های شاخص را هم، طبق شکل ۵-۲، برای گیت‌ها می‌پذیرد.

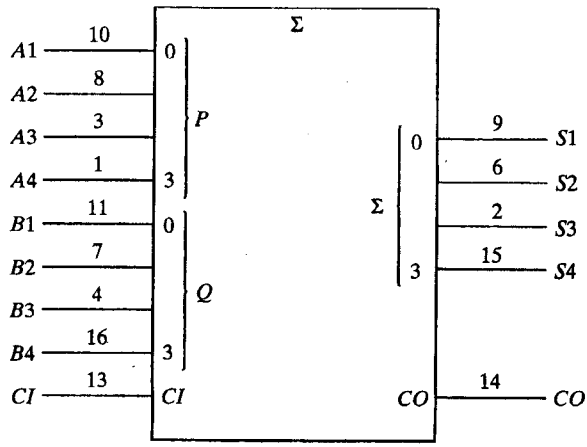
مثالی از یک سمبل گرافیک استاندارد MSI جمع‌کننده 4 بیت شکل ۲-۱۲ است. سمبل مصوب



شکل ۱-۱۲ . سمبل‌های گرافیکی مستطیلی گیت‌ها

برای یک جمع کننده حرف یونانی جمع، Σ است. حروف ارجح برای اعمال حسابی، P و Q هستند. سمبل‌های بیتی گروه‌بندی شده در ورودی و حاصل جمع خروجی مقادیر دهدهی معادل با وزن بیت‌ها با توانی از 2 است. بنابراین ورودی 3 مربوط به مقدار $8 = 2^3$ می‌باشد.

نقلی ورودی با CI و نقلی خروجی با CO مشخص شده است. اگر قطعات دیجیتال با طرح مدارهای تجاری نشان داده شوند، معمولاً پایه IC را در امتداد هر ورودی و خروجی می‌نویسند. بنابراین آی‌سی 7483 یک جمع کننده 4 بیت با نقلی پیش‌بینی شونده است. این آی‌سی در یک بسته 16 پایه محصور است شماره پایه‌ها برای 9 ورودی و پنج خروجی در شکل ۲-۱۲ دیده می‌شود. دو پایه دیگر مربوط به منبع تغذیه است.

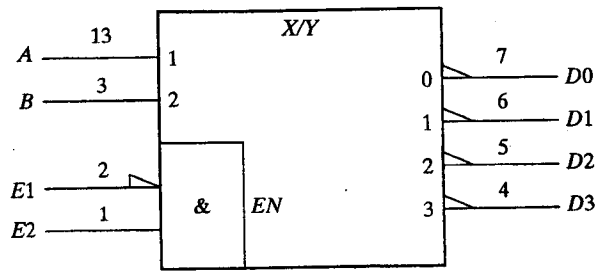


شکل ۲-۱۲ . سمبل گرافیکی استاندارد برای جمع کننده موازی چهار بیتی، آی سی 7483

قبل از معرفی سمبل‌های گرافیکی برای دیگر قطعات، لازم است بعضی از تکنولوژی‌ها مرور شوند. همانطور که در بخش ۷-۲ ذکر شد، یک سیستم منطقی مثبت، سطح مثبت‌تر سیگنال را به عنوان منطق 1 (با H نشان می‌دهد) و سطح منفی‌تر آن را با منطق 0، (L) نشان می‌دهد. قرارداد منطقی منفی سیگنال‌ها همه برحسب مقادیر H و L بیان می‌گردند. در هر نقطه از مدار، کاربر مجاز به تعریف منطق قبلی با تخصیص منطق 1 به هر یک از دو سیگنال H و L است. قرارداد منطق مختلط از یک علامت مثلث برای بیان یک قطبیت منطق منفی در هر ورودی یا خروجی است، شکل ۱۰-۲ (ج).

سازندگان مدارهای مجتمع عملکرد مدار را برحسب سیگنال‌های H و L بیان می‌کنند. وقتی که یک ورودی یا خروجی برحسب منطق مثبت بررسی گردد، به عنوان فعال بالا تعریف می‌شود. در منطق منفی، به صورت فعال پایین تعریف خواهد شد. در نمودارها ورودی‌ها یا خروجی‌های فعال پایین، با دایره کوچکی در محل پایانه نشان داده می‌شود. به هنگام استفاده از منطق مثبت در تمام سیستم، سمبل قطبیت منفی معادل با دایره کوچکی است که نفی را مشخص می‌کند. در این کتاب کلاً منطق مثبت به کار رفته است و هنگام رسم نمودار منطقی از دایره استفاده خواهد شد. اگر یک ورودی یا یک خروجی دایره نداشته باشد، آن را هنگامی فعال خواهیم خواند که در منطق 1 باشد. اگر یک ورودی یا یک خروجی دایره داشته باشد فعال بودن آن با منطق 0 است. با این وجود، ما سمبل قطبیت مثلثی را در همه نمودارهای استاندارد به کار خواهیم برد. این، با کتب داده‌های مدارات مجتمع که معمولاً در آنها سمبل قطبیت به کار می‌رود تطابق دارد. توجه کنید که چهارگیت پایین در شکل ۱-۱۲ می‌توانست با مثلث‌های کوچک به عوض دایره، در خطوط خروجی ترسیم گردد.

مثال دیگری برای سمبل‌های گرافیکی یک مدار MSI در شکل ۳-۱۲ ملاحظه می‌شود. این مدار یک دیکدر 2 به 4 است که نصف آی‌سی 74155 را نشان می‌دهد، ورودی‌ها در سمت چپ و خروجی‌ها در سمت راست هستند. سمبل نشانه X/Y به معنی مداری است که کد X را به کد Y تبدیل می‌نماید. به ورودی‌های A و B که وزن‌های 1 و 2 تخصیص داده شده و به ترتیب معادل 2^0 و 2^1 است. به خروجی‌ها اعداد 0 تا 3 تخصیص داده شده، که مربوط به خروجی‌های D_0 تا D_3 می‌باشد. دیکدر دارای یک وزن فعال پایین E_1 و یک ورودی فعال بالای E_2 است. این دو ورودی از یک گیت AND درونی می‌گذرند. خروجی گیت AND با EN (به معنی فعال) علامت خورده و وقتی E_1 در سطح پایین و E_2 در سطح بالا باشد، مدار فعال خواهد شد.



شکل ۳-۱۲ . سمبل گرافیکی استاندارد برای یک دیکدر 2 به 4 (نیمی از آی سی 74155)

۱۲-۲ سمبل‌های مصوب

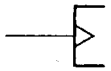
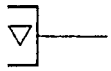
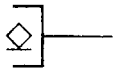
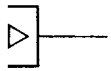
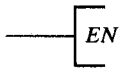
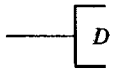
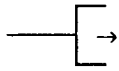
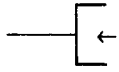
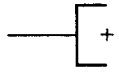
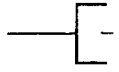
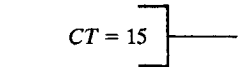
سمبل‌های گرافیکی استاندارد IEEE برای توابع منطقی، لیستی از سمبل‌های مصوبی را ارائه داده‌اند که به همراه طرح به کار می‌روند. یک سمبل مصوب برای نمایش مشخصه کلی یک قطعه یا مشخصات فیزیکی یک ورودی یا خروجی به طرح اولیه اضافه می‌گردد. جدول ۱-۱۲ بعضی از سمبل‌های مصوب استاندارد را لیست نموده است. یک سمبل مصوب جنرال، تابعی که به وسیله قطعه در نمودار اجرا می‌شود را تعریف می‌کند. این سمبل در وسط بالای طرح مستطیل شکل قرار می‌گیرد. سمبل‌های مصوب جنرال برای گیت، دیکدر و جمع کننده در نمودارهای قبلی نشان داده شدند. دیگر سمبل‌ها خود تشریح‌اند و بعد در نمایش عناصر دیجیتال مربوطه به کار خواهند رفت.

بعضی از سمبل‌های مصوب مربوط به ورودی‌ها و خروجی‌ها در شکل ۴-۱۲ نشان داده شده‌اند. سمبل‌های مربوط به ورودی‌ها در سمت چپ ستون سمبل قرار دارند. سمبل‌های مربوط به خروجی در سمت راست ستون واقعند. ورودی یا خروجی فعال پایین همان نشانگر قطبیت (پلاریته) است. همانطور که قبلاً ذکر شد، به هنگام فرض منطق مثبت، این علامت به معنی منفی منطقی است. در ورودی پالس ساعت فلیپ فلاپ از علامت ورودی دینامیکی استفاده شده است. این علامت بیانگر این حقیقت است که ورودی در یک گذر پایین به بالای سیگنال فعال می‌شود. وقتی که مدار فعال شود، خروجی در حالت منطقی معمولی 0 یا 1 است، ولی وقتی که مدار غیرفعال شود، خروجی سه حالته در حالت امپدانس بالاست. این حالت معادل با یک مدار باز است.

خروجی کلکتور باز دارای حالتی است که در آن حالت امپدانس بالا را به نمایش می‌گذارد. گاهی

جدول ۱-۱۲ . سمبل‌های توصیفی عمومی

توصیف	سمبل
گیت یا تابع AND	&
گیت یا تابع OR	≥1
گیت بافر یا معکوس کننده	1
گیت یا تابع OR انحصاری	=1
تابع زوج یا عنصر توازن زوج	2k
تابع فرد یا عنصر توازن فرد	2k + 1
کدر، دیکدر یا مبدل کد	X/Y
مولتی پلکسر	MUX
دی مولتی پلکسر	DMUX
جمع کننده	Σ
ضرب کننده	Π
مقایسه گر مقدار	COMP
واحد ریاضی منطقی	ALU
شیفت رجیستر	SRG
شمارنده	CTR
شمارنده موج گونه	RCTR
حافظه فقط خواندنی	ROM
حافظه با دستیابی تصادفی	RAM

توصیف	سمبل
ورودی یا خروجی فعال پایین	
ورودی یا خروجی منفی شده	
نشانه ورودی دینامیکی	
خروجی سه حالته شکل (۱۶-۱۰)	
خروجی کلکتور باز، شکل (۱۲-۱۰)	
خروجی با تقویت خاص	
ورودی توانا ساز	
ورودی داده به عنصر ذخیره کننده	
ورودی‌های فلیپ فلاپ	
شیفت به راست	
شیفت به چپ	
شمارش بالا	
شمارش پایین	
محتویات ثبات برابر 15 دودویی است	

شکل ۴-۱۲ . سمبل‌های توصیفی مرتبط با ورودی‌ها و خروجی‌ها

اوقات لازم است برای تولید یک سطح منطقی صحیح یک مقاومت بیرونی به آن متصل شود. سمبل لوزی شکل که گاهی در بالا (برای انواع بالا) و یا در پایین (برای نوع پایین) خط بار دارد، برای این نوع

مدارها به کار می‌رود. نوع بالا و یا پایین سطحی را مشخص می‌کند که خروجی در آن هنگام در حالت امپدانس بالا نیست. مثلاً، مدارهای مجتمع نوع TTL خروجی‌های خاصی به نام خروجی‌های کلکتور باز دارند. این خروجی‌ها با علامت لوزی و یک خط در زیر آن مشخص می‌شوند. سمبل به این معنی است که خروجی می‌تواند در یکی از دو حالت امپدانس بالا و یا سطح منطقی پایین قرار داشته باشد.

وقتی که به عنوان تابع توزیع به کار رود، اتصال دو یا چند تابع NAND کلکتور باز به یک مقاومت مشترک یک تابع AND با منطق مثبت یا یک تابع OR با منطق منفی را اجرا خواهد کرد.

خروجی‌ها که به میزان خاصی تقویت شده باشند در راه‌اندازی گیت‌هایی به کار می‌روند که قرار است قابلیت راه‌اندازی خاصی را دارا باشند. این گیت‌ها در عناصری مانند راه‌اندازی‌های ساعت و یا در فرستنده‌های مبتنی بر گذرگاه استفاده می‌شوند. سمبل EN یک ورودی فعال‌ساز را بیانگر است و هرگاه این ورودی فعال شود همه خروجی‌ها فعال خواهند شد. هرگاه EN غیرفعال باشد، تمام خروجی‌ها غیرفعال می‌گردند. سمبل‌های ورودی فلیپ فلاپ‌ها هم دارای مفهوم می‌باشند. ورودی D به عناصر ذخیره‌ساز دیگری مانند ورودی حافظه هم متعلق است.

سمبل جابجایی به راست و چپ به ترتیب فلش‌هایی به راست و چپ می‌باشند. سمبل‌های شمارش بالا و شمارش پایین، به ترتیب سمبل‌های بعلاوه و منها هستند. یک خروجی که با $CT = 15$ مشخص شده هنگامی فعال می‌گردد که محتوای ثبات به عدد 15 برسد. وقتی که اطلاعات غیراستاندارد در داخل طرح نشان داده شود در داخل کروشه از عبارات [مانند این] استفاده می‌گردد.

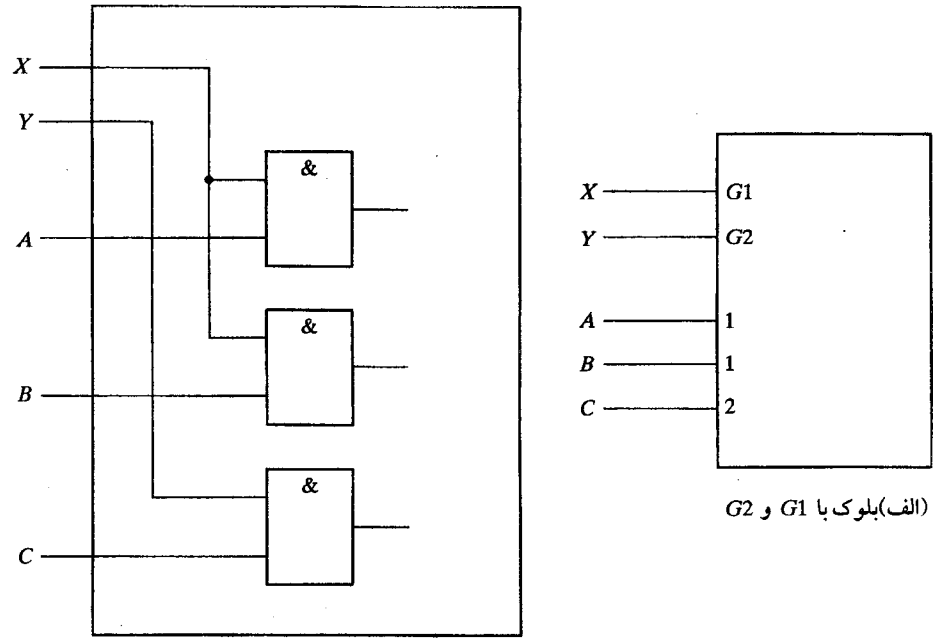
۱۲-۳ نشانه وابستگی

مهمترین مفهوم سمبل‌های منطقی استاندارد، نشانه وابستگی یا همبستگی آنهاست. نشانه وابستگی برای تهیه مفهوم ارتباط بین ورودی‌ها و خروجی‌های مختلف بدون ذکر اتصالات داخلی آنها به کار می‌رود. ما ابتدا نشانه وابستگی را با یک مثال از وابستگی AND نشان می‌دهیم و سپس همه سمبل‌های دیگر مربوط به این نشانه تعریف خواهند شد.

وابستگی AND با حرف G و به دنبال آن یک عدد نمایش داده می‌شود. هر ورودی یا خروجی در نمودار که با عدد مربوط به G علامت بخورد به معنی AND شدن با آن است. مثلاً، اگر یک ورودی در نمودار برچسب $G1$ و ورودی دیگری هم با عدد 1 برچسب بخورد، آنگاه دو ورودی $G1$ و 1 از نظر درونی AND تصور می‌شود.

مثالی از وابستگی AND در شکل ۵-۱۲ آمده است. در شکل (الف) ما بخشی از یک سمبل گرافیکی با دو برچسب وابستگی AND، یعنی $G1$ و $G2$ را داریم. دو ورودی با عدد 1 و یک ورودی با عدد 2 برچسب خورده‌اند. تفسیر معادل در بخش (ب) شکل مذکور آمده است. ورودی X مربوط به G_1 با ورودی‌های A و B که با شماره 1 برچسب خورده‌اند، AND شده‌اند. به طور مشابه ورودی Y با ورودی C، AND شده تا وابستگی بین $G2$ و 2 تحقق یابد.

استاندارد، 10 وابستگی دیگر را هم تعریف می‌کند. هر وابستگی به وسیله یک سمبل حرفی (به جز



(ب) تفسیر معادل

(الف)بلوک با $G1$ و $G2$

شکل ۵-۱۲ . مثالی از وابستگی G (AND)

EN) معین می‌گردد. حرف در ورودی یا خروجی نوشته شده و به دنبال آن یک عدد ذکر می‌گردد. هر ورودی یا خروجی که با آن وابستگی تحت تأثیر قرار گیرد با همان شماره برچسب می‌خورد. یازده وابستگی و حروف مربوط به آنها به قرار زیرند.

G	بیانگر یک رابطه AND است.
V	یک رابطه OR را نشان می‌دهد.
N	رابطه نفی (XOR) را بیانگر است.
EN	یک عمل فعال‌سازی را مشخص می‌نماید.
C	یک وابستگی کنترل را مشخص می‌کند.
S	عمل 1 شدن را بیان می‌کند.
R	عمل 0 شدن را مشخص می‌نماید.
M	یک وابستگی حالت را نشان می‌دهد.
A	وابستگی آدرس را بیان می‌کند.
Z	اتصالات درونی را بیان می‌دارد.
X	انتقال کنترل شده را مشخص می‌کند.

وابستگی‌های V و N برای بیان روابط بولی OR و XOR، مشابه با G برای AND، به کار می‌روند. وابستگی EN مشابه سمبل مصوب EN است با این تفاوت که یک عدد به دنبال آن می‌آید (مثلاً $EN2$). اگر EN فعال شود تنها خروجی مرتبط با آن فعال می‌گردد.

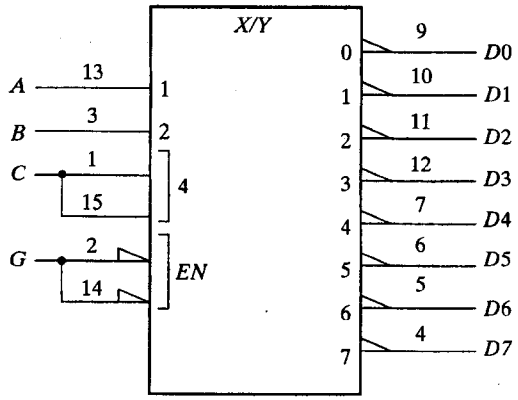
وابستگی کنترل C برای تشخیص یک ورودی ساعت در مدار ترتیبی به کار می‌رود و مشخص کننده ورودی کنترل شده با آن است. وابستگی‌های نشاندن S و بازنشانی R برای مشخص کردن حالات منطقی درونی فلیپ فلاپ SR استفاده می‌شوند. وابستگی‌های C، S و R در بخش ۵-۱۲ همراه با مدار فلیپ فلاپ توضیح داده شده‌اند. وابستگی M برای شناسایی ورودی‌هایی که حالت عملیات واحد را انتخاب می‌کنند به کار می‌رود. وابستگی حالت در بخش ۶-۱۲ به همراه ثبات‌ها و شمارنده‌ها ارائه شده است. وابستگی آدرس برای شناسایی ورودی آدرس یک حافظه است. بحث مربوط به آن در بخش ۸-۱۲ همراه با واحد حافظه آمده است.

وابستگی Z برای شناسایی اتصالات درونی قطعه است. این وابستگی وجود اتصالات درونی منطقی بین ورودی‌ها، خروجی‌ها، ورودی‌های داخلی و خروجی‌های داخلی در هر ترکیب را مشخص می‌نماید. وابستگی X برای مشخص کردن مسیر انتقال کنترل شده در یک گیت انتقال مورد استفاده قرار می‌گیرد.

۱۲-۴ سمبل‌های عناصر ترکیبی

مثال‌های این بخش و بقیه فصول کاربرد استاندارد در نمایش انواع عناصر دیجیتال با سمبل‌های گرافیکی را تشریح می‌نماید. مثال‌ها مدارهای مجتمع تجاری واقعی را با شماره پایه‌ها در ورودی‌ها و خروجی شامل می‌شود. اغلب آی‌سی‌های معرفی شده در این فصل شامل عناصری هستند که در آزمایشات فصل ۱۱ مطرح شدند.

سمبل‌های گرافیکی برای جمع کننده دیکدر در بخش ۲-۱۲ نشان داده شدند. آی‌سی 74155 مطابق شکل ۶-۱۲ قابل اتصال به صورت یک دیکدر 8 × 3 است. (جدول درستی این دیکدر در شکل ۷-۱۱ نشان داده شد.) در این آی‌سی دو عدد ورودی C و دود عدد ورودی G وجود دارد. مطابق نمودار، هر



شکل ۶-۱۲ . آی سی 74155 بعنوان یک دیکدر 3×8

جفت باید به یکدیگر وصل شوند. ورودی فعال‌ساز در حالت سطح پایین فعال می‌گردد. خروجی‌ها همگی فعال پایین هستند. به ورودی‌ها وزن 1، 2 و 4 به ترتیب معادل با 2^0، 2^1 و 2^2 تخصیص یافته است. به خروجی‌ها شماره‌های 0 تا 7 اختصاص داده شده است. جمع اوزان ورودی‌ها خروجی فعال را مشخص می‌سازد. بنابراین اگر دو خط ورودی با اوزان 1 و 4 فعال باشند وزن کل $5 = 4 + 1$ شده و خروجی 5 فعال می‌گردد. البته ورودی *EN* برای هر خروجی باید فعال باشد.

دیکدر نوع خاصی از یک مدار به نام کدر (کد ساز) است. یک کدر قطعه‌ای است که یک کد دودویی را از طریق تعدادی ورودی دریافت نموده و کد دودویی متفاوتی را در تعدادی خروجی تولید می‌نماید. به جای استفاده از سمبل مصوب *X/Y*، کدر را می‌توان با نام کد مشخص کرد. مثلاً، یک دیکدر 8×3 در شکل ۶-۱۲ را می‌توان با نام *BIN/OCT* نام‌گذاری کرد چون مدار یک عدد دودویی 3 بیت را به یک عدد مبنای 8 از 0 تا 7 تبدیل می‌کند.

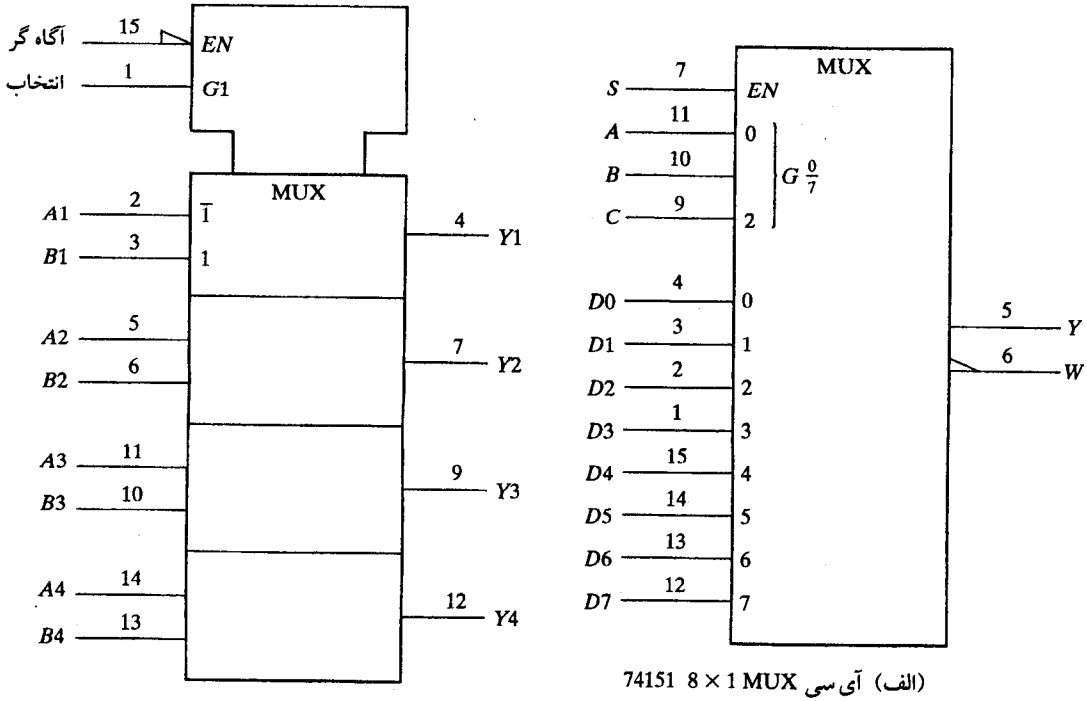
قبل از نمایش سمبل گرافیکی مولتی پلکسر، لازم است تا تغییری از وابستگی AND را نشان دهیم. وابستگی AND گاهی به صورت خلاصه‌تر $G\frac{0}{7}$ نشان داده می‌شود. این سمبل به معنی هشت سمبل وابستگی از 0 تا 7 مطابق زیر است.

$$G0, G1, G2, G3, G4, G5, G6, G7$$

در هر لحظه از زمان، فقط یکی از هشت گیت AND می‌تواند فعال باشد. گیت AND فعال از ورودی‌های مربوط به سمبل G مشخص می‌شود. این ورودی‌ها با اوزانی از توان 2 علامت زده شده‌اند. برای هشت گیت AND فوق، اوزان 0، 1 و 2 به ترتیب متعلق به 2^0، 2^1 و 2^2 می‌باشد. گیت AND فعال در هر لحظه از جمع اوزان ورودی‌های فعال بدست می‌آید. بنابراین اگر ورودی‌های 0 و 2 فعال باشند، گیت AND فعال برابر است با $2^0 + 2^2 = 5$. در نتیجه $G5$ فعال و هفت گیت AND دیگر غیرفعال خواهند بود.

سمبل گرافیکی استاندارد برای یک مولتی پلکسر 1×8 در شکل ۷-۱۲ (الف) نشان داده شده است. سمبل درون بلوک‌ها بخشی از قرارداد استاندارد است، ولی سمبل‌های بیرونی، سمبل‌های تعریف شده به وسیله کاربرند. جدول تابع آی‌سی 741551 در شکل ۹-۱۱ ارائه شد. وابستگی AND با $G\frac{0}{7}$ علامت خورده است. در واقع آنها همان ورودی‌های انتخاب‌اند. هشت ورودی داده با اعداد 0 تا 7 مشخص شده‌اند. وصل کل ورودی‌های فعال مربوط به سمبل G عدد موجود در روی خط داده فعال را معین می‌نماید. مثلاً اگر ورودی‌های انتخا[illegible] $CBA = 1$[illegible] باشد، آنگاه ورودی 1 و 2 مربوط به G فعالند. این نیز مقدار عدد $2^2 + 2^1 = 6$ را برای وابستگی AND بدست می‌دهد، که در نتیجه $G6$ فعال خواهد شد. چون $G6$ ورودی داده شماره 6 با یکدیگر AND شده‌اند، این ورودی را فعال خواهد کرد. بنابراین، به شرطی که ورودی فعال‌ساز، فعال باشد، خروجی با ورودی داده D_6 برابر خواهد بود.

شکل ۷-۱۲ (ب) آی‌سی مولتی پلکسر 1×2 نوع 74157 را نشان می‌دهد که جدول تابع آن در شکل ۱۷-۱۱ مشاهده شد. ورودی‌های انتخاب و فعال‌ساز برای هر چهار مولتی پلکسر مشترک‌اند. این اشتراک در نشان‌گذاری استاندارد با چهار گوشه فرو برده شده در قسمت بالای نمودار دیده می‌شود که



(ب) آی سی 74157 ، چهارتایی 2 × 1 MUX

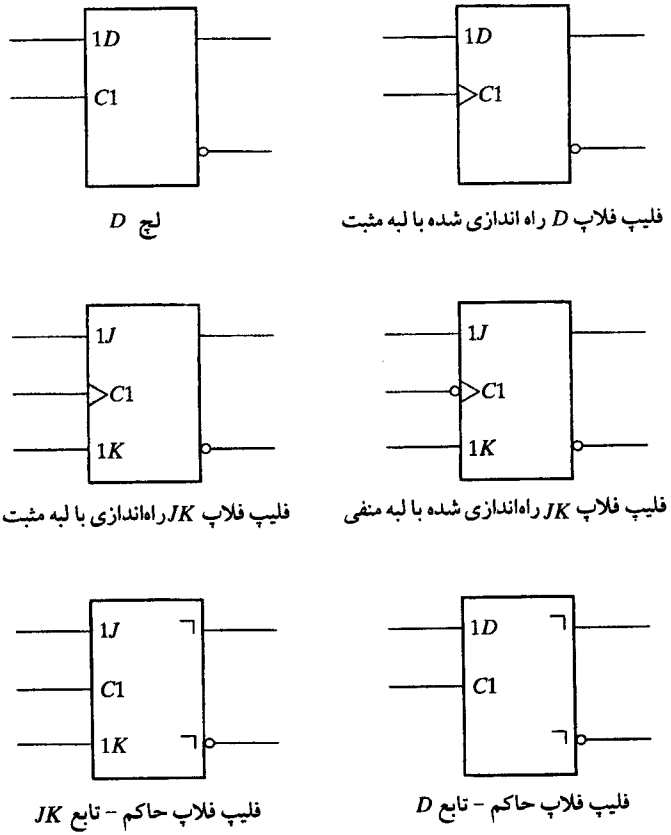
(الف) آی سی 74151 8 × 1 MUX

شکل ۷-۱۲ . سمبل‌های گرافیکی برای مولتی پلکسرها

بیانگر یک بلوک کنترل مشترک است. ورودی‌ها به بلوک کنترل مشترک کلیه قسمت‌های پایین نمودار را کنترل می‌نمایند. وقتی که ورودی فعال‌ساز مشترک EN در سطح پایین باشد، فعال خواهد بود. وابستگی AND مربوط به $G1$ معین کننده ورودی فعال در هر بخش از مولتی پلکسر است. وقتی $G1 = 0$ باشد، ورودی‌های A که با $\overline{1}$ علامت زده شده‌اند فعال می‌گردند. وقتی $G1 = 0$ است ورودی‌های B که با 1 نشانگذاری شده‌اند فعال خواهند بود. اگر EN فعال باشد ورودی‌های فعال نیز به خروجی‌های مربوطه خود متصل می‌شوند. توجه کنید که سمبل‌های $\overline{1}$ و 1 در عوض تکرار در هر بخش فقط در قسمت فوقانی نوشته شده‌اند.

۱۲-۵ سمبل فلیپ فلاپ‌ها

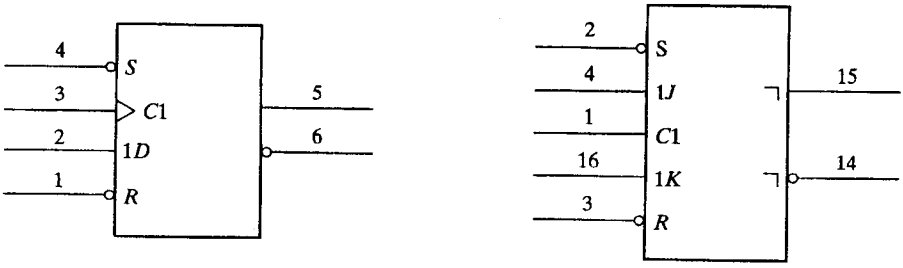
سمبل‌های گرافیک برای انواع متفاوت فلیپ فلاپ‌ها در شکل ۸-۱۲ دیده می‌شود. یک فلیپ فلاپ با یک بلوک مستطیل شکل، ورودی‌ها در سمت چپ و خروجی‌هایش در سمت راست نشان داده می‌شود. یکی از خروجی‌ها حالت معمولی فلیپ فلاپ و دیگری به همراه دایره کوچک منفی، متمم آن را نشان می‌دهد. سمبل‌های گرافیکی انواع فلیپ فلاپ‌ها را از یکدیگر جدا می‌کند: مثلاً ساختمان داخلی



شکل ۸-۱۲ ، سمبل گرافیکی استاندارد برای فلیپ فلاپ

لچ D در شکل ۵-۶ فلیپ فلاپ حاکم تابع در شکل ۹-۶؛ و فلیپ فلاپ حساس به لبه در شکل ۱۲-۶ دیده شدند. سمبل گرافیکی برای لچ D یا فلیپ فلاپ D دارای ورودی‌های D و C در داخل بلوک است. سمبل گرافیکی برای فلیپ فلاپ JK دارای ورودی‌های J و K و C در داخل است. علائم $C1$ ، $1D$ ، $1J$ و $1K$ مثال‌هایی از وابستگی کنترل است. ورودی $C1$ ، ورودی $1D$ را در فلیپ فلاپ D و ورودی‌های $1J$ و $1K$ را در فلیپ فلاپ JK کنترل می‌نماید.

لچ D در کنار $1D$ و $C1$ سمبل دیگری ندارد. فلیپ فلاپ حساس به لبه دارای سمبل فلش مانندی در جلو وابستگی کنترل $C1$ است تا یک ورودی دینامیک را معرفی نماید. سمبل شاخص دینامیکی بیانگر این نکته است که فلیپ فلاپ به یک گذر مثبت پالس‌های ساعت پاسخ می‌دهد. وجود دایره کوچک در بیرون بلوک و در امتداد شاخص دینامیک بیانگر گذر لبه منفی برای فلیپ فلاپ است. یک فلیپ فلاپ حاکم- تابع، حساس به پالس خوانده می‌شود و با یک L وارونه (⌐) در جلو خروجی‌ها نشان داده می‌شود. این بدان معنی است که خروجی سیگنال در لبه پایین رونده پالس تغییر حالت می‌دهد. توجه



(الف) نیمی از فلیپ فلاپ 7476 JK (ب) نیمی از فلیپ فلاپ 7474 D

شکل ۹-۱۲ . آی سی‌های فلیپ فلاپ

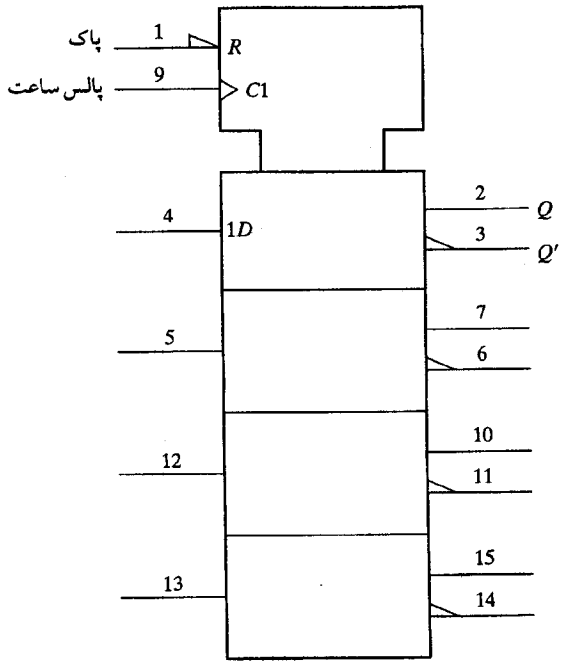
کنید که فلیپ فلاپ حاکم- تابع بدون شاخص دینامیکی ترسیم شده است.

فلیپ فلاپ‌های موجود در بسته‌های مدار مجتمع دارای ورودی‌های خاص برای نشاندن یا بازنشانی غیرهمزمانند. این ورودی‌ها را معمولاً ورودی‌های 1 شدن مستقیم و 0 شدن مستقیم می‌خوانند. آنها خروجی را بدون نیاز به پالس ساعت و در لبه منفی سیگنال تحت تأثیر قرار می‌دهند. سمبل گرافیکی فلیپ فلاپ حاکم- تابع نوع JK با ورودی‌های فوق‌الذکر در شکل ۹-۱۲ (الف) مشاهده می‌شود. علائم $C1$، $1J$ و $1K$ وابستگی کنترل را نشان می‌دهد، و به این معنی است که $C1$ ورودی‌های $1J$ و $1K$ را کنترل می‌کند. ورودی‌های S و R دارای دایره کوچکی در طول خطوط ورودی‌اند که این نشان دهنده فعال شدن آنها در سطح منطق 0 است. جدول تابع برای فلیپ فلاپ 7476 در شکل ۱۲-۱۱ ملاحظه می‌گردد.

سمبل گرافیکی یک فلیپ فلاپ D حساس به لبه مثبت با نشاندن و بازنشانی مستقیم در شکل ۹-۱۲ (ب) ملاحظه می‌شود. گذر مثبت ساعت در ورودی $C1$ ورودی $1D$ را کنترل می‌نماید. ورودی‌های S و R مستقل از ساعتند. این فلیپ فلاپ همان آی‌سی 7474 است که جدول تابع شکل ۱۳-۱۱ آورده شد.

۱۲-۶ سمبل ثبات‌ها

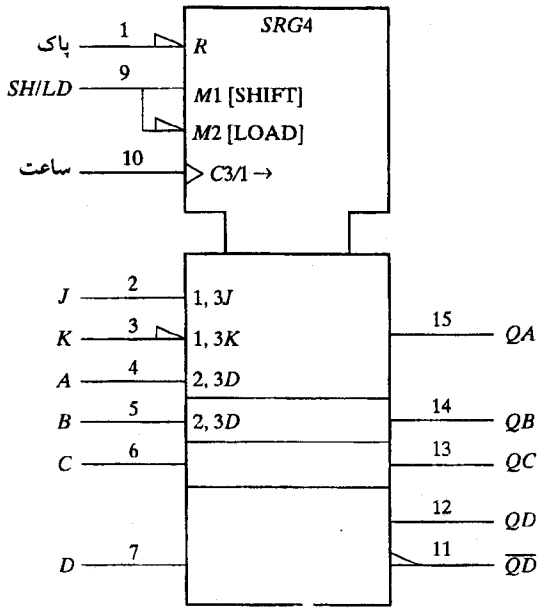
سمبل گرافیکی استاندارد برای یک ثبات معادل سمبل گروهی از فلیپ فلاپ‌ها با یک ورودی ساعت مشترک است. شکل ۱۰-۱۲ سمبل گرافیکی استاندارد برای آی‌سی 74175، شامل چهار فلیپ فلاپ D با ساعت و ورودی‌های پاک کردن مشترک، را نشان می‌دهد. ورودی‌های کنترل مشترک بلوک به هر عنصر در بخش‌های پایینی نمودار وصل است. $C1$ وابستگی کنترلی است که همه ورودی‌های $1D$ را کنترل می‌کند. بنابراین فلیپ فلاپ به وسیله یک ورودی ساعت مشترک تریگر می‌شود. سمبل ورودی دینامیکی مربوط به $C1$ بیانگر تریگر شدن فلیپ فلاپ در لبه مثبت پالس ساعت ورودی است. ورودی مشترک، وقتی در سطح پایین باشد، همه فلیپ فلاپ‌ها را پاک یا بازنشانی می‌کند. سمبل $1D$ به جای آن که در هر بخش تکرار شود فقط یک بار در بخش بالا قرار داده شده است. خروجی‌های متمم فلیپ



شکل ۱۰-۱۲ . سمبل گرافیکی برای ثبات چهار بیت، آی سی 74175

فلاپ‌ها در این نمودار با سمبل قطبیت در عوض نفی مشخص شده‌اند.

سمبل گرافیکی استاندارد برای یک شیفت رجیستر با بار شدن موازی در شکل ۱۱-۱۲ ملاحظه می‌شود. این سمبل یک آی‌سی 74195 است که جدول تابع آن در شکل ۱۶-۱۱ ملاحظه شد. سمبل

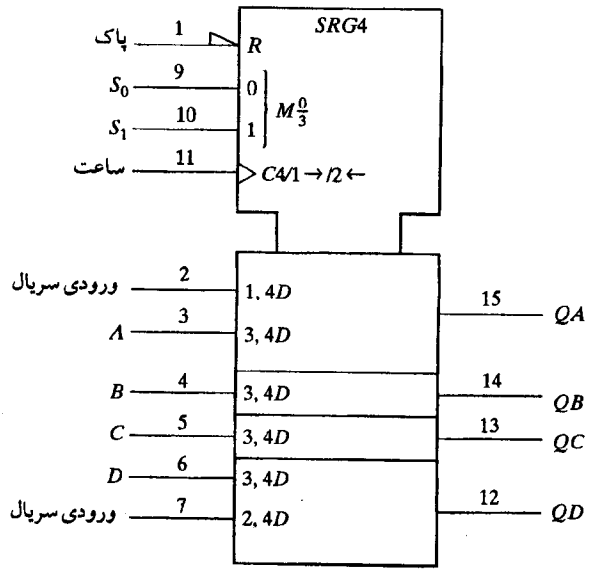


شکل ۱۱-۱۲ . سمبل گرافیکی یک شیفت رجیستر با بار کننده موازی، آی سی 74195

مصوب برای یک شیفت رجیستر SRG و به دنبال آن عددی است که تعداد طبقات آن را مشخص می‌کند. بنابراین $SRG4$ یک شیفت رجیستر 4 بیتی است. بلوک کنترل مشترک دارای دو حالت وابستگی $M1$ و $M2$ برای اعمال جابجایی و بارکردن است. توجه کنید که این آی‌سی دارای یک ورودی با برچسب SH/LD (بارکردن/ جابجایی) است که به دو بخش تقسیم شده تا دو حالت را نشان دهد. وقتی که ورودی SH/LD در سطح بالا باشد $M1$ فعال و وقتی SH/LD پایین باشد، $M2$ فعال خواهد بود. به این تخصیص سمبل توجه کنید: مشخص است که در واقع یک ورودی در پایه 9 وجود دارد ولی این ورودی به دو بخش تقسیم شده تا دو عدد $M1$ و $M2$ به آن اختصاص یابد. وابستگی کنترل $C3$ برای ورودی ساعت است. سمبل دینامیکی در برابر ورودی $C3$ به این معنی است که فلیپ فلاپ‌ها در لبه مثبت ساعت تریگر می‌شوند. سمبل →1/ که بعد از $C3$ آمده به این معنی است که عمل جابجایی به راست یا به سمت مقادیر پایین شکل با $M1$ فعال می‌شود.

چهار بخش زیر بلوک کنترل مشترک چهار فلیپ فلاپ را نشان می‌دهد. فلیپ فلاپ QA سه ورودی دارد: دو ورودی مربوط به عمل سریال (شیفت) و یکی متعلق به عمل موازی (بار شدن) است. ورودی سریال $1, 3J$ به معنی فعال بودن ورودی J از فلیپ فلاپ QA به هنگام فعال بودن $N1$ (جابجایی) و گذر مثبت ساعت در ورودی $C3$ است. ورودی سریال دیگر $3K$, 1 دارای سمبل قطبیت در ورودی K از فلیپ فلاپ JK می‌باشد. سومین ورودی QA و ورودی دیگر فلیپ فلاپ‌ها برای ورود موازی داده‌ها هستند. هر ورودی با $2, 3D$ مشخص شده است. عدد 2 برای $M2$ و عدد 3 برای ساعت $C3$ است. اگر ورودی در پایه شماره 9 در سطح پایین باشد، $M1$ فعال است و در این حال گذر مثبت ساعت در $C3$ یک انتقال موازی را از چهار ورودی A تا D به داخل فلیپ فلاپ‌ها $QD - QA$ موجب می‌شود. توجه دارید که بار شدن موازی فقط در بخش‌های اول و دوم مشخص شده است. فرض بر این است که در دو بخش پایین‌تر هم به همین منوال باشد.

شکل ۱۲-۱۲ سمبل گرافیکی را برای شیفت رجیستر دو طرفه با بار شدن موازی 74194 نشان



شکل ۱۲-۱۲ . سمبل گرافیکی برای یک شیفت رجیستر دو طرفه با بار شدن موازی، آی سی 74194

می‌دهد. جدول تابع برای این آی‌سی در شکل ۱۹-۱۱ لیست شده است. بلوک کنترل مشترک یک ورودی R را برای بازنشانی همه فلیپ فلاپ‌ها نشان می‌دهد. انتخاب حالت دارای دو ورودی است. حالت وابستگی M ممکن است مقادیر دودویی 0 تا 3 را بپذیرد. این قابلیت با $M\frac{0}{3}$ نشان داده شده است و به معنی $M0$، $M1$، $M2$ و $M3$ بوده و مشابه علامت‌گذاری برای وابستگی G در مولتی پلکسرها می‌باشد. سمبل مربوط به ساعت

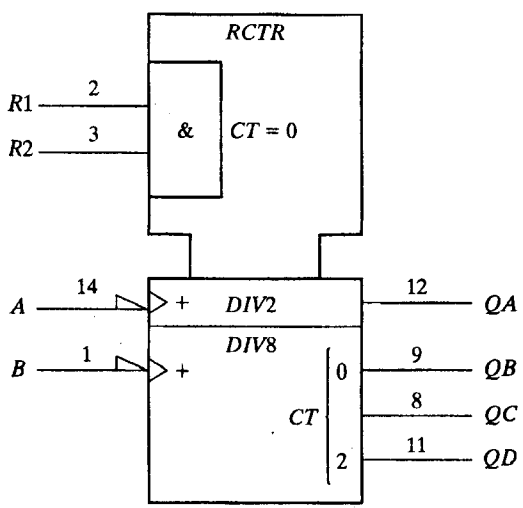
$$C4/1 \rightarrow \quad /2 \leftarrow$$

می‌باشد که $C4$ وابستگی کنترل برای ساعت است. سمبل → /1 به این معنی است که در حالت $M1$، ثبات به راست جابجا می‌شود (در این حالت رو به پایین). سمبل ← /2 به معنی جابجایی به چپ است (رو به بالا) و در این حالت مجموعه در حالت $M2$ ($S_1S_0 = 10$) قرار دارد. جهات چپ و راست را با چرخاندن صفحه به اندازه 90 درجه در خلاف عقربه‌های ساعت می‌توانید ملاحظه کنید.

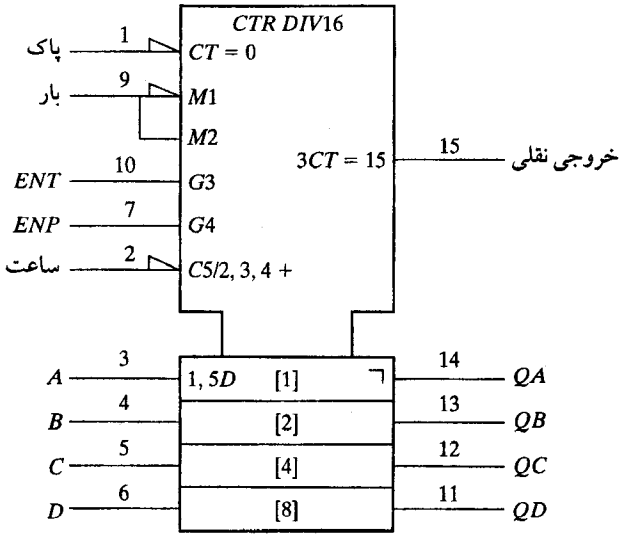
بخش‌های زیرین بلوک کنترل چهار فلیپ فلاپ را نشان می‌دهد. اولین فلیپ فلاپ دارای ورودی سریال به راست است که با $4D$, 1 ($M1$، ساعت $C4$، ورودی D) مشخص شده است. آخرین فلیپ فلاپ ورودی سریال به چپ را دارد که با $4D$, 2 ($M2$، ساعت $C4$، ورودی D) معرفی شده است. هر چهار فلیپ فلاپ دارای ورودی موازی‌اند که با $4D$, 3 نشان داده شده است (حالت $M3$، ساعت $C4$، ورودی D). بنابراین $M3$ ($S_1S_0 = 11$) برای بار کردن موازی است، حالت باقیمانده ($S_1S_0 = 00$) $M0$ تأثیری روی خروجی‌ها ندارد زیرا در برچسب ورودی‌ها لحاظ نشده است.

۱۲-۷ سمبل شمارنده‌ها

سمبل گرافیکی استاندارد برای یک شمارنده دودویی موج‌گونه در شکل ۱۳-۱۲ دیده می‌شود. سمبل مصوب برای یک شمارنده موج‌گونه $RCTR$ است. $DIV2$ به معنی مدار تقسیم بر 2 است و از



شکل ۱۳-۱۲. سمبل گرافیکی برای شمارنده موج‌گونه، آی سی 7493



شکل ۱۴-۱۲ . سمبل گرافیکی برای شمارنده دودویی چهار بیت با بار شدن موازی، آی سی 74161

یک فلیپ فلاپ QA بدست می‌آید. $DIV8$ به معنی شمارنده تقسیم بر 8 می‌باشد که از سه فلیپ فلاپ دیگر حاصل می‌گردد. نمودار، آی‌سی 7493 را نشان می‌دهد و مدار داخلی‌اش در شکل ۲-۱۱ مشاهده شد. بلوک کنترل مشترک یک AND درونی با ورودی‌های $R1$ و $R2$ دارد. وقتی که هر دو ورودی برابر 1 باشند محتوای شمارنده به سمت 0 می‌رود. این نکته با $CT = 0$ مشخص شده است. چون ورودی مورد شمارش به ورودی‌های ساعت همه فلیپ فلاپ‌ها نمی‌رود، دارای برچسب $C1$ نیست، و در عوض سمبل + برای شمارش رو به بالا به کار رفته است. سمبل دینامیک مجاور + همراه با سمبل قطبیت در امتداد خط ورودی به این معنی است که شمارش با گذر منفی سیگنال ورودی تحت تأثیر قرار می‌گیرد. گروه‌بندی 0 تا 2 در خروجی، ارزش توانی از 2 را مشخص می‌نماید. بنابراین 0 به معنی $2^0=1$ و 2 نمایشگر $2^2 = 4$ است.

سمبل گرافیکی استاندارد برای شمارنده 4 بیت با بار شدن موازی، آی‌سی 74161، در شکل ۱۴-۱۲ دیده می‌شود. سمبل مصوب برای شمارنده همزمان CTR و به دنبال آن $DIV16$ (تقسیم بر 16) است که طول سیکل شمارنده را بیان می‌نماید. یک ورودی بارکردن در پایه 9 دیده می‌شود که به دو حالت $M1$ و $M2$ تقسیم شده است. $M1$ هنگامی فعال است که ورودی پایه 9 در سطح پایین و $M2$ وقتی که ورودی بارکردن در پایه 9 بالاست، فعال می‌گردد. فعال بودن $M1$ در سطح پایین از شاخص پلاریته در طول خط ورودی مشخص است. ورودی‌های فعال‌ساز از وابستگی G استفاده می‌نمایند. $G3$ مربوط به ورودی فعال‌ساز T و $G4$ به فعال‌ساز P مربوط‌اند. برچسب مربوط به ساعت برابر زیر است

$$C5 / 2, 3, 4 +$$

و به این معنی است که مدار وقتی $M2$، $G3$ و $G4$ فعالند رو به بالا می‌شمارد (بار شدن = 1،

$ENT = 1$ و $ENP = 1$) و ساعت در $C5$ واردگذر منفی می‌گردد. این حالت در جدول تابع 74161 در شکل ۱۵-۱۱ مشخص شده است. ورودی‌های موازی برچسب $5D$,1 دارند و به این معنی است که وقتی $M1$ فعال است (بار شدن $= 0$) و ساعت از گذر مثبت عبور می‌کند ورودی‌های D فعال می‌شوند. نقلی خروجی با برچسب زیر معین می‌شود.

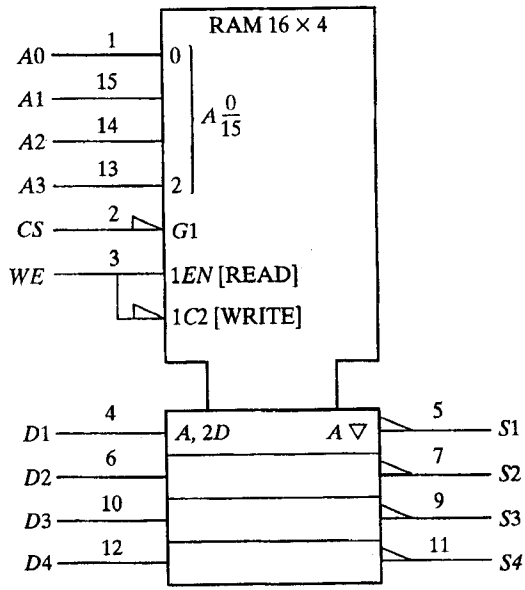
$$3CT = 15$$

تفسیر این است که نقلی خروجی به شرطی فعال است (برابر 1) که $G3$ فعال باشد ($ENT = 1$) و محتوای شمارنده (CT) 15 باشد (دودویی 1111). توجه کنید که خروجی‌ها سمبل L معکوس دارند، که به معنی حاکم- تابع بودن همه فلیپ فلاپ‌هاست. سمبل قطبیت ورودی $C5$ پالس معکوس را برای ورودی ساعت مشخص می‌نمایند. این بدان معنی است که حاکم در لبه منفی و تابع در لبه مثبت تغییر حالت می‌دهند. بنابراین خروجی در لبه مثبت پالس ساعت تغییر می‌یابد.

توجه کنید که آی‌سی 74LS161 (نوع شوتکی کم مصرف) دارای فلیپ فلاپ‌های حساس به لبه مثبت است.

۱۲-۸ سمبل RAM

سمبل گرافیک استاندارد برای حافظه (RAM) و 74189 در شکل ۱۵-۱۲ ملاحظه می‌شود. اعداد 16×4 که به دنبال RAM آمده است تعداد کلمات و تعداد بیت در هر کلمه را مشخص می‌کند. بلوک کنترل مشترک با چهار خط آدرس و دو ورودی کنترل نشان داده شده است. هر بیت از کلمه در بخش



شکل ۱۵-۱۲ . سمبل گرافیکی برای RAM ، 16×4 ، آی سی 74189

جداگانه‌ای با خط داده ورودی و خروجی نشان داده شده است. وابستگی آدرس A برای شناسایی ورودی‌های آدرس حافظه به کار رفته‌اند. ورودی‌ها و خروجی‌های داده‌ای که با آدرس تحت تأثیر واقع می‌شوند. با حرف A برچسب خورده‌اند. گروه‌بندی بیتی 0 تا 3 محدوده آدرسی از $A0$ تا $A15$ را فراهم می‌نماید. مثلث معکوس شده خروجی‌های سه حالته را بیان می‌نماید. سمبل قطبیت، معکوس شدن خروجی‌ها را مشخص می‌سازد.

طرزکار حافظه با توجه به نشانه وابستگی مشخص می‌گردد. سمبل گرافیک RAM از چهار وابستگی استفاده می‌کند: A(آدرس)، G (AND)، EN(فعال‌ساز) و C(کنترل)، ورودی $G1$ با $1EN$ و $1C2$، AND می‌شود زیرا $G1$ دارای 1 پس از حرف G بوده و دو ورودی دیگر دارای 1 در برچسب خود هستند. وابستگی $C2$ ورودی‌هایی راکه برچسب $2D$ دارند کنترل می‌کند. بنابراین برای یک عمل نوشتن، وابستگی $G1$ و $1C2$ $(CS = 0)$، وابستگی $C2$ و $2D$ $(WE = 0)$، وابستگی A را داریم که آدرس چهار خط آدرس را مشخص می‌سازد. برای یک خواندن، وابستگی $G1$ و $1EN$ $(CS = 0$، $WE = 1)$ و وابستگی A برای خروجی‌ها وجود دارد. تفسیر این وابستگی‌ها عملکرد حافظه را در لیست شکل ۱۸-۱۱ نتیجه می‌دهد.

مسائل

۱-۱۲ شکل ۱-۱۱ مدارهای مجتمع با فشردگی کم را با پایه‌های اختصاص یافته نشان می‌دهد. با استفاده از این اطلاعات، سمبل‌های گرافیکی مستطیل شکل را برای آی‌سی‌های 7400، 7404 و 7486 ترسیم نمایید.

۲-۱۲ موارد زیر را به زبان خود تعریف کنید.

(الف) منطق مثبت و منفی

(ب) فعال بالا و فعال پایین

(پ) شاخص قطبیت

(ت) شاخص دینامیکی

(ث) علامت وابستگی

۳-۱۲ از یک سمبل گرافیکی مثالی را نشان دهید که دارای سه وابستگی بولی G، V و N باشد. تفسیر معادلش را رسم کنید.

۴-۱۲ سمبل گرافیکی یک دیکدر BCD به دهدهی را رسم نمایید. این مشابه با یک دیکدر با 4 ورودی و 10 خروجی است.

۵-۱۲ سمبل گرافیکی یک دیکدر دودویی به هشت هشتی را با سه ورودی فعال‌ساز $E1$، $E2$ و $E3$ بکشید. اگر $E1 = 1$، $E2 = 0$ و $E3 = 0$ باشد مدار فعال می‌گردد (منطق مثبت فرض شود).

۶-۱۲ سمبل گرافیکی یک مولتی پلکسر دوگانه 4 به 1 خط را با ورودی‌های انتخاب مشترک و ورودی فعال‌ساز جداگانه برای هر مولتی پلکسر ترسیم نمایید.

۷-۱۲ سمبل گرافیکی را برای فلیپ فلاپ‌های زیر رسم کنید.

(الف) فلیپ فلاپ D حساس به لبه منفی

(ب) فلیپ فلاپ حاکم- تابع RS

(پ) فلیپ فلاپ T حساس به لبه مثبت

۱۲-۸ کار بلوک کنترل مشترک را وقتی با سمبل‌های گرافیکی استاندارد به‌کار می‌رود توضیح دهید.

۱۲-۹ سمبل گرافیکی یک ثبات 4 بیت با بار شدن موازی، برچسب $M1$ برای ورودی بارکردن و $C2$ برای ساعت رسم کنید.

۱۲-۱۰ همه سمبل‌های به‌کار رفته در نمودار شکل ۱۲-۱۲ را توضیح دهید.

۱۲-۱۱ سمبل گرافیکی یک شمارنده دودویی بالا- پایین‌شمار همزمان با ورودی حالت (برای بالا و پایین)، و ورودی شمارش فعال با وابستگی G را رسم نمایید.

۱۲-۱۲ سمبل گرافیکی یک RAM 256×1 را رسم نمایید. برای خروجی‌های سه حالته سمبل در نظر بگیرید.

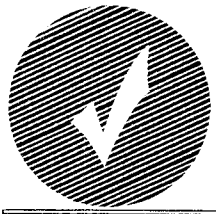
مراجع

1. 1984. *IEEE Standard Graphic Symbols for Logic Functions* (ANSI/IEEE Std. 91-1984). New York: Institute of Electrical and Electronics Engineers.

2. KAMPEL, I. 1985. *A Practical Introduction to the New Logic Symbols.* Boston: Butterworth.

3. MANN, F. A. 1984. *Explanation of New Logic Symbols.* Dallas: Texas Instruments.

4. 1985. *The TTL Data Book,* Volume 1. Dallas: Texas Instruments.



پاسخ به مسائل انتخابی

فصل ۱

۱-۲ (الف) 32,768 (ب) 67, 108, 864 (پ) 6, 871, 947, 674

۱-۴ $(198)_{12} = 260$ $(4310)_5 = 580$

۱-۵ (الف) 6 (ب) 8 (پ) 11

۱-۶ 8

۱-۷ 22.3125 (هر سه)

۱-۹ 64276 octal

۱-۱۲ (الف) 10000 و 110111 (ب) 958 و 62

۱-۱۹ (الف) 010627 (ب) 009025 (پ) 990975 (ت) 989373

۱-۲۳

دهدهی	1	1	3	6
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	0	0	1	0
4 (یا 0101)	0	1	1	0
5	1	1	1	0
6	0	0	0	1
7 (یا 1001)	1	0	0	1
8	1	1	0	1
9	0	0	1	1

۱-۳۰ 94 = 62 + 32 کاراکتر چاپی

۱-۳۱ بیت ششم از سمت راست

۱-۳۲ (الف) 897 (ب) 564 (پ) 871 (ت) 2,199

فصل ۲

۲-۲ (الف) x (ب) x (پ) y (ت) 0

۲-۳ (الف) B (ب) $z(x + y)$ (پ) $x'y'$ (ت) $x(w + z)$ (ث) 0

۲-۴ (الف) $AB + C'$ (ب) $x + y + x$ (پ) B (ت) $A'(B + C'D)$

۲-۶ (الف) $xy + x'y'$

۲-۸ $F(x, y, z) = \Sigma(1, 4, 5, 6, 7)$

۲-۹ (الف) 10001100 (پ) 00100011 (ت) 01010010

۲-۱۱ (ب) $(x' + y')' + (x + y)' + (y + z')'$

۲-۱۲ $T_1 = A'(B' + C')$ ، $T_2 = A + BC = T_1'$

۲-۱۴ (الف) $\Sigma(3, 5, 6, 7) = \prod(0, 1, 2, 4)$

۲-۱۵ (پ) $F = y'z + y(w + x)$

۲-۱۶ $\Sigma(1, 3, 5, 7, 9, 11, 13, 15) = \prod(0, 2, 4, 6, 8, 10, 12, 14)$

۲-۱۹ (الف) $AB + BC = (A + C)\,B$ (ب) $x' + y + z'$

فصل ۳

۳-۱ (الف) $xy + x'z'$ (ب) $C' + A'B$ (پ) $a' + bc$ (ت) $xy + xz + yz$

۳-۲ (الف) $x'y'$ (ب) $y + x'z$

۳-۳ (الف) $xy + x'z'$ (ب) $x' + yz$ (پ) $C' + A'B$

۳-۴ (الف) y (ب) $BCD + A'BD'$ (پ) $ABD + ABC + CD$

(ت) $wx + w'x'y$

۳-۵ (الف) $xz' + w'y'z + wxy$ (ب) $BD + B'D' + A'B$ یا $BD + B'D' + A'D'$

۳-۶ (الف) $B'D' + A'BD + ABC'$ (ب) $xy' + x'z + wx'y$

۳-۷ (الف) $x'y + z$ (پ) $AC + B'D' + A'BD + B'C$ (یا CD)

۳-۸ (الف) $F(x,y,z) = \Sigma(3,5,6,7)$ (ب) $F(A,B,C,D) = \Sigma(1,3,5,9,12,13,14)$

۳-۹ (الف) اصلی (اساسی): xz و $x'z'$: غیراصلی $w'x$ و $w'z'$

(ب) $F = B'D' + AC + A'BD + (CD$ یا $B'C)$

۳-۱۰ (پ) $F = BC + AC' + A'B'D$

اصلی B'C و AB غیراصلی AB, A'B'D, B'CD, A'C'D

۳-۱۱ (الف) $F = A'B'D' + AD'E + B'C'D'$

۳-۱۲ (ب) $F = (A + D')(B' + D')$

۳-۱۳ (الف) $F = xy + z' = (x + z')(y + z')$

۳-۱۵ (ب) $F = B'D' + CD' + ABC'D = \Sigma(0,2,6,8,10,13,14)$

۳-۱۷ $F' = BD + BC + AC$

۳-۱۹ (الف) $F' = (w + z')(x' + z')(w' + x' + y')$

۳-۳۰ $F = (A \oplus B)(C \oplus D)$

۳-۳۵ خط ۱ : خط تیره مجاز نیست، زیر خط (_) به کار ببرید: مثل 3_ . در پایان نقطه ویرگول (;) لازم دارد.

خط ۲ : ورودی‌ها باید وارد شوند (s در انتها نباشد) آخرین ویرگول (,) را به نقطه ویرگول (;) تبدیل نمایید.

خط ۳ : حرف بزرگ در "Output" نباشد آن را "output"کنید.

خط ۴ : "A" نمی‌تواند خروجی باشد (به عنوان ورودی تعریف شده است).

خط ۵ : "D" نمی‌تواند ورودی باشد (به عنوان خروجی تعریف شده است).

خط ۶ : OR باید با حرف کوچک باشد. آن را به "or" تغییر دهید.

خط ۷ : نقطه ویرگول را حذف کنید (پس از endmodule نقطه ویرگول نیست)

فصل ۴

۴-۱ (الف) $F_1 = A + B'C + BD' + B'D$

$F = A'B + D$

۴-۲ $F = ABC + A'D$

$G = ABC + A'D'$

۴-۳ (ب) 1024 ردیف و 14 ستون

۴-۴ $F = x'y' + x'z'$

۴-۶ $F = xy + xz + yz$

۴-۷ $w = A$ ، $x = A \oplus CB$ ، $y = x \oplus C$ ، $z = y \oplus D$

۴-۸ $w = AB + AC'D'$

۴-۱۰ ورودی‌ها: A, B, C, D : خروجی‌ها: w, x, y, z

$z = D$ ، $y = C \oplus D$ ، $x = B \oplus (C + D)$ ، $w = A \oplus (B + C + D)$

۴-۱۲ (ب) $D = x \oplus y \oplus z$

$$B = x'y + x'z + yz$$

۴-۱۳

	V	C	جمع
(الف)	1	0	1101
(ب)	1	1	0001
(پ)	0	1	0100
(ت)	1	0	1011
(ث)	0	0	1111

۴-۱۴ 60ns

۴-۱۸
$$w = A'B'C'$$
$$x = B \oplus C$$
$$y = C$$
$$z = D'$$

۴-۲۲
$$w = AB + ACD$$
$$x = B'C' + B'D' + BCD$$
$$y = C'D + CD'$$
$$z = D'$$

۴-۲۸
$$F_1 = \Sigma(0, 5, 7)$$
$$F_2 = \Sigma(2, 3, 4)$$
$$F_3 = \Sigma(1, 6, 7)$$

۴-۲۹
$$x = D_0'D_1'$$
$$y = D_0'D_1 + D_0'D_2'$$

۴-۳۴
$$F(A, B, C, D) = \Sigma(1, 6, 7, 9, 10, 11, 12)$$

۴-۳۵
$$AB = 00, F = D$$
$$AB = 01, F = (C + D)'$$
$$AB = 10, F = CD$$
$$AB = 11, F = 1$$

۴-۳۵

```
module Compare (A,B,Y);
input [3:0] A,B; //4-bit data inputs.
output [5:0] Y;  //6-bit comparator ouput.
reg [5:0] Y;     //EQ,NE,GT,LT,GE,LE
always @ (A or B)
      if (A==B)      Y = 6'b100011; //EQ,GE,LE
 else if (A < B)     Y = 6'b010101; //NE,LT,LE
      else           Y = 6'b011010; //NE,GT,GE
endmodule
```

۴-۴۲ (پ)

```
module Convrt_bhv (BCD, EXS3);
input [4:1] BCD;
output [4:1] EXS3;
reg [4:1] EXS3;
   always @ (BCD)
     EXS3 = BCD + 4'b0011;
endmodule
```

فصل ۵

۵-۴ (ب) $PQ' + NQ$

۵-۷ $S = x \oplus y \oplus Q$

$Q(t+1) = xy + xQ + yQ$

۵-۸ شمارنده‌ای با رشته تکراری 10, 01, 00

۵-۹ (الف) $A(t+1) = xA' + AB$

$B(t+1) = xB' + A'B$

۵-۱۰ (پ) $A(t+1) = xB + x'A + yA + y'A'B'$

$B(t+1) = xA'B' + x'A'B + y'A'B$

۵-۱۱

حالت فعلی:	00	00	01	00	01	11	00	01	11	10	00	01	11	10	10
ورودی:	0	1	0	1	1	0	1	1	1	0	1	1	1	1	0
خروجی:	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1
حالت بعدی:	00	01	00	01	11	00	01	11	10	00	01	11	10	10	00

۵-۱۲

	حالت بعدی		خروجی	
حالت فعلی	**0**	**1**	**0**	**1**
a	*f*	*b*	0	0
b	*d*	*a*	0	0
d	*g*	*a*	1	0
f	*f*	*b*	1	1
g	*g*	*d*	0	1

۵-۱۳ (الف)

State: *a f b c e d g h g g h a*
Input: 0 1 1 1 0 0 1 0 0 1 1
Output: 0 1 0 0 0 1 1 1 0 1 0

(ب)

State: *a f b a b d g d g g d a*
Input: 0 1 1 1 0 0 1 0 0 1 0
Output: 0 1 0 0 0 1 1 1 0 1 0

۵-۱۵ $D_Q = Q'J + QK'$

۵-۱۶ $D_A = Ax' + Bx$
$D_B = A'x + Bx'$

۵-۱۸ $J_A = K_A = (Bx + B'x')E$
$J_B = K_B = E$

۵-۱۹ (الف)
$D_A = A'B'x$
$D_B = A + C'x' + BCx$
$D_C = Cx' + Ax + A'B'x'$
$y = A'x$

۵-۲۳ (الف) RegA = 125, RegB = 125

(ب) RegA = 125, RegB = 30

۵-۲۶ $Q(t + 1) = JQ' + K'Q$
When $Q = 0, Q(t + 1) = J$
When $Q = 1, Q(t + 1) = K'$

```
always @ (posedge CLK)
   if (Q==0) Q = J;
   else Q = ~K;
```

۵-۳۰ دو فلیپ فلاپ E و Q با CLK

گیت AND با ورودی‌های A و B به ورودی D فلیپ فلاپ E می‌رود.

گیت OR با ورودی‌های E و C به ورودی D فلیپ فلاپ Q می‌رود.

اگر عبارتی که تخصیص‌های بلوکی را با سمبل (=) به کار برد، آنگاه فلیپ فلاپ Q باید با ورودی D که به جای اتصال به $C + E$ به $C + AB$ وصل است ترسیم شود (یک گیت AND و یک گیت OR).

فصل ۶

۶-۴ 1110; 0111; 1011; 1101; 0110; 1011

۶-۸ A = 0010, 0001, 1000, 1100. Carry = 1, 1, 1, 0

۶-۹ (ب) $J_Q = x'y$; $K_Q = (x' + y)'$

۶-۱۴ (الف) 4; (ب) 9; (پ) 10

۶-۱۵ 50ns; 20MHz

۶-۱۶ $1010 \rightarrow 1011 \rightarrow 0100$
$1100 \rightarrow 1101 \rightarrow 0100$
$1110 \rightarrow 1111 \rightarrow 0000$

۶-۱۷

$$D_{A0} = A_0 \oplus E$$
$$D_{A1} = A_1 \oplus (A_0 E)$$
$$D_{A2} = A_2 \oplus (A_1 A_0 E)$$
$$D_{A3} = A_3 \oplus (A_2 A_1 A_0 E)$$

۶-۱۹ (ب)

$$D_{Q1} = Q_1'$$
$$D_{Q2} = Q_2 Q_1' + Q_8' Q_2' Q_1$$
$$D_{Q4} = Q_4 Q_1' + Q_4 Q_2' + Q_4' Q_2' Q_1$$
$$D_{Q8} = Q_8 Q_1' + Q_4 Q_2 Q_1$$

۶-۲۱

$$J_{A0} = LI_0 + L'C$$
$$K_{A0} = LI_0' + L'C$$

۶-۲۴

$$T_A = A \oplus B$$
$$T_B = B \oplus C$$
$$T_C = AC + A'C' \quad \text{خود شروع نیست}$$
$$= AC + A'B'C \quad \text{خود شروع است}$$

۶-۲۶ یک شمارنده دو بیت به کار ببرید

۶-۲۸

$$D_A = A \oplus B$$
$$D_B = AB' + C$$
$$D_C = A'B'C'$$

۶-۳۴

```
module Shiftreg (SI,SO,CLK);
  input SI,CLK;
  output SO;
  reg [3:0] Q;
  assign SO = Q[0];
  always @ (posedge CLK)
    Q = {SI,Q[3:1]};
endmodule
```

۶-۳۶ (الف)

```
module updown (Up,Down,Load,IN,OUT,CLK)
  input Up,Down,Load,CLK;
  input [3:0] IN;
  output [3:0] OUT;
  reg [3:0] OUT;
  always @ (posedge CLK)
    if (Load) OUT = IN;
    else if (Up) OUT = OUT + 4'b0001;
    else if(Down)OUT = OUT -  4'b0001;
    else OUT = OUT;
endmodule
```

فصل ۷

۷-۲ (الف) 2^{13} (ب) 2^{31} (پ) 2^{26} (ت) 2^{21}

۷-۳ آدرس: 1011010011 = 2D3(hex)

داده: 0000 1101 0111 1011 = 0E7B(hex)

۷-۷ (الف) 7×128 decoders , 256 AND gates (ب) $x = 46; y = 112$

۷-۸ (الف) 8 chip (ب) 18; 15 (پ) 3×8 decoder

۷-۱۰ 0001 1011 1011 1

۷-۱۱ 101 110 011 001 010

۷-۱۲ (الف) 0101 1010; (ب) 1100 0110; (پ) 1111 0100

۷-۱۳ (الف) 6 (ب) 7 (پ) 7

۷-۱۴ (الف) 0101010

۷-۱۶ 24 پایه

۷-۱۸ (الف) 256×8 (ب) 512×5 (پ) 1024×4 (ت) 32×7

۷-۲۰ جملات ضرب: $x'y, z$, xy' , $x'y'z$, xz' , yz'

۷-۲۴
$$A = yz' + xz' + x'y'z$$
$$B = x'y' + xy + yz$$
$$C = A + xyz$$
$$D = z + x'y$$

فصل ۸

۸-۱ (الف) انتقال و افزایش در یک لبه ساعت رخ می‌دهند. پس از انتقال، محتوای R2 یکی بیشتر از $R1$ است.

(ب) محتوای $R3$ را یک واحد کم کنید.

(پ) اگر ($T1 = 1$) باشد، محتوای $R1$ را به $R0$ منتقل نمایید.

۸-۷ چارت ASM

$T0$: حالت اولیه: اگر ($S = 1$) آنگاه (ورودی $A \leftarrow RA$ ، ورودی $B \leftarrow RB$ ، سپس به $T1$ برو.)

$T1$: (متمم RB) + $RA \leftarrow RA$ ، (متمم نقلی) ← بیت قرض، به $T2$ برو

$T2$: اگر (قرض = 0) آنگاه به $T0$ برو. اگر (بیت قرض = 1) آنگاه $RA \leftarrow$ (متمم 2 ثبات RA ، به $T0$ برو

۸-۸ $T0$: داده ورودی $\leftarrow AR$ ، داده ورودی $\leftarrow BR$

$T1$: اگر (AR [15] = 1(بیت علامت منفی است) آنگاه AR(جابجایی به راست، گسترش علامت) ← CR در غیر این صورت ($AR = 0$) آنگاه ($CR \leftarrow 0$)
در غیر این صورت (مثبت غیرصفر) آنگاه ([15] ⊕ [14]) ← BR سرریز)
$CR \leftarrow BR$ (جابجایی به چپ)

۸-۹
$$D_{T0} = T_2 + S'T_0$$
$$D_{T1} = ST_0 + (A_3A_4)'T_1$$
$$D_{T2} = A_3A_4T_1$$

۸-۱۱
$$D_A = A'B + Ax$$
$$D_B = A'B'x + A'By + xy$$

۸-۱۴ چارت ASM:
$T0$: (مقدار اولیه) اگر $S = 0$ به حالت $T0$ بازگردد
اگر ($S = 1$) آنگاه مضروب ← BR، مضروب فیه ← AR، $0 \leftarrow PR$، به $T1$ برو

۸-۱۵ $(2^n - 1)(2^n - 1) < (2^{2n} - 1)$ for $n \geq 1$

۸-۱۶ (الف) سایز حاصلضرب ماکزیمم برابر 32 بیت در ثبات‌های A و Q
(ب) شمارنده P باید 5 بیتی باشد تا در ابتدا 10000 را ذخیره کند.
(پ) تشخیص Z (صفر)

۸-۱۸ $2(n + 1)$t

۸-۱۹
MUX1: 0, 1, 1, Z'
MUX2: S, 0, 1, 0

۸-۲۱ (الف) $E = 1$ (ب) $E = 0$

۸-۲۲
A = 0110, B = 0010, C = 0000.

A * B = 1100	A|B = 0110	A && C = 0
A + B = 1000	A ∧ B = 0100	|A = 1
A − B = 0100	&A = 0	A < B = 0
~C = 1111	~|C = 1	A > B = 1
A & B = 0010	A||B = 1	A !=B = 1

۸-۲۸

```
//Declare all inputs, outputs, and registers
     assign Z = ~|AR; //Z = 1 if AR = 0
     always @ (posedge CLK or negedge Clr)
        if (~Clr) pstate = T0;
        else pstate <= nstate;
     always @ (S or Z or pstate)
        case (pstate)
          T0: if (S) nstate = T1;
              else  nstate = T0;
```

```
        T1: if (Z) nstate = T0;
            else  nstate = T2;
        T2:    nstate = T1;
    endcase
always @ (posedge CLK)
    case (pstate)
        T0: if (S)
              begin
                AR <= Ain;
                BR <= Bin;
                PR <= 0;
              end
        T2:  begin
                PR <= PR + BR;
                AR <= AR - 8'b1;
              end
    endcase
```

فصل ۹

۹-۲ رشته Y_1Y_1 : 00, 01, 11, 11, 01, 00, 00

۹-۳ (ت) وقتی ورودی 01 باشد، خروجی 0 است. وقتی ورودی 10 است خروجی 1 می‌باشد. هر وقت ورودی دو ترکیب دیگر را دارا باشد، خروجی در مقدار قبلی باقی می‌ماند.

۹-۴ (پ)

	00	01	11	10
a	ⓐ, 0	b, 1	c, 1	d, 0
b	a , 0	ⓑ, 1	c, 1	ⓑ, 0
c	ⓒ, 1	b, 1	ⓒ, 1	d, 0
d	c, 1	b, 1	c, 1	ⓓ, 1

۹-۵ (پ)

$$Y_1 = x_1'x_2 + x_2y_1$$
$$Y_2 = x_2 + x_1y_2$$
$$z = x_1x_2y_1' + x_1y_2'$$

۹-۱۰

$$S = x_2x_2'$$
$$R = x_1'x_2$$

۹-۱۳ (ب) دو جدول گذر ممکن

	00	01	11	10
a	ⓐ, 0	*b*, –	–, –	*e*, –
b	ⓑ, 1	ⓑ, 1	–, –	*d*, –
d	*a*, –	ⓓ, 1	–, –	ⓓ, 1
e	ⓔ, 1	*d*, –	–, –	ⓔ, 1

	00	01	11	10
a	ⓐ, 0	*b*, –	–, –	*b*, –
b	*c*, –	ⓑ, 1	–, –	ⓑ, 0
c	ⓒ, 1	*d*, –	–, –	*d*, –
d	*a*, –	ⓓ, 1	–, –	ⓓ, 1

۹-۱۸ 3a: $(a, b)(c, d)(e, f, g, h)$
3b: $(a, e, f)(b, j)(c, d)(g, h)(k)$

۹-۲۰ به تخصیص دودویی، g و h را اضافه کنید.

	00	01	11	10
0	*a*	*g*	*b*	*f*
1	*c*	*h*	*d*	*e*

۹-۲۲ $F = A'D' + AC'D' + A'BC + A'CD'$

۹-۲۳ $Y = (x_1 + x_2')(x_2 + x_3)(x_1 + x_3)$

فصل ۱۰

۱۰-۱ گنجایش خروجی = 10؛ توان مصرفی = 18.75 mW، تأخیر انتشار = 3ns
حد پارازیت = 0.3V

۱۰-۲ (الف) 1.058V (ب) 0.82V (پ) 0.238V

۱۰-۳ $I_B = 0.44\text{mA}$, $I_{CS} = 2.4\text{mA}$

۱۰-۴ (الف) 2.4mA (ب) 0.82mA (پ) 2.4+0.82N (ت) 7.8 (ث) 7

۱۰-۵ (ب) 3.53 (پ) 2.585mA (ت) 16mA (ث) 300Ω

۱۰-۹ (الف) 4.62mA (ب) 4mA

۱۰-۱۰ 0.3V

واژه‌نامه

آرایه گیتی برنامه‌پذیر موردی field programmable gate array	
آرایه منطقی برنامه‌پذیر programmable logic array	
آگاه گر آدرس سطر	row address strobe
ادغام - سنتز - ترکیب	synthesis
ادغام	merging
اصل توزیع پذیری	distributive low
اصل جابجایی	commutative
اصل شرکت پذیری	associative
افکتورهای فورمت	format effector
امپدانس بالا	high impedance
انتخابگر	selector
انتخابگر تراشه	chip select
سطح انتقال بین ثباتی	register transfer level
انکدر	encoder
یک بارز	one-hot
بازگشت نورد	carriage return
باز نشان مستقیم	direct reset
بازنشانی	reset
بالا به پایین	top-down
بردار	vector
برنامه تست	test bench
بسته بودن	closure
بلوکی	blocking
بیت تست	check bit
پایین به بالا	buttom-up
پایین شمار	count - down counter
پردازشگر	processor
پسبر	backspace
پیش بینی نقلی	look a head carry
پیش تنظیم	preset
تئوری وفاق	consensus theorem
تابع	slave
تابع غیرکامل	incompletely specified function
تأخیر انتشار	propagation delay
تخصیص اجرایی	procedural state assignment

تراشه	chip
تست توازن	parity check
تصدیق منفی	negative acknowledge
تعریف شده به وسیله کاربر	user-defined
تکلیف	task
تنظیم مستقیم	direct set
توازن	parity
تواناساز	enable
توان مصرفی	power dissipation
توصیف ساختاری	structural description
توصیف مدار	circuit description module

(ث)

ثبات توالی	sequence register

(ج)

جداساز رکورد	record separator
جداساز فایل	file separator
جداسازی اطلاعات	information separator
جدول ایجاب	implication table
جدول‌بندی افقی	horizontol tabulation
جدول تحریک	exitation table
جدول روند	flow table
جدول گذر	transition table
جمع کننده کامل	full adder
جورج بول	G.boole

(چ)

چرخه - سیکل	cycle

(ح)

حافظه با دستیابی تصادفی	random access memory
حافظه فقط خواندنی	read only memory
حاکم	master
حالت	state
حالت کلی	total state
حالت متناهی	finite state machine
حد پارازیت	noise margin
حساس به اندازه	case sensetive

(خ)

خاص - فروشنده	vendor - specific
ختم متن	end of text
خط تیره	underscore

(د)

دگروضع	toggle
دهدهی کد شده به دودویی	binary coded decimal
دیسک چندکاره دیجیتال	digital versatile disk
دیکدر	decoder
دی مولتی پلکسر	demultiplexer

(ر)

رقابت بحرانی	critical race
رقابت غیربحرانی	noncritical race

(ز)

زبان توصیف سخت افزاری	hardware decription language
زبان طراحی سخت افزار	hardware description language
زمان برپایی	setup time
زمان نشست	settling time
زمان نگهداری	hold time
زیر خط	under score

(س)

ساعت آزادگرد	free-running clock
سرریز	overflow
سطح انتقال ثباتی	register level transfer
سعی و کاهش	cut - and - try
IC سفارشی	full - custom IC

hierarchical description	سلسه مراتبی
tri - state (three-state)	سه حالته
wire	سیمی

(ش)

simulation	شبیه سازی
start of text	شروع متن
switch-tail ring counter	شمارنده حلقوی دنباله چرخان

(ض)

schematic capture	ضبط تصویری

(ط)

computer aided design	طراحی باکمک کامپیوتر

(ظ)

stray capacitance	ظرفیت پارازیتی

(ع)

mixed notation	علائم مخلوط
bitwise	عمل بیتی
concatenation	عملگر ادغام
fundamental mode operation	عملیات اساسی
identiy element	عنصر شناسه

(غ)

non-bloking	غیربلوکی
asynchronous	غیر همزمان

(ف)

very large scale	فشردگی خیلی زیاد
large scale	فشردگی زیاد
small scale integration	فشردگی کم
medium scale	فشردگی متوسط

(ک)

communication control characters	کاراکترهای کنترل تبادل اطلاعات
appliction specific	کاربرد خاص
excess -3	کد افزونی -3
american standard code for information interchange	کددودویی استاندارد برای کاراکترهای الفبا عددی اسکی
binary coded decimal	کد دهدهی به دودویی
self complement	کدهای خود متمم

(گ)

fan out	گنجایش خروجی
fan in	گنجایش ورودی
primitive gate	گیت‌های اصلی

(ل)

transparent	لچ شفاف

(م)

algorithmic stste machine ASM	ماشین حالت الگوریتمی
canonical	متعارف
signed complement system	متمم - علامت‌دار
appliction standard IC	مدارات مجتمع خاص
clock generator	مدارهای ترتیبی ساعت‌دار
behavioral modeling	مدلسازی رفتاری
continuous state assignment	مدل سازی رونده داده
structural modeling	مدل‌سازی ساخت یافته
switch-level modeling	مدل‌سازی سطح سوئیچ
instantiation	ذکر
stimulus module	مدول محرک
modulus	مدولوس:عملی ریاضی که نتیجه آن باقیمانده یک تقسیم است
data path	مسیر داده
valied	معتبر

معکوس	inverse
سیستم مقدار-علامت‌دار	signed-magnitude system
خروجی مقصد یا هدف	target output
ممیز	radix
منطق آرایه‌ای برنامه‌پذیر	logic programmable array
مواردی که به وسیله کاربر تعریف می‌شود	user defined primitives
موجب‌های اصلی	prime implicant
موجب‌های اصلی اساسی	essential prime implicant
مولتی پلکسر	multiplexer

(ن)

نشاندن	set
نشانه لبه	edge qualifire
نقشه مرکب	composit map
نقلی	carry
نقلی انتهایی	end carry
نقلی چرخشی	end around carry
نمودار تصویری	schematic
نیم جمع‌کننده	half adder

(و)

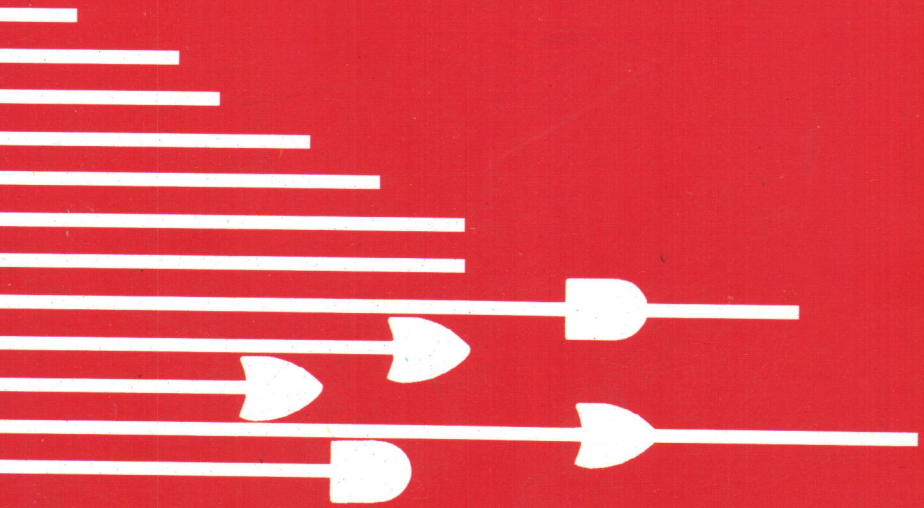
واحد پردازش مرکزی	central processing unit
واحدهای زمانی	time units
ورودی پاک	clear
وسیله منطقی برنامه‌پذیر	programmable logic device
وسیله منطقی برنامه‌پذیر پیچیده	complex programmable logic divice
وسیله منطقی برنامه‌پذیر ترتیبی	sequential (or simple) programmable logic device
وضعیت رقابتی	race condition

(ه)

همزمان	synchronous
NC	not connected

DIGITAL DESIGN

M.MORRIS MANO



انتشارات خراسان

مرکز پخش : مشهد، چهارراه دکترا ، انتشارات خراسان تلفن (دورنما) : ۸۴۱۱۸۲۱