پیشگفتار

این کتاب با توجه به سر فصل تعیین شده برای دانشجویان دانشگاه پیام‌نور در رشته کامپیوتر با گرایش نرم‌افزار تهیه و تنظیم شده است. در تهیه این کتاب سعی بر آن شده است تا مباحثی که برای تدریس درس سه واحدی مدار منطقی لازم به تدریس است، مطرح گردند. این کتاب مشتمل بر نه فصل می‌باشد.

در ابتدای کتاب لیست سر فصل مطالب قید شده است. در انتهای کتاب مجموعه‌ای از سؤالات شامل ۹۰ سئوال تستی و ۲۳ سئوال تشریحی به همراه پاسخ نامه سؤالات تستی ارائه شده است.

نظر به لزوم جاگذاری معادل فارسی کلمات تخصصی برای راحتی فهم دانشجویان دو واژه‌نامه به‌صورت انگلیسی به فارسی و فارسی به انگلیسی در انتهای کتاب آمده

است. در صفحه پایانی کتاب لیست منابع و ماخذ نیز برای آگاهی دانشجویان ارائه شده است.

این اثر با دقت نظر فراوان کارشناسان مدیریت تولید مواد و تجهیزات آموزشی مورد ارزیابی قرار گرفت که بدینوسیله از جناب آقای اکبری به نمایندگی از آن عزیزان قدردانی می‌نمایم.

کتاب حاضر بعنوان منبع درسی در دانشگاه پیام‌نور اعلام شده که بعلت کوتاه بودن زمان امکان رفع کلیه ایرادات تایپی و نگارشی میسر نشد. لذا با وجود سعی و دقت فراوان در پدید آوردن اثری خودخوان ضمن پذیرفتن ایرادات احتمالی، در نوبت اول تیراژ این اثر محدود خواهد بود تا در نیمسال آینده بعد از دریافت پیشنهادات اصلاحی صاحب‌نظران، اساتید و دانشجویان نسبت به چاپ در تیراژ بالاتر اقدام گردد.

در پایان از آقای مهندس کامیار آهنکوب که در تنظیم و تدوین کتاب همکاری شایانی داشته‌اند سپاسگزاری می‌کنم.

داود کریم‌زادگان‌مقدم

تابستان ۱۳۸۵

فصل ۱

اعداد مبنا در سیستم‌های دیجیتال

هدف کلی

در این فصل مباحث کلی سیستم‌ها و معماری مدارهای دیجیتال به صورت کلی مطرح شده و در ادامه مباحث مربوط به مبناهای اعداد و روش‌های تبدیل اعداد مبنا شرح داده خواهند شد. همچنین متمم‌های اعداد نیز مورد بحث و بررسی قرار گرفته و انواع اعداد دودویی علامت‌دار توضیح داده خواهند شد. در ادامه انواع کدهای دودویی و دهدهی و ... نیز ارائه خواهند شد.

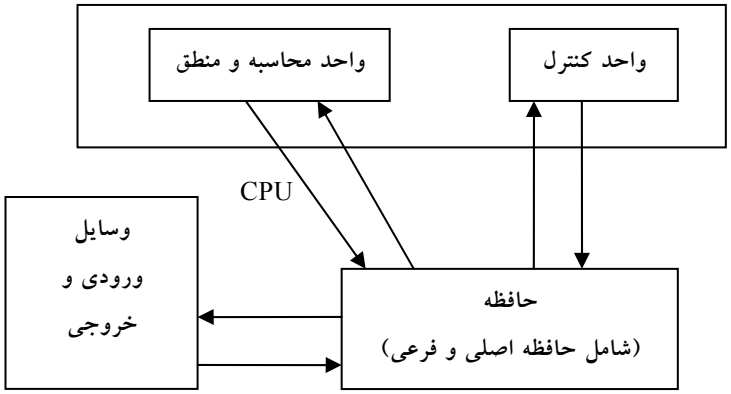
هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- سیستم‌های دیجیتال
- مبناهای اعداد
- روش‌های تبدیل اعداد در مبناهای مختلف
- متمم‌های اعداد
- اعداد دودویی علامت‌دار
- جمع و تفریق اعداد
- انواع جداول و کدهای دودویی و ...

۱-۱ معماری[1] سیستم‌های کامپیوتری

سیستم‌های کامپیوتری که شامل مجموعه‌ای از دستورالعمل‌ها و سخت‌افزارها می‌باشند، بر اساس ترکیبی از دو نوع معماری نرم‌افزاری و سخت‌افزاری شکل می‌گیرند. معماری نرم‌افزار[2] به‌صورت کلی شامل مجموعه‌ای از دستورات و فرمت استفاده از آنها می‌باشد. معماری سخت‌افزار[3] سیستم نیز شامل مولفه‌های سخت‌افزاری به شرح ذیل می‌باشد:



شکل ۱-۱: شمای کلی از معماری سخت‌افزاری سیستم کامپیوتری

- پردازنده[4] که ابزاری است برای تفسیر و اجراء دستورالعمل‌ها
- حافظه[5] که ابزاری است برای ذخیره‌سازی داده‌ها و برنامه‌ها
- ابزار انتقال اطلاعات بین اجزاء داخلی کامپیوتر و یا مابین کامپیوتر و محیط بیرونی

[1] Architecture
[2] Software Architecture
[3] Hardware Architecture
[4] Processor
[5] Memory

شکل ۱-۱ معماری یک سیستم کامپیوتری را به‌صورت شماتیک و کلی نشان می‌دهد که در آن نحوه ارتباط بین عناصر نیز آمده است:

۱-۲ سیستم‌های دیجیتال

امروزه سیستم‌های الکترونیکی یا به بیانی دیگر سیستم‌های دیجیتال که شامل مجموعه‌ای از مدارات برقی و... می‌باشند در تقریباً تمامی‌علوم مانند مخابرات، تجارت، محاسبات ریاضی و علمی، ناوبری هواپیماها و سفینه‌های فضایی، اعمال جراحی، اینترنت و بسیاری از دیگر زمینه‌های تجاری، صنعتی و علمی به کار می‌روند. بهترین مثال از یک سیستم دیجیتال، کامپیوتر دیجیتال همه منظوره است. 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. بخشهای اصلی یک کامپیوتر عبارتند از واحد حافظه، واحد پردازش مرکزی و واحدهای ورودی-خروجی، واحد حافظه برنامه‌ها و داده‌های وارده، خارج شونده و میانی را ذخیره می‌کند. واحد پردازش مرکزی اعمال محاسباتی و دیگر عملیات روی داده‌ها را بر حسب آنچه در برنامه مشخص شده، انجام می‌دهد. داده‌ها و برنامه‌هایی که به وسیله کاربر آماده شده‌اند توسط وسایل ورودی مانند صفحه کلید به حافظه انتقال می‌یابند. یک وسیله خروجی مثل چاپگر نتایج حاصل از محاسبات را دریافت کرده و به کاربر ارائه 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 نه‌تنها می‌تواند محاسبات ریاضی را انجام دهد، بلکه قادر است اعمال

منطقی را هم اجرا نماید. به‌علاوه می‌تواند جهت تصمیم‌گیری بر اساس شرایط داخلی یا خارجی برنامه‌ریزی شود.

۱-۲-۱ عناصر گسسته[1] در سیستم‌های دیجیتال

یکی از ویژگی‌های سیستم دیجیتال، توانمندی آنها در دستکاری عناصر گسسته اطلاعاتی است. هر مجموعه‌ای که به تعداد متناهی از عناصر محدود باشد، اطلاعاتی گسسته را داراست. مثال‌هایی از عناصر گسسته عبارتند از ۱۰ رقم دهدهی، ٢٦ حرف الفباء، ٦٤ 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 رقمی‌از این مفهوم حاصل شده است. عناصر گسسته اطلاعاتی در یک سیستم دیجیتال با کمیت‌های فیزیکی به نام سیگنال[2] 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 آنها را دودویی می‌نامند. یک رقم دودویی که بیت خوانده می‌شود دو مقدار دارد: 0 و 1. عناصر گسسته اطلاعاتی با گروهی از بیت‌ها به نام کدهای دودویی نمایش داده می‌شوند. مثلاً ارقام دهدهی 0 تا 9 در سیستم اعداد دیجیتال با کد چهار بیتی نشان داده می‌شوند. با به‌کارگیری 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 سیستمی‌است که با عناصر گسسته اطلاعاتی به شکل دودویی کار می‌کند.

---

[1] Discrete Elements
[2] Signal

کمیتهای اطلاعاتی یا ذاتاً گسسته‌اند و یا از نمونه‌برداری فرآیندهای پیوسته حاصل می‌شوند. به عنوان مثال یک لیست حقوق ذاتاً یک فرآیند یا رویداد گسسته بوده و حاوی: نام کارمند، شماره تامین اجتماعی، حقوق هفتگی، مالیات بر درآمد 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 به‌طور خودکار به‌وسیله دستگاهی به نام مبدل آنالوگ به دیجیتال انجام می‌شود.

برای استفاده از مدارهای دیجیتال در تولیدات تجاری دلایل اساسی وجود دارد. همچون کامپیوترهای دیجیتال، دستگاههای دیجیتال نیز قابل برنامه‌ریزی‌اند. با تعویض برنامه در یک وسیله برنامه‌پذیر، همان سخت‌افزار یگانه، قابلیت استفاده در کاربردهای متفاوت را خواهدداشت. 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 پیاده‌سازی شده، قیمت هر واحد کاهش یافته و قیمت دستگاههای دیجیتال روز بروز کاهش می‌یابد. دستگاههای ساخته شده با مدارهای مجتمع می‌توانند با سرعتی تا صد میلیون عمل در ثانیه را انجام دهند. می‌توان با استفاده از کدهای اصلاح خطا عملکرد سیستم‌های دیجیتال را به شدت اطمینان بخش نمود. مثالی از این نوع، دیسک چند کاره دیجیتال (DVD) است که در آن

اطلاعات ویدیویی، صوتی و دیگر گونه‌ها بدون از دست رفتن حتی یک قلم داده، ضبط می‌گردد.

۱-۲-۲ زبان توصیف سخت‌افزاری[1]

یک سیستم دیجیتال از به هم پیوستن ماژول‌های دیجیتال[2] به‌دست می‌آید. برای درک عمل هر ماژول، دانش و آگاهی مدارهای دیجیتال و عمل منطقی آنها لازم است.

یک گرایش مهم در طراحی دیجیتال، استفاده از زبان توصیف سخت‌افزاری 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.

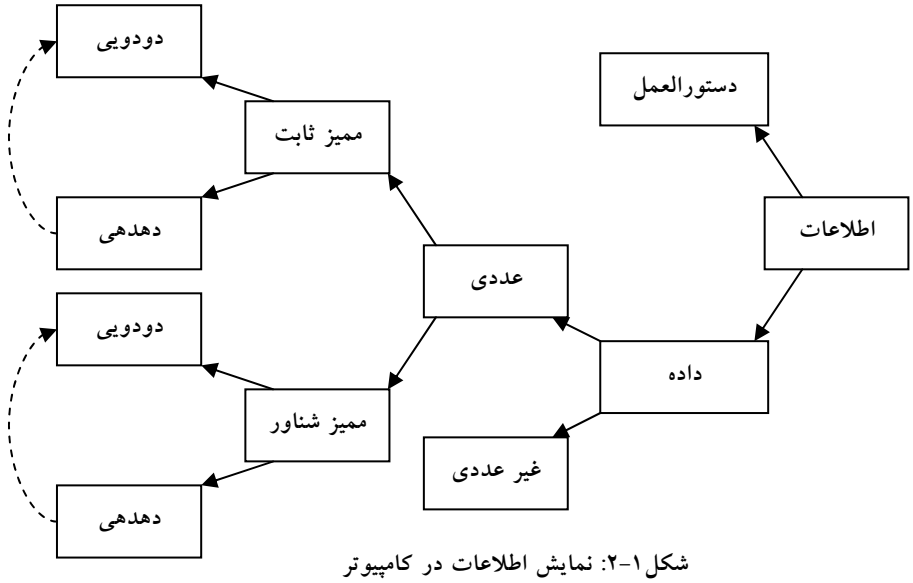
۱-۳ نمایش اطلاعات در کامپیوتر

همانطور که قبلاً گفته شد سیستم‌های دیجیتال کمیت‌های گسسته اطلاعات، که به فرم دودویی نمایش داده شده‌اند را دستکاری می‌نماید. عملوندهای به کار رفته در محاسبات را می‌توان در سیستم 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. کمیت‌ها نیز در عناطر دودویی ذخیره می‌شوند. شکل ۱-۲ ساختار درختواره‌ای نمایش انواع اطلاعات را در یک سیستم کامپیوتری نشان می‌دهد .هدف این فصل معرفی مفاهیم دودویی متعدد به صورت یک مرجع برای مطالعات بعدی در فصل‌های آینده است.

---

[1] Hardware Description Language (HDL)

[2] Digital Modules



شکل۱-۲: نمایش اطلاعات در کامپیوتر

۱-۳-۱ اعداد دودویی

به طور کلی یک عدد با نقطه اعشاری با یکسری ضرایب به صورت زیر نمایش داده می‌شود:

$$a_5a_4a_3a_2a_1a_0.a_{-1}a_{-2}a_{-3}$$

که ضرایب a_x هر یک از ده رقم (0 و 1 و 2و... و 9) بوده و x 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:

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

یک عدد دهدهی مانند 6548 کمیتی معادل با 6 هزارتایی، به علاوه 5 صدتایی، به علاوه 4 ده‌تایی به علاوه 8 واحد را نشان می‌دهد. هزارها، صدها و ... توانی از 10 هستند که با توجه به مکان ضرایب معین می‌گردند. به بیان دقیق‌تر، 6548 را می‌توان به صورت زیر نوشت:

$$6 * 10^{3} + 5 * 10^{2} + 4 * 10^{1} + 8 * 10^{0}$$

با این وجود معمول این است که فقط ضرایب را بنویسیم و توان‌های لازم 10 را از مکان آنها استنتاج کنیم.

سیستم اعداد دهدهی را در مبنای 10 گویند. 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_x در 2^x ضرب می‌گردد. مثلاً معادل دهدهی عدد دودویی 11010.11 برابر 26.75 می‌باشد، که از ضرب ضرایب در توان‌هایی از 2 به‌دست می‌آید:

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

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

$$a_n \cdot r_n + a_{n-1} \cdot r_{n-1} + \cdots + a_2 \cdot r_2 + a_1 \cdot r_1 + a_0 + a_{-1} \cdot r_{-1} + a_{-2} \cdot r_{-2} + \cdots + r_{-m}\, a_{-m}.$$

که ضرایب a_x بین 0 تا r-1 می‌باشند. برای تفکیک اعداد در مبناهای مختلف، ضرایب را در داخل پرانتزها نوشته و اندیس مبنا را در زیر آن می‌گذاریم (به جز در اعداد دهدهی که محتوا بیانگر دهدهی بودن است).

همانطور که 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. بنابر این تبدیل دودویی به دهدهی با جمع توانهایی از 2 که ضرایب آن 1 است صورت می‌گیرد. مثلاً عدد زیر را در نظر بگیرید:

$$(101101)_2 = 32+8+4+1 = (45)_{10}$$

در عدد فوق چهار عدد 1 دیده می‌شود. دهدهی مربوطه جمع چهار توان از 2 می‌باشد. 24 اعداد اول (مجموعاً معادل ۳ بایت) حاصل از 2 به توان n درجدول

شکل۱-۳ نشان داده شده‌اند. ستون‌ها یک در میان بیانگر یک بایت که معادل ۸ بیت می‌باشند در نظر گرفته شده‌اند تا راحت‌تر بتوان نمایش دودویی را یاد گرفت.

n	2^n	n	2^n	n	2^n
0	1	8	256	16	65536
1	2	9	512	17	131072
2	4	10	1024	18	262144
3	8	11	2048	19	524288
4	16	12	4096	20	1048576
5	32	13	8192	21	2097152
6	64	14	16384	22	4194304
7	458	15	32768	23	8388608

شکل۱-۳: توانهایی از ۲

اعمال حسابی با اعدادی در مبنای r از همان قواعد دهدهی استفاده می‌کنند. هنگامی‌که از مبنایی به جز 10 استفاده می‌شود باید دقت کرد که تنها r رقم مجاز به کار گرفته شود. جمع دو عدد دودویی مشابه قوانین دهدهی محاسبه می‌شود، به جز این که ارقام جمع در هر مکان با ارزش 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. به مثال زیر توجه نمایید:

```
  101101
+ 100111
---------
1010100      حاصل جمع:
```

تفریق کمی پیچیده‌تر است. قوانین باز هم همان قوانین دهدهی هستند، به جز این که قرض در یک مکان با ارزش، 2 را به رقم مفروق‌منه می‌افزاید. (قرض در سیستم دهدهی، 10 واحد به رقم مفروق‌منه اضافه می‌کند).

```
  10110
- 10011
---------
000110       باقیمانده:
```

عمل ضرب خیلی ساده است. ارقام مضروب‌فیه همیشه 1 یا 0 هستند. بنابراین حاصلضرب‌های جزیی برابر با 0 یا برابر با مضروب می‌باشند.

1011
×101
1011
0000
1011
10111 حاصل ضرب:

۱-۳-۲ 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 در مبنای شانزده متعلق به چهار رقم دودویی است.

هنگامی‌که تعداد ارقام کمتر از 10 باشد مرسوم است که r رقم مورد نیاز برای ضرایب از سیستم دهدهی گرفته شود. هنگامی‌که مبنای عدد از 10 بزرگتر است از حروف الفبا برای تکمیل 10 رقم دهدهی استفاده می‌گردد.

در زیر جدولی از شانزده عدد اول در مبناهای دهدهی، دودویی، هشت‌تایی و شانزده‌تایی نشان داده شده است:

همانطور که در جدول شکل ۱-٤ مشاهده می‌کنید دو ستون سمت راست مربوط به مبناهای هشت و شانزده می‌باشند که در ادامه به شرح هر یک خواهیم پرداخت. سیستم اعداد هشت هشتی یک سیستم مبنای 8 با هشت رقم 0، 1، 2، 3، 4، 5، 6، 7 می‌باشد. لازم به ذکر است که ارقام 8 و 9 نمی‌توانند در یک عدد هشت هشتی ظاهر شوند.

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

شکل۱-٤: اعداد با مبناهای متفاوت

مثالی از یک عدد مبنای هشت عدد 537.4 است. برای تعیین مقدار معادل دهدهی لازم است عدد را به صورت یک سری از توانها با مبنای 8 بسط دهیم.

$$(537.4)_8 = 5*8^2 + 3*8^1 + 7*8^0 + 4*8^{-1} = (351.5)_{10}$$

در سیستم اعداد شانزده شانزدهی (مبنای 16)، ده رقم اول از سیستم دهدهی گرفته می‌شوند. حروف A، B، C، D، Eو F به ترتیب به جای ارقام 10، 11، 12، 13، 14 و 15 به‌کار می‌روند. مثالی از یک عدد در مبنای 16 به صورت زیر است.

$$(B65F)_{16} = 11*16^3 + 6*16^2 + 5*16^1 + 15*16^0 = (46687)_{10}$$

۱-۳-۳ تبدیل مبنای اعداد

همانطور که قبلاً اشاره 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 به 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 جمع‌اًوری باقیمانده‌ها انجام می‌گردد.

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

$$(10\ 110\ 001\ 101\ 011.\ 111\ 1\ 0000\ 0110)_2=(26153.7406)_8$$

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 رقمی‌تفکیک می‌شوند:

$$(10\ 1101\ 0111\ 1010\ .\ 1111\ 0110)_2=(2D7A.\ F6)_{16}$$

2 D 7 A F 6

در ادامه مثال‌های مختلفی ارائه شده است:

مثال۱: عدد 41 را به دودویی تبدیل کنید.

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

		خارج قسمت صحیح		باقیمانده	ضریب عدد دودویی
41/2	=	**20**	+	**½**	**a0=1**
20/2	=	**10**	+	**0**	**a1=0**
10/2	=	**5**	+	**0**	**a2=0**
5/2	=	**2**	+	**½**	**a3=1**
2/2	=	**1**	+	**0**	**a4=0**
1/2	=	**0**	+	**½**	**a5=1**

جواب: $(41)_{10}=(a_5\ a_4\ a_3\ a_2\ a_1\ a_0)_2=(101001)_2$

بنابراین، پاسخ $(101001)_2=(a_5\ a_4\ a_3\ a_2\ a_1\ a_0)_2=(41)_{10}$ می‌باشد.

روال فوق را می‌توان به طریق ساده‌تر زیر دستکاری کرد:

41	
خارج قسمت	باقیمانده
20	1
10	0
5	0
2	1
1	0
جواب= 101001	

تبدیل اعداد صحیح دهدهی به هر سیستم مبنای r مشابه مثال فوق است به جزاین که تقسیم در عوض ۲ بر r انجام می‌گردد.

مثال۲: عدد $(0.6875)_{10}$ را به دودویی تبدیل کنید.

ابتدا0.6875 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 به‌دست 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 به‌دست می‌آید.

صحیح		کسری	ضریب
0.6875 * 2=1	+	0.3750	a-1 =1
0.3750* 2 = 0	+	0.7500	a-2 =0
0.7500 * 2 =1	+	0.5000	a-3 =1
0.5000 *2 =1	+	0.0000	a-4 =1

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

مثال۳: عدد 153 را به مبنای هشت ببرید.

مبنای مورد نظر 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 باقیمانده3 را به‌دست دهد. بالاخره 2 بر 8 تقسیم گردیده تا خارج قسمت 0 و باقیمانده 2 به‌دست آید. این روند به صورت مناسب زیر انجام می‌گردد:

	153
باقیمانده	**خارج قسمت**
1	**19**
3	**2**
جواب = 231	

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.

مثال۴: عدد $(0.513)_{10}$ را به مبنای هشت ببرید.

$0.513 * 8 = 4.104$

$0.104 * 8 = 0.832$

$0.832 * 8 = 6.656$

$0.656 * 8 = 5.248$

$0.248 * 8 = 1.984$

$0.984 * 8 = 7.872$

جواب تا هفت رقم با معنی که از بخش صحیح حاصلضرب‌ها به‌دست 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 و ۲ داریم:

$$(41.6875)_{10} = (101001.1011)_{2}$$

با استفاده از مثالهای ۳ و ٤ داریم:

$$(153.513)_{10} = (231.406517)_{8}$$

تبدیل اعداد از مبنای هشت یا شانزده به مبنای دودویی با روشی عکس روش بالا انجام می‌گردد که این رو به شرح زیر می‌باشد:

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:

$$(673.124)_{8} = (110\ 111\ 011.\ 001\ 010\ 100)_{2}$$

6 7 3 1 2 4

و

$$(306.D)_{16} = (0011\ 0000\ 0110.\ 1101)_{2}$$

3 0 6 D

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 به‌کار 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 بر حسب 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 مطلوب تر است زیرا که 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 مناسب تر به نظر می‌رسد ولی انتخاب یکی از این دو کاملاً اختیاری است.

۱-٤ متمم اعداد

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:

متمم مبنا

متمم مبنای کاهش یافته

فرم اول به متمم r و دومی‌به 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 9, 10 را خواهیم داشت.

۱-٤-۱ متمم مبنا

متمم r یک عدد n رقمی‌مانند 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 ۱۰ 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 010100 = 010011+1 و از جمع 1 با مقدار متمم 1 به‌دست می‌آید. چون 10^{n} عدد 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 کم ارزش‌تر و تفریق اولین رقم غیر صفر کم ارزش‌تر از 10 و تفریق همه رقم‌های با ارزش‌تر از 9 حاصل می‌گردد. برای نمونه

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 عدد دو 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 با ارزش‌تر حاصل می‌شود. به طور نمونه

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 باقی مانده به‌دست 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 آن را موقتاً حذف نمود تا متمم r و (r –1) به‌دست آید. آنگاه آن را به مکان مربوطه‌اش باز می‌گردانیم.

۱-٤-۲ متمم در مبنای کاهش یافته

با فرض داشتن عددی n رقمی‌مانند N در مبنای r، متمم (r – ۱) عدد به صورت (N– ($r^{\wedge n}$–1)) تعریف می‌شود. برای اعداد دهدهی، r = 10 و r–9=1 است، و به این ترتیب متمم 9 عدد N برابر (N– ($10^{\wedge n}$–1)) خواهد بود. در اینجا $10^{\wedge n}$ نمایشگر عددی است که متشکل از 1 و به دنبال آن n عدد 0 می‌باشد.

$10^{\wedge n}$–1 عددی است که با n عدد 9 نشان داده می‌شود.

مثلاً اگر n = 4 باشد، داریم $10^{\wedge 4}$ = 10000 و 9999 = 1– $10^{\wedge 4}$. به این ترتیب نتیجه می‌شود که 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،

(N–($2^{\wedge n}$–1)) خواهد بود. مجدداً، $2^{\wedge n}$ برابر با یک عدد دودویی است که از یک 1، و n عدد 0 تشکیل شده است. $2^{\wedge n}$–1 یک عدد دودویی متشکل از n عدد 1 می‌باشد. مثلاً اگر n = 4 باشد، داریم $(10000)_2$ = $2^{\wedge 4}$ و $(1111)_2$=1–$2^{\wedge 4}$. بنابراین متمم 1 یک عدد دودویی از تفریق هر رقم از 1 به‌دست می‌آید. با این وجود، هنگام تفریق ارقام دودویی از عدد 1، یکی از دو حالت 0=1–1 و یا 1=0–1 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 1هابه 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 به ترتیب از تفریق ارقام آنها از 7 یا F (15 دهدهی) حاصل می‌شود.

توجه: 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.

۱-٤-۳ 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 با ارزش‌تر قرض گرفته می‌شود. این روش هنگامی‌که 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 کارایی دارد. تفریق دو عدد n رقمی بی‌علامت $M - N$ در مبنای r به صورت زیر انجام می‌شود:

مفروق‌منه M را به متمم r مفروض N، اضافه کنید. یعنی

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

اگر $M \geq N$ باشد، عمل جمع یک رقم نقلی انتهایی تولید می‌کند که باید چشم پوشی شود ؛آنچه باقی می‌ماند نتیجه $M - N$ است.

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 به‌دست می‌آوریم و سپس یک علامت منفی در جلو آن می‌گذاریم.

مثال‌های زیر روال را تشریح می‌کنند.

مثال۵: با استفاده از متمم 10 تفریق 3250 –72532 را انجام دهید.

M=	**72532**
Nعدد 10 متمم =	**+96750**
حاصل جمع =	**169383**
چشم پوشی از رقم نقلی انتهایی 5 ^10 =	**–100000**
جواب =	**69282**

توجه کنید که M دارای 5 رقم ولی N فقط 4 رقمی‌است. 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 با ارزش‌ترین مکان تولید می‌نماید. تولید رقم نقلی در با ارزش‌ترین مکان دلالت بر $M \geq N$ دارد و نتیجه نیز مثبت است.

مثال ۶: با استفاده از متمم 10 تفریق 72532 –3250 را به‌دست آورید.

M=	**03250**
Nعدد 10 متمم =	**+ 27468**
حاصل جمع =	**30718**

که رقم نقلی در آن وجود ندارد. بنابراین

جواب (متمم 10 عدد30718) – = 69282 –

توجه کنید که چون $72532 > 3250$ است نتیجه منفی است. نظر به این که ما با اعداد بی علامت سر و کار داریم، نمی‌توان برای این حالت نتیجه بدون علامتی به‌دست 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 درآید. تفریق با متمم‌ها برای اعداد دودویی به روش مشابهی در مثال‌های زیر آمده است.

مثال۷: 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
= چشم پوشی از نقلی انتهایی 2^7	- 10000000
= X-Y: جواب	0010001

(ب)

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

رقم نقلی انتهایی وجود ندارد. بنابراین، جواب

Y – X = (متمم 2 عدد1101111) =0010001 -

تفریق اعداد بی علامت 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 نقلی انتهایی و افزودن 1 به حاصل جمع را رقم نقلی چرخشی می‌خوانند.

مثال۸: مثال ۷ را با استفاده از متمم 1 انجام دهید.

(الف) X–Y = 1010100 –1000011

X =	**1010100**
Y =	**+0111100**
= حاصل جمع	**1001000**
= رقم نقلی انتهایی	**+ 1**
= X-Y جواب	**0010001**

(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 به‌دست آمده است زیرا این نوع متمم به کار رفت. روال رقم نقلی انتهایی چرخشی در تفریق اعداد دهدهی بی علامت با متمم 9 نیز قابل استفاده است.

۱-٤-٤ اعداد دودویی علامت‌دار

اعداد صحیح مثبت و از آن جمله صفر را می‌توان با اعداد بی علامت نشان داد. با این وجود برای نمایش اعداد صحیح منفی به علامتی نیاز نداریم. 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 چپ ترین 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 بی علامت 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 علامت هستند. اگر عدد دودویی بدون علامت فرض شود، سمت چپ‌ترین بیت، با ارزش‌ترین بیت عدد خواهد بود. مثلاً رشته بیت‌های 01001 می‌تواند به عنوان 9 (دودویی بی علامت) و یا 9+ (دودویی علامت‌دار) در نظر گرفته شود زیرا سمت چپ‌ترین بیت 0 است. رشته بیت‌های 11001 به عنوان 25 بی علامت و یا 9 علامت‌دارمنفی 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 علامت‌دارمنفی 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 علامت‌دارمنفی برای ارائه اعداد منفی به کار گرفته می‌شود. در این سیستم، یک عدد منفی با متمم‌اش مشخص می‌شود. در حالی که سیستم مقدار علامت‌دارمنفی، عدد را با تغییر علامتش منفی می‌کند، سیستم متمم علامت‌دارمنفی ، منفی عدد را با متمم سازی‌اش 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 علامت‌دارمنفی 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 چپ ترین 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 علامت‌دارمنفی 10001001
- نمایش متمم 1علامت منفی‌دار 11110110
- نمایش متمم 2علامت منفی‌دار 11110111

در سیستم مقدار علامت‌دارمنفی، با تغییر بیت علامت در سمت چپ‌ترین مکان، از 0 به 1، عدد 9- به 9+ تبدیل می‌شود. در متمم 1 علامت منفی‌دار، 9- را با متمم کردن همه بیت‌های9+، از جمله بیت علامت به‌دست می‌آوریم. در متمم2 علامت منفی‌دار، 9- با متمم2 کردن تمام بیت‌های عدد مثبت از جمله بیت علامت حاصل می‌شود.

شکل۱-۵، 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 چپ ترین مکان می‌باشند.

دهدهی	متمم 2علامت منفی‌دار	متمم1علامت منفی‌دار	مقدارعلامت منفی‌دار
+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	-	-

شکل۱-۵: اعداد دودویی علامت‌دار

سیستم متمم 2علامت منفی‌دار 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 علامت‌دارمنفی 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 علامت‌دارمنفی در حساب معمولی مورد استفاده است و هنگامی‌که در کامپیوتر به کار رود، مشکلاتی به همراه دارد زیرا باید علامت و مقدار به طور جداگانه دستکاری شوند. بنابراین، معمولاً متمم علامت منفی 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.

۱-۵ جمع حسابی

جمع دو عدد در سیستم مقدارعلامت منفی 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:

(+14) +(−39) = −(39−14) =−25است، که با تفریق مقدار کوچکتر 14 از مقدار بزرگتر 39 و استفاده از علامت 39 برای علامت نتیجه انجام شده است. این فرایند به مقایسه علامت‌ها و اندازه‌ها و سپس اجرای جمع و تفریق نیاز دارد. روال مشابهی در نمایش مقدارعلامت منفی برای اعداد دودویی قابل اعمال است. بر عکس قوانین جمع

اعداد در سیستم مقدار علامت منفی 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:

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

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.

توجه کنید که اعداد منفی باید از ابتدا به صورت متمم 2 باشد 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 عدد دو هستند.

برای یافتن یک جواب صحیح، باید مطمئن بود 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. هنگامی‌که جمع با کاغذ و قلم انجام می‌شود سرریز مسئله‌ای نیست زیرا ما از نظر عرض صفحه محدودیت نداریم. در این‌گونه 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 نمائیم. 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 مقدارعلامت منفی عادت کرده‌اند، ناآشنا است. 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 درآید. مثلاً عدد دودویی علامت‌دار 11111001 یک عدد منفی است زیرا سمت چپ ترین 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.

۱-٦ 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:

۱. متمم 2 مفروق (ازجمله بیت علامت) را به‌دست آورید.

۲. آنرا به مفروق منه(ازجمله بیت علامت)اضافه کنید.

۳. رقم نقلی خروجی از مکان بیت علامت چشم پوشی 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 به‌دست 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 (11110011 – 11111010) است. با یافتن

متمم 2 مفروق (13–)، یعنی (13+)، تفریق به فرم جمع در می‌آید. در دودویی این عمل به‌صورت زیر می‌باشد:

$$11111010 + 00001101 = 100000111$$

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 منفی 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 به‌طور 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 باثبات دارند استفاده می‌کنند. بین سیگنال‌های دودویی، عناصر مدار دودویی و ارقام دودویی رابطه مستقیمی وجود دارد. مثلاً یک عدد دودویی n رقمی‌را 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 اجزا 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 ۱ها و 0 ها را نگه دارند. باید 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 بیتی، 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:

11،10 ،01 ،00

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 1- 2^n حاصل می‌گردد. به هر عنصر باید یک ترکیب بیتی دودویی منحصر بفرد اختصاص یابد و هیچ دو عنصر دارای مقدار یکسانی نمی‌باشند؛ در غیر این صورت تخصیص کد گنگ و بی معنی خواهد بود.

گرچه حداقل تعداد بیت‌های لازم برای 2^n 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 نه 0 و یک 1 تخصیص می‌یابد. در این کد دهدهی، رقم 6 به ترکیب بیتی 0001000000 اختصاص می‌یابد.

۱-۷-۱ کد BCD

گرچه سیستم اعداد دودویی طبیعی ترین 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. هنگامی‌که 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 از هم تفکیک کند باید حداقل 4 بیت داشته باشد، ولی 6 ترکیب از 16 ترکیب ممکن تخصیص نیافته باقی می‌ماند. با مقدار دهی 10 گانه به 4 بیت می‌توان کدهای دودویی متفاوتی به دست آورد. کدی که برای ارقام دهدهی معمولاً به‌کار می‌رود در شکل ۱-٦ نشان داده می‌شود. این کد را دهدهی کد شده به دودویی می‌خوانند و به طور خلاصه آنرا با BCD نمایش می‌دهند. چند کد دهدهی دیگر بعدا در این بخش نمایش داده خواهند شد.

همان گونه که در شکل بالا مشاهده می‌شود، هر کد 4 بیتی به یک رقم دهدهی نسبت داده می‌شود. در این بین توجه به چند نکته الزامی‌است:

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

شکل۱ - ٦: دهدهی کد شده به دودویی (BCD)

یک عدد k رقمی‌در BCD به 4k بیت نیاز دارد.

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. ترکیبات دودویی 1010 تا 1111 در BCD مفهومی ندارند.

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. بعلاوه 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 اعدادBCD اعداد دهدهی هستند و نه اعداد دودویی، هرچند 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 به کار 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.

۱-۷-۲ جمع BCD

جمع دو رقم دهدهی در BCD، همراه با رقم نقلی احتمالی از جفت رقم کم ارزش‌تر قبلی را در نظر بگیرید. چون هر رقم از 9 تجاوز نمی‌کند، جمع نمی‌تواند بزرگتر از 9+9+19=1 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 11001 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 با ارزش‌ترین مکان بیتی حاصل از جمع دودویی و نقلی دهدهی برابر است با 6 = 10 – 16. 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 10010 را همراه با یک رقم نقلی به وجود می‌آورد. در مثال سوم، جمع دودویی یک رقم نقلی خواهد داشت. این وضعیت هنگامی‌رخ می‌دهد که حاصل جمع مساوی یا بزرگتر از 16 باشد. گر چه 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 7=(0111) و یک نقلی BCD حاصل می‌گردد.

BCD carry	**1**	**1**		
	0001	**1000**	**0100**	**184**
	+0101	**0111**	**0110**	**+576**
Binary sum	**0111**	**10000**	**1010**	
Add 6	-----	**0110**	**0110**	-----
BCD sum	**0111**	**10110**	**0000**	**760**

جمع دو عدد BCD بی علامت n رقمی‌روال مشابهی دارد. جمع 760=576+184 را به BCD در نظر بگیرید:

اولین جفت رقم BCD کم ارزش‌تر، 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.

۱-۷-۳ حساب دهدهی

نمایش اعداد دهدهی علامت‌دار در BCD مشابه اعداد علامت‌دار در دودویی است. ما می‌توانیم از هر یک از هر دو سیستم مقدارعلامت‌دارمنفی یا متمم علامت منفی‌دار 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 BCD 9 یعنی 1001 نشان داده می‌شود.

سیستم مقدار علامت‌دارمنفی به ندرت در کامپیوتر ها به کار می‌رود. این سیستم می‌تواند متمم 9 یا متمم 10 باشد، ولی اغلب متمم 10 به کار گرفته می‌شود. برای به‌دست آوردن متمم 10 یک عدد BCD، ابتدا متمم 9 را به‌دست آورده و به کم ارزش‌ترین 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 چشم پوشی 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.

0 375
+9 760
0 135

عدد 9 واقع در سمت چپ‌ترین مکان عدد دوم نمایشگر یک علامت منفی و 9760 متمم 10 عدد 0240 است. برای به‌دست آوردن 135+، دو عدد با هم جمع و رقم نقلی نادیده گرفته می‌شود. البته اعداد دهدهی داخل کامپیوتر، از جمله ارقام علامت باید 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. متمم10 مفروق را به‌دست 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.

۱-۷-٤ دیگر کدهای دهدهی

کدهای دودویی برای ارقام دهدهی به حداقل چهار بیت در قبال هر رقم نیاز دارند. با

رقم دهدهی	BCD ۸٤۲۱	۲٤۲۱	افزونی۳	۱-۲-٤-۸
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
ترکیبات	1010	0101	0000	0001
	1011	0110	0001	0010
بیتی بکار	1100	0111	0010	0011
	1101	1000	1101	1100
نرفته	1110	1001	1110	1101
	1111	1010	1111	1110

شکل۱-۷ : چهار کد دودویی متفاوت برای ارقام دهدهی

ایجاد 10 ترکیب مختلف در چهار بیت کدهای مختلف را می‌توان ایجاد کرد. کدهای BCD و سه نوع کد دیگر در جدول ۵-۱ نشان داده شده‌اند. هر کد تنها 10 ترکیب بیتی از 16 ترکیب ممکن را در چهار بیت به کار می‌برند. 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 ۱ها در ترکیب کد به‌دست می‌آید. کد BCD دارای وزن های 8، 4، 2، 1 است که مربوط به توازنی از دو برای هر بیت است. مثلاً تخصیص بیتی 0110 با توجه به وزن ۱ها برای6 تفسیر می‌شود زیرا

$$8 * 0 + 4 * 1 + 2*1 +1 * 0=6$$

ترکیب بیتی 1101 وقتی با کد 2421 وزین شود معادل دهدهی 7=1 * 1+ 2*0 + 1 * 4 + 1 * 2 را خواهد داد توجه کنید که در کد 2421 بعضی از ارقام به دو طریق کدگذاری می‌شوند. عدد 4 دهدهی به ترکیب های 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-۳ 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 به‌دست می‌آید (مثل متمم 1 عدد دودویی). کد افزونی-۳ به دلیل خود متممی‌اش در کامپیوترهای قدیمی‌به کار رفت. این کد بی وزن 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 -، 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

2 =0 ❋(1 −)+ 1 ❋ (2 −) + 1 ❋4 +0 ❋8 محاسبه می‌شود.

۱-۷-۵ کد کاراکتراسکی

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 قرار داد، 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 شش بیت و در حالت دوم به هفت بیت نیاز است.

	b7b6b5							
b4b3b2b1	**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**

شکل۱-۸: کدهای استاندارد آمریکایی اسکی (ASCII)

کد دودویی استاندارد برای کاراکترهای الفبا عددی، اسکی (ASCII) است. این کد از هفت بیت برای کد کردن 128 کاراکتر، طبق جدول (۷-۱) استفاده می‌کند. هفت بیت کد با b1 تا b7 مشخص شده‌اند که b7 با ارزش‌ترین بیت را تشکیل می‌دهد. مثلاً حروف A در اسکی به صورت 1000001 (ستون 100 و سطر 0001) می‌باشد. جدول کدهای استاندارد اسکی حاوی:

94 کاراکتر گرافیکی

34 کاراکتر غیر چاپی برای عملیات کنترلی مختلف

می‌باشد که در آن کاراکترهای گرافیکی نیز از 26 حرف بزرگ (A تا Z)، 26 حرف کوچک (a تاz)، 10 عدد (0 تا 9) و 32 کاراکتر قابل چاپ مانند ٪، * و $ تشکیل شده است.

۱-۷-٦ کدهای کنترل کننده در ASCII

34 کاراکتر کنترل در جدول اسکی با اسامی‌خلاصه شده‌ای مشخص شده‌اند. این کاراکترها در پایین شکل همراه با نوع عملشان ذکر شده‌اند. سه نوع کاراکترهای کنترلی وجود دارند:

- افکتورهای فرمت
- جداسازی اطلاعات
- کاراکترهای کنترل تبادل اطلاعات

افکتور فرمت قالب آنچه را که باید چاپ شود کنترل می‌نماید. این گروه شامل پسبر (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 قاب بندی یک پیام متنی به هنگام انتقال از خط تلفن به کار می‌روند.

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

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	**From 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**

شکل ۱-۹: لیست کاراکترهای کنترلی

به کاربرد بستگی دارد. مثلاً بعضی از چاپ‌گرها کاراکترهای اسکی را به صورت 8 بیتی می‌شناسند که در آن با ارزش‌ترین بیت برابر 0 است. 128 کاراکتر 8 بیتی دیگر را با قرار دادن 1 در با ارزش‌ترین مکان برای فونت های نوع ایتالیک یا الفبای یونانی می‌توان به کار برد.

۱-۷-۷ کد تشخیص خطا

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

		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 اینکه تعداد ۱ها در کاراکتر برای توازن فرد، فرد گردد. به طور کلی یکی از دو توازن اختیار می‌شود ولی توازن زوج معمول تر 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 باز پس می‌فرستند. 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.

سؤالات

١- عدد $(0.2498)_{10}$ را به دودویی تبدیل کنید.

٢- عدد 2406 را به مبنای هشت ببرید.

٣- عدد 673.124 در مبنای هشت معادل چه اعدادی در مبناهای دودویی و دهدهی است.

٤- متمم 10 عدد 256703 چه عددی خواهد بود.

٥- متمم 2 عدد 0100101 چه عددی خواهد بود.

٦- متمم 9 عدد 062374 چه عددی خواهد بود.

٧- با استفاده از متمم 10 تفریق 46532 –3248 را به‌دست آورید.

فصل دوم

گیت‌های منطقی، جبر بول و توابع بولی

هدف کلی

در این فصل مباحث کلی مربوط به منطق دودویی به همراه مفاهیم اساسی جبر بول مورد بحث و بررسی قرار خواهند گرفت. در ادامه مفاهیم و تئوری‌های اساسی جبر بول بررسی شده و در ادامه توابع بول نیز به‌صورت کامل مورد نقد و بررسی قرار خواهند گرفت. در ادامه نیز انواع گیت‌های منطقی به همراه جداول درستی هر یک بررسی خواهند شد.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- منطق دودویی
- انواع گیت‌های منطقی
- اصول اساسی جبر بول
- تئوری‌های اساسی جبر بول
- تقدم عمل‌گرها در جبر بول
- توابع بول
- متمم توابع بول
- مدارهای مجتمع

۲-۱ منطق دودویی

منطق دودویی با متغیرهایی که دو ارزش گسسته و عملیاتی که مفهوم منطقی دارند، سرو کار دارد و ارزشی که متغیرها اختیار می‌کنند ممکن است با اسامی‌مختلفی نام‌گذاری شوند (مثل صحیح و غلط، بله و خیر و غیره)، اما برای ما بهتر است آن را بر حسب 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. در این بخش جبر بول به روشی غیر مستدل بوده و ارتباط آن با مدارهای منطقی دیجیتال و سیگنال‌های دودویی بیان شده است.

۲-۱-۱ 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 و1 دارد. سه نوع عملیات منطقی اصلی وجود دارند: AND، OR و NOT. در ادامه به شرح هر یک از عملیات می‌پردازیم:

۱- AND: این عمل به وسیله یک "."یا بدون ذکر هر عملگری نمایش داده می‌شود. مثلاً x.y = z یا x y = 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 وy 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 و 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.

AND

x	y	x. y
0	0	0
0	1	0
1	0	0
1	1	1

OR

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

NOT

x	x'
0	1
1	0

شکل۲-۱: جدول درستی عملیات منطقی

۳- NOT: این علامت با یک علامت پریم نشان داده می‌شود (و گاهی با یک خط بار). مثلاً x′=z (یا 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.

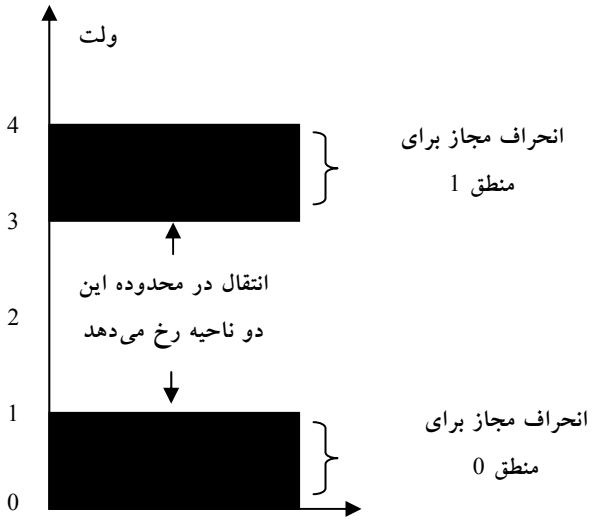
منطق دودویی شبیه حساب دودویی است، و اعمال AND و OR به ترتیب به اعمال ضرب و جمع شباهت دارند. در حقیقت سمبل‌های به کار رفته برای AND و OR همان هایی هستند که برای ضرب و جمع مورد استفاده قرار می‌گیرند. معهذا منطق دودویی را نباید با حساب دودویی اشتباه کرد. مسئله‌ای که باید مورد توجه قرار گیرد این است که یک 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=10+1 (می‌خوانیم: "یک بعلاوه یک برابر است با 2")، در صورتیکه 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 اعمال 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 متغیرها ی 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 زیر نشان داده شده‌اند. این جداول تعریف عملیات مذکور را به طور شفاف بیان می‌دارند.

۲-۱-۲ 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 به دو سطح ولتاژ که نمایشگر

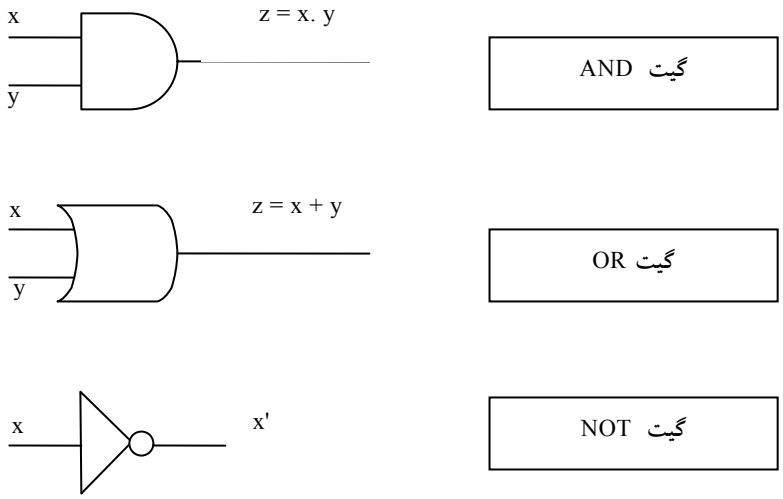


شکل ۲-۲: محدوده انتقال سیگنال 0 و 1

یک متغیر دودویی و برابر با منطق 1 و منطق 0 اند واکنش نشان می‌دهند. مثلاً یک سیستم دیجیتال خاص ممکن است منطق 0 را به عنوان سیگنالی برابر با 0 ولت و منطق 1 را به صورت سیگنالی برابر با 4 ولت تعریف کند. در عمل، هر سطح ولتاژ، محدوده مورد قبولی مانند شکل۲-۲ را داراست.

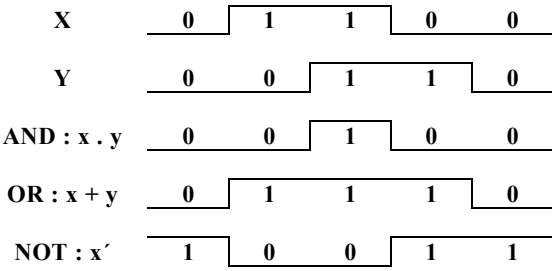
پایانه‌های ورودی مدارهای دیجیتال سیگنال‌های دودویی را در محدوده مجازی می‌پذیرند و در پایانه‌های خروجی در محدوده مجازی پاسخ می‌دهند. ناحیه میانی بین دو ناحیه مجاز تنها هنگام گذر از یک حالت به حالت دیگر قطع می‌شود. هر اطلاعات محاسباتی یا کنترلی مورد نظر را می‌توان با عبور سیگنال‌هایی دودویی از میان ترکیباتی 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 شکل۲-۳ دیده می‌شوند:



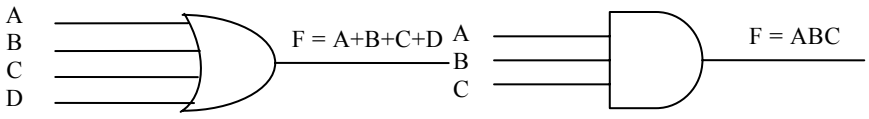
شکل ۲-۳: سمبل‌های مداری منطقی دیجیتال

گیت‌ها، بلوکهایی سخت‌افزاری‌اند که با ورودی منطقی مناسبی، در خروجی خود 0 یا 1 تولید می‌نمایند. سیگنال ورودی x وy در گیت‌های AND و OR در یکی از چهار



شکل۲-٤: سیگنال‌های ورودی-خروجی برای گیت‌های AND , OR ,NOT

حالت ممکن قرار دارند: 00، 01، 10 و 11. این سیگنال‌ها همراه با خروجی خود در شکل۲-٤ دیده می‌شوند. نمودارهای زمانی پاسخ هر گیت را برای چهار گیت فوق نشان می‌دهند. محور افقی نمودار 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 هنگامی‌خروجی 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.



الف- گیت AND با سه ورودی **ب- گیت OR با چهار ورودی**

شکل ۲-۵: مدار سوئیچینگ نمایش دهنده منطق دودویی

گیت‌های AND وOR ممکن است بیش از دو ورودی داشته باشند. یک گیت AND با سه ورودی و یک OR با چهار ورودی در شکل۲-۵ 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 هنگامی‌خروجی 1 تولید می‌کند که یکی از ورودی‌ها در 1 منطقی باشد. خروجی هنگامی0 می‌شود که همه ورودی‌ها در منطق 0 باشند.

۲-۲- جبر بول

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 وy عناصر مشخصی از آن باشند، آنگاه $x \in s$ به این معنی است که x عضوی از مجموعه s است و $y \notin s$ یعنی y عضوی از مجموعه 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 4,3,2,1. 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 aو 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 به‌کار رفته در فرموله کردن ساختارهای جبری عبارتند از:

۱. بسته بودن

۲. عنصر شناسه

۳. عنصر معکوس

٤. اصل شرکت‌پذیری

۵. اصل جابجایی

٦. اصل توزیع‌پذیری

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 $N=\{1,2,3,4,\ldots\}$ را نسبت به عملگر جمع (+) بسته گوییم زیرا برای هر دو عنصر $a,b\in N$ 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- عنصر شناسه: مجموعه S نسبت به عملگر (*) روی مجموعه S دارای عنصر شناسه است، اگر عنصر $e \in S$ با خاصیت زیر موجود باشد.

به ازای هر $x\in S$ داشته باشیم: $e*x = x*e = x$

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

به ازای هر $x\in I$ داشته باشیم: $x + 0 = 0 + x = x$

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

۳- عنصر 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

$$a + (-a) = 0$$

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

به ازای همه مقادیر $x,y,z \in S$ داشته باشیم: $(x*y) * z = x* (y*z)$

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

به ازای هر $x,y \in S$ داشته باشیم: $x*y = y*x$

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

$$x*(y.z) = (x*y).(x*z)$$

مثالی جبری در این مورد میدان یا حوزه است. میدان مجموعه‌ای از عناصر است، همراه 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.

معکوس جمع، تفریق می‌باشد..

شناسه جمع، 0 است

شناسه ضرب 1 می‌باشد.

عملگر دودویی (.) ضرب را تعریف می‌نماید.

معکوس ضرب a = 1/a تقسیم را تعریف می‌کند، یعنی

$a.(1/a) = 1$

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

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

۲-۲-۱ تعریف اصول اساسی جبر بول

در سال ۱۸۵٤ جورج بول یک سیستم جبری را که امروزه آن را جبر بول می‌نامیم پایه‌ریزی کرد. در سال ۱۹۳۸ نیز 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 هانتینگتون در سال ۱۹۰٤ را به‌کار می‌بریم. **اصول هانتیگتون** به شرح زیر بودند:

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- برای هر عنصر $x\in B$، عنصری مثل $x'\in B$ وجود دارد (به آن متمم x می‌گوییم) به نحوی که: (a) $x'+x= I$ و (b) $x'.\ x = 0$

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

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

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

۶- (a) (.) نسبت به (+) توزیع پذیر است: $x.\ (y+z) = (x.y) + (x.z)$

(b) (+) نسبت به (.) توزیع پذیر است: $x+ (y.z) = (x+y) + (x+y)$

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. به بیانی دیگر 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.

۲-۲-۱-۱ تفاوت‌های جبر بول با جبر معمولی

با مقایسه جبر بول با حساب و جبر معمولی (حوزه یا میدان اعداد حقیقی) تفاوت های 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.

٤- اصل ۳ 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 بی نهایت 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 به‌صورت مجموعه‌ای از دو عنصر 0 و 1 تعریف می‌شود.

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 به‌کار می‌روند، سمبل‌هایی هستند که به جای اعداد حقیقی به‌کار می‌روند. به‌طور مشابه در جبر بول مجموعه B دارای متغیرهایی مانند x، y و z می‌باشد که صرفاً سمبل هستند و عناصر را نشان می‌دهند.

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، سرو کار خواهیم داشت. جبر بول دو ارزشی در تئوری مجموعه‌ها و منطق کاربرد دارد. هدف ما در این کتاب کاربرد جبر بول در مدار منطقی گیتی است.

۲-۲-۱-۲ جبر بول دو ارزشی

جبر بول دو ارزشی روی مجموعه دو عنصری، { 1وB = 0 }، به همراه قوانین برای دو

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

عملگر دودویی (+) و (.) که در جدول زیر نشان داده شده تعریف می‌شود (قانون عملگر متمم برای تصدیق اصل ۳ است):این قوانین دقیقاً مثل اعمال AND ، OR و

NOT در شکل ۲-۱ می‌باشند. اکنون نشان می‌دهیم که اصول هانتینگتون برای مجموعه، { 1و0 = B } 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 , ∈ B می‌باشند.

۲- با توجه به جداول می‌بینیم که:

$$(a)\ 1+1 = 0+1 = 1+0 = 1$$

$$(b)\ 1.0 = 0.1 = 0.0 = 0$$

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

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

٤- (a) صحت اصل توزیع‌پذیری $x.(y+z) = (x.y)+(x.z)$ را با ایجاد جدول برای تمام مقادیر ممکن z,y,x ، می‌توان تحقیق کرد. برای هر ترکیب، $x.(y+z)$ را به‌دست آورده و نشان می‌دهیم که برابر $(x.y)+(x.z)$ 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 مانند بند قبل تحقیق نمود.

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

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

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

٦- اصل ٦ نیز صادق است زیرا جبر بول دو ارزشی دارای دو مقدار مجزای 0 و 1 با $1\neq0$است.

تا اینجا ما یک جبر دو ارزشی با عملگرهای OR , AND 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 غیر مستدل است. بیان غیر مستدل 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 تاکید بر تشابه بین جبر بول دو ارزشی و دیگر سیستم‌های دودویی، این جبر در بخش قبل "منطق دودویی" نامیده شد. از این پس، کلمه "دو ارزشی" را در بحث‌های بعدی از جبر بول حذف می‌کنیم.

۲-۲-۲ قضایای اصلی و خواص جبر بول

اصول هانتینگتون به‌صورت جفت جفت لیست شده‌اند و به‌صورت بخشهای (a) و (b) مشخص شدند. هر یک از این دو را با تعویض عملگرها و عنصر شناسه می‌توان از دیگری به‌دست 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 و0اند. اصل دوگانگی کاربردهای متعددی دارد. اگر دوگان یک عبارت جبری مورد نظر باشد تنها کافی است عملگرهای AND و OR تعویض و 0ها به 1 و ۱ها به 0 تبدیل گردند.

۲-۲-۲-۱ تئوری‌های اساسی جبر بول

تئوری‌ها و اصول لیست شده اساسی ترین روابط در جبر بول اند. تئوری‌ها نیز مانند اصول به صورت 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 به‌کار رفته نوشته شده است.

تئوری ۱ (a): x+x=x

x+x

=(x+x).1
=(x+x)(x+x′)
=x+xx′
=x+0
=x

تئوری ۱ (b):x.x=x

x.x

=xx+0
=xx+xx′
=x(x+x′)
=x.1
=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.(x+1)
=(x+x′)(x+1)
= x+x′.1
= x+x′
=1

تئوری ۲ (b): بر اساس دوگانگی x.0=0

تئوری ۳: $x = (x')'$. با توجه به اصل3 ، داریم $x+x'= 1$, $x.x'=0$ ، 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 به‌صورت جبری ثابت کرد. مثلاً: تئوری جذب را در نظر بگیرید.

تئوری ٤: شرکت‌پذیری

(a): x + (y + z) = (x + y) + z

(b): x (y z) = (x y) z

تئوری ۵:

(a): $(x + y)' = x' y'$

(b): $(x y)' = x' + y'$

x+xy

= x.1+xy

=x(1+y)

=x(y+1)

=x.1

=x

تئوری ٦ (a): x+xy=x

تئوری ٦ (b): بر اساس دوگانگی x(x+y)=x

۲-۲-۲-۲ تقدم عملگرها

در ارزیابی عبارت جبر بول تقدم اول با پرانتز، دوم با NOT، سوم با AND و چهارم 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.

۲-۳ 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 یا 1 و سمبل‌های عملیاتی منطقی تشکیل شده است. برای مقدار مفروضی از متغیرهای دودویی، تابع می‌تواند 1 یا 0 باشد. یک تابع بول رابطه‌ای منطقی را بین متغیرها بیان می‌کند. این تابع با تعیین مقدار دودویی عبارت بر حسب همه مقادیر ممکن متغیرها ارزیابی می‌شود.

یک جدول بولی به‌صورت یک جدول درستی هم می‌تواند نشان داده شود. جدول درستی لیستی از ۱ها و 0 ها است که به متغیرهای دودویی تخصیص می‌یابد، و ستونی که مقدار نتایج را برای هر ترکیب نشان می‌دهد. تعداد سطر ها 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 به‌دست می‌آید. یک تابع بول را می‌توان از یک عبارت جبری به یک نمودار مداری متشکل از گیت‌های منطقی تبدیل کرد. برای درک بهتر موضوع دو تابع زیر را در نظر بگیرید:

$$F_1 = x + y' z$$

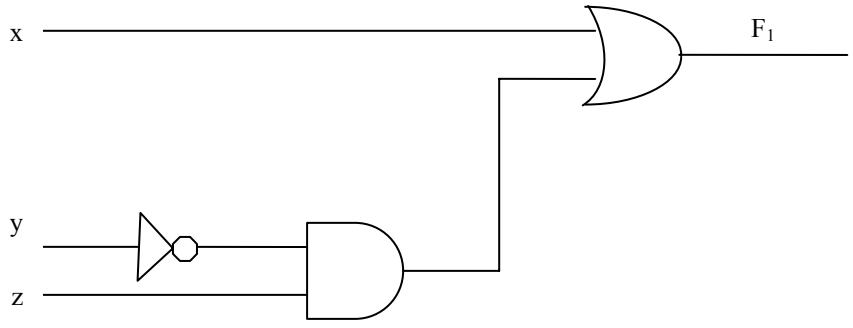
$$F_2 = x' y' z + x' y z + x y'$$

جدول شکل ۲-٦، درستی دو تابع F_1 و F_2 را نشان می‌دهد:

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

شکل ۲-٦: جدول درستی توابع سه متغیره F_1 و F_2

در این جدول هشت ترکیب دودویی ممکن برای تخصیص بیتی به سه متغیر z,y,x وجود دارد. ستونی که بر چسب F_1 دارد در ازاء هر ترکیب 0 یا 1 است. جدول نشان می‌دهد که وقتی x=1 یا yz=1 باشد تابع F1 برابر 1 است. در غیر این صورت 0 خواهد بود. 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. نمودار مدار منطقی تابع بولی F_1 در شکل۲-۷ نشان داده شده است:

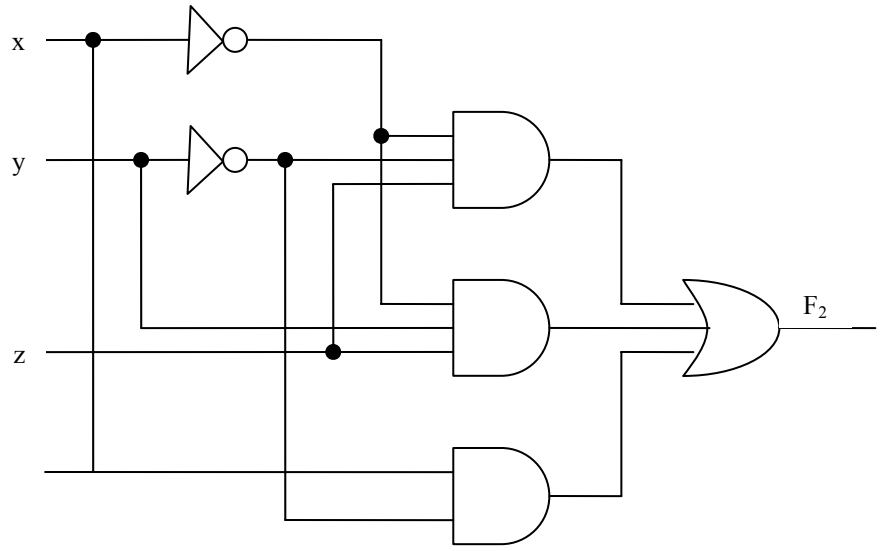


شکل۲-۷:پیاده‌سازی با گیت $F_1 = x + y' z$

برای تولید متمم ورودی y از گیت NOT استفاده شده است. برای جمله y′z یک گیت AND وبرای ترکیب آن دو یک گیت OR به‌کار رفته است. در نمودارهای مدار

منطقی، متغیرها ی تابع به عنوان ورودی مدار و متغیر دودویی F_1 به عنوان خروجی مدار در نظر گرفته می‌شوند.

در تابع بولی F2 که جدول درستی آن در بالا آمده است، متغیرهای x,y به کمک وارون‌گر متمم شده‌اند تا x' , y' به‌دست آیند. سه جمله در عبارت با سه گیت AND پیاده‌سازی شده‌اند. گیت OR نیز، OR منطقی سه جمله را فراهم می‌سازد. نمودار مدار منطقی تابع بولی F_2 در شکل ۲-۸ نشان داده شده است:



شکل ۲-۸: پیاده‌سازی تابع بول F_2 با گیت

۲-۳-۱ متمم یک تابع

متمم یک تابع F برابر F' است و از تعویض 0 ها با 1 و ۱ها با 0 در مقدار F به‌دست می‌آید. متمم یک تابع را می‌توان به‌صورت جبری از تئوری دمورگان نیز به‌دست آورد. تئوری‌های دمورگان به سه یا چند متغیر هم قابل گسترش اند. با استفاده از اصول و تئوری‌های ارائه شده، فرم سه متغیره اولین تئوری دمورگان به طریق زیر ثابت می‌شود.

با فرض $(B + C = x)$ داریم: $(A + B + C)' = (A + x)'$

با تئوری ۵ (a) دمورگان $(A + B + C)' = A'x'$

$B + C = x$ را جایگزین کنید $(A + B + C)' = A' (B + C)'$

با تئوری ۵ (a) دمورگان $(A + B + C)' = A'(B'C')$

با تئوری ٤ (b) شرکت‌پذیری $= A'B'C'$

تئوری‌های دمورگان برای هرتعدادی از متغیرها، مشابه حالت دو متغیره بوده و با روش جایگزینی متوالی، مشابه روشی که در فوق مشاهده شد، می‌توان آن را به‌دست آورد. **فرم عمومی‌تئوری 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 ۱: متمم توابع $F_1 = x' y z' + x' y' z$ و $F_2 = x(y' z' + y z)$ را به‌دست آورید. متمم‌ها را با اعمال هر تعداد تئوری دمورگان به‌صورت زیر به‌دست آورید:

$$F'_1 = (x'y z' + x'y'z)'$$
$$= (x'y z')'(x'y'z)'$$
$$= (x + y' + z)(x + y + z')$$

$$F'_2 = [x(y'z' + yz)]'$$
$$= x' + (y'z' + yz)'$$
$$= x' + (y'z')'(yz)'$$
$$= x' + (y + z)(y' + z')$$

روال ساده‌تری برای به‌دست آوردن متمم یک تابع این است که دوگان تابع و متمم هر لیترال به‌دست آید. این روش با توجه به فرم عمومی‌تئوری دمورگان نتیجه می‌شود. به خاطر داشته باشید که دوگان یک تابع با تبدیل عملگر AND به OR و تبدیل ۱ها و 0 ها به یکدیگر به‌دست می‌آید.

مثال ۲: متمم توابع F_1 وF_2 مثال ۲-۲ را با استفاده از دوگان‌ها و متمم‌های هر لیترال به‌دست آورید.

1)$F_1=x'yz'+x'y'z'$

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

متمم هر لیترال برابر است با: $(x+y'+z)(x+y+z')=F'_1$

2) $F_2= x(y\,z'+yz)$

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

متمم هر لیترال برابر است با: $x'+(y+z)(y'+z') =F'_2$

۲-۳-۲ سایر اعمال منطقی

وقتی که عملگرهای AND و OR بین دو متغیر y,x قرار می‌گیرند، به ترتیب دو تابع بولی x.y , x+y را تشکیل می‌دهند. قبلاً بیان شد که برای n متغیر $2^{\wedge}2^{\wedge}n$ 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. جدول درستی 16 تابع که با دو متغیر دودویی y , x تشکیل گردیده در جدول زیر لیست شده است. هر یک از 16 ستون F_{15} , F_0 ، جدول درستی یک تابع ممکن برای دو متغیر y,x 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 (شکل۲-۹).

x	y	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
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

شکل۲-۹: جدول درستی برای 16 تابع از دو متغیر دودویی

اگر چه هر تابع را می‌توان بر حسب عملگرهای دودویی NOT,OR,AND بیان کرد، اما دلیلی وجود ندارد که کسی عملگر خاصی را برای بیان سایر توابع تعیین ننماید. چنین عملگرهایی در ستون دوم جدول شکل۲-۱۰ 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 زیر، نام و توضیحی که تابع را به نحوی تشریح می‌کند، آورده شده است. 16 تابع لیست شده فوق به سه گروه زیر تقسیم می‌شوند.

توابع بول	سمبل عملگر	نام	توضیحات
$F_0=0$	.	Null	Binary constant 0
$F_1=xy$	x.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↓y	NOR	Not-OR
$F_9=xy+x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10}=y'$	y´	Complement	Not y
$F_{11}=x+y'$	xy	Implication	If y , then x
$F_{12}=x'$	x´	Complement	Not x
$F_{13}=x'+y$	xy	Implication	If x , than y
$F_{14}=(xy)'$	x↑y	NAND	Not-AND
$F_{15}=1$	.	Identity	Binary constant 1

شکل ۲-۱۰: عبارات بولی ١٦ تابع تعریف شده در شکل ۲-۹

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

۲- چهار تابع یکانی از نوع متمم[۱] و انتقال [۲]

۳- ده تابع باقیمانده شامل هشت عمل مختلف به شرح زیر می‌باشد:

- AND
- OR
- NAND
- NOR
- XOR (یا OR انحصاری)
- XNOR (یا هم ارزی)
- نهی[۳]
- استلزام (استنباط)[۴]

ثابت‌ها برای توابع دودویی فقط می‌توانند 0 یا 1 باشند. تابع متمم، متمم هر متغیر دودویی را تولید می‌نمایند. تابعی که برای یک متغیر ورودی است را انتقال[۵] 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 به‌کار می‌روند، ولی به ندرت در منطق کامپیوتر از آنها استفاده می‌شود. عملگرهای AND و OR قبلاً در جبر بول ذکر شدند. چهار تابع دیگر به‌طور گسترده در طراحی دستگاه‌های دیجیتال مورد استفاده‌اند.

تابع NOR متمم OR بوده و نام آن از NOT–OR اخذ شده است. به‌طور مشابه NAND، متمم AND است و از NOT–AND مشتق می‌شود. OR انحصاری یا XOR

---

[1] _ Complement

[2] _ Transfer

[3] Inhibition

[4] Implication

[5] Transfer - Buffer

مشابه با OR است ولی حالتی که در آن هر دو متغیر y,x متفقا برابر 1 باشند، را شامل نمی‌شود. تابع XNOR یا هم ارزی 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 شکل۲-۹ قابل تشخیص است. جدول درستی برای OR عبارت است از F6 و برای XNOR نیز F9 است. این دو تابع متمم یکدیگرند. به این دلیل تابع هم ارزی را NOR انحصاری هم می‌گویند و با XNOR نشان می‌دهند.

جبر بول دو عملگر دودویی با نام های AND , 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 NOT , OR , AND را بر حسب آنها تعریف کنیم. به هر حال دلایل مکفی برای روش منتخب وجود دارد و در واقع مفاهیم NOT , OR , AND 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.

۲-٤ گیت‌های منطقی دیجیتال

- چون توابع بول بر حسب عملگرهای NOT , OR , AND 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:
- امکان سنجی و اقتصادی بودن روش ساخت به هنگام استفاده از قطعات فیزیکی
- امکان گسترش ورودی گیت‌ها به بیش از دو

- در نظر گرفتن خواص اصلی عملگرهای دودویی مثل جابجایی و شرکت‌پذیری
- توانایی گیت در پیاده‌سازی توابع به تنهایی یا همراه با سایر گیت‌ها

از شانزده تابع معرفی شده در قسمت قبل، دو تابع برابر با مقدار ثابت و چهار تای 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.

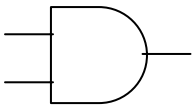
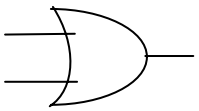
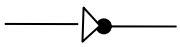
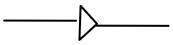
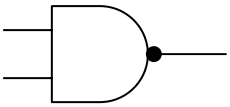
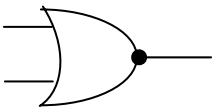
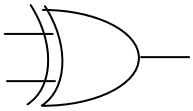
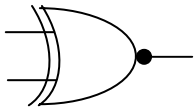
هشت تابع دیگر یعنی: Buffer، NOT، AND، OR، NAND، NOR، XOR و XNOR به عنوان گیت‌های استاندارد در طراحی سیستم‌های دیجیتال به‌کار می‌روند.

سمبل‌های گرافیکی و جدول درستی هشت گیت فوق در شکل۲-۱۱ نشان داده شده‌اند. هر گیت موجود در شکل، دارای دو متغیر ورودی دودویی y,x و یک متغیر خروجی دودویی F می‌باشد. مدارهای NOR,OR,AND از قبل تعریف 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 به تنهایی علامت بافر می‌باشد. یک بافر عمل انتقال را انجام می‌دهد، ولی یک عمل منطقی تولید نمی‌کند زیرا مقدار دودویی خروجی برابر مقدار ورودی دودویی است. این مدار صرفاً در تقویت توان سیگنال‌ها استفاده شده 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.

نام	سمبل گرافیکی	تابع جبری	جدول درستی
AND		F=XY	x y F 0 0 0 0 1 0 1 0 0 1 1 1
OR		F=x+y	x y F 0 0 0 0 1 1 1 0 1 1 1 1
Inverter		F = x′	x F 0 1 1 0
Buffer		F=x	x F 0 0 1 1
NAND		F=(xy) ′	x y F 0 0 1 0 1 1 1 0 1 1 1 0
NOR		F= (x+y) ′	x y F 0 0 1 0 1 0 1 0 0 1 1 0
Exclusive-OR (XOR)		F= xy′ + x′y = x⊕y	x y F 0 0 0 0 1 1 1 0 1 1 1 0
Exclusive-NOR or equivalence		F=xy+ x′y′= (x⊕y) ′	x y F 0 0 1 0 1 0 1 0 0 1 1 1

شکل ۲-۱۱: گیت‌های منطقی به همراه مشخصات و جدول درستی

تابع NOR هم متمم OR است و با یک سمبل OR و به دنبال آن یک حباب نمایش داده می‌شود.

گیت‌های NAND و NOR به‌طور گسترده ای به عنوان گیت‌های استاندارد مورد استفاده قرار گرفته و بیشتر OR و AND مورد توجه اند. این بدان علت است که گیت‌های NAND و NOR به سادگی به وسیله مدارات ترانزیستوری قابل تولید بوده و می‌توان به راحتی توابع بول را با آنها پیاده‌سازی کرد.

گیت NOR 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.

۲-٤-۱ گسترش ورودی گیت‌ها

گیت‌هایی که در شکل۲-۱۱ 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 ورودی‌های آن را گسترش داد. اعمال AND و OR که در جبر بول تعریف شده‌اند این خاصیت را از خود به نمایش گذاشته اند. برای تابع OR داریم:

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

و

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

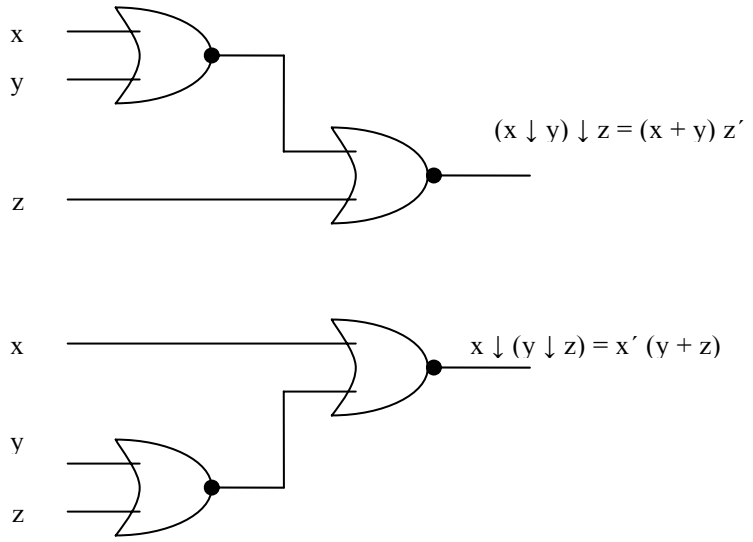
این روابط بیانگر تعویض پذیری ورودی‌های گیت و قابل گسترش بودن متغیرهای ورودی به بیش از دو در تابع OR است.

توابع NAND و NOR

توابع NAND و NOR جابجا پذیرند 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 ۲-۱۲ و معادلات زیر مشاهده می‌گردد:



شکل۲-۱۲ شرکت‌ناپذیری عملگر NOR – $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

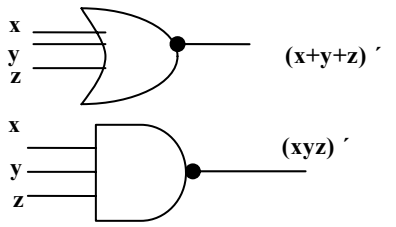
$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$

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

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

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

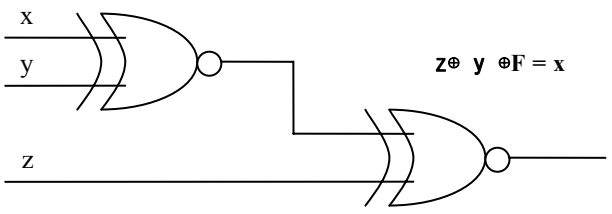
$$x \uparrow y \uparrow z = (xyz)'$$



سمبل‌های گرافیکی گیت‌های سه ورودی در شکل۲-۱۳ نشان داده شده‌اند. در نوشتن متوالی اعمال NAND,NOR باید پرانتزها به فرم صحیحی انتخاب شوند تا بیانگر ترتیب صحیح گیت‌ها باشند.

گیت‌های XOR و XNOR

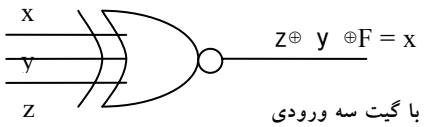
XOR و XNOR هر دو خواص جابجایی و شرکت‌پذیری را دارند و ورودی‌هایشان قابل توسعه به بیش از دو می‌باشد. با این وجود گیت‌های XOR چند ورودی از نقطه نظر سخت‌افزاری متداول نیستند. در واقع حتی فرم دو ورودی آن نیز معمولاً از سایر



(الف) با گیت‌های دوورودی

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



(ب) با گیت سه ورودی

شکل ۲-۱۳: گیت XOR

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 شکل۲-۱۳ دیده می‌شود.

این مدار معمولاً با گیت‌های دو ورودی تهیه می‌شود، شکل۲-۱۳ (الف). به‌صورت گرافیکی، آن را می‌توان با یک گیت سه ورودی نشان داد، شکل۲-۱۳ (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 ۱ها در متغیرهای ورودی فرد است، تابع 1 است.

۲-٤-۲ مدارهای مجتمع

با گسترش علم الکترونیک و طراحی مدارات ترکیبی و پیچیده نیاز به طراحی بسته‌های یکپارچه و کوچکتر از مدارها بیش از پیش احساس می‌شد. یک مدار مجتمع (IC) یک کریستال نیمه‌هادی از جنس سیلیکان است که به آن تراشه می‌گویند و حاوی اجزاء الکترونیکی در ساخت گیت‌های دیجیتال می‌باشد و 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. تعدادپایه‌ها ممکن است گاه چند هزار در یک بسته بزرگ برسد. در روی هر IC یک شماره برای شناسایی چاپ می‌شود.

۲-٤-۲-۱ سطوح مجتمع سازی

IC های دیجیتال اغلب بر اساس پیچیدگی مدار درونی‌شان که به تعداد گیت‌های منطقی مرتبط است دسته بندی می‌شوند. تفکیک تراشه‌هایی که تنها چند یا چند صد و یا چندین هزار گیت دارند با ارجاع به بسته و دسته بندی 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 بسته اند. ورودی‌ها خروجی‌های گیت‌ها مستقیماً به پایه‌های بسته متصل می‌شوند. تعداد گیت‌ها معمولاً کمتر از 10 بوده و به وسیله پایه‌های موجود در IC محدود می‌گردند.

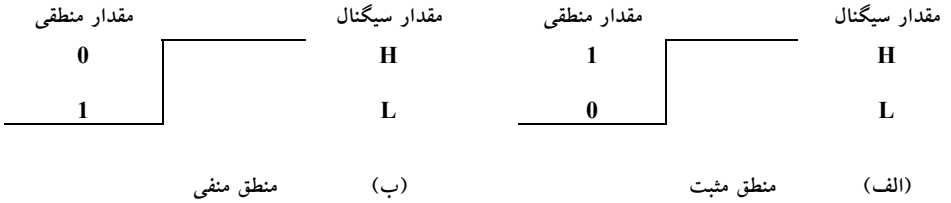
مدارهای مجتمع با فشردگی متوسط (MSI) دارای فشردگی بین 10 تا 1000 گیت در یک بسته دارند. این دسته از مدارات معمولاً اعمال دیجیتال خاصی را اجرا می‌کنند. توابع دیجیتال MSI با عناوین دیکدر ها، جمع کننده‌ها و مولتی‌پلکسرها در فصل ٤ و ثبات‌ها و شمارنده‌ها در فصل ٦ مطرح شده‌اند.

مدارهای مجتمع با فشردگی زیاد (LSI) حاوی هزاران گیت در یک بسته می‌باشند. این دسته از مدارات، پردازنده‌ها، حافظه‌ها و مدارات منطقی برنامه پذیر را شامل می‌شوند. بعضی از اجزا 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 کامپیوتر های 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.

۲-٤-۲-۲ 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، شکل۲-۱٤ سطح سیگنال بالاتر با H و سطح سیگنال



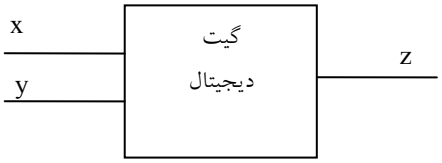
شکل۲-۱٤: تخصیص سیگنال و قطبیت منطق

پایین‌تر با L مشخص شده است. اگر سطح بالا، H، برای منطق 1 به‌کار 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 سیگنال ممکن است مثبت یا منفی باشند. در واقع، این قطب های سیگنال نیستند که بیانگر نوع منطق

می‌باشند، بلکه انتخاب مقادیر منطقی بر حسب سطوح نسبی سیگنال‌ها نسبت به هم، نوع منطق را مشخص می‌کنند.

گیت‌های دیجیتال سخت‌افزاری بر حسب مقادیر سیگنال L , H تعریف می‌شوند. از این پس انتخاب منطق مثبت و منفی به عهده کاربر است. به عنوان مثال گیت الکترونیک شکل زیر را به همراه جدول درستی آن در نظر بگیرید:

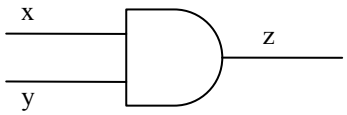


نمودار بلوکی گیت

x	y	z
L	L	L
L	H	L
H	L	L
H	H	H

جدول درستی با L و H

این جدول رفتار فیزیکی گیت را وقتی H برابر 3V و L برابر 0 ولت است نشان می‌دهد. شکل۲-۱۵ جدول درستی منطق مثبت را فرض می‌کند که در آن H=1 و L=0 است. این جدول درستی همانند جدول عمل AND با منطق مثبت در شکل زیر دیده می‌شود.



گیت AND با منطق مثبت

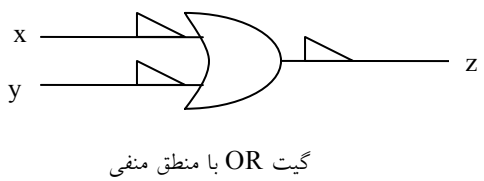
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

جدول درستی منطق مثبت

شکل۲-۱۵: نمایش جدول درستی گیت AND با منطق مثبت

اکنون تخصیص منطق منفی را برای همان گیت فیزیکی با L=1 وH=0 در نظر بگیرید. نتیجه جدول درستی شکل زیر خواهد بود. گرچه داده‌ها معکوس شده‌اند ولی

جدول عمل OR را نشان می‌دهد. سمبل گرافیکی گیت OR با منطق منفی در شکل۲-۱۶ دیده می‌شود.



گیت OR با منطق منفی

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

شکل۲-۱۶: نمایش جدول درستی گیت OR با منطق منفی

مثلث‌های کوچک در ورودی‌ها و خروجی نشانگر قطبیت هستند. وجود 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 0 ها به 1 و ۱ها 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.

۲-٤-۲-۳ خانواده‌های منطقی دیجیتال

جدا از بحث پیچیدگی و عمل منطقی مدارهای مجتمع دیجیتال که باعث دسته‌بندی آنها نیز می‌گردد، این مدارات بر اساس تکنولوژی مدار خاصی که به آن تعلق دارند نیز دسته بندی 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 به‌کار 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.

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

ECL: در سیستم‌هایی که به سرعت بالا نیاز دارد ارجحیت دارد.

TTL: مدت مدیدی است که مورد استفاده بوده و به عنوان یک گیت استاندارد شناخته شده است.

MOS: در مدارهایی که نیاز به چگالی قطعه بالایی دارند مورد استفاده است.

CMOS: در مواقعی که توان مصرفی باید کم باشد مورد توجه می‌باشد.

نظر به اینکه توان مصرفی کم در طراحی VLSI از اصول است، CMOS تبدیل به یک خانواده منطقی غالب شده است در حالی که از کاربرد خانواده‌های ECL,TTL به تدریج کاسته می‌شود.

سؤالات

۱- با استفاده از جدول درستی نشان دهید که گیتهای NAND و NOR (هر یک با سه ورودی) متمم یکدیگر هستند یا خیر؟

۲- جداول درستی توابع ذیل را تهیه کنید.

$F_1 = (x+y).\ (x'+z).(x+y'+z')$

$F_2 = x' + yz'$

۳- متمم توابع زیر را به‌دست آورید.

$F_1 = x'\ y\ z' + x'\ y'$

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

$F_3 = (xy' + z)\ x'z'$

٤- با استفاده از جدول درستی نشان دهید که گیت‌های X-OR و X-NOR (هر یک با دو ورودی x و y) متمم یکدیگرند.

۵- نمودار منطقی عبارات ذیل را رسم نمایید.

$F_1 = (x + y).\ (x' + y' + z)$

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

فصل ۳

فرم‌های متعارف و استاندارد در جبر بولی

هدف کلی

در این فصل مباحث اصلی مربوط به استانداردسازی عبارات بولی و روش‌های حداقل‌سازی عبارات بولی به منظور کاهش هزینه ساخت گیت‌ها مورد بحث و بررسی قرار خواهد گرفت. تهیه عبارات متعارف و استاندارد به عنوان هدف اصلی این فصل می‌باشد.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- فرم‌های استاندارد
- جمع حاصل ضرب‌ها
- ضرب حاصل‌جمع‌ها
- مفاهیم فرم‌های متعارف
- حداقل‌سازی سطوح گیت‌ها
- جمع مینترم‌ها
- حاصل‌ضرب ماکسترم‌ها
- تبدیل فرم‌های متعارف به یکدیگر

۳-۱ فرم‌های استاندارد

همانطور که می‌دانید طراحان قادر هستند بر اساس نیاز خود اقدام به تعریف توابع بولی متنوع نمایند که این توابع بولی می‌توانند بر اساس دو عملگر (+) و (.) دسته‌بندی شوند. با گسترش دامنه استفاده از توابع بولی در ساختن مدارات منطقی دیجیتال، طراحان را با شرایطی مواجه کرد که در آن توابع دارای حداقل متغیرها نبودند. لذا لزوم داشتن استاندارد نگارش عبارات بولی بیش از پیش احساس شد. بر این اساس محققین با ارائه الگوهایی، سعی کردند روش‌های استانداردی را برای نوشتن عبارات بولی طراحی نمایند. 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:

- جمع حاصل‌ضرب‌ها
- ضرب حاصل‌جمع‌ها

۳-۱-۱ جمع حاصل‌ضرب‌ها

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

مینترم‌ها

x	y	z	جمله	علامت
0	0	0	x ′y′z′	m_0
0	0	1	x ′y′z	m_1
0	1	0	x ′yz′	m_2
0	1	1	x ′yz	m_3
1	0	0	x y′z′	m_4
1	0	1	x y′z	m_5
1	1	0	x yz′	m_6
1	1	1	x yz	m_7

شکل ۳-۱: جدول نمایش مینترم‌ها برای سه متغیر

متصور است: xy,xy′,x′y,x′y′. هر یک از این چهار جمله AND را یک مینترم یا یک جمله ضرب استاندارد گویند. به‌طور مشابه n متغیر را می‌توان ترکیب کرده و $2^{\wedge n}$ مینترم به وجود آورد. $2^{\wedge n}$ مینترم مختلف را می‌توان با روشی مشابه با آنچه در شکل۳- ۱ آمده، نشان داد.

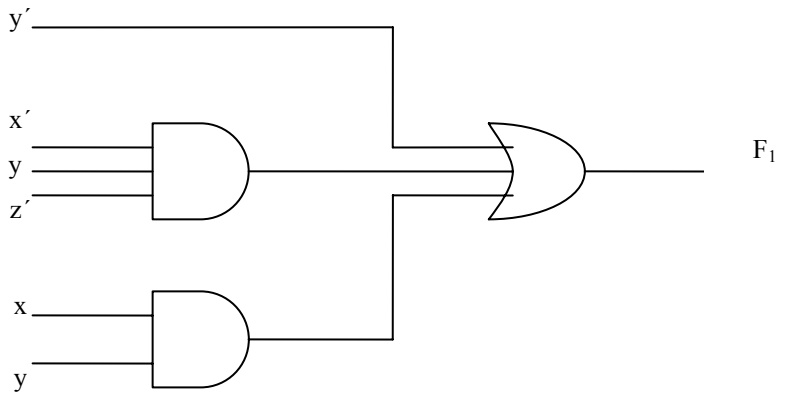
اعداد دودویی از صفر تا $2^{\wedge n}$ -1 زیر ستون n متغیر لیست شده‌اند. هر مینترم از AND تمام n متغیر به‌دست می‌آید که در آن هر متغیر پریم‌دار متعلق به بیت 0 و بدون پریم با 1 نشان داده می‌شود. سمبل هر مینترم نیز در جدول با m_j نشان داده شده است. که در آن J معادل دهدهی عدد دودویی مربوط به مینترم است.

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 به صورت 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.



شکل۳-۲: مدار منطقی عبارت $F_1 = y' + xy + x'y\,z'$

نمودار منطقی جمع حاصلضرب‌ها متشکل از گروهی گیت AND است که بدنبال یک گیت OR می‌آید. الگوی این آرایش در شکل ۳-۲ آمده است. هر جمله ضرب نیاز به یک گیت AND دارد. این نکته در یک ورودی تک لیترال 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.

۳-۱-۲ ضرب حاصل جمع‌ها

همانند آنچه که در بالا گفته شد، 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 ۳-۳ لیست شده‌اند. هر 2^n ماکسترم برای n متغیر به‌طریق مشابهی حاصل می‌شود. هر ماکسترم از یک جمله OR با n متغیر به‌دست 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.

ماکسترم‌ها

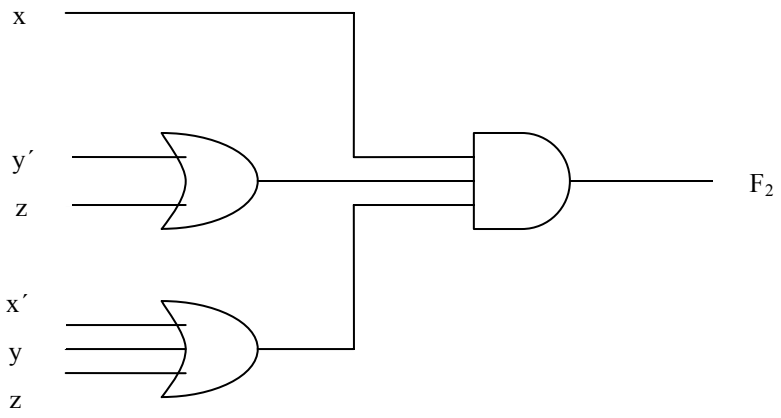
x	y	z	جمله	علامت
0	0	0	x +y+z	m_0
0	0	1	x +y+z´	m_1
0	1	0	x +y´+z	m_2
0	1	1	x +y´+z´	m_3
1	0	0	x´ +y+z	m_4
1	0	1	x´ +y+z´	m_5
1	1	0	x´ +y´+z	m_6
1	1	1	x´ +y´+z´	m_7

شکل۳-۳ : جدول نمایش ماکسترم‌ها برای سه متغیر

ضرب حاصل جمع‌ها، یک عبارت بولی حاوی جملات OR است که به آن جملات جمع می‌گویند. هر جمله می‌تواند به هر تعداد لیترال داشته باشد. ضرب به معنی AND این جملات است.

مثالی از یک تابع به‌صورت ضرب حاصل جمع‌ها چنین است:

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



شکل ۳-٤: مدار منطقی عبارت $F_2 = x\,(y' + z)\,(x' + y + z')$

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 شکل۳-٤ دیده می‌شود، این نوع استاندارد عبارت به یک ساختار دو سطحی (یا دو طبقه) از گیت‌ها منجر می‌گردد.

۳-۱-۳ مفهوم فرم‌های متعارف

یک تابع بول می‌تواند به‌صورت جبری با استفاده از جدول درستی و با تشکیل مینترم‌های هر ترکیب از متغیرهایی که برای تابع، 1 را تولید می‌کنند، و اجرای هر عملگر روی OR همه این جملات ایجاد شود.

مثلاً F_1 در جدول زیر با ترکیبات 001 , 100 , 111 به‌صورت $x'y'z$, $xy'z'$, xyz بیان می‌شود. چون هر یک از 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:

$$F_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

x	y	z	F_1	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

این مثال‌ها خصوصیت مهمی‌از جبر بول را به نمایش می‌گذارند: یعنی هر تابع بولی را می‌توان به‌صورت 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 به‌دست آوریم تابع F1 به‌دست خواهد آمد.

$$F_1 = (x+y+z)\,(x+y'+z)\,(x'+y+z')\,(x'+y'+z) = m_0.m_2.m_3.m_5.m_6$$

به‌طور مشابه، می‌توان عبارت F2 را از جدول به‌دست آورد:

$$F_2 = (x+y+z)\,(x+y+z')\,(x+y'+z)\,(x'+y+z) = m_0\, m_1\, m_2\, m_4$$

این مثال‌ها نیز دومین خاصیت جبر بول را به نمایش می‌گذارند: هر تابع بول را می‌توان به‌صورت ضرب ماکسترم‌ها (ضرب به معنی AND جملات است) درآورد. 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 به‌دست آورید. توابع بول که به‌صورت جمع مینترم‌ها یا ضرب ماکسترم بیان شوند را **فرم متعارف** نامند.

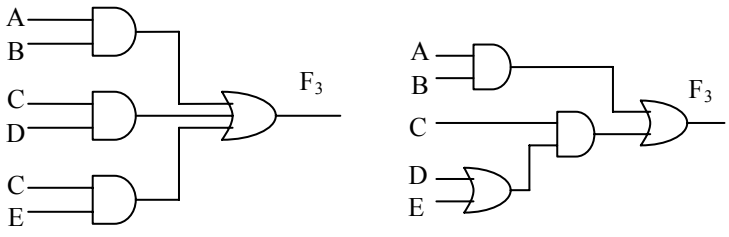
۳-۱-٤ حداقل سازی سطوح گیت

یک تابع بول ممکن است به‌صورت غیر استاندارد نیز بیان شود. مثلاً تابع F_3 که در زیر آمده است نه جمع حاصلضرب و نه ضرب حاصل جمع است.

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

پیاده‌سازی این عبارت در شکل ۳-۵ (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.

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



AB + CD + CE (ب) AB + C (D +E) (الف)

شکل ۳-۵: پیاده‌سازی سه و دو سطحی

عبارت جمع حاصلضرب در شکل ۳-۵ (ب) پیاده شده است. به‌طور کلی، یک پیاده‌سازی دو سطحی ترجیح داده می‌شود. زیرا به هنگام انتشار ورودی‌ها به سمت خروجی‌ها حداقل مقدار تاخیر را در گیت تولید می‌کند.

۳-۱-۵ مجموع مینترم‌ها

قبلاً بیان شد که برای هر n متغیر دودویی 2^{n} مینترم مجزا وجود دارد و هر تابع بولی می‌تواند به‌صورت مجموعی از مینترم‌ها در آید. مینترم‌هایی که جمع آنها توابع بول را تعریف می‌کنند، آنهایی هستند که ۱های 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 بر حسب جمع مینترم‌ها بیان کرد. اگر در این فرم نبود، می‌توان ابتدا آن را به‌صورت 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 ۱: تابع بولی F=A+B′C را به‌صورت جمع مینترم‌ها در آورید. تابع سه متغیر C,B,A دارد. در اولین جمله A، دو متغیر مفقود است ؛ بنابراین:

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

این تابع هنوز هم یک متغیر کسر دارد

$$A = AB\,(C + C') + AB'(C + C')$$
$$= ABC + ABC' + AB'C + AB'C'$$

جمله دوم B′C یک متغیر کم دارد

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

با ترکیب همه جملات داریم:

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

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

$$F = A'B'C + AB'C + AB'C + ABC' +ABC$$
$$= m_1 + m_4 + m_5 + m_6 + m_7$$

گاهی بهتر است تابع بول را وقتی به‌صورت 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 زیر را در نظر بگیرید:

$$F = A + B'C$$

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

شکل۳-٦ : جدول درستی برای $F = A + B'C$

جدول درستی در جدول ۳-٦ مستقیماً از عبارت جبری با لیست هشت ترکیب زیر متغیرهای A، C,B 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 و7 می‌باشند.

۳-۱-٦ ضرب ماکسترم‌ها

هر یک از $2^{\wedge 2^{\wedge n}}$ تابع متشکل از n متغیر را می‌توان به‌صورت 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 استفاده از قانون توزیع‌پذیری انجام داد. سپس هر متغیر مفقود در هر جمله OR با ´xx، OR می‌شود. روش با مثال زیر روشن‌تر خواهد شد.

$$x + (yz) = (x+y)(x+z)$$

مثال ۲: تابع بول $F = xy + x'z$ را به‌صورت ضرب جملات ماکسترم نشان دهید. ابتدا تابع را با استفاده از اصل توزیع‌پذیری به فرم جملات OR در آورید:

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

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

$$x'+ y = x'+ y + zz' = (x'+y+z)(x'+y+z')$$
$$x + z = x + z + yy' = (x+y+z)(x+y'+z)$$
$$y + z = y + z + xx' = (x+y+z)(x'+y+z)$$

با ترکیب همه جملات و حذف تکراری‌ها، خواهیم داشت:

$$F = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')$$
$$= M_0 M_2 M_4 M_5$$

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.

۳-۲ تبدیل فرم‌های متعارف به یکدیگر

متمم یک تابع که به‌صورت 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 اگرمتمم F' را با روش تئوری دمورگان به‌دست آوریم، F را به فرم متفاوتی خواهیم داشت:

$$F = (m_0 + m_2 + m_3)' = m'_0 . m'_2 . m'_3 = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

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 j ام، متمم مینترم j ام است و بالعکس.

آخرین مثال تبدیل یک تابع مینترمی‌به معادل ماکسترمی‌را 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 باهم عوض کنید و شماره‌های مفقود شده را از فرم اصلی اصلی تابع، لیست نمایید. برای یافتن جملات مفقود، باید بدانیم که تعداد کل جملات $2^{\wedge n}$ است، که در آن n تعداد متغیرها ی 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 ۳-۷ به‌دست می‌آوریم. 1های زیر ستون F از ترکیب xy = 11 یا xz = 01 به‌دست می‌آیند. مینترم‌های تابع در جدول درستی شماره‌های 1، 3، 6 و 7 می‌باشند. تابع بر حسب مجموع مینترم‌ها چنین است:

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

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

شکل۳-۷ : جدول درستی برای تابع $F = xy + xz$

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 بر حسب ضرب ماکسترم‌ها بیان شود برابر زیر است:

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

سؤالات

۱- تابع بولی زیر را به‌صورت جملات جمع ماکسترم نشان دهید.

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

۲- متمم توابع زیر را به‌صورت جمع مینترم‌ها بنویسید.

$F\ (x,y,z) = II(0,1,5,7)$

$F\ (x,y,z,w) = II(0,2,4,11,14)$

$F(x,y,z) = \Sigma\ (1,4,5,6,7)$

$F(x,y,z,w) = \Sigma\ (0,3,5,9,12,13)$

۳- اگر تابع $F_1(x,y,z)$ به‌صورت زیر باشد، متمم تابع F_1 را به‌دست آورید.

$F_1\ (x,y,z) = M_0.M_2.M_5$

٤- تابع ذیل را با حداقل تعداد سطح پیاده سازی و نمودار منطقی آنرا رسم نمایید.

$F_1 = AB' + C\ (\ D + E\ ) + AD'$

فصل ٤

ساده کردن عبارات بولی پیچیده

هدف کلی

در این فصل مباحث اصلی مربوط به ساده کردن عبارات بولی پیچیده با استفاده از منطق جدول کارنو مورد بحث و بررسی قرار خواهد گرفت. عناصر اصلی جدول کارنو بررسی شده و جدول‌های کارنو با تعداد دو الی پنج متغیر مورد بحث قرار خواهند گرفت.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می گیرند

- ساده‌سازی با استفاده از نقشه کارنو
- عناصر اصلی جدول کارنو
- نقشه های دو الی پنج متغیره کارنو
- بررسی حالات بی‌اهمیت در جدول کارنو

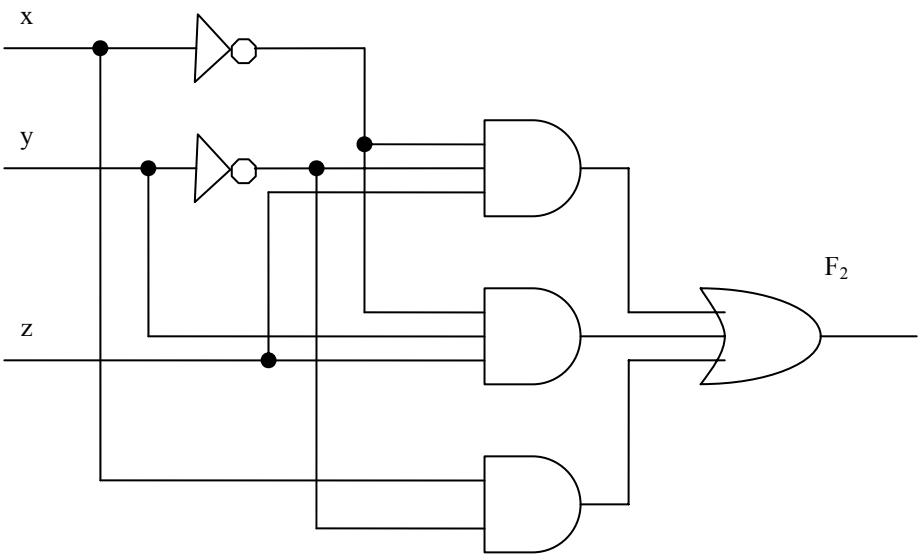
٤-١ دستکاری جبری

گاهی اوقات ممکن است با دستکاری یک عبارت بولی توسط قوانین جبر بول، عبارت ساده‌تری برای یک تابع به‌دست آوریم. 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. بنابراین با ساده کردن عبارات، تعداد گیت‌ها در مدار و تعداد ورودی‌ها به هر گیت را کاهش دهیم. مثلاً تابع بولی زیر را در نظر بگیرید:

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

مدار منطقی این تابع که دارای سه جمله و هشت لیترال است که در شکل زیر نشان داده شده‌است. منظور از لیترال، یک متغیر تک در یک جمله‌است که ممکن است متمم شود یا نشود.



شکل٤-۱: مدار منطقی سه سطحی تابع $F_2 = xy'z + x'yz + xy'$

آنگونه که مشاهده می‌کنید متغیرهای y,x به کمک وارون‌گر متمم شده‌اند تا x' , y' به‌دست آیند. سه جمله در عبارت با سه گیت AND پیاده‌سازی شده‌اند. گیت OR نیز، OR منطقی سه جمله را فراهم می‌سازد. جدول درستی تابع F_2 در زیر آمده‌است:

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. اکنون ساده‌سازی ممکن برای تابع را با اعمال بعضی از

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

شکل ٤-۲: جدول درستی تابع F_2

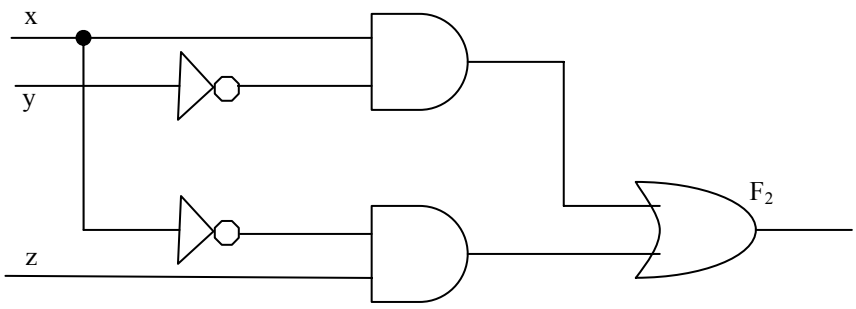
ویژگی‌های جبر بول ملاحظه کنید:

$$F_2 = x'y'z + x'yz + xy' \qquad = x'z\,(y' + y) + xy'$$
$$. \qquad = x'z + xy'$$

همانگونه که مشاهده می‌کنید تابع جدید دو جمله و چهار لیترال دارد.

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

مدار منطقی عبارت ساده شده به صورت زیر می‌باشد:



شکل٤-۳: مدار منطقی تابع F_2 به‌صورت ساده شده

تابع تنها به دو جمله کاهش یافته و قابل پیاده‌سازی با گیت مطابق شکل ٤-۳ است. بدیهی است که‌این مدار از مدار اولیه ساده‌تر 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.

در ادامه برای درک بهتر موضوع مثال‌های دیگری ارائه می‌گردد:

مثال ۱: توابع بولی زیر را با حداقل لیترال‌ها ساده کنید.

۱- $x\,(x' + y)$

$$\begin{aligned} x\,(x' + y) &= xx' + xy \\ &= 0 + xy \\ &= xy \end{aligned}$$

۲- $x + x'y$

$$\begin{aligned} x + x'y &= (x + x')\,(x + y) \\ &= 1\,(x + y) \\ &= x + y \end{aligned}$$

۳- $(x + y)(x + y')$

$$\begin{aligned} (x + y)(x + y') &= x + xy + xy' + yy' \\ &= x(1 + y + y') \\ &= x \end{aligned}$$

٤- $xy + x'z + yz$

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

٤-۲ ساده‌سازی با استفاده‌از نقشه کارنو

توابعی که تا پنج متغیر دارند قابل ساده‌سازی با روش جدول کارنو هستند. برای توابع بول پیچیده‌تر، طراحان دیجیتال از برنامه‌های کامپیوتر کوچک‌سازی استفاده می‌کنند. تنها روش موجود، روال سعی و کاهش می‌باشد که از روابط ساده و تکنیک‌های دستکاری آشنا استفاده می‌کند.

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. عبارت بول را می‌توان به صورت جبری ساده کرد. با این وجود، این روش حداقل سازی 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 عبارت جبری مختلفی برای یک تابع به‌دست آورده و از میان آنها ساده‌ترین را انتخاب کند.

عبارت ساده شده حاصل از نقشه همیشه به یکی از دو فرم استاندارد زیر می‌باشد:

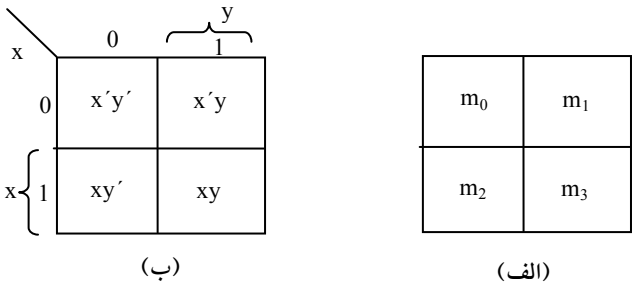
- جمع حاصلضرب‌ها

- ضرب حاصل‌جمع‌ها

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 حداقل سازی را برآورد. در این حالت هر یک از دو حل رضایت بخش خواهد بود. در ادامه روش حل جدول کارنو برای حداکثر پنج متغیر توضیح داده خواهد شد. لازم به ذکر است که روش نقشه کارنو صرفاً برای توابع دارای دو الی پنج متغیر کاربرد دارد و برای توابع با تعداد متغیر بیشتر قابل استفاده نیست.

٤-٢-١ نقشه دو متغیره کارنو

نقشه دو متغیره در شکل٤-٤ نشان داده شده‌است. 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. 0 , 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.



شکل ٤-٤: نقشه دو متغیره کارنو

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 به‌دست می‌آید. به عنوان مثال تابع xy در شکل٤-٤ (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 ٤-٤ (ب) نشان داده شده‌است که در آن سه مربع با 1 علامت زده شده‌اند. این مربعات تابع به دست آمده‌اند:

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

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

٤-٢-٢ نقشه سه متغیره کارنو

یک نقشه سه متغیره در شکل ٤-٥ مشاهده می‌شود. برای سه متغیر هشت مینترم وجود دارد. بنابراین نقشه‌از هشت مربع تشکیل یافته‌است. توجه کنید که مینترم‌ها بر اساس ترتیب دودویی مرتب نشده‌اند. بلکه ترتیب این است که هنگام عبور از یک ستون به ستون مجاور تنها یک بیت از نظر مقدار تغیر می‌کند. برای نشان دادن رابطه بین مربع‌ها و سه متغیر نقشه، بخش (ب) با اعدادی در هر سطر و هر ستون علامت‌گذاری شده‌است. 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.

	y			
yz x	00	01	11	10
0	x′y′z′	x′y′z	x′yz	x′yz′
1	xy′z′	xy′z	xyz	xyz′

x

z

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

شکل ٤-٥: نمایش توابع در نقشه

به طریقی دیگر هم می‌توان به مربع m5=xy′z نگاه کرد به‌این 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 جدول 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_7,m_5 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 به‌اصول 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_7,m_5 را ملاحظه کنید.

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

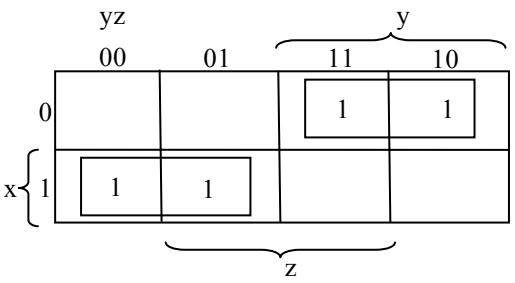
در اینجا دو مربع در متغیر y با هم اختلاف دارند که هنگام تشکیل جمع دو مینترم حذف می‌شود. بنابراین هر دو مینترمی‌که در دو مربع مجاور با هم OR شوند موجب حذف متغیری می‌گردند که در آن دو مینترم متفاوت‌اند. مثال‌های زیر روال حداقل‌سازی یک تابع بول را با یک نقشه توضیح می‌دهد.

مثال۲: تابع بولی زیر را ساده کنید.

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

ابتدا در هر مربعی که مینترم تابع را نشان دهد، مقدار ۱ قرار می‌دهیم. این کار در شکل٤-٦ به‌این ترتیب انجام شده‌است که مربعات مینترم‌های 010, 011, 100, 101 با 1

علامت زده شده‌اند. قدم بعدی یافتن مربع‌های مجاور است. این کار در نقشه با زیر مربع‌هایی که هر یک دو عدد 1 را در بر می‌گیرند صورت گرفته‌است. زیر مربع 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



شکل ٤-٦: جدول کارنو تابع $F(x , y , z) = \Sigma (2 , 3 , 4 , 5)$

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.

مواردی وجود دارد که در آنها دو مربع همجوارند ولی به هم نچسبیده‌اند. در شکل٤-٥، m_0 مجاور m_2 , 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.

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

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 ۳: تابع بول زیر را ساده کنید.

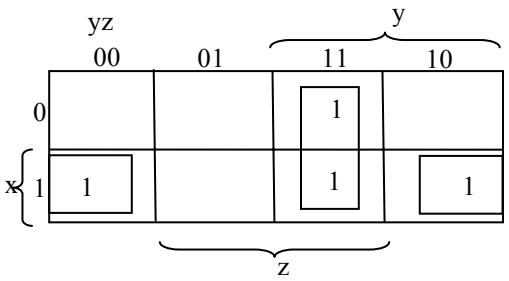
$$F(x , y , z) = \Sigma (3 , 4 , 6 , 7)$$

نقشه این تابع در شکل٤-۷ ترسیم شده‌است. 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. دو مربع باقیمانده هم بر اساس تعریف جدید مجاورند و در نمودار با نیم مربع‌ها محصور شده‌اند. این دو مربع، وقتی ترکیب شوند جمله دو لیترالی xz' را به‌دست 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 6,4,2,0 عبارت را به جمله یک لیترالی z' کاهش می‌دهد.

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



شکل ٤-٧: جدول کارنو تابع $F(x , y , z) = \Sigma (3 , 4 , 6 , 7)$

در حل جدول کارنو لازم است تا به نکات زیر توجه گردد:

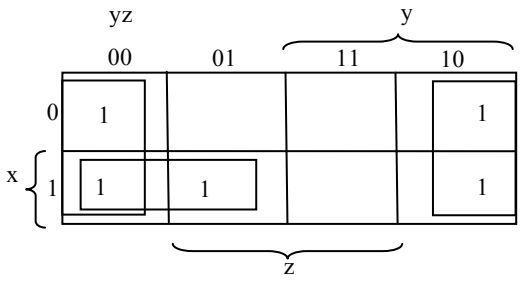
- تعداد مربعات مجاوری که ممکن است ترکیب شوند همواره برابر توانی از 2، مانند 8,4,2,1 می‌باشد.
- هر چقدر تعداد بیشتری از مربعات همجوار ترکیب شوند جمله حاصلضرب نتیجه، تعداد کمتری لیترال خواهد داشت.
- یک مربع یک مینترم را نمایش می‌دهد و دارای سه لیترال است.
- دو مربع مجاور یک جمله دو لیترال را نشان می‌دهند.
- 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 ۴: تابع بول زیر را ساده کنید.

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

نقشه تابع F در شکل ٤-٨ نشان داده شده‌است. ابتدا چهار مربع مجاور در اولین و آخرین ستون را با هم ترکیب می‌کنیم تا جمله تک لیترال z' به دست آید. تنها مینترم باقیمانده که متعلق به مینترم 5 است با مربع 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.

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



شکل ٤-۸: جدول کارنو تابع $F(x , y , z) = \Sigma\ (0, 2, 4, 5, 6)$

اگر تابعی به صورت مجموع مینترم‌ها بیان نشود، می‌توان از نقشه برای به‌دست 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 ۵: 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.

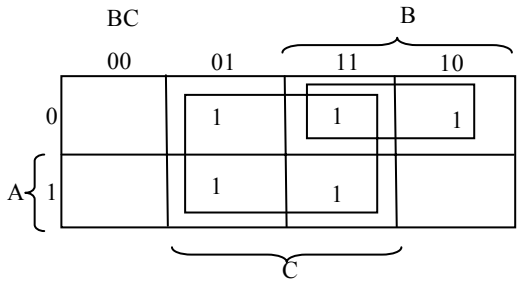
سه جمله ضرب در عبارت دو لیترال دارند و در نقشه سه متغیره، هر یک با دو مربع نشان داده شده‌اند.

دو مربع مربوط به جمله‌اول،$A'C$ ، در شکل ٤-۹ از تلاقی A' (اولین سطر) و C (دو ستون میانی) به‌دست می‌آید تا مربعات 001 و 011 را بدهند. توجه کنید که وقتی 1ها را در مربعات می‌گذارید، ممکن است یک 1 را که‌از 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 جمله‌اول، $A'C$، 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 ٤-٩ هم با پنج عدد 1 نشان داده شده‌است. مینترم‌هایی که مستقیماً از نقشه خوانده می‌شوند عبارتند از 1، 2، 3، 5 و7. تابع را می‌توان به صورت جمع مینترم‌ها نشان داد.

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



شکل٤-٩: جدول کارنو تابع $F = A'C + A'B + AB'C + BC$

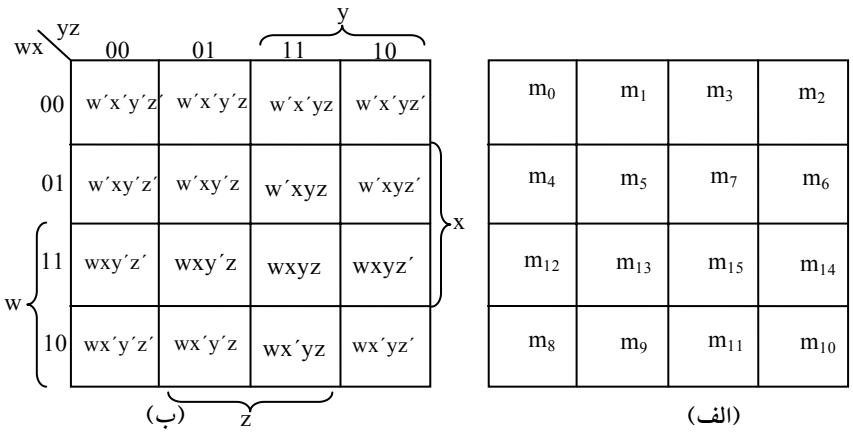
عبارت جمع حاصلضرب مفروض اولیه چندین جمله دارد. همانطور که در نقشه مشاهده می‌شود می‌توان آن را ساده کرده و عبارتی دو جمله‌ای به‌دست آورد.

$$F = C + A'B$$

٤-٢-٣ نقشه چهار متغیره

نقشه توابع بول چهار متغیره در شکل٤-١٠ نشان داده شده‌است. در (الف) 16 جمله مینترم فهرست شده به هر یک مربعی تخصیص داده شده‌است. در (ب) نقشه دوباره رسم شده تا بیانگر ارتباط بین چهار متغیر باشد. سطرها و ستون‌ها بر اساس کد گری 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 به‌دست 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.



شکل ٤-۱۰: جدول چهار متغیره کارنو

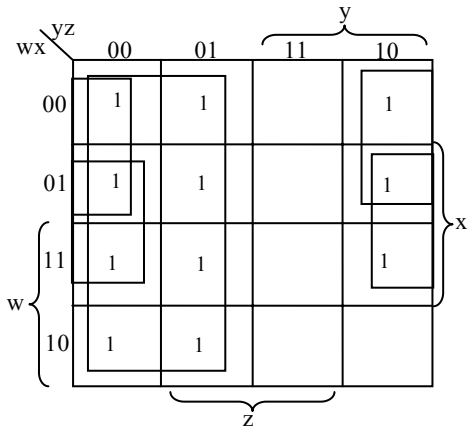
ساده کردن توابع بول چهار متغیره مشابه با روش به کار رفته برای توابع سه متغیره‌است. مربعات مجاور مربعاتی هستند که در کنار یکدیگرند. به علاوه نقشه در سطحی واقع است و لبه‌های بالا و پایین و چپ و راست نیز مجاور است تا به‌این ترتیب مربعات همجوار را بسازند. مثلاً m_0 و m_2 و نیز m_{11} و m_3 هر کدام مربعات مجاور را می‌سازند. ترکیب مربعات همجوار به راحتی با بررسی نقشه چهار متغیره قابل تشخیص است. در جدول کارنو چهار متغیره نکات زیر باید مورد توجه قرار گیرند:

- 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.

مثال ۶: 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 متغیره‌استفاده کرد. مینترم‌های لیست شده در مجموع فوق با 1ها در نقشه شکل ٤-١١ علامت 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.



شکل ٤-١١: جدول کارنو مثال ۵

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) باقیمانده‌است. در عوض انتخاب این مربع به تنهایی، آن را با مربع‌هایی که قبلاً به 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:

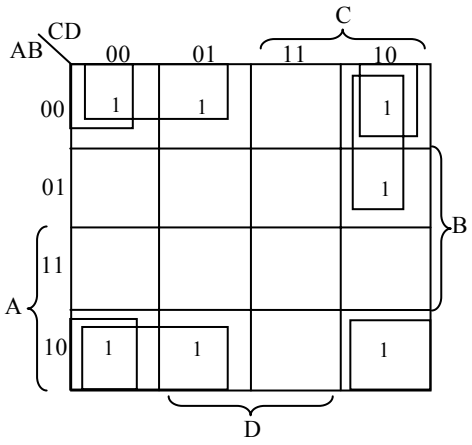
$$F = y' + w'z' + xz'$$

مثال ۷: 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 ٤-۱۲ با 1 علامت زده شده‌است. این تابع دارای چهار متغیره 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 شده‌است. مثلاً $A'B'C'$ در مربعات 0000 , 0001 نشان داده شده‌است. تابع را می‌توان با انتخاب چهار 1 در گوشه‌ها و ترکیب آنها برای به‌دست 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 و 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.

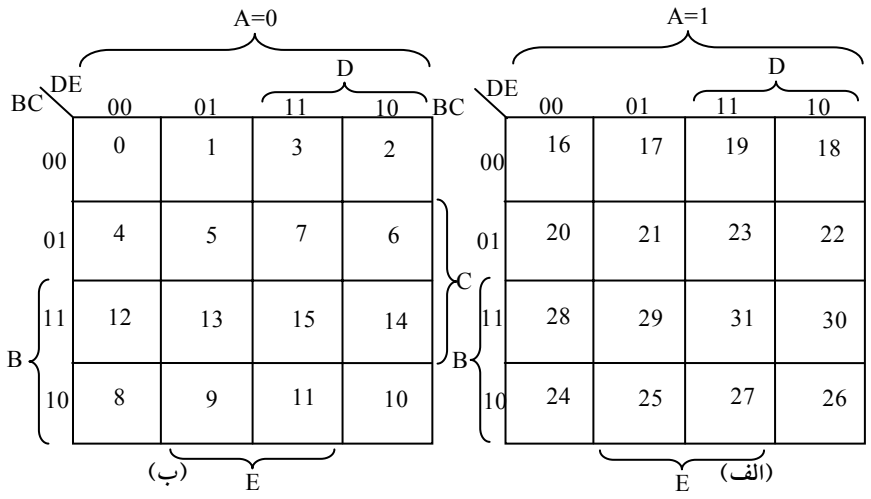
$$F = B'D' + B'C' + A'CD'$$



شکل ٤-۱۲: جدول کارنو مثال ٦

٤-۲-٤ نقشه پنج متغیره کارنو

استفاده‌از 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 شکل٤-۱۳ نشان داده شده‌است. این نقشه، از دو نقشه چهار متغیره با متغیرهای E , D , C , B , A تشکیل یافته و متغیر A آن دو را از هم تفکیک کرده‌است. 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



شکل ٤-۱۳: جدول پنج متغیره کارنو (دو جدول مجزا)

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 که‌این 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 به‌دست آورد تا 64 مربع مورد نیاز حاصل گردد. نقشه‌هایی با شش یا تعداد بیشتری متغیر، نیاز به تعداد بی شماری مربع داشته و استفاده از آنها غیر عملی است. روش دیگر، استفاده‌از برنامه‌های کامپیوتری در ساده‌سازی توابع بول با متغیرهای بی شمار می‌باشد.

با بررسی و در نظر گرفتن تعریف جدید همجواری مربعات، می‌توان نشان داد که $2^{\wedge k}$ مربع همجوار به‌ازاء $k = (0 , 1 , 2 , \ldots , n)$ در یک نقشه n متغیره ناحیه را مشخص می‌کند که نمایش دهنده 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 یکانی (1) را تولید می‌کند. جدول٤-١٤ 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.

	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**

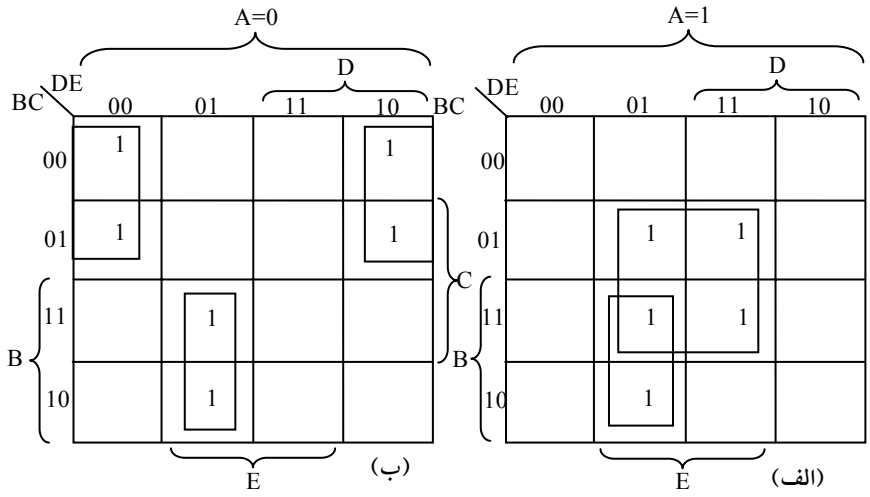
شکل ٤-١٤: رابطه بین تعداد مربعات مجاور و تعداد لیترال‌ها در یک

مثال ۸ : تابع بول زیر را ساده کنید.

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

نقشه پنج متغیره برای این تابع در شکل ٤-۱۵ 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



شکل ٤-۱۵: نقشه مثال ۷

متغیر A آورده نشده‌است زیرا مربع‌های مجاور به هر دو A=0 و A=1 متعلق اند. جمله ACE از چهار مربع همجوار در نقشه A=1 به‌دست می‌آید. تابع ساده شده جمع منطقی سه جمله می‌باشد.

$$F = A′B′E′ + BD′E + ACE$$

٤-۲-۵ عناصر اصلی در جدول کارنو

هنگام انتخاب مربع‌های مجاور در یک نقشه باید مطمئن شویم که همه مینترم‌های تابع هنگام 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 صحه بگذارند. روش ترکیب مربع‌ها در نقشه را می‌توان سیستماتیک‌تر کرد به شرطی که مفهوم جملات عناصراصلی و عناصراصلی اساسی خوب فهمیده شوند.

یک **عنصر اصلی** جمله‌ای حاصلضربی است که از ترکیب حداکثر مربعات مجاور به هم حاصل می‌گردد. اگر مینترمی در یک مربع تنها با یک عنصر اصلی پوشش یابد، به آن عنصر اصلی اساسی گوییم.

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

مثال ۹: تابع چهار متغیره زیر را در نظر بگیرید:

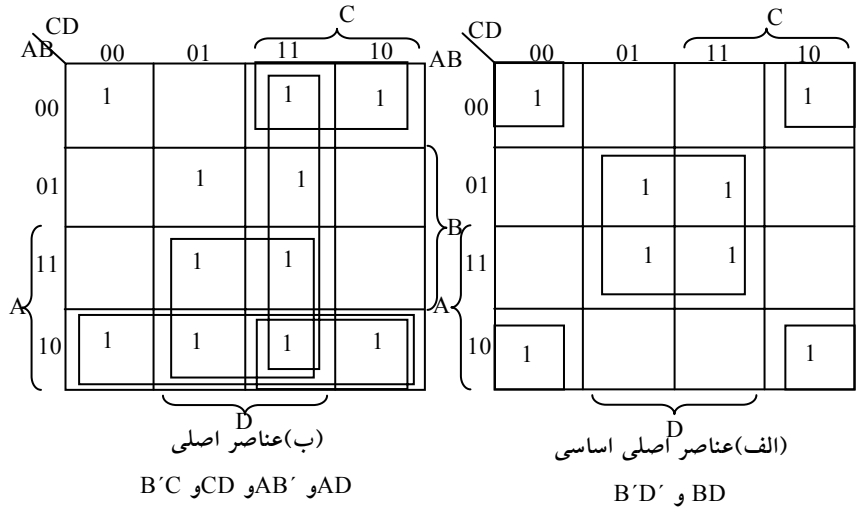
$$F(A, B, C, D) = \Sigma\,(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

مینترم‌های تابع با 1 در نقشه‌های شکل ٤-١٦ علامت زده شده‌اند. بخش (الف) از شکل، دو عنصر 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 عنصر اصلی اساسی هشت مینترم را پوشش می‌دهند. سه مینترم باقیمانده m_3 و m_9 و m_{11} باید بعد ملاحظه شوند.

شکل ٤-١٦ (ب) همه راه‌های ممکن که سه مینترم با عناصراصلی پوشش می‌یابند را نشان می‌دهد. مینترم m_3 می‌تواند با عنصر اصلی CD یا B′Cپوشش یابد. مینترمm_9 با هر یک از AD یا AB′ پوشش می‌یابد. مینترم m_{11} نیز با هر یک از چهار عنصر اصلی می‌تواند پوشش پیدا کند. عبارت ساده شده‌از جمع منطقی دو عنصر اصلی اساسی، و هر دو عنصر اصلی دیگر که مینترم‌های m_3 و m_9 و m_{11} را پوشش دهند به‌دست می‌آید. چهار امکان برای بیان تابع با چهار جمله ضرب که هر یک دو لیترال دارند وجود دارد:

$$F = BD + B'D' + CD + AD$$
$$= BD + B'D' + CD + AB'$$
$$= BD + B'D' + B'C + AD$$
$$= BD + B'D' + B'C + AB'$$



شکل ٤-١٦: نمایش ساده‌سازی با استفاده‌از عناصر اصلی مثال ۸

مثال فوق نشان داد که شناسایی عناصراصلی در نقشه در تعیین صور متفاوت تابع ساده شده کمک موثری می‌نمایند.

روال یافتن عبارت ساده شده‌از نقشه لازم می‌دارد که‌ابتدا تمام عناصراصلی اساسی را معین کنیم. تابع ساده شده از جمع منطقی همه عناصراصلی اساسی، به علاوه دیگر عناصراصلی حاصل می‌گردد. این عناصر اصلی ممکن است برای پوشش مینترم‌های باقیمانده‌ای که در عنصر اصلی اساسی وجود ندارد 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.

٤-۳ ساده‌سازی با ضرب حاصل جمع‌ها

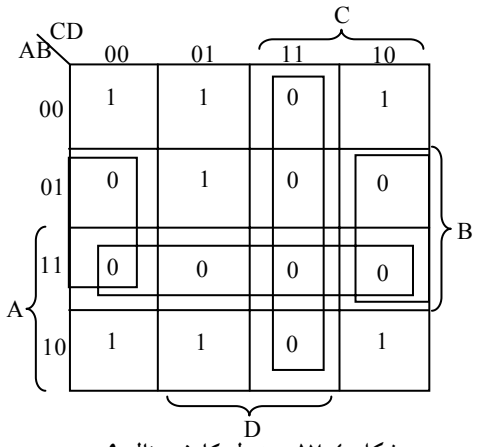
در تمام مثال‌های قبلی، توابع بول حاصل از نقشه به فرم جمع حاصلضرب‌ها بیان شدند. با کمی‌اصلاح می‌توان فرم ضرب حاصل جمع‌ها را به‌دست آورد.

روال تهیه یک تابع حداقل بر حسب 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 به‌این مطلب مشاهده می‌کنیم که متمم یک تابع به وسیله مربع‌هایی که با 1 علامت زنی 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 به‌دست 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. برای درک بهتر موضوع، مثالی در این رابطه‌ارائه می‌گردد:

مثال ۱۰: 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 شکل٤-۱۷، همه مینترم‌های تابع را نمایش می‌دهند. مربع‌هایی که با 0 علامت‌زده شده‌اند مینترم‌های غایب در F را نشان می‌دهند، بنابراین متمم F را بیانگر هستند.



شکل ٤-۱۷: جدول کارنو مثال ۹

ترکیب مربعات حاوی 1ها تابع ساده شده را به صورت جمع حاصلضرب‌ها به‌دست می‌دهد:

$$F = B'D' + B'C' + A'C'D \quad \text{(الف)}$$

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

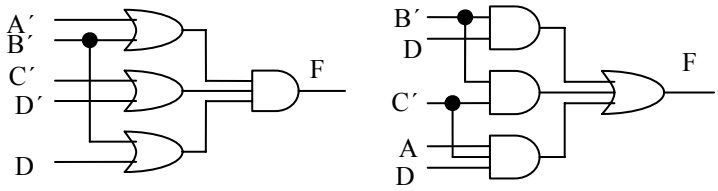
$$F' = AB + CD + BD'$$

با اعمال تئوری دمورگان (استفاده‌از دوگان و متمم کردن هر متغیر) تابع ساده شده را به صورت ضرب حاصل جمع‌ها به‌دست می‌آوریم:

$$F = (A' + B')(C' + D')(B' + D) \quad \text{(ب)}$$

پیاده‌سازی عبارت ساده شده حاصل از مثال ۹ در شکل٤-۱۸ دیده می‌شود. عبارت جمع حاصلضرب‌ها در بخش (الف) با گروهی از گیت‌های AND پیاده‌سازی شده‌است. 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 گشته‌است. در هر دو حال فرض بر این است که متمم متغیرها نیز مستقیماً در دست‌رسند و بنابراین نیازی به وارون‌گر نمی‌باشد.



F = (A′ + B′) (C′ + D′) (B′ + D) (ب) F = B′D′ + B′C′ + A′C′D (الف)

شکل٤-۱۸: پیاده‌سازی با گیت تابع مثال ۹

الگوهای ایجاد شده در شکل٤-۱۸ 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 پیکر بندی فوق دارای دو سطح از گیت‌ها می‌باشند. بنابراین پیاده‌سازی یک تابع استاندارد دو سطحی می‌گویند.

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 ٤-۱۹: جدول درستی تابع F

مثال ۱۰ روالی را برای محاسبه فرم ساده شده یک تابع بر حسب ضرب حاصل جمع‌ها، وقتی تابع ابتدا به صورت جمع مینترم‌ها است، نشان می‌دهد. این روال

هنگامی‌که تابع در آغاز بر حسب ماکسترم‌ها بیان شود نیز معتبر است. برای مثال به جدول درستی تابع F که در شکل ٤-۱۹ آمده‌است 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. نقشه‌این تابع در شکل ٤-۲۰ دیده می‌شود. برای ساده کردن این تابع، در مربع مربوط به هر جمله مینترم که تابع به‌ازاء آن، مقدار 1 گذاشته 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.

	yz	00	01	11	10
				y	
x	0	0	1	1	0
	1	1	0	0	1

شکل ٤-۲۰: جدول کارنو شکل ٤-۱۹

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 هم ارزی (XNOR) است . با متمم گیری مجدد از F′ تابع ساده شده را به ضرب حاصل جمع‌ها به‌دست خواهیم آورد.

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

برای وارد کردن یک تابع در یک نقشه که بر حسب ضرب حاصل جمع‌ها بیان شده‌است، می‌باید متمم تابع را به‌دست آورد و در آن مربع‌های مربوطه را با 0 پر کرد. مثلاً تابع

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

را می‌توان با متمم گیری 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.

٤-٤ 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 غیر کامل** نامیده می‌شود. در بسیاری از کاربردها، توجهی به مقدار منتسب به تابع در ازاء مینترم‌های نامعین نخواهیم داشت. به‌این دلیل مرسوم است که همه مینترم‌های نا مشخص در تابع را حالات بی‌اهمیت بخوانیم. از حالات بی‌اهمیت می‌توان برای ساده‌سازی بیشتر عبارت بول در یک نقشه‌استفاده کرد.

باید توجه داشت که یک مینترم بی‌اهمیت، ترکیبی از متغیرهاست که مقدار منطقی آن نامشخص است. به‌این 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 به‌این معنی است که تخصیص 1 یا 0 به تابع به‌ازاء 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 به‌دست 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. برای درک بهتر موضوع مثالی در زیر آمده‌است که در آن حالات بی‌اهمیت نشان داده می‌شوند:

مثال ۱۱: تابع بول زیر

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

که حالات بی‌اهمیت زیر را دارا می‌باشد، ساده کنید.

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

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 ٤-۲۱ نشان داده شده‌است. مینترم‌های F با 1 علامت زده شده‌اند، مینترم‌های d با x علامت گذاری شده‌اند و بقیه مربع‌ها با 0 پر شده‌اند. برای به‌دست 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

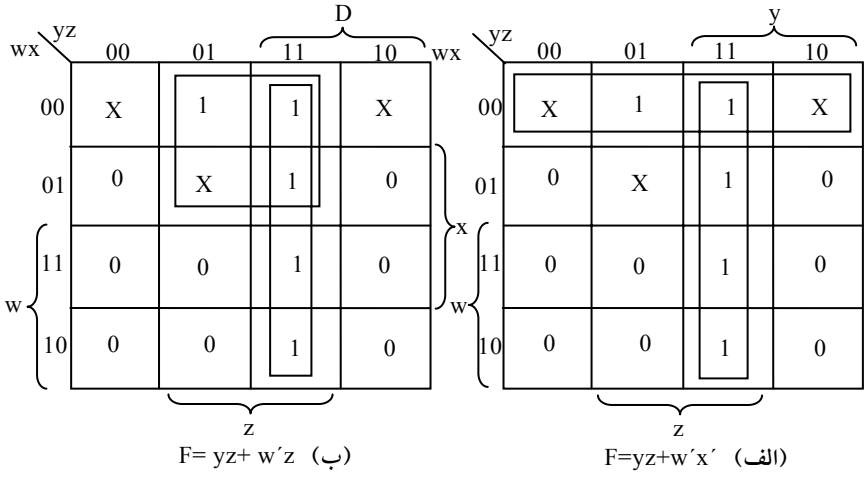
احتساب یک یا دو x همجوار، می‌توانیم چهار مربع مجاور را ترکیب نماییم تا جمله دو متغیره حاصل گردد. در بخش (الف) از نمودار، مینترم‌های بی‌اهمیت 0 و 2 با 1 جایگزین شده‌اند و تابع ساده شده به صورت زیر است.

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

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.



شکل ٤-٢١: جدول کارنو دارای حالات بی‌اهمیت (مثال ۱۰)

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 غیر کامل 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 ۱۰ 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 شده‌است. 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 غیر کامل است، هر دو عبارت قابل قبول‌اند زیرا تنها اختلاف در مقدار F مینترم‌های بی‌اهمیت می‌باشند.

می‌توان عبارت ضرب حاصل جمع‌ها را هم برای تابع شکل٤ - ۲۱ به‌دست آورد. در این حالت، تنها راه برای ترکیب 0ها جایگزینی مینترم‌های بی‌اهمیت شماره 0 و 2 با مقدار 0 می‌باشد و به‌این ترتیب تابع متمم ساده شده به‌دست می‌آید:

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

با متمم گیری 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.

سؤالات

۱- عبارت بولی زیر را ساده کرده و نمودار آن را رسم نمایید

$F_1 = x' y z + x' y z' + x y'$

۲- جدول کارنو تابع زیر را رسم کرده و سپس تابع را ساده نمایید.

$F(x , y , z) = \Sigma (2 , 3 , 4 , 6)$

۳- تابع بولی زیر را با استفاده از جدول کارنو ساده نمایید و متمم آنرا به‌دست آورید.

$F(x , y , z) = \Sigma (0, 1 , 4 , 6 , 7)$

٤- تابع بول چهار متغیره زیر را ساده کنید.

$F(w, x , y , z) = \Sigma (0 , 2 , 4 , 5 , 7 , 8 , 9, 14)$

۵- تابع بول زیر را با استفاده از جدول کارنو ساده کنید.

$F(A,B,C,D,E) = \Sigma(4 , 6 , 8 , 9, 13 , 21 , 22 , 25 , 27 , 30)$

٦- تابع بول $F(w , x , y , z) = \Sigma (0 , 3 , 7 , 8 , 14)$ که حالات بی‌اهمیت زیر را دارا می‌باشد، ساده کنید.

$d (w , x , y , z) = \Sigma (1 , 2 , 5)$

فصل ۵

پیاده‌سازی مدارهای دیجیتال با گیت‌هایNAND و NOR

هدف کلی

در این فصل مباحث اصلی مربوط به پیاده‌سازی گیت‌ها با استفاده از مدارهای NAND و NOR مورد بحث و بررسی قرار گرفته و علت استفاده از این نوع مدارها به همراه سادگی طراحی مطرح خواهند شد. همچنین مباحث تکمیلی سایر گیت‌های خاص که به نحوی در تکمیل مدارهای فوق‌الذکر تاثیر گذار هستند نیز مورد بحث خواهند بود.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

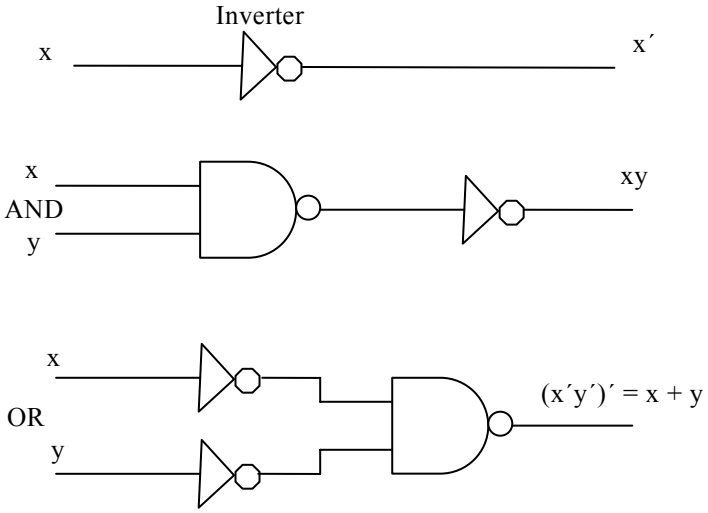
- مدارهای **NAND** و **NOR**
- علل استفاده از مدارهای **NAND** و **NOR**
- مدارهای **AND-OR-INVERT**
- مدارهای **OR-AND- INVERT**
- مدارهای **OR** انحصاری

مدارهای دیجیتال اغلب به جای AND , 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 ICهای دیجیتال به‌کار می‌روند. به دلیل مزیت

گیت‌های NAND و NOR در طراحی مدارهای دیجیتال، قواعد و روال‌هایی برای تبدیل توابع بول بیان شده بر حسب NOT , OR , AND به نمودارهای منطقی معادل بر حسب NAND و NOR بوجود آمده است.

۵-۱ مدارهای NAND

گیت NAND را یک گیت یونیورسال می‌گویند زیرا هر سیستم دیجیتالی را می‌توان با آن پیاده‌سازی کرد. برای اینکه نشان دهیم هر تابع بولی قابل پیاده‌سازی با گیت‌هایNAND می‌باشد، کافی است فقط نشان دهیم که اعمال منطقی , OR , NOT AND را می‌توان با NAND پیاده‌سازی کرد. این کار در شکل۵-۱ نشان داده شده است.

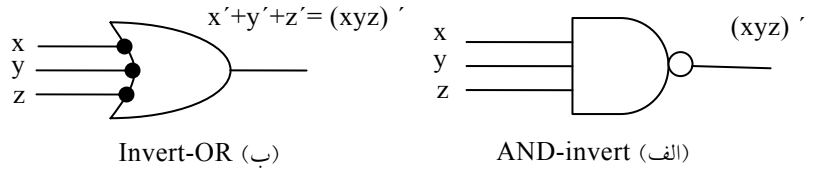


شکل۵-۱: عملیات منطقی با گیت‌های NAND

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.

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

برای ساده‌سازی تبدیل به منطق NAND، بهتر است سمبل گرافیکی دیگری برای گیت تعریف کنیم. دو سمبل گرافیکی معادل برای گیت NAND در شکل ۵-۲ 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 ونقش 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 قرار داد که دوایر کوچک به منزله متمم کردن هستند به‌دست می‌آید.



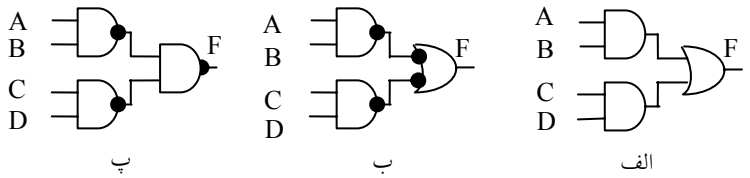
شکل۵-۲: دو سمبل گرافیکی برای گیت NAND

دو سمبل گرافیکی فوق در طراحی و تحلیل مدارهای NAND مفید هستند. وقتی هر دو سمبل در یک نمودار به کار روند گوییم مدار با علائم مخلوط نشان داده شده است.

۵-۱-۱ پیاده‌سازی دو سطحی گیت NAND

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 ۵-۳ توجه کنید. هر سه نمودار معادل بوده و تابع زیر را پیاده می‌نمایند.

$$F = AB + CD$$



شکل۵-۳: سه راه پیاده‌سازی تابع $F = AB + CD$

در (الف) تابع با گیت‌های AND , OR پیاده‌سازی شده است. در (ب) گیت‌های AND با گیت‌های NAND و گیت OR نیز با یک گیت NAND که با سمبل OR –invert مشخص شده پیاده‌سازی شده است. توجه داشته باشید که یک حباب به معنی متمم و دو حباب در یک مسیر دوبار متمم‌سازی را 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 ۵-۳ (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 ۵-۳ (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:

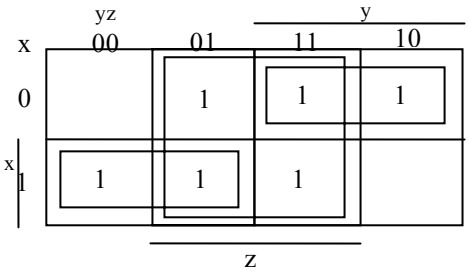
$$F = ((AB)'\,(CD)')' = AB + CD$$

مثال ۱: تابع بول زیر را با گیت‌های NAND پیاده کنید.

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

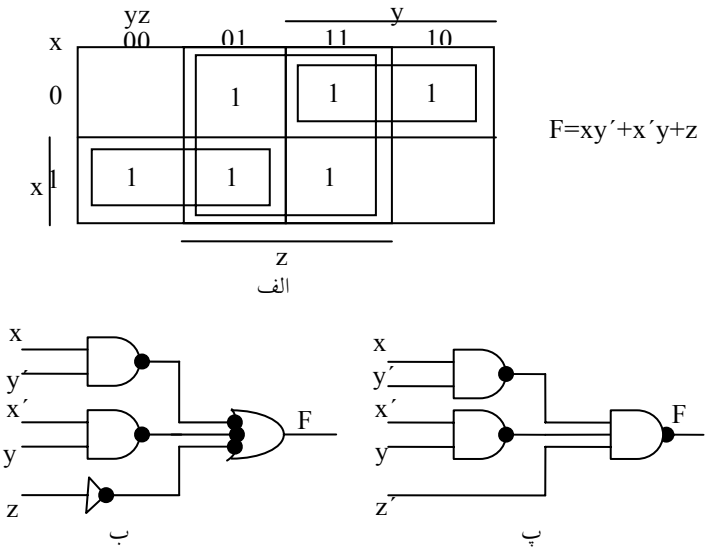
اولین قدم در تبدیل، ساده‌سازی تابع در جمع حاصلضرب‌هاست. این کار به کمک نقشه شکل۵-٤ انجام شده است و تابع حاصل به صورت زیر است.

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



شکل۵-٤: جدول کارنو تابع مثال ۱

پیاده‌سازی NAND دو سطحی در شکل۵-۵ (الف) 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.



شکل۵-۵: مدار منطقی مثال ۱

روش دیگری برای ترسیم نمودار منطقی در شکل۵-۵ (ب) نشان داده شده است. در اینجا تمام 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.

۵-۱-۲ روال تهیه مدار NAND از تابع بول

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:

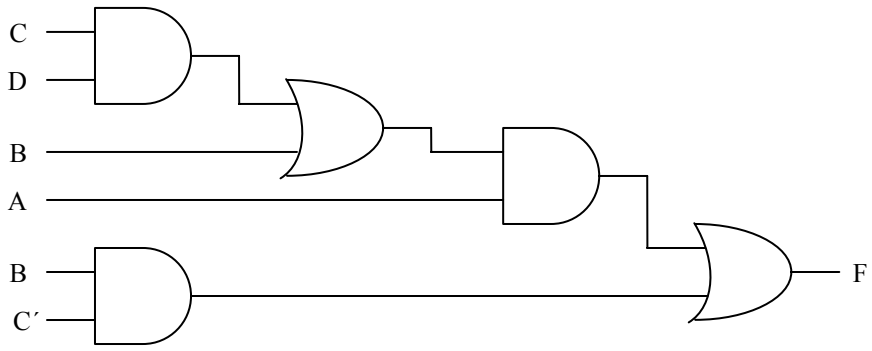
- تابع را ساده کرده آن را به فرم جمع حاصل ضرب‌ها بنویسید.
- برای هر جمله ضرب موجود در تابع که حداقل دو لیترال دارد یک گیت NAND بکشید. ورودی به هر یک گیت NAND لیترال‌های جمله‌اند. این مجموعه، گیت‌های سطح اول را تشکیل می‌دهد.
- در سطح دوم، یک گیت NAND با ورودی‌هایی که از خروجی‌های سطح اول می‌آیند بکشید. از سمبل گرافیکی AND–invert یا invert–OR استفاده نمایید.
- 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.

۵-۱-۳ 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 چهار طبقه را نتیجه می‌دهد. رایج‌ترین روش طراحی مدارهای چند طبقه بیان تابع بول بر حسب عملیات AND، OR , NOT می‌باشد. سپس می‌توان تابع را با گیت‌های AND , OR پیاده‌سازی کرد. آنگاه در صورت لزوم تمام مدار را می‌توان به NAND تبدیل نمود. به عنوان مثال تابع بول زیر را ملاحظه کنید:

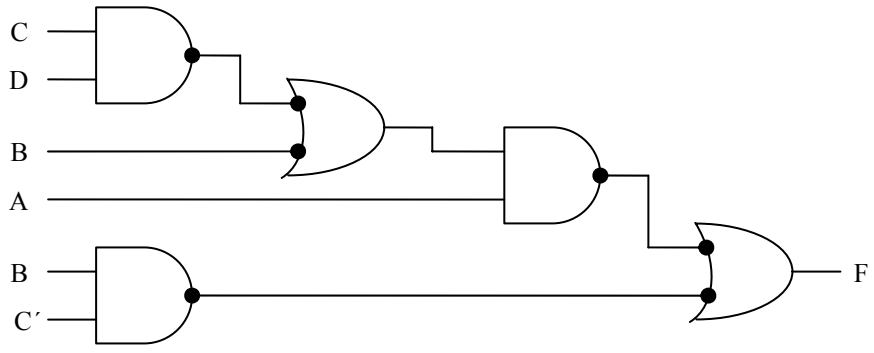
$$F = A\,(CD + B) + BC'$$

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 ۵–٦ 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.



شکل۵–٦: مدار تابع $F = A(CD + B) + BC'$ با استفاده از گیت AND - OR

با استفاده از علائم مخلوط، می‌توان یک نمودار منطقی با الگویی از سطوح متناوب AND , OR را به سادگی به مدار NAND تبدیل کرد. این تبدیل در شکل ۵–۷ دیده می‌شود. روال این است که هر گیت AND را به سمبل AND–invert و هر گیت OR را



شکل۵–۷: مدار تابع $F = A(CD + B) + BC'$ با استفاده از گیت NAND

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.

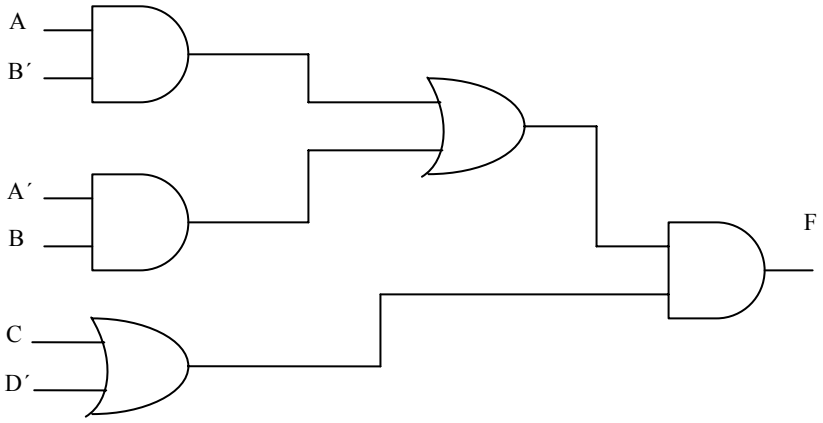
روال کلی نمودار چند طبقه AND –OR به نمودار تمام NAND با استفاده از علائم مخلوط به شرح زیر است:

- همه گیت‌های AND را با استفاده از سمبل‌های گرافیکی AND–invert به گیت NAND تبدیل کنید.
- همه گیت‌های OR را با سمبل‌های گرافیکیinvert– OR به NAND تبدیل نمایید.

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.

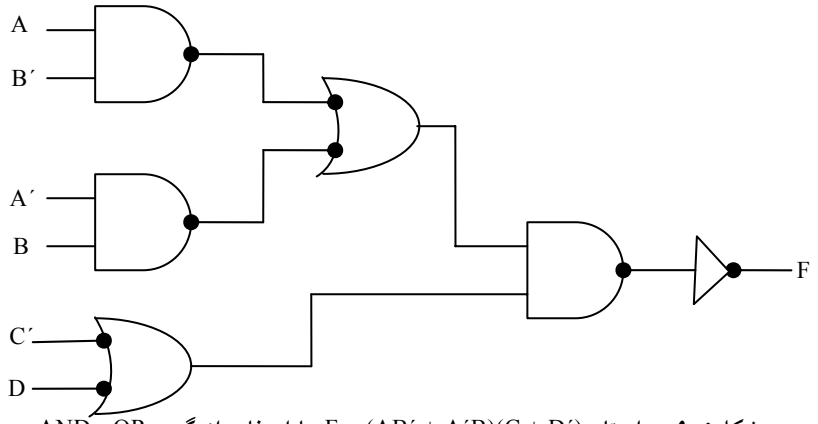
- $F = (AB' + A'B)(C + D')$



شکل۵–۸: مدار تابع $F = (AB' + A'B)(C + D')$ با استفاده از گیت AND - OR

پیاده‌سازی AND–OR تابع در شکل۵–۸ ملاحظه می‌شود که در آن سه طبقه گیت به کار رفته است.

در شکل۵–۹ 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 به‌دست آوردن مقدار اصلی تابع مجبوریم یک گیت وارون‌گر در خروجی به کار ببریم.

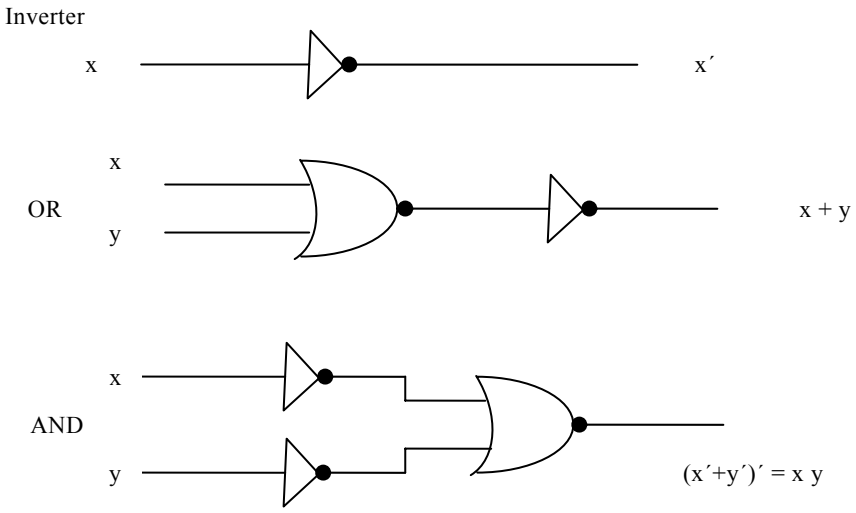


شکل۵–۹: مدار تابع $F = (AB' + A'B)(C + D')$ با استفاده از گیت AND - OR

۵–۲ مدارهای NOR و روش پیاده‌سازی آنها

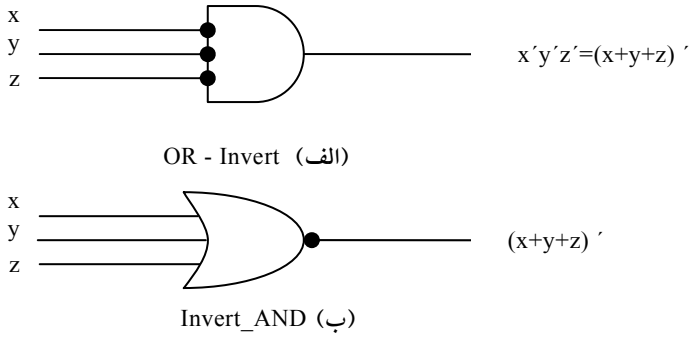
گیت NOR به‌عنوان گیت یونیورسال دیگری است که برای پیاده‌سازی هر تابع بول به کار می‌رود. عمل NOR دوگان NAND است. بنابراین تمام روال‌ها و قوانین منطق NOR دوگان روال‌های متناظر و قوانین حاصل در منطق NAND هستند. پیاده‌سازی اعمال AND، OR , NOT با گیت‌های NOR در شکل۵–۱۰ ملاحظه می‌گردد. عمل متمم، با گیت NOR یک ورودی حاصل شده و عیناً مثل وارون‌گر 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.



شکل۵-۱۰: عملیات منطقی با گیت‌های NOR

دو سمبل گرافیکی برای علائم مخلوط در شکل ۵ - ۱۱ 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.

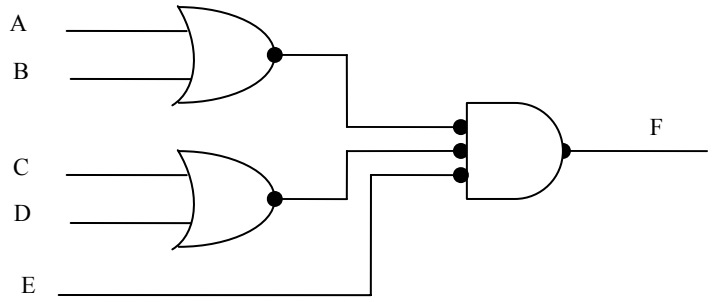


شکل ۵-۱۱: دو سمبل گرافیکی برای گیت NOR

پیاده‌سازی دو طبقه با گیت 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 به‌دست می‌آید. عبارت ضرب حاصل جمع‌ها 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 با 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 ۵–۱۲ 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.

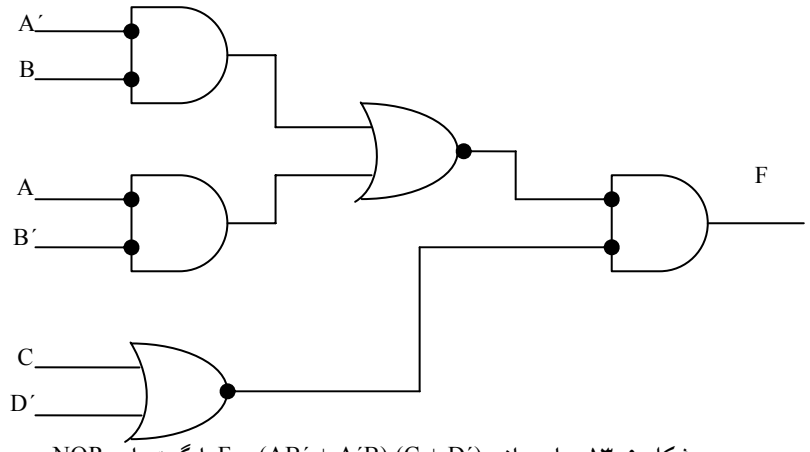


شکل ۵–۱۲: پیاده‌سازی $F = (A + B)(C + D)E$

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 گیتAND را به یک invert–AND تبدیل نماییم. هر حبابی که به وسیله حباب دیگر در همان مسیر جبران نشود نیاز به یک وارون‌گر یا متمم شدن لیترال ورودی دارد. برای مثال 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 ۵-۱۳: پیاده‌سازی $F = (AB' + A'B)(C + D')$ با گیت‌های NOR

۵-۳ منطق سیمی

دیگر پیاده‌سازی‌های دو سطحی گیت‌هایی که بیشتر در مدارهای مجتمع یافت می‌شوند از نوع NOR , NAND هستند. به همین دلیل، پیاده‌سازی‌های منطقیNAND , 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. منطق AND سیمی‌که با گیت‌های NAND انجام می‌شود در شکل ۵-۱٤ (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 سیمی‌است.

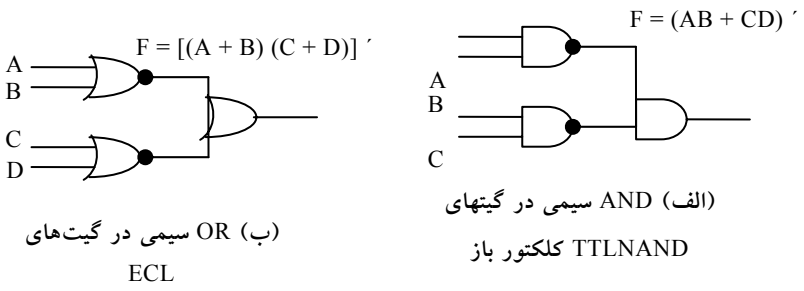
تابع منطقی پیاده شده با مدار شکل۵-۱٤ (الف) برابر زیر است.

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

و به آن تابع AND–OR – INVERT می‌گویند. به طور مشابه خروجی NOR گیت‌های ECL برای اجرای یک تابع OR سیمی‌به هم گره زده می‌شوند. تابع منطقی پیاده‌سازی شده با مدار شکل۵-۱٤ (ب) چنین است:

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

و به آن تابع OR – AND –INVERT می‌گویند.



(الف) AND سیمی در گیتهای TTLNAND کلکتور باز

(ب) OR سیمی در گیت‌های ECL

شکل ۵-۱٤: 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 ۵-۱٤ 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 سیمی‌در سمبل گرافیکی حذف می‌گردد.

٤-٥ 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 نوع دیگر را به سطح دوم نسبت دهیم، در می یابیم که 16 ترکیب ممکن از فرم دو سطحی وجود دارد. می‌توان در هر دو سطح یک نوع گیت مانند NAND–NAND را هم به‌کار 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 حاصل ضرب‌ها و یا ضرب حاصل جمع‌ها را تولید می‌کند این هشت فرم مفید عبارتند از:

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

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 قرار گرفته است. توجه کنید هر دو فرمی‌که در یک سطر آمده‌اند دوگان یکدیگرند. فرم‌های AND–OR و OR–AND، فرم‌های دو سطح اصلی بحث شده در بخش‌های قبل می‌باشند. فرم‌های

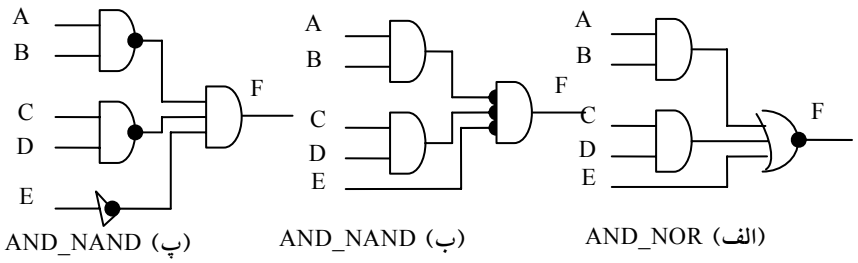
NAND–NAND و NOR–NOR در بخش‌های قبل ارائه شدند. چهار فرم باقیمانده نیز در این بخش بررسی می‌شوند.

۵-٤-۱ 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 ۵-۱۵، 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.

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

با استفاده از سمبل گرافیکی معادل دیگری برای گیت NOR، نمودار شکل ۵-۱۵ (ب) به‌دست 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 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 ۵-۱۵ (پ)، فرم NAND–AND است و برای پیاده‌سازی تابع AND–OR–INVERT نشان داده شده است.



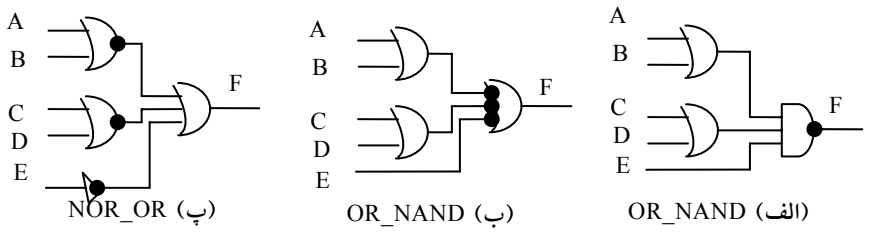
شکل ۵-۱۵: مدارهای AND–OR–INVERT ؛

پیاده‌سازی AND–OR نیاز به یک عبارت جمع حاصلضرب‌ها دارد. پیاده‌سازی AND–OR–INVERT مشابه آن است، به جز اینکه 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 بدنبال آمده است.

۵-٤-۲ پیاده‌سازی OR–AND–INVERT

فرم‌های OR–NAND و NOR–OR تابع OR–AND–INVERT را اجرا می‌کنند. این فرم‌ها در شکل ۵-۱٦ نشان داده شده‌اند. فرمOR–NAND، فرم OR–AND را تداعی می‌کند. به جز اینکه در خروجی گیت NAND، عمل متمم با حباب انجام می‌شود. در این شکل تابع زیر پیاده‌سازی شده است.

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



شکل۵-۱٦: مدارهای OR–AND–INVERT ؛ $F=[(A+B)(C+D)E]'$

با استفاده از فرم دیگر گیت NAND نمودار شکل ۵-۱٦ (ب) به‌دست می‌آید. مدار در (پ) با انتقال دوایر کوچک از ورودی‌های گیت سطح دوم به خروجی‌های گیت‌های سطح اول حاصل می‌گردد. مدار شکل ۵-۱٦ (پ) یک فرم NOR–OR است و قبلاً برای پیاده‌سازی تابع OR–AND–INVERT در شکل ۵-۱٤ نشان داده شد. پیاده‌سازی OR–AND–INVERT به عبارتی به فرم ضرب حاصل جمع‌ها احتیاج دارد.

اگر متمم تابع به صورت ضرب حاصل‌جمع‌ها 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.

مثال ۲: تابع شکل۵–۱۷ (الف) را به فرم دو سطحی پیاده‌سازی کنیم.

متمم تابع با ترکیب 0ها در نقشه به فرم جمع حاصل ضرب‌های ساده شده به‌دست می‌آید.

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

خروجی نرمال این تابع می‌تواند به صورت زیر بیان شود.

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

که به فرم AND–OR–INVERT است. پیاده‌سازی‌های AND–NOR و NAND–AND در شکل۵–۱۷ (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 z′e استفاده کنیم به وارون‌گر احتیاجی نیست. فرم‌های OR–AND–INVERT نیاز به عملیات ساده شده‌ای از متمم تابع به فرم حاصل جمع‌ها دارد. برای تهیه این عبارت، ابتدا 1ها را در نقشه ترکیب می‌کنیم.

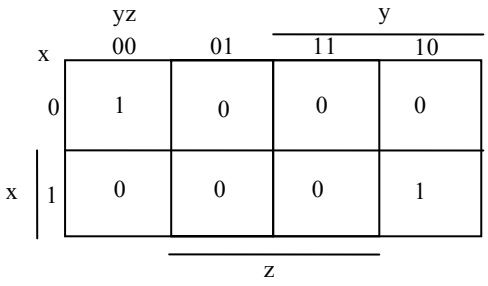
آنگاه متمم تابع را به‌دست می‌آوریم.

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

خروجی نرمال F اکنون به فرم زیر نوشته می‌شود:

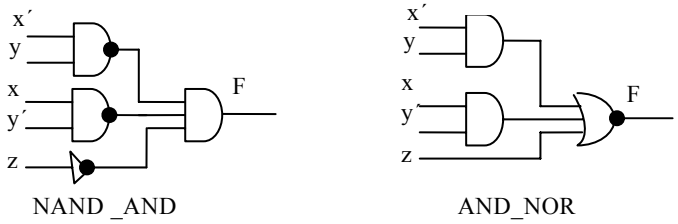
$$F=[(x+y+z)(x'+y'+z)]'$$

که به فرمOR–AND– INVERT بیان شده است. با استفاده از این عبارت تابع را می‌توانیم به فرم OR–NAND یا NOR–OR که در شکل۵–۱۷ (پ) نشان داده شده است، نیز پیاده کنیم.

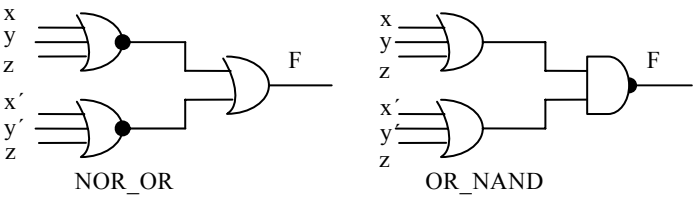


$F = x'y'z' + xyz'$
$F = x'y + xy' + z$

$F = (x'y + xy' + z)'$ **(الف)**



$F = (x'y + xy' + z)'$ **(ب)**



$F=[(x + y + z)\,(x' + y' + z)]'$ **(پ)**

شکل ۵-۱۷: پیاده‌سازی دو طبقه تابع F

۵-۵ تابع OR انحصاری

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

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

این تابع هنگامی‌برابر 1 است که فقط x یا y برابر 1 باشند، ولی هردوآنها به‌طور همزمان 1 نباشند. NOR انحصاری (XNOR)که به آن هم ارزی هم می‌گویند عمل زیر را اجرا می‌نماید.

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

هنگامی‌که 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 به‌کارگیری جدول درستی و جایگزینی ⊕ با عبارت بولی هم ارزی ثابت شود. و نیز می‌توان نشان داد که عمل OR انحصاری خاصیت جابجایی و شرکت‌پذیری را دارد. یعنی:

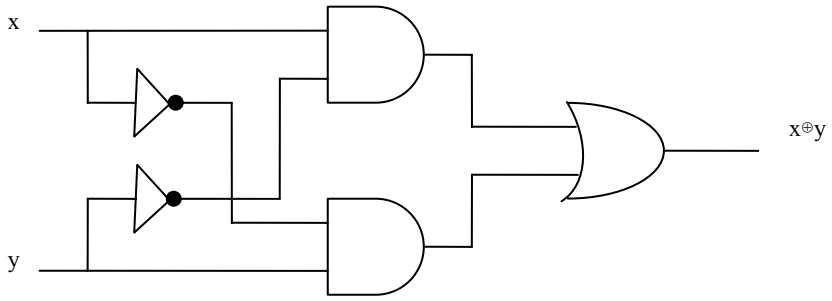
$$A \oplus B = B \oplus A$$

و

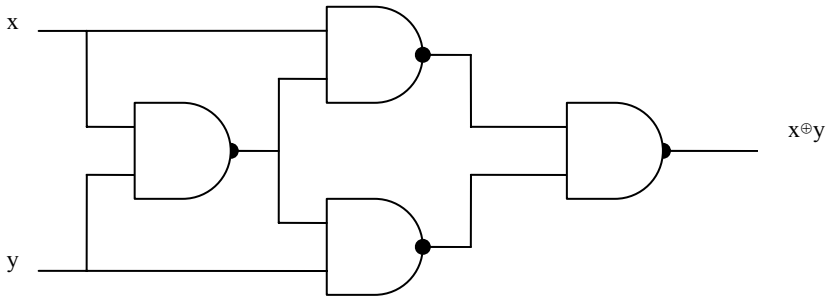
$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

این بدان معنی است که دو ورودی گیت OR انحصاری بدون تاثیر 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 عنوان گیت‌های دیگر ساخته می‌شود. یک تابع OR انحصاری دو ورودی که با گیت‌های معمولی AND، OR و NOT ساخته شده در شکل ۵–۱۸ (الف) دیده می‌شود.



(الف) با گیتهای AND-OR-NOT



(ب) با گیتهای NAND

شکل ۵–۱۸: پیاده‌سازی گیت XOR

شکل۵–۱۸ (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 بر حسب OR انحصاری بیان می‌شوند. با این وجود این تابع به کرات ضمن طراحی سیستم‌های دیجیتال به کار گرفته می‌شود. خصوصاً در عملیات حسابی و خطایابی و تصحیح خطا بسیار مفید است.

۵-۵-۱ تابع فرد

عملگر OR انحصاری با سه متغیر یا بیشتر را می‌توان با جایگزینی سمبل ⊕ 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.

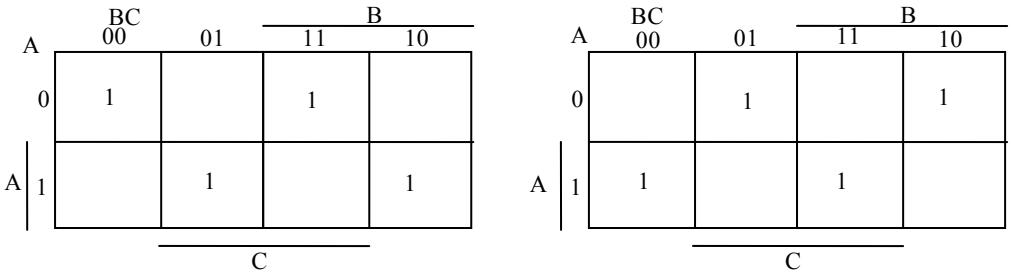
$$A \oplus B \oplus C = (AB'+A'B)C'+(AB+A'B')C$$
$$= AB'C'+A'BC'+ABC+A'B'C$$
$$= \Sigma(1,2,4,7)$$

عبارت بول به وضوح نشان می‌دهد که تابعOR انحصاری سه متغیره برابر با 1 است به شرطی که فقط 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 مینتروم دیگری که در تابع لحاظ نشده‌اند 000، 011، 101 و110 بوده و تعداد زوجی 1 در مقدار دودویی آنها وجود دارد. به‌طور 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.

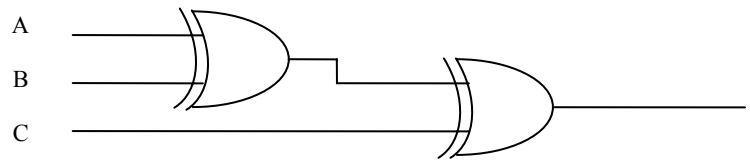
تعریف یک تابع فرد با ترسیم آن در یک نقشه شفاف تر می‌کنیم. شکل۵-۱۹ (الف) نقشه را برای تابع XOR سه متغیره نشان می‌دهد. چهار مینترم تابع یک مربع در میان با

هم فاصله دارند. تابع فرد از چهار مینترمی‌که مقادیر دودویی‌اش تعداد فردی 1 دارند شناسایی می‌شود.

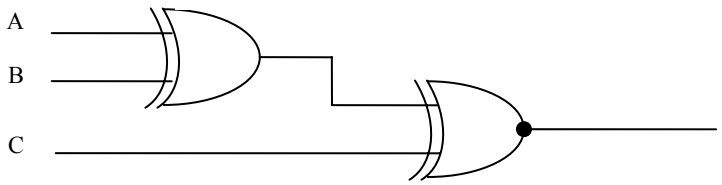


شکل ۵-۱۹ نقشه تابع XOR سه متغیره

متمم یک تابع فرد، یک تابع زوج است. طبق شکل۵-۱۹ (ب)، تابع زوج سه متغیره هنگامی1 است که تعداد زوجی متغیر در یک مینترم، 1 باشد (از جمله مینترمی‌که هیچ یک از متغیرها در آن 1 نیست).



(الف) تابع فرد سه ورودی



(ب) تابع زوج سه ورودی

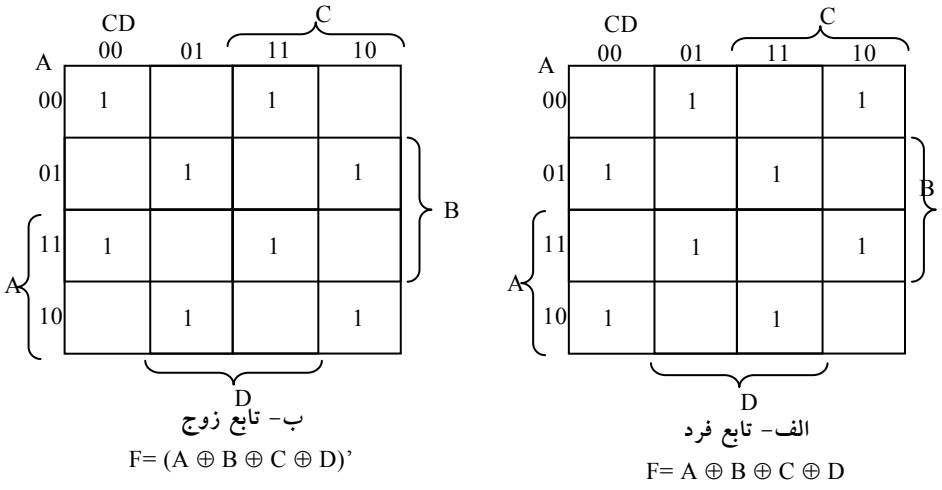
شکل ۵-۲۰: نمودار منطقی توابع فرد و زوج

تابع فرد ۳ ورودی را می‌توان با گیت دو ورودی هم طبق شکل ۵-۲۰ (الف) پیاده‌سازی کرد. متمم یک تابع فرد با جایگزینی گیت خروجی با یک گیت XNOR طبق شکل ۵-۲۰ (ب) به‌دست می‌آید.

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

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

برای تابع بول چهار متغیره 16 مینترم وجود دارد. نیمی‌از 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 به‌دست می‌آید.



شکل ۵-۲۱: نقشه برای تابع XOR چهار متغیره

نقشه شکل ۵-۲۱ (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 ۵-۲۱ (ب) تابع زوج چهار متغیره هنگامی1 است که تعداد زوجی از متغیرها در مینترم برابر 1 باشد.

۵-۵-۲ تولید و چک توازن

توابع XOR در سیستم‌هایی که به کدهای عیب یاب و تصحیح کننده خطا نیاز دارند بسیار مفیدند. همانطور که در فصل اول 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.

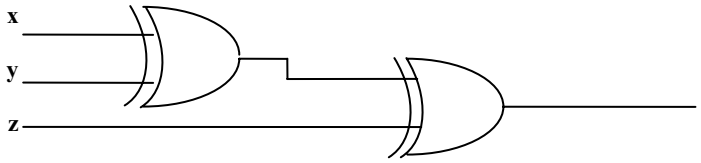
فرض کنید بخواهیم یک پیام سه بیتی را همراه با یک بیت توازن زوج ارسال کنیم. جدول شکل۵-۲۲، جدول درستی را برای مولد توازن نشان می‌دهد. سه بیت x وy وz که پیام را تشکیل می‌دهند ورودی به مدار هستند. بیت توازن P، خروجی است. برای توازن زوج، بیت P باید طوری باشد که تعداد کل 1ها را زوج کند (از جمله P). از جدول درستی می‌بینیم که P یک تابع فرد را تشکیل می‌دهد زیرا برای مینترم‌هایی که

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

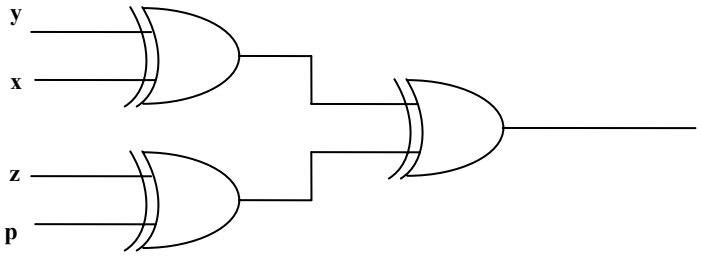
شکل ۵-۲۲: جدول درستی مولد توازن زوج

تعداد فردی 1 دارند باید برابر 1 شود. بنابراین P به صورت یک تابع XOR سه متغیره بیان می‌شود.

نمودار منطقی مولد توازن در شکل ۵-۲۳ (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.



(الف) مولد توازن زوج ۳ بیتی



(ب) چک کننده توازن زوج ٤ بیتی

شکل ۵-۲۳: نمودار منطقی مولد و چک کننده توازن

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

خطا در حین انتقال هنگامی‌رخ می‌دهد که چهار بیت دریافتی تعداد فردی 1 دارد، و این به معنی رخ داد خطا در حین انتقال است. خروجی چک کننده توازن که با C مشخص شده است به هنگام رخ داد خطا برابر 1 می‌شود. یعنی اگر چهار بیت دریافتی تعداد فرد 1 داشته باشد خطا رخ داده است. جدول شکل۵-۲٤ جدول درستی برای چک کننده توازن زوج است.

با توجه به آن ملاحظه می‌شود که تابع C متشکل از هشت مینترم با مقدار دودویی دارای تعداد فردی 1 است. این مطلب مربوط به شکل۵-۲۳ (الف) است که تابع فرد را نشان می‌دهد. می‌توان چک کننده توازن را با گیت‌های XOR پیاده‌سازی کرد:

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

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

شکل ۵-۲٤: جدول درستی چک کننده توازن زوج

نمودار منطقی چک کننده توازن در شکل۵-۲۳ (ب) ملاحظه می‌گردد. لازم به تذکر است که مولد توازن را با مدار شکل۵-۲۳(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 نیمی‌از 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.

۵-۶ 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 برنامه نویسی است، ولی خصوصا حول توصیف ساختارهای سخت‌افزاری و رفتار آنها بنا نهاده شده است. می‌توان از آن برای نمایش نمودارهای منطقی، عبارت 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 موثر و نیز به سادگی قابل ذخیره ، بازیابی و پردازش به وسیله نرم‌افزار کامپیوتر است.

در پردازش HDL دو کاربرد وجود دارد:

- شبیه سازی
- سنتز

شبیه‌سازی منطقی نمایشی از ساختار و رفتار یک سیستم منطقی دیجیتال به کمک کامپیوتر است. یک شبیه ساز توصیف HDL را تفسیر کرده و یک خروجی قابل خواندن مانند نمودار زمانی، تولید می‌نماید و بدین وسیله رفتار سخت‌افزار 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 فرایندی 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 زبان سطح بالاست. تفاوت در این است که، در عوض تولید کد منتج، یک بانک اطلاعاتی تولید می‌نماید که در آن دستور العمل‌های ساخت یک قطعه سخت‌افزاری دیجیتال فیزیکی توصیف شده با کد HDL آمده است. سنتز منطقی بر روال‌های مبتنی است که مدارهای دیجیتال را پیاده‌سازی می‌کنند و شامل آن بخش از یک طراحی دیجیتال که قابل اتوماتیک شدن با نرم‌افزار کامپیوتر باشد.

در صنعت HDLهای 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 سخت تری است. چونVerilog برای یادگیری ساده‌تر است، ما آن را در این کتاب انتخاب کرده ایم. با این وجود، توصیف‌های Verilog HDL لیست شده در سرتاسر این کتاب تنها درباره Verilog نیست، بلکه معرفی نمایش سیستم‌های دیجیتال به کمک کامپیوتر به وسیله نوعی زبان توصیف سخت‌افزاری است.

۵-۶-۱ نمایش مدول

زبان Verilog HDL دارای دستور زبانی است که دقیقاً ساختارهای مجازی که در زبان می‌توانند به کار روند را توصیف می‌نماید. خصوصا، Verilog حدود 100 کلمه کلیدی از پیش تعریف شده ، حروف کوچک و شناسه‌هایی دارد که ساختار زبان را تعریف می‌کنند. تعدادی از کلمات کلیدی این زبان که از اهمیت بالایی برخوردارند، به شرح ذیل می‌باشد:

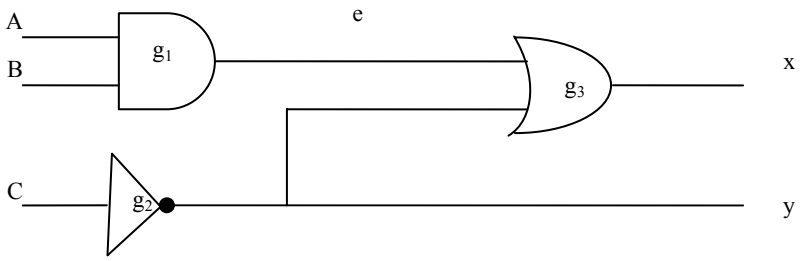
module	input	wire	Or
endmodule	output	and	not

هر متن بین دو اسلش (//) و انتهای خط به عنوان یک توضیح تفسیر می‌گردد. فاصله های Blank و نام‌های 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 end module پایان می‌یابد. اکنون برای تشریح بعضی از مفاهیم زبان مثال ساده‌ای را تشریح می‌کنیم.

توصیف HDL مدار شکل۵-۲۵ در مثال ۳ 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. شناسه ها نام‌هایی هستند که به متغیرها داده می‌شوند و به این ترتیب در طراحی قابل ارجاع می‌گردند. آنها از کاراکترهای الفبا عددی و زیر خط (–) ساخته می‌شوند و حساس به اندازه اند. 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.

مثال ۳:

```
//Description of simple circuit
Module smpl-circuit (A, B, C ,X, Y);
Input A , B , C;
Output X , Y;
Wire e;
And g1 (e ,A, B);
Not g2 (y,c);
Or g3(x ,e, y);
End module
```



شکل ۵-۲۵: مداری برای نمایش عبارت HDL مثال ۳

همه کلمات کلیدی که باید به حروف کوچک باشند به منظور وضوح با خط پر رنگ چاپ می‌شوند، ولی این زبان نیاز نیست. input، 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 و g2 وغیره و به دنبال آن خروجی و ورودی‌هایی که با ویرگول از هم جدا شده و در داخل پرانتزاند، صورت می‌گیرد. همواره خروجی در ابتدا معرفی می‌شود و سپس 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.

۵-۶-۲ تاخیر در گیت‌ها

یکی از مباحث مهم در طراحی مدارات دیجیتال، ایجاد تاخیر در بین گیت‌ها می‌باشد. هنگام استفاده از HDL در شبیه سازی، گاهی لازم است مقداری تاخیر بین ورودی تا خروجی گیت در نظر گرفته شود.در Verilog تاخیر بر حسب 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 " ' " (backquote) شروع می‌شوند. چنین رهنمونی قبل از اعلان مدول مشخص می‌گردد. مثالی از رهنمون timescale در زیر آمده است.

timescale 1ns/ 100 Ps'

عدد اول نشان دهنده واحد اندازه‌گیری برای زمان‌های تاخیر است. عدد دوم دقتی که تحت آن تاخیرها گرد شده‌اند را نشان می‌دهد که در این حالت 0.1ns است. اگر timescale مشخص نشود، شبیه ساز واحد زمان معینی را، مثل 1ns، پیش فرض می‌کند. در این کتاب، واحد زمان پیش فرض را انتخاب خواهیم کرد.

مثال ۴: HDL توصیف مثال قبلی را همراه با تاخیر در هر گیت مشخص می‌نماید. گیت‌های AND، OR، NOT به ترتیب زمان تاخیر 30ns، 20ns و10ns را دارند. اگر مدار شبیه‌سازی شود و ورودی‌ها از 000 به 111 تغیر یابند، خروجی‌ها طبق جدول شکل۵-۲۶ تغییر می‌کنند.

خروجی وارون‌گر در y پس از تاخیر 10ns از 1 به 0 تغییر می‌یابد. خروجی گیت AND در e پس از 30ns تاخیر 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، تاخیرهای گیتی قبل از آن برای مدت 20ns یک جرقه منفی ایجاد می‌نمایند.

```
//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);
End module
```

	واحد زمانی	ورودی	خروجی
	(ns)	**ABC**	**yex**
تغییر اولیه	–	000	101
	–	111	101
	10	111	001
	20	111	001
	30	111	010
	40	**111**	**010**
	50	**111**	**011**

شکل ۵-۲٦: جدول خروجی گیت‌ها پس از تاخیر

برای شبیه‌سازی یک مدار با HDL، لازم است ورودی‌ها را برای شبیه ساز به مدار اعمال کنیم تا پاسخ خروجی تولید گردد. یک توصیف HDL که محرک را برای یک طرح فراهم می‌کند برنامه تست خوانده می‌شود. در اینجا بدون آن که توضیحاتی اضافی را ارائه کنیم روال را برای مثال ساده‌ای شرح می‌دهیم.

مثال ۵: این مثال HDL یک برنامه تست را برای شبیه‌سازی مدار تاخیر دار نشان می‌دهد. دو ماژول در این برنامه تست لحاظ شده است:

- ماژول محرک
- ماژول 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. مثال ۵ عبارت HDL

```
//stimulus for simple circuit
Module stimulus ;
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
```

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 b0 ' 1 تنظیم شده‌اند، و به معنی یک رقم دودویی با مقدار صفر است.) پس از 100ns ثانیه دیگر شبیه‌سازی خاتمه می‌یابد. ($ finish) یک تکلیف در سیستم 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 غیر مشخص 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 ۵-۳ پیش‌بینی شده بود.

۵-۶-۳ عبارت بولی

عبارت 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 سمبل‌های (&)، (|) و (~) به ترتیب برای AND، OR و NOT استفاده می‌کند. بنابراین برای توصیف مدار ساده شکل۵-۲۵ با یک عبارت بولی عبارت زیر را به کار می‌بریم.

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

مثال ۶: عبارت HDL

این مثال 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.

```
// 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);
End module
```

دیدیم که یک مدار دیجیتال می‌تواند با عبارت HDL درست مثل ترسیم در یک نمودار مداری، یا با عبارت بولی توصیف گردد. مزیت HDL این است که برای پردازش به وسیله کامپیوتر مناسب است.

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 Primitiveهای دیگری را با تعریف آنها به صورت جدول اضافه نماید. این نوع مدارها را تعریف شده به‌وسیله کاربر یا UDP 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 ۷: عبارت HDL یک UDP را با یک جدول درستی تعریف می‌کند. حل آن بر اساس دستورالعمل زیر است:

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 به‌کارگیری 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 و در پایان endprimitive ذکر می‌شود.

```
//User defined primitive (UDP)
Primitive crctp ( x, A , B, C);
Output X;
Input A , B , C;
// Truth table for x(A, B , C )= ∑(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;
Crcpt ( w, x,y, z );
endmodule
```

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 HDLها و سیستم‌های طراحی کامپیوتری (CAD) روال‌های دیگری را برای مشخص کردن مدارهای دیجیتال به صورت جدول استفاده می‌کنند. جداول می‌توانند با نرم‌افزار CAD برای به‌دست آوردن یک ساختار گیتی بهینه پردازش شوند. در این بخش، HDL را معرفی و مثال‌های ساده‌ای از مدل‌سازی ساخت یافته را ارائه دادیم.

سؤالات

۱- تابع بولی زیر را با استفاده از گیت‌های NAND طراحی نمایید.

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

۲- تابع بولی زیر را با استفاده از گیت‌های معمولی و گیت‌های NAND پیاده‌سازی نمایید.

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

۳- تابع بولی زیر را با استفاده از گیت های NOR طراحی نمایید.

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

٤- تابع زیر را با استفاده از مدارهای AND–OR–INVERT طراحی نمایید.

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

فصل ٦

مدارهای ترکیبی

هدف کلی

در این فصل مباحث اصلی مربوط به مدارهای ترکیبی به همراه روشهای تحلیل و طراحی مورد بحث و بررسی قرار گرفته و انواع مدارهای جمع‌کننده و تفریق کننده در حالات دودویی و دهدهی ارائه خواهند شد. همچنین عبارات ضرب دودویی نیز به همراه مدارهای مقایسه‌ای مورد بررسی قرار خواهند گرفت.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مفهوم مدارهای ترکیبی
- روش‌های تحلیل و طراحی مدارهای ترکیبی
- انواع جمع‌کننده‌ها و تفریق‌گرها
- مدارهای ضرب دودویی
- مدارهای مقایسه‌گرها

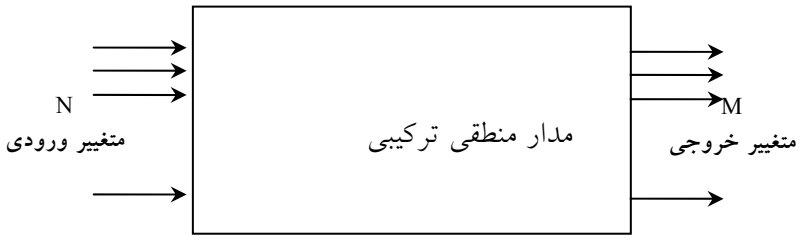
٦-١ مدارهای ترکیبی

یکی از نکات مهم در طراحی مدارهای منطقی مورد استفاده در سیستم‌های دیجیتال، بحث ترکیبی بودن و یا ترتیبی بودن مدار می‌باشد و این مهم به‌عنوان یکی از

پارامترهای مهم در دسته‌بندی مدارهای دیجیتال مطرح می‌گردد. در این فصل مدارهای ترکیبی مورد بحث و بررسی قرار خواهند گرفت و مدارهای ترتیبی در فصل هشتم توضیح داده خواهند شد.

یک *مدار ترکیبی* متشکل از تعدادی گیت منطقی است که خروجی آنها در هر لحظه از زمان مستقیماً به وسیله ورودی‌های همان لحظه معین می‌شوند 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.

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 مورد نیاز در خروجی تبدیل می‌کند. نمودار کلی یک مدار ترکیبی در شکل۶-۱ دیده می‌شود.



شکل ۶-۱: نمودار بلوکی یک مدار ترکیبی

n متغیر دودویی ورودی از منبع 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 بر حسب n متغیر ورودی بیان می‌گردد. در این فصل با استفاده از دانش فصل های 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 کرات 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 جمع کننده‌ها، تفریق‌گرها، مقایسه‌گرها، دیکدرها، انکدرها و مولتی‌پلکسرها را معرفی می‌کنیم. این قطعات به صورت مدارهای مجتمع MIS (مجتمع با فشردگی متوسط) در دسترسند. به آنها در مدارهای پیچیده VLSI، مانند مدارات مجتمع خاص (ASIC)، سلول های استاندارد هم می‌گویند. توابع سلول های استاندارد در داخل مدارهای VLSI به همان شکل به هم متصل می‌شوند که در طراحی MIS متشکل از چند IC، وصل شدند.

٦-۲ روش تحلیل

در تحلیل یک مدار ترکیبی، 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.

به محض این که محقق شد مدار از نوع ترکیبی است، می‌توان برای به‌دست آوردن توابع بول خروجی یا جدول 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.

٦-۲-۱ تهیه توابع بول خروجی از یک مدار منطقی

برای به‌دست آوردن توابع بول خروجی از یک مدار منطقی به ترتیب زیر باید عمل کرد:

تمام خروجی‌های گیت‌هایی که تابعی از ورودی هستند باید با سمبل‌های دلخواه نام‌گذاری شوند. برای خروجی هر گیت تابع بول را معین کنید. 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 به‌دست آورید.

فرآیند مرحله ۲ را تا دستیابی به خروجی‌های مدار ادامه دهید. با جایگزینی توابع به‌دست آمده در قبل، توابع بولی خروجی را بر حسب متغیرهای ورودی اولیه به‌دست آورید. تحلیل مدارهای ترکیبی شکل۶-۲ روال پیشنهادی را تشریح می‌نماید. توجه دارید که مدار دارای سه ورودی دودویی A، B و C و دو خروجی F_1 وF_2 است. خروجی گیت‌هایی که تابعی از متغیرهای ورودی‌اند عبارتند از T_1 وT_2. خروجیF2 به سادگی از متغیرهای ورودی به دست می‌آید. توابع بول 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 به‌دست آوردن F_1 بر حسب A، B و C، یکسری جایگزینی‌ها را به فرم زیر انجام می‌دهیم:

$$F_1 = T_3 + T_2 = F'_2\, T_1 + ABC$$

$$\cdot = (AB + AC + BC)'(A + B + C) + ABC$$

$$\cdot = (A'+B')\,(A'+C')\,(B'+C')\,(A + B + C) + ABC$$

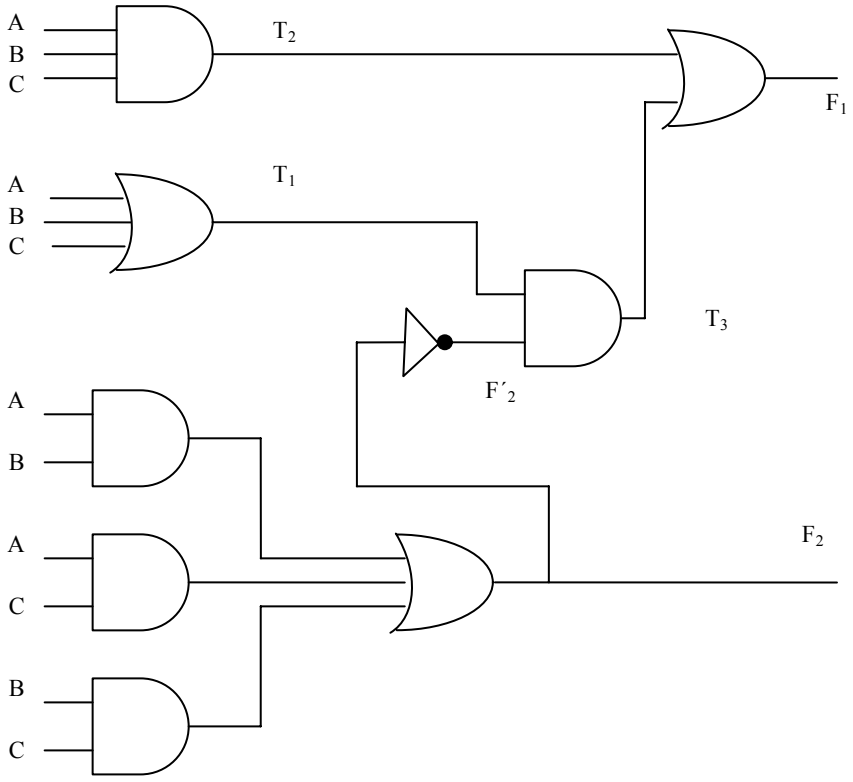
$$\cdot = (A'+B'C')(AB'+AC'+BC'+B'C)+ABC$$

$$\cdot = A'BC' + A'B'C + AB'C' + ABC$$

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.

٦-۲-۲ تهیه جدول درستی از نمودار منطقی

به‌دست 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:



شکل ٦-۲: نمودار منطقی برای مثال تحلیل

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 به‌دست آورید.
٤. برای به‌دست 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 شکل٦-۲ تشریح می‌شود. در جدول شکل٦-۳، هشت ترکیب ممکن را برای سه متغیر ورودی تشکیل می‌دهیم. جدول درستی برایF_2 مستقیماً از مقادیر A، B وC 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 وT_2 به ترتیب توابع OR وAND متغیرهای ورودی می‌باشند. مقدار T_3 از T_1 وF'_2 حاصل می‌شود: وقتیکه هر دوT_1 و$F'2$ برابر با 1 باشندT_3 نیز برابر 1 است، در غیر این صورت T_3 برابر0 خواهد بود. بالاخره F1 برای آن دسته از ترکیبات 1 است که در آنهاT_2 یا T_3 یا هر دو برابر 1 باشند. بررسی ترکیبات جدول درستی برای A، B و C وF_1 و F_2 نشان می‌دهد که این جدول با جدول جمع‌کننده کامل بخش ٦-٤ با متغیرهای x وy و z و S و C برابر است.

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

شکل ٦-۳: جدول درستی برای نمودار منطقی شکل ٦-۲

٦-۳ 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:

- با استفاده از مشخصات مدار تعداد ورودی‌ها و خروجی‌ها را معین کرده و به هر کدام سمبلی تخصیص دهید.
- جدول درستی مربوط به ورودی‌ها و خروجی‌های مدار را تشکیل دهید.
- توابع بولی ساده شده را برای خروجی به صورت تابعی از متغیرهای ورودی به دست آورید.
- 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 به دست 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 به‌دست می‌دهند. تفسیر لفظی صحیح جدول درستی از اهمیت خاصی بر خوردار 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 روش های موجود مانند دستکاری جبری، جدول کارنو یا برنامه‌های ساده‌سازی مبتنی بر کامپیوتر ساده می‌شوند. غالباً عبارات ساده شده متعددی حاصل می‌شود که باید مناسب ترین راه 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 سیگنال‌ها در گیت‌ها، تعداد اتصالات داخلی، محدودیت های مربوط به قابلیت راه اندازی هر

گیت، و دیگر معیارهایی که باید در طراحی با مدارهای مجتمع مد نظر باشد، را در نظر می‌گیرد. در بسیاری از حالات ساده‌سازی با تصدیق و تایید یک هدف ساده، مثل تولید توابع بولی به فرم استاندارد آغاز شده و سپس با برآورده کردن دیگر معیار های رفتاری پیش می‌رود.

٦-٣-١ مکانیزم های تبدیل اعداد در مبناهای متفاوت

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 سیستمی‌دیگر 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 به‌کارگیری کد دودویی متفاوت، با هم سازگار می‌سازد.

برای تبدیل کد دودویی A به کد دودویی 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. 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 روابط بین ورودی‌ها و خروجی‌ها در جدول شکل ٦-٤ دیده می‌شود. 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

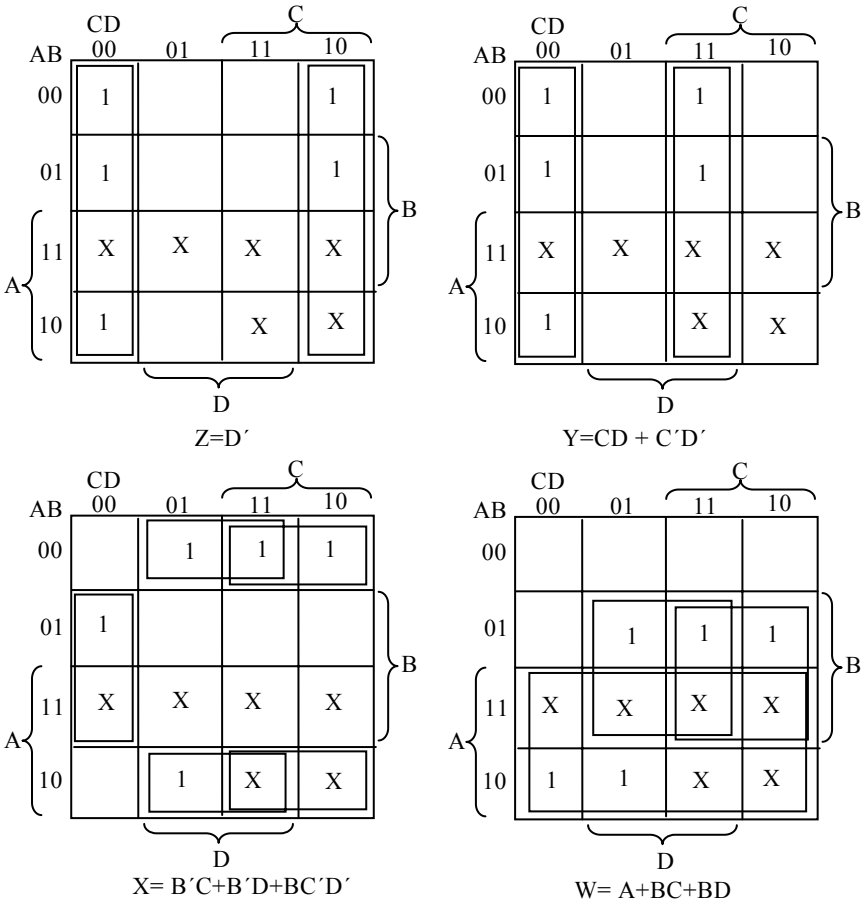
هرگز رخ نمی‌دهند. بنابراین به متغیرهای خروجی مربوط به آنها به دلخواه 0 یا 1 خواهیم داد و این تخصیص به نحوی خواهد بود که از آن مدار ساده‌تری حاصل گردد.

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 ٦-٤: جدول درستی برای مثال تبدیل کد

نقشه‌ها در شکل٦ - ٥ برای به‌دست 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. ۱هایی که در مربع ها نوشته شده‌اند از مینترم‌هایی که خروجی را 1 کنند به‌دست می‌آیند. این ۱ها با در نظر گرفتن تک تک 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 حاصل ضرب‌ها در زیر نقشه هر متغیر خروجی نوشته شده است.

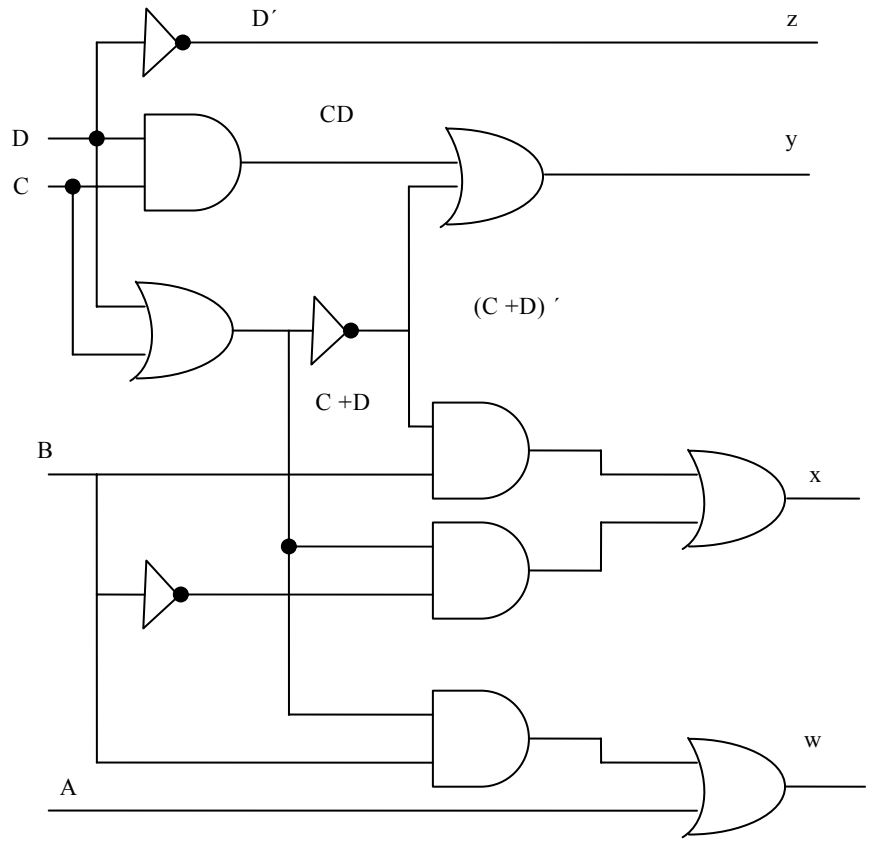
نموار دو سطحی را می‌توان مستقیماً از عبارات بولی حاصل از نقشه‌ها به دست آورد. البته فرم‌های متعدد دیگری نیز برای به‌دست آوردن نمودار منطقی که همین مدار را پیاده‌سازی کند وجود دارند. عبارات حاصل در شکل٦-٥ را به منظور استفاده از گیت‌های مشترک می‌توان درستکاری جبری نمود. این دستکاری جبری که در زیر آمده



شکل ٦-٥: نقشه‌های تبدیل BCD به افزونی-3

است انعطاف‌پذیری حاصل با سیستم‌های چند خروجی را وقتی با سه، یا چهار سطر و یا بیشتر پیاده‌سازی می‌شوند، نشان می‌دهد.

نمودار منطقی که این توابع را پیاده‌سازی می‌کند در شکل٦-٦ دیده می‌شود. مشاهده می‌شود. گیت OR که خروجی‌اش (C+D) است به نحوی در پیاده‌سازی هر سه خروجی به‌کار رفته است. بدون احتساب گیت‌های وارون‌گر در ورودی، پیاده‌سازی به صورت جمع حاصل‌ضرب‌ها به هفت گیت AND و سه گیت OR نیاز دارد. در



شکل٦-٦: نمودار منطقی برای تبدیل BCD به افزونی- 3

شکل٦-٦ 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

w = A + BC + BD
. = A + (C+D)
.
x = B´C + B´D +BC´D´
. = B´(C +D)+BC´D´
. =B´(C+D)+B(C+D)´
.
y = CD + C´D´
. = CD + (C+D)´
.
z = D´

فقط B و D نیاز به وارون‌گر دارند.

٦-٤ جمع‌کننده‌ها و تفریق‌گرهای دودویی

اصلی‌ترین عمل حسابی جمع دو رقم دودویی است. این جمع ساده شامل چهار عمل پایه به شرح ذیل می‌باشد:

$$0+0=0 \qquad 1+0=1$$
$$0+1=1 \qquad 1+1=1$$

سه عمل اول جمعی یک رقمی‌تولید می‌کنند، ولی وقتی هر دو بیت مضاف و مضاف‌الیه برابر 1 با شند، جمع دودویی از دو رقم تشکیل خواهد شد. با ارزش‌تر این نتیجه را *نقلی* می‌گویند. وقتی مضاف و مضاف‌الیه دارای ارقام باارزش‌تر بیشتری باشند، نقلی حاصل از جمع دو بیت با جفت بیت با ارزش‌تر 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 جمع‌کننده خوانده می‌شود. اسم مدارها به این علت انتخاب شده است که از دو نیم جمع‌کننده می‌توان در پیاده‌سازی یک جمع‌کننده کامل استفاده کرد.

یک جمع- تفریق‌گر دودویی مداری ترکیبی 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 نیم جمع‌کننده انجام می‌شود، و با استفاده از آن جمع‌کننده کامل را طراحی خواهیم کرد. با اتصال سری n جمع‌کننده کامل جمع دو عدد n بیتی تولید می‌گردد. مدار تفریق‌گر با افزودن مدار متمم ساز به آن ساخته می‌شود.

٦-٤-١ نیم جمع کننده

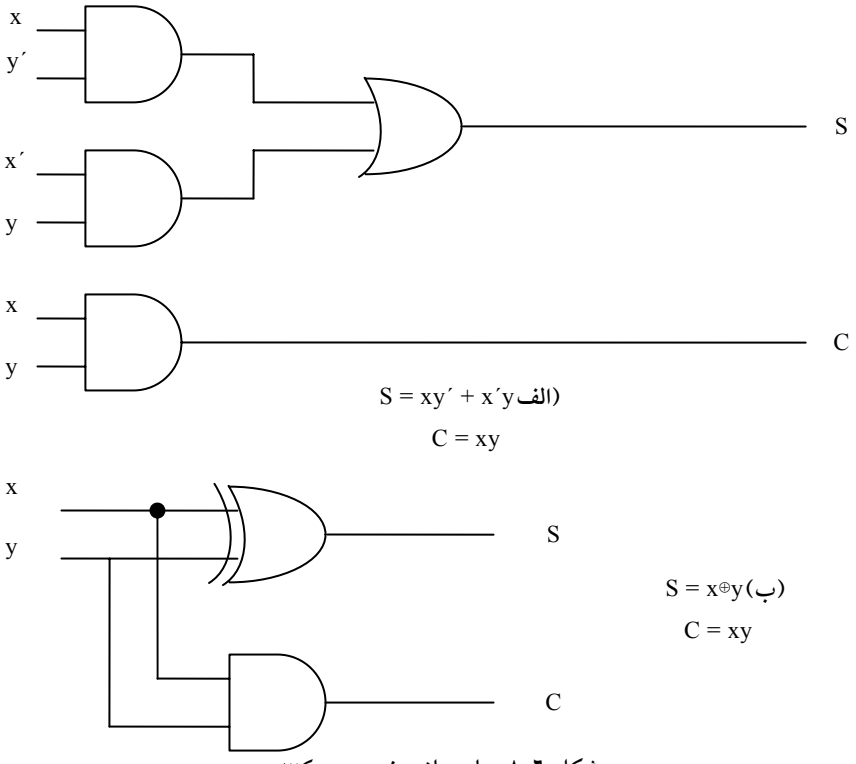
با توجه به توضیحات لفظی یک نیم جمع کننده، در می‌یابیم که مدار نیاز به دو ورودی دودویی و دو خروجی دودویی دارد. متغیرهای ورودی بیت‌های مضاف و مضاف‌الیه را

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 شکل٦-٧ نشان داده شده است.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

شکل ٦-٧: جدول نیم جمع کننده

خروجی C فقط هنگامی1 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 به‌دست می‌آیند. عبارات جمع حاصلضرب ساده عبارتند از:



(الف $S = xy' + x'y$

$C = xy$

(ب $S = x \oplus y$

$C = xy$

شکل ٦-٨: پیاده‌سازی نیم جمع کننده

$$S = x'y + xy'$$

و

$$C = xy$$

نمودار منطقی نیم جمع‌کننده پیاده شده با جمع حاصلضرب‌ها در شکل ٦-٨ (الف) دیده می‌شود. می‌توان آن را با گیت‌های XOR و AND طبق شکل ٦-٨ (ب) هم پیاده کرد. این نوع برای ساخت جمع‌کننده کامل از دو نیم جمع‌کننده به کار می‌رود.

٦-٤-٢ 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 با ارزش جمع شونده را نشان می‌دهند. ورودی سوم، z، نقلی حاصل از مکان کم ارزش‌تر 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 کم ارزش‌تر جمع را به‌دست می‌دهد. متغیر دودویی C نقلی خروجی را بیان گر است. جدول درستی جمع‌کننده کامل در جدول شکل ٦-٩ 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 هنگامی1 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.

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

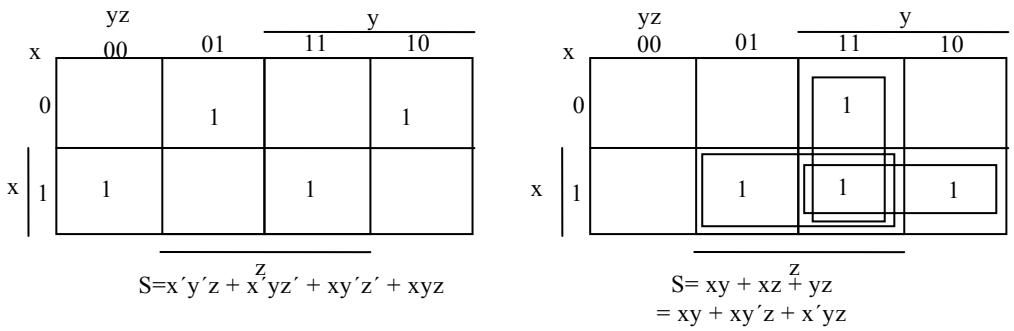
شکل٦-٩: جدول جمع‌کننده کامل

تفسیر بیت‌های ورودی و خروجی مدار ترکیبی در مراحل مختلف طراحی متفاوت است. به طور فیزیکی سیگنال‌های دودویی ورودی‌ها ارقامی‌دودویی تصور می‌شوند که به صورت حسابی باید با هم جمع شده و جمع دو رقمی‌را در خروجی تولید کنند. از طرف دیگر، در جدول درستی و یا هنگام پیاده‌سازی با گیت‌های منطقی، همان مقادیر به عنوان متغیرهای بول تعبیر می‌شوند. نقشه خروجی‌های جمع‌کننده کامل در شکل٦-۱۰ ملاحظه می‌شود. عبارات ساده شده عبارتند از:

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

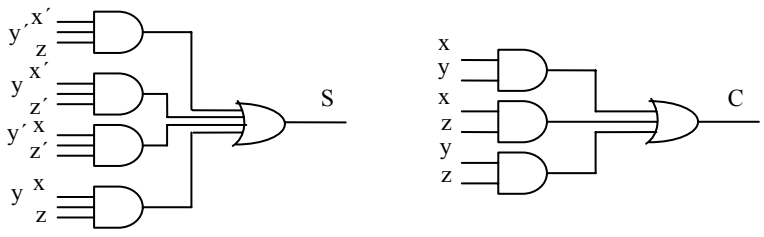
و

$$C = xy + xz + yz$$



شکل ٦-۱۰: نقشه جمع کننده کامل به صورت جمع حاصلضرب‌ها

نمودار منطقی پیاده شده به صورت جمع حاصلضرب‌ها در شکل٦-۱۱ مشاهده می‌شود.



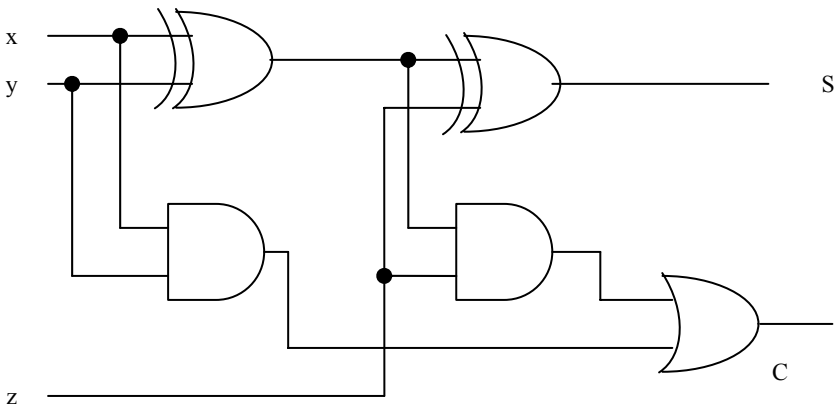
شکل ٦-۱۱: پیاده‌سازی جمع کننده کامل با جمع حاصلضرب‌ها

می‌توان با دو نیم جمع‌کننده و یک OR هم طبق شکل۶-۱۲ آن را پیاده‌سازی کرد. خروجی S از نیم جمع‌کننده دوم XOR متغیر z و خروجی نیم جمع‌کننده 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

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

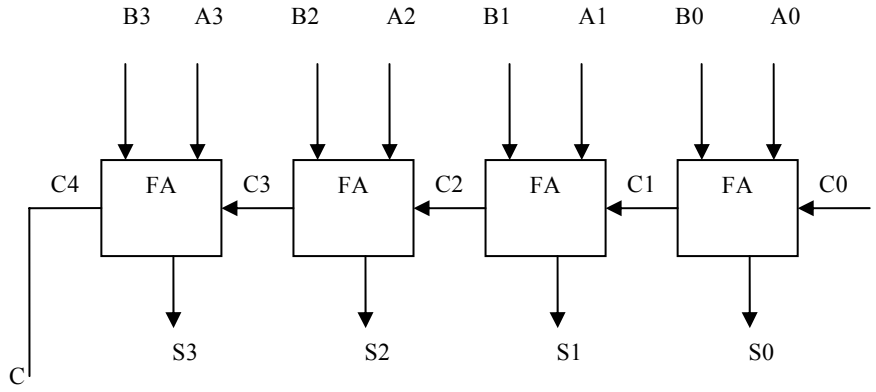


شکل۶-۱۲: پیاده‌سازی یک جمع‌کننده کامل با دو نیم جمع‌کننده و یک گیت OR

۶-۴-۳ 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 زنجیر وار بسته می‌شود. شکل٦-۱۳ اتصالات درونی مدار چهار جمع‌کننده کامل (FA)، برای تهیه جمع‌کننده دودویی 4 بیت با نقلی موج گونه را نشان می‌دهد. بیت‌های مضاف از A و مضاف‌الیه از B با اعداد اندیس‌دار از راست به چپ و با اندیس 0 در بیت کم ارزش‌تر مشخص شده است. نقلی‌ها به صورت زنجیر جمع کننده‌های کامل را به هم 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 رتبه‌بالاتر وصل می‌شود.



شکل ٦-۱۳: جمع کننده 4 بیت

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 به‌دست می‌آید.

اندیس i	:	3	2	1	0	
نقلی ورودی		0	1	1	0	Ci
مضاف		1	0	1	1	Ai
مضاف الیه		0	0	1	1	Bi
حاصل جمع		1	1	1	0	Si
نقلی خروجی		0	0	1	1	Ci+1

شکل ٦-١٤: جدول وضعیت جمع‌کننده دودویی

بیت‌ها با کمک جمع‌کننده کامل و از کم ارزش‌ترین مکان (اندیس 0) با هم جمع می‌شوند تا بیت حاصل جمع و نقلی را تشکیل دهند. نقلی ورودی C0 در کم ارزش‌ترین مکان باید 0 باشد. مقدارCi+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 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.

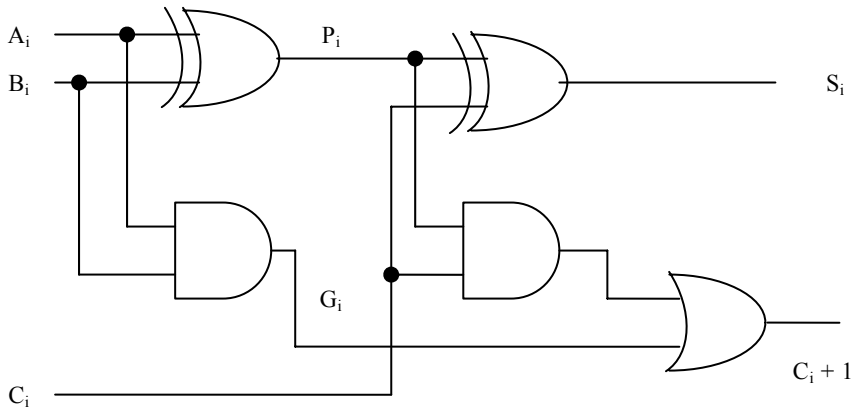
یک جمع‌کننده 4 بیت مثالی از یک قطعه استاندارد است. می‌توان از آن در کاربردهای متعددی مثل عملیات حسابی استفاده کرد. به خاطر بسپارید که طراحی این مدار با روش‌های کلاسیک به یک جدول درستی با $2^9 = 512$ وارده نیاز دارد زیرا 9 ورودی به مدار موجود است. با استفاده از روش بستن متوالی یک تابع استاندارد می‌توان به یک پیاده‌سازی ساده و مستقیم دست یافت.

٦-٤-٤ 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 تاخیر انتشار یک نمونه گیت ضرب در طبقات گیت‌ها در مدار.

طولانی‌ترین زمان تاخیر انتشار در یک جمع‌کننده زمانی است که نقلی برای انتشار در جمع‌کننده‌ای کامل لازم دارد. چون هر بیت از خروجی جمع به نقلی ورودیش 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 شکل۶-۱۳ در نظر بگیرید. به محض اعمال ورودی‌ها به جمع کننده، 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. به‌طور مشابهC_2 منتظر C_1 وC_1 منتظر C_0 خواهد بود. بنابر این پس از انتشار موج گونه 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 به‌دست آورد. به منظور سهولت این مدار در شکل ۶-۱۵ دوباره ترسیم شده است. متغیرهای ورودی و خروجی از اندیس i برای مشخص کردن شماره طبقه جمع‌کننده استفاده کرده اند سیگنال‌ها در Pi و Gi هنگامی‌به ثبات می‌رسند که از گیت‌های مربوطه شان انتشار یافته باشند. این دو سیگنال که در همه جمع کننده‌های کامل وجود دارند، به بیت‌های مضاف و مضاف‌الیه ورودی وابسته اند. سیگنال نقلی ورودی Ci از طریق



شکل ۶-۱۵: جمع کننده کامل با P و G

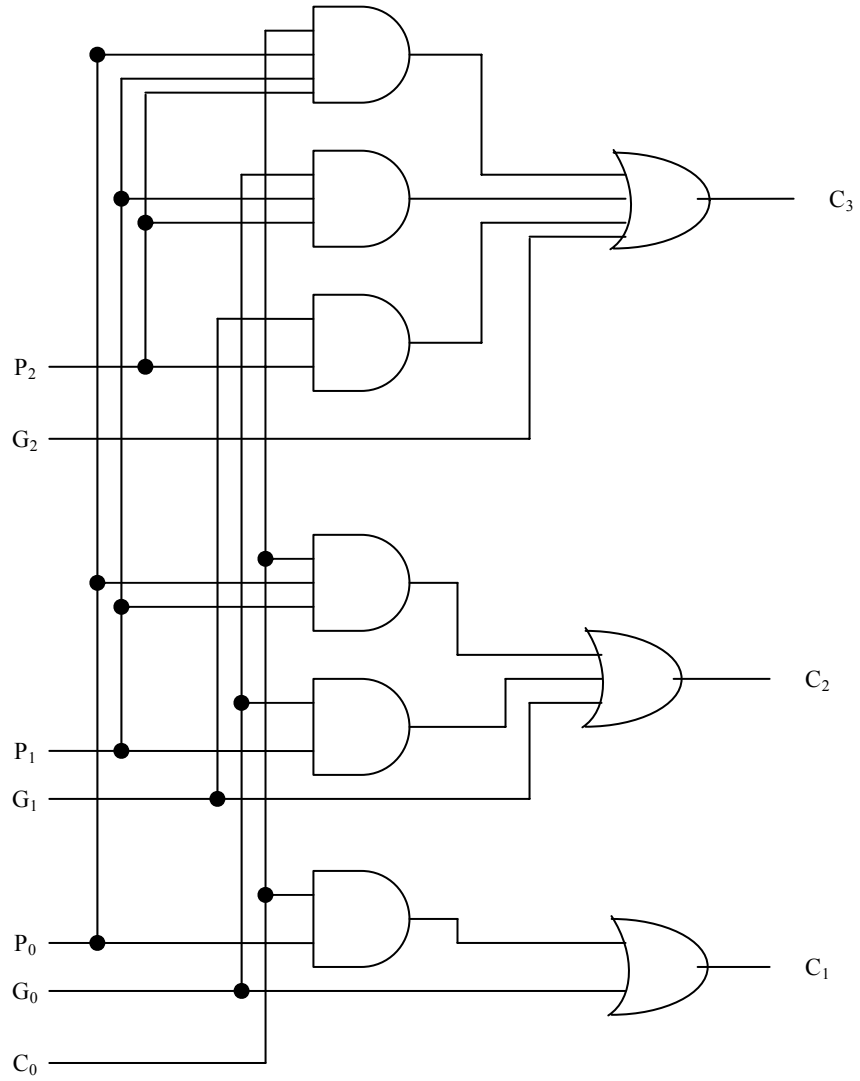
گیت AND یک گیت OR، که دو سطح گیت را تشکیل می‌دهند، به C_{i+1} می‌رسد. اگر در مجموع، چهار جمع‌کننده کامل وجود داشته باشد بین نقلی خروجی C_0 تا C_4، $2\times4=8$ بقه گیت وجود خواهد داشت. برای یک جمع‌کننده n بیت، 2n طبقه گیت وجود دارد تا نقلی ورودی از طریق آنها انتشار یافته و به خروجی برسد.

زمان انتشار نقلی، فاکتور محدود کننده‌ای روی سرعت جمع دو عدد می‌باشد. گرچه جمع کننده، یا هر مدار ترکیبی دیگر دارای مقداری در پایانه‌اش است، ولی خروجی‌ها صحیح نخواهند بود مگر اینکه فرصتی کافی برای انتشار سیگنال از گیت‌های متصل بهم 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. راه حل روشنی برای کاهش زمان تاخیر انتشار استفاده از گیت‌های سریعتر است. با این وجود، مدارهای فیزیکی در قابلیت خود محدودیت دارند. راه حل دیگر افزایش پیچیدگی مدار به طریقی است که زمان تاخیر نقلی کاهش یابد. چند تکنیک برای کاهش زمان انتشار نقلی در جمع کننده‌های موازی وجود دارد. رایج‌ترین تکنیک استفاده از اصل پیش بینی نقلی می‌باشد. مدار جمع‌کننده کامل شکل۶-۱۵ 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 ونقلی آن را می‌توان به صورت زیر تعریف کرد.



شکل ٦-١٦: نمودار منطقی مولد پیش بینی نقلی

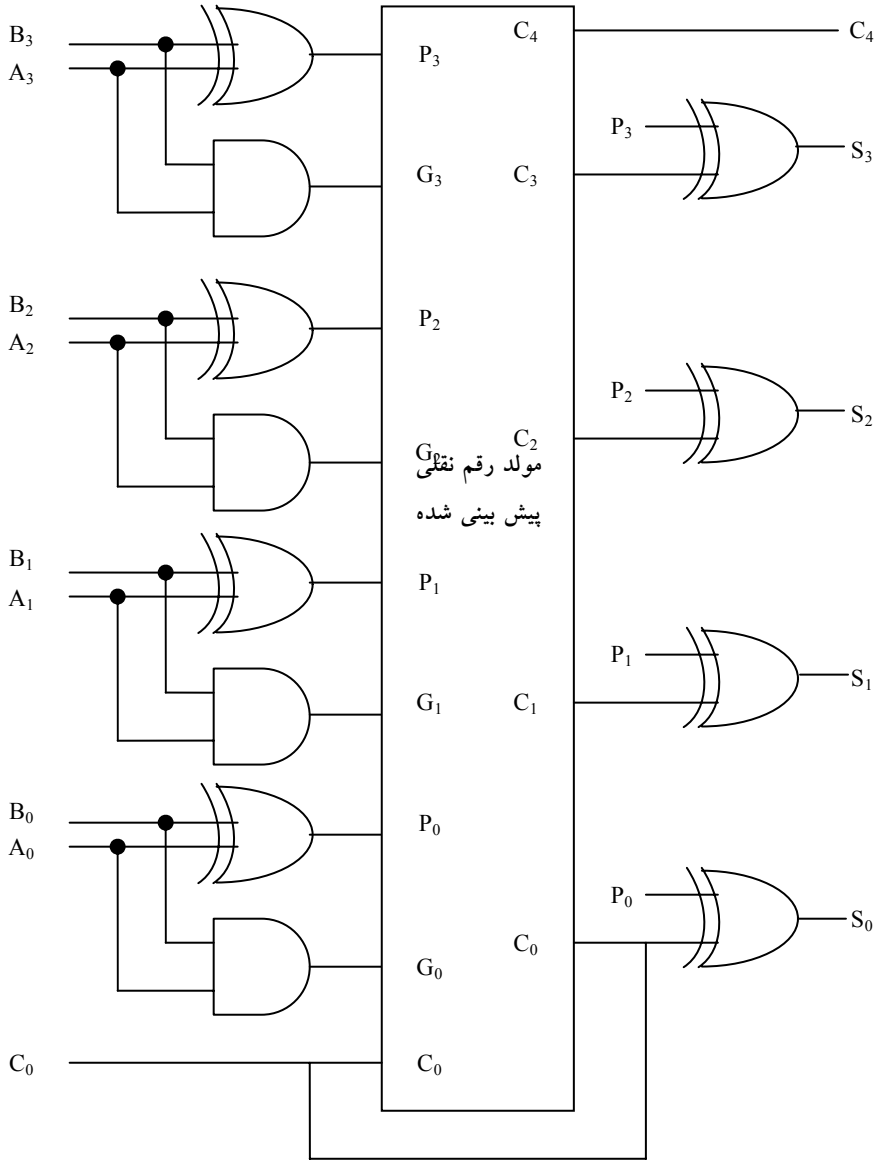
$S_i = P_i \oplus C_i$

$C_{i+1} = G_i + P_iC_i$

G_i را مولد نقلی نامند که وقتی هر دوA_i وB_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 هرC_i را با مقدار معادل قبلی جایگزین می‌کنیم:

C_0	**=نقلی ورودی**
C_1	$= G_0+P_0C_0$
C_2	$= G_1 + P_1C_1$
	$= G_1 + P_1 (G_0+ P_0C_0)$
	$= G_1 + P_1G_0 + P_1P_0C_0$
C_3	$= G_2+P_2C_2$
	$= G_2+P_2G_1+P_2P_1G_0+P_2P_1P_0C$

چون تابع بول برای هر نقلی خروجی برحسب جمع حاصل‌ضربها بیان شده است، هر تابع قابل پیاده‌سازی با یک طبقه گیت AND و بدنبال آن گیت OR (یا با دو طبقه NAND) است. سه تابع بول برای C_1،C_2 وC_3 در مولد نقلی پیش بینی شونده و در شکل٦-١٦ دیده می‌شوند. توجه کنید که C_3 نیاز ندارد به انتظار C_2 وC_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 ٦-١٧ مشاهده می‌شود.



شکل ٦-۱۷: جمع کننده 4 بیت با پیش بینی نقلی

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 شکل٦-١٦) و به عنوان ورودی به

گیت XOR دوم اعمال می‌گردند. همه نقلی‌های خروجی پس از یک تاخیر در دو طبقه گیت به طور همزمان تولید می‌شوند. بنابراین S_1 تا S_3 دارای زمان تاخیر 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.

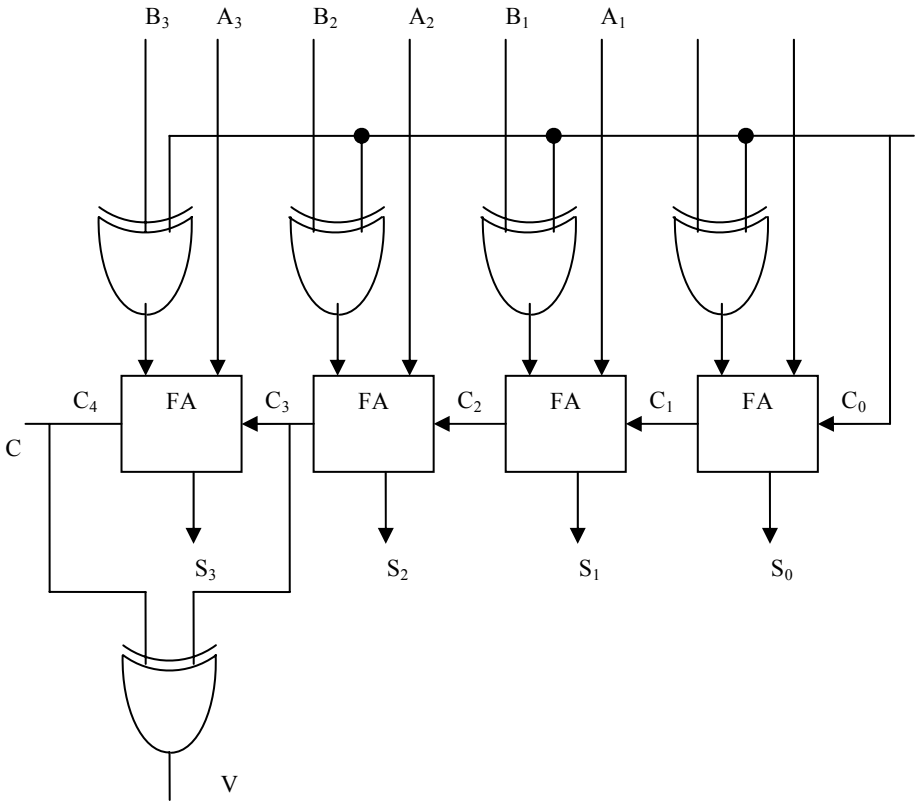
٦-٤-٥ تفریق دودویی

همانطور که در فصل اول در مبحث متمم‌ها مطرح شد، تفریق اعداد دودویی بی علامت با استفاده از متمم راحت تر 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 به‌دست آوردن متمم 1 و جمع آن با 1 محاسبه می‌کنند. متمم 1 را هم با وارون‌گر به‌دست آورده و عدد 1 را هم از طریق ورودی نقلی به آن اضافه می‌نمایند.

مدار تفریق گر A−B متشکل از یک جمع‌کننده با وارون‌گرهای واقع در بین ورودی B و ورودی مربوطه اش در تمام جمع‌کننده می‌باشد. نقلی ورودی C_0 به هنگام تفریق باید برابر با 1 شود. بنابراین عمل به صورت A به علاوه متمم 1 عدد B به علاوه 1 اجرا می‌شود. این عمل برابر با جمع A با متمم 2 عدد B خواهد بود. برای اعداد بی‌علامت اگر $A \geq B$ 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.

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- تفریق گر در شکل٦-١٨ دیده می‌شود. وقتی M=0 است، مدار یک جمع‌کننده و وقتی M=1 باشد، مدار یک تفریق گر 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 به علاوه 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 می‌شوند. در این حالت مدار یک عمل A به علاوه متمم 2 عدد B را انجام می‌دهد. (XOR با خروجی V، یک سرریز را شناسایی می‌نماید).



شکل٦-۱۸: جمع- تفریق‌گر

نکته‌ای که در این بین مطرح می‌گردد این است که در سیستم متمم علامت‌دار، اعداد دودویی هم چون اعداد بی علامت، با قوانین جمع و تفریق یکسانی ترکیب می‌شود. بنابراین، کامپیوترها نیاز به یک سخت‌افزار مشترک دارند تا هر دو نوع محاسبه را انجام دهند. کاربر یا برنامه نویس باید نتایج چنین جمع یا تفریقی را متفاوت تفسیر کنند و این به علامت‌دار یا بی علامت بودن اعداد بستگی دارد.

٦-٤-٦ مفهوم سرریز

هر گاه دو عدد n رقمی‌با هم جمع شوندو 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 یا 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 نگهداری عدد محدود بوده و نتیجه ای را که n+1 بیت دارد نمی‌توانند در خود جای دهند. به این دلیل، بسیاری از کامپیوتر ها 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 بی علامت‌اند. وقتی دو عدد بی علامت با هم جمع شوند، یک سرریز از نقلی با ارزش‌ترین مکان تشخیص داده می‌شود. در حالتی که اعداد علامت‌دار باشند، سمت چپ ترین 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 هنگامی‌که 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 علامت‌دار 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 اینکه روش به خوبی کار کند متمم 2 باید از طریق به‌دست 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 شکل۶-۱۸ دیده می‌شود. اگر دو عدد دودویی بی علامت 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+1 ام علامت واقعی است به یک مکان بیرونی منتقل شده است.

٦-٤-٧ 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 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 با ارزش 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 این گونه 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.

٦-٤-٨ 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. جمع کننده، حاصل جمع را به فرم دودویی اجرا می‌کند و نتیجه تولید شده بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول(٥-٤) مشاهده می‌شود که با K، z_8، z_4، z_2، z_1 بر چسب خورده‌اند.K یک رقم نقلی است و اندیس زیر حرف z وزن های 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 در آید و نیز باید آن طور که در زیرستون 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 بی اعتباری 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.

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

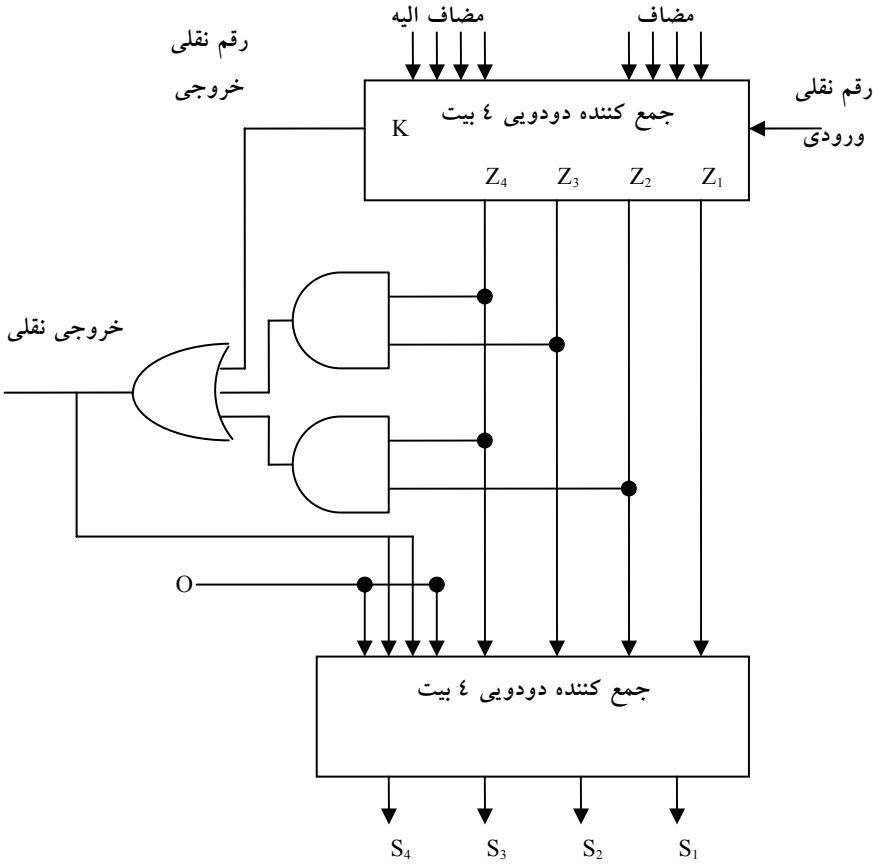
شکل ٦-١٩: جدول طریقه طراحی جمع کننده BCD

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 شش گانه از 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_4و z_2 مراجعه می‌کنیم که در 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:

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

وقتی C=1 است، لازم است 0110 به جمع دودویی اضافه شود تا یک نقلی خروجی برای طبقه بعدی فراهم شود.



شکل ٦-۲۰: نمودار بلوکی یک جمع کننده BCD

یک جمع‌کننده BCD که دو رقم BCD را با هم جمع کرده و ارقام جمع را به BCD نشان می‌دهد در شکل٦-۲۰ ملاحظه می‌گردد. دو رقم دهدهی همراه با نقلی ورودی ابتدا در جمع‌کننده 4 بیت فوقانی جمع شده و حاصل جمع دودویی تولید می‌کنند.

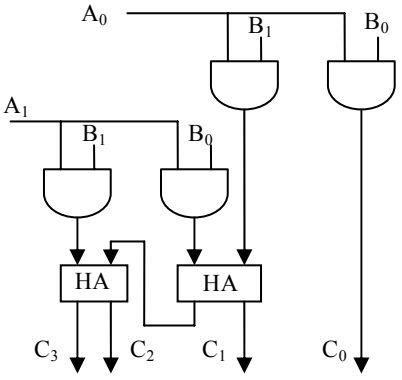
وقتی نقلی خروجی برابر 0 باشد، چیزی به جمع دودویی اضافه نمی‌شود. وقتی این 6 نقلی برابر 1 باشد، عدد دودویی 0110 از طریق جمع‌کننده 4 بیت پایینی به جمع دودویی اضافه می‌گردد. نقلی خروجی تولید شده در جمع‌کننده پایین می‌تواند صرف نظر شود زیرا اطلاعاتی را حمل می‌کند که قبلاً در پایانه نقلی خروجی وجود داشته است. یک جمع‌کننده دهدهی موازی کهn رقم دهدهی را جمع می‌کند به n طبقه جمع‌کننده BCD نیاز دارد. نقلی خروجی هر طبقه باید به ورودی طبقه بالاتر متصل گردد.

٦-٥ ضرب دودویی

یکی از مهمترین مدارت دیجیتال مورد استفاده در سیستم‌ها مدار ضرب اعداد می‌باشد. ضرب اعداد دودویی همچون ضرب اعداد دهدهی انجام می‌شود. هر بیت مضروب، در کم ارزش‌ترین بیت مضروب‌فیه ضرب می‌شود. چنین حاصلضربی، حاصلضرب جزیی خوانده می‌شود. حاصلضرب‌های جزیی هر بار یک مکان به چپ انتقال می‌یابند. حاصلضرب نهایی از جمع حاصلضرب‌های جزیی به‌دست می‌آید.

برای این که ببینیم که یک ضرب کننده چگونه با یک مدار ترکیبی پیاده می‌شود،

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



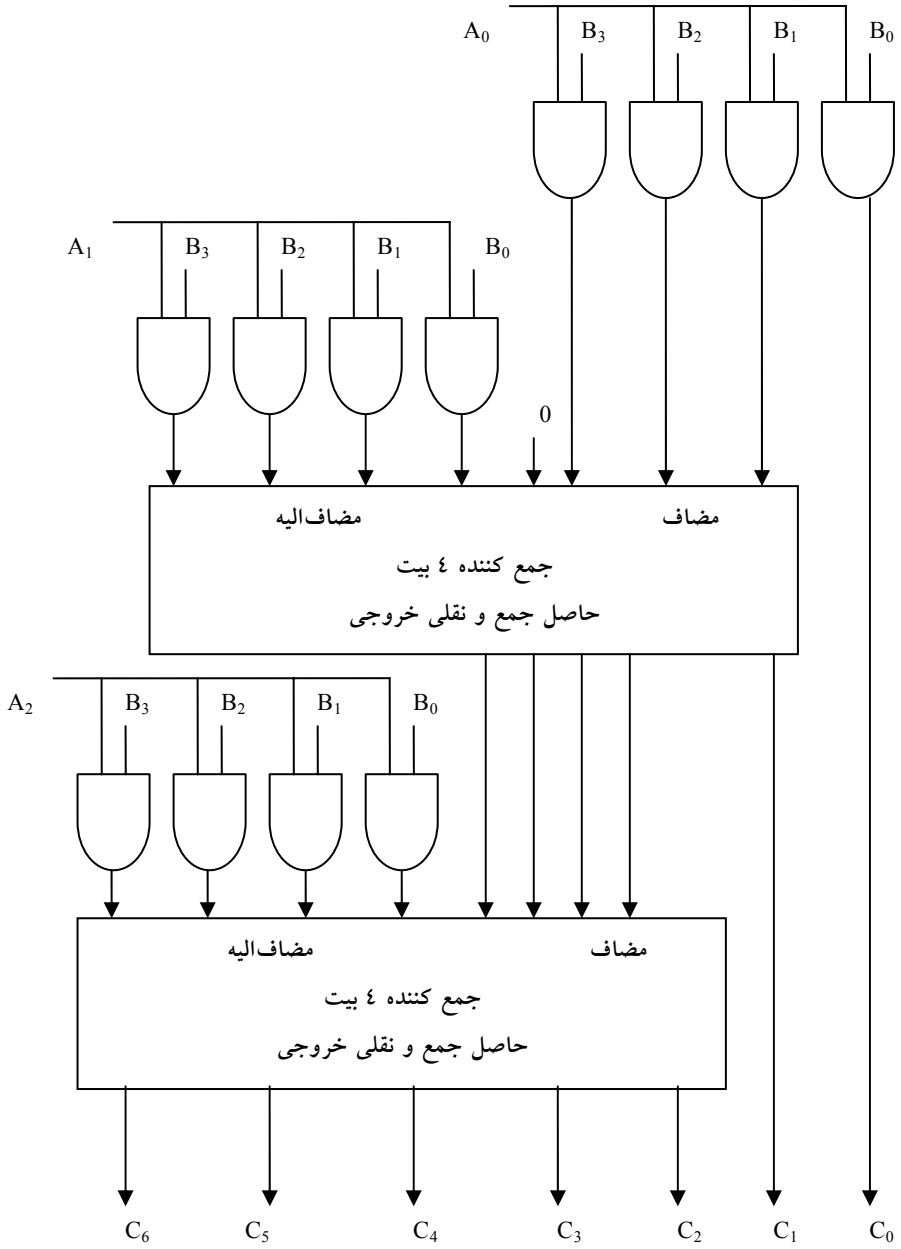
شکل ٦-٢١: ضرب دودویی 2 بیت در 2 بیت

ضرب اعداد دو بیت را طبق شکل٦-٢١ در نظر بگیرید. بیت‌های مضروب، B_0 و B_1 و بیت‌های مضروب فیه A_0 و A_1 و حاصلضرب $C_3C_2C_1C_0$ فرض می‌شوند. اولین حاصلضرب جزیی با ضربA_0 در$B_1 B_0$ حاصل می‌گردد. ضرب دو بیت مثلA_0 وB_0 هنگامی1 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 $B_1 B_0$ به‌دست می‌آید که باید یک مکان هم به چپ جابجا شود. دو حاصلضرب جزیی به وسیله مدار دو نیم جمع‌کننده (HA) با هم جمع می‌شوند. معمولاً در حاصلضرب‌های جزیی بیت‌های بیشتری وجود دارند و لازم است از تمام جمع‌کننده برای تولید جمع حاصلضرب‌های جزیی استفاده شود. توجه کنید لزومی‌ندارد که کم ارزش‌ترین 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 مضروب فیه 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 کننده ای 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 مضروب فیه$A_2A_1A_0$ 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 شکل٦-٢٢ دیده می‌شود.



شکل ٦-۲۲: ضرب کننده دودویی 4 بیت در 3 بیت

٦-٦ مقایسه گر مقدار

مقایسه دو عدد عملی است که توسط آن بزرگتر بودن، کوچکتر بودن یا مساوی بودن آنها معین می‌شود. یک مقایسه گر 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.

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 مقایسه ممکن است مقدار قابل توجهی نظم در خود داشته باشد. توابع دیجیتال که ذاتاً دارای نظمی‌درونی 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. دو عدد A و B را که هر کدام چهار رقم دارند در نظر بگیرید. ضرایب اعداد را به ترتیب نزولی زیر 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

$$A_0 = B_0 \text{ ، } A_1 = B_1 \text{ ، } A_2 = B_2 \text{ ، } A_3 = B_3$$

وقتی که اعداد دودویی باشند، ارقام 1 یا 0 اند و رابطه تساوی هر جفت بیت به طور منطقی با یک تابع XOR نمایش داده می‌شود.

$$X_i = A_iB_i + A'_iB'_i \text{ for } i=0,1,2,3$$

که در آن $x_i=1$ به شرطی صحت دارد که بیت‌های مکان i ام 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:

$$(A = B) = x_3x_2x_1x_0$$

عدد دودویی (A = B) هنگامی 1 است که فقط همه جفت ارقام برابر باشند.

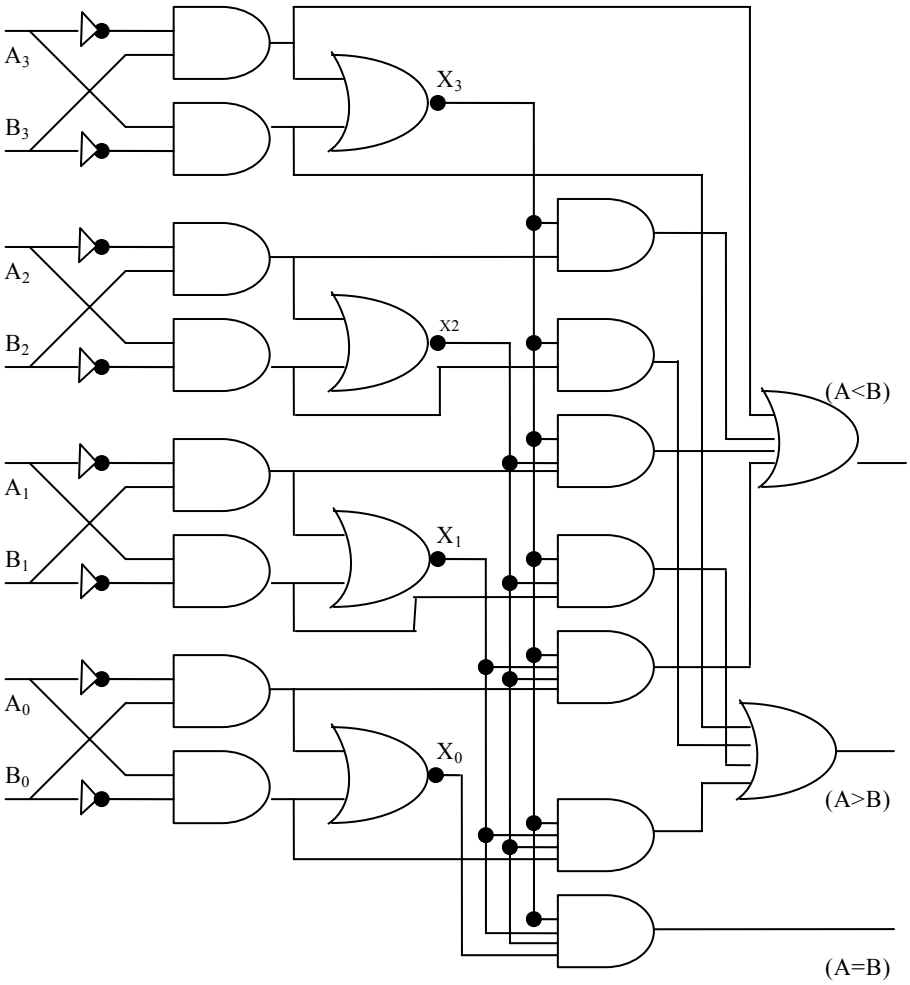
برای اینکه معین کنیم آیا A بزرگتر یا کوچکتر از B است، اندازه‌های نسبی دو رقم را با شروع از باارزش‌ترین مکان آغاز می‌نماییم. اگر دو رقم مساوی باشند، دو رقم پایین‌تر را مقایسه می‌کنیم. این مقایسه تا رسیدن به یک جفت غیر مساوی ادامه خواهد داشت و اگر در این هنگام بیت متعلق به A برابر 1 و B برابر 0 باشد، نتیجه می‌گیریم A>B است. اگر بر عکس 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:

$$(A>B) = A_3B'_3 + x_3A_2B'_2 + x_3x_2A_1B'_1 + x_3x_2x_1A_0B'_0$$
$$(A<B) = A'_3B_3 + x_3A'_2B_2 + x_3x_2A'_1B_1 + x_3x_2x_1A'_0B_0$$

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 مقایسه گر مقدار 4 بیتی در شکل٦-۲۳ 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 به‌دست آوردن مدارهای مقایسه‌گر اندازه برای اعداد دودویی با بیش از چهار بیت از این مثال کاملاً آشکار است.



شکل ٦-۲۳: مقایسه‌گر مقدار چهار بیتی

سؤالات

۱- یک مدار ترکیبی با سه ورودی و یک خروجی طراحی نمایید.

۲- اگر تولید و انتشار نقلی را به‌صورت $P_i = A_i + B_i$ و $G_i = A_iB_i$ داشته باشیم، نشان دهید که نقلی خروجی و جمع خروجی یک جمع‌کننده کامل به صورت زیر خواهد بود.

$$C_{i+1} = (C'_iG'_i + P'_i)'$$

$$S_i = (P_iG'_i) \oplus C_i$$

۳- یک مدار ترکیبی طراحی کنید تا متمم ۹ یک رقم BCD را تولید نماید.

٤- یک ضرب کننده دودویی برای ضرب دو عدد چهار بیتی طراحی نمایید.

۵- یک مدار جمع‌کننده کامل BCD بسازید.

٦- یک مدار ترکیبی افزایش‌گر ٤ بیتی را با چهار نیم جمع‌کننده طراحی نمایید.

فصل ۷

مدارهای رمزگذار و رمزگشا

هدف کلی

در این فصل مباحث اصلی مربوط به مدارهای رمزگذار و رمزگشا مورد بحث و بررسی قرار گرفته و روش پیاده‌سازی این نوع مدارها شرح داده خواهد شد. همچنین مولتی‌پلکسرها نیز در این فصل بررسی خواهند شد.

هدف ساختاری

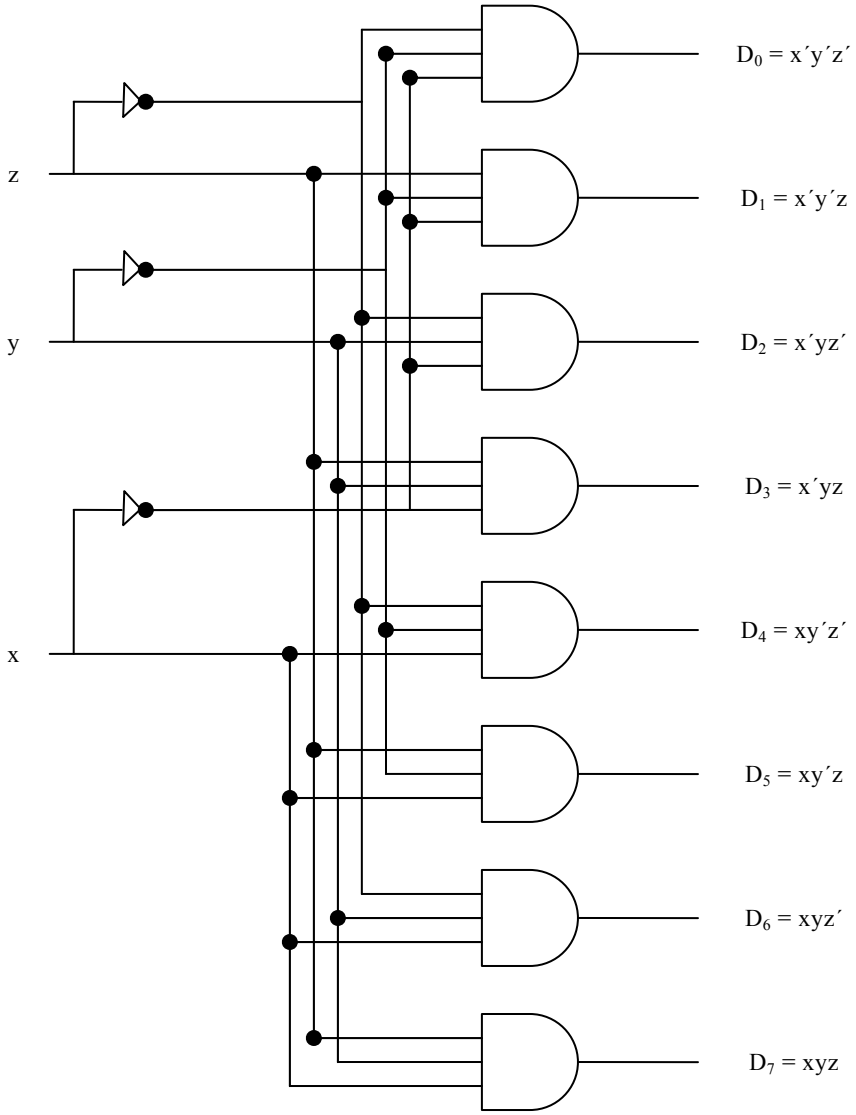
در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مدارهای رمزگشا (دیکدر)
- مدارهای رمزگذار (انکدر)
- پیاده‌سازی مدارهای رمزگذار و رمزگشا
- مولتی‌پلکسرها

یکی از مباحث مطرح در طراحی مدارات دیجیتال، ایجاد مدارات رمزگذار و رمزگشا می‌باشند. در این فصل این دو نوع مدار به تفصیل مورد بحث و بررسی قرار خواهند گرفت.

۷-۱ مدارات رمزگشا (دیکدر)

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



شکل ۷-۱: دیکدر 3 به 8

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 بی استفاده باشد، دیکدر ممکن است خروجی 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 شکل۷-۱ 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. کاربر رایج این نوع دیکدر، تبدیل دودویی به هشت هشتی است. متغیرهای ورودی یک عدد دودویی را نشان می‌دهند، و خروجی بیانگر هشت رقم در سیستم اعداد مبنای هشت است. با این وجود دیکدر 3 به 8 خط را می‌توان برای دیکدر کردن هر کد 3 بیت در تولید هشت خروجی، یکی برای هر عنصر از کد، به کار برد.

ورودی‌ها			خروجی‌ها							
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**

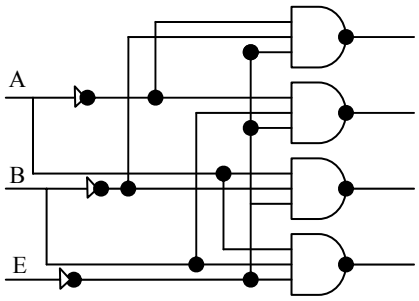
شکل ۷-۲: جدول درستی دیکدر 3 به 8 خط

طرز کار یک دیکدر می‌تواند با لیستی در جدول درستی شکل۷-۲ آشکار شود. برای هر ترکیب ورودی ممکن، هفت خروجی وجود دارد که برابر 0 هستند و فقط

یکی از آنها برابر 1 است. خروجی حاوی 1 بیانگر مینترم عدد دودویی حاضر در خطوط ورودی است.

۷-۱-۱ پیاده‌سازی دیکدر با گیت NAND

از آنجائیکه پیاده‌سازی گیت NAND ساده‌تر است، لذا 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 شکل۷-۳ دیده می‌شود.



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

(الف) دیاگرام منطقی **(ب) جدول درستی**

شکل۷-۳: دیکدر 2 به 4 خط با ورودی فعال‌ساز

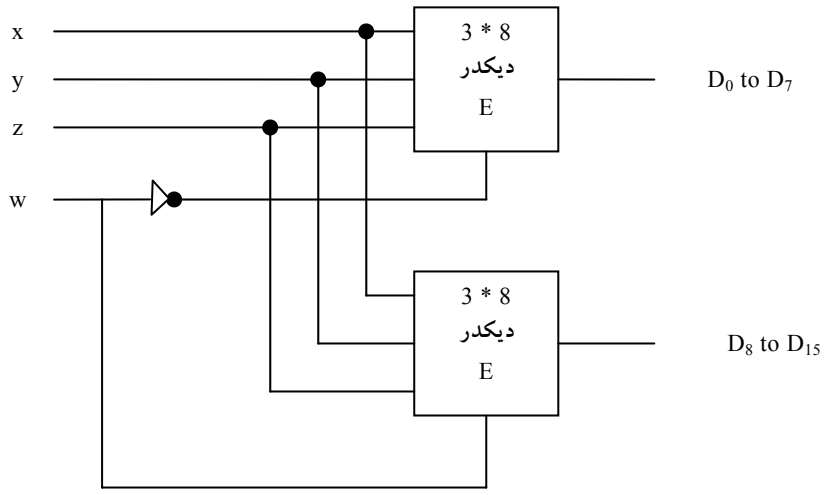
مدار با خروجی‌های متمم شده و یک ورودی فعال‌ساز متمم شده کار می‌کند. دیکدر هنگامی‌که E برابر 0 باشد فعال می‌گردد. همانطور که توسط جدول درستی مشاهده می‌شود، هر بار تنها یک خروجی برابر 0 بوده و دیگر خروجی‌ها در وضعیت 1 قرار دارند. وقتی E=1 باشد مدار غیر فعال است و به دو ورودی دیگر بستگی ندارد. هنگام غیر فعال 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 بر آورده سازند تا مدار فعال شود.

یک دیکدر با ورودی فعال‌ساز می‌تواند به عنوان یک دی مولتی‌پلکسر عمل کند (مباحث مربوط به مولتی‌پلکسر و دی مولتی‌پلکسر در همین فصل توضیح داده خواهد شد). دی مولتی‌پلکسر 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 ۷-۳ را می‌توان به عنوان یک دی مولتی‌پلکسر 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 چهار خروجی دارد، ولی اطلاعات ورودی تنها به یکی از خروجی‌ها هدایت می‌شود. این خروجی با ترکیب دودویی دو خط 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 در مولتی‌پلکسر با استفاده از یک مدار حاصل می‌شود، یک دیکدر با ورودی فعال‌ساز را دیکدر/ دی مولتی‌پلکسر هم می‌خوانند.

برای تهیه مدار دیکدر بزرگتر می‌توان دیکدرها با ورودی‌های فعال‌ساز را به هم متصل کرد. شکل۷-٤ 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



شکل ۷-٤: دیکدر 16× 4 ساخته شده با دو دیکدر 8 ×3

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 طور کلی ورودی‌های فعال‌ساز ابزارهای مناسبی برای اتصالات درونی دو یا چند قطعه استاندارد برای گسترش آنها با عملکردی مشابه و ورودی‌ها و خروجی‌های بیشتر است.

۷-۱-۲ پیاده‌سازی مدار منطقی ترکیبی با دیکدر

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 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.

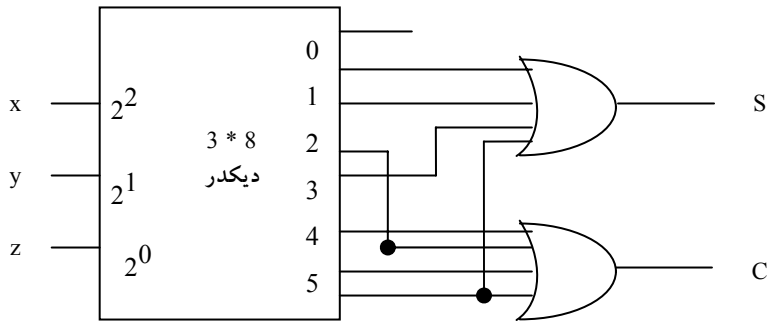
روال پیاده‌سازی یک مدار ترکیبی با دیکدر و گیت‌های OR، لازم می‌دارد که تابع بول مدار برحسب جمع مینترم‌ها بیان شود. سپس یک دیکدر برای تولید همه

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 به‌کار می‌برد تشریح می‌گردد.

با توجه به جدول درستی جمع‌کننده کامل که در شکل۷-۵ ارائه شده است، توابع مدار ترکیبی را به صورت مجموع مینترم‌ها به‌دست می‌آوریم:

$$S(x,y,z) = \Sigma(1, 2, 4, 7)$$
$$C(x,y,z) = \Sigma(3, 5, 6, 7)$$

چون سه ورودی و جمعاً هشت مینترم وجود دارد، به یک دیکدر 3 به 8 خط احتیاج است. پیاده‌سازی در شکل۷-۵ 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 به کار می‌برد.



شکل ۷-۵: پیاده‌سازی یک جمع کننده کامل با دیکدر

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 مینترم‌ها 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 شکل۷-۳ 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.

۷-۲ مدارات رمز گذار (انکدر)

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 شکل۷-٦ 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 هنگامی1 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 شکل۷-٦ تعریف شد دارای این محدودیت است که در آن در هر لحظه از زمان فقط یک ورودی فعال می‌باشد. اگر دو ورودی به طور هم زمان فعال شوند، خروجی ترکیبات نامفهومی‌را 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 نه3 ونه6 را نمایش می‌دهد. برای حل این مشکل، مدارهای انکدر باید اولویتی در ورودی ایجاد کنند تا مطمئن شویم که فقط یک ورودی انکدر شده است. اگر اولویت بالاتر را با اندیس های 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.

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

شکل ۷-٦: جدول درستی یک انکدر هشت هشتی به دودویی

مشکل دیگری که در انکدرهشت 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.

۷-۲-۱ انکدر اولویت

انکدر اولویت مداری است که تابع اولویت دهی را در خود دارد. عملکرد این نوع انکدر چنان است که اگر ده 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 ۷-٦ 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 و0 =V 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 Xها در ستون ورودی‌ها در کاهش جدول درستی نقش عمده ای 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.

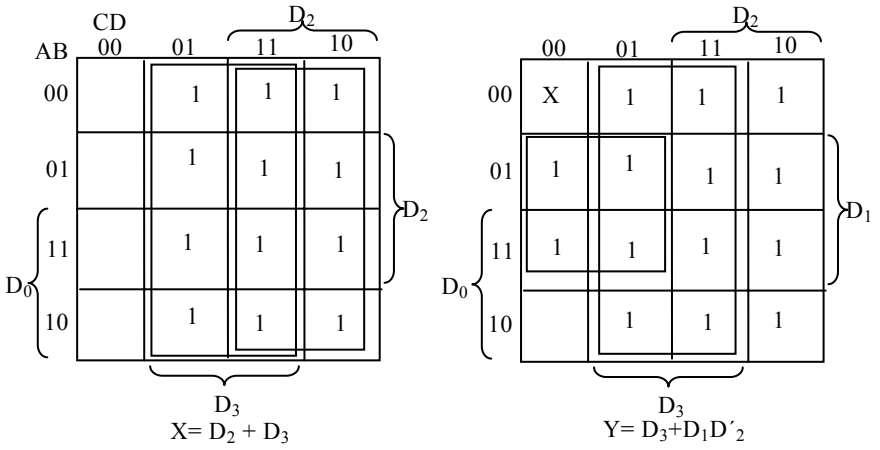
ورودی‌ها				خروجی‌ها		
D_0	D_1	D_2	D_3	X	Y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

شکل ۷-۷: جدول درستی انکدر اولویت

طبق جدول ارائه شده در شکل۷-۷، 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. اگرD_2=1 شود خروجی 10 می‌گردد به شرطی که D_3= 0 باشد، ولی این خروجی مستقل از دو ورودی با اولویت پایین تر است. اگر دیگر

ورودی‌های با اولویت تر 0 باشند، خروجی مربوط به D_1 تولید می‌گردد و به همین ترتیب.

نقشه ساده‌سازی خروجی‌های x و y در شکل۷-۸ نشان داده شده است. مینترم‌های دو تابع از جدول شکل ۷-۷ استنتاج شده‌اند.



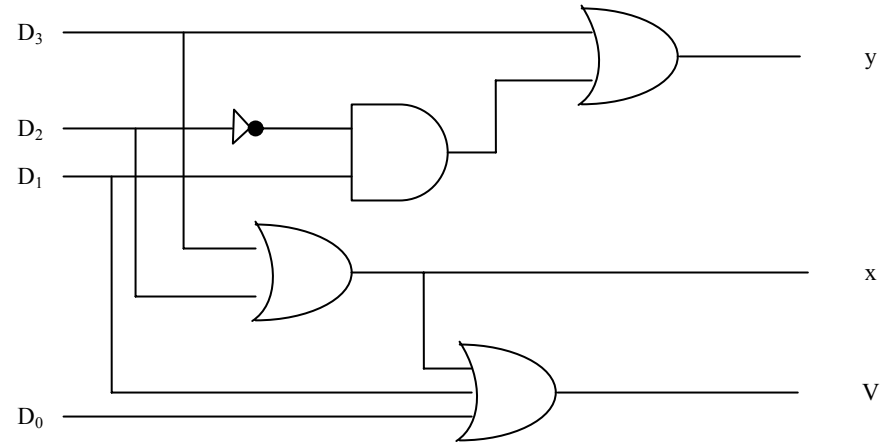
شکل ۷-۸:. نقشه یک انکدر اولویت دار

$$x = D_2 + D_3$$

$$y = D_3 + D_1D_2'$$

$$v = D_0 + D_1 + D_2 + D_3$$

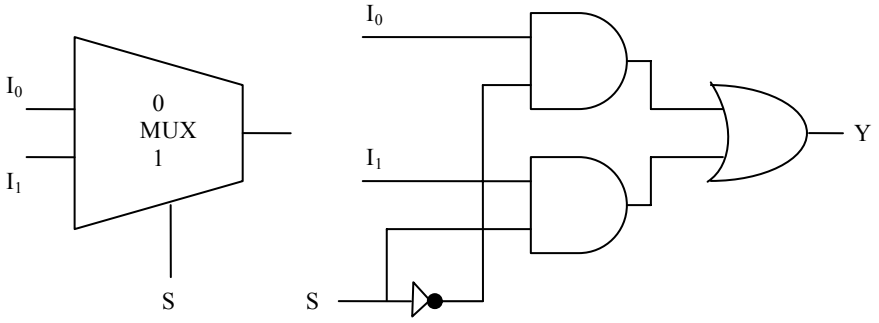
گرچه جدول دارای تنها پنج سطر است، وقتی هر X در هر سطر ابتدا با صفر و سپس با 1 جایگزین شود، آنگاه تمام 16 ترکیب ورودی را خواهیم داشت. مثلاً سطر چهارم در جدول با XX10 چهار مینترم 0010، 0110، 1010 و1110 را نمایش می‌دهد. عبارات بول ساده شده برای انکدر اولویت از نقشه‌ها حاصل می‌شود شرط خروجی V یک تابع OR از همه متغیرهای ورودی است. انکدر اولویت برطبق توابع بولی زیر در شکل۷-۹ پیاده‌سازی شده است.



شکل ۷-۹: پیاده‌سازی تابع بول با یک مولتی‌پلکسر

۷-۳ مولتی‌پلکسر

یک مولتی‌پلکسر 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.



(ب) نمودار بلوکی **(الف) نمودار منطقی**

شکل ۷-۱۰: مولتی‌پلکسر 2 به 1

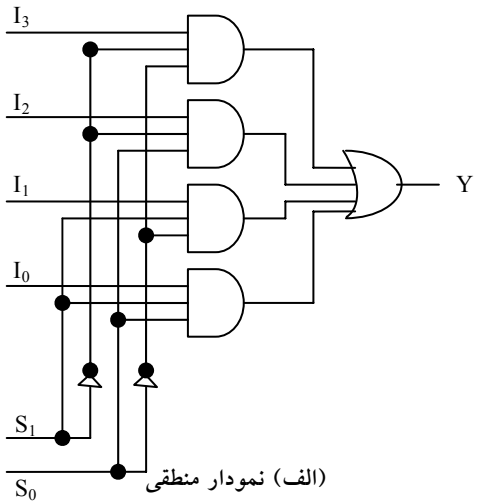
یک مولتی‌پلکسر 2 به 1 یکی از دو منبع 1 بیت را طبق شکل۷-۱۰ به یک مقصد مشترک متصل می‌کند. مدار دارای دو خط ورودی داده، یک خروجی و یک خط انتخاب S است. وقتی S=0 با شد، گیت AND فوقانی فعال شده و I_0 به خروجی راه می‌یابد. وقتی S = 1 باشد، گیت AND تحتانی فعال شده و I_1 به خروجی متصل می‌شود.

مولتی‌پلکسر مثل یک کلید الکترونیک عمل کرده و یکی از دو منبع را انتخاب می‌نماید. نمودار بلوکی یک مولتی‌پلکسر گاهی به شکل ذوزنقه شکل ۷-۱۰ (ب) نشان داده می‌شود. این مدار چگونگی انتخاب و هدایت منابع متعدد داده را به یک مقصد نشان می‌دهد. مولتی‌پلکسر اغلب با نمودار های بلوکی و کلمه MUX نشان داده می‌شود.

یک مولتی‌پلکسر 4 به 1 در شکل۷-۱۱ دیده می‌شود. هر یک از چهار ورودی I_0 تا I_3 به یک ورودی گیت AND اعمال می‌شود. خطوط انتخاب S_1 و S_0برای انتخاب گیت AND خاص دیکدر می‌شوند. خروجی گیت‌های AND به یک گیت OR اعمال می‌شوند تا خروجی 1 خط را ایجاد کنند.

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(ب) جدول تابع



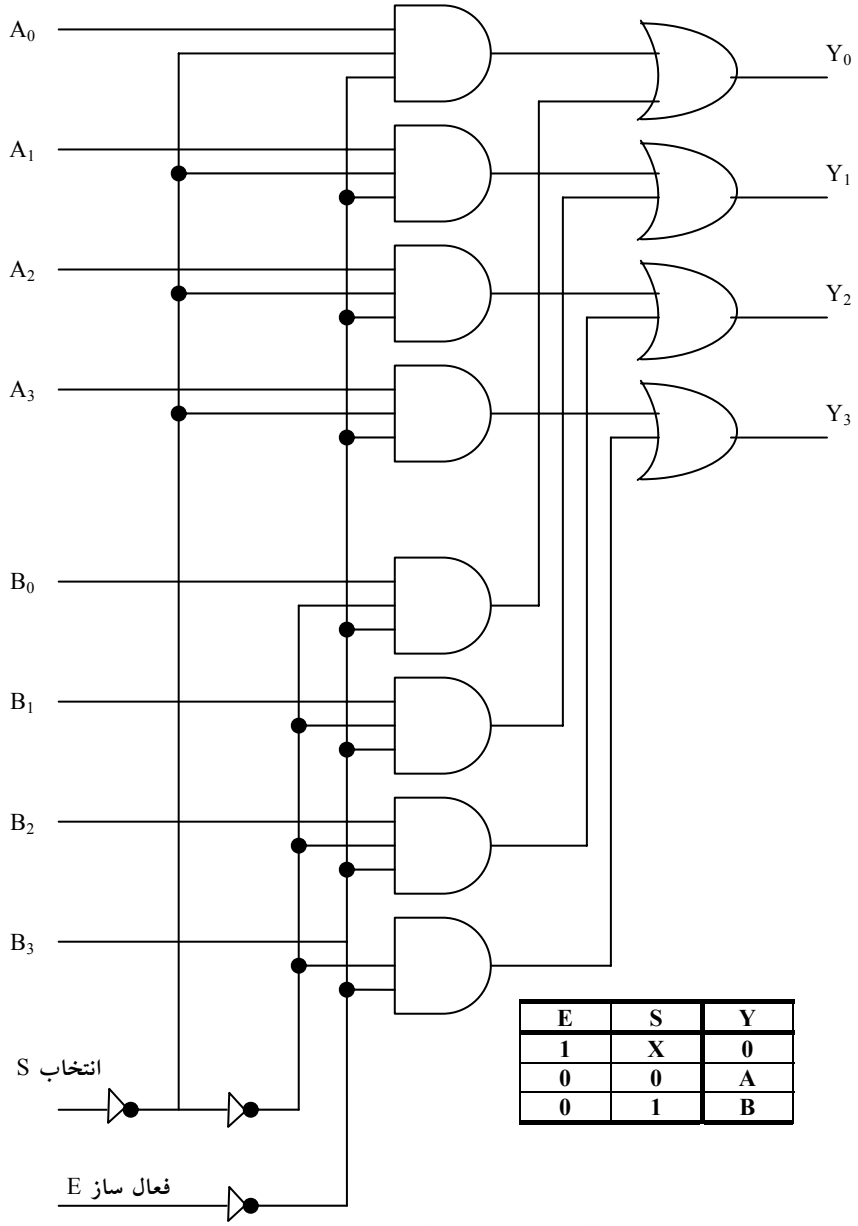
(الف) نمودار منطقی

شکل ۷-۱۱: مولتی‌پلکسر 4 به 1

جدول تابع، ورودی را که از مولتی‌پلکسر عبور کرده نشان می‌دهد. برای نمایش عمل مدار، حالتی را که $S_1S_0 = 10$ 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 خروجی شان 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 مولتی‌پلکسر را انتخابگر داده هم می‌خوانند، زیرا یکی از چند ورودی را انتخاب کرده و اطلاعات دودویی را به خط خروجی هدایت می‌کند.

وجود گیت‌های AND و وارون‌گرها در مولتی‌پلکسر، مدار دیکدر را به خاطر می‌آورد، و به علاوه آنها خطوط ورودی انتخاب را دیکدر می‌کنند. به طور کلی یک مولتی‌پلکسر 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 مولتی‌پلکسر با 2^n خط ورودی داده و تنها خط خروجی‌اش مشخص می‌شود. همچون دیکدر، مولتی‌پلکسرها هم ممکن است خط فعال‌سازی داشته باشندتا عملکرد کل قطعه را کنترل کنند. وقتی که ورودی فعال‌ساز در وضعیت غیر فعال قرار دارد، خروجی‌ها غیر فعالند، و وقتی در حالت فعال خود قرار گیرد، مدار به عنوان یک مولتی‌پلکسرمعمولی عمل می‌کند.

مدارهای مولتی‌پلکسر را می‌توان برای تهیه مولتی‌پلکسر چند بیتی با هم ترکیب کرد. به منظور تشریحی بر این مطلب، یک مولتی‌پلکسر 2 به 1 چهارتایی در شکل۷-۱۲ نشان داده شده است. مدار دارای چهار مولتی‌پلکسر 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 به‌طور مشابه Y_1 هم می‌تواند مقدار A_1 یاB_1 را داشته باشند، و به همین ترتیب. خط انتخاب ورودی یکی از خطوط ورودی را در هر یک از چهار مولتی‌پلکسر انتخاب می‌کند. خط فعال‌ساز E با به هنگام کار معمولی فعال شود.



E	S	Y
1	X	0
0	0	A
0	1	B

شکل ۷-۱۲: مولتی‌پلکسر 2 به 1 چهارتایی

گرچه مدار حاوی چهار مولتی‌پلکسر 2 به 1 است، ولی ما بیشتر علاقه‌مندیم 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.

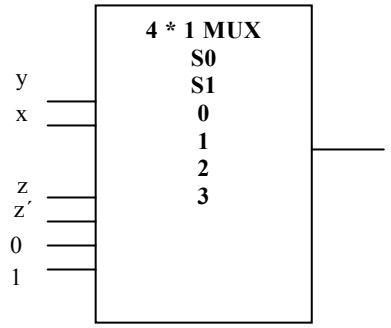
۷-۳-۱ پیاده‌سازی تابع بول

همانطور که می‌دانید با افزودن یک گیت OR به خروجی‌های یک دیکدر می‌توان از آن برای پیاده‌سازی توابع بول استفاده کرد. با بررسی نمودار منطقی یک مولتی‌پلکسر 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 مولتی‌پلکسر تولید می‌شوند. هر مینترم به وسیله ورودی‌های داده انتخاب می‌شود. این مطلب روشی را برای پیاده‌سازی هر تابع n متغیر به وسیله یک مولتی‌پلکسر که دارای 2^n ورودی داده و n خط ورودی انتخاب است، فراهم می‌سازد.

اکنون روش کاراتری را برای پیاده‌سازی یک تابع بول n متغیر با مولتی‌پلکسری که n–1 ورودی انتخاب دارد، معرفی می‌نماییم. ابتدا n–1 متغیر به ورودی‌های انتخاب مولتی‌پلکسر وصل می‌شود. تنها متغیر باقی‌مانده تابع برای ورودی‌های داده مورد استفاده قرار می‌گیرد. اگر متغیر باقی‌مانده را z بنامیم هر ورودی داده مولتی‌پلکسر 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 شکل۷-۱۳ با یک مولتی‌پلکسر 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.



(ب) پیاده سازی مولتی‌پلکسر

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		1	F=1
1	1	1	1	

(الف) جدول درستی

شکل ۷-۱۳: پیاده‌سازی یک تابع بول با یک مولتی‌پلکسر

مقادیر خطوط ورودی 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 وروده داده 0 متصل شود. عملکرد مولتی‌پلکسر 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 بترتیب به F وصل می‌شوند. این مثال خاص هر چهار حالت ممکن را برای ورودی‌های داده تهیه می‌کند.

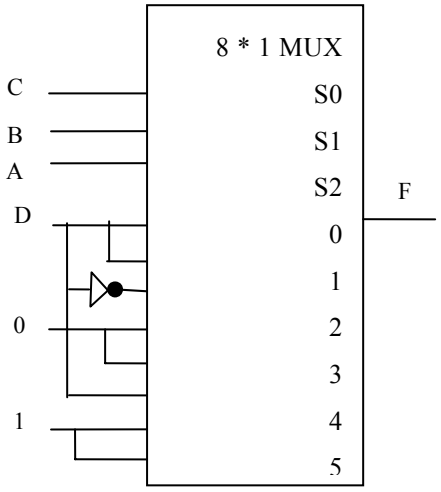
روال کلی برای پیاده‌سازی هر تابع بول n متغیره با یک مولتی‌پلکسر که n–1 خط ورودی انتخاب و 2^{n-1} ورودی داده دارد از مثال قبل نتیجه می‌گردد. ابتدا تابع بول را در جدول درستی لیست می‌کنیم. اولین n–1 متغیر در جدول ورودی‌های انتخاب مولتی‌پلکسر وصل می‌شوند. برای هر ترکیبی از متغیرهای انتخاب، خروجی را به عنوان 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 مولتی‌پلکسری که سه ورودی انتخاب دارد، مطابق شکل۷-۱٤ 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 وS_0 وصل شوند. مقادیر ورودی 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.

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	

(الف) جدول درستی



(ب) پیاده سازی مولتی‌پلکسر

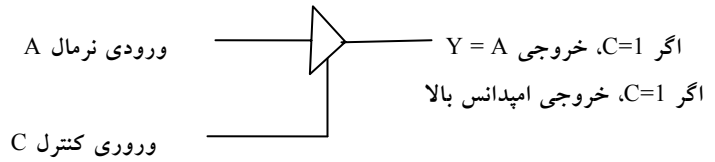
شکل ۷-۱٤: پیاده‌سازی یک تابع 4 ورودی با یک مولتی‌پلکسر

۷-۳-۲ گیت‌های سه حالته

یک مولتی‌پلکسر را می‌توان با گیت‌های سه حالته ساخت. یک گیت سه حالته مداری دیجیتالی 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 بالا است.

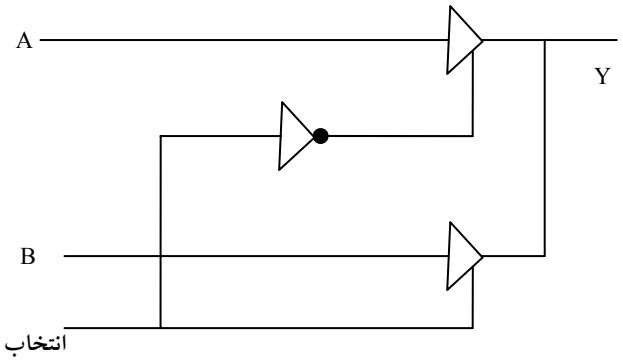
حالت امپدانس بالا مثل مدار باز عمل می‌کند و به این معنی است که خروجی از درون قطع بوده و مدار دارای مفهوم منطقی با ارزشی نیست. گیت‌های سه حالته ممکن است به عنوان گیت‌های AND و NAND نیز عمل کنند. با این وجود اغلب به عنوان بافر مورد استفاده قرار می‌گیرند.

نمودار گرافیکی یک گیت بافر سه حالته در شکل ۷-۱۵ 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 غیر فعال شده و گیت بدون توجه به مقدار 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 تاثیر بر روی بار شدن، یک خط مشترکی را تشکیل دهند.



شکل ۷-۱۵: سمبل گرافیکی برای یک بافر سه حالته

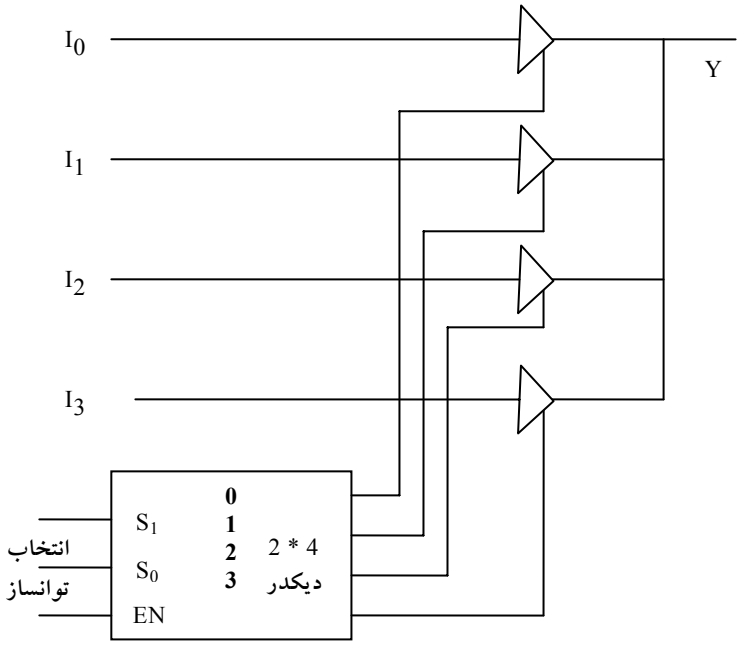
ساخت مولتی‌پلکسرها با بافرهای سه حالته در شکل ۷-۱۶ دیده می‌شود. این شکل یک مولتی‌پلکسر 2 به 1 را با دو بافر سه حالته و یک وارون‌گر نشان می‌دهد. دو خروجی به هم وصل شده‌اند تا یک خروجی مشترک را به وجود آورند.



شکل ۷-۱۶: مدار یک مولتی‌پلکسر ۲ به ۱ با بافر سه حالته

کنترل به بافر مشخص می‌کنند که کدام یک از چهار ورودی نرمال I_0 تاI_3 به خط

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



شکل۷-۱۷: مولتی‌پلکسرهای ٤ به ۱ با گیت‌های سه حالته

فعال می‌گردد و در این حال بافر پایینی غیر فعال است.

ساختار یک مولتی‌پلکسر 4 به 1 در شکل۷-۱۷ ملاحظه می‌شود. خروجی‌های چهار بافر سه حالته به هم متصل شده‌اند تا یک خروجی مشترک را بسازند. ورودی‌های 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 اینکه همه دیگر بافرها در حالت امپدانس بالا قرار خواهند گرفت. برای اطمینان از اینکه تنها یک ورودی 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 وخط گذرگاه در حالت امپدانس بالاست زیرا هر چهار بافر غیر فعال‌اند. 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 مولتی‌پلکسر 4 به 1 است.

۷-٤ زبان HDL برای مدارهای ترکیبی

در این بخش، روش دیگری را برای توصیف مدارهای ترکیبی HDL نشان خواهیم داد. همانطور که قبلاً ذکر شد، ماژول، یک بلوک ساختاری پایه درVerilog HDL است. ماژول می‌تواند در هر یک از تکنیک‌های مدل‌سازی زیر توصیف گردد.

مدل‌سازی سطح گیت با ذکر گیت‌های اصلی (Primitive) و ماژولهای تعریف شده به وسیله کاربر.

مدل‌سازی روند داده با به‌کارگیری عبارات تخصیص مداوم (پیوسته) که با کلمه کلیدی assign انجام می‌شوند.

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 اینکه 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 بجز روش‌های فوق وجود دارد که به آن مدل‌سازی سطح سوئیچ گویند. این نوع مدل‌سازی قابلیت طراحی را در سطح ترانزیستور MOS فراهم می‌سازد.

۷-٤-۱ مدل‌سازی سطح گیت

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 ذیل آمده‌اند. این گیت‌ها با کلمات کلید حروف کوچک زیر معرفی می‌شوند که عبارتند از: and، nand، or، xor، xnor، not، buf.

and	۰	1	x	z
0	۰	0	0	0
1	۰	1	x	x
x	۰	x	x	x
z	۰	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

شکل ۷-۱۸: جدول درستی برای گیت‌های اصلی پیش تعریف شده

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 برایand ، xor،or و not در جدول شکل۷-۱۸ دیده می‌شود.

جدول درستی چهار گیت دیگر مشابه است با این تفاوت که خروجی‌ها متمم شده‌اند. توجه کنید که برای گیتand ، 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 خروجی آن 0 است و اگر هر یک از ورودی‌ها 1 باشد خروجی هم 1 است، در غیر اینصورت x است.

وقتی که یک گیت اصلی در یک ماژول لحاظ شود گوییم در ماژول 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 ماژول سطح بالاتر ایجاد می‌نمایند. بنابراین ماژولی 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 می‌شوند، 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 3:0 [ ; ] D
wire [ 0:7 ; 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 شماره گذاری شده‌اند را نشان می‌دهد. اولین عدد لیست شده با ارزش‌ترین بیت بردار است. بیت‌های خاص در داخل کروشه ذکر می‌گردند، بنابراین [2] D، بیت 2 از D را نشان می‌دهد. همچنین ممکن است بخش‌هایی از یک بردار را ذکر کند. مثلاً [0: 2] SUM سه بیت کم ارزش‌تر بردار SUM را بیان می‌کند.

۷-٤-۲ گیت‌های سه حالته

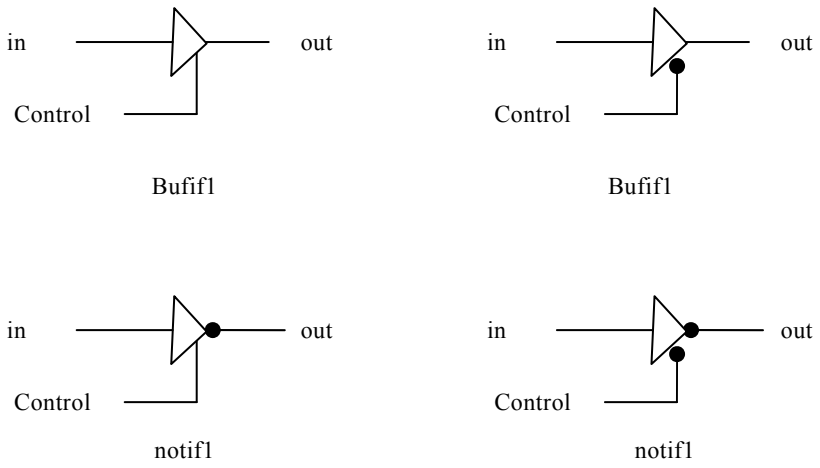
هر گیت سه حالته دارای یک ورودی کنترل است که می‌تواند آن را به حالت امپدانس بالا ببرد. این حالت در HDL با سمبل z مشخص می‌شود. چهار نوع گیت سه حالته طبق شکل۷-۱۹ وجود دارد. گیت bufif1 مانند یک بافر معمولی عمل می‌کند به شرطی که control=1 باشد. وقتی control=0 است، خروجی به حالت امپدانس بالای z می‌رود. گیت 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:

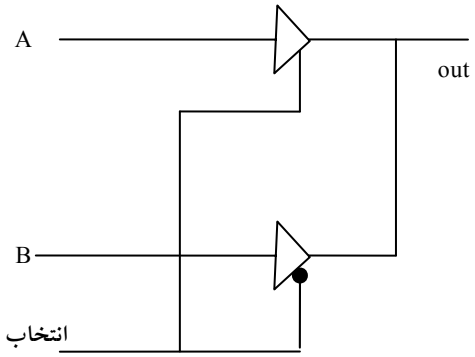
OUT, A , control (bufif1);
notifo (Y,B,enable) ;

در مثال اولی، ورودی A هنگامی‌که control=1 است به خروجی منتقل می‌گردد وقتی control=0 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.



شکل۷-۱۹: گیت سه حالته

خروجی گیت‌های سه حالته می‌توانند برای تشکیل یک خروجی مشترک به هم وصل شوند. برای شناخت چنین اتصالی HDL از کلمه کلید tri (برای state tri) استفاده می‌کند تا بیان نماید که خروجی دارای قابلیت راه‌اندازی چندگانه است.



شکل ۷-۲۰: مولتی‌پلکسر 2 به 1 با بافر های سه حالته

مثلاً مولتی‌پلکسر 2 به 1 را در شکل۷-۲۰ با گیت‌های سه حالته در نظر بگیرید. توصیف HDL 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
```

دو بافر سه حالته دارای خروجی مشترک‌اند. برای اینکه 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 وسیله ای 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 مکرر به کار برده می‌شود. مدل‌های wor پیاده‌سازی سخت‌افزاری آرایش wired–OR را نشان می‌دهند. مدل wand هم برای آرایش wired– AND در نظر گرفته شده‌اند.شبکه‌های supply1 و supply0 منبع تغذیه و زمین هستند. از آنها در توصیف سطح- سوئیچ استفاده می‌شود.

۷-٤-۳ مدل‌سازی روند داده

مدل‌سازی روند داده از تعدادی عملگر روی عملوندها استفاده می‌کند تا خروجی مورد نظر را تولید کند. Verilog HDL حدود 30 عملگر در اختیار می‌گذارد. جدول شکل۷-۲۱ بخشی از این عملگرها، سمبل‌های آنها و عملی که اجرا می‌کنند را نشان می‌دهند.

لازم است تا بین اعمال حسابی و منطقی تفکیک به عمل آید، بنابراین برای هر یک، از سمبل جداگانه ای 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

مکانیزمی‌است 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.

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

شکل ۷-۲۱: عملگرهای Verilog 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);
End module
```

مثال ۱: 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 می‌روند. 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 مولتی‌پلکسر 2 به 1 با ورودی‌های داده A وB ، ورودی انتخاب S و خروجیY با عبارت مداوم زیر تعریف می‌گردد:

assign Y = (A & S) | (B & ~ S) ;

عبارت با کلمه کلیدی assign و به دنبال آن خروجی مورد نظر 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 مدل های روند داده دو مثال سطح گیت قبلی را نشان می‌دهند. توصیف روند داده یک دیکدر 2 به 4 در مثال ۱ 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 جمع‌کننده 4 بیت در مثال زیر HDL آورده شده است.

مثال ۲: در این مثال توصیف روند داده یک جمع‌کننده 4 بیت ارائه 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 مورد نظر ادغام نقلی خروجی Cout و چهار بیت SUM است.

```
// dataflow  description  of  4 – bit  adder
Module  binary  -adder (A,B,cin,SUM,COUT);
Input  [0:3]  A,B;
Input  cin ;
Output  [0:3]  SUM;
Output  COUT ;
 Assign   { COUT ,SUM }  = A +B +cin ;
End module
```

ادغام عملوند ها در پرانتز بیان شده و یک ویرگول آنها را از هم جدا می‌سازد. بنابراین پنج بیت { SUM و Cout }، نتیجه عمل جمع را به نمایش می‌گذارد.

مثال ۳: مدل‌سازی روند داده امکان توصیف مدارهای ترکیبی را با تابع به جای ساختار گیتی اش فراهم می‌سازد برای ملاحظه چگونگی انجام طراحی دیجیتال با روند داده، مقایسه‌گر مقدار چهار بیتی توصیف شده در این مثال را در نظر بگیرید. مدل دو گروه ورودی چهار بیت A وB 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 اینکه اگر A برابر با B است خروجی (AEQB) وجود داشته باشد. توجه کنید که تساوی با دو علامت مساوی تعریف می‌شود.

```
// dataflow  description  of  4 – bit  comparator
Module  magcomp  (A,B,ALSB,AGTB,AEQB);
Input  [0:3]  A,B;
Output  ALSB,AGTB,AEQB ;
 Assign     ALSB = (A<B) ,
AGTB  = (A>B),
AEQB = (A=  =B);
End module
```

مثال بعدی از عملگر شرطی (:?) استفاده می‌نماید. این عملگر سه عملوند را اختیار می‌کند.

مثال ۴: مولتی‌پلکسر 2 به 1 خط را با عملگر شرطی توصیف می‌نماید.

Condition ? true – expression: false – expression

شرط همواره ارزیابی می‌شود. اگر نتیجه منطق 1 بود true – expression ارزیابی می‌گردد. اگر نتیجه منطق 0 بود،false – expression ارزیابی خواهد شد. این معادل با یک شرط if– else است. تخصیص مداوم

assign OUT = select ? A: B ;

شرط زیر را بیان می‌کند.

OUT = A if select = 1 , else OUT = B if select = 0

```
// dataflow description of 2- to – 1 – line multiplexer
Module mu×2×1– df (A,B,select,OUT);
Input A,B,select;
Output OUT ;
Assign OUT = select? A : B ;
End module
```

۷-٤-٤ مدل‌سازی رفتاری

مدل‌سازی رفتاری مدارهای دیجیتال را در سطح الگوریتمی‌و 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.

توصیف‌های رفتاری از کلمه کلیدی always و بدنبال آن لیستی از عبارات تخصیص اجرایی (رویه‌ای) استفاده می‌کنند. خروجی مورد نظر این عبارات باید نوع داده reg باشد. بر خلاف داده نوع wire که خروجی مورد نظر یک تخصیص ممکن است مرتبا به روز شود، داده نوع reg مقدارش را تا تخصیص مقدار جدید حفظ می‌کند.

مثال ۵: مثال ۵ HDL توصیف رفتاری مولتی‌پلکسر 2 به 1 را نشان می‌دهد (آن را با مثال ٤ HDL مقایسه کنید). چون OUT یک خروجی مورد نظر یا مقصد است، باید علاوه بر اعلان output، به صورت داده reg هم اعلام گردد.عبارات تخصیص اجرایی در داخل بلوکalways 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 تصمیمی‌را که مبتنی بر مقدار ورودی select است 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.

```
// Behavior  description  of  2- to – 1 – line  multiplexer
Module  mu×2×1– 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  ;
End module
```

مثال ۶: یک مولتی‌پلکسر 4 به 1 را توصیف نمایید.

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 مقایر لیست عباراتی که به دنبالش می‌آیند ارزیابی و مقایسه می‌شود. اولین مقداری که با شرط صحیح تطبیق کند اجرا می‌گردد. چون select یک عدد دو بیتی 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 شده و سپس مقدار آن ذکر می‌شود. بنابراین 2′b01 به معنی عدد دودویی دو رقمی‌است 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، پیش فرض دهدهی خواهد بود. اگر سایز عدد نامشخص باشد، سیستم سایز را 32 بیت فرض خواهد کرد.

```
// Behavioral  description  of  4- to – 1 – line  multiplexer
// Describes the function table of fig. 4-25  (b)
Module  mu×4×1– bh  (i0,i1,i2,i3,select,y);
Input  i0,i1,i2,i3 ;
```

```
Input [1:0] select;
Output y ;
Reg y ;
Always @ (i0or i1or i2or i3or select )
Case ( select)
2 ' b 00 : y = i0 ;
2 ' b 01 : y = i1 ;
2 ' b 10 : y = i2 ;
2 ' b 11 : y = i3 ;
End case
End module
```

در اینجا مثال‌های ساده‌ای را از توصیف رفتاری مدارهای ترکیبی نشان دادیم. مدل‌سازی رفتاری و عبارات تخصیص اجرایی دانش مدارهای ترتیبی را لازم دارد که در فصل بعد مورد بحث و بررسی قرار خواهد گرفت.

۷-٤-۵ 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 شبیه ساز 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 ماژول‌های محرک HDL ارائه شده‌اند.

علاوه بر عبارات always، برنامه تست از عبارت initial برای تهیه محرک به مدار تحت تست استفاده می‌کند. عبارت always حلقه ای را به‌صورت تکراری اجرا می‌کند. عبارت initial فقط یک بار با شروع از t=0 شبیه‌سازی را انجام داده و ممکن است هر عملی را با تاخیری 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 گرفته اند. 10 واحد زمان بعد، A به 1 تغییر یافته است، پس از 20 واحد زمانی، (در t = 30) 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 ' b 0 0 0 ;
repeat (7)
#10 D = D + 3 ' b 0 0 0 ;
end
```

برای 3 بیت D در t = 0 با 000 مقدار دهی 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 ماژول محرک در یک برنامه HDL ساختار زیر را دارد.

- نام module تست
- اعلان شناسه‌های reg و wire محلی
- ذکر ماژول طراحی مورد تست
- ایجاد عبارات محرک initial وalways
- نمایش پاسخ خروجی
- endmodule

یک ماژول تست معمولاً ورودی یا خروجی ندارد. سیگنال‌هایی که به عنوان ورودی به ماژول طراحی برای شبیه‌سازی اعمال می‌شوند معمولاً در ماژول محرک به عنوان نوع داده یا reg محلی اعلان می‌گردند. خروجی‌های ماژول طراحی که برای تست نمایش داده می‌شوند در ماژول محرک به عنوان داده نوع wire اعلان می‌شوند. آنگاه ماژول تحت تست با به‌کار گیری شناسنامه‌های محلی ذکر می‌گردد. شکل۷-۲۲ این ارتباط را نشان می‌دهد.

ماژول محرک		ماژول طراحی
Module testcircuit		Module circuit(A,B,C);
Reg TA , TB;	→	Input A,B;
Wire TC;	←	Output C;
Circuit cr(TA,TB,TC);		

شکل ۷-۲۲: ماژول های محرک و طراحی محاوره‌ای

ماژول محرک ورودی‌ها را برای ماژول طراحی با اعلان شناسه‌های TA و TB به عنوان نوعی reg تولید می‌کند و خروجی طرح را با شناسه نوع wire، یعنی TC چک می‌نماید. سپس شناسه‌های محلی برای ذکر ماژول زیر تست به کار می‌روند. پاسخ به محرک تولید شده با بلوک‌های intial وalways در خروجی محرک به صورت نمودار زمان بندی ظاهر می‌شود. و نیز می‌توان با استفاده از Verilog system tasks خروجی عددی نیز تولید 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$	**یکبار 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 وB 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

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 قالب0d% را به جای d% به کار ببریم. این کار رقم‌های با ارزش‌تر را بدون فضای خالی تولید می‌نماید (d% حدود 10 فضای خالی را تولید می‌کند زیرا زمان با عدد 32 بیت تولید می‌گردد). مثالی از ماژول محرک در مثال ۷ HDL نشان داده شده است.

مثال ۷: این مثال حالت ماژول محرک را نشان می‌دهد. مدار مورد تست یک مولتی‌پلکسر 1×2 است که در مثال ٦ توصیف گردید ماژول testmux پورت ندارد.

```
// stimulus for   mu×2×1– df .
Module   testmux  ;
Reg TA, TB, TS ; //inputs  for mux
Wire  y ; //output from mux
  mu×2×1– 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 = %bA = %bB=%b OUT= %b Time = % 0d",
TS ,TA, TB, Y ,$ time );
End module
// dataflow   description   of    2- to – 1 – line  multiplexer
// from example 4- 6
Module  mu×2×1– df  (A,B,select,OUT);
Input A, B, select ;
Output    OUT ;
Assign OUT = select? A  :  B  ;
End module
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
```

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 تکلیفmonitor $

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. می‌بینیم که وقتی S = 1 باشد OUT =A و وقتی S=0 باشد OUT = B است، که بدین ترتیب عملکرد مولتی‌پلکسر را تایید می‌کند.

```
// 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, F2 not);
Or  g9  (F1 , T2, T3);
End module
// stimulus  to analyze  the  circuit
Module  test- circuit ;
Reg [2:0]D;
Wire  F1 , F2 ;
Analysis   fig 42  ( 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);
End module
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
```

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 تاخیر در گیت‌ها مطالعه می‌کنیم. این کار با مشاهده امواج در خروجی گیت‌ها وقتی به یک ورودی مفروض پاسخ می‌دهند، صورت می‌گیرد.

سؤالات

۱- نمودار منطقی یک دیکدر ۲ به ٤ را فقط با گیت‌های NOR طراحی نمایید.

۲- یک مدار ترکیبی با سه تابع بولی زیر تعریف شده است. مدار را با دیکدر و گیت‌های بیرونی بسازید.

$F_1 = x'y' + xyz'$

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

$F_3 = x'y'z + xz$

۳- تابع بولی زیر را با استفاده از یک مولتی‌پلکسر پیاده‌سازی کنید.

$F(x,y,z,w) = \sum(0,2,5,7,12,14)$

٤- یک مولتی‌پلکسر 1 * 16 را با دو مولتی‌پلکسر 1 * 8 پیاده سازی نمایید.

۵- جدول درستی یک انکدر اولویت هشت هشتی به دودویی را معین کنید.

٦- یک جمع‌کننده کامل را با دو مولتی‌پلکسر 1 * 4 پیاده سازی نمایید.

۷- یک دیکدر افزونی 3 به دودویی را با استفاده از ترکیبات به‌کار نرفته بی‌اهمیت طراحی نمایید.

فصل ۸

مدارهای ترتیبی همزمان

هدف کلی

در این فصل مباحث اصلی مربوط به مدارهای ترتیبی بالاخص مدارهای ترتیبی همزمان مورد بحث و بررسی قرار خواهند گرفت. مفاهیم فلیپ‌فلاپ‌ها به همراه مدارها و نحوه کار هر یک از آنها توضیح داده خواهد شد. همچنین در بحث مدارهای ترتیبی ساعت‌دار، روش‌های تحلیل معادلات ورودی با فلیپ‌فلاپ‌ها مورد بحث و بررسی قرار خواهند گرفت.

هدف ساختاری

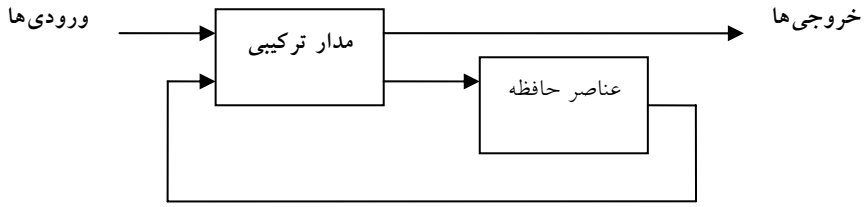
در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- مفهوم مدارهای ترتیبی
- مفاهیم فلیپ‌فلاپ‌ها و لچ‌ها
- فلیپ‌فلاپ T
- فلیپ‌فلاپ D
- فلیپ‌فلاپ K
- مدارهای ترتیبی ساعت‌دار
- تحلیل معادلات با کمک فلیپ‌فلاپ‌ها

۸-۱ مدارهای ترتیبی

کلیه مدارهای دیجیتالی که در فصول ششم و هفتم مورد بررسی قرار گرفته بودند از نوع مدارهای ترکیبی بودند. در این مدارها خروجی‌ها همه به ورودی‌های دیجیتال 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 بر حسب منطق ترتیبی مورد بررسی قرار گیرند. همچنین لازم است در مواردی در فرایند طراحی گیت عمدا تاخیراتی اعمال گردد. در این فصل مدارهای ترتیبی که قادر به پیاده‌سازی این نوع نیازها هستند، مورد بحث و بررسی قرار می‌گیرند.

نمودار بلوکی یک مدار ترتیبی در شکل۸-۱ 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.



شکل ۸-۱: نمودار بلوکی یک مدار ترتیبی

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 به

حافظه نیز تابعی از ورودی‌های بیرونی و حالت فعلی است. بنابراین یک مدار ترتیبی با ترتیب زمانی ورودی‌ها، خروجی‌ها و حالات داخلی مشخص می‌گردد.

۸-۱-۱ انواع مدارهای ترتیبی

دو نوع مدار ترتیبی وجود دارد که دسته‌بندی آنها به زمان‌بندی سیگنال آنها وابسته است. این نوع مدارها عبارتند از:

- مدارهای ترتیبی همزمان
- مدارهای ترتیبی غیر همزمان

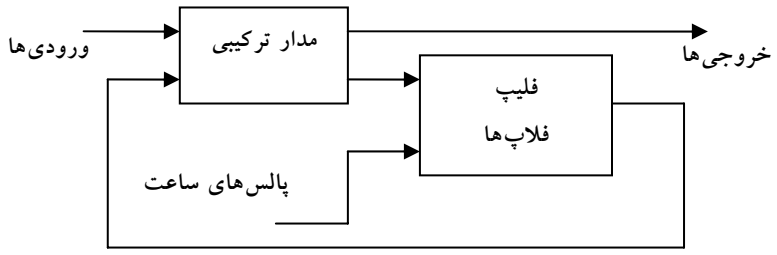
مدار ترتیبی همزمان یا همگام سیستمی است که رفتار آن با توجه به دانش و آگاهی از سیگنال‌هایش در هر لحظه گسسته‌ای از زمان قابل تعریف می‌باشد. رفتار یک مدار ترتیبی غیر همزمان به ترتیب تغییر سیگنال‌های ورودی آن که می‌توانند در هر لحظه از زمان روی مدار تاثیر کنند وابسته می‌باشد. عناصر حافظه‌ای که به طور معمول در مدارهای ترتیبی غیر همزمان به‌کار می‌روند، نوعی وسایل تاخیر زمانی هستند. قابلیت نگهداری یک وسیله تاخیر زمانی به زمان انتشار سیگنال در وسیله بستگی دارد. در عمل تاخیر انتشار در گیت‌های منطقی درونی برای ایجاد تاخیر کفایت می‌کند بنابراین واحد تاخیر واقعی می‌توانند مورد نیاز نباشد. در سیستم‌های غیر همزمان نوع گیتی، عناصر حافظه متشکل از گیت‌های منطقی است که در واقع تاخیر انتشار آنها عمل ذخیره‌سازی را تداعی می‌نماید. بنابراین در چنین مواقعی یک مدار ترتیبی غیر همزمان را می‌توان مداری با پسخورد دانست. به دلیل وجود پسخورد در بین گیت‌های منطقی، هر مدار ترتیبی غیر همزمان هر لحظه ممکن است ناپایدار شود. مسئله بی‌ثباتی حاکم مشکلات عدیده‌ای را برای طراح تحمیل خواهد کرد.

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

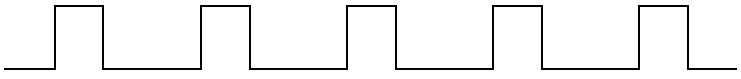
متناوبی از پالس ساعت به وسیله این دستگاه تولید می‌گردد. پالس‌های ساعت در سرتاسر سیستم توزیع می‌گردند به نحوی که عناصر حافظه تنها هنگام رسیدن هر پالس تحت تاثیر ورودی خود قرار می‌گیرند. در عمل پالس‌های ساعت به همراه دیگر پالس‌ها که تغییرات لازم را در حافظه ایجاد می‌کند همراه هستند. مدارهای ترتیبی همزمانی که پالس‌های ساعت را در ورودی عناصر ذخیره ساز 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.

۸-۲ فلیپ‌فلاپ‌ها و لچ‌ها

یکی از نکات مهم در مدارهای ترتیبی بحث ذخیره‌سازی اطلاعات در هنگام اجرای ترتیبی گیت‌های مدار است. عناصر ذخیره‌سازی در مدارهای ترتیبی ساعت‌دار را



(لف) نمودار بلکی



(ب) نمودار زمان بندی پالس های ساعت

شکل ۸-۲: مدار ترتیبی ساعت‌دار همزمان

فلیپ‌فلاپ می‌گویند. فلیپ‌فلاپ 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 فلیپ‌فلاپ‌ها را به‌کار ببرد. نمودار بلوکی یک مدار ترتیبی ساعت‌دار همزمان در شکل ۸-۲ دیده می‌شود. خروجی‌ها می‌توانند از یک مدار ترکیبی، یا از فلیپ‌فلاپ‌ها و یا هر دو حاصل شوند. فلیپ‌فلاپ‌ها ورودی‌های خود را از مدار ترکیبی و نیز از سیگنال ساعت که با فواصل زمانی رخ می‌دهند، طبق نمودار زمانی دریافت می‌کنند.

حالت فلیپ‌فلاپ‌ها 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 فلیپ‌فلاپ تغییر نمی‌نمایند. بنابراین تغییر وضعیت از یک حالت به بعدی فقط در فواصل زمانی دیکته شده به وسیله پالس‌های ساعت امکان‌پذیر است.

۸-۲-۱ لچ‌ها

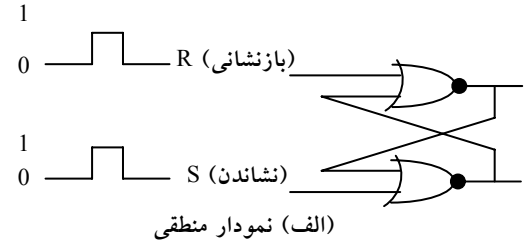
یک فلیپ‌فلاپ می‌تواند یک حالت دودویی را مادامی که تغذیه به مدارش اعمال شود، تا مدتی نامحدود نگهدارد. تفاوت عمده بین انواع فلیپ‌فلاپ‌ها، در تعداد ورودی‌ها و نحوه تاثیر آنها در تغییر حالت دودویی است. ساده‌ترین انواع فلیپ‌فلاپ‌ها که با سطوح سیگنال عمل می‌کنند، لچ نامیده می‌شوند. لچ‌ها (یا نگهدارها) مدارهای مبنایی هستند که فلیپ‌فلاپ‌ها با آنها ساخته می‌شوند. گرچه لچ‌ها برای ذخیره اطلاعات دودویی و طراحی مدارهای ترتیبی غیر همزمان مفیدند، ولی عملاً در مدارهای ترتیبی همزمان به کار نمی‌روند. انواع فلیپ‌فلاپ‌هایی که در مدارهای ترتیبی مورد استفاده قرار می‌گیرند در بخش بعد معرفی شده‌اند.

۸-۲-۱-۱ 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 که با S به معنی نشاندن (set) و R برای بازنشانی (Reset) نام‌گذاری شده‌اند. لچ SR ساخته شده از دو گیت NOR در شکل۸-۳ دیده می‌شود. لچ دارای دو حالت مفید است. وقتی 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). 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.

S	R	Q	Q´
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

(ب) جدول تابع



(الف) نمودار منطقی

شکل ۸-۳: لچ SR با گیت NOR

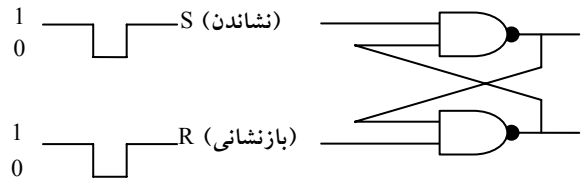
تحت شرایط معمولی، هر دو ورودی در 0 نگهداری می‌شوند مگر اینکه بخواهیم حالت لچ را عوض کنیم. اعمال یک لحظه 1 به ورودی S موجب می‌شود که لچ به حالت 1 برود. قبل از اینکه حالت تعریف نشده ای رخ دهد، ورودی S باید به 0 باز گردد. طبق جدول تابع در شکل ۸-۳(ب)، دو حالت از ورودی موجب می‌شود تا مدار در حالت 1 قرار گیرد. اولین حالت (R=0، S=1) نقشی است که طی آن ورودی S، مدار را به حالت 1 می‌برد،که 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 حالت0 باقی خواهد ماند. لذا وقتی هر دو ورودی S و R برابر 0 اند، بسته به اینکه 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 پیش بینی 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 ۸-٤ مشاهده می‌شود. این مدار به طور معمول با 1 در هر دو ورودی‌اش کار می‌کند مگر اینکه بخواهیم حالت لچ را تغییر دهیم. اعمال 0 به S موجب می‌شود Q به 1 برود، و لچ را به حالت نشانده وادارد. وقتی که ورودی S به 1 باز گردد ، مدار در همان حالت 1 باقی می‌ماند. پس از بازگشت هر دو ورودی به 1، ما مجاز به تغییر حالت لچ با استقرار0 در R هستیم. این موجب می‌شود تا مدار به حالت باز نشانی برود و حتی پس از بازگشت هر دو ورودی به 1، لچ در همان حال بماند. حالتی که برای لچ NAND غیر مجاز است، هنگامی‌است که هر دو ورودی به طور همزمان در 0 باشند. بنابراین از وقوع این حالت باید ممانعت کرد.

S	R	Q	Q´
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

(ب) جدول تابع



(الف) نمودار منطقی

شکل۸-٤: لچ SR با گیت‌های NAND

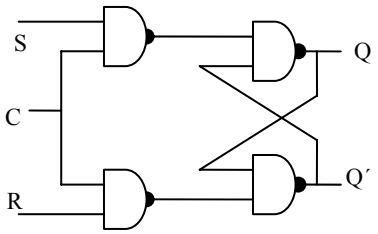
با مقایسه لچ NOR با NAND مشاهده می‌شود که سیگنال‌های ورودی برای NAND متمم ورودی‌های لچ NOR است. چون لچ NAND برای تغییر حالت خود به سیگنال 0 نیاز دارد، گاهی آن را لچ S´-R´ می‌خوانند. علامت پریم یا خط بار بر روی حروف،

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 کنترل دار در شکل ۸-۵ 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.

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



(الف) نمودار منطقی

شکل ۸-۵: لچ SR با ورودی کنترل

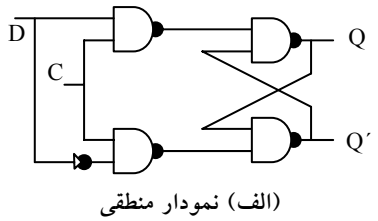
مادامی‌که ورودی کنترل در 0 باقی بماند، خروجی گیت‌های NAND در سطح منطقی1 باقی می‌مانند. این وضعیت حالت سکون برای لچ SR است. حالت نشاندن باS=1، R = 0، C = 1 حاصل می‌گردد. برای تغییر وضعیت باید S = 0، R = 1 و C = 1 باشد. در هر حال، وقتی که C به 0 بازگردد، مدار در حالت جاری باقی می‌ماند. در ورودی کنترل با اعمال 0 به C، مدار غیر فعال 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.

در بعضی شرایط حالت لچ را نا معین می‌نامند. حالت نامعین هنگامی‌رخ می‌دهد که هر سه ورودی برابر1 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 باز می‌گردد، 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 فلیپ‌فلاپ‌ها با آن ساخته می‌شوند.

۸-۲-۱-۲ 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 ۸-٦ میسر است. 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. مادامی‌که ورودی کنترل در 0 قرار دارد، لچ SR متقاطع دارای 1 در هر دو ورودی بوده و مدار نمی‌تواند تغییر حالت دهد. در واقع مقدار D هم نقشی ندارد. وقتی C=1 باشد ورودی D نمونه برداری 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.



(الف) نمودار منطقی

C	D	حالت بعدی Q
0	X	بلاتغییر
1	0	حالت بازنشانی Q=0
1	1	حالت نشاندن

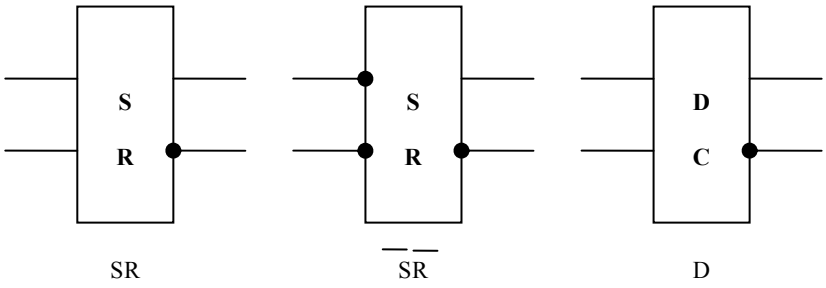
(ب) جدول درستی

شکل۸-٦: لچ D

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 هنگامی‌که ورودی کنترل فعال شود، به خروجی

Q منتقل می‌گردد. مادامی‌که 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 غیر فعال شود، اطلاعات دودویی حاضر قبلی در ورودی، در خروجی Q باقی می‌ماند تا دوباره ورودی کنترل فعال گردد.

نماد گرافیک برای انواع لچ در شکل ۸-۷ آمده است. لچ با یک بلوک مستطیلی مشخص می‌شود، که در آن ورودی‌ها در سمت چپ و خروجی‌ها درسمت 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 ۸-۷: سمبل‌های گرافیکی لچ

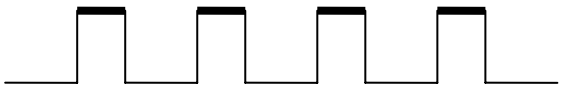
۸-۳ مکانیزم تغییر حالت لچ‌ها

حالت یک لچ یا یک فلیپ‌فلاپ با تغییر در ورودی کنترل عوض می‌شود. این تغییر لحظه‌ای را تریگر گویند و انتقال مربوط به آن را تریگر کردن فلیپ‌فلاپ خوانند. لچ D با پالس‌ها در ورودی کنترلش اساساً یک فلیپ‌فلاپ است که در هر زمان پالس به سطح منطقی 1 برود تریگر می‌شود. مادامی‌که پالس ورودی کنترل در این سطح بماند هر تغییری در ورودی داده، خروجی و حالت لچ را عوض خواهد کرد.

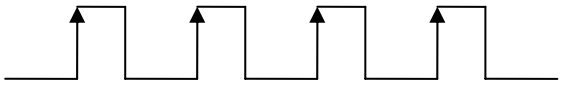
همانطور که از نمودار بلوکی شکل۸-۲ ملاحظه می‌شود، یک مدار ترتیبی از خروجی‌های فلیپ‌فلاپ به ورودی‌های مدار ترکیبی دارای مسیر پسخورد است. در نتیجه ورودی‌های فلیپ‌فلاپ ممکن است از خروجی همان یا دیگر فلیپ‌فلاپ‌ها راه‌اندازی شوند. وقتی که لچ‌ها به عنوان عناصر مورد استفاده قرار گیرند، مشکلی اساسی به وجود می‌آید. 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 پیش بینی نشده ای است زیرا حالت لچ‌ها ممکن است با قرار داشتن پالس ساعت در سطح فعال همچنان به تغییر خود ادامه دهد. به دلیل این عملکرد غیر مطلوب، 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.

فلیپ‌فلاپ‌ها طوری ساخته می‌شوند که وقتی بخشی از نوع مدار ترتیبی اند و از ساعت مشترکی استفاده می‌کنند، عملکردشان صحیح باشد. مشکل لچ این است که به سطح پالس ساعت پاسخ می‌دهد. همانطور که در شکل۸-۸ (الف) مشاهده می‌شود،

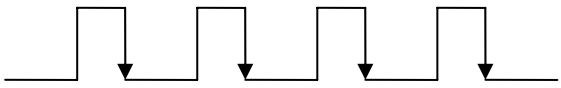
وقتی که پالس ساعت در منطق 1 قرار دارد، هر تغییر مثبت در ورودی کنترل موجب می‌شود تا به ازاء هر تغییر در ورودی D، تغییری در خروجی به وجود آید. نکته کلیدی در یک عملکرد صحیح فلیپ‌فلاپ‌ها تریگر شدن آنها در زمان گذر سیگنال است. پالس ساعت از دو انتقال 1 به 0 و 0 به 1 گذر می‌کند. طبق شکل۸–۸ گذر مثبت به عنوان لبه مثبت و گذر منفی به نام لبه منفی شناخته می‌شود. برای اصلاح یک لچ به یک فلیپ‌فلاپ، دو راه وجود دارد. یکی از این روش‌ها استفاده از دو لچ با آرایشی خاص است که خروجی فلیپ‌فلاپ را در حین تغییر ورودی، از آن جدا می‌سازد.راه دیگر تهیه فلیپ‌فلاپی است که فقط در حین گذر سیگنال تریگر می‌شود (از 0 به 1 یا از 1 به 0) و در بقیه لحظات پالس ساعت غیر فعال است. اکنون هر دو روش را مطالعه می‌کنیم.



(الف) پاسخ به سطح مثبت



(ب) پاسخ به لبه مثبت

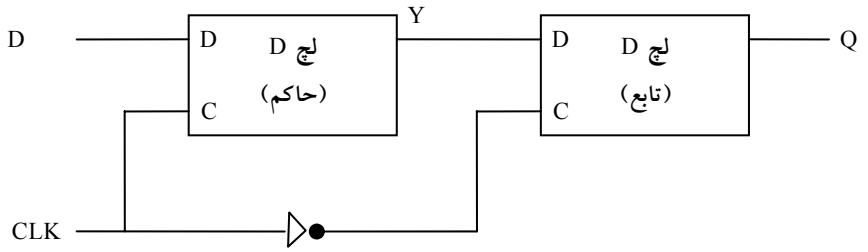


(پ) پاسخ به لبه منفی

شکل ۸–۸: پاسخ ساعت در لچ و فلیپ‌فلاپ

۸-۳-۱ فلیپ‌فلاپ D حساس به لبه

ساخت یک فلیپ‌فلاپ D با دو لچ D و یک وارون‌گر در شکل۸-۹ ملاحظه می‌گردد. اولین لچ را حاکم[1] و دومی‌را تابع[2] می‌گویند. مدار، ورودی D را نمونه برداری کرده و خروجی Q را فقط در لبه منفی پالس کنترل ساعت (CLK) تغییر می‌دهد.



شکل۸-۹: فلیپ‌فلاپ D حاکم – تابع

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 غیر فعال 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، مادامی‌که ساعت در سطح 1 بماند، تابع غیر فعال خواهد بود زیرا ورودی C آن برابر 0 است. هرتغییر در ورودی، خروجی Y را عوض می‌کند، ولی نمی‌تواند خروجی تابع را عوض کند. وقتی که پالس ساعت به 0 بازگردد، حاکم غیر فعال شده و از ورودی D جدا می‌شود. در همان زمان تابع فعال شده و مقدار Y به خروجی فلیپ‌فلاپ در Q انتقال

---

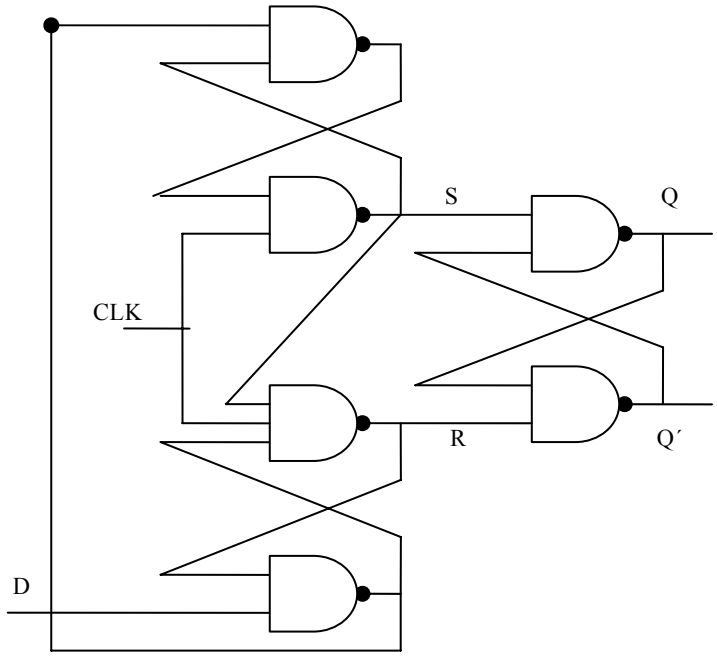
[1] _ Master

[2] _ Slave

می‌یابد. بنابراین خروجی فلیپ‌فلاپ فقط در حین گذر پالس ساعت از 1 به 0 تغییر می‌کند.

رفتار فلیپ‌فلاپ 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 فلیپ‌فلاپی 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 فلیپ‌فلاپ D حساس به لبه از سه لچ SR، مطابق شکل۸–۱۰ استفاده می‌کند. دو لچ موجود در این شکل به ورودی‌های بیرونی D (داده) و CLK (ساعت) پاسخ می‌دهد. لچ سوم خروجی را برای فلیپ‌فلاپ 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 فلیپ‌فلاپ 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 فلیپ‌فلاپ علیرغم تغییر در ورودی خود به حالت قفل باقی خواهد ماند. وقتی که ساعت به 0 باز گردد، 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، مادامی‌که CLK = 1 است، نمی‌تواند خروجی را تحت تاثیر قرار دهد.

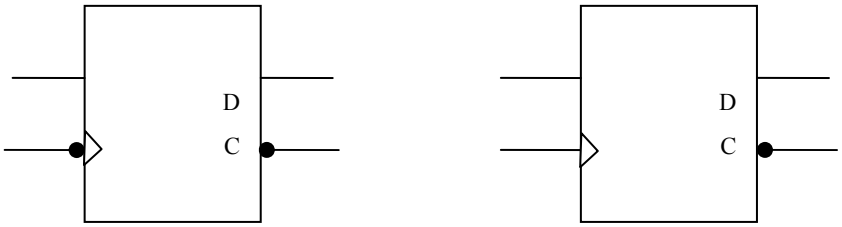


شکل۸-۱۰: فلیپ‌فلاپ D حساس به لبه مثبت

به طور خلاصه، وقتی ساعت ورودی در فلیپ‌فلاپ حساس به لبه مثبت یک انتقال مثبت انجام دهد، مقدار D به Q منتقل می‌شود. یک لبه منفی از 1 به 0 تاثیری 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 فلیپ‌فلاپ تنها به لبه 0 به 1 و لاغیر پاسخ می‌دهد.

هنگام استفاده از فلیپ‌فلاپ حساس به لبه باید زمان بندی پاسخ فلیپ‌فلاپ تحت بررسی قرار گیرد. در این زمان بندی، 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 بنام زمان نگهداری وجود دارد که طی آن ورودی D نباید پس از اعمال لبه مثبت ساعت، تغییر کند. تاخیر 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 فلیپ‌فلاپ D حساس به لبه در شکل۸-۱۱ مشاهده می‌شود. این سمبل مشابه با سمبل لچ D است به جزء اینکه در جلو حرف C علامت فلشی وجود دارد که دینامیکی بودن ورودی را نشان می‌دهد. نشانگر دینامیک به این معنی است که فلیپ‌فلاپ 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.



(ب) لبه منفی (الف) لبه مثبت

شکل ۸-۱۱: سمبل گرافیکی فلیپ‌فلاپ D حساس به لبه

۸-٤ فلیپ‌فلاپ‌های T و JK

به غیر از فلیپ‌فلاپ نوع D، فلیپ‌فلاپ‌های دیگری نیز وجود دارند. اقتصادی‌ترین و بهترین فلیپ‌فلاپ قابل ساخت، نوع D حساس به لبه می‌باشد که به تعداد کمتری گیت نیاز دارد. دیگر فلیپ‌فلاپ‌ها را می‌توان با فلیپ‌فلاپ D و مقداری مدار بیرونی به وجود آورد. دو فلیپ‌فلاپ رایج در طراحی سیستم‌های دیجیتال عبارتند از: فلیپ‌فلاپ JK و T.

با یک فلیپ‌فلاپ سه عمل را می‌توان انجام داد:

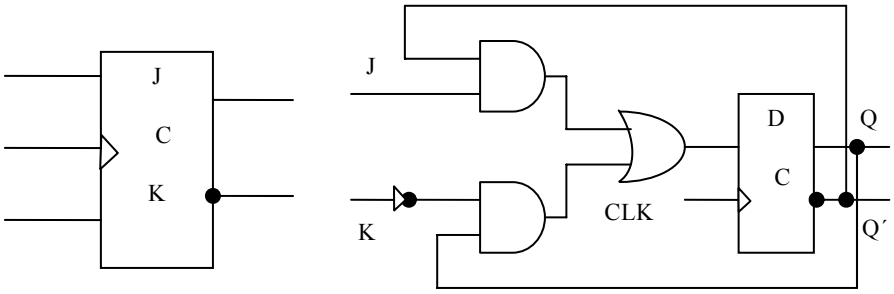
- نشاندن در 1
- بازنشانی در 0
- متمم شدن خروجی

۸-٤-۱ فلیپ‌فلاپ JK

فلیپ‌فلاپ JK هر سه کار را انجام می‌دهد. نمودار مدار 1 فلیپ‌فلاپ JK که از یک فلیپ‌فلاپ D ساخته شده است، در شکل۸-۱۲ (الف) دیده می‌شود. ورودی J، فلیپ‌فلاپ را در 1، ورودی K، آنرا 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 ورودی $J = K = 1$ باشد، $D = Q'$ است و بنابراین لبه ساعت



(ب) سمبل گرافیکی (الف) نمودار مدار

شکل ۸-۱۲: فلیپ‌فلاپ JK

بعدی خروجی را متمم می‌کند. هنگامی‌که $J = K = 0$ باشد، $D = Q$ است و لبه پالس ساعت بعدی خروجی را بدون تغییر رها خواهد کرد. سمبل گرافیکی برای فلیپ‌فلاپ JK در شکل ۸-۱۲(ب) ملاحظه می‌گردد. این سمبل مشابه فلیپ‌فلاپD است به جزء اینکه اکنون ورودی‌ها با J و K نام‌گذاری شده‌اند.

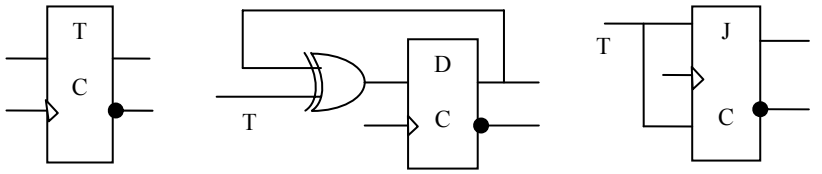
۸-٤-۲ فلیپ‌فلاپ T

فلیپ‌فلاپ T(دگر وضع)یک فلیپ‌فلاپ متمم ساز است و می‌توان آن را با گره زدن دو ورودی J و K ایجاد کرد. این عمل در شکل ۸-۱۳(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. فلیپ‌فلاپ متمم‌ساز در طراحی شمارنده‌های دودویی بسیار مورد توجه است.

یک فلیپ‌فلاپ T را می‌توان با یک فلیپ‌فلاپ D و یک گیت XOR مطابق شکل۸-۱۳ (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 فلیپ‌فلاپ دارای حرف T در ورودی است.



(پ) سمبل گرافیکی (ب) با فلیپ فلاپ D (الف) با فلیپ فلاپ JK

شکل ۸-۱۳: فلیپ‌فلاپ T

۸-٤-۳ جدول مشخصه فلیپ‌فلاپ‌ها

جدول مشخصه خواص منطقی یک فلیپ‌فلاپ را تعریف می‌کند و بدین ترتیب عملکرد آن به صورت جدول توصیف می‌گردد. جداول مشخصه سه نوع فلیپ‌فلاپ در جدول شکل۸-۱٤ 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.

فلیپ فلاپ JK

J	K	Q(T+1)	
0	0	Q(t)	بلا تغییر
0	1	0	بازنشانی
1	0	1	نشاندن
1	1	Q´(t)	متمم

فلیپ فلاپ D

D	Q(t+1)	
0	0	بازنشانی
1	1	نشاندن

فلیپ فلاپ T

T	Q(t+1)	
0	Q(t)	بلا تغییر
1	Q´(t)	متمم

شکل ۸-۱٤: جداول مشخصه فیلیپ فلاپ

جدول مشخصه فلیپ‌فلاپ 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 فلیپ‌فلاپ را به 0 بازنشانی می‌کند و بنابراین Q(t + 1)= 0 خواهد شد.

اگر J = 1 و K = 0 گردد فلیپ‌فلاپ 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 فلیپ‌فلاپ 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 فلیپ‌فلاپD حالت بی تغییر را دارا نیست. ولی این کار با غیر فعال کردن ساعت و یا با اتصال خروجی به ورودی D انجام می‌شود. طی آن خروجی یا حالت فلیپ‌فلاپ همواره بی تغییر خواهد ماند.

جدول درستی فلیپ‌فلاپ T فقط دو حالت دارد. وقتی T = 0 باشد، لبه ساعت حالت را تغییر نمی‌دهد. وقتی T = 1 باشد، لبه ساعت حالت فلیپ‌فلاپ را متمم می‌کند.

۸-٤-٤ معادلات مشخصه

خواص منطقی یک فلیپ‌فلاپ که در جدول مشخصه ملاحظه شد را می‌توان به صورت معادله مشخصه هم بیان کرد. برای فلیپ‌فلاپD، 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 فلیپ‌فلاپ JK را از جدول مشخصه و یا از مدار شکل۸-۱۲ می‌توان به‌دست آورد. یعنی $Q(t+1)= JK' + K'Q$ که Q مقدار خروجی فلیپ‌فلاپ قبل از اعمال یک پالس ساعت است. معادله مشخصه برای فلیپ‌فلاپ T از شکل۸-۱۳ حاصل می‌شود.

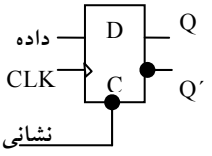
$$Q(t+1)= T\oplus Q = TQ'+ T'Q$$

۸-٤-۵ ورودی‌های سیستم

بعضی از فلیپ‌فلاپ‌ها دارای ورودی‌های غیر همزمان برای داشتن آن به یک حالت خاص مستقل از پالس ساعت می‌باشند. ورودی که فلیپ‌فلاپ را در 1 می‌نشاند، پیش تنظیم یا تنظیم مستقیم می‌نامند. ورودی که فلیپ‌فلاپ را به 0 پاک می‌کند، ورودی پاک یا بازنشان مستقیم(غیر همزمان)می‌خوانند. وقتی تغذیه در یک سیستم دیجیتال

روشن شود، حالت فلیپ‌فلاپ نامعلوم است. ورودی‌های مستقیم در استقرار همه فلیپ‌فلاپ‌های سیستم به یک حالت آغازین معلوم، قبل از اعمال پالس ساعت مفید هستند. یک فلیپ‌فلاپ D حساس به لبه مثبت با بازنشانی غیر همزمان R در شکل۸-۱۵ ملاحظه می‌شود. نمودار مدار مشابه شکل۸-۱۰ است با این تفاوت که یک ورودی بازنشانی اضافی، به سه گیت NAND متصل شده‌اند. وقتی که این ورودی در 0 است، Q′ را بماندن در 1 وامی‌دارد، و این به نوبه خود به معنی پاک شدن خروجی Q به 0 است و بنابراین فلیپ‌فلاپ 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.

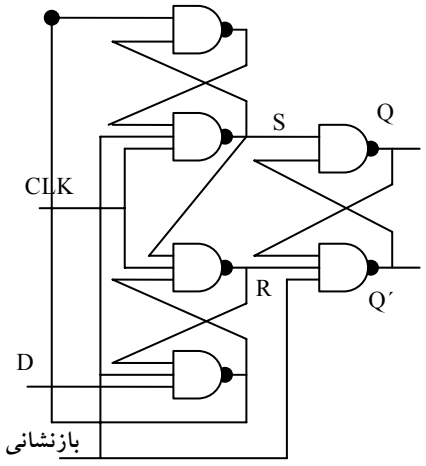
سمبل گرافیکی فلیپ‌فلاپ D با یک ورودی بازنشان مستقیم دارای یک ورودی اضافی است که با R علامت گذاری شده است. وجود حباب در ورودی به این معنی است که بازنشانی با سطح منطق 0 فعال می‌گردد. فلیپ‌فلاپ‌هایی که از نشاندن مستقیم استفاده می‌کنند از سمبل S در ورودی نشاندن غیر همزمان استفاده می‌کنند.



(ب) سمبل گرافیکی

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(پ) جدول درستی



(الف) نمودار مدار

شکل ۸-۱۵: فلیپ‌فلاپ D با بازنشانی غیر همزمان

جدول تابع، عملکرد مدار را مشخص می‌کند. وقتی R=0 باشد، خروجی به 0 بازنشانی می‌شود. این حالت مستقل از D و C است. هنگامی‌مدار می‌تواند به روند عادی خود بازگردد که ورودی بازنشانی به 1 برود. ساعت در C با یک فلش روبه بالا، که به معنی عملکرد فلیپ‌فلاپ 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.

۸-۵ تحلیل مدارهای ترتیبی ساعت دار

رفتار یک مدار ترتیبی ساعت‌دار با ورودی‌ها، خروجی‌ها و حالت فلیپ‌فلاپ‌ها 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 عبارت بول را نوشت و به وسیله آنها رفتار مدار را توصیف کرد. این عبارات باید رشته زمانی لازم را چه مستقیماً و چه غیر مستقیم مشخص کند.

یک نمودار منطقی، وقتی دارای فلیپ‌فلاپ‌ها با ورودی‌های ساعت باشد، مدار ترتیبی ساعت‌دار خوانده می‌شود. فلیپ‌فلاپ‌ها 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 بر حسب 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.

۸-۵-۱ معادلات حالت

رفتار مدار ترتیبی ساعت دار را می‌توان با معادلات حالت توصیف کرد. یک معادله حالت (که به آن معادله گذر هم می‌گویند) حالت بعدی را بر حسب تابعی از حالات فعلی و ورودی‌ها بیان می‌نماید. مدار شکل۸-۱۶ را ملاحظه نمایید. این مدار از دو فلیپ‌فلاپ A و B نوع D و یک ورودی x و یک خروجی y تشکیل شده است. چون ورودی D یک فلیپ‌فلاپ، 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 معادله ای است که شرایط گذر حالت را برای یک فلیپ‌فلاپ بیان می‌کند. سمت چپ معادله با (t+1) حالت بعدی فلیپ‌فلاپ 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 صرف نظر کرده و معادلات حالت را به صورت فشرده تری مطابق زیر می‌نویسیم:

$$A(t+1)= Ax + Bx$$
$$B(t+1)= A'x$$

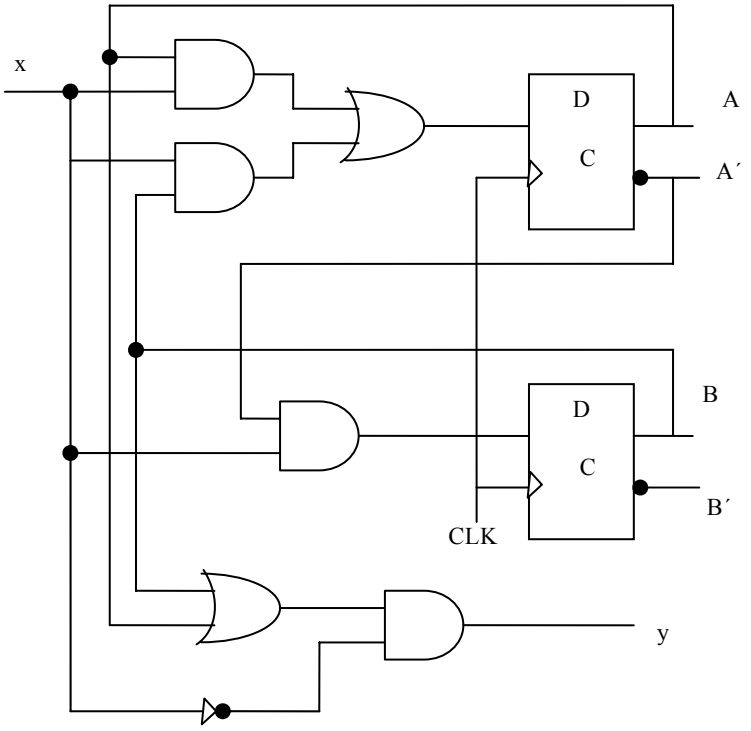
عبارت بولی برای معادلات حالت مستقیماً از گیت‌های تشکیل دهنده بخش ترکیبی در مدار ترتیبی به‌دست می‌آیند، زیرا مقادیر D در مدار ترکیبی حالت بعدی را تعیین می‌کنند.

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 به‌دست می‌آید:

$$y =(A + B)x'$$



شکل ۸-۱٦: مثال مدار ترتیبی

۸-۵-۲ جدول حالت

رشته‌های زمانی ورودی‌ها، و خروجی‌ها و حالات فلیپ‌فلاپ را می‌توان در یک جدول حالت (به آن جدول گذر هم می‌گویند) جمع آوری کرد. جدول حالت برای مدار شکل۸-۱٦ در جدول شکل۸-۱۷ دیده می‌شود. جدول متشکل از چهار بخش با نام های حالت فعلی، ورودی، حالت بعدی و خروجی است. بخش حالت فعلی، حالت فلیپ‌فلاپ‌های A و B را هر لحظه از زمان t نشان می‌دهد. بخش ورودی مقدار x را برای حالت فعلی ممکن به‌دست می‌دهد. بخش حالت بعدی، وضعیت فلیپ‌فلاپ‌ها را

یک سیکل ساعت بعد، در زمان t+1 بیان می‌دارد. بخش خروجی مقدار y را درهر 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 ورودی‌ها نیاز دارد.

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

شکل ۸-۱۷: جدول حالت برای شکل ۸-۱۶

در این حال، ما هشت ترکیب دودویی 000 تا 111 را داریم. سپس مقادیر حالت بعدی از نمودار منطقی یا از معادلات حالت به دست می‌آیند. حالت بعدی فلیپ‌فلاپ 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 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 بر آورده می‌سازند. به‌طور مشابه حالت بعدی فلیپ‌فلاپ B از معادله حالت زیر حاصل می‌گردد.

$$B(t+1)= A'x$$

و هنگامی‌برابر 1 است که حالت فعلی A=0 و ورودی x=1 باشد. ستون خروجی از معادله زیر حاصل می‌شود:

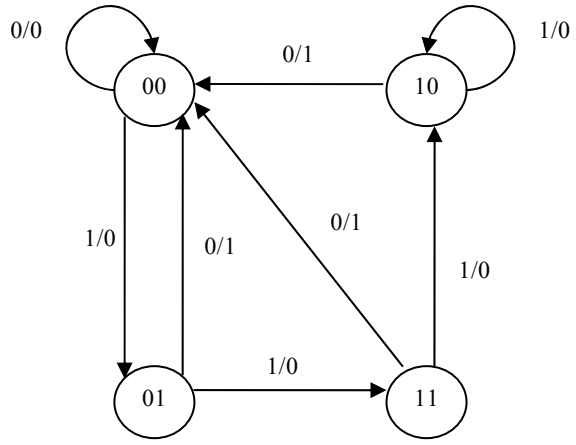
$$y = Ax' + Bx'$$

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

جدول حالت یک مدار ترتیبی با فلیپ‌فلاپ‌های نوع D با روال مشابهی به‌دست می‌آید. به طور کلی، یک مدار ترتیبی با m فلیپ‌فلاپ 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 فلیپ‌فلاپ، 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 به‌دست 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. حالت ورودی در زیر ستون حالت بعدی و ستون خروجی ذکر می‌شود. جدول حالت شکل۸-۱۷ با توجه به این روش، به جدول زیر 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 ارجعیت دارد. اطلاعات موجود در جدول حالت را می‌توان به صورت گرافیکی با نمودار حالت نشان داد. در این 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 شکل۸-۱٦ در شکل۸-۱۸ ملاحظه می‌گردد. نمودار حالت همان اطلاعات جدول حالت را بیان می‌کند که مستقیماً از جدول شکلهای۸-۱۷ به‌دست می‌آید. عدد دودویی داخل هر دایره حالت فلیپ‌فلاپ‌ها را بیان می‌نماید.

خطوط جهت دار 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 جهت دار، 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 جهت دار که از دایره 01 سرچشمه گرفته، حاصل شده است. یک خط جهت دار که دایره را به خودش وصل کند، به معنی عدم وجود تغییر در حالت است.



شکل ۸-۱۸: نمودار حالت مدار شکل ۸-۱۶

بین جدول حالت و نمودار حالت تفاوتی به جزء نحوه ارائه وجود ندارد. جدول درستی به راحتی از یک معادله حالت و نمودار منطقی حاصل می‌گردد. نمودار حالت مستقیماً از جدول حالت به‌دست می‌آید. نمودار حالت تصویری از گذر حالات را مجسم می‌کند و برای تفسیر عملکرد مدار مناسب‌تر است. مثلاً، نموار حالت شکل۸-۱۸ به وضوح نشان می‌دهد که، با شروع از حالت 00، مادامی‌که 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.

۸-٦ تحلیل معادلات ورودی با فلیپ‌فلاپ

نمودار منطقی یک مدار ترتیبی متشکل از فلیپ‌فلاپ‌ها و گیت‌هاست. اتصالات میان گیت‌ها مدار ترکیبی را می‌سازند و ممکن است با عبارات بولی نشان داده شوند. آگاهی از نوع فلیپ‌فلاپ‌ها 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 فلیپ‌فلاپ‌ها را تولید می‌کنند با توابع بولی به نام معادلات ورودی فلیپ‌فلاپ نام‌گذاری شده‌اند (گاهی به آنها معادلات تحریک هم می‌گویند). ما از سمبل ورودی فلیپ‌فلاپ برای نام‌گذاری متغیر معادله ورودی و نام خروجی فلیپ‌فلاپ‌ها به عنوان اندیس استفاده خواهیم کرد. مثلاً معادله ورودی زیر یک گیت OR، با ورودی‌های x و y که به ورودیD از فلیپ‌فلاپ متصل‌اند و خروجی آن با Q نام‌گذاری شده است را نشان می‌دهد.

$$DQ = x + y$$

مدار ترتیبی شکل ۸-۱٦ متشکل از دو فلیپ‌فلاپ 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_Aیک فلیپ‌فلاپ D با نام A را مشخص می‌نماید. به همین ترتیب D_B فلیپ‌فلاپ B از نوع D است. عبارات بولی مربوط به این دو متغیر و عبارات خروجی y، بخش ترکیبی مدار ترتیبی را معین می‌کنند.

معادلات ورودی فلیپ‌فلاپ‌ها فرم جبری مناسبی را برای نمودار منطقی یک مدار ترتیبی تشکیل می‌دهند. آنها نوع فلیپ‌فلاپ را با توجه به سمبل فلیپ‌فلاپ مشخص می‌نمایند و مدار ترکیبی که فلیپ‌فلاپ‌ها 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:

$$Q(t + 1) = DQ$$

۸-۶-۱ تحلیل معادلات با کمک فلیپ‌فلاپ‌های D

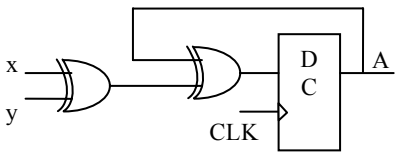
در اینجا روال تحلیل یک مدار ترتیبی متشکل از فلیپ‌فلاپ‌های D را با یک مثال ساده خلاصه می‌کنیم. مداری که برای این هدف در نظر گرفته شده با معادله ورودی زیر توصیف گردیده است.

$$DA = A \oplus x \oplus B$$

سمبل DA یک فلیپ‌فلاپ 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 فلیپ‌فلاپ اخذ شده است. نمودار منطقی از معادله ورودی حاصل و در شکل۸-۱۹(الف)رسم شده است.

حالت بعدی	ورودی‌ها		حالت فعلی
A	x	y	A
0	0	0	0
1	0	1	0
1	1	0	0
0	1	1	0
1	0	0	1
0	0	1	1
0	1		1
1	1	1	1

(ب) جدول حالت



(الف) نمودار مدار

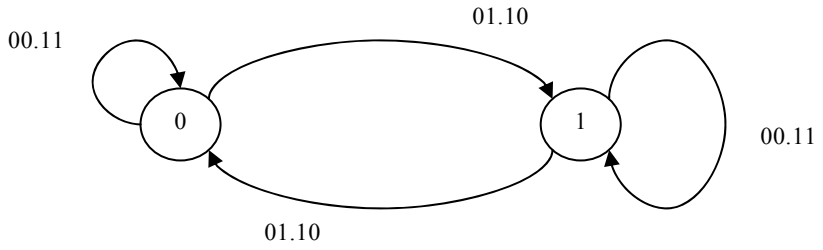
شکل ۸-۱۹: مدار ترتیبی با فلیپ‌فلاپ D

جدول حالت برای حالت فعلی یک ستون داشته و متعلق به فلیپ‌فلاپ A است، دو ستون هم برای ورودی‌ها و یک ستون برای حالت بعدی A لازم است. اعداد دودویی زیر ستون Axy از 000 تا111 مطابق شکل ۸-۱۹(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 هنگامی‌برابر 1 است که فقط یک یا سه متغیر برابر 1 باشد. این نکته در ستون حالت بعدی A قابل ملاحظه است.

مدار دارای یک فلیپ‌فلاپ و دو حالت است. نمودار حالت از دو دایره که هر یک مطابق شکل۸-۲۰ 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 جهت دار 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.



شکل ۸-۲۰: نمودار حالت مدار ترتیبی با فلیپ‌فلاپ D

۸-۶-۲ تحلیل معادلات با کمک فلیپ‌فلاپ‌های JK

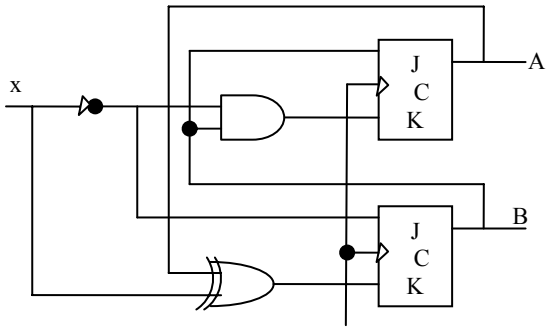
یک جدول حالت متشکل از چهار بخش، حالت فعلی، ورودی‌ها، حالت بعدی و خروجی‌هاست. دو مورد اول با لیست حاصل از همه ترکیبات به‌دست می‌آیند. بخش خروجی از معادلات خروجی حاصل می‌شوند. مقادیر حالت بعدی از معادلات حالت ارزیابی می‌گردند. در فلیپ‌فلاپ نوع D معادله حالت با معادله ورودی یکی است. هنگامی‌که فلیپ‌فلاپ‌هایی 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 به‌دست 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 فلیپ‌فلاپ‌هایی چون نوع JK و T استفاده می‌کنند از رویه زیر به‌دست می‌آید.

- تعیین معادلات ورودی بر حسب حالت فعلی و متغیرهای ورودی
- لیست مقادیر دودویی هر معادله ورودی
- استفاده از جدول مشخصه فلیپ‌فلاپ برای تعیین مقادیر حالت در جدول حالت.

به عنوان یک مثال، مدار ترتیبی متشکل از دو فلیپ‌فلاپ A و B از نوع JK و یک ورودی x را طبق شکل۸-۲۱ ملاحظه نمایید. مدار دارای خروجی خاص نیست و

بنابراین نیازی به ستون خروجی در جدول حالت وجود ندارد. می‌توان مدار را با معادلات ورودی زیر بیان کرد. $JA= B\ KA = Bx'$

$$JB =x'\ KB = A'x + Ax'=A\oplus x$$



شکل ۸-۲۱: مدار ترتیبی با فلیپ‌فلاپ JK

جدول حالت مدار ترتیبی در جدول شکل۸-۲۲ نشان داده شده است. ستون‌های حالت فعلی و ورودی، هشت حالت ممکن را لیست کرده. مقادیر دودویی زیر ستون‌های "ورودی فلیپ‌فلاپ‌ها" بخشی از جدول حالت نیستند، ولی برای ارزیابی حالت بعدی که در مرحله 2 از رویه ذکر شده لازم‌اند. این مقادیر دودویی مستقیماً از چهار معادله ورودی مشابه با آنچه برای جدول درستی یک عبارت بول حاصل می‌شود، به‌دست آمده‌اند.

حالت فعلی		ورودی	حالت بعدی		ورودی های فلیپ فلاپ			
A	B	X	A	B	JA	KA	JB	KB
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

شکل ۸-۲۲: جدول حالت برای مدار ترتیبی با فلیپ‌فلاپ JK

حالت بعدی هر فلیپ‌فلاپ از ورودی‌های J و K و جدول مشخصه فلیپ‌فلاپ JK در جدول شکل۸-۱٤ 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 JA = KA=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 JB = KB=1 است. چون حالت فعلی B=0 می‌باشد، حالت بعدی B متمم شده و به 1 تغییر می‌یابد. می‌توان مقادیر حالت بعدی را ارزیابی معادلات حالت در معادله مشخصه هم به‌دست آورد. این کار با دنبال کردن روال زیر میسر است:

معادلات ورودی فلیپ‌فلاپ را بر حسب حالت فعلی و متغیرهای ورودی به‌دست آورید.

معادلات ورودی را در معادلات مشخصه فلیپ‌فلاپ جایگزین نمایید تا معادلات حالت حاصل شود.

از معادلات حالت برای تعیین مقادیر حالت بعدی در جدول حالت استفاده نمایید.

معادلات ورودی فلیپ‌فلاپ JK شکل۸-۲۱ در فوق ملاحظه گردید. معادلات مشخصه برای فلیپ‌فلاپ‌ها از جایگزینی A و B به جای اسم Q به‌دست می‌آیند:

$$A(t+1)= JA' + K'A$$
$$B(t+1)= JB' + K'B$$

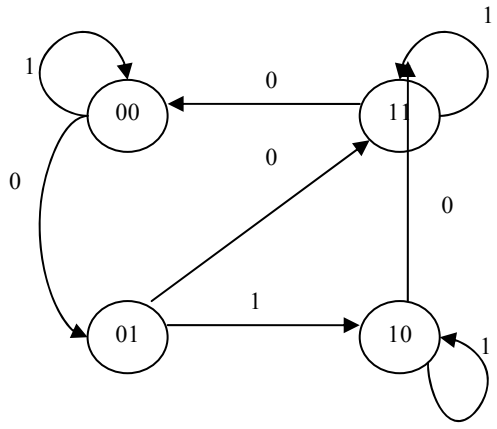
از جایگزینی JA و KA از معادلات ورودی، معادله حالت برای A به‌دست می‌آید:

$$\mathbf{A(t+1)} = BA' +(Bx')'A$$
$$= A'B + AB'+ Ax$$

معادله حالت مقادیر بیتی ستون زیر "حالت بعدی"A را در جدول حالت فراهم می‌سازد. و به‌طور مشابه، معادله حالت برای فلیپ‌فلاپ B با جایگزینی مقادیرJB و KB به‌دست می‌آید.

$$B(t+1) = x'B' + (A \oplus x)'B$$
$$= B'x' + ABx + A'Bx'$$

معادله حالت مقادیر بیتی را برای ستون زیر "حالت بعدیB" را در جدول حالت فراهم می‌نماید. توجه کنید که معادله حالت به کار رود ستون‌های زیر ورودی‌های "فلیپ‌فلاپ" در جدول شکل۸-۲۲ لازم نیستند. نمودار حالت مدار ترتیبی در شکل۸-۲۳ نشان داده شده است. چون مدار دارای خروجی نیست اعداد روی خطوط جهت‌دار خارج شده از دوایر، تنها مقدار ورودی x را بیانگر هستند.



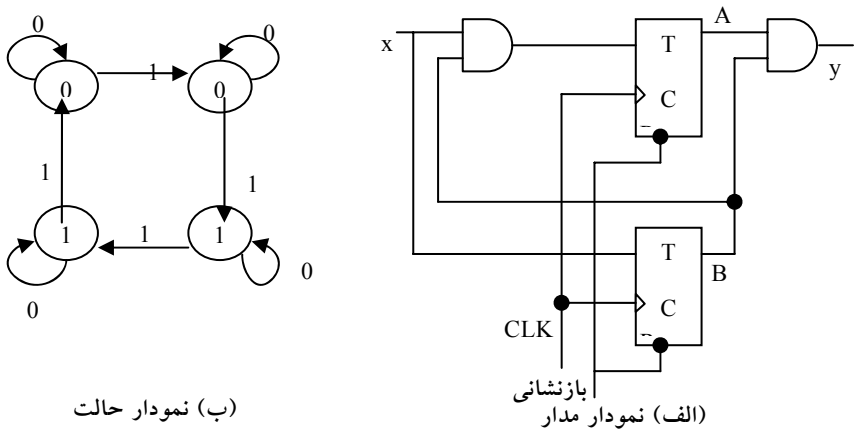
شکل ۸-۲۳: نمودار حالت شکل ۸-۲۱

۸-٦-۳ تحلیل معادلات با کمک فلیپ‌فلاپ‌های T

تحلیل یک مدار ترتیبی با فلیپ‌فلاپ‌های T روال یکسانی با نوع JK دارد. مقادیر حالت بعدی در جدول حالت با جدول مشخصه شکل۸-۲٤ یا با معادله مشخصه زیر به‌دست می‌آیند.

$$Q(t+1)= T \oplus Q = T'Q + TQ'$$

مدار ترتیبی شکل۸-۲٤ را ملاحظه نمایید. این مدار دارای دو فلیپ‌فلاپ‌های A و B، یک ورودی x و یک خروجی y است.



شکل ۸-۲٤: مدار ترتیبی با فلیپ‌فلاپ‌های T

این مدار با دو معادله ورودی و یک معادله خروجی قابل توصیف می‌باشد.

$$TA = Bx$$
$$TB = x$$
$$y = AB$$

جدول حالت برای مدار در جدول شکل ۸-۲۵ لیست شده است. مدار y از معادله خروجی به‌دست می‌آیند. مقادیر حالت بعدی از معادلات حالت و با جایگزینی TA , TB در معادلات مشخصه حاصل می‌شوند، یعنی:

$$\mathbf{A(t+1)} \quad = (Bx)'A + (Bx)A'$$
$$= AB' + Ax' + A'Bx$$
$$\mathbf{B(t+1)} \quad = x \oplus B$$

مقادیر حالت بعدی در جدول حالت از عبارات مربوط به دو معادله حالت به‌دست می‌آید. نمودار حالت مدار در شکل ۸-۲٤(ب)ملاحظه می‌شود. مادامی‌که 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.

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

شکل ۸-۲۵: جدول حالت برای مدار ترتیبی با فلیپ‌فلاپ‌های T

سؤالات

۱- با استفاده از فلیپ‌فلاپ D و یک مولتی‌پلکسر و یک وارونگر، یک فلیپ‌فلاپ JK بسازید.

۲- یک مدار ترتیبی با دو فلیپ‌فلاپ D و معادلات زیر مشخص شده است که در آن A و B فلیپ‌فلاپ‌ها، x و y ورودی ها و z خروجی می‌باشد.

$A(t + 1) = x'y + xA$

$B(t + 1) = x'B + xA$

$Z = B$

نمودار منطقی و نمودار حالت مرتبط را رسم کرده و جدول حالت را برای مدار ترتیبی لیست کنید.

۳- یک مدار ترتیبی با دو فلیپ فلاپ از نوع D و یک ورودی x طراحی کنید . شرایط زیر باید در طراحی در نظر گرفته شود

- وقتی $x = 0$ است، حالت مدار بدون تغییر باقی می‌ماند.
- وقتی $x = 1$ است، مدار وارد حالات 00 و 01 و 11 و 10 و بازگشت به 00 شده و کار تکرار شود.

٤- مدار نمودار ترتیبی حالت شکل۱۸-۸ را با استفاده ار فلیپ فلاپ T طراحی نمایید.

۵- یک مدار ترتیبی با دو فلیپ فلاپ JK و دو ورودی E و x طراحی نمایید که شرایط ذیل را داشته باشد:

- اگر $E = 0$ باشد ، مدار بدون توجه به x در حالت فعلی خود می‌ماند
- اگر $E = 1$ و $x = 1$ باشد مدار وارد حالات 00 و 01 و 10 و 11 و بازگشت به 00 شده و کار را تکرار کند.

- اگر E = 1 و x = 0 باشد، مدار وارد حالات 00 و 11 و 10 و 01 شده و به حالت 00 رفته و عمل را تکرار نماید.

فصل ۹

ثبات‌ها و شمارنده‌ها

هدف کلی

در این فصل مباحث اصلی مربوط ثبات‌ها مورد بحث و بررسی قرار خواهند گرفت و انواع عملیات شامل انتقال و شیفت و ... در ثبات‌ها بررسی خواهند شد. همچنین شمارنده‌ها به‌عنوان مدارهای پایه مورد بررسی قرار گرفته و انواع شمارنده‌ها با جزئیات بررسی خواهند شد.

هدف ساختاری

در این فصل عناوین زیر مورد بحث و بررسی قرار می‌گیرند:

- ثبات‌ها
- روش‌های بار شدن ثبات‌ها
- انتقال و شیفت اطلاعات بین ثبات‌ها
- مفاهیم شمارنده‌ها
- شمارنده‌های موج گونه
- شمارنده‌های دودویی
- شمارنده‌های همزمان
- انواع دیگر شمارنده‌ها

۹-۱ ذخیره‌سازی دودویی و ثبات‌ها

اطلاعات دودویی در یک کامپیوتر دیجیتال، نوعی موجودیت فیزیکی در یک محیط ذخیره‌سازی اطلاعات برای ذخیره تک تک بیت‌ها باید داشته باشد. یک سلول دودویی وسیله‌ای است که از خود دو حالت با ثبات 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 یا 0 است.

یک مدار ترتیبی ساعت دار متشکل از گروهی از فلیپ‌فلاپ و گیت‌های ترکیبی است که به منظور تشکیل یک مسیر پسخورد به هم متصل شده‌اند. فلیپ‌فلاپ‌ها 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 فلیپ‌فلاپ حتی در نبود گیت‌های ترکیبی باز هم یک مدار ترتیبی است. مدارهای حاوی فلیپ‌فلاپ‌ها معمولاً بر حسب کارشان و نه با نام مدار ترتیبی دسته‌بندی می‌شوند. دو نوع از این مدارها ثبات‌ها و شمارنده‌ها هستند.

۹-۱-۱ ثبات‌ها

یک ثبات در مفهومی‌ساده و ابتدایی، گروهی از سلول های 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 ۱ها و 0 ها 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:

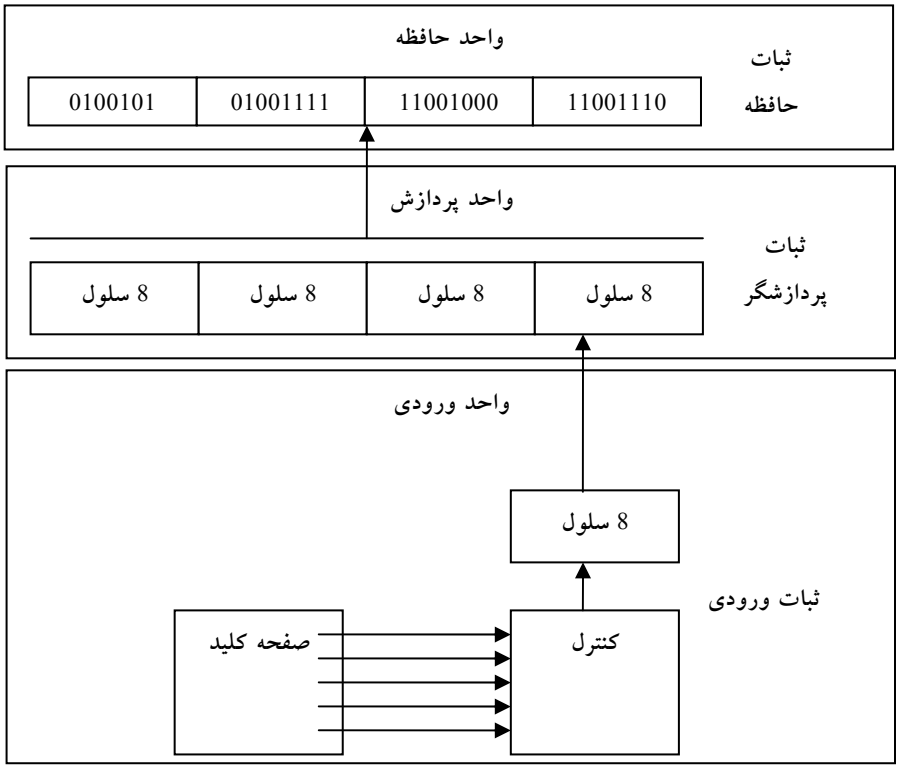
1100001111001001

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 با ارزش، ثبات حاوی دو کاراکتر C(هشت بیت سمت چپ) و I (هشت بیت سمت راست) می‌باشد. از طرف دیگر اگر محتوای ثبات به‌صورت چهار رقم دهدهی تفسیر شود، محتوای ثبات یک عدد دهدهی چهاررقمی‌خواهد بود. در کد افزونی -3 مثال بالا عدد دهدهی 9096 است. در کد 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.

۹-۱-۲ انتقال بین ثباتی

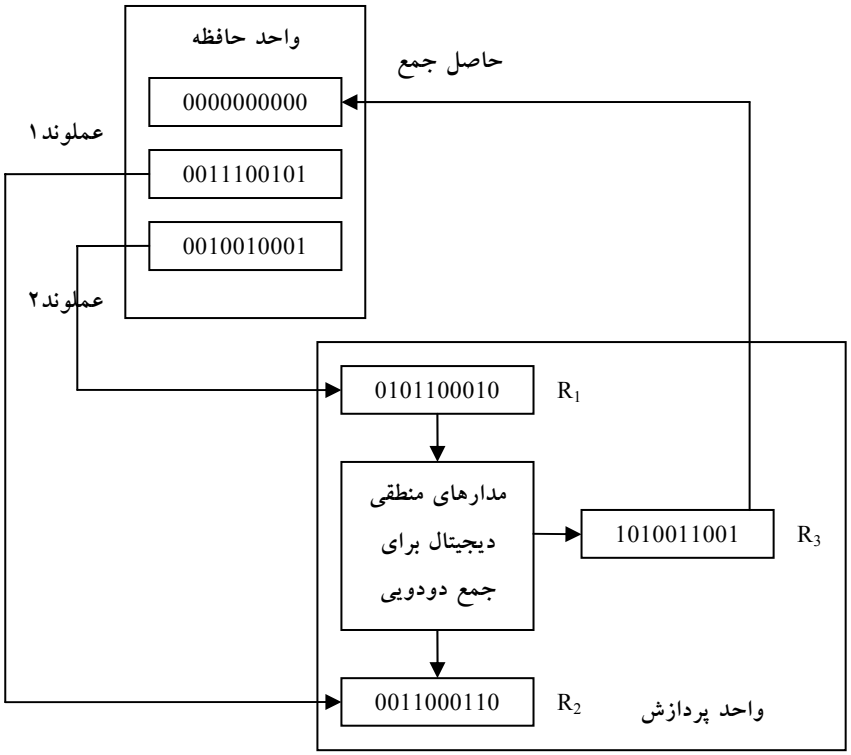
یک سیستم دیجیتال با ثبات‌هایش و قطعاتی که پردازش داده را اجرا می‌کنند مشخص می‌شود. عمل انتقال بین ثباتی یک عمل بنیادی در سیستم‌های دیجیتال است. این عمل متشکل از انتقال اطلاعات دودویی از یک مجموعه ثبات به مجموعه دیگر است. انتقال ممکن است مستقیماً از یک ثبات به دیگری باشد و یا از طریق مدارهای پردازش داده برای انجام یک عمل صورت گیرد. شکل۹-۱ انتقال 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 کم ارزش‌تر 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 شکل۹-۱ ملاحظه می‌گردد از انتقال کاراکترهای "J"، "O" ، "H" ، "N" پس از زدن چهار کلید مناسب حاصل شده است.



شکل ۹-۱: انتقال اطلاعات به وسیله ثباتها

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. شکل۹-۲ فرایند جمع دو عدد دودویی 10 بیتی را نشان می‌دهد. واحد حافظه که معمولاً متشکل از میلیون‌ها ثبات است، در نمودار تنها با سه ثباتش نشان داده شده است.



شکل ۹-۲: مثالی برای پردازش اطلاعات دودویی

بخشی از واحد پردازشگر که در شکل آمده شامل سه ثبات R_1، R_2، 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، R_2 نشان می‌دهد. مدارهای منطقی حاصل جمع را تولید می‌کنند، که بعد به ثباتR_3 منتقل می‌گردد. اکنون محتوای R_3 قابل بازگشت به یکی از ثبات‌های حافظه است.

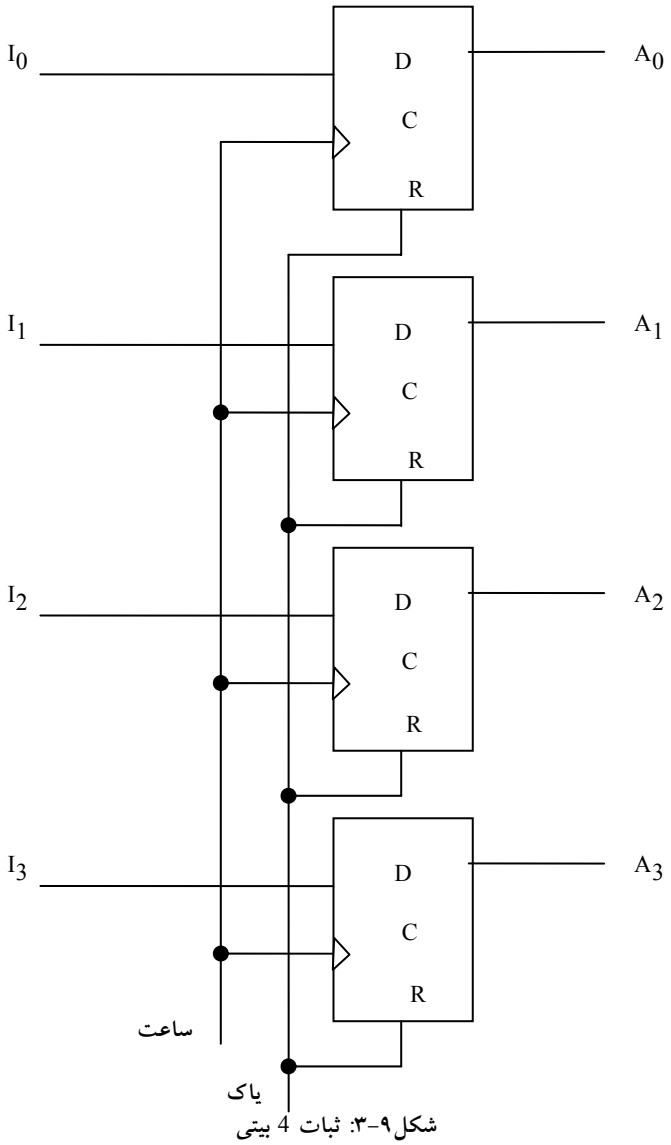
۹-۱-۳ شمارنده‌ها

یک شمارنده در مفهومی‌ساده، 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. هرچند که شمارنده‌ها نوع خاصی از ثبات می‌باشند، معمولاً آنها را با نام های متفاوت از ثبات‌ها جدا می‌کنند.

۹-۲ کاربرد فلیپ‌فلاپ در ثبات‌ها

از آنجائیکه فلیپ‌فلاپ‌ها به عنوان اصلی ترین عضو مدارهای ترتیبی هستند و همچنین در طراحی ثبات بحث استفاده مدارهای ترتیبی مطرح است، لذا می‌توان گفت یک ثبات گروهی از فلیپ‌فلاپ‌ها ست. هر فلیپ‌فلاپ قادر است یک بیت از اطلاعات را در خود ذخیره نماید. یک ثبات n بیت، مجموعه‌ای از n فلیپ‌فلاپ می‌باشد که قادر است n بیت از اطلاعات دودویی را در خود ذخیره نماید. علاوه بر فلیپ‌فلاپ، یک ثبات ممکن است گیت‌های ترکیبی را نیز برای اجرای کارهای پردازشی مختلف داشته باشد. در تعریف جامع‌تر، یک ثبات متشکل از یک گروه فلیپ‌فلاپ و گیت‌هاست که در عمل انتقال با یکدیگر تشریک مساعی دارند. فلیپ‌فلاپ‌ها اطلاعات دودویی را نگه می‌دارند و گیت‌ها چگونگی انتقال اطلاعات رابه ثبات معین می‌کنند.

انواع متنوعی از ثبات‌ها در بازار وجود دارند. ساده‌ترین ثبات، فقط از فلیپ‌فلاپ‌ها و بدون هر گونه گیتی تشکیل شده است. شکل۹-۳ چنین ثباتی را که از چهار فلیپ‌فلاپ D ساخته شده نشان می‌دهد.



شکل۹-۳: ثبات 4 بیتی

ساعت ورودی مشترک همه فلیپ‌فلاپ‌ها را با لبه مثبت هر پالس تریگر می‌کنند و به این ترتیب اطلاعات دودویی در چهار ورودی به داخل ثبات 4 بیت منتقل می‌گردند. می‌توان هر لحظه چهار خروجی را نمونه برداری کرد و اطلاعات دودویی ذخیره شده در ثبات را به‌دست آورد. ورودی پاک به ورودی بازنشان (R) همه فلیپ‌فلاپ‌ها می‌رود. وقتی این ورودی به 0 رود، همه فلیپ‌فلاپ‌ها به طور غیر همزمان بازنشانی (0) می‌شوند. ورودی پاک کردن برای 0 کردن ثبات قبل از عمل ساعت‌زنی مفید است.

در حین عمل معمول ساعت زنی، ورودی‌های R باید در منطق 1 قرار گیرند. توجه کنید که برای 0 کردن همه حالات در یک ثبات، می‌توان از پاک کردن، یا بازنشانی استفاده کرد.

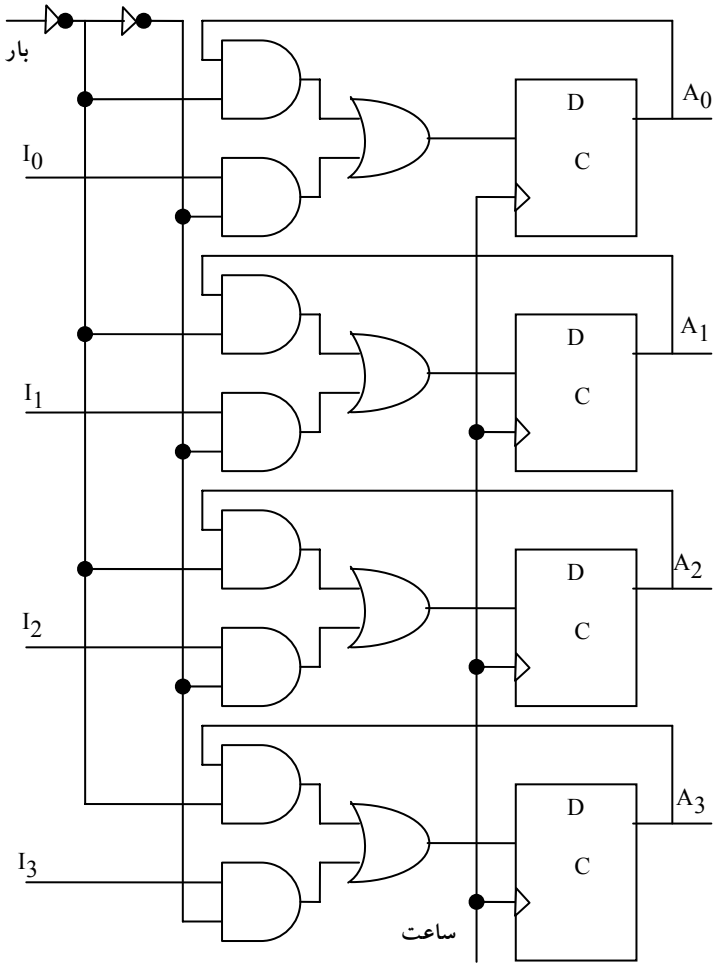
۹-۲-۱ ثبات با بارشدن موازی

سیستم‌های دیجیتال هم زمان دارای یک مولد ساعت اصلی اند که رشته ای از پالسهای ساعت را به طور پیوسته فراهم می‌سازند. پالسهای ساعت به همه فلیپ‌فلاپ‌ها و ثبات‌ها در سیستم اعمال می‌گردند. ساعت اصلی مانند پمپی است که ضربان ثابتی را برای همه بخش های سیستم فراهم می‌نماید. برای تاثیر یک پالس ساعت خاص بر روی یک ثبات خاص، باید یک کنترل جداگانه به کار برده شود.

انتقال اطلاعات جدید به یک ثبات را بارشدن ثبات نامند. اگر همه بیت‌های ثبات به طور همزمان با یک پالس بار شوند گوییم بارشدن موازی است. لبه ساعت اعمال شده به ورودی‌های C ثبات شکل۹-۳ 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 تاخیرهای نابرابر در فلیپ‌فلاپ‌ها می‌گردد. برای همزمانی کامل سیستم، باید مطمئن بود که همه پالس‌های

ساعت به طور همزمان به هر نقطه از سیستم می‌رسد و بنابراین همه فلیپ‌فلاپ‌ها به طور همزمان تریگر می‌شوند. اعمال پالس ساعت از طریق گیت، تاخیرهای 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 فلیپ‌فلاپ‌ها انجام گیرد.

یک ثبات 4 بیتی با ورودی کنترل بارشدن که از طریق گیت‌ها به ورودی‌های D



شکل ۹-٤: ثبات 4 بیتی با بارشدن موازی

فلیپ‌فلاپ هدایت شده در شکل۹-٤ ملاحظه می‌گردد. ورودی بارشدن 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 پبات منتقل

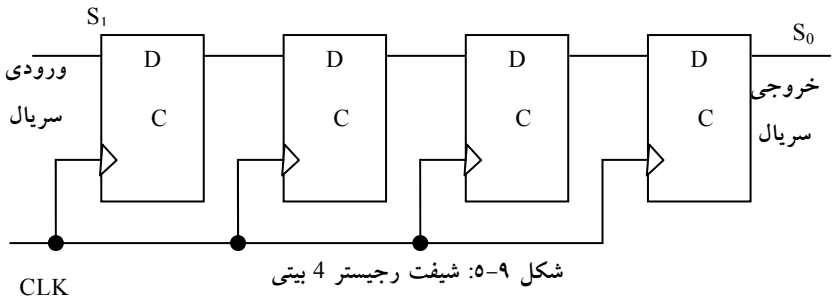
می‌شود. وقتی ورودی بار شدن 0 است، خروجی‌های فلیپ‌فلاپ‌ها به ورودی‌های خودشان وصلند. اتصال پسخوردی از خروجی به ورودی لازم است زیرا فلیپ‌فلاپ D دارای حالت "بی تغییر" نیست. در هر لبه پالس ساعت، ورودی D حالت بعدی فلیپ‌فلاپ را مشخص می‌نماید. برای بدون تغییر نگهداشتن فلیپ‌فلاپ، لازم است D را با حالت فعلی فلیپ‌فلاپ یکسان نماییم.

پالس‌های ساعت مرتباً به ورودی‌های C اعمال می‌گردند. ورودی بارشدن، 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.

۹-۲-۲ شیفت رجیسترها

ثباتی که بتوند اطلاعات دودویی‌اش را به سمت راست یا چپ شیفت یا جابه‌جا کند، شیفت رجیستر یا ثبات جابه‌جایی نامیده می‌شوند. ساختار منطقی یک شیفت رجیستر، از زنجیره‌ای از فلیپ‌فلاپ‌ها تشکیل شده که در آن خروجی یک فلیپ‌فلاپ به ورودی فلیپ‌فلاپ دیگر متصل است. همه فلیپ‌فلاپ‌ها پالس ساعت مشترکی دریافت می‌کنند. پالس‌های ساعت اطلاعات را از یک طبقه به طبقه دیگر جابه‌جا می‌کنند.

ساده‌ترین شیفت رجیستر طبق شکل۹-۵ فقط از فلیپ‌فلاپ‌ها استفاده می‌کند. خروجی یک فلیپ‌فلاپ مفروض به ورودی D فلیپ‌فلاپ سمت راست خود متصل است. هر پالس ساعت محتوای ثبات را یک بیت به راست جابه‌جا می‌کند.



شکل ۹-۵: شیفت رجیستر 4 بیتی

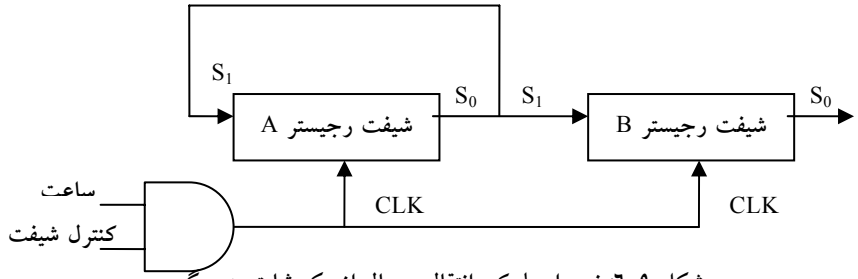
ورودی سریال، تعیین کننده اطلاعاتی است که از منتهی الیه سمت چپ در حین جابه‌جایی وارد می‌شود. خروجی سریال از خروجی سمت راست ترین فلیپ‌فلاپ اخذ می‌گردد. گاهی لازم است تا جابه‌جایی را طوری کنترل کنیم که فقط با پالس‌های معینی رخ دهد. این کار با ممانعت از پالس ساعت در رسیدن به ثبات امکان‌پذیر است. بعد نشان خواهیم داد که عمل جابه‌جایی می‌تواند از ورودی‌های D به جای ورودی ساعت ثبات کنترل گردد. در هر صورت اگر شیفت رجیستر شکل۹-۵ به کاررود، می‌توان عمل جابه‌جایی را به وسیله یک گیت AND و ورودی که جابه‌جایی را کنترل می‌کند تحت کنترل در آورد.

۹-۲-۳ انتقال سریال

اگر یک سیستم دیجیتال هر بار یک بیت را انتقال دهد و یا دستکاری نماید، آنگاه سیستم را فعال در مد سریال می‌نامیم. با جابه‌جایی یک بیت به خارج ثبات مبدا و ورود به ثبات مقصد، اطلاعات هر بار یک بیت انتقال می‌یابد. این بر خلاف انتقال موازی است که در آن همه بیت‌های ثبات به طور همزمان انتقال می‌یابند.

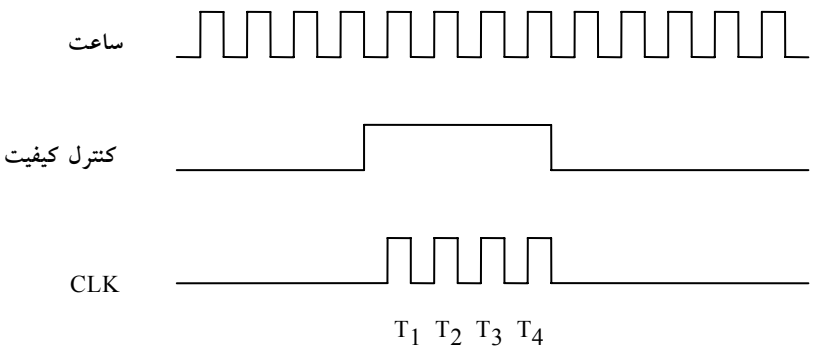
انتقال سریال اطلاعات از ثبات A به ثبات B طبق نمودار بلوکی شکل۹-٦ با شیفت رجیستر انجام می‌شود. خروجی سریال(SO) از ثبات A به ورودی سریال(SI) در ثبات B وصل است. برای پیش گیری در از دست دادن اطلاعات ذخیره شده در ثبات مبدا، اطلاعات ثبات A از خروجی سریال به ورودی سریال چرخانده می‌شود. در حین عمل

جابه‌جایی مقدار اولیه ثبات B به بیرون منتقل شده و از بین می‌رود، مگر اینکه به ثبات سومی‌انتقال یابد. ورودی کنترل جابه‌جایی زمان و تعداد دفعاتی که ثبات‌ها جابه‌جا 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 جابه‌جا خواهد داشت.



شکل ۹-٦: نمودار بلوکی انتقال سریال از یک ثبات به دیگری

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 جابه‌جایی برای مدت چهار پالس ساعت فعال سازد.



شکل ۹-۷: نمودار زمانی انتقال سریال از یک ثبات به دیگری

این مطلب در نمودار زمان بندی شکل۹-۷ ملاحظه می‌شود. سیگنال کنترل جابه‌جایی با ساعت هنگام است و مقدارش درست پس از لبه منفی پالس ساعت تغییر می‌یابد. در چهار پالس ساعت بعدی سیگنال کنترل جابه‌جایی 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 بالارونده پالس یک جابه‌جایی را در هر ثبات انجام می‌دهد. چهارمین پالس کنترل جابه‌جایی را 0 نموده و موجب می‌شود تا شیفت رجیسترها غیر فعال شوند.

فرض کنید که محتوای دودویی A قبل از جابه‌جایی 1011 و B برابر 0010 باشد. انتقال سریال از A به B در چهار مرحله رخ می‌دهد، جدول شکل ۹-۸. 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 جابه‌جا 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 جابه‌جا می‌کنند. پس از چهارمین جابه جایی، کنترل جابه‌جایی به 0 رفته و هر دو ثبات A و B دارای مقدار 1011 خواهند بود. بنابراین محتوای A به B منتقل شده است، ضمن اینکه A همچنان بدون تغییر باقی می‌ماند.

پالس زمانی	شیفت رجیستر A				شیفت رجیستر B			
مقدار اولیه	1	0	1	1	0	0	1	0
پس از T1	1	1	0	1	1	0	0	1
پس از T2	1	1	1	0	1	1	0	0
پس از T3	0	1	1	1	0	1	1	0
پس از T4	1	0	1	1	1	0	1	1

شکل۹-۸: مثالی از انتقال سریال بین ثبات‌ها

با توجه به این مثال تفاوت بین مد های 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 جابه‌جا می‌شوند.

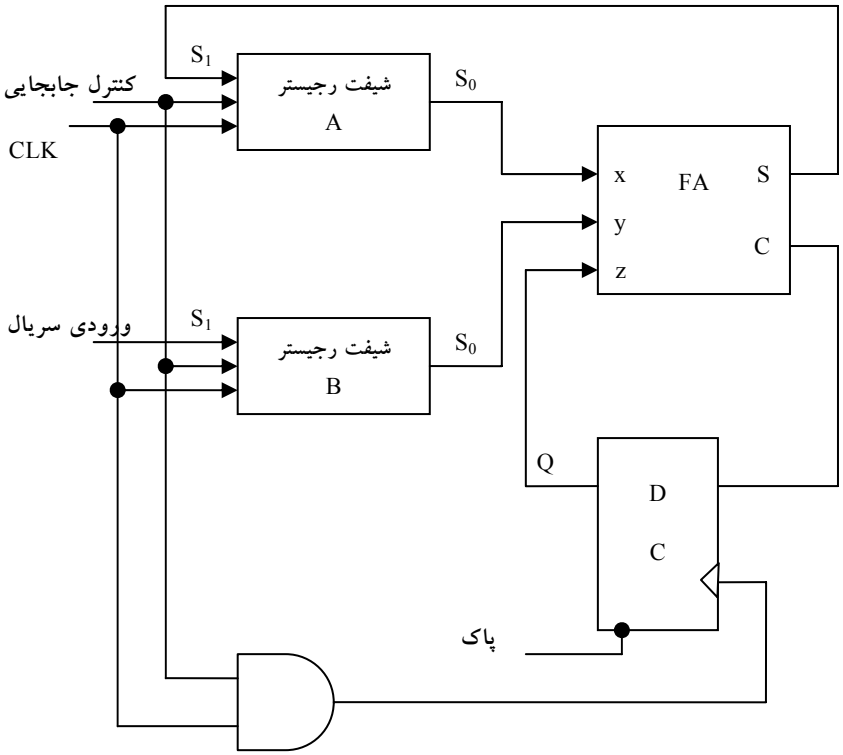
۹-۲-٤ جمع‌کننده سریال

همانگونه که در فصول قبل مشاهده گردید، عملیات در کامپیوترهای دیجیتال معمولاً به صورت موازی صورت می‌گیرد، زیرا این روش سریع‌ترین نوع است. در مقابل عملیات سریال کندتر است، ولی به قطعات کمتری نیاز دارد. برای ارائه مد سریال، در اینجا یک جمع‌کننده سریال را نشان می‌دهیم.

دو عددی که قرار است به طور سریال با هم جمع شوند در دو شیفت رجیستر ذخیره می‌شوند. بیت‌ها، هر جفت یک بار به وسیله یک جمع‌کننده کامل سریال (FA)، با هم جمع می‌شوند، شکل۹-۹. نقلی خروجی جمع‌کننده کامل به فلیپ‌فلاپ D انتقال می‌یابد. آنگاه خروجی این فلیپ‌فلاپ به عنوان نقلی ورودی به جفت بیت با ارزش‌تر بعدی اضافه می‌شود. با انتقال حاصل جمع به A، که با جا به جایی 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 جابه‌جا 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 اینکه فلیپ‌فلاپ نقلی به 0 پاک شده است. خروجی Q از فلیپ‌فلاپ نقلی ورودی در z را تهیه می‌کند. کنترل جابجایی هر دو ثبات و فلیپ‌فلاپ 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

فلیپ‌فلاپ A می‌رود و رقم نقلی به فلیپ‌فلاپ 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.



شکل ۹-۹: جمع کننده سریال

در آغاز، ثبات A در فلیپ‌فلاپ نقلی به 0 پاک می‌شود و سپس اولین عدد از B به آن اضافه می‌گردد. ضمن جابجایی B به جمع کننده، 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 جمع‌کننده موازی (که در فصول قبل ارائه شد)، چندین تفاوت ملاحظه می‌گردد. جمع‌کننده موازی از ثبات‌های با امکان بار شدن موازی استفاده می‌کند، در حالی که جمع‌کننده سری شیفت رجیسترها را به کار می‌برد. تعداد جمع کننده‌های مدار موازی برابر تعداد بیت‌های اعداد دودویی است، در صورتی که جمع‌کننده سریال از یک جمع‌کننده کامل و یک فلیپ‌فلاپ 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 فلیپ‌فلاپ‌ها 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 مورد نظر 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 ببت حاصل جمع را تولید می‌نماید و یک فلیپ‌فلاپ Q برای ذخیره رقم نقلی است. جدول حالتی که مدار ترتیبی را تعریف می‌کند در جدول شکل۹-۱۰ 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 اینکه نقلی ورودی اکنون حالت فعلی Q، و نقلی خروجی اکنون حالت بعدی آن است.

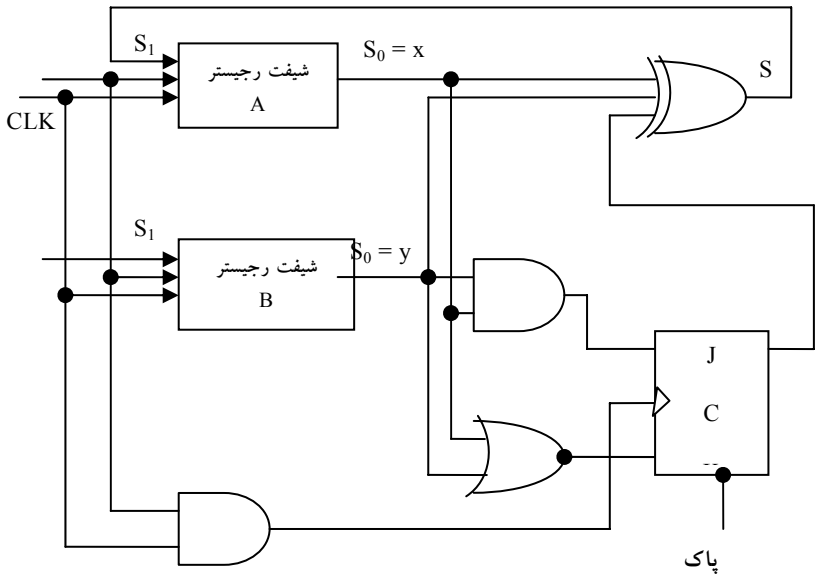
حالت فعلی	ورودی‌ها		حالت بعدی	خروجی	ورودی‌های فلیپ فلاپ ها	
Q	x	y	Q	S	JQ	KQ
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0

شکل ۹-۱۰: جدول حالت یک جمع کننده سریال

اگر از فلیپ‌فلاپ D برای Q استفاده شود مدار به شکل۹-۹ کاهش می‌یابد. اگر برای Q از JK استفاده نماییم، لازم است مقادیر ورودی‌های J و K را با ارجاع به جدول تحریک فصل قبل مربوط به فلیپ‌فلاپ‌های JK معین کنیم. این کار در دو ستون آخر جدول ارائه شده در شکل۹-۱۰ انجام شده است. دو معادله ورودی فلیپ‌فلاپ و معادله خروجی با نقشه به صورت زیر ساده می‌شوند.

$$JQ = xy$$
$$KQ = x'y' = (x+y)'$$
$$S = x \oplus y \oplus Q$$

نمودار مدار در شکل۹-۱۱ نشان داده شده است. مدار متشکل از سه گیت و یک فلیپ‌فلاپ 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 وy 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.



شکل ۹-۱۱: فرم دوم یک جمع کننده سریال

۹-۲-۵ شیفت رجیستر

اگر خروجی فلیپ‌فلاپ‌های یک شیفت رجیستر قابل دسترسی باشد، آنگاه می‌توان اطلاعات وارده سریال را با جابجایی از خروجی فلیپ‌فلاپ‌ها به صورت موازی خارج کرد. اگر به شیفت رجیستر یک قابلیت بارشدن 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:

- کنترل پاک برای پاک کردن ثبات به 0.
- ورودی ساعت برای همزمانی اعمال.
- کنترل جابجایی به راست برای فعال کردن عمل جابجایی به راست و خطوط ورودی و خروجی سریال مربوط به جابجایی به راست.

- کنترل جابجایی به چپ برای فعال کردن عمل جابجایی به چپ و خطوط ورودی و خروجی سریال مربوط به جابجایی به چپ.
- یک کنترل بارکردن موازی برای فعال کردن انتقال موازی و n خط ورودی مربوط به انتقال موازی.
- n خط خروجی موازی.
- حالت کنترلی که علیرغم وجود پالس ساعت اطلاعات را در ثبات بدون تغییر نگه می‌دارد.

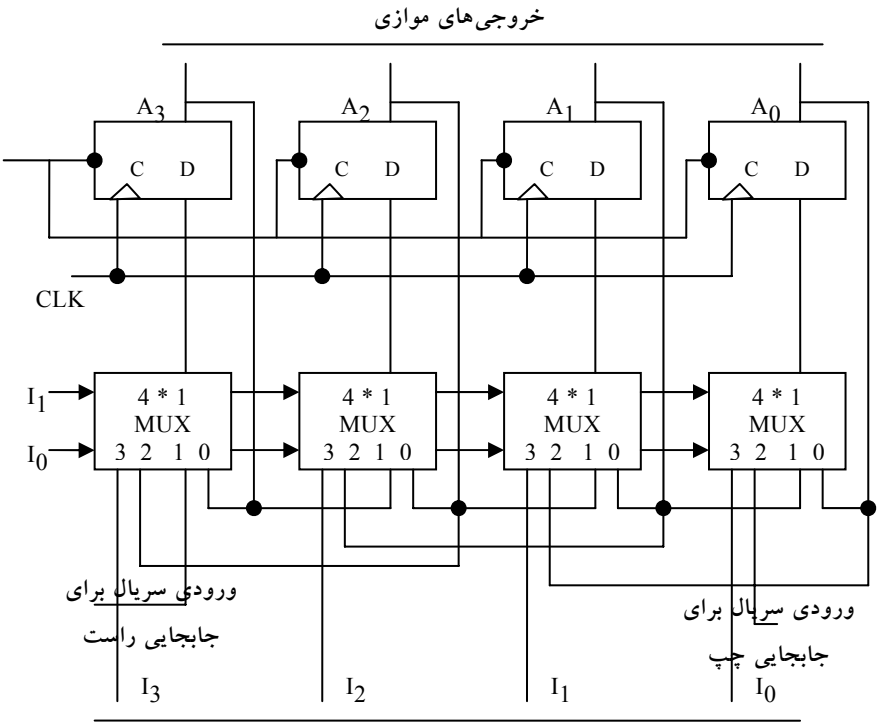
۹-۲-٦ انواع شیفت رجیسترها

دیگر شیفت رجیسترها ممکن است بعضی از امکانات فوق را با حداقل یک عمل جابجایی یا شیفت داشته باشد. انواع شیفت رجیسترها عبارتند از:

- یک جهته
- دو جهته
- یونیورسال

ثباتی که فقط قادر به جابجایی داده در یک جهت باشد را شیفت رجیستر یک جهته گویند. اگر در دو جهت جابجا نماید آنرا 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 فلیپ‌فلاپ‌های فوق الذکر در شکل ۹-۱۲ نشان داده شده است. این مدار از چهار فلیپ‌فلاپ D و چهار مولتی‌پلکسر ساخته شده است. چهار مولتی‌پلکسر دو ورودی انتخاب مشترک S_1 وS_0 دارند.ورودی 0 در هر مولتی‌پلکسر وقتی انتخاب می‌شود که $S_1S_0 = 00$ باشد، ورودی 1 با $S_1S_0 = 01$ انتخاب می‌گردد، و به طور مشابه دو ورودی باقی مانده انتخاب می‌گردند.



شکل ۹-۱۲: شیفت رجیستر یونیورسال 4 بیتی

ورودی‌های انتخاب مد عملیات ثبات را طبق وارده‌های جدول شکل۹-۱۳ کنترل می‌کنند. وقتی $S_1S_0 = 00$ است، مقدار فعلی ثبات به ورودی‌های D از فلیپ‌فلاپ‌ها اعمال می‌گردد. این وضعیت مسیری را از خروجی هر فلیپ‌فلاپ به ورودی‌اش ایجاد می‌نماید. لبه ساعت بعدی، مقداری را که از قبل در آن ذخیره شده وارد فلیپ‌فلاپ می‌کند و بنابراین هیچ تغییری در حالت رخ نمی‌دهد. وقتی $S_1S_0 = 01$ است، پایه ورودی 1 از مولتی‌پلکسر دارای مسیری به ورودی‌های D فلیپ‌فلاپ‌هاست. این موجب عمل جابجایی به راست می‌گردد، که در آن ورودی سریال به فلیپ‌فلاپ A3 وارد می‌شود. وقتی $S_1S_0 = 10$ است، یک عمل جابجایی به چپ صورت می‌گیرد و طی آن دیگر ورودی به فلیپ‌فلاپ 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.

وضعیت کنترل		عملکرد ثبات
S_1	S_0	
0	0	**بلا تغییر**
0	1	**شیفت –راست**
1	0	**شیفت – چپ**
1	1	**بار کردن موازی**

شکل ۹-۱۳ جدول عملکرد ثبات شکل ۹_۱۲

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 اقتصادی تر 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 آنهارا 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.

۹-۳ شمارنده‌های موج‌گونه

ثباتی که بر اساس 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. شمارنده ای که رشته اعداد دودویی را دنبال می‌کند، شمارنده دودویی نامیده می‌شود. یک شمارنده n بیتی متشکل از n فلیپ‌فلاپ بوده و می‌تواند از 0 تا 2^n-1 را بشمارد. شمارنده‌ها به دو صورت وجود دارند:

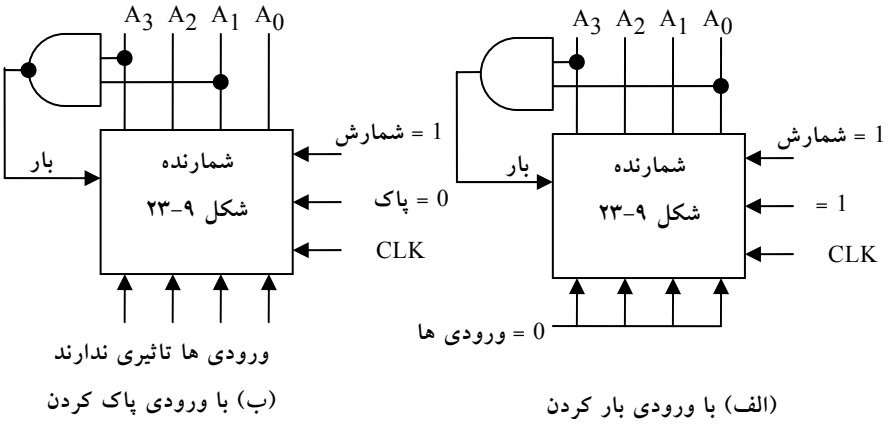
- شمارنده‌های موج گونه
- شمارنده‌های همزمان

در یک شمارنده موج گونه، تغییر وضعیت خروجی فلیپ‌فلاپ به عنوان منبع تریگر کردن دیگر فلیپ‌فلاپ‌ها عمل می‌کند. به بیان دیگر، ورودی C بعضی از و یا همه فلیپ‌فلاپ‌ها با پالس‌های ساعت مشترکی تریگر یا راه اندازی نمی‌شوند. بر عکس در شمارنده همزمان ورودی‌های C همه فلیپ‌فلاپ‌ها ساعت مشترکی، را دریافت می‌نمایند.در اینجا شمارنده‌های موج گونه BCD و دودویی را ارائه کرده و نحوه کار آنها را توضیح می‌دهیم.

۹-۳-۱ شمارنده موج گونه دودویی

یک شمارنده موج گونه دودویی از یک سری اتصال بین فلیپ‌فلاپ‌های متمم ساز تشکیل شده است. که خروجی هر فلیپ‌فلاپ به ورودی C فلیپ‌فلاپ مرتبه بالاتر وصل است. فلیپ‌فلاپی که کم ارزش‌ترین بیت را نگه می‌دارد، پالس‌های مورد شمارش را دریافت می‌کند. فلیپ‌فلاپ متمم ساز را می‌توان با یک فلیپ‌فلاپ JK که در آن J و K به هم وصل اند و یا از یک فلیپ‌فلاپ T ساخت. سومین امکان استفاده از فلیپ‌فلاپ 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 شکل۹-۱٤ نشان داده شده است. شمارنده با فلیپ‌فلاپ‌های متمم ساز نوع T در بخش(الف) و نوع D در بخش(ب) ساخته شده است. خروجی هر فلیپ‌فلاپ به

ورودی فلیپ‌فلاپ بعدی در رشته متصل است. همانطور که گفته شد فلیپ‌فلاپی که کم ارزش‌ترین بیت را نگه می‌دارد پالس‌های شمارش را دریافت می‌کند. ورودی‌های T همه فلیپ‌فلاپ‌ها در(الف) به طور دایم به منطق 1 متصل‌اند. این شرایط موجب می‌شود تا با گذر منفی در ورودی C فلیپ‌فلاپ متمم شود. حباب جلوی نشانه گر دینامیک(*) در کنار C به این معنی است که فلیپ‌فلاپ‌ها به لبه منفی ورودی واکنش نشان می‌دهند. گذر منفی هنگامی‌رخ می‌دهد که خروجی فلیپ‌فلاپ قبل که به C وصل است از 1 به 0 برود.



شکل ۹-۱٤: شمارنده موج گونه دودویی 4 بیتی

برای درک عملکرد شمارنده دودویی 4 بیت، به 9 عدد دودویی اول در جدول شکل۹-۱۵ مراجعه کنید. 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 کم ارزش‌تر 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 موج گونه تغییر می‌کنند. به عنوان مثال، گذر از 0011 به 0100 را در نظر بگیرید. A_0 با پالس ساعت

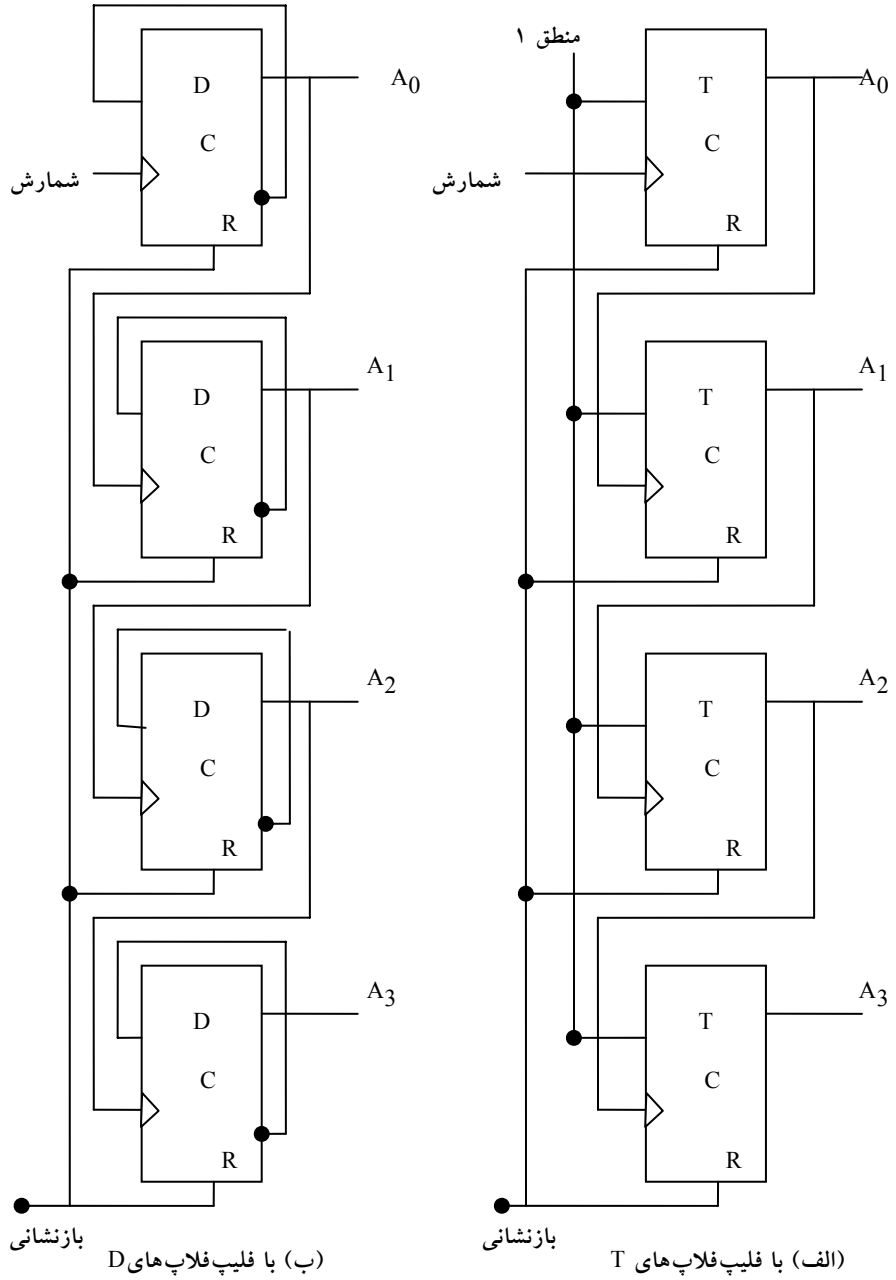
متمم می‌شود. چون A_0 از 1 به 0 می‌رود، A_1 تریگر شده و متمم می‌گردد. در نتیجه A_1 از 1 به 0 می‌رود که به نوبه خود موجب متمم شدن A_2 گشته و آن را از 0 به 1 خواهد برد. A_2 نمی‌تواند A_3 را تریگر کند زیرا A_2 یک گذر مثبت را تولید می‌کند و فلیپ‌فلاپ هم تنها به گذر منفی واکنش نشان می‌دهد.

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

شکل۹-۱۵: جدول نمایانگر رشته شمارش دودویی

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 فلیپ‌فلاپ‌ها تغییر کرده و تغییر به پیش می‌رود و انتشار سیگنال در شمارنده از یک طبقه به طبقه دیگر مثل حرکت موج می‌ماند. برای درک بهتر موضوع به شمارنده ارائه شده در شکل۹-۱۶ توجه نمایید.

یک شمارنده دودویی با شمارش معکوس را پایین شمار گویند. 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 شمارش یک شمارنده پایین شمار نشان می‌دهد که کم ارزش‌ترین بیت با هر پالس شمارش متمم شده است. هر بیت دیگر در رشته، اگر بیت کم ارزش‌تر قبل از آن از 0 به 1 برود، متمم می‌گردد. بنابراین نمودار یک پایین شمار مشابه شکل۹-۱۶ خواهند بود، به شرطی که همه فلیپ‌فلاپ‌ها با لبه مثبت ساعت تریگر شوند.(حباب در

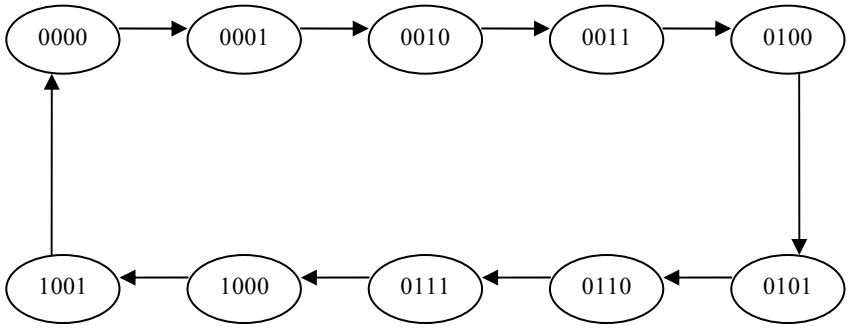


شکل ۹-۱۶: شمارنده موج گونه دودویی ٤ بیتی با فلیپ‌فلاپ‌های D و T

ورودی C باید حذف شود.) اگر از فلیپ‌فلاپ‌های حساس به لبه منفی استفاده شود، آنگاه ورودی C هر فلیپ‌فلاپ باید به خروجی متمم فلیپ‌فلاپ قبلی وصل گردد. آنگاه وقتی که خروجی غیر متمم از 0 به 1 برود، متمم از 1 به 0 رفته و فلیپ‌فلاپ بعدی را آنطور که باید متمم خواهد کرد.

۹-۳-۲ شمارنده BCD موج گونه

یک شمارنده دهدهی رشته‌ای از ده حالت را دنبال کرده و پس از 9 به 0 باز می‌گردد.



شکل ۹-۱۷: نمودار حالت یک شمارنده دهدهی BCD

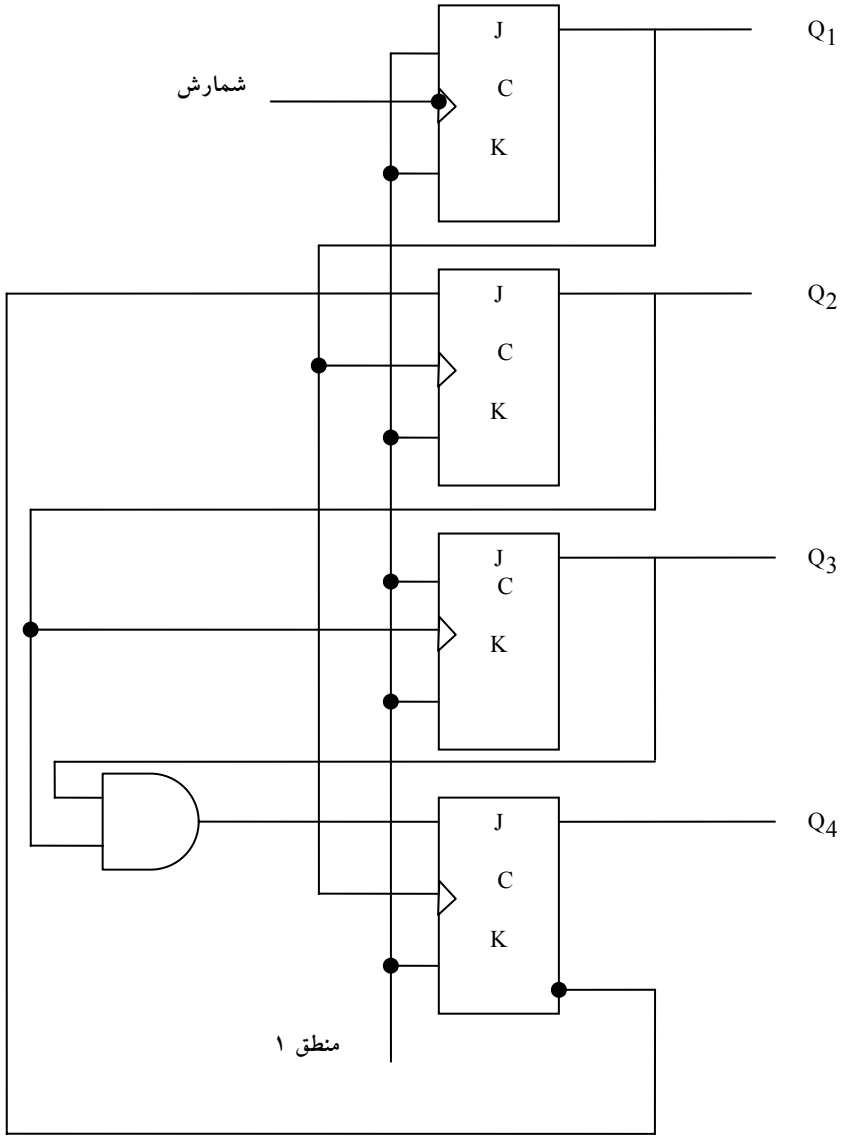
چنین شمارنده‌ای باید حداقل چهار فلیپ‌فلاپ برای نمایش هر رقم دهدهی داشته باشد، زیرا یک رقم دهدهی با کد چهار بیتی نشان داده می‌شود. رشته حالات در شمارنده دهدهی به‌وسیله کد دودویی مربوطه برای نمایش هر رقم دیجیتال معین می‌گردد اگر BCD به کار رود، رشته حالات مطابق نمودار حالت شکل۹-۱۷ خواهد بود. این جدول مشابه با جدول دودویی است. به جز اینکه پس از 1001 برای عدد دهدهی 9، 0000 را برای رقم دهدهی 0 خواهیم داشت.

نمودار منطقی یک شمارنده BCD موج گونه با استفاده از فلیپ‌فلاپ JK در شکل۹-۱۸ دیده می‌شود. چهار خروجی با حروف Q و اندیسی در زیر آن برای مشخص کردن وزن آن در BCD علامت گذاری شده است. توجه کنید که خروجی Q_1 به ورودیهای C در هر دو ورودی Q_2 و Q_3 اعمال شده است و خروجی Q_2 هم به

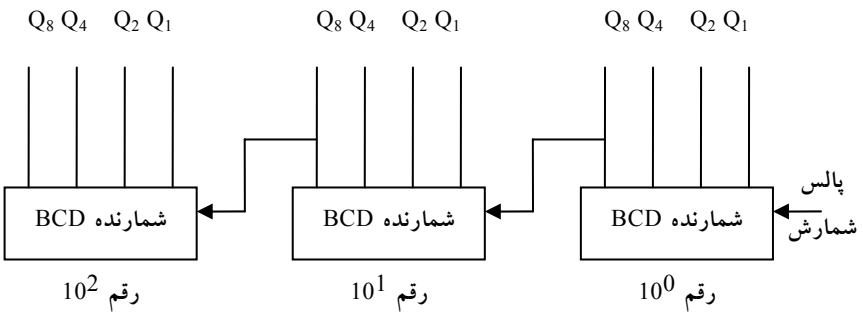
ورودی C از Q_4 وصل است. ورودی‌های J و K یا دائما به 1 وصلند و یا به خروجی فلیپ‌فلاپ‌های دیگر وصل شده‌اند. شمارنده موج گونه یک مدار غیر همزمان است. سیگنال‌ها بسته به ترتیبی که در آن از 1 به 0 می‌روند، روی فلیپ‌فلاپ‌ها اثر می‌گذارند. عمل یک شمارنده با لیستی از حالات گذر هر فلیپ‌فلاپ قابل تفسیر است. این حالات از نمودار منطقی و دانستن چگونگی عملکرد یک فلیپ‌فلاپ JK حاصل می‌شود. به خاطر بسپارید وقتی که ورودی C از 1 به 0 می‌رود، اگر J = 1 باشد، فلیپ‌فلاپ 1 می‌شود و اگر K = 1 باشد به 0 پاک می‌گردد و نیز اگر J = K = 1 باشد متمم شده و بلاخره با J = K =0 حالت فلیپ‌فلاپ بی تغییر خواهد بود.

برای تحقیق و اطمینان از اینکه این حالات به ترتیب در شمارنده BCD رخ می‌دهند باید مطمئن شویم که گذر حالات فلیپ‌فلاپ‌ها رشته ای را که به وسیله نمودار حالت شکل۹-۱۷ مشخص شده دنبال می‌کند. حالت Q_1 پس از هر پالس ساعت عوض می‌شود. هر بار Q1 از 1 به 0 برود و $Q_3 = 0$ باشد Q_2 متمم می‌شود. وقتی $Q_3 = 1$ شود، Q_2 در 0 می‌ماند. هر بار Q2 از 1 به 0 برود Q4 متمم می‌گردد.مادامی‌که Q2 و Q4 در 0 باشند، Q_3 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_3 پاک می‌شود.

شمارنده BCD شکل۹-۱۸ 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 رقمی‌با اتصال سری شمارنده‌های BCD ساخته می‌شوند که هر کدام برای یک دهه است. یک شمارنده دهدهی سه رقمی‌در شکل۹-۱۹ دیده می‌شود. ورودی‌ها به دومین و سومین دهه از Q_3 دهه قبل وارد می‌شوند. وقتی Q_3 در یک دهه از 1 به 0 می‌رود، شمارنده دهه بالاتر را تریگر می‌کند، ضمن اینکه خودش از 9 به 0 باز می‌گردد.



شکل ۹-۱۸: شمارنده موج گونه BCD



شکل ۹-۱۹: نمودار بلوکی یک شمارنده دهدهی BCD با سه دهه

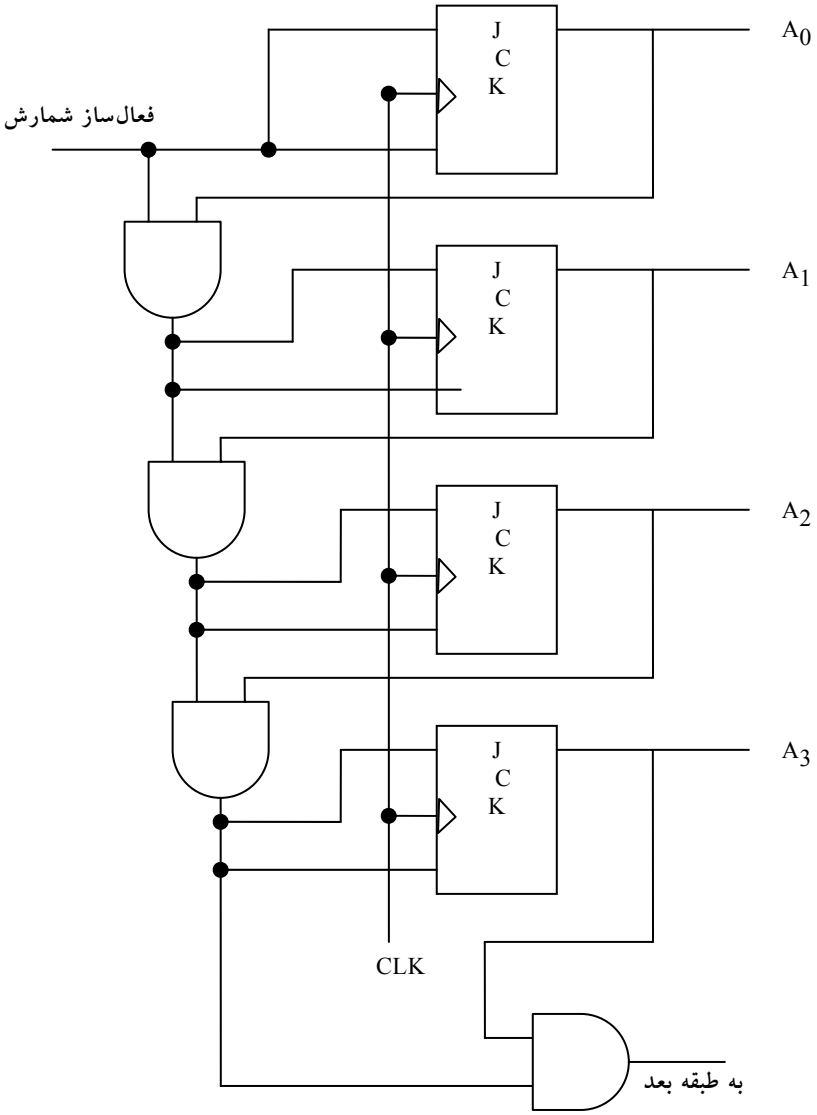
۹-٤ شمارنده‌های همزمان

تاکنون مباحث مختلفی در خصوص شمارنده‌های موج گونه مطرح شد. دسته دوم شمارنده‌ها، عبارتند از شمارنده‌های همزمان. شمارنده‌های همزمان در اعمال پالس ساعت به ورودی فلیپ‌فلاپ‌ها با شمارنده‌های موج گونه تفاوت دارند. یک ساعت مشترک همه فلیپ‌فلاپ‌ها را به طور همزمان تریگر می‌کند در صورتی که در نوع شمارنده‌های موج گونه هر بار فقط یکی از فلیپ‌فلاپ‌ها تریگر می‌شود. تصمیم بر متمم شدن یک فلیپ‌فلاپ از مقادیر داده‌های ورودی مانند T و J و K در لبه ساعت معین می‌شود. اگر T=0 یا J=K=0 باشد، حالت فلیپ‌فلاپ تغییر نمی‌نماید. اگر T=1 یا J=K=1 باشد، فلیپ‌فلاپ متمم می‌گردد.

۹-٤-۱ شمارنده دودویی

در شمارنده دودویی همزمان، فلیپ‌فلاپ واقع در کم ارزش‌ترین مکان با هر پالس یکبار متمم می‌شود. فلیپ‌فلاپ‌های واقع در هر مکان هنگامی‌متمم می‌شود که همه فلیپ‌فلاپ‌های پایین‌تر 1 باشند. مثلاً اگر حالت فعلی یک شمارنده 4

بیت



شکل ۹-۲۰: شمارنده دودویی همزمان ٤ بیتی

$A_3A_2A_1A_0 = 0011$ باشد، شماره بعدی 0100 خواهد بود. لازم به یاد آوری است که در مثال فوق A0 مرتبا متمم می‌شود. A_1 هنگامی‌متمم می‌گردد که A0 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 منظمی‌دارند و می‌توان آنها را با متمم کردن فلیپ‌فلاپ‌ها و گیت‌ها ساخت. نظم الگو را می‌توان با توجه به شکل۹-۲۰ ملاحظه کرد. ورودی‌های C همه فلیپ‌فلاپ‌ها به ساعت مشترکی وصل‌اند. شمارنده با ورودی فعال‌ساز شمارش، فعال می‌گردد. اگر ورودی فعال‌ساز 0 باشد، ورودی همه J ها و K ها 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، J ها و K ها به شرطی 1 هستند که همه طبقات کم ارزش‌تر 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 فلیپ‌فلاپ اضافی خواهد داشت و هرگاه همه فلیپ‌فلاپ‌های طبقات قبل 1 شوند خروجی AND برابر با 1 خواهد بود.

توجه داشته باشید که فلیپ‌فلاپ‌ها در لبه مثبت ساعت تریگر می‌شوند. قطبیت ساعت، آنطور که در شمارنده‌های موج گونه مهم بود، در اینجا اهمیت ندارد. شمارنده همزمان با هر یک از دو لبه مثبت یا منفی پالس ساعت تریگر می‌گردد. فلیپ‌فلاپ‌های متمم ساز در یک شمارنده دودویی می‌توانند از نوع JK، T یا D با گیت‌های XOR باشند.

۹-٤-۲ 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 منظمی‌همچون شمارنده دودویی نیست. برای راه اندازی مدار یک شمارنده همزمان BCD، لازم است از روال طراحی یک مدار ترتیبی استفاده شود.

جدول حالت یک شمارنده BCD در جدول ارائه شده در شکل۹-۲۱ لیست شده است. وضعیت ورودی فلیپ‌فلاپ‌های T از حالت فعلی و بعدی به‌دست 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 با ارزش‌تر بعدی را فعال کرده و به طور همزمان از 1001 به 0000 برود.

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**

شکل ۹-۲۱: جدول حالت یک شمارنده BCD

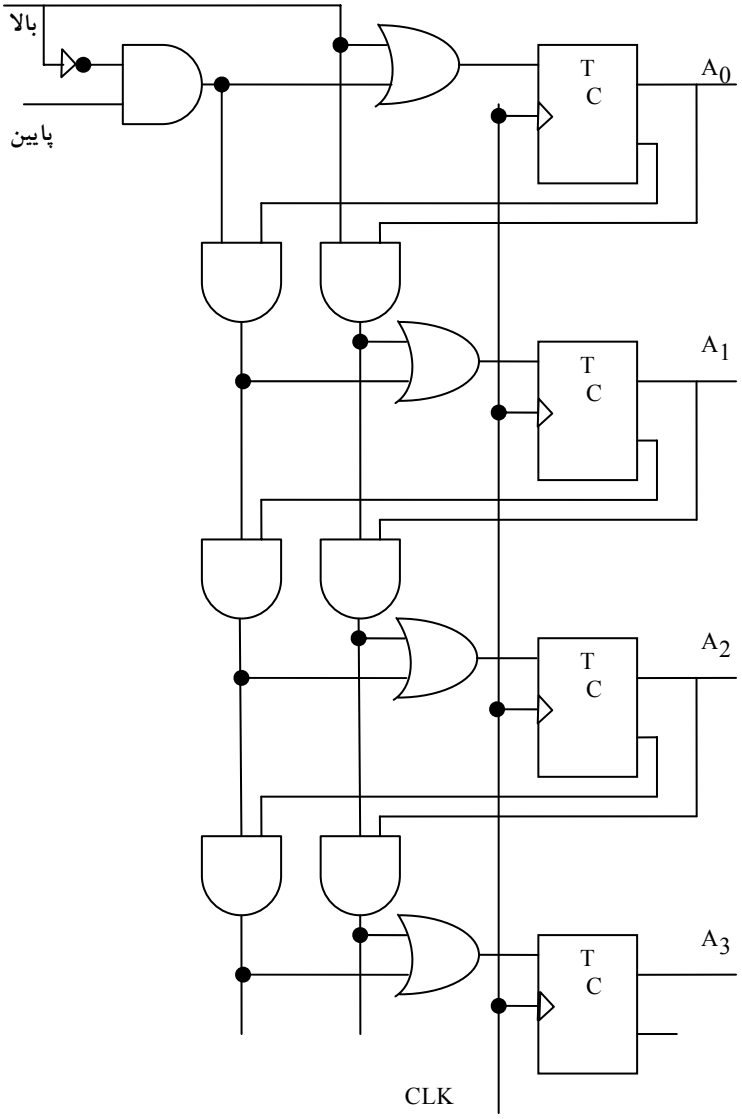
معادلات ورودی فلیپ‌فلاپ‌ها را می‌توان به کمک نقشه ساده کرد. حالات بی استفاده برای مینترم‌های 10 الی 15 جملات بی‌اهمیت تلقی می‌شوند. توابع ساده شده عبارتند از:

$$TQ_1 = 1$$
$$TQ_2 = Q'_8Q_1$$
$$TQ_4 = Q_2Q_1$$
$$TQ_8 = Q_8Q_1 + Q_4Q_2Q_1$$
$$y = Q_8Q_1$$

می‌توان به سادگی با چهار فلیپ‌فلاپ T و پنج گیت AND و یک OR طراحی کرد. شمارنده‌های BCD همرمان را می‌توان با شمارش اعداد دهدهی با هر طول به صورت متوالی به یکدیگر متصل کرد. این نوع سری‌سازی در شکل۹-۱۹ دیده شد، با این تفاوت که خروجی y باید به ورودی شمارش دهه با ارزش‌تر بعدی وصل گردد.

۹-٤-۳ 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 پیش بینی هست. به این ترتیب که بیت مکان کم ارزش‌تر با هر پالس متمم می‌شود. هر بیت در هر مکان دیگر اگر همه بیت‌های کم ارزش‌تر 0 باشند، متمم می‌گردد. مثلاً پس از حالت فعلی 0100، حالت بعدی 0011 قرار دارد. کم ارزش‌ترین بیت همواره متمم می‌گردد. بیت با ارزش‌تر 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 شکل۹-۲۰ ساخته شود، با این تفاوت که ورودی گیت‌ها از خروجی‌های متمم فلیپ‌فلاپ‌های 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 فلیپ‌فلاپ‌های T استفاده می‌کند در شکل۹-۲۲ آمده است.



شکل ۹-۲۲: شمارنده دودویی چهار بیتی، بالا- پایین شمار

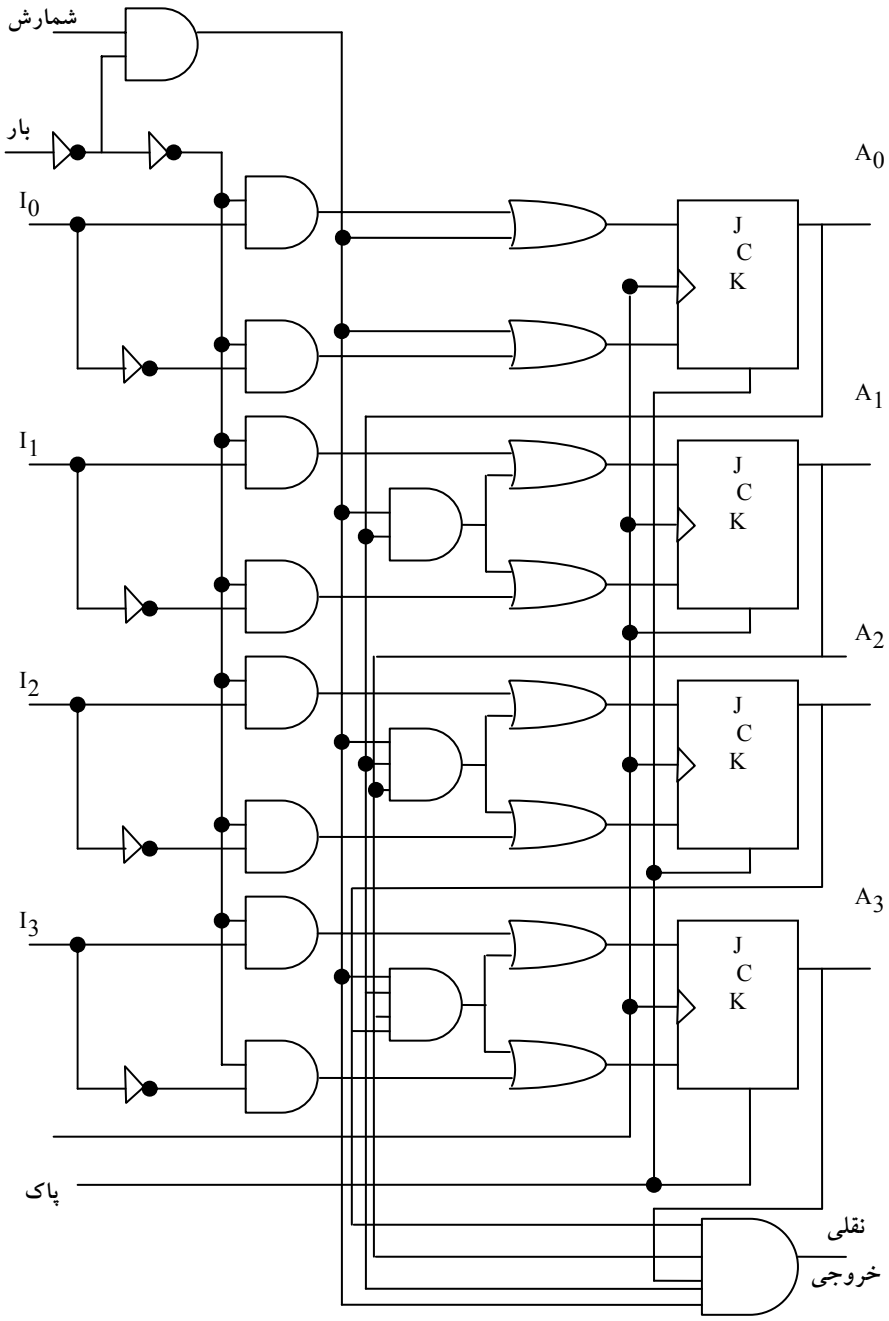
این مدار دارای یک ورودی کنترل بالا و یک ورودی کنترل پایین است. وقتی ورودی بالا برابر 1 است، مدار رو به بالا می‌شمارد، زیرا ورودی‌های T سیگنال‌های خود را از مقادیر خروجی نرمال فلیپ‌فلاپ‌ها دریافت می‌کنند. وقتی ورودی پایین

برابر 1 است ورودی بالا برابر 0 است، مدار رو به پایین می‌شمارد زیرا خروجی‌های متمم شده فلیپ‌فلاپ‌های 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.

۹-٤-٤ شمارنده دودویی با بار شدن موازی

اغلب شمارنده‌های که در سیستم‌های دیجیتال به کار می‌روند نیاز به قابلیت انتقال موازی یک عدد دودویی اولیه به داخل شمارنده، قبل از شروع دارند. شکل۹-۲۳ 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 غیر فعال می‌کند و موجب انتقال داده از چهار ورودی داده به چهار فلیپ‌فلاپ می‌گردد.

اگر هر دو ورودی کنترل 0 باشد، پالس‌های ساعت حالت ثبات را عوض نمی‌کند. ضمن فعال بودن ورودی شمارش، اگر همه فلیپ‌فلاپ‌ها در 1 باشند، خروجی نقلی 1 می‌شود. این حالتی است که طی آن فلیپ‌فلاپ بیت با ارزش‌تر بعدی متمم می‌گردد. خروجی نقلی برای گسترش شمارنده به بیش از چهار بیت نیز مفید است. هنگام تولید مستقیم نقلی خروجی به دلیل کاهش تاخیر آن سرعت شمارنده افزایش می‌یابد. در رفتن از 1111 به 0000، تنها یک گیت تاخیر وجود دارد، در صورتی که در زنجیره گیت AND شکل۹-۲۳، چهار گیت تاخیر موجود است.



شکل ۹-۲۳: شمارنده دودویی ٤ بیتی با امکان بار شدن موازی

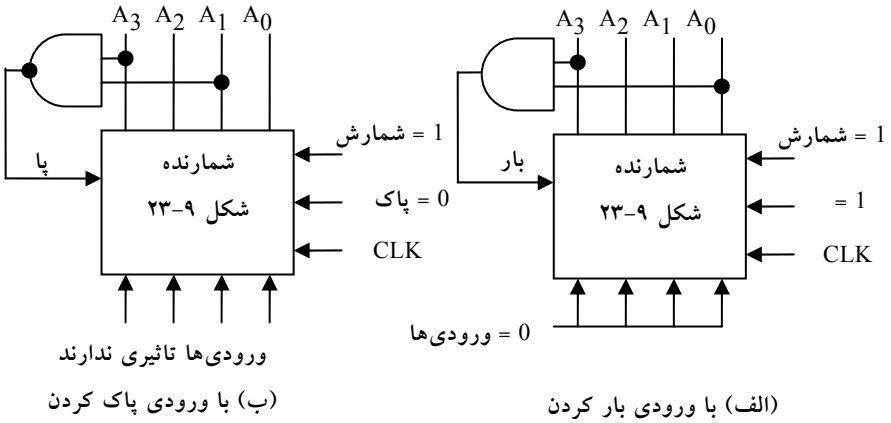
به طور مشابه هر فلیپ‌فلاپ به گیت AND مرتبط است که مستقیماً خروجی همه فلیپ‌فلاپ‌ها را دریافت می‌کند.

عملکرد این شمارنده در جدول شکل۹-۲٤ خلاصه شده است. چهار ورودی کنترل، یعنی پاک، CLK، بار و شمارش حالت بعدی را معین می‌کنند. ورودی پاک غیر همزمان 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 ۱ قرار دارد. با قرار داشتن ورودی‌های بار و شمارش در ۰، 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.

تابع	شمارش	بار	CLK	پاک کننده
به 0 پاک می‌شود	X	X	X	0
ورودی بار شونده	X	1	↑	1
شمارش حالت دودویی بعدی	1	0	↑	1
بلا تغییر	0	0	↑	1

شکل ۹-۲٤: جدول عملکرد شمارنده شکل ۹-۲۳

یک شمارنده با بار شدن موازی را می‌توان برای تولید هر رشته شمارش مورد نظر به کار برد. شکل۹-۲۵ 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.



شکل ۹-۲۵: دو روش دسترسی به شمارنده BCD با امکان بار شدن موازی

گیت AND در شکل۹-۲۵(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. مادامی‌که خروجی گیت AND برابر 0 است، هر لبه ساعت مثبت، شمارنده را یکبار افزایش می‌دهد. وقتی که خروجی به 1001 برسد، A0 و A3 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 ۹-۲۵(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 A0 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 ورودی پاک کردن باشد، می‌توان پس از 1001 آن را پاک کرد.

۹-۵ انواع دیگر شمارنده‌ها

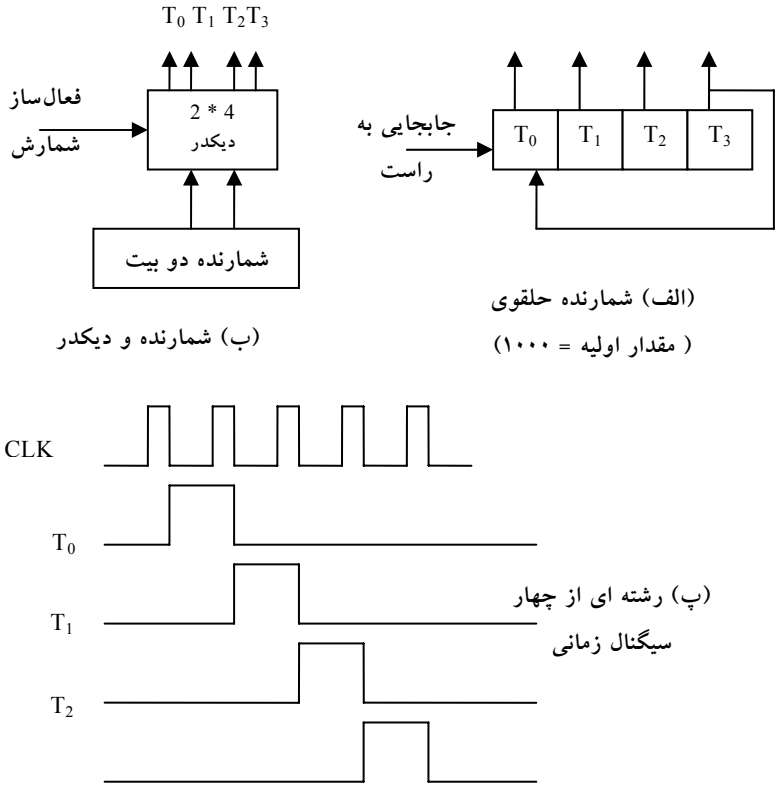
بغیر از دو گروه اصلی شمارنده‌ها که به آنها شمارنده‌های موج گونه و همزمان می‌گفتند، انواع دیگری از شمارنده‌ها نیز وجود دارند.می‌توان شمارنده‌ها را برای تولید هر رشته از حالات طراحی کرد. یک شمارنده تقسیم بر N که به آن N شمار هم می‌گویند، شمارنده ای است که یک رشته N حالتی را تکرار می‌کند. رشته ممکن است 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. در ادامه چند نوع از شمارنده‌های غیر دودویی را ارائه خواهیم نمود:

۹-۵-۱ 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 فلیپ‌فلاپ در حالت 1 است و همه دیگر فلیپ‌فلاپ‌ها صفراند. تنها بیت نشانده (1) به فلیپ‌فلاپ بعدی جابجا می‌شود تا رشته‌ای از سیگنال‌های زمانبندی تولید گردد.

شکل ۹-۲۶(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 یکبار یکی از فلیپ‌فلاپ‌ها 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.

سیگنال‌های زمانبندی با یک شمارنده 2 بیت که به چهار حالت جدا از هم می‌رود نیز تولید می‌گردد. دیکدر شکل۹-۲۶(ب) چهار حالت شمارنده را دیکدر می‌کند و رشته سیگنال‌های زمانبندی لازم را ایجاد می‌نماید.



شکل۹-۲۶: تولید سیگنال‌های زمانی با شمارنده حلقوی

برای تولید 2^n سیگنال، به یک شیفت رجیستر با 2^n فلیپ‌فلاپ 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 فلیپ‌فلاپ نیاز است. در دومی، 4 فلیپ‌فلاپ و 16 گیت AND چهار ورودی برای دیکدر لازم

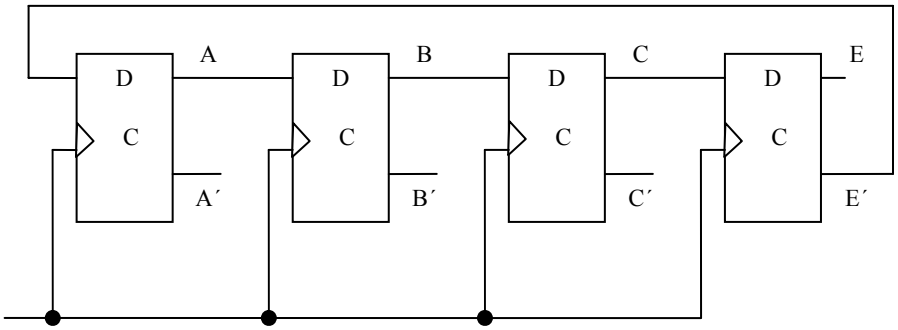
می‌باشد. در این حالت تعداد فلیپ‌فلاپ‌ها کمتر از شمارنده حلقوی است و دیکدر تنها گیت دو ورودی نیاز دارد. این ترکیب را شمارنده جانسون می‌نامند.

۹-۵-۲ شمارنده جانسون

یک شمارنده حلقوی K بیتی، یک بیت را بین K فلیپ‌فلاپ 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 شیف رجیستر دوار است که خروجی متمم آخرین فلیپ‌فلاپ به ورودی اولین فلیپ‌فلاپ متصل شده است. شکل۹-۲۷(الف) چنین شیفت رجیستری را نشان می‌دهد. اتصال پس خوردی از خروجی متمم سمت راست ترین فلیپ‌فلاپ به ورودی سمت چپ ترین فلیپ‌فلاپ ایجاد می‌گردد. شیفت رجیستر مذبور محتوای خود را ب هر پالس ساعت یکبار به سمت راست جابجا می‌کند، و در همان زمان مقدار متمم فلیپ‌فلاپ E به فلیپ‌فلاپ A منتقل می‌گردد.

با شروع از حالت پاک شده، شمارنده حلقوی وارد رشته حالات هشتگانه ای می‌شود، شکل۹-۲۷(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 0 ها از سمت چپ وارد می‌شوند تا وقتی که همه بیت‌های ثبات مجددا 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 شکل۹-۲۷ 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 اینکه هر گیت در حین یک حالت ویژه توانا می‌شود، خروجی گیت‌ها، به ترتیب هشت سیگنال زمانبندی را تولید خواهند کرد.

رمز گشایی یک شمارنده K بیتی حلقوی دنباله چرخان در به دست آوردن 2K دنباله زمانی، از یک الگوی منظم پیروی می‌کند. حالتی که همه بیتها 0 هستند توسط خروجی‌های متمم دو فلیپ‌فلاپ 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 فلیپ‌فلاپ‌های B و C است. بنابراین خروجی دیکد شده با گرفتن متمم B و خروجی طبیعی C یعنی B′C به دست می‌آید.



(الف) شمارنده حلقوی چهار طبقه دنباله چرخان

رشته اعداد	خروجی های فلیپ فلاپ				گیت AND لازم برای خروجی
	A	B	C	E	
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 دیکدکردن مورد نیاز

شکل۹-۲۷ : جدول حالت شمارنده

یکی از معایب مدار شکل۹-۲۷(الف) این است که اگر مدار به حالت بی استفاده وارد شود، شروع به دنبال کردن حالات نا معتبر 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 فلیپ‌فلاپ B است که به ورودی D فلیپ‌فلاپ C می‌رود.

$$DC = (A + C)\,B$$

که DC معادله ورودی فلیپ‌فلاپ برای ورودی D در فلیپ‌فلاپ C است. شمارنده جانسون را می‌توان با هر طول زمانی ساخت. تعداد فلیپ‌فلاپ‌های لازم نصف تعداد سیگنال‌های زمانبندی است. تعداد گیت‌ها برای ساخت آن نصف تعداد سیگنال‌های زمانبندی بوده و فقط گیت دو ورودی نیاز دارد.

۹-۵-۳ شمارنده با حالات بی استفاده

مداری با n فلیپ‌فلاپ 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 به‌دست آورد.

به منظور تشریح، شمارنده مربوط به جدول شکل۹-۲۸ را ملاحظه کنید. شمارش، رشته‌ای تکراری از شش حالت است، که فلیپ‌فلاپ‌های 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 011 111 در شمارش لحاظ نشده‌اند. انتخاب فلیپ‌فلاپ‌های JK شرایط ورودی جدول را به دنبال خواهد داشت. ورودی‌های KB و KC 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 زیر می‌باشند:

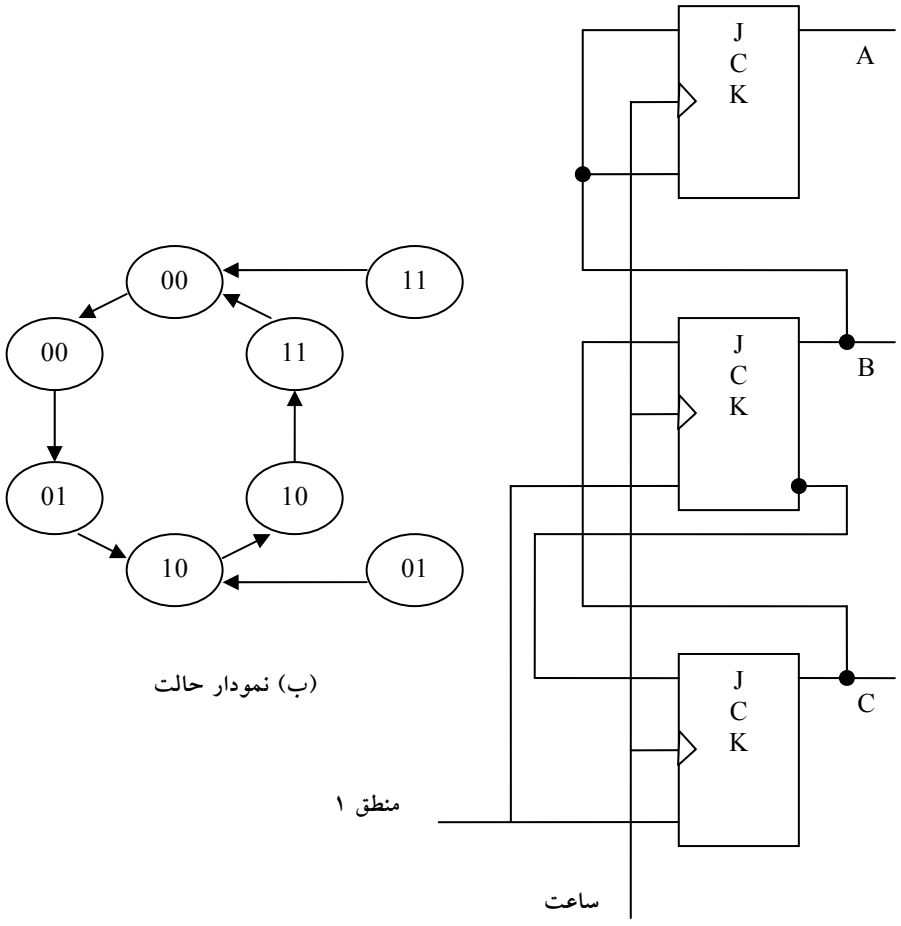
$$KA = B \qquad JA = B$$
$$KB = 1 \qquad JB = C$$
$$KC = 1 \qquad JC = B'$$

حالت فعلی				حالت بعدی				ورودی‌های فلیپ‌فلاپ‌ها					
A	B	C		A	B	C		JA	KA	JB	KB	JC	KC
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

شکل۹-۲۸ : جدول حالت شمارنده

نمودار منطقی شمارنده در شکل۹-۲۹(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 شکل۹-۲۹(ب) ملاحظه می‌شود. اگر مدار به علت عوامل خارجی وارد یکی از حالات بی استفاده 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.



(ب) نمودار حالت

(الف) نمودار منطقی

شکل۹-۲۹: شمارنده‌ای با حالات به کار رفته

سؤالات

۱- محتوای یک ثبات ٤ بیتی در آغاز 1101 است. ثبات شش بار با ورودی سریال 101101 به راست جابجا می‌شود. محتوای ثبات پس از هر جابجایی چیست؟

۲- یک شیفت رجیستر ٤ بیتی با بار شدن موازی را با استفاده از فلیپ‌فلاپ D و دو ورودی shift و load بگونه‌ای طراحی نمایید که شرایط زیر را داشته باشد:

- وقتی shift = 1 است محتوای ثبات یک مکان جابجا می‌شود.
- وقتی load=1 و shift=1 است داده جدید وارد شیفت رجیستر می‌گردد.
- اگر هر دو ورودی کنترل برابر 0 باشند، محتوای ثبات تغییر نمی‌کند.

۳- نمودار منطقی یک پایین شمار موج گونه دودویی ٤ بیت را بار با فلیپ‌فلاپ‌های زیر رسم کنید:

- فلیپ‌فلاپ‌هایی که در لبه مثبت ساعت تریگر می‌شوند
- فلیپ‌فلاپ‌هایی که در لبه منفی ساعت تریگر می‌شوند

٤- یک مدار زمانبندی که یک سیگنال خروجی را به مدت هشت پالس ساعت نگهداری می‌کند، طراحی نمایید. سیگنال شروع حالت 1 را خارج می‌کند و پس از هشت سیکل ساعت سیگنال به حالت 0 باز می‌گردد.

۵- شمارنده‌ای را با استفاده از فلیپ‌فلاپ JK و با رشته دودویی 0 , 1 , 2 , 4 , 6 طراحی نمایید.

٦- نشان دهید که یک شمارنده جانسون با n فلیپ فلاپ یک رشته 2n حالتی تولید می‌کند.

مجموعه سؤالات خودآزمایی

۱- عدد $(۱۱۰۱۰۱/۱۱۱۰۱۱)_۲$ را در مبنای ۸ به‌دست آورید.

الف) $(۶۵ / ۷۲)_۸$ ب) $(۶۵ / ۷۳)_۸$

ج) $(۶۵ / ۳۷)_۸$ د) $(۶۵ / ۳۲)_۸$

۲- عدد $(۲۳۲/۲۲۳)_۴$ را در مبنای ۱۶ بنویسید.

الف) $(۲C / A۳)_{۱۶}$ ب) $(۲A / B۲)_{۱۶}$

ج) $(۲D / ۸۹)_{۱۶}$ د) $(۲C /۵۳)_{۱۶}$

۳- مکمل ۹ عدد $(۳۸۹۴۰)_{۱۰}$ برابر است با:

الف) $(۶۱۰۳۷)_{۱۰}$ ب) $(۶۱۰۵۹)_{۱۰}$

ج) $(۶۹۱۷۸)_{۱۰}$ د) $(۶۱۰۶۰)_{۱۰}$

۴- مکمل ۲ عدد $(۱۰۱۱۱۰۱۱)_۲$ برابر است با:

الف) $(۰۱۰۰۰۱۰۱)_۲$ ب) $(۰۱۰۰۰۱۰۰)_۲$

ج) $(۰۱۰۰۱۱۰۱)_۲$ د) $(۰۱۰۰۱۰۰۰)_۲$

۵-عدد $(۳۵)_{۱۰}$ را با کد دهدهی ۲۴۲۱ بنویسید.

الف) (۱۰۰۱۱۰۱) ب) (۰۰۱۱۱۰۱)

ج) (۱۰۰۱۱۰۱۱) د) (۰۰۱۱۱۰۱۱)

۶- مکمل تابع $F = xy + yz\,(x+y')$ را به‌دست آورید.

الف) $F' = (x'+y')\,(y+z'+x')\,(y+z')$ ب) $F' = (x'+y')\,(y+z'+x')\,z'$

ج) $F' = (x'+y')(y'+z'+x')$ د) $F' = (x'+y')(y zx+zy')$

۷- عبارت $F = AB'C + A'BC+A'B'C+ ABC+AB'C'+A'BC'$ برابر است با:

الف) $F = C' +BC$ ب) $F = C + AB$

ج) $F = C + A$ د) $F= C' + B$

۸- تابع $F(A,B,C,D) = \Sigma(0,1,2,4,5,6,8,9,10,12,13,14)$ برابر است با:

الف) $F =D' +B'$ ب) $F = C' + A$

ج) $F = C' + D'$ د) $F= D + BC$

۹- تابع $F(A,B,C)=\Sigma(0,2,3,7)$ را با شرایط بی اثر $d(A,B,C)=\Sigma(4,5,6)$ در نظر بگیرید تابع F برابر است با:

الف) $F = B +C'$ ب) $F = BC + A$

ج) $F = B' + C$ د) $F= AB+ C'$

۱۰- اگر $A+B=1010$ و $AB=10101$

الف) $B=110$ و $A=100$ ب) $B=111$ و $A=011$

ج) $B=111$ و $A=010$ د) $B=010$ و $A=101$

۱۱- با اتصال ورودی J و K به هم در یک JK-FF اگر خط کنترلی UP برابر 1 باشد، کدام FF حاصل می‌گردد؟

الف) JK-FF کاهنده ب) D-FF افزاینده

ج) RS-FF کاهنده د) T-FF افزاینده

۱۲- حاصلضرب کلیه ماکسترم های یک تابع بول X متغیره برابر است با:

الف) $X/0$ ب) $0/X$

ج) XX د) X^X

۱۳- کدام پاسخ صحیح است؟

الف) $(156)_{10} = (10111001)_2$ ب) $(156)_{10} = (12310)_3$

ج) $(156)_{10} = (2130)_4$ د) $(156)_{10} = (213)_7$

۱٤- متمم دو کدام عدد صحیح نیست؟

الف) $3 \rightarrow 1101$ ب) $7 \rightarrow 1000$

ج) $9 \rightarrow 0110$ د) $12 \rightarrow 0101$

۱۵- در یک دریچه NOR در چه حالتی خروجی پایین است؟

الف)هر ورودی پایین باشد ب) هر ورودی بالا باشد

ج) تمام ورودی ها پایین باشند د) تمام ورودی ها بالا باشند

۱٦- کدام گروه از FF های زیر دارای حالت مبهم نخواهد بود؟

الف) T ، JK ، RS ب) T ، D ، RS

ج) JK ، RS ، JK–MS د) JK ، T ، D

۱۷- یک حافظه با n خط ورودی داده و k خط آدرس دارای چه ظرفیتی است .

الف) $2^{\wedge n} \times k$ ب) $2^{\wedge k} \times n$

ج) 2nk د) n k

۱۸- کدام پاسخ صحیح نیست؟

الف) $(۰/۱۰۱۱)_۲ = (۰/۶۸۷۵)_{۱۰}$ ب) $(۰/۱۰۱)_۲ = (۰/۶۲۵)_{۱۰}$

ج) $(۰/۱۱۰۱)_۲ = (۰/۶۸۷۵)_{۱۰}$ د) $(۰/۱۰۱۰۱)_۲ = (۰/۶۵۶۲۵)_{۱۰}$

۱۹- عدد $(511)_{10}$ در مبنای ۱٦ چه مقدار است؟

الف) 1EF ب) 1FE

ج) FF1 د) 1FF

۲۰- متمم ۸ عدد ۱۷٦= N کدام یک می‌باشد؟

الف) ٦۰۲ ب) ٦۰۱

ج) ۷۱۱ د) ۷۱۲

۲۱- کدام گزینه برای معادل تمام NOR یک گیت XOR صحیح می‌باشد؟

الف) سه دریچه NOR در سه سطح ب) پنج دریچه NOR در سه سطح

ج) چهار دریچه NOR در چهار سطح د) شش دریچه NOR در چهار سطح

۲۲- فرض کنید xy=0 باشد آنگاه y x برابر کدامیک از گزینه‌های زیر است؟

الف) x+y ب) xy'

ج) x−y د) $x'y$

۲۳- یک PAL دارای چند سطح و کدام دریچه و چه ساختار برنامه‌ریزی است؟

الف) دارای دو سطح AND ثابت و OR قابل برنامه‌ریزی

ب) دارای دو سطح AND و OR و هر دو قابل برنامه‌ریزی

ج) دارای دو سطح OR ثابت و AND قابل برنامه‌ریزی

د) دارای دو سطح OR و AND هر دو ثابت

۲٤- هر FF دارای چند حالت است؟

الف) ۱ ب) ۲

ج) ٤ د) n

۲۵- در یک SR–FF، به ازای کدام حالت Qn+1 مبهم است؟

الف) R=1 ، S=1 ب) R=1 ، S=0

ج) R=0 ، S=1 د) R=0 ، S=0

۲٦- دوره تناوب ضربان ساعت یک سیستم که از یک ساعت ۵۰۰ مگا هرتز استفاده می‌کند چند میکرو ثانیه است؟

الف) ۸ ب) ٤

ج) ۲ د) ۱/۲

۲۷- ساده شده عبارت بولی ABC + A′B + ABC′ به‌صورت:

الف) B′ ب) B

ج) A+B د) A

۲۸- عبارت بولی زیر را به منظور کاهش حروف به تعداد تعیین شده ساده کنید؟

(A′ + C) (A′ + C′) (A + B + C′D) به چهار متغیر

الف) A′B + A′C′D ب) B′A+ A′C′D

ج) A + A′C′D د) B′ + AC′D′

۲۹- مکمل تابع F1 = xyz′ + x′ y′ z

الف) (x′ + y + z) (x + y + z) ب) (x + y ′+ z) (x + y + z)

ج) (x + y + z) (x′ + y + z) د) (x + y′ + z) (x + y + z′)

۳۰- مجموع حاصلضرب $(AB + C)(B + C'D)$

الف) $A'B + BC$ ب) $AB' + BC$

ج) $AB + BC$ د) هیچکدام

۳۱- ساده شده عبارت بولی $x y + x' y' z' + x' y z'$

الف) $x 'z' + xy$ ب) $x z' + xy$

ج) $x 'z' + xy '$ د) هیچکدام

۳۲- معادل مبنای 10 بزرگترین عدد n بیتی برابر است با:

الف) $2'n$ ب) $2'n-1$

ج) $3'n-1$ د) هیچکدام

۳۳- عدد $(۱۹۸)_{۱۲}$ در مبنای ۱۰:

الف) $(۲۶۰)_{۱۰}$ ب) $(۲۵۸)_{۱۰}$

ج) $(۲۵۶)_{۱۰}$ د) هیچکدام

۳۴- عدد $(۱۲۳۱)_{۱۰}$ به عدد دودویی برابر است با:

الف) $(۱۱۰۱۱۰۰۱۱۱۱)_{۲}$ ب) $(۱۰۱۱۱۰۰۱۱۱۱)_{۲}$

ج) $(۱۰۰۱۱۰۰۱۱۱۱)_{۲}$ د) هیچکدام

۳۵- عدد ۸۶۲۰ در مبنای ۱۰ به فرم BCD

الف) ۰۰۰۱ ۰۰۱۰ ۰۱۱۰ ۱۰۰۰ ب) ۰۰۱۰ ۰۰۱۰ ۰۱۱۰ ۱۰۰۰

ج) ۰۰۱۰ ۰۰۱۰ ۰۱۱۰ ۱۰۰۰ د) هیچکدام

۳۶- کدام عبارت درست است؟

الف) یک فلیپ فلاپ RS را می توان با ترکیب دو گیت OR و یک گیت NAND ساخت.

ب) یک فلیپ فلاپ RS را می توان با ترکیب دو گیت OR و دو گیت NAND ساخت.

ج) یک فلیپ فلاپ RS را می‌توان با ترکیب دو گیت NAND و یا دو گیت NOR که فیدبک دارند ساخت.

د) هیچکدام

۳۷- در یک مدار مولتی‌پلکسر (ادغام کننده) بین تعداد ورودی‌ها و خطوط آدرس چه ارتباطی برقرار است. (تعداد خطوط آدرس را برابر با n در نظر می‌گیریم و تعداد خطوط ورودی را m می‌نامیم).

الف) $m = 2^{n}$ ب) $n = 2^{m}$

ج) $m = 2^{n}+1$ د) $n = 2^{m}+1$

۳۸- معادل عدد $(10011/1011)_2$در مبنای ۱۰ مساوی:

الف)۱۱/۱۹ ب)۱۹/۱۱

ج) ۱۱/۵۹۳۷۵ د)۱۹/۶۸۷۵

۳۹- کدامیک از پاسخ ها صحیح است ؟

الف) $(34)_8 = (29)_{10} = (43)_6$ ب) $(14/8)_8 = (13/5)_{10} = (1101/1)_2$

ج) $(23)_4 = (11)_{10} = (1101)_2$ د) $(31)_4 = (13)_{10} = (1101)_2$

٤٠- متمم ۲ عدد $(101101)_2$ برابر است با:

الف) $(010010)_2$ ب) $(010011)_2$

ج) $(010001)_2$ د) $(101110)_2$

٤١- عدد $(11001/101101)_2$در مبنای ۸ مساوی است با:

الف) $(31/45)_8$ ب) $(55/31)_8$

ج) $(31/55)_8$ د) $(55/14)_8$

٤٢- مکمل ۹ و ۱۰ عدد $(0/32677)_{10}$ چیست؟

الف) ۰/٦٧٣٢ ، ۰/٦٧٣٢ ب) ۰/٦٧٣٣ ، ۰/٦٧٣٤

ج) ۰/٦٧٣٣ ، ۰/٦٧٣٣ د) ۰/٦٧٣٣ ، ۰/٦٧٣٢

٤٣- تفاوت فلیپ فلاپ RS با JK در چیست؟

الف) هیچ تفاوتی ندارند.

ب) بر خلاف RS اگر در JK هر دو ورودی فعال باشد خروجی مکمل حالت فعلی خواهد شد.

ج) خروجی JK مکمل خروجی RS است.

د) اصلا این دو قابل مقایسه نیست.

٤٤- اگر رشته زیر را به فلیپ فلاپ T بدهیم خروجی برابر خواهد بود:

ورودی = ۱۰۰۰۱۱۰

حالت اولیه خروجی = 0 (Q=0)

الف) ۱۱۱۱۰۱۱ ب) ۱۰۰۰۱۱۰

ج) ۰۱۱۱۰۰۱ د) ۱۰۱۰۱۱۰

٤٥- تفاوت مدار ترتیبی و ترکیبی چیست؟

الف) مدارات ترتیبی حافظه دارند. ب) هیچ تفاوتی ندارند.

ج) مدارات ترکیبی حافظه دارند. د) مدارات ترتیبی نیاز به ورودی ندارند.

٤٦- عدد دهدهی 250.5 در مبنای 3 کدامست؟

الف) 100021.111 ب)3322.2

ج) 10021.21 د) 21211.111

٤٧- عدد دهدهی 12.0625در مبنای 2 کدامست ؟

الف)1101.01 ب)1100.10

ج)1100.0001 د) 1010.1010

٤٨- عدد دودویی 1110101.110 در مبنای ده کدامست ؟

الف)101.25 ب)117.75

ج) 117.11 د) 101.11

٤٩- عدد هشت هشتی 623.77 در مبنای 16 کدامست ؟

الف)193.FC ب)402.984

ج) 311.88 د) C47.54

٥٠- عدد دهدهی 620 به‌صورت BCD و کد افزونی 3 به ترتیب کدام گزینه است؟

الف) قابل نمایش به‌صورت کد افزونی 3 نمی باشد .

ب) 0110 0010 0000 و 1001 0011 0001

ج) 0000 0010 0110 و 1111 1101 1001

د) 0000 0010 0110 و 0011 1001 0101

۵۱- در ارزیابی عبارات جبر بول کدام گزینه صحیح نمی باشد ؟

الف) تقدم اول با پرانتزاست. ب) تقدم AND از NOT بیشتر است.

ج)تقدم NOT از OR بیشتر است. د) تقدم AND از OR بیشتر است.

۵۲- عبارت $(A + B)'(A' + B')'$ معادل است با:

الف) صفر ب) یک

ج) $A + B$ د) $A' + B'$

۵۳- متمم تابع بول $(AB' + CD')(BC' + A'D)$ برابر است با :

الف) ABCD ب) $(A + C')(B + D')$

ج) صفر د) یک

٥٤- کدام گزینه بیان دیگری از تابع $F(x, y, z) = (xy + z)(y + xz)$ میباشد:

الف)$\Sigma(3, 5, 6, 7)$ ب)$\Sigma(2, 3, 5, 6)$

ج)$\prod(0, 1, 2, 4)$ د)الف و ج

۵۵-ساده شده تابع $xyz + xyz' + x'yz + xy'z$ کدام گزینه است ؟

الف) $xy + xz + yz$ ب) $(xy' + yx')z$

ج) xyz د) $x + yz$

٥٦- تابع $F(x, y, z) = \prod(0, 1, 4, 5)$ معادل است با:

الف) y ب) $x'y$

ج) xy د) صفر

۵۷-کدام گزینه صحیح نمی‌باشد ؟

الف) در نیم جمع‌کننده $C = xy$ است.

ب) در نیم جمع‌کننده $S= x \oplus y$ است.

ج) درتمام جمع‌کننده $S=z\oplus(x \oplus y)$ است.

د) در نیم جمع‌کننده $B= xy$ است.

۵۸- عدد دهدهی(۱۲۰/۵) در مبنای ۸ کدام یک از موارد ذیل است؟

الف) ۱۷۰/٤ ب) ۲۲٦/٤

ج) ۲۲٦/۵ د) ٦٤/۵

۵۹- عدد دودویی ۱۰۱/ ۱۱۰۱۱۰۱ در مبنای ۱۰ چه عددی است ؟

الف)۱٤/۷ ب) ۱۰۹/۱۲۵

ج) ۱۰۹/٦۲۵ د) ۲۰۹/۰۵

٦۰- عدد شانزده شانزدهی(در مبنای ۱٦) ۷٦ معادل کدام یک از موارد ذیل است؟

الف) ۱۱۸ در مبنای ۱۰ ب) ۱۱۱۰۱۱۰ در مبنای ۲

ج) ۲۳٤ در مبنای ۷ د) ۱۱٦ در مبنای ۸

٦۱- مکمل ۲ عدد ۱۰۰۱۰۰ چیست؟

الف) ۰۱۱۱۰۰ ب) ۰۱۱۰۱۱

ج) ۰۱۱۱۰۱ د)۱۱۱۱۰۰

٦۲-کدام یک از روابط ذیل غلط است؟

الف) $x (x + y) = x$

ب) $x + xy = x$

ج) $\Sigma(1 , 4 , 5 , 6 , 7) = \prod(0 , 1 , 2, 3)$

د) هیچکدام

٦۳- تابع $F(A , B , C , D) = \prod(0 , 1 , 2 , 4 , 5 , 6 , 8 , 9 , 10 , 12 , 13)$ (با استفاده از جدول کارنو) معادل کدام یک از توابع ذیل است؟

الف) $F = C'D' + AB + A'$ ب) $F = A'B' + B'D'$

ج) $F = D' + B'C'$ د) $F = C' + A'D' + B'D'$

٦٤- اگر با استفاده از جدول کارنو تابع $F(A , B , C , D) = \Sigma(1 , 2 , 3 , 4 , 9)$ با حالات بی‌اهمیت $d(A , B , C , D) = \Sigma(10 , 11 , 12 , 13 , 14 , 15)$ ساده شود ساده ترین تابع ممکن کدام یک از موارد ذیل است؟

الف) $CB'A' + DB'A + BD'C' + DA$

ب) $B'C + DB'A' + BD'C' + DA$

ج) $B'C + B'D + BC'D'$

د) $A + B'D + BC'D'$

٦٥- با توجه به جدول تحریک فلیپ فلاپ ذیل ورودی y کدام یک از موارد است.

ورودی y	حالت بعدی Q(t+1)	حالت فعلی Q(t)
0	0	0
1	1	0
1	0	1
0	1	1

الف) ورودی فلیپ فلاپ D ب) ورودی فلیپ فلاپ T

ج) ورودی s فلیپ فلاپ r s　　د) ورودی r فلیپ فلاپ r s

٦٦- عدد 0.6875 در مبنای 10 معادل جه عددی در مبنای 4 است ؟

الف) 0.23　　ب) 0.32

ج) 3.2　　د)2.3

٦٧- عدد FD7F در مبنای 16 معادل جه عددی در مبنای 2 است؟

الف) 1111110101111111　　ب) 1101011110111101

ج) 1110110111011111　　د) 1101111111100110

٦٨-مکمل مبنای کاهش یافته (مکمل 4) عدد231 در مبنای 5 کدامیک از موارد ذیل است؟

الف)213　　ب) 214

ج) 324　　د) 325

٦٩- کدامیک از روابط ذیل در مورد نتیجه M- N در سیستم مکمل 2 صحیح است؟

الف) $M- N= M + (r^{n} – N)$　　ب) $M- N= M + (r^{n} – N - 1)$

ج)$M- N = M + (N - r^{n})$　　د) $M- N = M + (N - r^{n} + 1)$

٧٠- تابع $f (x , y , z) = x + y +z$ معادل کدامیک از توابع ذیل است؟

الف)$f (x , y , z) = \prod(1)$　　ب)$f (x , y , z) = \prod(0)$

ج)$f (x , y , z) = \prod(7)$　　د) $f (x , y , z) =1$

٧١- معادل تابع$f(A , B , C , D) = \Sigma(0 , 2 , 4 , 6 , 8)$ و

$d(A , B , C , D) = \Sigma(10 , 11 , 12 , 13 , 14 , 15)$ کدامیک از موارد ذیل است؟

الف) $A' + D'$ ب) $CD + C'D'$

ج) D' د) AD

۷۲- با استفاده از کدامیک از ترکیبهای ذیل نمی‌توان توابع منطقی را پیاده‌سازی کرد؟

الف) NAND – AND ب) NAND – NAND

ج) AND – NOR د) NOR – NAND

۷۳- اگر در کد گری، 0011 برای عدد 2 در نظر گرفته شود برای عدد 3 کدامیک از کدهای ذیل را می‌توان استفاده کرد؟

الف) 0010 ب) 1010

ج) 1100 د) 0110

۷٤-کدامیک از توابع ذیل نشان دهنده توابع مدار نیم جمع‌کننده است؟

الف) $s = x \oplus y$ و $c = xy$ ب) $s = xy + y$ و $c = x \oplus y$

ج) $s = x + (x \oplus y)$ و $c = xy'$ د) $s = x' + (x \oplus y)$ و $c = x' y$

۷۵-در استفاده از کد همینگ اگر تعداد بیتهای تست 4 باشد ($k=4$) حداکثر تعداد بیتهای داده کدامیک از موارد ذیل است؟

الف) 11 ب) 12

ج) 13 د) 14

۷٦-اگر به ورودی یک T فلیپ فلاپ که در آن $Q = 1$ است رشته صفر و یک $x =$ 10101 وارد شود رشته خروجی Q کدام یک از موارد ذیل است؟

الف) $Q = 01100$ ب) $Q = 11001$

ج) Q = 10101 د) Q = 01010

۷۷-کدامیک از موارد ذیل غلط است ؟

الف)به‌وسیله یک دیکدر و 3 گیت OR می‌توان 3 تابع منطقی را پیاده‌سازی کرد.

ب)به‌وسیله مالتی پلکستر فقط می‌توان یک تابع را پیاده‌سازی کرد.

ج)یک دیکدر با تواناساز معادل یک دی مالتی پلکستر است.

د)با 8 گیت AND می توان یک انکدر 3×8 ساخت.

۷۸-برای پیاده سازی تابع f(A , B , C) از یک مولتی‌پلکستر 1×4 با دو خط انتخاب S_0 و S_1 و چهار خط ورودی I_0 تا I_3 استفاده می‌شود به طوریکه ورودی B به S_1 و ورودی C به S_0 وصل است و داده‌های ورودی I_0 تا I_3 بدین قرارند:

$I_0 = 1$ و $I_1 = 1$ و $I_2 = A$ و $I_3 = A'$. خروجی این مالتی پلکستر کدامیک از توابع ذیل را پیاده‌سازی می‌کند؟

الف) $f(A , B , C) = \Sigma(1 , 3 , 5 , 6)$

ب) $f(A , B , C) = \Sigma(0 , 2 , 4)$

ج) $f(A , B , C) = \Sigma(3 , 1 , 5)$

د) $f(A , B , C) = \Sigma(0 , 1)$

۷۹-بزرگترین عدد دودویی 16 بیتی در مبنای ده کدام است؟

الف) 2^{16} ب) $2^{16} - 1$

ج) 2^{15} د) $2^{15} - 1$

۸۰- یک مدار ترکیبی دارای سه ورودی ویک خروجی است. خروجی زمانی یک می شود که دو ورودی از سه ورودی یک باشند. تابع خروجی کدامیک از موارد ذیل است؟

الف) $\Sigma(3, 5, 7)$ ب) $\Sigma(3, 5, 6)$

ج) $\Sigma(5, 7)$ د) $\Sigma(3, 6)$

۸۱- کدامیک از موارد ذیل در مورد تولید سیگنالهای زمانی غلط است؟

الف) در شمارنده حلقوی برای تولید 4 سیگنال زمانی به یک شیفت رجیستر 4 بیتی احتیاج است.

ب) در شمارنده دیکدر برای تولید 4 سیگنال زمانی به یک شمارنده 2 بیتی با یک دیکدر 4×2 احتیاج است.

ج) در شمارنده حلقوی برای تولید 8 سیگنال زمانی به یک شیفت رجیستر با 3 بیت احتیاج است.

د) در شمارنده دیکدر برای تولید 8 سیگنال زمانی احتیاج به شمارنده 3 بیتی با یک دیکدر 8×3 است.

۸۲- عدد 1010.011 در مبنای 2 معادل چه عددی در مبنای 10 است؟

الف) 10.375 ب) 10.5

ج) 10.3 د) 6.8

۸۳-کدامیک از روابط ذیل غلط است؟

الف) $x \oplus y' = (x \oplus y)'$ ب) $x' \oplus y = x \oplus y'$

ج) $x' \oplus y' = (x \oplus y)'$ د) $(x' \oplus y) = (x \oplus y)'$

۸٤- کدام نامساوی زیر می‌تواند شرط وجود یک کد برای تصحیح کننده یک بیت غلط در هر حرف دانست (حروفm بیتی و k بیت اضافه به حروف می‌شود).

الف) $m \leq 2^k - k - 1$ ب) $m \leq 2^k + k$

ج) $m - 1 \leq 2^k + k$ د) $m - k \leq 1$

۸۵- جدول شکل روبرو

D_3	D_2	D_1	D_0	A_1	A_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	X	0	1	1
0	1	X	X	1	0	1
1	X	X	X	1	1	1

الف) جدول دیکدر است

ب) جدول اینکدر است با اولویت

ج) جدول دیکدر اولویت‌دار

د) جمع‌کننده دو عدد دو بیتی است

۸٦- جدول شکل مقابل کدام فلیپ فلاپ را نشان می‌دهد.

ورودی X	$Q(t)$	$Q(t + \Delta t)$
0	0	0
1	1	0
0	0	1
1	1	1

الف)RS فلیپ فلاپ ب) JK فلیپ فلاپ

ج) T فلیپ فلاپ د) D فلیپ فلاپ

۸۷–از کدام یک از موارد زیر می‌توان برای نمایش مینترم‌های یک تابع بولی استفاده کرد؟

الف) مالتی پلکستر ب) دیکدر

ج) اینکدر د) مالتی پلکستر ودیکدر

۸۸– با توجه به نحوه کار JK فلیپ فلاپ در جای خالی چه چیزی باید در جدول قرار داد.

J	K	Q (t+Δt)
0	0	Q(t)
0	1	0
1	0	?
1	1	?

الف) به ترتیب 1 و Q ب) به ترتیب 1 و Q′(t)

ج) به ترتیب 0 و Q′(t) د) به ترتیب 1 و نامشخص

۸۹–تریگر نمودن (Triggering) فلیپ فلاپها یعنی:

الف) تغییرات در ورودی ب) تغییرات در خروجی

ج) سنکرون کردن آنها د) همه موارد

۹۰– با توجه به جدول زیر با استفاده از فلیپ فلاپ نوع T تابع ورودی به فلیپ فلاپ را به‌دست آورید .

Q (t)	X (ورودی)	Q (t+1)
0	0	1
0	1	0
1	0	1
1	1	0

الف) $T(Q , X) = \Sigma (2 , 3)$ ب) $T(Q , X) = 1$

ج) $T(Q , X) = \Sigma(0 , 3)$ د) $T(Q , X) = \Sigma(0 , 1 , 3)$

پاسخ نامه

۱-ب	۲-الف	۳-ب	٤-الف
٥-د	٦-ج	۷-د	۸-ج
۹-الف	۱۰-ب	۱۱-د	۱۲-ب
۱۳-ج	۱٤-الف	۱٥-ب	۱٦-د
۱۷-ب	۱۸-ج	۱۹-د	۲۰- الف
۲۱-د	۲۲-الف	۲۳-ج	۲٤-ب
۲٥-الف	۲٦-ج	۲۷-ب	۲۸-الف
۲۹-د	۳۰-ج	۳۱-الف	۳۲-ب
۳۳-الف	۳٤-ج	۳٥-د	۳٦-ج
۳۷-الف	۳۸-د	۳۹-د	٤۰-ب
٤۱-ج	٤۲-د	٤۳-ب	٤٤-الف
٤٥-الف	٤٦-الف	٤۷-ج	٤۸-ب
٤۹-الف	٥۰-د	٥۱-ب	٥۲-الف
٥۳-د	٥٤-د	٥٥-الف	٥٦-الف
٥۷-د	٥۸-الف	٥۹-ج	٦۰-ج
٦۱-الف	٦۲-ج	٦۳-د	٦٤-ج
٦٥-ب	٦٦-الف	٦۷-الف	٦۸-الف
٦۹-الف	۷۰-ب	۷۱-ج	۷۲-د
۷۳-الف	۷٤-الف	۷٥-الف	۷٦-الف
۷۷-د	۷۸-الف	۷۹-ب	۸۰ -ب
۸۱-ج	۸۲-الف	۸۳-ج	۸٤-الف
۸٥-ج	۸٦ -د	۸۷-د	۸۸-د
۸۹-د	۹۰-ج		

سوالات تشریحی

۱- تابع بولی زیر را با یک متمرکز کننده (multiplexer)1× 8 پیاده کنید.

$F(A,B,C,D)= \Sigma(0,1,4,5,7,10,11,15)$

۲- با فلیپ فلاپ های نوع T شمارنده‌ای طراحی کنید تا اعداد بایری000 تا110 را شمارش نموده و دوباره به 000 برگردد.

۳- توابع منطقی f و d به ترتیب بیانگر جمع حاصل ضرب‌ها و حالات بی‌اهمیت یک مدار می‌باشند. مدار را طوری طراحی کنید که:

الف) با حداکثر صرفه‌جویی قابل پیاده‌سازی باشد.

ب) فاقد هرگونه خروجی تصادفی باشد.

$f(A , B , C , D) =\Sigma (1 , 2 , 5 , 6 , 7, 13 , 15)$

$d(A , B , C , D) =\Sigma (0 , 10 ,12)$

٤- مدار تابع $F(w , x, y , z) =\Sigma (1 , 3 , 6 , 7 , 8, 11 , 13 , 15)$ را با استفاده از یک تسهیم کننده طراحی کرده و اتصالات آنرا با رسم ساختار کلی مدار نشان دهید.

۵- فرکانس ورودی یک شمارنده همزمان 3 بیتی 120 مگاهرتز است و کلیه FF ها در حالت سکون قرار دارند.

الف) فرکانس خروجی آنرا حساب کنید.

ب) ساختار کلی شمارنده را با استفاده از D-FF رسم کنید.

ج) این شمارنده اعداد فرد صعودی تک رقمی را تولید می‌کند.

مدار آن را طراحی کنید.

٦- تسهیم کننده تابع زیر را پیاده سازی کرده و بلوک دیاگرام آنرا رسم نمایید.

$F(w , x, y , z) =\Sigma (0 , 3 , 4 , 8 , 9 , 12 , 15)$

۷- فرکانس ورودی یک شمارنده همزمان 3 بیتی 180 مگاهرتز است و کلید FF در حالت سکون می‌باشند.

الف) فرکانس خروجی آنرا به‌دست آورید.

ب) بلوک دیاگرام این شمارنده را به کمک D-FF رسم نمایید.

ج) مدار شمارنده را به گونه‌ای طراحی کنید که اعداد فرد صعودی تک رقمی را تولید کند.

۸- مدار ترکیبی طراحی کنید که یک عدد ۲ بیتی را به توان ۲ برساند.

۹- مدار تولید توازن سه بیتی را طراحی نمائید. (خروجی در صورتی یک شود که تعداد بیت‌های یک زوج باشد).

۱۰- شمارنده‌ای با استفاده از فلیپ فلاپ T طراحی نمائید که سری زیر را بشمارد.

۰→۱→۲→۳→۵→۷→۰

۱۱- تابع زیر را به کمک مالتی پلکسر طراحی کنید.

$f(a , b, c , d) = \Sigma (0 , 1 , 3 , 4 , 8 , 9 , 15)$

۱۲- یک مدار ترکیبی طراحی کنید که چهار ورودی و سه خروجی داشته باشد به طوریکه خروجی نشان دهنده مجموع ارقام ورودی باشد.

راهنمایی: به طور مثال اگر ورودی ۱۰۱۱ باشد خروجی باید ۰۱۱ باشد زیرا مجموع ارقام ورودی ۳ است (۳=۱+۱+۰+۱)

۱۳- بلوک دیاگرامی رسم کنید که نشان دهد چگونه می توان یک دیکدر 4*16 را با استفاده از دو دیکدر 3*8 که دارای ورودی تواناساز هستند؟ (بلوک دیاگرام را با رسم کلیه ورودیها و خروجی‌ها مشخص نمایید.)

۱٤- با استفاده از مولتی‌پلکسر تابع $F(A , B , C) = \prod(0 , 2 , 3 , 6)$ را پیاده سازی نمایید؟

۱۵-فقط با استفاده از تمام جمع کننده‌ها یک مدار ترکیبی طراحی کنید که کد BCD را به کد افزونی 3 معادل آن تبدیل کند.

۱٦- یک مدار ترکیبی طراحی کنید که مساوی بودن دو عدد دو بیتی را چک کند، مدار دارای یک خروجی است به طوریکه اگر دو عدد ورودی برابر باشند خروجی یک است و اگر دو عدد ورودی نا مساوی باشند خروجی صفر است. (ارائه تابع خروجی پس از ساده‌سازی و رسم مدار و ارائه جدول درستی مدار الزامی است)

۱۷- یک مدار ترتیبی با استفاده از دو فلیپ فلاپ T به نام A و B و یک ورودی x طراحی کنید، به طوری‌که وقتی $x = 0$ است حالت مدار ثابت می‌ماند (یعنی در هر حالتی که قرار دارد، همان حالت تکرار می‌شود) و اگر $x = 1$ باشد مدار به ترتیب به حالات 00 و01 و 10 و 11 رفته و این سری تکرار می‌شود.

۱۸- مدار یک شیفت رجیستر 4 بیتی دو جهته با امکان بار شدن موازی را با استفاده از فلیپ فلاپ D بسازید به طوریکه عملکرد آن به صورت زیر باشد. (S_0S_1 خطوط انتخاب هستند)

$S_0S_1 = 00$ بلا تغییر

$S_0S_1 = 01$ شیفت به راست

$S_0S_1 = 10$ شیفت به چپ

$S_0S_1 = 11$ بار کردن موازی

۱۹- یک دیکدر کد سه افزونی (۳- EXCESS) به BCD طراحی کنید.

۲۰- تابع F را $F(A , B , C) = \Sigma(3 , 4 , 6 , 7)$

الف) به‌وسیله جدول کارنو ساده کنید.

ب) به‌وسیله دیکدر پیاده‌سازی نمائید.

ج) به‌وسیله مالتی پلکستر پیاده سازی نمائید.

۲۱- با استفاده از مدارهای ترتیبی طراحی کنید که مسئله زیر را حل نمائید. یک سیستم ترتیبی دارای دو متغیر X_2,X_1 و یک تابع خروجی Z است. اگر X1 حداقل یکبار از صفر به یک و از یک به صفر تغییر نماید و X_2 از صفر به یک و از یک به صفر و دوباره از صفر به یک تغییر نماید در این صورت Z از صفر به یک تبدیل می‌شود اگر بعد از اینکه X_2 از صفر به یک رسید بلافاصله از یک به صفر تغییر نماید آنوقت Z برابر صفر خواهد شد.

۲۲-یک شمارشگر سه بیتی BCD با فلیپ فلاپ j k طراحی کنید.

۲۳- با ثبات شیفت دهنده 4 بیتی مداری طراحی نمائید که به ازاء هر پالس ساعت فقط یکی از خروجی‌های آن یک می‌شود.

واژه نامه

انگلیسی به فارسی

A

ماشین حالت الگوریتمی	algorithmic state machine (ASM)
کد دودویی استاندارد برای کاراکترهای الفبا عددی اسکی	American standard code for information interchange
مدارات مجتمع خاص	application standard IC
کاربرد خاص	application specific
اصل شرکت‌پذیری	associative
غیرهمزمان	asynchronous

B

مدلسازی رفتاری	behavioral modeling
کد دهدهی به دودویی	binary coded decimal
دهدهی کد شده به دودویی	binary coded decimal
عمل بیتی	bitwise
بلوکی	blocking
پایین به بالا	bottom- up

C

متعارف	canonical
بازگشت نورد	carriage return
نقلی	carry
حساس به حالت	case sensitive
واحد پردازش مرکزی	central processing unit
بیت تست	check bit
تراشه	chip
انتخابگر تراشه	chip select
توصیف مدار	circuit description module
ورودی پاک	Clear
مدارهای ترتیبی ساعت دار	clock generator
بسته بودن	closure
کاراکترهای کنترل تبادل اطلاعات	communication control characters
اصل جابجایی	commutative
وسیله منطقی برنامه پذیر پیچیده	complex programmable logic device
نقشه مرکب	composite map
طراحی با کمک کامپیوتر	computer aided design
عملگر ادغام	concatenation
تئوری وفاق	consensus theorem
مدل‌سازی رونده داده	continuous state assignment
پایین شمار	count- down counter
رقابت بحرانی	critical race

سعی و کاهش	cut-and-try
چرخه-سیکل	cycle

D

مسیر داده	data path
دی مولتی‌پلکسر	de multiplexer
رمز گشا - دیکدر	decoder
تاخیر انتشار	delay propagation
دیسک چندکاره دیجیتال	digital versatile disk
تنظیم مستقیم	direct set
باز نشان مستقیم	direct reset
اصل توزیع‌پذیری	distributive low

E

نشانه لبه	edge qualifier
فعال	enable
رمز گذار - انکدر	encoder
نقلی چرخشی	end around carry
نقلی انتهایی	end carry
ختم متن	end of text
موجب های اصلی اساسی	essential prime implicant
کدافزونی-۳	excess-3
جدول تحریک	excitation table

F

گنجایش ورودی	fan in
گنجایش خروجی	fan out
آرایه گیتی برنامه پذیر موردی	field programmable gate array
جداساز فایل	file separator
حالت متناهی	finite state machine
جدول روند	flow table
افکتور های فورمت	format effector
ساعت آزادگرد	free-running clock
جمع‌کننده کامل	full adder
ICسفارشی	full-custom IC
عملیات اساسی	fundamental mode operation

G

گیت	Gate

H

نیم جمع کننده	half adder
زبان توصیف سخت	hardware description language

افزاری	
زبان طراحی سخت افزار	hardware design language
سلسله مراتبی	hierarchical description
امپدانس بالا	high impedance
زمان نگهداری	hold time
جدول بندی افقی	horizontal tabulation

I

عنصر شناسه	identity element
جدول ایجاب	implication table
تابع غیر کامل	incompletely specified function
جداسازی اطلاعات	information separator
ذکر	instantiation
معکوس	inverse

L

فشردگی زیاد	large scale
منطق آرایه ای برنامه پذیر	logic programmable array
پیش بینی نقلی	look a head carry

M

حاکم	Master
فشردگی متوسط	medium scale
ادغام	merging
علائم مخلوط	mixed notation
ماژول– قطعه نرم‌افزاری یا سخت‌افزاری	module
مدولوس: عملی ریاضی که نتیجه آن باقیمانده یک تقسیم است	Modulus
مولتی‌پلکسر	multiplexer

N

تصدیق منفی	negative acknowledge
حد پارازیت	noise margin
رقابت غیر بحرانی	non critical race
غیربلوکی	non-blocking
غیر متصل	not connected

O

سرریز	overflow

P

توازن	parity
تست توازن	parity check
توان مصرفی	power dissipation
پیش تنظیم	preset
موجب های اصلی	prime implicant
گیت اصلی	primitive gate
تخصیص اجرایی	procedural state assignment
پردازشگر	processor
آرایه منطقی برنامه پذیر	programmable logic array
وسیله منطقی برنامه پذیر	programmable logic device

R

وضعیت رقابتی	race condition
ممیز	radix
حافظه با دستیابی تصادفی	random access memory
حافظه فقط خواندنی	read only memory
جداساز رکورد	record separator
سطح انتقال ثباتی	register level transfer
سطح انتقال بین ثباتی	register transfer level
باز نشانی	Reset
آگاه گر آدرس سطر	row address strobe

S

نمودار تصویری	Schematic
ضبط تصویری	schematic capture
انتخابگر	selector
کدهای خود متمم	self complement
ثبات توالی	sequence register
وسیله منطقی برنامه پذیر ترتیبی	Sequential (or simple) programmable logic device
نشاندن	set
زمان نشست	settling time
زمان برپایی	setup time
متممعلامت‌دارمنفی	signed complement system
سیستم مقدارعلامت‌دارمنفی	signed-magnitude system
شبیه سازی	simulation
تابع	slave
فشردگی کم	small scale integration
شروع متن	start of text
حالت	state
ماژول محرک	stimulus module
ظرفیت پارازیتی	stray capacitance

مدل‌سازی ساخت یافته	structural modeling	
توصیف ساختاری	structural description	
مدل‌سازی سطح سوئیچ	switch-level modeling	
شمارنده حلقوی دنباله چرخان	switch-tail ring counter	
همگام سازی	synchronization	
همزمان	synchronous	
سنتز – ترکیب - ادغام	synthesis	

T

خروجی مقصد یا هدف	target output	
تکلیف	task	
برنامه تست	test bench	
واحدهای زمانی	time units	
دگر وضع	toggle	
بالا به پایین	top-down	
حالت کلی	total state	
جدول گذر	transition table	
شفاف	transparent	
سه حالته	tri-state(three-state)	

U

مواردی که به وسیله کاربر تعریف می شود	user defined primitives	
تعریف شده به وسیله کاربر	user-defined	

V

معتبر	Valid	
بردار	Vector	
مختص فروشنده	vendor-specific	
فشردگی خیلی زیاد	very large scale	

W

سیمی	Wire	
بی سیم	wire Less	

واژه نامه

فارسی به انگلیسی

الف

آرایه گیتی برنامه پذیر موردی — field programmable gate array

آرایه منطقی برنامه پذیر — programmable logic array

آگاه گر آدرس سطر — row address strobe

آی سی سفارشی — full-custom IC

ادغام — merging

اصل توزیع‌پذیری — distributive low

اصل جابجایی — commutative

اصل شرکت‌پذیری — associative

افکتور های فورمت — format effector

امپدانس بالا — high impedance

انتخابگر — selector

انتخابگر تراشه — chip select

ب

باز نشان مستقیم — direct reset

باز نشانی — reset

بازگشت نورد — carriage return

بالا به پایین — top-down

بردار — vector

برنامه تست — test bench

بسته بودن — closure

بلوکی — blocking

بی سیم — wire Less

بیت تست — check bit

پ

پایین به بالا — bottom- up

پایین شمار — count- down counter

پردازشگر — processor

پیش بینی نقلی — look a head carry

پیش تنظیم — preset

ت

تئوری وفاق — consensus theorem

تابع — slave

تابع غیر کامل — incompletely specified function

تاخیر انتشار — delay propagation

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

توان مصرفی — power dissipation

توصیف ساختاری	**structural description**
توصیف مدار	**circuit description**
	module

ث

ثبات	**register**
ثبات توالی	**sequence register**

ج

جداساز رکورد	**record separator**
جداساز فایل	**file separator**
جداسازی اطلاعات	**information separator**
جدول ایجاب	**implication table**
جدول بندی افقی	**horizontal tabulation**
جدول تحریک	**excitation table**
جدول روند	**flow table**
جدول گذر	**transition table**
جمع‌کننده کامل	**full adder**

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 حالت	**case sensitive**

خ

ختم متن	**end of text**
خروجی مقصد یا هدف	**target output**

د

دگر وضع	**toggle**
دهدهی کد شده به دودویی	**binary coded decimal**
دی مولتی‌پلکسر	**de multiplexer**
دیسک چندکاره دیجیتال	**digital versatile disk**

ذ

ذکر کردن – توضیح دادن – نمایش با نمونه	**instantiation**

ر

رقابت بحرانی	**critical race**
رقابت غیر بحرانی	**non critical race**
رمز گذار - انکدر	**encoder**
رمز گشا - دیکدر	**decoder**

ز

زبان توصیف سخت افزاری	**hardware description language**

زبان طراحی سخت افزار	**hardware design language**
زمان برپایی	**setup time**
زمان نشست	**settling time**
زمان نگهداری	**hold time**
س	
ساعت آزادگرد	**free-running clock**
سرریز	**overflow**
سطح انتقال بین ثباتی	**register transfer level**
سطح انتقال ثباتی	**register level transfer**
سعی و کاهش	**cut-and-try**
سلسله مراتبی	**hierarchical description**
سنتز – ترکیب – ادغام	**synthesis**
سه حالته	**tri-state(three-state)**
سیستم مقدارعلامت‌دارمنفی	**signed-magnitude system**
سیمی	**wire**
ش	
شبیه سازی	**simulation**
شروع متن	**start of text**
شفاف	**transparent**
شمارنده حلقوی دنباله چرخان	**switch-tail ring counter**
ض	
ضبط تصویری	**schematic capture**
ط	
طراحی	**design**
طراحی با کمک کامپیوتر	**computer aided design**
ظ	
ظرفیت پارازیتی	**stray capacitance**
ع	
علائم مخلوط	**mixed notation**
عمل بیتی	**bitwise**
عملگر ادغام	**concatenation**
عملیات اساسی	**fundamental mode operation**
عنصر شناسه	**identity element**
غ	
غیر متصل	**not connected**
غیربلوکی	**non-blocking**
غیرهمزمان	**asynchronous**
ف	
فشردگی متوسط	**medium scale**
فشردگی خیلی زیاد	**very large scale**
فشردگی زیاد	**large scale**
فشردگی کم	**small scale integration**
فعال	**enable**
ک	

کاراکترهای کنترل تبادل اطلاعات	**communication control characters**
کاربرد خاص	**application specific**
کد دودویی استاندارد برای کاراکترهای الفبا عددی اسکی	**american standard code for information interchange**
کد دهدهی به دودویی	**binary coded decimal**
کدافزونی-۳	**excess_3**
کدهای خود متمم	**self complement**
گ	
گیت	**gate**
گنجایش ورودی	**fan in**
گنجایش خروجی	**fan out**
گیت اصلی	**primitive gate**
م	
ماژول – قطعه نرم افزاری یا سخت افزاری	**module**
ماژول محرک	**stimulus module**
ماشین حالت الگوریتمی	**algorithmic state machine (ASM)**
متعارف	**canonical**
متممعلامت‌دارمنفی	**signed complement system**
مختص فروشنده	**vendor-specific**
مدارات مجتمع خاص	**application standard IC**
مدارهای ترتیبی ساعت دار	**clock generator**
مدل‌سازی رونده داده	**continuous state assignment**
مدل‌سازی ساخت یافته	**structural modeling**
مدل‌سازی سطح سوئیچ	**switch-level modeling**
مدلسازی رفتاری	**behavioral modeling**
مدولوس : عملی ریاضی که نتیجه آن باقیمانده یک تقسیم است	**modulus**
مسیر داده	**data path**
معتبر	**valid**
معکوس	**inverse**
ممیز	**radix**
منطق آرایه ای برنامه پذیر	**logic programmable array**
مواردی که به وسیله کاربر تعریف می شود	**user defined primitives**
موجب های اصلی	**prime implicant**
موجب های اصلی اساسی	**essential prime implicant**
مولتی‌پلکسر	**multiplexer**
ن	
نشاندن	**set**
نشانه لبه	**edge qualifier**
نقشه مرکب	**composite 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 برنامه پذیر	**programmable logic device**
وسیله منطقی برنامه پذیر پیچیده	**complex programmable logic device**
وسیله منطقی برنامه پذیر ترتیبی	**Sequential (or simple) programmable logic device**
وضعیت رقابتی	**race condition**
ه	
همگام سازی	**synchronization**
همزمان	**synchronous**

لیست منابع و مراجع

۱.	Logic and Computer Design Fundamentals _ 2nd Ed Morris M . Mano	Prentice Hall 2000	
۲.	Digital Logic Circuit Analysis and Design Nelson _ Nagle _ Irwin _ Carroll	Prentice Hall 1995	
۳.	Introduction to Logic Design Hayes	Addison_wesley 1993	
٤.	Digital Design : Principles and Practices _ 3rd Ed Wakerly	Prentice Hall 2000	
٥.	Contemporary Logic Design Katz	Prentice Hall 1994	
٦.	Logic Design of Digital Systems _ 3rd Ed	Allyn Bacon	

Dietmeyer 1988

٧. Principles of Digital Design Prentice Hall

Gajski 1997

٨. Fundamentals of Logic Design _ 4th Ed West

Roth 1992

٩. **ویرایش سوم _ طراحی دیجیتال** **انتشارات خراسان**

دکتر قدرت سپید نام **١٣٨٤**