

Invert-AND (ب)　　OR-invert (الف)

شکل ۲۵-۳. دو سمبل گرافیکی برای گیت NOR

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

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

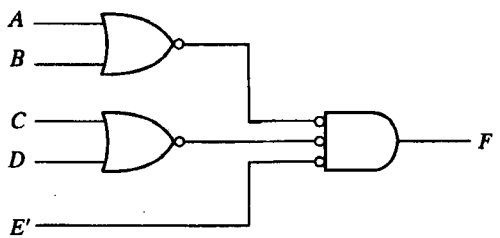
الگوی OR–AND را به سادگی با حذف حباب‌ها در طول هر مسیر می‌توان شناسایی کرد. متغیر E برای جبران سومین حباب در ورودی گیت سطح دوم متمم شده است.

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

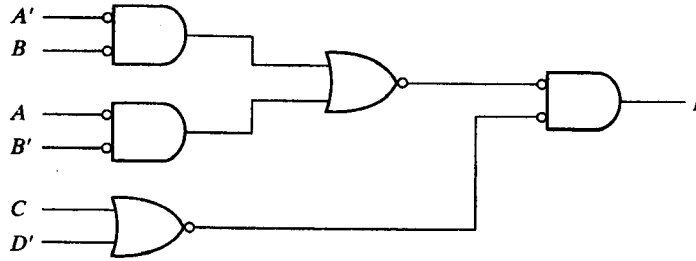
تبدیل نمودار AND-OR شکل ۲۳-۳(الف) به یک نمودار NOR در شکل ۲۷-۳ نشان داده شده است. تابع بول برای این مدار به شکل زیر است

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

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



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



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

۳-۷ دیگر پیاده‌سازی‌های دو سطحی

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

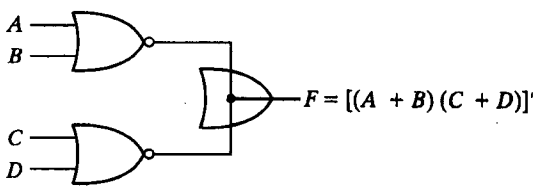
$$F = (AB)' \cdot (CD)' = (AB + CD)'$$

و به آن تابع AND–OR–INVERT می‌گویند.

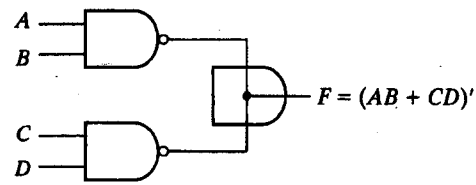
به طور مشابه خروجی NOR گیت‌های ECL برای اجرای یک تابع OR سیمی به هم گره زده می‌شوند. تابع منطقی پیاده‌سازی شده با مدار شکل ۲۸-۳ (ب) چنین است:

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

و به آن تابع OR–AND–INVERT می‌گویند.



(ب) OR سیمی در گیت‌های ECL



(الف) AND سیمی در گیت‌های TTL NAND کلکتور باز

شکل ۲۸-۳. منطق سیمی

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

فرم‌های مفید گیت‌ها

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

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

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

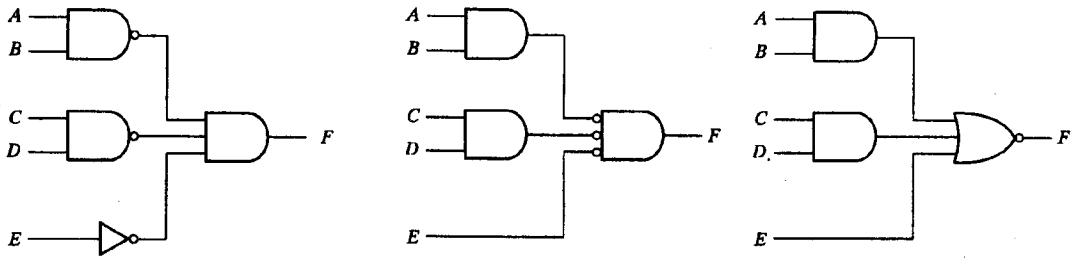
فرم‌های AND–OR و OR–AND، فرم‌های دو سطح اصلی بحث شده در بخش ۴-۳ می‌باشند. فرم‌های NAND–NAND و NOR–NOR در بخش ۶-۳ ارائه شدند. چهار فرم باقیمانده نیز در این بخش بررسی می‌شوند.

پیاده‌سازی AND–OR–INVERT

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

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

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



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

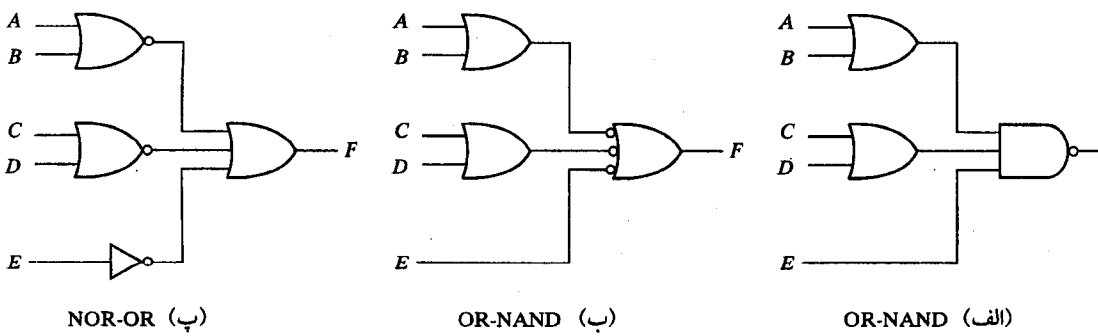
اول انتقال می‌دهیم. برای جبران هر حباب یک وارونگر در ازاء هر متغیر لازم است. به همین ترتیب می‌توان وارونگر را حذف کرد به شرطی که E متمم شود. مدار شکل ۲۹-۳ (پ)، فرم NAND–AND است و برای پیاده‌سازی تابع AND–OR–INVERT در شکل ۲۸-۳ نشان داده شده است.

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

پیاده‌سازی OR–AND–INVERT

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

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



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

جدول ۳-۳. پیاده‌سازی با دیگر فرم‌های دو طبقه

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

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

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

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

خلاصه جدول وار و مثال

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

مثال ۱۱-۳

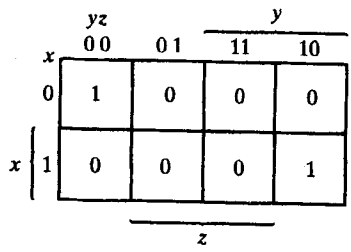
تابع شکل ۳۱-۳ (الف) را به فرم دو سطحی لیست شده در جدول ۳-۳ پیاده‌سازی کنید.

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

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

خروجی نرمال این تابع می‌تواند به صورت زیر بیان شود.

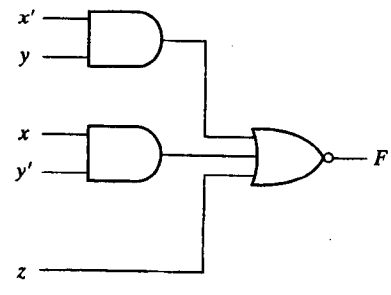
$$F = (x'y + xy' + z)'$$



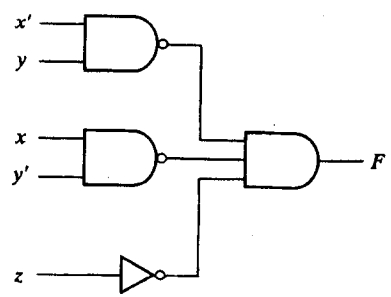
$F = x'y'z' + xyz'$

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

(الف) ساده سازی نقشه در جمع حاصلضرب ها

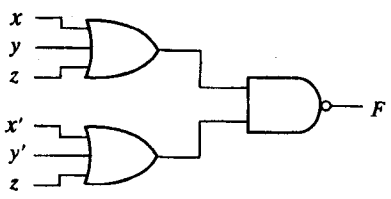


AND-NOR

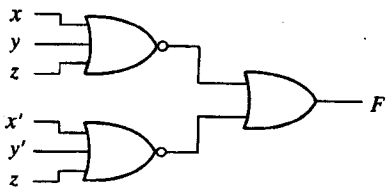


NAND-AND

$F = (x'y + xy' + z)'$ (ب)



OR-NAND



NOR-OR

$F = [(x + y + z)(x' + y' + z)]'$ (پ)

شکل ۳۱-۳. دیگر پیاده‌سازی‌های دو طبقه

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

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

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

آنگاه متمم تابع را بدست می‌آوریم.

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

خروجی نرمال F اکنون به فرم زیر نوشته می‌شود.

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

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

❑

۳-۸ تابع OR انحصاری

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

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

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

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

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

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

روابط زیر در مورد OR انحصاری معتبرند

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

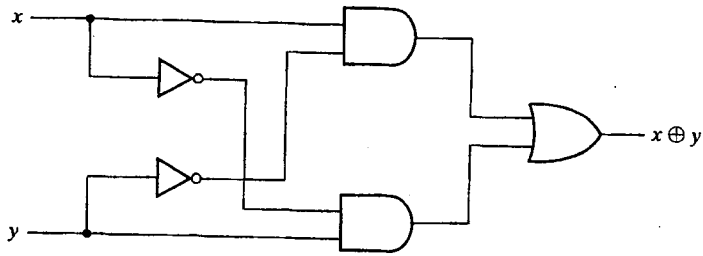
هر یک از این روابط می‌تواند با به کارگیری جدول درستی و جایگزینی ⊕ با عبارت بولی هم‌ارزی ثابت شود. و نیز می‌توان نشان داد که عمل OR انحصاری خاصیت جابجایی و شرکت‌پذیری را دارد. یعنی:

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

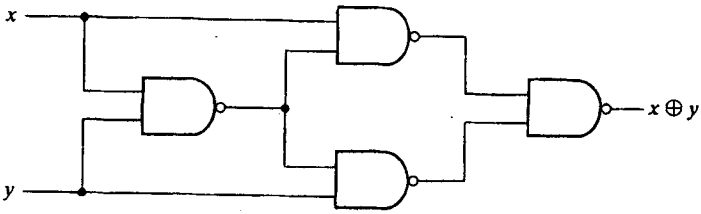
و

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

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



(الف) با گیت‌های AND-OR-NOT



(ب) با گیت‌های NAND

شکل ۳۲-۳. پیاده‌سازی XOR

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

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

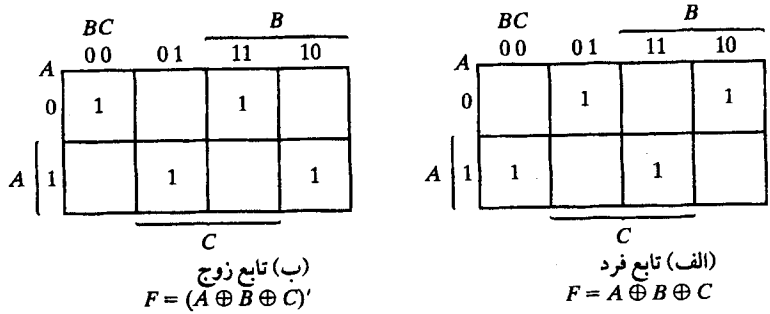
تنها توابع بولی محدودی برحسب OR انحصاری بیان می‌شوند. با این وجود این تابع به کرّات ضمن طراحی سیستم‌های دیجیتال به کار گرفته می‌شود. خصوصاً در عملیات حسابی و خطایابی و تصحیح خطا بسیار مفید است.

تابع فرد

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

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

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



شکل ۳۳-۳. نقشه تابع XOR سه متغیر

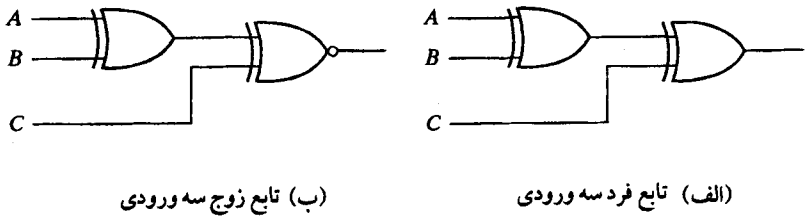
یک متغیر 1 باشد و یا هر سه متغیر برابر 1 باشند. برخلاف حالت دو متغیره، که فقط یک متغیر باید برابر 1 می‌بود، در حالت سه یا چند متغیر، نیاز این است که تعداد فردی از متغیرها برابر 1 باشند. در نتیجه عمل XOR چند متغیره را تابع فرد می‌خوانند.

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

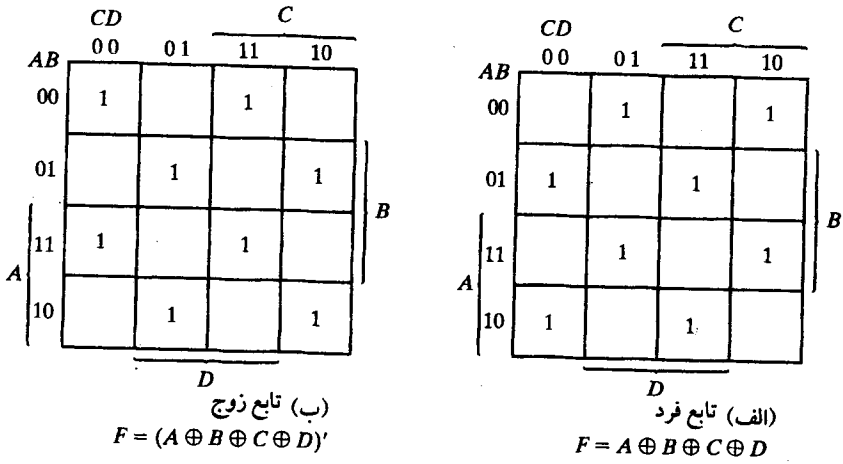
تعریف یک تابع فرد را با ترسیم آن در یک نقشه شفاف‌تر می‌کنیم. شکل ۳۳-۳ (الف) نقشه را برای تابع XOR سه متغیره نشان می‌دهد. چهار مینترم تابع یک مربع در میان با هم فاصله دارند. تابع فرد از چهار مینترمی که مقادیر دودویی‌اش تعداد فردی 1 دارند شناسایی می‌شود. متمم یک تابع فرد، یک تابع زوج است. طبق شکل ۳۳-۳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

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

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



شکل ۳۴-۳. نمودار منطقی توابع فرد و زوج



شکل ۳۵-۳. نقشه برای تابع XOR چهار متغیره

این تابع را بدست آوریم.

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

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

تولید و چک توازن

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

جدول ۴-۳. جدول درستی مولد توازن زوج

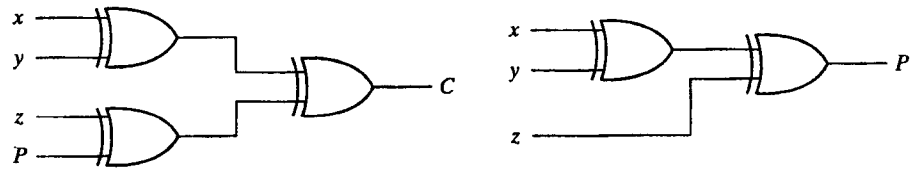
پیام سه بیتی			بیت توازن
x	**y**	**z**	**P**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

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

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

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

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



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

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

جدول ۵-۳. جدول درستی چک کننده توازن زوج

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

چک کننده توازن را با گیت‌های XOR پیاده‌سازی کرد:

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

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

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

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

۳-۹ زبان توصیف سخت‌افزاری (HDL)

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

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

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

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

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

نمایش مدول

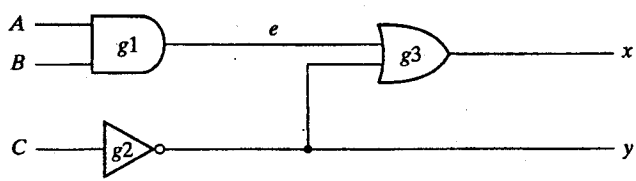
Verilog HDL دارای دستور زبانی است که دقیقاً ساختارهای مُجازی که در زبان می‌توانند به کار روند را توصیف می‌نماید. خصوصاً، Verilog حدود 100 کلمه کلیدی از پیش‌تعریف شده، حروف

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

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

مثال ۱-۳، HDL

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



شکل ۳۷-۳. مداری برای نمایش HDL

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

تأخیر در گیت‌ها

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

`timescale 1ns/100ps

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

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

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

مثال ۲-۳، HDL

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

جدول ۶-۳. خروجی گیت‌ها پس از تأخیر

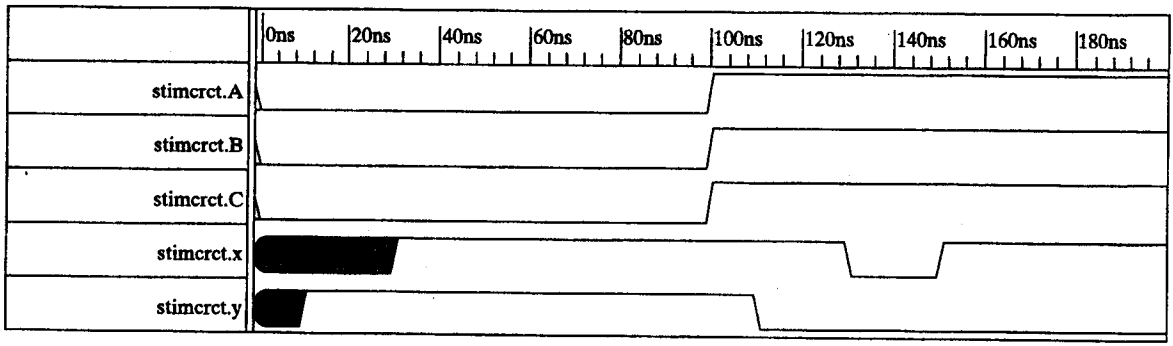
	واحد زمانی	ورودی	خروجی
	(ns)	ABC	y e x
تغییر اولیه	–	000	1 0 1
	–	111	1 0 1
	10	111	0 0 1
	20	111	0 0 1
	30	111	0 1 0
	40	111	0 1 0
	50	111	0 1 1

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

مثال ۳-۳، HDL

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

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



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

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

عبارات بولی

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

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

مثال ۴-۳ HDL توصیف مداری که با دو عبارت بولی زیر بیان شده را نشان می‌دهد:

$$x = A + BC + B'D$$

$$y = B'C + BC'D'$$

مدار دارای دو خروجی x و y و چهار ورودی A، B، C و D است. دو عبارت assign معادلات بول را توصیف می‌نمایند.

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

مثال ۴-۳، HDL

```
//Circuit specified with Boolean expressions
module circuit_bln (x,y,A,B,C,D);
   input A,B,C,D;
   output x,y;
   assign x = A | (B & C) | (~B & D);
   assign y = (~B & C) | (B & ~C & ~D);
endmodule
```

مواردی که به وسیله کاربر تعریف می‌شود (UDP)

گیت‌های به کار رفته در توصیف‌های HDL ، با لغات کلیدی and ، or و غیره به وسیله سیستم تعریف می‌شوند و primitives سیستم نام‌گذاری می‌گردند. کاربر می‌تواند primitive های دیگری را با تعریف آنها به صورت جدول اضافه نماید. این نوع مدارها را تعریف شده بوسیله کاربر یا user–defind می‌نامند. یکی از راه‌های معرفی مدار به فرم جدول، معرفی آن با جدول درستی است. توصیف‌های UDP از کلمه کلیدی module استفاده نمی‌کنند. در عوض با کلمه کلیدی primitive (اصلی) تعریف می‌شوند. بهترین راه معرفی primitive ارائه یک مثال می‌باشد.

مثال ۵-۳ HDL یک UDP را با یک جدول درستی تعریف می‌کند. حل آن براساس قواعد کلی زیر است:

مثال ۵-۳، HDL

```
//User defined primitive(UDP)
primitive crctp (x,A,B,C);
   output x;
   input A,B,C;
//Truth table for x(A,B,C) = Σ(0,2,4,6,7)
   table
//      A   B   C  :  x   (Note that this is only a comment)
        0   0   0  :  1;
        0   0   1  :  0;
        0   1   0  :  1;
        0   1   1  :  0;
        1   0   0  :  1;
        1   0   1  :  0;
        1   1   0  :  1;
        1   1   1  :  1;
   endtable
endprimitive

//Instantiate primitive
module declare_crctp;
  reg x,y,z;
  wire w;
  crctp (w,x,y,z);
endmodule
```

- از کلمه کلیدی primitive استفاده شده و به دنبال آن یک نام و لیست پورت‌ها آورده می‌شود.
- تنها یک خروجی می‌تواند وجود داشته باشد که با بکارگیری کلمه کلیدی output و قبل از همه در لیست پورت اعلام می‌شود.
- به هر تعداد ورودی (input) می‌تواند تعریف شود. ترتیب معرفی آنها با اعلام input با ترتیب مقادیرشان در جدولی که به دنبال می‌آید، باید همخوانی داشته باشد.
- جدول درستی باید در داخل کلمات کلیدی table و endtable محصور شود.
- مقادیر ورودی با (:) پایان می‌یابند. خروجی همواره آخرین وارده در هر سطر است و بعد از آن (;) می‌آید.
- در پایان endprimitive ذکر می‌شود.

توجه کنید که متغیرهای لیست شده در بالای جدول بخشی از توضیحات بوده و به منظور آشنایی ذکر شده‌اند. سیستم متغیرها را به ترتیبی که در بخش ورودی ذکر شده‌اند تشخیص می‌دهد. یک UDP نیز مثل primitive سیستم به کار گرفته می‌شود. مثلاً

```
crctp (w,x,y,z)
```

مداری با تابع

$$w(x, y, z) = \Sigma(0, 2, 4, 6, 7)$$

و ورودی‌های x و y و z و خروجی w را پیاده می‌کند.

گرچه Verilog HDL این نوع توصیف را فقط برای UDP به کار می‌برد. دیگر HDL ها و سیستم‌های طراحی کامپیوتری (CAD) روال‌های دیگری را برای مشخص کردن مدارهای دیجیتال به صورت جدول استفاده می‌کنند. جداول می‌توانند با نرم‌افزار CAD برای بدست آوردن یک ساختار گیتی بهینه پردازش شوند.

در این بخش، HDL را معرفی و مثال‌های ساده‌ای از مدل‌سازی ساخت یافته را ارائه دادیم. مشروح مطالب فوق را برای Verilog HDL می‌توان در فصل بعدی یافت. خوانندگانی که با مدارهای ترکیبی آشنا هستند می‌توانند مستقیماً به بخش ۱۱-۴ مراجعه کرده و این موضوع را ادامه دهند.

مسائل

۳-۱ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

(الف) $F(x, y, z) = \Sigma(0, 2, 6, 7)$ (ب) $F(A, B, C) = \Sigma(0, 2, 3, 4, 6)$

(پ) $F(a, b, c) = \Sigma(0, 1, 2, 3, 7)$ (ت) $F(x, y, z) = \Sigma S(3, 5, 6, 7)$

۳-۲ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده کنید.

(الف) $F(x, y, z) = \Sigma(0, 1, 5, 7)$ (ب) $F(x, y, z) = \tilde{\Sigma}(1, 2, 3, 6, 7)$

۳-۳ توابع بولی زیر را با استفاده از نقشه سه متغیره ساده نمایید.

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

(پ) $A'B + BC' + B'C'$

۳-۴ توابع بولی زیر را با استفاده از نقشه‌های چند متغیره ساده کنید.

(الف) $F(x, y, z) = \sum (2, 3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(4, 6, 7, 15)$

(پ) $F(A, B, C, D) = \sum (3, 7, 11, 13, 14, 15)$

(ت) $F(w, x, y, z) = \sum(2, 3, 12, 13, 14, 15)$

۳-۵ با استفاده از نقشه چهار متغیره توابع زیر را ساده نمایید.

(الف) $F(w, x, y, z) = \sum(1, 4, 5, 6, 12, 14, 15)$

(ب) $F(A, B, C, D) = \sum(0, 1, 2, 4, 5, 7, 11, 15)$

(پ) $F(w, x, y, z) = \sum(2, 3, 10, 11, 12, 13, 14, 15)$

(ت) $F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

۳-۶ با استفاده از نقشه‌های چهار متغیره توابع زیر را ساده کنید.

(الف) $A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$

(ب) $x'z + w'xy' + w(x'y + xy')$

۳-۷ عبارات بولی زیر را با نقشه‌های چهار متغیره ساده کنید.

(الف) $w'z + xz + x'y + wx'z$

(ب) $B'D + A'BC' + AB'C + ABC'$

(پ) $AB'C + B'C'D' + BCD + ACD' + A'B'C + A'BC'D$

(ت) $wxy + yz + xy'z + x'y$

۳-۸ با رسم هر تابع در یک نقشه کارنو، مینترم‌های عبارات بولی زیر را بدست آورید.

(الف) $xy + yz + xy'z$

(ب) $C'D + ABC' + ABD' + A'B'D$

(پ) $wxy + x'z' + w'xz$

۳-۹ موجب‌های اصلی عبارات بولی زیر را بدست آورید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۰ با یافتن موجب‌های اصلی اساسی توابع بول زیر را ساده کنید.

(الف) $F(w, x, y, z) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15)$

(ب) $F(A, B, C, D) = \sum(0, 2, 3, 5, 7, 8, 10, 11, 14, 15)$

(پ) $F(A, B, C, D) = \sum(1, 3, 4, 5, 10, 11, 12, 13, 14, 15)$

۳-۱۱ توابع بولی زیر را با نقشه‌های پنج متغیره ساده کنید.

(الف) $F(A, B, C, D, E) = \sum(0, 1, 4, 5, 16, 17, 21, 25, 29)$

(ب) $F = A'B'CE' + A'B'C'D' + B'D'E' + B'CD' + CDE' + BDE'$

۳-۱۲ توابع بولی زیر را به صورت ضرب حاصل جمع‌ها ساده کنید.

(الف) $F(w, x, y, z) = \sum (0, 2, 5, 6, 7, 8, 10)$ (ب) $F(A, B, C, D) = \prod(1, 3, 5, 7, 13, 15)$

۳-۱۳ عبارات بولی زیر را به صورت (۱) جمع حاصلضرب‌ها و (۲) ضرب حاصل جمع‌ها ساده کنید.

(الف) $x'z' + y'z' + yz' + xy$

(ب) $AC' + B'D + A'CD + ABCD$

(پ) $(A' + B' + D')(A + B' + C')(A' + B + D')(B + C' + D')$

۳-۱۴ به سه طریق تابع بولی زیر را با هشت لیترال یا کمتر نشان دهید:

$$F = A'B'D' + AB'CD' + A'BD + ABC'D$$

۳-۱۵ تابع بولی F زیر را با حالات بی‌اهمیت d ساده کرده و سپس تابع ساده شده را به صورت جمع مینترم‌ها در آورید:

(الف) $F(x, y, z) = \sum (0, 1, 2, 4, 5)$, $d(x, y, z) = \sum (3, 6, 7)$

(ب) $F(A, B, C, D) = \sum(0, 6, 8, 13, 14)$, $d(A, B, C, D) = \sum(2, 4, 10)$

(پ) $F(A, B, C, D) = \sum(1, 3, 5, 7, 9, 15)$, $d(A, B, C, D) = \sum(4, 6, 12, 13)$

۳-۱۶ عبارات زیر را ساده کنید و آنها را با مدارات دو طبقه گیت NAND پیاده‌سازی نمایید:

(الف) $AB' + ABD + ABD' + A'C'D' + A'BC'$

(ب) $BD + BCD' + AB'C'D'$

۳-۱۷ یک نمودار گیت NAND رسم کنید به نحوی که تابع زیر را پیاده‌سازی کرده آن را متمم نماید:

$$F(A, B, C, D) = \sum(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۱۸ یک نمودار منطقی با استفاده از گیت‌های NAND دو ورودی برای پیاده‌سازی عبارت زیر رسم کنید.

$$(AB + A'B')(CD' + C'D)$$

۳-۱۹ توابع زیر را ساده کنید و آنها را با مدارهای دو طبقه گیت NOR پیاده نمایید:

(الف) $F = wx' + y'z' + w'yz'$ (ب) $F(w, x, y, z) = \sum(5, 6, 9, 10)$

۳-۲۰ یک مدار NAND چند طبقه برای عبارت زیر رسم کنید.

$$(AB' + CD')E + BC(A + B)$$

۳-۲۱ یک مدار چند طبقه NOR برای عبارت زیر رسم کنید.

$$w(x + y + z) + xyz$$

۳-۲۲ نمودار مدار شکل ۴-۴ را به مدار چند طبقه NAND تبدیل نمایید.

۳-۲۳ تابع بول F را همراه با حالات بی‌اهمیت d با کمتر از دو گیت NOR پیاده‌سازی نمایید.

$$F(A, B, C, D) = \sum(0, 1, 2, 9, 11)$$
$$d(A, B, C, D) = \sum(8, 10, 14, 15)$$

فرض کنید که هر دو نوع ورودی معمولی و متمم موجودند.

۳-۲۴ تابع بول F را با فرم‌های دو طبقه (الف) NAND–AND، (ب) AND–NOR، (پ) OR–NAND و (ت) NOR–OR پیاده‌سازی کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 2, 3, 4, 8, 9, 12)$$

۳-۲۵ هشت فرم دو طبقه غیرمفید (زاید) نمایش تابع را لیست کنید و نشان دهید که به یک عمل کاهش می‌یابند. نشان دهید که این فرم‌ها چگونه برای گسترش تعداد ورودی‌ها به کار می‌روند.

۳-۲۶ با استفاده از نقشه‌ها، ساده‌ترین فرم جمع حاصلضرب تابع F = fg را بیابید، که در آن f و g برابرند با

$$f = wxy' + y'z + w'yz' + x'yz'$$

و

$$g = (w + x + y' + z')(x' + y' + z)(w' + y + z')$$

۳-۲۷ نشان دهید که دوگان XOR، متمم آن هم هست.

۳-۲۸ با توازن فرد، مدارهای مولد توازن سه بیتی و چک‌کننده توازن چهار بیتی را بدست آورید.

۳-۲۹ عبارات بولی زیر را با نیم جمع‌کننده پیاده‌سازی کنید.

$$D = A \oplus B \oplus C$$
$$E = A'BC + AB'C$$
$$F = ABC' + (A' + B')C$$
$$G = ABC$$

۳-۳۰ عبارت بولی زیر را با گیت‌های XOR و AND پیاده کنید.

$$F = AB'CD' + A'BCD' + AB'C'D + A'BC'D$$

۳-۳۱ توصیف ساختار گیتی HDL مدار شکل ۳-۲۲ (الف) را بنویسید.

۳-۳۲ مدار XOR شکل ۳-۲۲ (الف) دارای گیت‌هایی با تأخیر 10ns برای هر وارونگر، 20ns تأخیر برای هر گیت AND و 30ns برای هر گیت OR است. ورودی مدار از xy = 00 به xy = 01 می‌رود.
(الف) سیگنال‌ها را در خروجی هر گیت از t = 0 تا t = 50ns معین کنید.
(ب) توصیف HDL را برای مدار با در نظر گرفتن تأخیرها بنویسید.
(پ) یک مدول محرک (مشابه مثال ۳-۳ HDL) بنویسید و مدار را برای تأیید پاسخ بخش (الف) شبیه‌سازی کنید.

۳-۳۳ توصیف HDL مدار شکل ۳-۳۷ را با دو عبارت بنویسید.

۳-۳۴ توصیف HDL مداری را که با توابع بولی زیر مشخص شده بنویسید. از عبارات تخصیص یافته پیوسته استفاده نمایید.

$$x = A(CD + B) + BC'$$
$$y = (AB' + A'B)(C + D')$$
$$z = [(A + B)(C' + D'B)]'$$

۳-۳۵ خطاهای نحوی را در اعلان‌های زیر بیابید (توجه کنید که نام گیت‌های primitive اختیاری‌اند):

```
module Exmpl-3(A,B,C,D,F)
inputs A,B,C,
Output D,F;
and g1(A,B,D);
not (D,B,A);
OR (F,B,C);
endmodule;
```

۳-۳۶ نمودار منطقی مدار دیجیتالی که با توصیف HDL زیر بیان شده را رسم کنید.

```
module circt (A,B,C,D,F);
input A,B,C,D;
output F;
wire w,x,y,z,a,d;
and (x,B,C,d);
and (y,a,C);
and (w,z,B);
or (z,y,A);
or (F,x,w);
not (a,A);
not (d,D);
endmodule
```

۳-۳۷ یک تابع منطقی اکثریت، تابعی است که اگر اکثریت متغیرها 1 باشند برابر 1 می‌شود در غیر این صورت 0 است. UDP را برای تابع اکثریت 3 بیتی بنویسید.

مراجع

1. BHASKER, J. 1997. *A Verilog HDL Primer.* Allentown, PA: Star Galaxy Press.
2. HILL, F. J., and G. R. PETERSON. 1981. *Introduction to Switching Theory and Logical Design,* 3rd ed. New York: John Wiley.
3. *IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language* (IEEB Std 1364–1995). 1995. New York: The Institute of Electrical and Electronics Engineers.
4. KARNAUGH, M. A Map Method for Synthesis of Combinational Logic Circuits. *Transactions of AIEE, Communication and Electronics,* 72, part I (Nov. 1953): 593–99.
5. KOHAVI, Z. 1978. *Switching and Automata Theory,* 2nd ed. New York: McGraw-Hill.
6. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.
7. MCCLUSKEY, E. J. 1986. *Logic Design Principles.* Englewood Cliffs, NJ: Prentice-Hall.
8. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis.* SunSoft Press (A Prentice Hall Title).

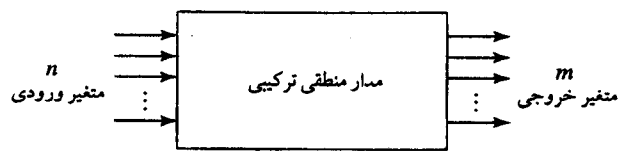


منطق ترکیبی

۴-۱ مدارهای ترکیبی

مدارهای منطقی در سیستم‌های دیجیتال می‌توانند از نوع ترکیبی و یا ترتیبی باشند. یک مدار ترکیبی متشکل از تعدادی گیت منطقی است که خروجی آنها در هر لحظه از زمان مستقیماً به وسیله ورودی‌های همان لحظه معین می‌شود و به ورودی‌های قبلی بستگی ندارد. این نوع مدار، پردازشی را انجام می‌دهد که با مجموعه‌ای از توابع بولی مشخص می‌گردد. مدارهای ترتیبی علاوه بر گیت‌های منطقی از عناصر حافظه نیز استفاده می‌کنند. خروجی‌های آنها تابعی از ورودی‌ها و حالت عناصر حافظه است. در نتیجه خروجی یک مدار ترتیبی نه تنها به مقادیر فعلی ورودی‌ها بلکه به ورودی‌های قبلی وابسته بوده و عملکرد مدار باید به وسیله حالات داخلی و ترتیب زمانی ورودی‌ها مشخص گردد. مدارهای ترتیبی در فصل‌های ۵ و ۹ بحث شده‌اند.

یک مدار ترکیبی از متغیرهای ورودی، گیت‌های منطقی، و متغیرهای خروجی تشکیل شده است. گیت‌های منطقی سیگنال‌هایی را از ورودی‌ها دریافت کرده و سیگنال‌هایی را برای خروجی‌ها تولید می‌نمایند. این فرآیند اطلاعات دودویی مفروض در ورودی را به اطلاعات موردنیاز در خروجی تبدیل می‌کند. نمودار کلی یک مدار ترکیبی در شکل ۴-۱ دیده می‌شود. n متغیر دودویی ورودی از منبع



شکل ۴-۱ . نمودار بلوکی یک مدار ترکیبی

بیرونی دریافت و m متغیر خروجی به مقصد بیرونی ارسال می‌شوند. هر متغیر ورودی و یا خروجی به طور فیزیکی به صورت یک سیگنال نشان داده می‌شوند و این سیگنال‌ها نیز 0 و 1 منطقی را نمایش می‌دهند. در بسیاری از کاربردها، منبع و مقصد، ثبات‌های ذخیره‌سازی هستند. اگر ثبات‌ها به همراه گیت‌های منطقی به کار روند، کل مدار با نام مدار ترتیبی شناخته خواهد شد.

برای n متغیر ورودی، 2^n ترکیب ممکن دودویی از ورودی‌ها وجود دارد. برای هر ترکیب ممکن از ورودی‌ها فقط یک مقدار برای خروجی موجود است. بنابراین، یک مدار ترکیبی با یک جدول درستی، که مقادیر خروجی‌ها را در برابر هر ترکیب از متغیرهای ورودی لیست می‌نماید، نشان داده می‌شود. یک مدار ترکیبی با m تابع بولی نیز قابل نمایش است، که هر یک متعلق به یک خروجی است. هر تابع خروجی برحسب n متغیر ورودی بیان می‌گردد.

در فصل ۱ در مورد اعداد و کدهای دودویی که کمیت‌های گسسته‌ای از اطلاعات را نمایش می‌دهند، مطالبی آموختیم. متغیرهای دودویی به طور فیزیکی با ولتاژها و دیگر انواع سیگنال‌ها نشان داده می‌شوند. سیگنال‌ها نیز در سیستم‌های منطقی دیجیتال برای اجرای توابع موردنیاز دستکاری می‌شوند. در فصل ۲، جبر بول را به عنوان روشی جبری در بیان توابع منطقی معرفی نمودیم. در فصل ۳ آموختیم که چگونه عبارت بول را برای دستیابی به یک پیاده‌سازی اقتصادی، ساده کنیم. در این فصل با استفاده از دانش فصل‌های قبل، تحلیل و طراحی مدارهای ترکیبی را فرموله می‌نماییم. با حل مثال‌های نمونه فهرستی از توابع اصلی مهم برای درک سیستم‌های دیجیتال فراهم خواهد شد.

مدارهای ترکیبی متعددی وجود دارند که در طراحی سیستم‌های دیجیتال به کرّات به کار می‌روند. این مدارها به صورت مجتمع در دسترس بوده و به عنوان قطعات استاندارد دسته‌بندی شده‌اند. آنها توابع دیجیتال خاصی را که عموماً در طراحی سیستم‌های دیجیتال مورد نیازند، اجرا می‌کنند. در این فصل ما مهمترین مدارهای ترکیبی استاندارد مانند جمع‌کننده‌ها، تفریق‌گرها، مقایسه‌گرها، دیکدرها، انکدرها و مولتی پلکسرها را معرفی می‌کنیم. این قطعات به صورت مدارهای مجتمع MSI (مجتمع با فشردگی متوسط) در دسترسند. به آنها در مدارهای پیچیده VLSI ، مانند مدارات مجتمع خاص (ASIC) ، سلول‌های استاندارد هم می‌گویند. توابع سلول‌های استاندارد در داخل مدارهای VLSI به همان شکل به هم متصل می‌شوند که در طراحی MSI متشکل از چند IC ، وصل شدند.

۲-۴ روش تحلیل

تحلیل یک مدار ترکیبی به این معنی است که ما تابعی را که مدار پیاده‌سازی می‌کند، معین نماییم. این کار با یک نمودار منطقی مفروض آغاز شده و با مجموعه‌ای از توابع بول، یک جدول درستی، یا توضیحاتی از عمل مدار پایان می‌یابد. اگر نمودار منطقی مورد بررسی با نام تابع یا توضیحی از کار آن همراه باشد، آنگاه تحلیل به تصدیق تابع بیان شده کاهش می‌یابد. تحلیل را می‌توان به طور دستی با یافتن توابع بول یا جدول درستی، و یا با استفاده از یک برنامه شبیه‌سازی کامپیوتری اجرا نمود.

اولین قدم در تحلیل این است که مطمئن شویم مدار از نوع ترکیبی است و نه ترتیبی. نمودار یک

مدار ترکیبی حاوی گیت‌هایی است که فاقد مسیرهای پسخورد یا حافظه است. یک مسیر پسخورد، اتصالی است از خروجی یک گیت به ورودی گیت دیگری که خود بخش ورودی آن را (گیت خروجی) تشکیل می‌دهد. مسیرهای پسخوردی در یک مدار دیجیتال مدار ترتیبی را تعریف می‌کنند و باید طبق روال فصل ۹ با آنها رفتار کرد.

به محض این که محقق شد مدار از نوع ترکیبی است، می‌توان برای بدست آوردن توابع بول خروجی یا جداول درستی پیش رفت. اگر تابع مدار تحت بررسی است، لازم است عمل مدار را از توابع بول حاصل یا جداول درستی تفسیر کرد. موفقیت در چنین بررسی‌هایی به شرطی میسر است که فرد تجربه قبلی و آشنایی لازم با چنین مدارهایی داشته باشد.

برای بدست آوردن توابع بول خروجی از یک مدار منطقی به ترتیب زیر باید عمل کرد.

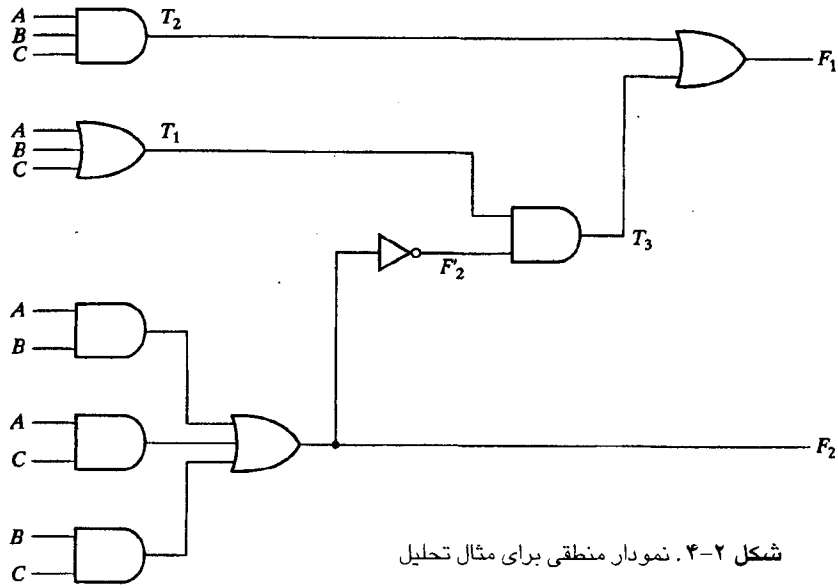
۱- تمام خروجی‌های گیت‌هایی که تابعی از ورودی هستند باید با سمبل‌های دلخواه نام‌گذاری شوند. برای خروجی هر گیت تابع بول را معین کنید.

۲- گیت‌هایی که تابعی از متغیرهای ورودی و گیت‌های برچسب خورده قبلی‌اند را با سمبل‌های اختیاری دیگری برچسب بزنید. برای این گیت‌ها نیز توابع بول خروجی را بدست آورید.

۳- فرآیند مرحله ۲ را تا دستیابی به خروجی‌های مدار ادامه دهید.

۴- با جایگزینی توابع بدست آمده در قبل، توابع بولی خروجی را برحسب متغیرهای ورودی اولیه بدست آورید.

تحلیل مدارهای ترکیبی شکل ۲-۴ روال پیشنهادی را تشریح می‌نماید. توجه دارید که مدار دارای سه ورودی دودویی A، B و C و دو خروجی F_1 و F_2 است. خروجی گیت‌هایی که تابعی از متغیرهای ورودی‌اند عبارتند از T_1 و T_2. خروجی F_2 به سادگی از متغیرهای ورودی بدست می‌آید. توابع بول



شکل ۲-۴. نمودار منطقی برای مثال تحلیل

برای این سه خروجی عبارتند از:

$$F_2 = AB + AC + BC$$
$$T_1 = A + B + C$$
$$T_2 = ABC$$

اکنون خروجی گیت‌هایی که تابعی از سمبل‌های قبلی می‌باشند را ملاحظه می‌نماییم:

$$T_3 = F_2'T_1$$
$$F_1 = T_3 + T_2$$

برای بدست آوردن F_1 برحسب A، B و C، یک سری جایگزینی‌ها را به فرم زیر انجام می‌دهیم:

$$\begin{aligned} F_1 &= T_3 + T_2 = F_2'T_1 + ABC = (AB + AC + BC)'(A + B + C) + ABC \\ &= (A' + B')(A' + C')(B' + C')(A + B + C) + ABC \\ &= (A' + B'C')(AB' + AC' + BC' + B'C) + ABC \\ &= A'BC' + A'B'C + AB'C' + ABC \end{aligned}$$

اگر بخواهیم این بررسی را دنبال کرده و عمل انتقال اطلاعات را با این مدار مشخص نماییم، می‌توانیم مدار را از عبارات بولی فوق رسم کرده و سعی کنیم عملیات آشنا را تشخیص دهیم. توابع بول F_1 و F_2 با مدار شکل ۷-۴ پیاده‌سازی می‌شوند (بخش ۴-۴) و معادل با یک مدار جمع کننده (یا تمام جمع کننده) است.

بدست آوردن جدول درستی برای مدار به محض شناختن توابع بولی خروجی روندی ساده است. برای تهیه مستقیم جدول درستی از نمودار منطقی و بدون نیاز به توابع بول به طریق زیر عمل کنید.

۱- تعداد متغیرهای ورودی در مدار را مشخص کنید. برای n ورودی 2^n ترکیب از ورودی‌ها را تشکیل دهید. آنگاه اعداد دودویی را در جدول از 0 تا 2^n-1 لیست نمایید.

۲- خروجی‌های گیت‌های انتخابی را با سمبل‌های دلخواه برچسب بزنید.

۳- برای آن دسته از خروجی گیت‌ها که فقط تابعی از متغیرهای ورودی هستند جدول درستی را بدست آورید.

۴- برای بدست آوردن خروجی گیت‌هایی که تابعی از مقادیر تعریف شده قبلی هستند پیش بروید تا ستون همه خروجی‌ها معین شود.

این فرآیند با استفاده از مدار شکل ۲-۴ تشریح می‌شود. در جدول ۱-۴، هشت ترکیب ممکن را برای

جدول ۱-۴. جدول درستی برای نمودار منطقی شکل ۲-۴

A	B	C	F_2	F_2	T_1	T_2	T_3	F_1
0	0	0	0	1	0	0	0	0
0	0	1	0	1	1	0	1	1
0	1	0	0	1	1	0	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	1	1	0	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	0	1	1	0	1

سه متغیر ورودی تشکیل می‌دهیم. جدول درستی برای F_2 مستقیماً از مقادیر A ، B و C تشکیل می‌شود که در آن برای هر ترکیبی که دو یا سه ورودی برابر با 1 دارد، F_2 برابر 1 است. جدول درستی برای F'_2 متمم F_2 است. جداول درستی برای T_1 و T_2 به ترتیب توابع OR و AND متغیرهای ورودی می‌باشند. مقدار T_3 از T_1 و F'_2 حاصل می‌شود: وقتی که هر دو T_1 و F'_2 برابر با 1 باشند T_3 نیز برابر 1 است، در غیر این صورت T_3 برابر 0 خواهد بود. بالاخره F_1 برای آن دسته از ترکیبات 1 است که در آنها T_2 یا T_3 یا هر دو برابر 1 باشند. بررسی ترکیبات جدول درستی برای A ، B و C و F_1 و F_2 نشان می‌دهد که این جدول با جدول جمع کننده کامل بخش ۴-۴ با متغیرهای x و y و z و S و C برابر است.

روش دیگر در تحلیل یک مدار ترکیبی شبیه‌سازی منطقی آن است. در بخش ۱۱-۴ شبیه‌سازی منطقی و تصدیق مدار شکل ۲-۴ را با کمک Verilog–HDL خواهیم دید (مثال ۱۰-۴ ملاحظه شود.)

۳-۴ روش طراحی

طراحی مدارهای ترکیبی با مشخصات مسئله آغاز و به فرم نمودار مدار منطقی یا مجموعه‌ای از توابع بول که به کمک آنها نمودار منطقی حاصل می‌شود، پایان می‌یابد. روال شامل موارد زیر است:

۱- با استفاده از مشخصات مدار تعداد ورودی‌ها و خروجی‌ها را معین کرده و به هر کدام سمبلی تخصیص دهید.

۲- جدول درستی مربوط به ورودی‌ها و خروجی‌های مدار را تشکیل دهید.

۳- توابع بولی ساده شده را برای هر خروجی به صورت تابعی از متغیرهای ورودی بدست آورید.

۴- نمودار منطقی را رسم کرده و صحت طراحی را تحقیق نمایید.

جدول درستی یک مدار ترکیبی، از ستون‌های ورودی و ستون‌های خروجی تشکیل می‌شود. ستون‌های ورودی از 2^n ترکیب مربوط به n متغیر ورودی بدست می‌آید. مقادیر دودویی خروجی‌ها از مشخصات بیان شده در مسئله حاصل می‌گردد. توابع خروجی مشخص شده در جدول درستی تعریف دقیقی از مدار ترکیبی را بدست می‌دهند. تفسیر لفظی صحیح جدول درستی از اهمیت خاصی برخوردار است. اغلب مشخصات لفظی کامل نیستند و تفسیر غلط ممکن است جدول درستی غلطی را تولید کند.

توابع دودویی خروجی لیست شده در جدول با روش‌های موجود مانند دستکاری جبری، نقشه کارنو یا برنامه‌های ساده‌سازی مبتنی بر کامپیوتر ساده می‌شوند. غالباً عبارات ساده شده متعددی حاصل می‌شوند که باید مناسب‌ترین را انتخاب کرد. در یک کاربرد خاص، معیارهای مختلفی در انتخاب یک پیاده‌سازی نقش دارند. یک طرح عملی قیودی چون تعداد گیت‌ها، تعداد ورودی‌ها به یک گیت، زمان انتشار سیگنال در گیت‌ها، تعداد اتصالات داخلی، محدودیت‌های مربوط به قابلیت راه‌اندازی هر گیت، و دیگر معیارهایی که باید در طراحی با مدارهای مجتمع مدّنظر باشند، را در نظر می‌گیرد. چون اهمیت هر قید را کاربرد خاص دیکته می‌کند، بیان یک عبارت جامع درباره ساده‌سازی مشکل است. در بسیاری از حالات ساده‌سازی با تصدیق و تأیید یک هدف ساده، مثل تولید توابع بولی به فرم استاندارد آغاز شده و سپس با برآورده کردن دیگر معیارهای رفتاری پیش می‌رود.

مثال تبدیل کد

وجود کدهای گوناگون و متنوع برای بیان اجزاء اطلاعات گسسته، باعث شده است تا سیستم‌های دیجیتال مختلف از کدهای متفاوتی استفاده کنند. گاهی لازم است خروجی یک سیستم به عنوان ورودی به سیستمی دیگر استفاده شود. اگر این دو سیستم از کدهای متفاوتی برای بیان اطلاعات یکسان استفاده کنند، یک مدار مبدل باید بین آن دو قرار داده شود. بنابراین یک مبدل کد مداری است که دو سیستم را، علیرغم به کارگیری کد دودویی متفاوت، با هم سازگار می‌سازد.

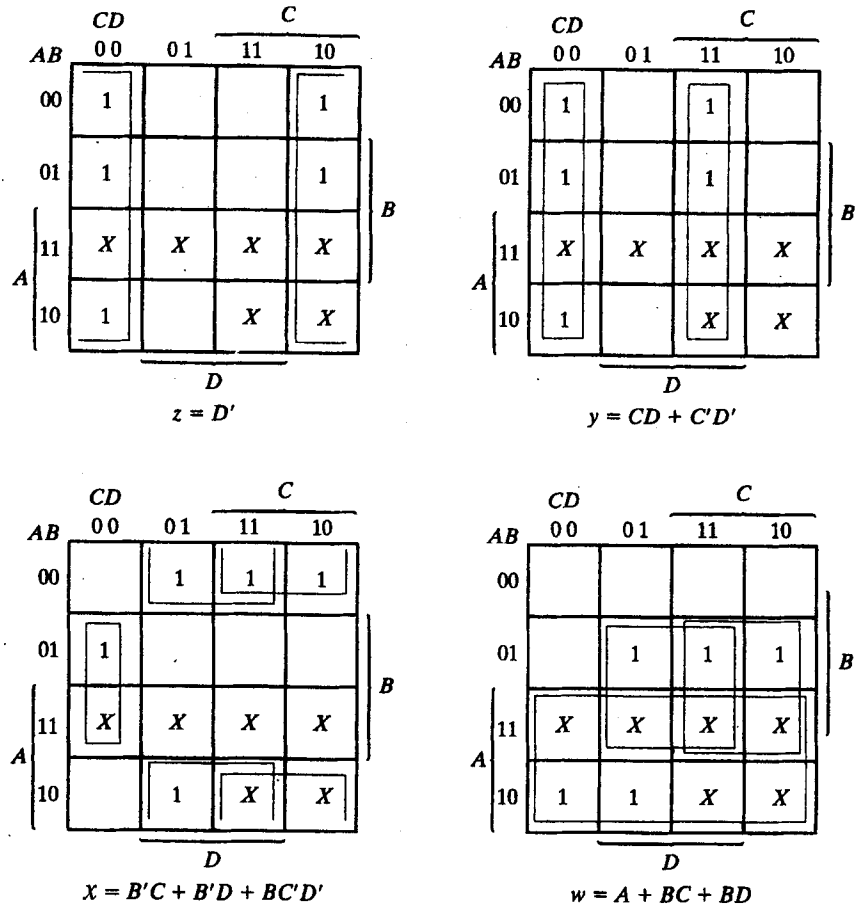
برای تبدیل کد دودیی A به کد دودیی B، خطوط ورودی باید ترکیبات بیتی اجزاء مشخص شده با کد A را تهیه نموده و خطوط خروجی نیز باید ترکیبات کد B مربوطه را تولید نمایند. یک مدار ترکیبی به کمک گیت‌ها این تبدیل را انجام می‌دهد. روش طراحی با مثالی که دهدهی کد شده به دودویی (BCD) را به کد افزونی -3 تبدیل می‌نماید، تشریح خواهد شد.

ترکیبات تخصیص داده شده به BCD و افزونی -3 در جدول ۵-۱ لیست شده است (بخش ۷-۱). چون هر کد، از چهار بیت برای نمایش یک رقم دهدهی استفاده می‌نماید، باید چهار متغیر ورودی و چهار متغیر خروجی داشته باشیم. چهار متغیر دودویی را با A، B، C و D و چهار متغیر خروجی را با w، x، y و z نام‌گذاری کنید. جدول درستی رابط بین ورودی‌ها و خروجی‌ها در جدول ۲-۴ دیده می‌شود. ترکیبات بیتی برای ورودی‌ها و خروجی‌های متناظر آنها مستقیماً در بخش ۷-۱ بدست آمد. توجه کنید که چهار متغیر دودویی دارای 16 ترکیب‌اند ولی تنها 10 عدد از آنها در جدول درستی ذکر شده‌اند. 6 ترکیب بیتی ذکر نشده برای متغیرهای ورودی ترکیبات بی‌اهمیت هستند. این مقادیر در BCD مفهوم ندارند و فرض بر این است که هرگز رخ نمی‌دهند. بنابراین به متغیرهای خروجی مربوط به آنها به دلخواه 0 یا 1 خواهیم داد و این تخصیص به نحوی خواهد بود که از آن مدار ساده‌تری حاصل گردد.

نقشه‌ها در شکل ۳-۴ برای بدست آوردن توابع بول خروجی ساده شده رسم شده‌اند. هر یک از چهار نقشه به یکی از خروجی‌های مدار به عنوان تابعی از چهار متغیر ورودی مربوط است. 1هایی که

جدول ۲-۴. جدول درستی برای مثال تبدیل کد

ورودی BCD				خروجی کد افزونی - 3			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



شکل ۳-۴. نقشه‌های تبدیل BCD به افزونی -3

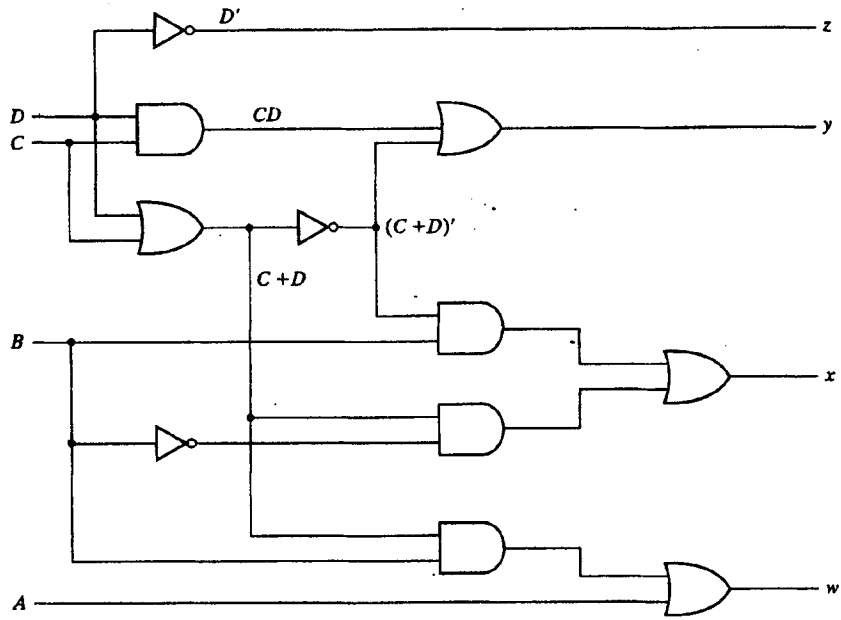
در مربع‌ها نوشته شده‌اند از مینترم‌هایی که خروجی را 1 کنند بدست می‌آیند. این 1ها با در نظر گرفتن تک‌تک ستون‌های خروجی در جدول درستی مشخص می‌شوند. مثلاً ستون زیر خروجی z دارای پنج عدد 1 است؛ بنابراین، نقشه z دارای پنج 1 می‌باشد که هر یک متعلق به مینترمی است که z توسط آن برابر 1 می‌شود. شش مینترم بی‌اهمیت از 10 تا 15 با علامت X علامت زده شده‌اند. یکی از نتایج ساده‌سازی توابع در جمع حاصلضرب‌ها در زیر نقشه هر متغیر خروجی نوشته شده است.

نمودار دو سطحی را می‌توان مستقیماً از عبارات بولی حاصل از نقشه‌ها بدست آورد. البته فرم‌های متعدد دیگری نیز برای بدست آوردن نمودار منطقی که همین مدار را پیاده‌سازی کند وجود دارد. عبارات حاصل در شکل ۳-۴ را به منظور استفاده از گیت‌های مشترک می‌توان دستکاری جبری نمود. این دستکاری جبری که در زیر آمده است انعطاف‌پذیری حاصل با سیستم‌های چند خروجی را وقتی با سه، یا چهار سطح و یا بیشتر پیاده‌سازی می‌شوند، نشان می‌دهد.

$$z = D'$$
$$y = CD + C'D' = CD + (C + D)'$$
$$x = B'C + B'D + BC'D' = B'(C + D) + BC'D'$$
$$= B'(C + D) + B(C + D)'$$
$$w = A + BC + BD = A + B(C + D)$$

نمودار منطقی که این توابع را پیاده‌سازی می‌کند در شکل ۴-۴ دیده می‌شود. مشاهده می‌شود. گیت OR که خروجی‌اش (C + D) است به نحوی در پیاده‌سازی هر سه خروجی به کار رفته است.

بدون احتساب گیت‌های وارونگر در ورودی، پیاده‌سازی به صورت جمع حاصلضرب‌ها به هفت گیت AND و سه گیت OR نیاز دارد. در شکل ۴-۴ همین سیستم به چهار گیت AND، چهار گیت OR و یک وارونگر احتیاج دارد. اگر تنها ورودی‌های معمولی یا نرمال در دسترس باشند، پیاده‌سازی اول به وارونگرهایی برای متغیرهای B و C و D نیاز خواهد داشت، ولی در پیاده‌سازی دوم فقط B و D نیاز به وارونگر دارند.



شکل ۴-۴. نمودار منطقی برای تبدیل BCD به افزونی -3

۴-۴ جمع‌کننده- تفریق‌گر دودویی

کامپیوترهای دیجیتال اعمال پردازش اطلاعات گوناگونی را انجام می‌دهند. از آن میان می‌توان توابع مربوط به انواع اعمال حسابی را نام برد. اصلی‌ترین عمل حسابی جمع دو رقم دودویی است. این جمع

ساده شامل چهار عمل پایه است: یعنی $0 + 0 = 0$، $0 + 1 = 1$، $1 + 0 = 1$ و $1 + 1 = 10$. سه عمل اول جمعی یک رقمی تولید می‌کنند، ولی وقتی هر دو بیت مضاف و مضاف‌الیه برابر 1 باشند، جمع دودویی از دو رقم تشکیل خواهد شد. بیت باارزش‌تر این نتیجه را نقلی می‌گویند. وقتی مضاف و مضاف‌الیه دارای ارقام باارزش‌تر بیشتری باشند، نقلی حاصل از جمع دو بیت با جفت بیت باارزش‌تر بعدی افزوده می‌شود. مدار ترکیبی که جمع دو بیت را انجام می‌دهد، نیم جمع‌کننده نام دارد. مداری که سه بیت را با هم جمع کند، (دو بیت به علاوه بیت نقلی) جمع‌کننده کامل یا تمام جمع کننده خوانده می‌شود. اسم مدارها به این علت انتخاب شده است که از دو نیم جمع‌کننده می‌توان در پیاده‌سازی یک جمع کننده کامل استفاده کرد.

یک جمع- تفریق‌گر دودویی مداری ترکیبی است که عملیات حسابی جمع و تفریق را با اعداد دودویی انجام می‌دهد. ما این مدار را به صورت سلسله مراتبی طراحی خواهیم کرد. ابتدا طراحی نیم‌جمع‌کننده انجام می‌شود، و با استفاده از آن جمع کننده کامل را طراحی خواهیم کرد. با اتصال سری n جمع کننده کامل جمع دو عدد n بیتی تولید می‌گردد. مدار تفریق‌گر با افزودن مدار متمم‌ساز به آن ساخته می‌شود.

نیم جمع‌کننده

با توجه به توضیحات لفظی یک نیم‌جمع کننده، در می‌یابیم که مدار نیاز به دو ورودی دودویی و دو خروجی دودویی دارد. متغیرهای ورودی بیت‌های مضاف و مضاف‌الیه را مشخص می‌کنند. متغیرهای خروجی جمع و نقلی را تولید می‌نمایند. ما سمبل‌های x و y را به دو ورودی و S (برای جمع) و C (نقلی) را به خروجی‌ها تخصیص می‌دهیم. جدول درستی برای نیم جمع‌کننده در جدول ۳-۴ نشان داده شده است. خروجی C فقط هنگامی 1 است که هر دو ورودی 1 باشند. خروجی S، بیت کم‌ارزش‌تر حاصل جمع را نشان می‌دهد.

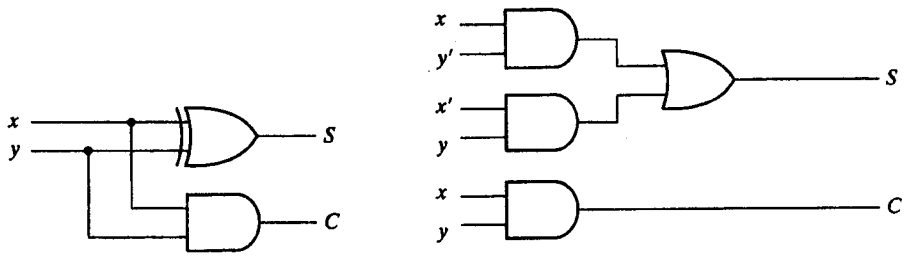
توابع بولی ساده شده برای دو خروجی مستقیماً از جدول درستی بدست می‌آیند. عبارات جمع حاصلضرب ساده شده عبارتند از:

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

نمودار منطقی نیم‌جمع کننده پیاده شده با جمع حاصلضرب‌ها در شکل ۵-۴ (الف) دیده می‌شود. می‌توان آن را با گیت‌های XOR و AND طبق شکل ۵-۴ (ب) هم پیاده کرد. این نوع برای ساخت جمع‌کننده کامل از دو نیم جمع‌کننده به کار می‌رود.

جدول ۳-۴. نیم جمع‌کننده

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



(ب) $S = x \oplus y$, $C = xy$

(الف) $S = xy' + x'y$, $C = xy$

شکل ۵-۴. پیاده‌سازی نیم جمع‌کننده

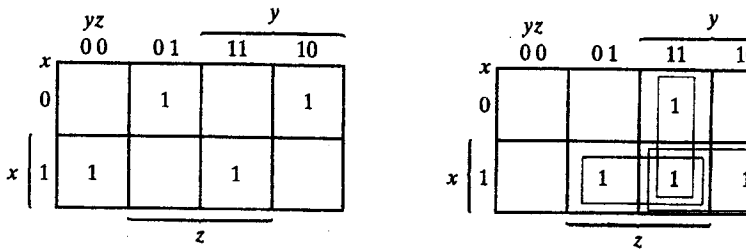
جمع‌کننده کامل

یک جمع‌کننده کامل مداری ترکیبی است که جمع حسابی سه بیت را تشکیل می‌دهد. این مدار دارای سه ورودی و دو خروجی است. دو متغیر ورودی که با x و y نشان داده شده‌اند دو بیت باارزش جمع شونده را نشان می‌دهند. ورودی سوم، z، نقلی حاصل از مکان کم‌ارزش‌تر قبلی است. دو خروجی لازم است زیرا جمع حسابی سه رقم دودویی بین 0 تا 3 می‌باشد و اعداد 2 و 3 به دو رقم دودویی نیاز دارند. دو خروجی با سمبل S برای جمع و C برای نقلی مشخص شده‌اند. متغیر دودویی S مقدار کم‌ارزش‌تر جمع را بدست می‌دهد. متغیر دودویی C نقلی خروجی را بیان‌گر است. جدول درستی جمع‌کننده کامل در جدول ۴-۴ دیده می‌شود. هشت سطر زیر سه متغیر همه ترکیبات ممکن سه متغیر را نشان می‌دهند. متغیرهای خروجی از جمع حسابی بیت‌های ورودی معین می‌شوند. وقتی همه بیت‌های ورودی 0 هستند، خروجی 0 است. خروجی S هنگامی 1 می‌شود که فقط یک ورودی برابر 1 باشد و یا اگر هر سه ورودی 1 باشند. خروجی C هم موقعی 1 است که دو یا سه ورودی برابر 1 باشند.

تفسیر بیت‌های ورودی و خروجی مدار ترکیبی در مراحل مختلف طراحی متفاوت است. به طور فیزیکی سیگنال‌های دودویی ورودی‌ها ارقامی دودویی تصور می‌شوند که به صورت حسابی باید با هم جمع شده و جمع دو رقمی را در خروجی تولید کنند. از طرف دیگر، در جدول درستی و یا هنگام

جدول ۴-۴. جمع کننده کامل

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



شکل ۶-۴. نقشه جمع‌کننده کامل به صورت جمع حاصلضرب‌ها

پیاده‌سازی با گیت‌های منطقی، همان مقادیر به عنوان متغیرهای بول تعبیر می‌شوند. نقشه خروجی‌های جمع‌کننده کامل در شکل ۶-۴ ملاحظه می‌شود. عبارات ساده شده عبارتند از:

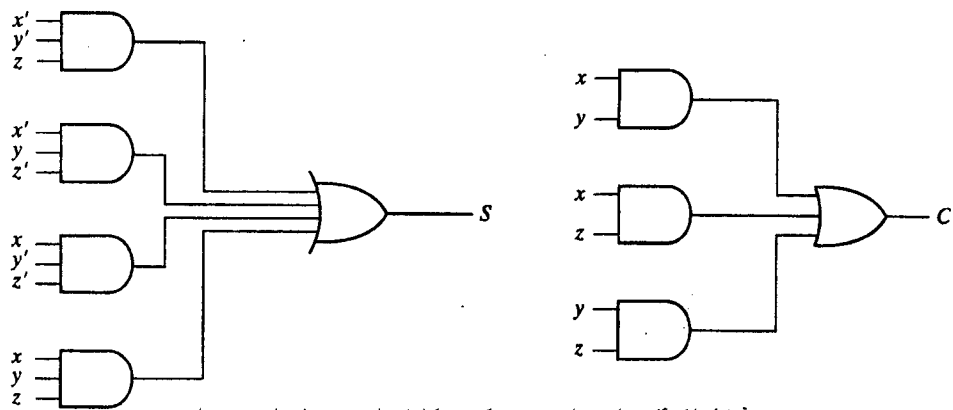
$$S = x'y'z + x'yz' + xy'z' + xyz$$
$$C = xy + xz + yz$$

نمودار منطقی پیاده شده به صورت جمع حاصلضرب‌ها در شکل ۷-۴ مشاهده می‌شود. می‌توان با دو نیم جمع‌کننده و یک OR هم طبق شکل ۸-۴ آن را پیاده‌سازی کرد. خروجی S از نیم‌جمع‌کننده دوم XOR متغیر z و خروجی نیم‌جمع کننده اول حاصل می‌شود، زیرا

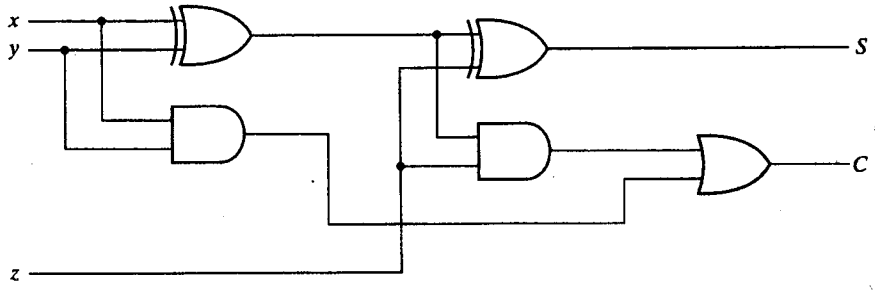
$$\begin{aligned} S &= z \oplus (x \oplus y) \\ &= z'(xy' + x'y) + z(xy' + x'y)' \\ &= z'(xy' + x'y) + z(xy + x'y') \\ &= xy'z' + x'yz' + xyz + x'y'z \end{aligned}$$

و نقلی خروجی برابر است با

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



شکل ۷-۴. پیاده‌سازی جمع کننده کامل با جمع حاصلضرب‌ها



شکل ۸-۴ . پیاده‌سازی یک جمع‌کننده کامل با دو نیم جمع‌کننده و یک گیت OR

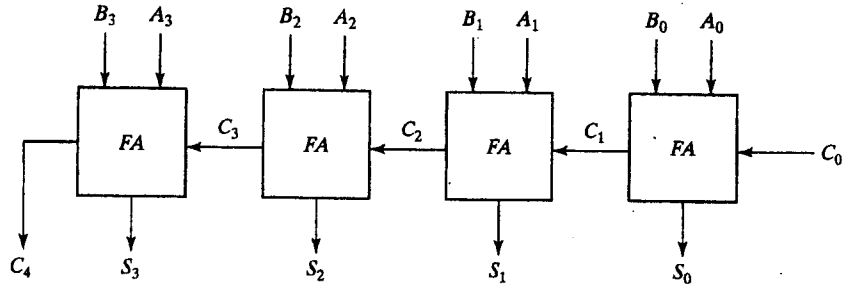
جمع‌کننده دودویی

یک جمع‌کننده دودویی مداری دیجیتال است که جمع حسابی دو عدد دودویی را تولید می‌کند. می‌توان آن را از به هم پیوستن متوالی جمع‌کننده کامل ساخت، و در آن هر خروجی نقلی از هر جمع‌کننده کامل به ورودی نقلی جمع‌کننده کامل بعدی زنجیروار بسته می‌شود. شکل ۹-۴ اتصالات درونی مدار چهار جمع‌کننده کامل (FA)، برای تهیه جمع‌کننده دودویی 4 بیت با نقلی موج‌گونه را نشان می‌دهد. بیت‌های مضاف از A و مضاف‌الیه از B با اعداد اندیس‌دار از راست به چپ و با اندیس 0 در بیت کم‌ارزش‌تر مشخص شده است. نقلی‌ها به صورت زنجیر جمع‌کننده‌های کامل را بهم وصل کرده‌اند. نقلی ورودی به جمع‌کننده C_0 وصل بوده و موج‌گونه‌وار تا نقلی خروجی C_4 انتشار می‌یابد. خروجی‌های S بیت‌های حاصل جمع را تولید می‌کنند. یک جمع‌کننده n بیت به n جمع‌کننده کامل نیاز دارد و هر خروجی نقلی به ورودی نقلی جمع‌کننده بالا رتبه‌تر وصل می‌شود.

به منظور تشریح بیشتر مثالی را با اعداد دودویی A = 1011 و B = 0011 در نظر بگیرید. حاصل جمع آنها S = 1110 است که از جمع چهار بیت مطابق زیر بدست می‌آید.

اندیس i :	3	2	1	0	
نقلی ورودی	0	1	1	0	C_i
مضاف	1	0	1	1	A_i
مضاف الیه	0	0	1	1	B_i
حاصل جمع	1	1	1	0	S_i
نقلی خروجی	0	0	1	1	C_{i+1}

بیت‌ها با کمک جمع‌کننده کامل و از کم‌ارزش‌ترین مکان (اندیس 0) با هم جمع می‌شوند تا بیت حاصل‌جمع ونقلی را تشکیل دهند. نقلی ورودی C_0 در کم‌ارزش‌ترین مکان باید 0 باشد. مقدار C_{i+1} در یک مکان مفروض، نقلی خروجی جمع‌کننده کامل است. این مقدار به نقلی ورودی تمام جمع‌کننده‌ای که بیت‌های یک مکان بالاتر در سمت چپ را جمع می‌کند انتقال می‌یابد. بنابراین بیت‌های جمع از



شکل ۹-۴. جمع‌کننده 4 بیت

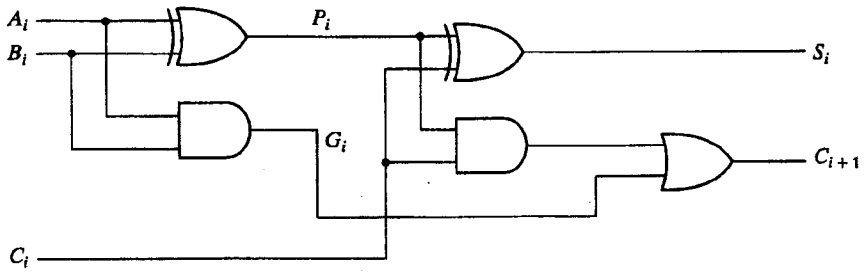
راست به چپ تولید شده و به محض تولید نقلی قبل از خود، در اختیار خواهند بود. برای داشتن خروجی جمع صحیح همه نقلی‌ها باید تولید شده باشند.

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

انتشار رقم نقلی

جمع دو عدد دودویی به صورت موازی لازم می‌دارد که مضاف و مضاف‌الیه به طور همزمان برای محاسبه موجود باشند. همچون دیگر مدارهای ترکیبی، در این مدار هم قبل از داشتن یک جواب صحیح، سیگنال باید از گیت‌ها عبور کند. زمان کل انتشار برابر است با زمان تأخیر انتشار یک نمونه گیت ضربدر طبقات گیت‌ها در مدار. طولانی‌ترین زمان تأخیر انتشار در یک جمع‌کننده زمانی است که نقلی برای انتشار در جمع‌کننده‌ای کامل لازم دارد. چون هر بیت از خروجی جمع به نقلی ورودی‌اش وابسته است مقدار S_i در هر طبقه مفروض در جمع‌کننده تنها موقعی به مقدار پایدار نهایی خود می‌رسند که نقلی ورودی به آن طبقه رسیده باشد. مثلاً خروجی S_3 را در شکل ۹-۴ در نظر بگیرید. به محض اعمال ورودی‌ها به جمع‌کننده، ورودی‌های A_3 و B_3 در دسترسند. با این وجود نقلی ورودی C_3 تا تولید C_2 به وسیله طبقه قبل در مقدار نهایی‌اش پایدار نمی‌شود. به طور مشابه C_2 منتظر C_1 و C_1 منتظر C_0 خواهد بود. بنابراین پس از انتشار موج‌گونه نقلی در همه طبقات، خروجی S_3 و نقلی C_4 در مقادیر صحیح نهایی خود پایدار خواهند شد.

تعداد طبقات گیت برای انتشار نقلی را باید از مدار هر جمع‌کننده کامل بدست آورد. به منظور سهولت این مدار در شکل ۱۰-۴ دوباره ترسیم شده است. متغیرهای ورودی و خروجی از اندیس i برای مشخص کردن شماره طبقه جمع‌کننده استفاده کرده‌اند. سیگنال‌ها در P_i و G_i هنگامی به ثبات می‌رسند که از گیت‌های مربوطه‌شان انتشار یافته باشند. این دو سیگنال که در همه جمع‌کننده‌های کامل وجود دارند، به بیت‌های مضاف و مضاف‌الیه ورودی وابسته‌اند. سیگنال نقلی ورودی C_i از طریق گیت AND و یک



شکل ۱۰-۴. جمع‌کننده کامل با P و G

گیت OR، که دو سطح گیت را تشکیل می‌دهند، به C_{i+1} می‌رسد. اگر در مجموع، چهار جمع‌کننده کامل وجود داشته باشد بین نقلی خروجی C_0 تا C_4، 8=4×2 طبقه گیت وجود خواهد داشت. برای یک جمع کننده n بیت، 2n طبقه گیت وجود دارد تا نقلی ورودی از طریق آنها انتشار یافته و به خروجی برسد.

زمان انتشار نقلی، فاکتور محدود کننده‌ای روی سرعت جمع دو عدد می‌باشد. گرچه جمع‌کننده، یا هر مدار ترکیبی دیگر دارای مقداری در پایانه‌اش هست، ولی خروجی‌ها صحیح نخواهند بود مگر این که فرصتی کافی برای انتشار سیگنال از گیت‌های متصل به هم از ورودی تا خروجی داده شود. چون همه عملیات حسابی با جمع تکراری صورت می‌گیرد، زمان مصرف شده در طول فرآیند جمع بسیار حساس خواهد بود. راه‌حل روشنی برای کاهش زمان تأخیر انتشار استفاده از گیت‌های سریع‌تر است. با این وجود، مدارهای فیزیکی در قابلیت خود محدودیت دارند. راه‌حل دیگر افزایش پیچیدگی مدار به طریقی است که زمان تأخیر نقلی کاهش یابد. چند تکنیک برای کاهش زمان انتشار نقلی در جمع‌کننده‌های موازی وجود دارد. رایج‌ترین تکنیک استفاده از اصل پیش‌بینی نقلی می‌باشد. مدار جمع‌کننده کامل شکل ۱۰-۴ را ملاحظه نمایید. اگر دو متغیر دودویی جدید زیر را معرفی کنیم:

$$P_i = A_i \oplus B_i$$
$$G_i = A_i B_i$$

حاصل جمع خروجی و نقلی آن را می‌توان به صورت زیر تعریف کرد.

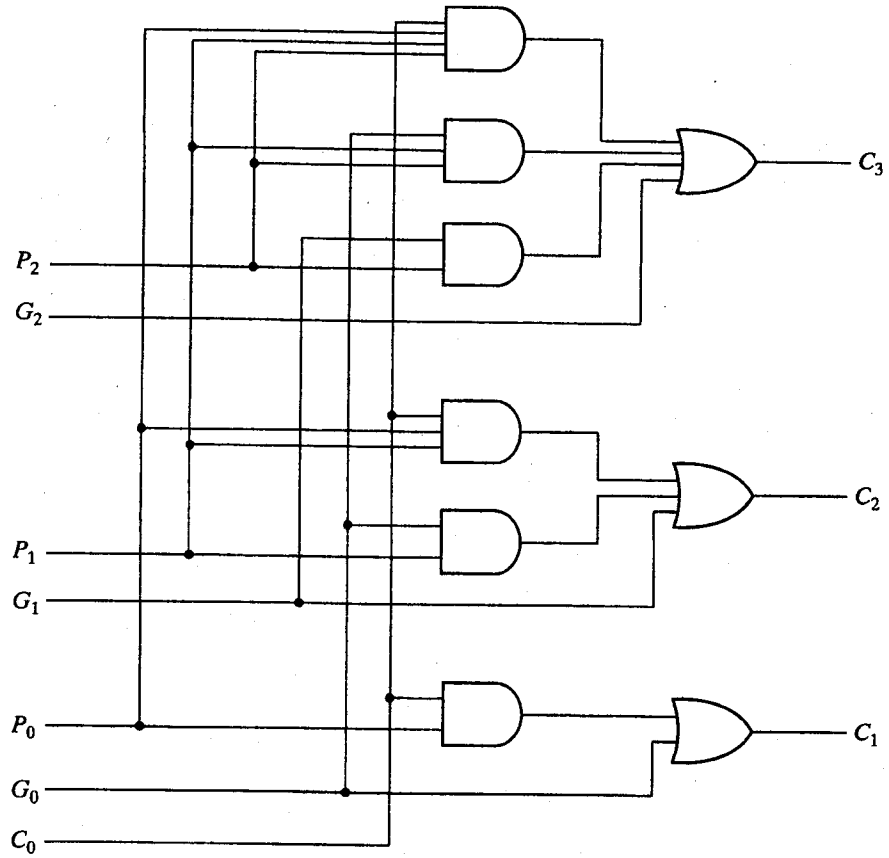
$$S_i = P_i \oplus C_i$$
$$C_{i+1} = G_i + P_i C_i$$

G_i را مولد نقلی نامند که وقتی هر دو A_i و B_i برابر 1 باشند، یک نقلی 1 را تولید می‌نماید، و این تولید مستقل از C_i می‌باشد. P_i را انتشار نقلی گویند زیرا جمله‌ای است که در انتشار نقلی از C_i به C_{i+1} نقش دارد.

اکنون توابع بول را برای خروجی‌های نقلی هر طبقه نوشته و هر C_i را با مقدار معادل قبلی جایگزین می‌کنیم:

$C_0 =$ نقلی ورودی

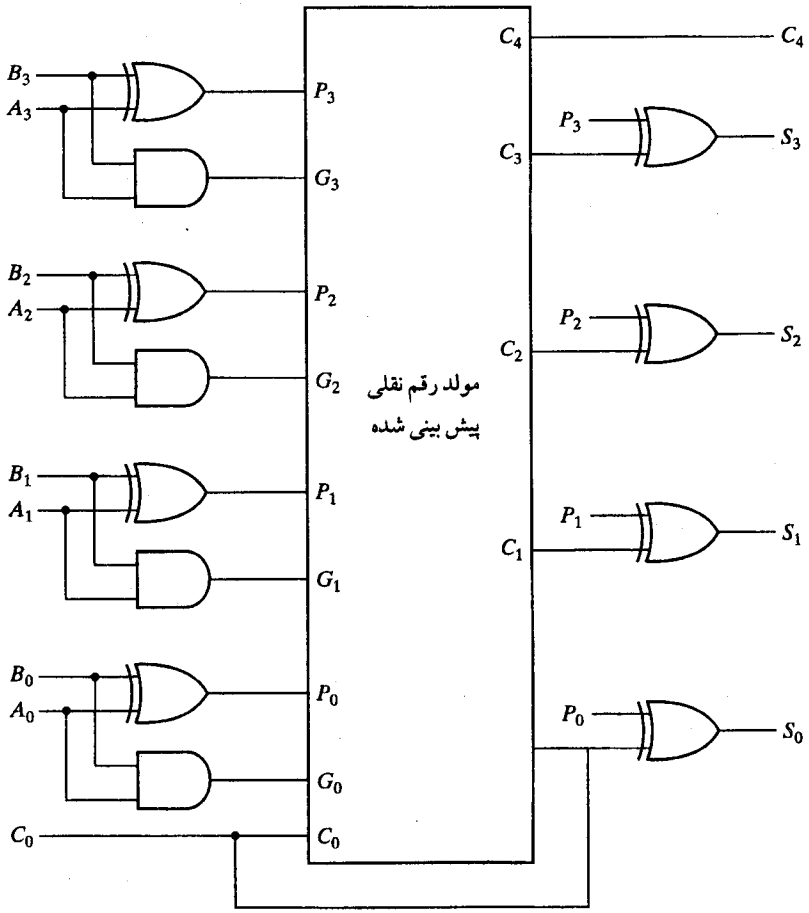
$$C_1 = G_0 + P_0 C_0$$
$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$$



شکل ۱۱-۴. نمودار منطقی مولد پیش‌بینی نقلی

چون تابع بول برای هر نقلی خروجی برحسب جمع حاصلضرب‌ها بیان شده است، هر تابع قابل پیاده‌سازی با یک طبقه گیت AND و به دنبال آن گیت OR (یا با دو طبقه NAND) است. سه تابع بول برای C_1 ، C_2 و C_3 در مولد نقلی پیش‌بینی شونده و در شکل ۱۱-۴ دیده می‌شوند. توجه کنید که C_3 نیاز ندارد به انتظار C_2 و C_1 بماند. در واقع C_3 با C_1 و C_2 همزمان منتشر می‌گردد.

ساخت یک جمع‌کننده 4 بیت با پیش بینی نقلی در شکل ۱۲-۴ مشاهده می‌شود. خروجی اولین گیت XOR متغیر P_i و گیت AND متغیر G_i را تولید می‌نماید. نقلی‌ها از درون مولد پیش بینی نقلی انتشار می‌یابند (مشابه شکل ۱۱-۴) و به عنوان ورودی به گیت XOR دوم اعمال می‌گردند. همه نقلی‌های خروجی پس از یک تأخیر در دو طبقه گیت به طور همزمان تولید می‌شوند. بنابراین S_1 تا S_3 دارای زمان تأخیر انتشار یکسانی هستند. مدار دو طبقه برای نقلی خروجی C_4 نشان داده نشده است. این مدار هم به سادگی با روش جایگزینی قابل دستیابی است.

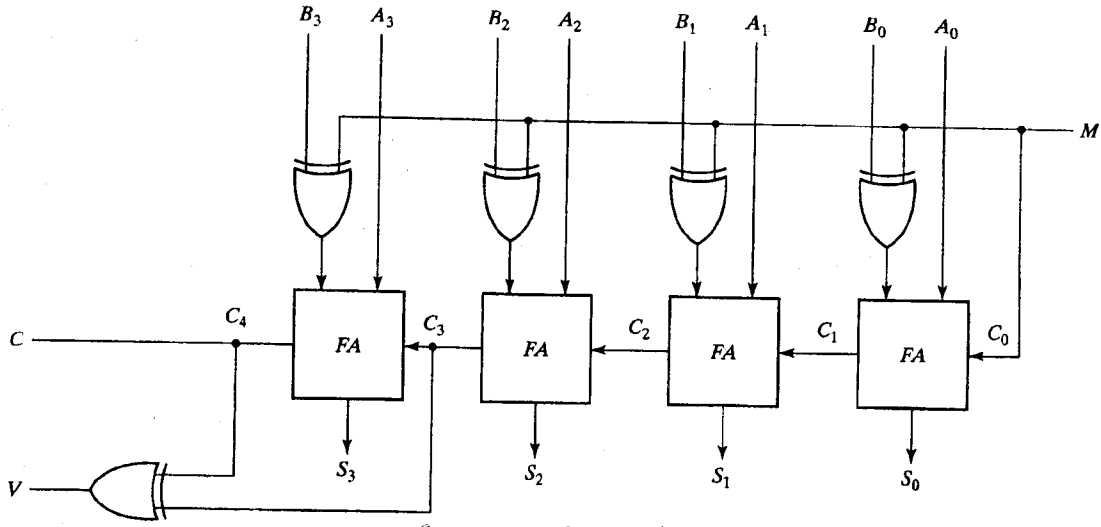


شکل ۱۲-۴. جمع‌کننده 4 بیت با پیش‌بینی نقلی

تفریق دودویی

در بخش ۵-۱ دیدیم که تفریق اعداد دودویی بی‌علامت با استفاده از متمم راحت‌تر انجام می‌گیرد. به خاطر دارید که A−B را می‌توان با محاسبه متمم 2 عدد B و جمع آن با A معین کرد. متمم 2 را با بدست آوردن متمم 1 و جمع آن با 1 محاسبه می‌کنند. متمم 1 را هم با وارونگر بدست آورده و عدد 1 را هم از طریق ورودی نقلی به آن اضافه می‌نمایند.

مدار تفریق‌گر A−B متشکل از یک جمع‌کننده با وارونگرهای واقع در بین ورودی B و ورودی مربوطه‌اش در تمام جمع‌کننده می‌باشد. نقلی ورودی C_0 به هنگام تفریق باید برابر با 1 شود. بنابراین عمل به صورت A بعلاوه متمم 1 عدد B بعلاوه 1 اجرا می‌شود. این عمل برابر با جمع A با متمم 2 عدد B خواهد بود. برای اعداد بی‌علامت اگر A≥B باشد، عمل فوق A−B و اگر A<B باشد (B−A) است. برای اعداد علامت‌دار، نتیجه A−B است به شرطی که سرریز وجود نداشته باشد (بخش ۶-۱ ملاحظه شود).



شکل ۱۳-۴ . جمع- تفریق‌گر

عملیات جمع و تفریق را می‌توان با یک مدار در هم ادغام کرده و با یک جمع‌کننده دودویی مشترک انجام داد. این کار با افزودن یک گیت XOR در هر جمع‌کننده کامل صورت می‌گیرد. یک مدار جمع- تفریق‌گر در شکل ۱۳-۴ دیده می‌شود. وقتی $M = 0$ است، مدار یک جمع‌کننده و وقتی $M = 1$ باشد، مدار یک تفریق‌گر خواهد بود. هر گیت XOR ورودی M و یکی از ورودی‌های B را دریافت می‌کند. وقتی $M = 0$ است داریم، $B \oplus 0 = B$. جمع‌کننده کامل B را دریافت می‌نماید، نقلی ورودی 0 است و بنابراین مدار عمل A بعلاوه B را اجرا می‌کند. اگر $M = 1$ باشد، $B \oplus 1 = B'$ بوده و $C_0 = 1$ است. ورودی‌های B همگی متمم شده و از طریق ورودی نقلی، یک 1 به آن اضافه می‌شود. در این حالت مدار یک عمل A بعلاوه متمم 2 عدد B را انجام می‌دهد. (XOR با خروجی V ، یک سرریز را شناسایی می‌نماید).

لازم است متذکر شویم که در سیستم متمم علامت‌دار، اعداد دودویی هم چون اعداد بی‌علامت، با قوانین جمع و تفریق یکسانی ترکیب می‌شوند. بنابراین، کامپیوترها نیاز به یک سخت‌افزار مشترک دارند تا هر دو نوع محاسبه را انجام دهند. کاربر یا برنامه‌نویس باید نتایج چنین جمع یا تفریقی را متفاوت تفسیر کنند و این به علامت‌دار یا بی‌علامت بودن اعداد بستگی دارد.

سرریز

هرگاه دو عدد n رقمی با هم جمع شوند و حاصل جمع n+1 رقم را اشغال کند، گوییم سرریز رخ داده است. این مطلب جدا از علامت‌دار بودن یا نبودن برای اعداد دهدهی یا دودویی صحیح است. وقتی که جمع با کاغذ و قلم انجام می‌شود، سرریز مسئله‌ای نیست زیرا محدودیتی برای عرض صفحه جهت نوشتن جمع وجود ندارد. ولی سرریز در کامپیوترهای دیجیتال مشکلاتی ایجاد می‌کند، زیرا تعداد

بیت‌های نگهدارنده عدد محدود بوده و نتیجه‌ای را که n+1 بیت دارد نمی‌توانند در خود جای دهند. به این دلیل، بسیاری از کامپیوترها وقوع یک سرریز را، اگر رخ دهد، شناسایی می‌کنند و فیلپ فلاپ مربوطه را در 1 می‌نشانند تا بعد به وسیله کاربر چک شود.

تشخیص یک سرریز پس از جمع دو عدد دودویی به این بستگی دارد که آیا اعداد علامت دارند یا بی‌علامت‌اند. وقتی دو عدد بی‌علامت با هم جمع شوند، یک سرریز از نقلی باارزش‌ترین مکان تشخیص داده می‌شود. در حالتی که اعداد علامت‌دار باشد، سمت چپ‌ترین بیت همواره علامت را نشان داده و اعداد منفی هم به صورت متمم 2 هستند. وقتی دو عدد علامت‌دار جمع شوند، با بیت علامت به عنوان بخشی از عدد رفتار می‌شود و رقم نقلی انتهایی هیچ سرریزی را مشخص نمی‌کند.

در جمع وقتی که یکی از اعداد مثبت و دیگری منفی باشد، سرریز رخ نمی‌دهد، زیرا جمع یک عدد مثبت با یک عدد منفی نتیجه‌ای تولید می‌کند که از بزرگترین آن دو کوچکتر است. سرریز هنگامی رخ می‌دهد که هر دو عدد جمع شونده مثبت یا منفی باشند. برای درک بهتر موضوع مثال زیر را ملاحظه کنید. دو عدد علامت‌دار دودویی 70+ و 80+ دودویی در دو ثبات 8 بیتی ذخیره شده‌اند. محدوده اعدادی که هر یک از ثبات‌ها داراست از 127+ تا 128− دودویی است. چون مجموع دو عدد 150+ است، حاصل از ظرفیت ثبات 8 بیتی تجاوز خواهد کرد. این مطالب هنگامی که هر دو عدد مثبت یا منفی باشند صحت دارد. دو جمع مذکور همراه با ارقام نقلی در زیر نشان داده شده‌اند:

نقلی‌ها :	0	1		نقلی‌ها :	1	0	
+70		0	1000110	−70		1	0111010
+80		0	1010000	−80		1	0110000
+150		1	0010110	−150		0	1101010

توجه کنید که حاصل جمع هشت بیتی که باید مثبت باشد یک بیت علامت منفی دارد و نتیجه 8 بیتی که باید منفی باشد دارای بیت علامت مثبت است. با این وجود اگر رقم نقلی خارج شده از بیت علامت به عنوان بیت علامت در نظر گرفته شود، آنگاه جواب 9 بیتی حاصل صحیح خواهد بود. چون پاسخ نمی‌تواند در 8 بیت جای داده شود، گوییم سرریز رخ داده است.

وضعیت سرریز را می‌توان با وجود رقم نقلی به بیت علامت و نقلی خروجی از بیت علامت مشاهده کرد. اگر این دو نقلی یکی نباشند، یک سرریز رخ داده است. این نکته در مثال‌های فوق که در آن دو نقلی به طور جداگانه نشان داده شده‌اند دیده می‌شود. اگر دو رقم نقلی را به یک گیت XOR اعمال کنیم، وقوع سرریز با 1 شدن خروجی این گیت شناسایی می‌شود. برای این که روش به خوبی کار کند متمم 2 باید از طریق بدست آوردن متمم 1 و جمع آن با 1 انجام گردد. این کار موجب مراقبت از حالتی می‌شود که در آن عدد منفی ماکزیمم متمم شود.

مدار جمع- تفریق‌گر با خروجی‌های C و V در شکل ۱۳-۴ دیده می‌شود. اگر دو عدد دودویی بی‌علامت تصور شوند، آنگاه بیت C، نقلی بعد از جمع یا قرض بعد از تفریق است. اگر اعداد علامت‌دار

فرض شوند، آنگاه بیت V یک سرریز را مشخص می‌کند. اگر $V = 0$ بعد از یک جمع یا تفریق باشد، بیانگر نبود سرریز بوده و نتیجه n بیتی حاصل صحیح است. اگر $V = 1$ باشد، در این صورت نتیجه عمل حاوی n+1 بیت می‌باشد، ولی بیت n+1ام علامت واقعی است که به یک مکان بیرونی منتقل شده است.

۴-۵ جمع‌کننده دهدهی

کامپیوترها یا ماشین‌های حسابی که اعمال محاسباتی را مستقیماً در سیستم اعداد دهدهی انجام می‌دهند، اعداد دهدهی را به فرم کد دودویی ارائه می‌کنند. یک جمع‌کننده در این نوع کامپیوترها ، از نوعی مدار محاسباتی استفاده می‌کند که اعداد دهدهی کد شده را می‌پذیرد و نتایج را در همان کد ارائه می‌نماید. برای جمع دودویی کافی است جفت بیت باارزش را همراه با رقم نقلی قبلی در نظر بگیرید. یک جمع‌کننده دهدهی به حداقل ده ورودی و پنج خروجی نیاز دارد زیرا برای کد هر رقم دهدهی چهار بیت لازم است و مدار باید ورودی و خروجی نقلی هم داشته باشد. برای انجام این‌گونه جمع، مدارهای جمع‌کننده دهدهی متعددی وجود دارند که انتخاب آنها به کد به کار رفته در نمایش ارقام دهدهی بستگی دارد. در اینجا ما جمع‌کننده دهدهی را برای کد BCD بررسی می‌کنیم.

جمع‌کننده BCD

جمع حسابی دو رقم دهدهی در BCD را همراه با یک رقم نقلی از مرحله قبل در نظر بگیرید. چون هر رقم ورودی از 9 تجاوز نمی‌کند، حاصل جمع خروجی از $19 = 1 + 9 + 9$ بیشتر نخواهد شد. عدد 1 در جمع فوق، نقلی ورودی است. فرض کنید که دو رقم BCD را به جمع‌کننده دودویی 4 بیتی اعمال نماییم. جمع‌کننده، حاصل جمع را به فرم دودویی اجرا می‌کند و نتیجهٔ تولید شده بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول (۵-۴) مشاهده می‌شود که با K ، z_8 ، z_4 ، z_2 و z_1 برچسب خورده‌اند. K یک رقم نقلی است و اندیس زیر حرف z وزن‌های 8 ، 4 ، 2 و 1 می‌باشند که به چهار بیت کد BCD تخصیص یافته‌اند. ستون زیر حاصل جمع دودویی، مقادیر دودویی ظاهر شده در خروجی‌های جمع‌کننده چهار بیت را نشان می‌دهد. حاصل جمع خروجی دو رقم دهدهی باید به فرم BCD درآید و نیز باید آن طور که در زیر ستون جمع BCD ملاحظه می‌شود ظاهر گردد. مسئله این است که برای تبدیل جمع دودویی به رقم BCD عدد که در ستون جمع BCD مشاهده می‌شود باید قانونی پیدا شود.

ضمن بررسی محتوای جدول، ملاحظه می‌شود که وقتی جمع دودویی برابر با یا کمتر از 1001 باشد، با عدد BCD نظیر خود برابر است، و بنابراین تبدیلی لازم نیست. وقتی جمع دودویی بزرگتر از 1001 باشد، نمایش بی‌اعتباری را برای BCD خواهیم داشت. افزایش دودویی 6 (0110) به جمع دودویی آن را به نمایش BCD صحیح تبدیل می‌کند، ضمن این که یک رقم نقلی نیز در صورت لزوم تولید خواهد کرد.

مدار منطقی برای تشخیص این اصلاح، می‌تواند از وارده‌های جدول حاصل گردد. واضح است که وقتی نقلی خروجی $K = 1$ باشد نیاز به اصلاح جمع دودویی وجود دارد. دیگر ترکیبات شش‌گانه از

جدول ۵-۴. طریقه طراحی جمع کننده BCD

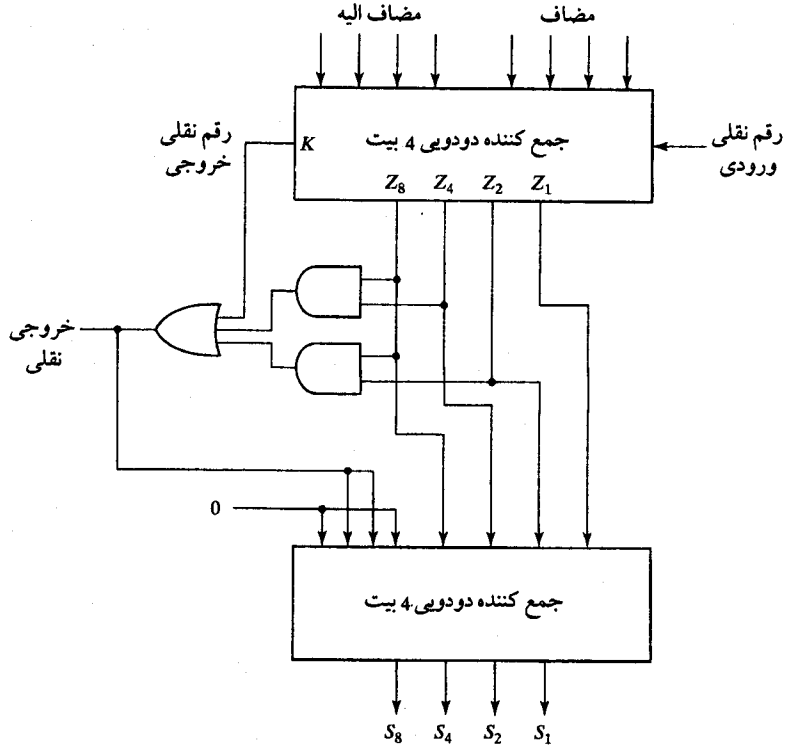
جمع دودویی					جمع BCD					دهدهی
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

1010 تا 1111 که به اصلاح نیاز دارند دارای 1 در مکان z_8 می‌باشند. برای تفکیک این شش حالت از 1000 و 1001، که آنها نیز دارای 1 در مکان z_8 هستند، به z_4 و z_2 مراجعه می‌کنیم که در هر حال حداقل یکی از آنها 1 است. به این ترتیب شرط اصلاح و داشتن یک نقلی خروجی را می‌توان با تابع بولی زیر بیان کرد:

$$C = K + Z_8Z_4 + Z_8Z_2$$

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

یک جمع‌کننده BCD که دو رقم BCD را با هم جمع کرده و ارقام جمع را به BCD نشان می‌دهد در شکل ۱۴-۴ ملاحظه می‌گردد. دو رقم دهدهی همراه با نقلی ورودی ابتدا در جمع‌کننده 4 بیت فوقانی جمع شده و حاصل جمع دودویی تولید می‌کنند. وقتی نقلی خروجی برابر 0 باشد، چیزی به جمع دودویی اضافه نمی‌شود. وقتی این نقلی برابر 1 باشد، عدد دودویی 0110 از طریق جمع‌کننده 4 بیت پایینی به جمع دودویی اضافه می‌گردد. نقلی خروجی تولید شده در جمع‌کننده پایین می‌تواند صرف‌نظر شود زیرا اطلاعاتی را حمل می‌کند که قبلاً در پایانه نقلی خروجی وجود داشته است. یک جمع‌کننده دهدهی موازی که n رقم دهدهی را جمع می‌کند به n طبقه جمع‌کننده BCD نیاز دارد. نقلی خروجی هر طبقه باید به ورودی طبقه بالاتر متصل گردد.



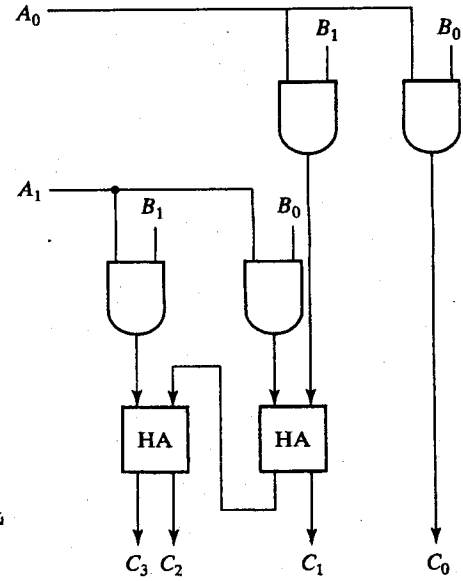
شکل ۱۴-۴. نمودار بلوکی یک جمع‌کننده BCD

۴-۶ ضرب دودویی

ضرب اعداد دودویی همچون ضرب اعداد دهدهی انجام می‌شود. هر بیت مضروب، در کم‌ارزش‌ترین بیت مضروب فیه ضرب می‌شود. چنین حاصلضربی، حاصلضرب جزیی خوانده می‌شود. حاصلضرب‌های جزیی هر بار یک مکان به چپ انتقال می‌یابند. حاصلضرب نهایی از جمع حاصلضرب‌های جزیی بدست می‌آید.

برای این که ببینیم که یک ضرب کننده چگونه با یک مدار ترکیبی پیاده می‌شود، ضرب اعداد دو بیت را طبق شکل ۱۵-۴ در نظر بگیرید. بیت‌های مضروب، B_0 و B_1 و بیت‌های مضروب‌فیه A_0 و A_1 و حاصلضرب $C_3C_2C_1C_0$ فرض می‌شوند. اولین حاصلضرب جزیی با ضرب A_0 در B_1B_0 حاصل می‌گردد. ضرب دو بیت مثل A_0 و B_0 هنگامی 1 تولید می‌کند که هر دوی آنها 1 باشند؛ در غیر این صورت 0 تولید خواهد کرد. این پاسخ مشابه با عمل AND است. بنابراین حاصل ضرب جزیی را می‌توان با گیت‌های AND مطابق شکل پیاده کرد. دومین حاصلضرب جزیی از ضرب A_1 در B_1B_0 بدست می‌آید که باید یک مکان هم به چپ جابجا شود. دو حاصلضرب جزیی به وسیله مدار دو نیم جمع‌کننده (HA) با هم جمع می‌شوند.

مضروب		B_1	B_0
مضروب فیه		A_1	A_0
		A_0B_1	A_0B_0
	A_1B_1	A_1B_0	
C_3	C_2	C_1	C_0



شکل ۱۵-۴. ضرب دودویی 2 بیت در 2 بیت

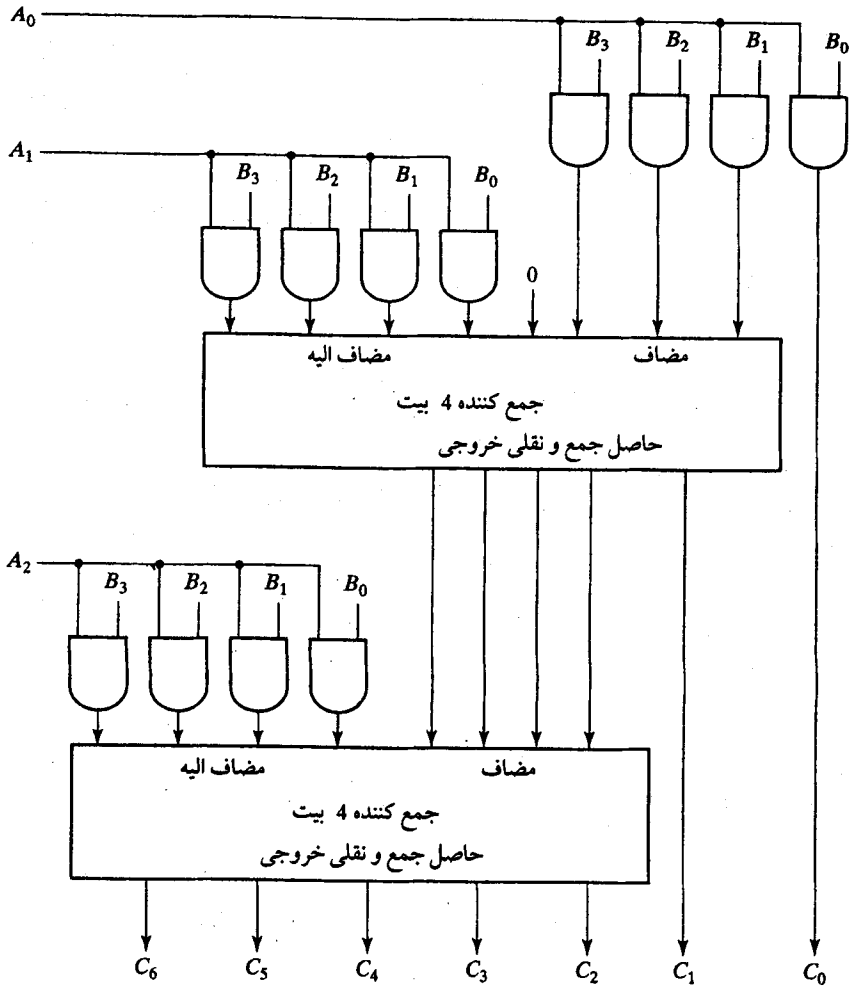
معمولاً در حاصلضرب‌های جزیی بیت‌های بیشتری وجود دارند و لازم است از تمام جمع‌کننده برای تولید جمع حاصلضرب‌های جزیی استفاده شود. توجه کنید لزومی ندارد که کم‌ارزش‌ترین بیت از جمع‌کننده عبور کند زیرا با خروجی اولین گیت AND، تشکیل شده است.

به طریقی مشابه می‌توان یک مدار ترکیبی ضرب دودویی با بیت‌های بیشتر ساخت. هر بیت از مضروب‌فیه در بیت‌های مضروب، AND می‌گردد. خروجی دودویی در هر سطحی از گیت‌های AND با حاصلضرب جزیی سطح قبلی برای تشکیل حاصلضرب جزیی جدید جمع می‌شود. آخرین سطح حاصلضرب کل را تولید می‌کند. برای J بیت مضروب فیه و K بیت مضروب به (J × K) گیت AND و (J−1) عدد جمع‌کننده K بیت نیاز است تا حاصلضرب J + K بیتی تولید شود.

به عنوان دومین مثال مدار ضرب کننده‌ای را ملاحظه نمایید که یک عدد دودویی 4 بیتی را در یک عدد 3 بیتی ضرب می‌کند. فرض کنید مضروب با $B_3B_2B_1B_0$ و مضروب‌فیه $A_2A_1A_0$ باشد. چون $K = 4$ و $J = 3$ است 12 گیت AND و دو جمع‌کننده 4 بیت برای تولید حاصلضرب 7 بیتی لازم است. نمودار منطقی ضرب کننده در شکل ۱۶-۴ دیده می‌شود.

۴-۷ مقایسه‌گر مقدار

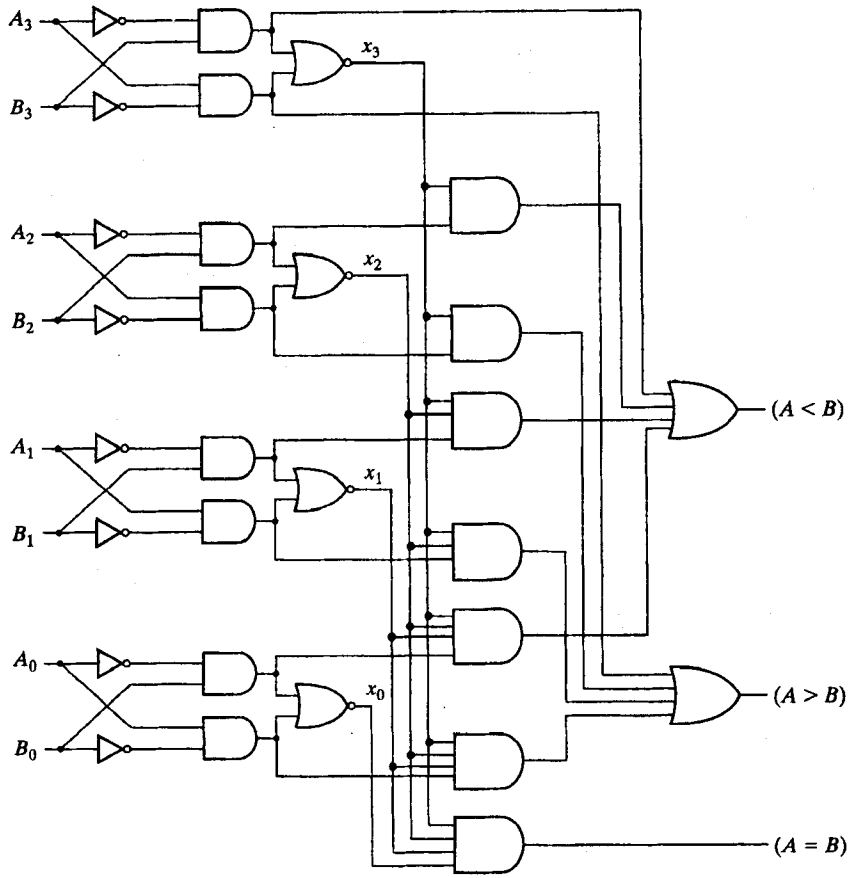
مقایسه دو عدد عملی است که توسط آن بزرگتر بودن، کوچکتر بودن یا مساوی بودن آنها معین می‌شود. یک مقایسه گر مقدار مداری ترکیبی است که دو عدد A و B را مقایسه می‌نماید و اندازه نسبی آنها را تعیین می‌کند. نتیجه این مقایسه با سه متغیر دودویی که بیانگر $A > B$ یا $A < B$ می‌باشد، مشخص می‌گردد.



شکل ۱۶-۴. ضرب کننده دودویی 4 بیت در 3 بیت

مدار مقایسه دو عدد n بیت 2^{2^n} وارده در جدول دارد و حتی با n=3 خسته کننده خواهد شد. از طرف دیگر، یک مدار مقایسه‌گر ممکن است مقدار قابل توجهی نظم در خود داشته باشد. توابع دیجیتال که ذاتاً دارای نظمی درونی هستند، معمولاً با روال‌های الگوریتمی قابل طراحی‌اند. یک الگوریتم روالی است که مجموعه مراحل معینی را مشخص می‌نماید و اگر دنبال شوند، از آن حلی برای مسئله حاصل می‌گردد. ما در اینجا از این روش برای ارائه یک الگوریتم جهت پیاده‌سازی مقایسه‌گر مقدار 4 بیتی استفاده خواهیم کرد.

در اینجا الگوریتم، کاربرد مستقیم روالی است که فرد برای مقایسه نسبی اندازه‌های دو عدد به کار می‌برد. دو عدد A و B را که هر کدام چهار رقم دارند در نظر بگیرید. ضرایب اعداد را به ترتیب نزولی زیر



شکل ۱۷-۴. مقایسه گر مقدرا چهار بیتی

هم می‌نویسیم:

$$A = A_3A_2A_1A_0$$
$$B = B_3B_2B_1B_0$$

هر حرف اندیس‌دار یک رقم را در عدد نشان می‌دهد. دو عدد هنگامی مساوی‌اند که همه جفت ارقام متناظر با هم برابر باشند: یعنی $A_3 = B_3$، $A_2 = B_2$، $A_1 = B_1$ و $A_0 = B_0$. وقتی که اعداد دودویی باشند، ارقام 1 یا 0اند و رابطه تساوی هر جفت بیت به طور منطقی با یک تابع *XOR* نمایش داده می‌شود.

$$x_i = A_iB_i + A_i'B_i' \qquad \text{for } i = 0, 1, 2, 3$$

که در آن $x_i = 1$ به شرطی صحت دارد که بیت‌های مکان iام برابر باشند (یعنی اگر هر دو 1 یا هر دو 0).

برابری دو عدد A و B در مدار ترکیبی با یک متغیر خروجی و با علامت (A = B) نشان داده می‌شود. این متغیر دودویی هنگامی 1 است که همه اعداد ورودی A و B مساوی باشند، در غیر این صورت 0 است. برای این که شرایط برابری برقرار باشد همه متغیرهای x_i باید برابر 1 شوند. در این

صورت AND همه متغیرها دیکته خواهد شد:

$$(A = B) = x_3 x_2 x_1 x_0$$

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

برای این که معین کنیم آیا A بزرگتر یا کوچکتر از B است، اندازه‌های نسبی دو رقم را با شروع از باارزش‌ترین مکان آغاز می‌نماییم. اگر دو رقم مساوی باشند، دو رقم پایین‌تر را مقایسه می‌کنیم. این مقایسه تا رسیدن به یک جفت غیرمساوی ادامه خواهد داشت. اگر در این هنگام بیت متعلق به A برابر 1 و B برابر 0 باشد، نتیجه می‌گیریم A > B است. اگر برعکس رقم مربوط به A برابر 0 و B برابر با 1 باشد، خواهیم داشت A < B. مقایسه فوق را می‌توان با کمک دو تابع بولی به صورت زیر نوشت:

$$(A > B) = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$
$$(A < B) = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

سمبل‌های (A > B) و (A < B) متغیرهای خروجی دودویی هستند که بترتیب هنگام A > B یا A<B برابر 1 می‌شوند.

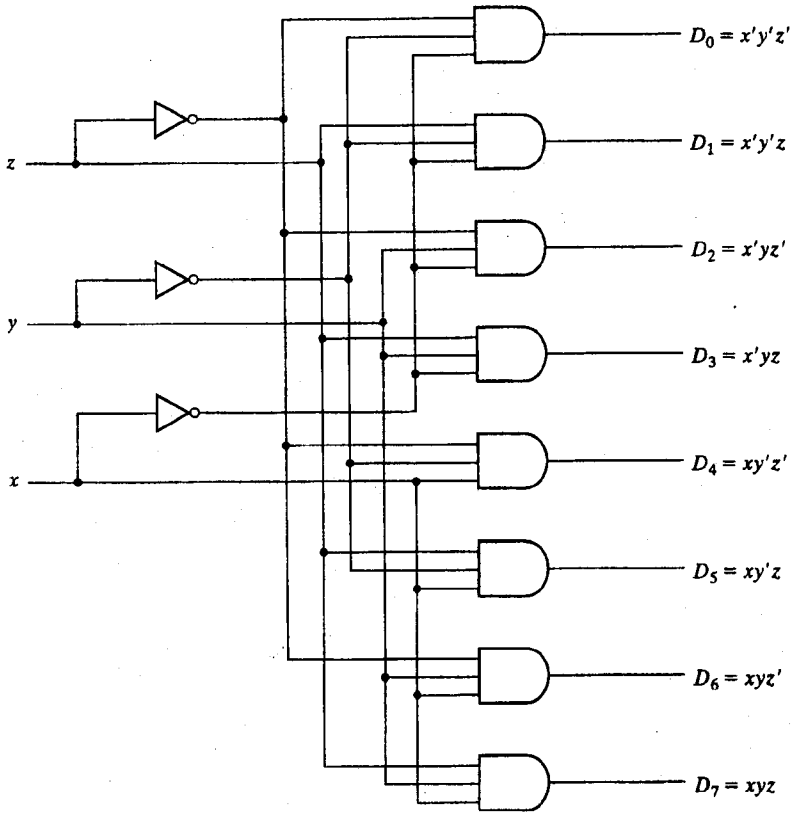
پیاده‌سازی گیتی سه متغیر خروجی ساده‌تر از آنچه به نظر می‌رسد انجام می‌شود زیرا شامل مقدار قابل توجهی اعمال تکراری است. خروجی‌های نامساوی می‌توانند از گیت‌هایی که برای تولید خروجی مساوی لازم بود استفاده کنند. نمودار منطقی یک مقایسه‌گر مقدار 4 بیتی در شکل ۱۷-۴ ملاحظه می‌شود. چهار خروجی x با مدارهای XNOR تولید شده و به گیت AND اعمال شده‌اند تا متغیر دودویی خروجی (A = B) تولید گردد. دو خروجی دیگر از متغیر x برای تولید توابع بولی لیست شده قبلی استفاده می‌کنند. این یک پیاده‌سازی چند طبقه است که الگوی منظمی دارد. روال برای بدست آوردن مدارهای مقایسه‌گر اندازه برای اعداد دودویی با بیش از چهار بیت از این مثال کاملاً آشکار است.

۸-۴ دیکدرها

کمیت‌های گسسته اطلاعاتی در سیستم‌های دیجیتال با کدهای دودویی نشان داده می‌شوند. یک کد دودویی n بیتی قادر است تا 2^n عنصر گسسته اطلاعات کد شده را نشان دهد. یک دیکدر مداری ترکیبی است که اطلاعات دودویی را از n خط ورودی به حداکثر 2^n خط خروجی منحصر به فرد تبدیل می‌کند اگر کد n بیتی دارای ترکیبات بی‌استفاده باشد، دیکدر ممکن است خروجی‌های کمتر از 2^n داشته باشد.

دیکدرهایی که در اینجا ارائه شده‌اند دیکدرهای n به m خوانده می‌شوند که $m \leq 2^n$ است. هدف از آنها تولید 2^n مینترم (یا کمتر) از n متغیر ورودی است. نام دیکدر همراه با دیگر مبدل‌های کد مانند دیکدر BCD به هفت قسمتی هم به کار می‌رود.

به عنوان مثال دیکدر 3 به 8 قسمتی شکل ۱۸-۴ را ملاحظه نمایید. سه ورودی به هشت خروجی دیکدر شده است که هر یک نمایشگر یکی از مینترم‌های متعلق به سه متغیر ورودی است. سه وارونگر، متمم ورودی‌ها را تهیه کرده و هشت گیت AND هر کدام یک مینترم تولید می‌کنند. کاربرد رایج این نوع دیکدر، تبدیل دودویی به هشت هشتی است. متغیرهای ورودی یک عدد دودویی را نشان می‌دهند، و



شکل ۱۸-۴. دیکدر 3 به 8

خروجی بیانگر هشت رقم در سیستم اعداد مبنای هشت است. با این وجود دیکدر 3 به 8 خط را می‌توان برای دیکد کردن هر کد 3 بیت در تولید هشت خروجی، یکی برای هر عنصر از کد، به کار برد.

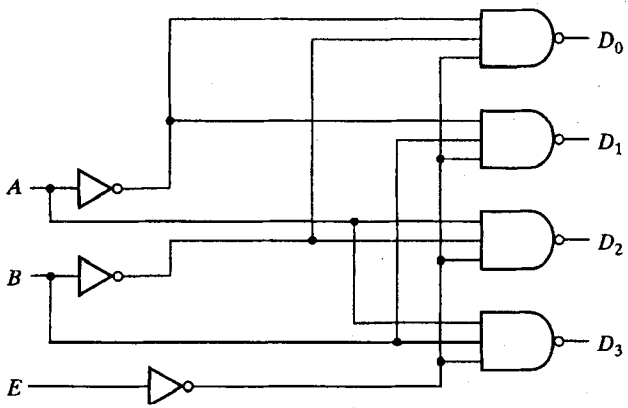
طرز کار یک دیکدر می‌تواند با لیستی در جدول درستی ۶-۴ آشکار شود. برای هر ترکیب ورودی

جدول ۶-۴. جدول درستی دیکدر 3 به 8 خط

ورودی‌ها			خروجی ها							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

(ب) جدول درستی



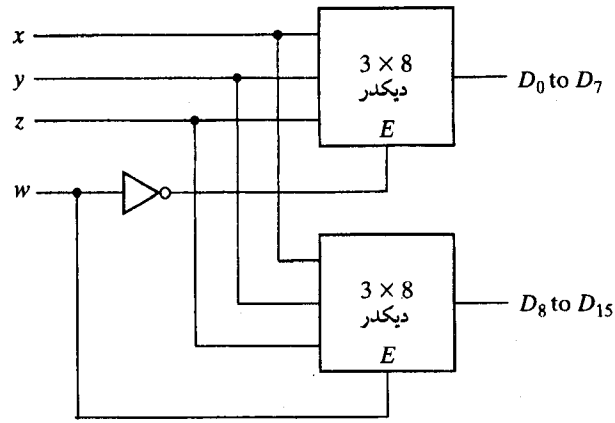
(الف) دیاگرام منطقی

شکل ۱۹-۴. دیکدر 2 به 4 خط با ورودی فعال ساز

ممکن، هفت خروجی وجود دارد که برابر 0 هستند و فقط یکی از آنها برابر 1 است. خروجی حاوی 1 بیانگر مینترم عدد دودیی حاضر در خطوط ورودی است.

بعضی از دیکدرها با گیت‌های NAND ساخته می‌شوند. چون گیت NAND عمل AND را با یک خروجی معکوس تولید می‌کند، تولید مینترم‌های دیکدر در شکل متمم اقتصادی‌تر است. به علاوه، دیکدر معمولاً دارای یک یا دو ورودی تواناساز یا فعال‌ساز برای کنترل کار مدار می‌باشند. یک دیکدر 2 به 4 با یک ورودی فعال‌ساز که با گیت‌های NAND ساخته شده در شکل ۱۹-۴ دیده می‌شود. مدار با خروجی‌های متمم شده و یک ورودی فعال‌ساز متمم شده کار می‌کند. دیکدر هنگامی که E برابر 0 باشد فعال می‌گردد. همانطور که توسط جدول درستی مشاهده می‌شود، هر بار تنها یک خروجی برابر 0 بوده و دیگر خروجی‌ها در وضعیت 1 قرار دارند. وقتی $E = 1$ باشد مدار غیرفعال است و به دو ورودی دیگر بستگی ندارد. هنگام غیرفعال شدن مدار، هیچ یک از خروجی‌ها در 0 نبوده و هیچ یک از مینترم‌ها انتخاب نمی‌شوند. به طور کلی، یک دیکدر ممکن است خروجی‌های متمم شده یا متمم نشده داشته باشد. ورودی فعال‌ساز ممکن است با سیگنال 0 یا 1 فعال گردد. بعضی از دیکدرها دارای دو یا چند ورودی فعال‌ساز می‌باشند که باید یک شرط منطقی مفروضی را برآورده سازند تا مدار فعال شود.

یک دیکدر با ورودی فعال‌ساز می‌تواند به عنوان یک دی‌مولتی پلکسر عمل کند. دی‌مولتی پلکسر مداری است که اطلاعات را از یک خط دریافت کرده و آن را به یکی از 2^n خط خروجی ممکن هدایت می‌نماید. انتخاب یک خروجی خاص با ترکیب بیتی n خط انتخاب صورت می‌گیرد. دیکدر شکل ۱۹-۴ را می‌توان به عنوان یک دی‌مولتی پلکسر 1 به 4 به کار برد. در این مدار E به عنوان ورودی داده و A و B ورودی‌های انتخاب هستند. تنها متغیر ورودی E مسیری به تمام چهارخروجی دارد، ولی اطلاعات ورودی تنها به یکی از خروجی‌ها هدایت می‌شود. این خروجی با ترکیب دودویی دو خط



شکل ۲۰-۴. دیکدر 16 × 4 ساخته شده با دو دیکدر 8 × 3

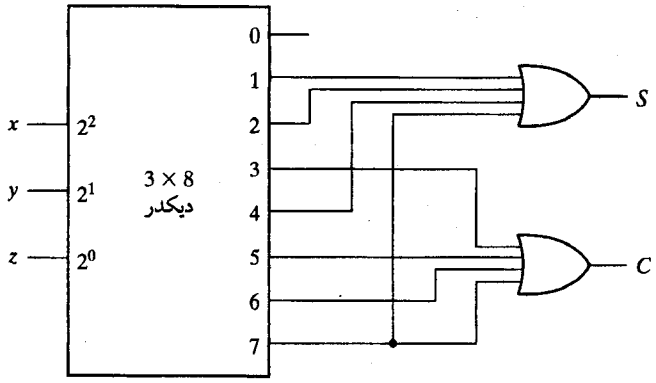
انتخاب A و B انتخاب می‌گردد. می‌توان انتخاب مسیر را از جدول درستی تحقیق کرد. مثلاً اگر خطوط انتخابی AB = 10 باشند، خروجی D_2 مثل ورودی E خواهد بود در حالی که دیگر خروجی‌ها در 1 نگهداشته خواهند شد. چون عمل دیکدر و دی‌مولتی پلکسر با استفاده از یک مدار حاصل می‌شود، یک دیکدر با ورودی فعال‌ساز را دیکدر/ دی‌مولتی پلکسر هم می‌خوانند.

برای تهیه مدار دیکدر بزرگتر می‌توان دیکدرها با ورودی‌های فعال‌ساز را به هم متصل کرد. شکل ۲۰-۴ دو دیکدر 3 به 8 را با ورودی‌های فعال‌ساز به هم پیوسته برای تشکیل یک دیکدر 4 به 16 خط نشان می‌دهد. وقتی $w = 0$ است، دیکدر فوقانی فعال می‌شود و دیگری غیرفعال است. خروجی‌های دیکدر پایینی همگی در 0 خواهند بود و هشت خروجی بالایی مینترم‌های 0000 تا 0111 را تولید می‌کنند. وقتی $w = 1$ باشد، وضعیت فعال شدن معکوس میگردد. خروجی‌های دیکدر پایینی مینترم‌های 1000 تا 1111 را تولید می‌نمایند، در حالی که خروجی‌های فوقانی همه 0 هستند. این مثال حسن ورودی‌های فعال‌ساز را در دیکدرها و دیگر قطعات منطقی ترکیبی نشان می‌دهد. به طورکلی ورودی‌های فعال‌ساز ابزارهای مناسبی برای اتصالات درونی دو یا چند قطعه استاندارد برای گسترش آنها با عملکردی مشابه و ورودی‌ها و خروجی‌های بیشتر است.

پیاده‌سازی مدار منطقی ترکیبی

یک دیکدر، 2^n مینترم را برای n متغیر ورودی تهیه می‌کند. چون هر تابع بولی می‌تواند برحسب جمع مینترم‌ها بیان شود، می‌توان از دیکدر برای تولید مینترم استفاده کرده و با یک گیت OR بیرونی جمع منطقی آنها را تشکیل داد. به این ترتیب هر مدار ترکیبی n ورودی و m خروجی با یک دیکدر n به 2^n و m گیت OR قابل پیاده‌سازی است.

روال پیاده‌سازی یک مدار ترکیبی با دیکدر و گیت‌های OR لازم می‌دارد که تابع بول مدار برحسب



شکل ۲۱-۴. پیاده‌سازی یک جمع‌کننده کامل با دیکدر

جمع مینترم‌ها بیان شود. سپس یک دیکدر برای تولید همه مینترم‌های حاصل از متغیرهای ورودی انتخاب می‌گردد. ورودی‌های هر گیت OR از خروجی‌های دیکدر برحسب لیست مینترم هر تابع انتخاب می‌گردند. این روال با مثالی که مدار جمع‌کننده کامل را به کار می‌برد تشریح می‌گردد.

با توجه به جدول درستی جمع‌کننده کامل (جدول ۴-۴)، توابع مدار ترکیبی را به صورت مجموع مینترم‌ها بدست می‌آوریم:

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

چون سه ورودی و جمعاً هشت مینترم وجود دارد، به یک دیکدر 3 به 8 خط احتیاج است. پیاده‌سازی در شکل ۲۱-۴ ملاحظه می‌گردد. دیکدر هشت مینترم را برای x، y و z تولید می‌کند. گیت OR برای خروجی S، جمع منطقی مینترم‌های 1، 2، 4 و 7 را تشکیل می‌دهد. گیت OR جمع منطقی مینترم‌های 3، 5، 6 و 7 را برای تولید خروجی C بکار می‌برد.

یک تابع با لیست طویلی از مینترم‌ها نیاز به یک گیت OR با ورودی‌های متعدد دارد. تابعی که k مینترم دارد می‌تواند به فرم متمم خود، F′، با $2^n - k$ مینترم نشان داده شود. اگر تعداد مینترم‌های موجود در تابع بزرگتر از $2^n/2$ باشد، آنگاه می‌توان F′ را با تعداد مینترم کمتری بیان کرد. در چنین وضعیتی، استفاده از گیت NOR برای تشکیل جمع مینترم‌های F′ مزیت دارد. خروجی گیت NOR این جمع را متمم کرده و تولید خروجی نرمال F را خواهد کرد. اگر از گیت‌های NAND، مثل شکل ۱۹-۴ استفاده شود، آنگاه گیت‌های خروجی در عوض OR باید از نوع NAND باشند. دلیل این است که یک مدار گیتی دو طبقه NAND تابع جمع مینترم‌ها را پیاده‌سازی می‌کند و معادل با مدار دو طبقه AND-OR است.

۹-۴ انکدرها

یک انکدر مداری است که عمل عکس یک دیکدر را انجام می‌دهد. یک انکدر دارای 2^n (یا کمتر) خط ورودی و n خط خروجی است. خطوط خروجی کد دودویی مربوط به مقدار دودویی ورودی را

تولید می‌نمایند.

مثالی از یک انکدر، انکدر هشت هشتی به دودویی است که جدول درستی آن در جدول ۷-۴ داده شده است. این مدار دارای هشت ورودی (یک ورودی برای هر رقم هشت هشتی) و سه خروجی است که عدد دودویی مربوطه را تولید می‌نماید. فرض بر این است که در هر لحظه‌ای از زمان تنها یک ورودی مقدار 1 را داشته باشد.

انکدر را می‌توان با گیت‌های OR که ورودی‌هایشان مستقیماً از جدول درستی تهیه می‌شود، پیاده‌سازی کرد. خروجی z هنگامی 1 است که رقم هشت هشتی ورودی در 1، 3، 5 یا 7 برابر 1 باشد. خروجی y به ازاء ارقام هشت هشتی ورودی 2، 3، 6 و 7، برابر 1 می‌شود. این شرایط را می‌توان با معادلات بولی خروجی بیان کرد.

$$z = D_1 + D_3 + D_5 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$x = D_4 + D_5 + D_6 + D_7$$

این انکدر با سه گیت OR قابل پیاده‌سازی است.

انکدری که در جدول ۷-۴ تعریف شد دارای این محدودیت است که در آن در هر لحظه از زمان فقط یک ورودی فعال می‌باشد. اگر دو ورودی به طور همزمان فعال شوند، خروجی ترکیبات نامفهومی را تولید خواهد کرد. مثلاً اگر D_3 و D_6 همزمان برابر 1 شوند، خروجی انکدر 111 خواهد شد زیرا در این حالت هر سه خروجی برابر 1 است. این خروجی نه 3 و نه 6 را نمایش می‌دهد. برای حل این مشکل، مدارهای انکدر باید اولویتی در ورودی ایجاد کنند تا مطمئن شویم که فقط یک ورودی انکد شده است. اگر اولویت بالاتر را با اندیس‌های بالاتر ایجاد نماییم، و اگر در یک زمان D_6 و D_3 برابر 1 شوند، خروجی 110 می‌شود زیرا، D_6 اولویت بالاتری نسبت به D_3 دارد.

مشکل دیگری که در انکدر هشت هشتی به دودویی وجود دارد این است که در آن یک خروجی تمام 0 به ازاء حالتی که همه ورودی‌ها 0 هستند تولید می‌شود. این خروجی برابر با حالتی است که در آن $D_0 = 1$ است. ایراد را می‌توان با تهیه یک خروجی بیشتر حل کرد تا به این ترتیب نشان دهد که حداقل یک ورودی 1 است.

جدول ۷-۴. جدول درستی یک انکدر هشت هشتی به دودویی

ورودی‌ها								خروجی‌ها		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

جدول ۸-۴ . جدول درستی انکدر اولویت

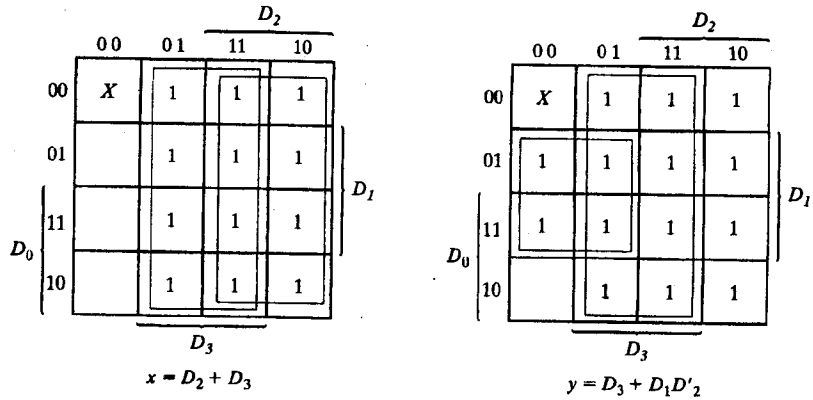
ورودی‌ها				خروجی ها		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

انکدر اولویت

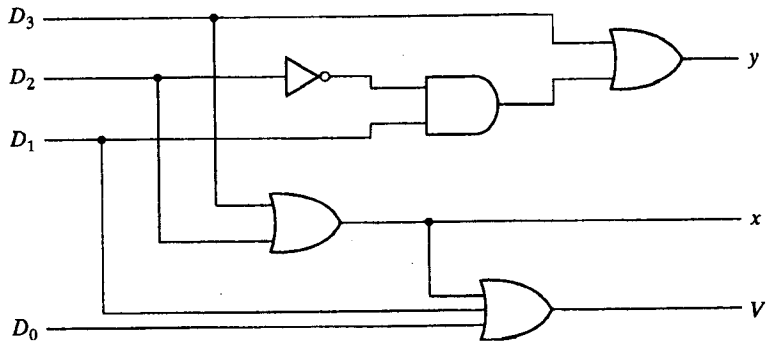
انکدر اولویت مداری است که تابع اولویت‌دهی را در خود دارد. عملکرد این نوع انکدر چنان است که اگر دو یا چند ورودی به طور همزمان برابر 1 شوند، ورودی با اولویت بالاتر پیش خواهد افتاد. جدول درستی یک انکدر با تقدم چهار ورودی در جدول ۸-۴ ملاحظه می‌گردد. علاوه بر دو خروجی x و y، مدار دارای سومین خروجی با علامت V است؛ که به معنی بیت معتبر می‌باشد و هرگاه یک یا چند ورودی برابر 1 شوند این خروجی 1 می‌گردد. اگر همه ورودی‌ها 0 باشند بیت معتبر وجود نخواهد داشت و $V = 0$ است. وقتی $V = 0$ باشد در خروجی دیگر وارسی نمی‌شوند و حالت بی‌اهمیت را خواهند داشت. توجه کنید که گرچه حالات بی‌اهمیت در ستون‌های خروجی اهمیت ندارند ولی X ها در ستون ورودی‌ها درکاهش جدول درستی نقش عمده‌ای دارند. به جای ذکر هر 16 مینترم برای چهار متغیر، جدول درستی از X که می‌تواند 0 یا 1 باشد استفاده می‌کند. مثلاً X100 دو مینترم 1100 ، 0100 را می‌پوشاند.

طبق جدول ۸-۴، هر عدد با مقدار بالاتر اولویت بالاتری را داراست. در این جدول D_3 اولویت بالاتری دارد، بنابراین بدون توجه به مقادیر دیگر ورودی‌ها، وقتی این ورودی 1 شود، خروجی xy برابر 11 می‌گردد (دودویی 3). D_2 دارای اولویت بعدی است. اگر $D_2 = 1$ شود خروجی 10 می‌گردد به شرطی که $D_3 = 0$ باشد، ولی این خروجی مستقل از دو ورودی با اولویت پایین‌تر است. اگر دیگر ورودی‌های با اولویت‌تر 0 باشند، خروجی مربوط به D_1 تولید می‌گردد و به همین ترتیب.

نقشه ساده‌سازی خروجی‌های x و y در شکل ۲۲-۴ نشان داده شده است. مینترم‌های دو تابع از



شکل ۲۲-۴ . نقشه یک انکدر اولویت‌دار



شکل ۲۳-۴ . پیاده‌سازی تابع بول با یک مولتی پلکسر

جدول ۷-۴ استنتاج شده‌اند. گرچه جدول دارای تنها پنج سطر است، وقتی هر X در هر سطر ابتدا با صفر و سپس با 1 جایگزین شود، آنگاه تمام 16 ترکیب ورودی را خواهیم داشت. مثلاً سطر چهارم در جدول با XX10 چهار مینترم 0010 ، 0110 ، 1010 و 1110 را نمایش می‌دهد. عبارات بول ساده شده برای انکدر اولویت از نقشه‌ها حاصل می‌شود. شرط خروجی V یک تابع OR از همه متغیرهای ورودی است. انکدر اولویت بر طبق توابع بولی زیر در شکل ۲۳-۴ پیاده‌سازی شده است.

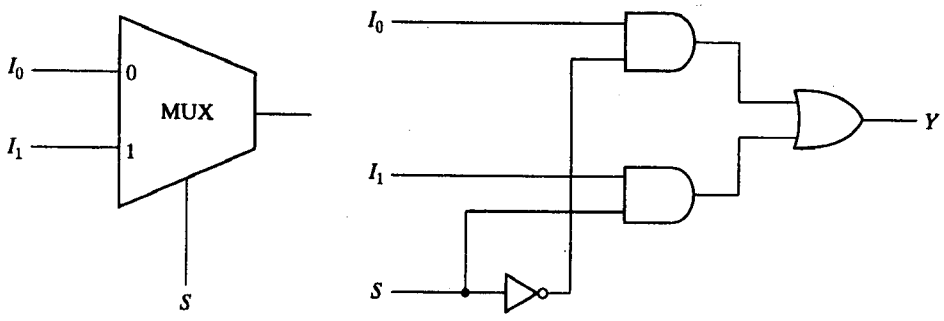
$$x = D_2 + D_3$$
$$y = D_3 + D_1 D_2'$$
$$V = D_0 + D_1 + D_2 + D_3$$

۴-۱۰ مولتی پلکسرها

یک مولتی پلکسر مداری ترکیبی است که اطلاعات دودویی را از تعدادی خط ورودی دریافت کرده و آنها را به یک خط خروجی هدایت می‌نماید. انتخاب یک ورودی خاص به وسیله مجموعه‌ای از خطوط انتخاب انجام می‌شود. معمولاً 2^n خط ورودی و n خط انتخاب وجود دارد و ترکیب بیتی تعیین کننده ورودی انتخاب شده است.

یک مولتی پلکسر 2 به 1 یکی از دو منبع 1 بیت را طبق شکل ۲۴-۴ به یک مقصد مشترک متصل می‌کند. مدار دارای دو خط ورودی داده، یک خروجی و یک خط انتخاب S است. وقتی S = 0 باشد، گیت AND فوقانی فعال شده و I_0 به خروجی راه می‌یابد. وقتی S = 1 باشد، گیت AND تحتانی فعال شده و I_1 به خروجی متصل می‌شود. مولتی پلکسر مثل یک کلید الکترونیک عمل کرده و یکی از دو منبع را انتخاب می‌نماید. نمودار بلوکی یک مولتی پلکسر گاهی به شکل ذوزنقه شکل ۲۴-۴ (ب) نشان داده می‌شود. این مدار چگونگی انتخاب و هدایت منابع متعدد داده را به یک مقصد نشان می‌دهد. مولتی پلکسر اغلب با نمودارهای بلوکی و کلمه MUX نشان داده می‌شود.

یک مولتی پلکسر 4 به 1 در شکل ۲۵-۴ دیده می‌شود. هر یک از چهار ورودی I_0 تا I_3 به یک



(ب) نمودار بلوکی

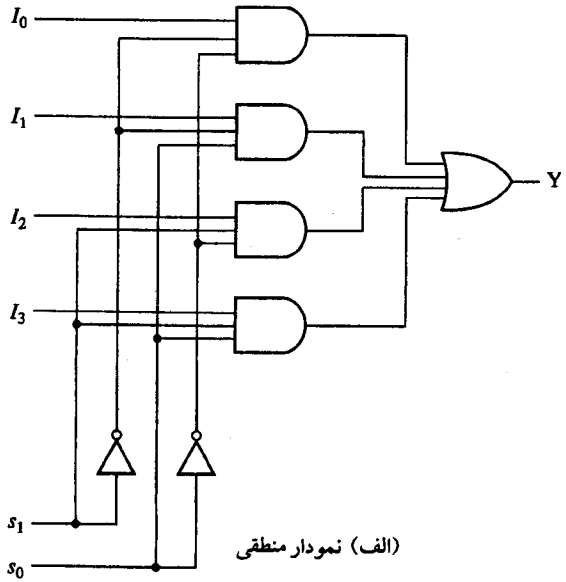
(الف) نمودار منطقی

شکل ۲۴-۴. مولتی‌پلکسر 2 به 1

ورودی گیت AND اعمال می‌شود. خطوط انتخاب S_1 و S_0 برای انتخاب گیت AND خاص دیکد می‌شوند. خروجی گیت‌های AND به یک گیت OR اعمال می‌شوند تا خروجی 1 خط را ایجاد کنند. جدول تابع، ورودی راکه از مولتی پلکسر عبور کرده نشان می‌دهد. برای نمایش عمل مدار، حالتی راکه $S_1S_0 = 10$ است ملاحظه کنید. گیت مربوط به ورودی I_2 دارای دو ورودی 1 و یک ورودی متصل به I_2 است. سه گیت دیگر هر یک حداقل یک 0 در ورودی خود دارند و بنابراین خروجی‌شان 0 می‌شود. خروجی گیت OR ، اکنون برابر مقدار I_2 است و به این ترتیب مسیری از ورودی انتخابی به خروجی ایجاد شده است. یک مولتی پلکسر را انتخابگر داده هم می‌خوانند، زیرا یکی از چند ورودی را انتخاب کرده و اطلاعات دودویی را به خط خروجی هدایت می‌کند.

s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(ب) جدول تابع

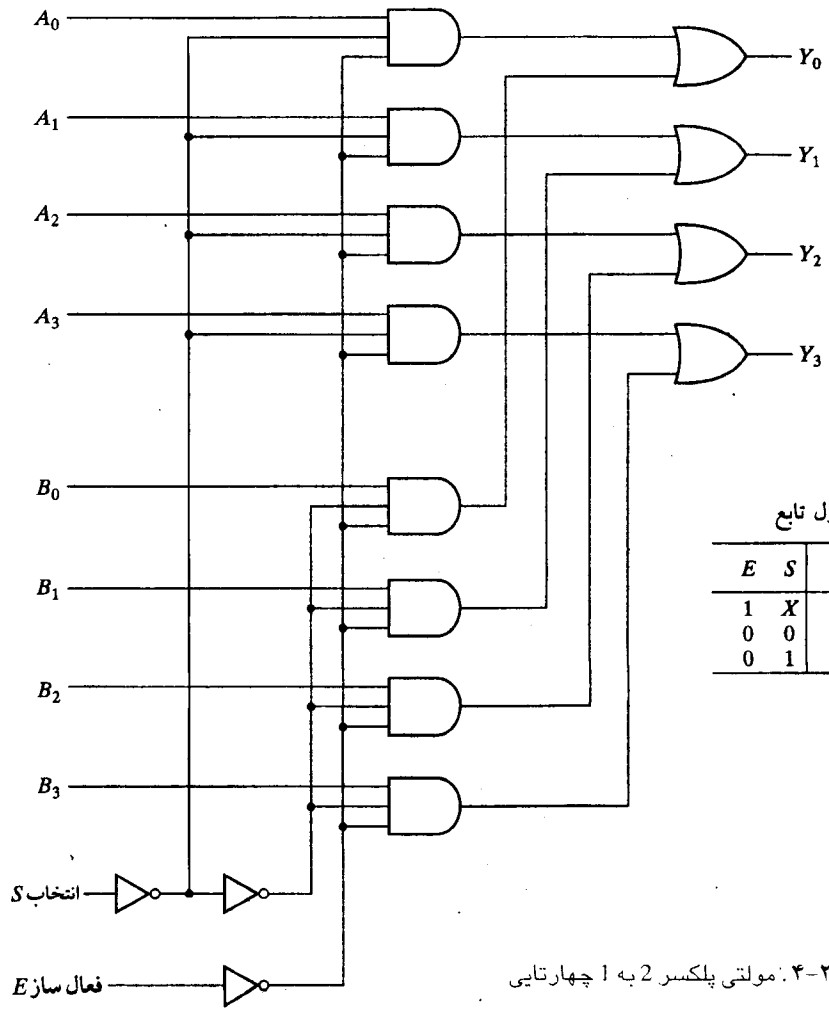


(الف) نمودار منطقی

شکل ۲۵-۴. مولتی پلکسر 4 به 1

وجود گیت‌های AND و وارونگرها در مولتی پلکسر، مدار دیکدر را به خاطر می‌آورد، و به علاوه آنها خطوط ورودی انتخاب را دیکد می‌کنند. به طور کلی یک مولتی پلکسر 2^n به 1 از یک دیکدر n به 2^n، ساخته شده که در آن 2^n خط به 2^n گیت AND، یعنی هر خط به یک گیت وصل شده است. خروجی گیت‌های AND به تنها گیت OR اعمال می‌گردند. سایز مولتی پلکسر با 2^n خط ورودی داده و تنها خط خروجی‌اش مشخص می‌شود. همچون دیکدر، مولتی پلکسرها هم ممکن است خط فعال‌سازی داشته باشند تا عملکرد کل قطعه را کنترل کنند. وقتی که ورودی فعال‌ساز در وضعیت غیرفعال قرار دارد، خروجی‌ها غیرفعالند، و وقتی در حالت فعال خود قرار گیرد، مدار به عنوان یک مولتی پلکسر معمولی عمل می‌کند.

مدارهای مولتی پلکسر را می‌توان برای تهیه مولتی پلکسر چند بیتی با هم ترکیب کرد. به منظور تشریحی بر این مطلب، یک مولتی پلکسر 2 به 1 چهارتایی در شکل ۲۶-۴ نشان داده شده است. مدار



جدول تابع

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

شکل ۲۶-۴: مولتی پلکسر 2 به 1 چهارتایی

دارای چهار مولتی پلکسر است که هر یک قادر است یکی از دو خط را انتخاب نماید. خروجی Y_0 می‌تواند به یکی از ورودی‌های A_0 یا B_0 وصل شود و به طور مشابه Y_1 هم می‌تواند مقدار A_1 یا B_1 را داشته باشند، و به همین ترتیب. خط انتخاب ورودی یکی از خطوط ورودی را در هر یک از چهار مولتی پلکسر انتخاب می‌کند. خط فعال‌ساز E باید به هنگام کار معمولی فعال شود. گرچه مدار حاوی چهار مولتی پلکسر 2 به 1 است، ولی ما بیشتر علاقمندیم به آن به عنوان مداری که یکی از دو مجموعه چهار بیتی خطوط داده را انتخاب می‌کند، بنگریم. همانطور که در جدول تابع دیده می‌شود، مدار وقتی که $E = 0$ است فعال می‌شود. آنگاه اگر $S = 0$ باشد چهار ورودی A مسیری به چهار خروجی دارند. از طرف دیگر اگر $S = 1$، چهار ورودی B به خروجی‌ها اعمال می‌شوند، وقتی $E = 1$ باشد، مستقل از وضعیت S، همه خروجی‌ها 0 خواهند بود.

پیاده‌سازی تابع بول

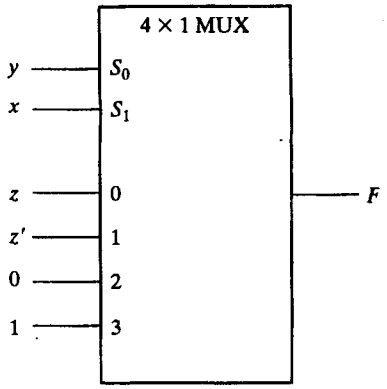
در بخش ۸-۴ دیده شد که با افزودن یک گیت OR به خروجی‌های یک دیکدر می‌توان از آن برای پیاده‌سازی توابع بول استفاده کرد. با بررسی نمودار منطقی یک مولتی پلکسر ملاحظه می‌شود که این مدار در واقع همان دیکدر است که یک گیت OR به آن اضافه شده است. مینترم‌های یک تابع با خطوط انتخاب مولتی پلکسر تولید می‌شوند. هر مینترم به وسیله ورودی‌های داده انتخاب می‌شود. این مطلب روشی را برای پیاده‌سازی هر تابع n متغیره به وسیله یک مولتی پلکسر که دارای 2^n ورودی داده و n خط ورودی انتخاب است، فراهم می‌سازد.

اکنون روش کاراتری را برای پیاده‌سازی یک تابع بول n متغیره با مولتی پلکسری که $n-1$ ورودی انتخاب دارد، معرفی می‌نماییم. ابتدا $n-1$ متغیر به ورودی‌های انتخاب مولتی پلکسر وصل می‌شود. تنها متغیر باقیمانده تابع برای ورودی‌های داده مورد استفاده قرار می‌گیرد. اگر متغیر باقیمانده را z بنامیم هر ورودی داده مولتی پلکسر برابر z، z'، 1 و یا 0 خواهد بود. برای نمایش این رویه، تابع سه متغیره زیر را ملاحظه کنید:

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

تابع را می‌توان مطابق شکل ۲۷-۴ با یک مولتی پلکسر 4 به 1 پیاده‌سازی کرد. دو متغیر x و y به خطوط انتخاب وصل می‌شود، به این ترتیب که x به ورودی s_1 و y به ورودی s_0 متصل می‌گردد. مقادیر خطوط ورودی داده از جدول درستی تابع معین می‌شود. وقتی $xy = 00$ باشد خروجی F برابر z است زیرا وقتی $z = 0$ است F هم برابر 0 می‌باشد و وقتی $z = 1$ شود F نیز برابر 1 می‌گردد. این وضع لازم می‌دارد تا متغیر z به ورودی داده 0 متصل شود. عملکرد مولتی پلکسر به نحوی است که وقتی $xy = 00$ گردد، خط داده شماره 0 به خروجی وصل شده F را برابر z می‌نماید. به طریقی مشابه می‌توان نشان داد که خطوط ورودی 1، 2 و 3 وقتی که $xy = 01, 10$ و 11 است به ترتیب به F وصل می‌شوند. این مثال خاص هر چهار حالت ممکن را برای ورودی‌های داده تهیه می‌کند.

روالی کلی برای پیاده‌سازی هر تابع بول n متغیره با یک مولتی پلکسر که $n-1$ خط ورودی انتخاب



(ب) پیاده سازی مولتی پلکسر

x	y	z	F	
0	0	0	0	$F = z$
0	0	1	1	
0	1	0	1	$F = z'$
0	1	1	0	
1	0	0	0	$F = 0$
1	0	1	0	
1	1	0	1	$F = 1$
1	1	1	1	

(الف) جدول درستی

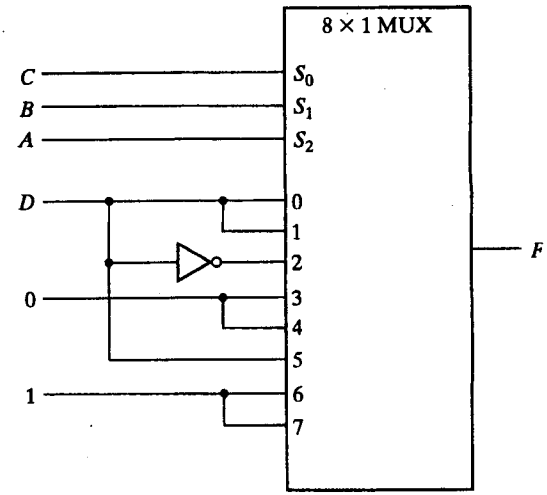
شکل ۲۷-۴ . پیاده سازی یک تابع بول با یک مولتی پلکسر

و 2^{n-1} ورودی داده دارد از مثال قبل نتیجه می‌گردد. ابتدا تابع بول را در جدول درستی لیست می‌کنیم. اولین n−1 متغیر در جدول به ورودی‌های انتخاب مولتی پلکسر وصل می‌شوند. برای هر ترکیبی از متغیرهای انتخاب، خروجی را به عنوان تابعی از آخرین متغیر ارزیابی می‌کنیم. این تابع می‌تواند 0، 1، متغیر و یا متمم متغیر باشد. آنگاه این مقادیر به ورودی‌های داده به نحوی صحیح اعمال می‌شوند. به عنوان دومین مثال، تابع بول زیر را ملاحظه نمایید:

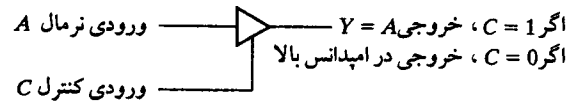
$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15).$$

این تابع با مولتی پلکسری که سه ورودی انتخاب دارد، مطابق شکل ۲۸-۴ پیاده‌سازی می‌شود. توجه کنید که اولین متغیر A باید به ورودی انتخاب S_2 و دو خط B و C به ترتیب به S_1 و S_0 وصل شوند.

A	B	C	D	F	
0	0	0	0	0	$F = D$
0	0	0	1	1	
0	0	1	0	0	$F = D$
0	0	1	1	1	
0	1	0	0	1	$F = D'$
0	1	0	1	0	
0	1	1	0	0	$F = 0$
0	1	1	1	0	
1	0	0	0	0	$F = 0$
1	0	0	1	0	
1	0	1	0	0	$F = D$
1	0	1	1	1	
1	1	0	0	1	$F = 1$
1	1	0	1	1	
1	1	1	0	1	$F = 1$
1	1	1	1	1	



شکل ۲۸-۴ . پیاده‌سازی یک تابع 4 ورودی با یک مولتی پلکسر



شکل ۲۹-۴. سمبل گرافیکی برای یک بافر سه حالته

مقادیر ورودی‌های داده از جدول درستی لیست شده در شکل معین می‌گردند. شماره خط داده مربوطه از ترکیب دودویی ABC حاصل می‌شود. مثلاً وقتی ABC = 101 باشد، جدول نشان می‌دهد که F = D است، بنابراین متغیر D به ورودی داده 5 وصل می‌شود. ثابت‌های دودویی 0 و 1 مربوط به دو مقدار سیگنال ثابت است. وقتی از مدارهای مجتمع استفاده کنیم، منطق 0 مربوط به سیگنال زمین و منطق 1 معادل با سیگنال تغذیه است که معمولاً 5 ولت می‌باشد.

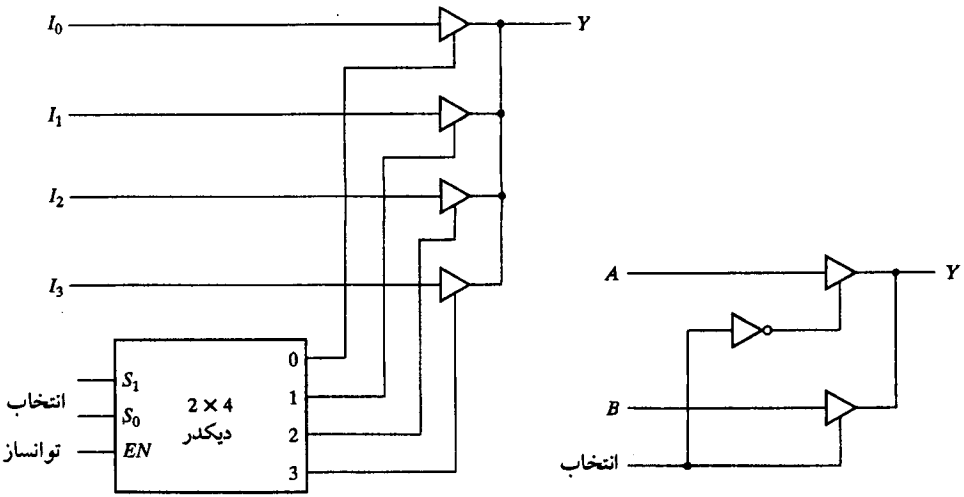
گیت‌های سه حالته

یک مولتی پلکسر را می‌توان با گیت‌های سه حالته ساخت. یک گیت سه حالته مداری دیجیتال است که سه حالت را از خود به نمایش می‌گذارد. دو حالت، همچون گیت‌های معمولی همان منطق 1 و 0 است. حالت سوم، حالت امپدانس بالاست. حالت امپدانس بالا مثل مدار باز عمل می‌کند و به این معنی است که خروجی از درون قطع بوده و مدار دارای مفهوم منطقی باارزشی نیست. گیت‌های سه حالته ممکن است به عنوان گیت‌های AND و NAND نیز عمل کنند. با این وجود اغلب به عنوان بافر مورد استفاده قرار می‌گیرند.

نمودار گرافیکی یک گیت بافر سه حالته در شکل ۲۹-۴ دیده می‌شود. این قطعه با ورودی کنترلی که وارد ضلع پایینی آن می‌شود از نوع معمولی‌اش تفکیک می‌شود. یک بافر معمولی دارای یک ورودی، یک خروجی و یک خط کنترل می‌باشد که وضع خروجی را مشخص می‌نماید. وقتی که ورودی کنترل برابر 1 است، خروجی فعال شده و گیت مانند یک بافر معمولی عمل می‌کند و در این حالت خروجی برابر ورودی اصلی است. وقتی که ورودی کنترل 0 شود، خروجی غیرفعال شده و گیت بدون توجه به مقدار ورودی اصلی به حالت امپدانس بالا می‌رود. حالت امپدانس بالای یک گیت سه حالته ویژگی خاصی را فراهم می‌کند که در دیگر گیت‌ها وجود ندارد. به علت این ویژگی، تعداد زیادی از خروجی‌های سه حالته می‌توانند به هم وصل و بدون تأثیر بر روی بار شدن، یک خط مشترکی را تشکیل دهند.

ساخت مولتی پلکسرها با بافرهای سه حالته در شکل ۳۰-۴ دیده می‌شود. بخش (الف) شکل، یک مولتی پلکسر 2 به 1 را با دو بافر سه حالته و یک وارونگر نشان می‌دهد. دو خروجی به هم وصل شده‌اند تا یک خروجی مشترک را به وجود آورند. باید اشاره کرد که این گونه اتصالات را با گیت‌هایی که خروجی سه حالته ندارند نمی‌توان اجرا کرد. وقتی ورودی انتخاب 0 است، بافر فوقانی به وسیله ورودی کنترلش فعال می‌گردد و در این حال بافر پایینی غیرفعال است.

ساختار یک مولتی پلکسر 4 به 1 در شکل ۳۰-۴ (ب) ملاحظه می‌شود. خروجی‌های چهار بافر سه



شکل ۳۰-۴. مولتی پلکسرها با گیت‌های سه حالته

حالته به هم متصل شده‌اند تا یک خروجی مشترک را بسازند. ورودی‌های کنترل به بافر مشخص می‌کنند که کدام یک از چهار ورودی نرمال I_0 تا I_3 به خط خروجی متصل خواهند شد. در هر لحظه از زمان تنها یکی از بافرها در حالت فعال قرار خواهد داشت. بافرهای متصل باید طوری وصل شوند که تنها یکی از بافرهای سه حالته با خروجی ارتباط داشته باشد، ضمن این که همه دیگر بافرها در حالت امپدانس بالا قرار خواهند گرفت. برای اطمینان از این که تنها یک ورودی کنترل در هر لحظه فعال است، از دیکدری طبق نمودار استفاده می‌کنیم. وقتی که ورودی فعال‌ساز دیکدر 0 است، هر چهار خروجی آن 0 خواهد بود و خط گذرگاه در حالت امپدانس بالاست زیرا هر چهار بافر غیرفعالند. وقتی که ورودی فعال‌ساز فعال گردد، یکی از بافرها، بسته به مقدار دودویی در ورودی‌های انتخاب دیکدر، فعال خواهد شد. با بررسی دقیق در می‌یابیم که این مدار راهی دیگر در ساخت یک مولتی پلکسر 4 به 1 است.

۴-۱۱ HDL برای مدارهای ترکیبی

زبان سخت‌افزاری verilog (HDL) در بخش ۹-۳ معرفی شد. در این بخش، روش دیگری را برای توصیف مدارهای ترکیبی HDL نشان خواهیم داد. مدارهای ترتیبی در فصل بعد توصیف خواهند شد. همانطور که قبلاً ذکر شد، مدول، یک بلوک ساختاری پایه در Verilog HDL است. مدول می‌تواند در هر یک از تکنیک‌های مدل‌سازی زیر توصیف گردد.

- مدل‌سازی سطح گیت با ذکر گیت‌های اصلی (Primitive) و مدول‌های تعریف شده بوسیله کاربر.
- مدل‌سازی روند داده با بکارگیری عبارات تخصیص مداوم (پیوسته) که با کلمه کلیدی assign انجام می‌شود.

- مدل‌سازی رفتاری با استفاده از عبارات تخصیص اجرایی (رویه‌ای) که با کلمه کلیدی always صورت می‌گیرد.

مدل‌سازی سطح گیت، مدار را با تعیین گیت‌ها و این که چگونه به هم وصل شده‌اند توصیف می‌نماید. مدل‌سازی روند داده اغلب برای توصیف مدارهای ترکیبی به کار می‌رود. مدل‌سازی رفتاری برای سیستم‌های دیجیتال در سطح بالاتر مورد استفاده است. مدل دیگری به جز روش‌های فوق وجود دارد که به آن مدل‌سازی سطح سوئیچ گویند. این نوع مدل‌سازی قابلیت طراحی را در سطح ترانزیستور MOS فراهم می‌سازد که در بخش ۱۰-۱۰ بررسی شده است.

مدل‌سازی سطح گیت

مدل‌سازی در سطح گیت در بخش ۹-۳ همراه با یک مثال ساده معرفی شد. در این نوع نمایش، یک مدار با گیت‌های منطقی و اتصالات بین آنها نشان داده می‌شود. این مدل توصیف متنی برای نمودار مداری را فراهم می‌کند. Verilog قادر است تا 12 گیت را به عنوان گیت‌های اصلی پیش تعریف شده تشخیص دهد. چهار گیت از آنها از نوع سه حالته است. هشت نوع دیگر آنهایی هستند که در بخش ۷-۲ لیست شدند. این گیت‌ها با کلمات کلید حروف کوچک زیر معرفی می‌شوند که عبارتند از: and، nand، or، nor، xor، xnor، not، buf. وقتی که گیت‌ها شبیه‌سازی شوند، سیستم به هر گیت یک مجموعه چهار مقداری را تخصیص می‌دهد. علاوه بر مقدار منطقی 0 و 1، دو مقدار نامشخص و امپدانس بالا هم لحاظ شده‌اند. مقدار نامشخص با x و امپدانس بالا با z مشخص شده است. مقدار نامشخص در حین شبیه‌سازی برای حالتی است که یک ورودی یا خروجی نامعلوم باشد و به عنوان مثال به آن مقدار 1 یا 0 تخصیص نیافته باشد. حالت امپدانس بالا در خروجی گیت‌های سه حالته هنگامی رخ می‌دهد که یک سیستم ناخودآگاه باز رها گردد. جدول درستی برای and، or، xor و not در جدول ۹-۴ دیده می‌شود. جدول درستی چهار گیت دیگر مشابه است با این تفاوت که خروجی‌ها متمم شده‌اند. توجه کنید که برای گیت and، خروجی فقط هنگامی 1 است که هر دو ورودی 1 باشند و هنگامی 0 است که هر یک از دو ورودی 0 باشد. در غیر این صورت اگر یک ورودی x یا z باشد خروجی x است. وقتی هر دو ورودی گیت or برابر 0 باشد

جدول ۹-۴. جدول درستی برای گیت‌های اصلی پیش‌تعریف شده

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

or	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

xor	0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

not	خروجی ورودی
	0 1
	1 0
	x x
	z x

خروجی آن 0است و اگر هر یک از ورودی‌ها 1 باشد خروجی هم 1است، در غیر اینصورت x است.
وقتی که یک گیت اصلی در یک مدول لحاظ شود گوییم در مدول ذکر شده است. به طور کلی ذکر قطعات عباراتی هستند که به قطعات سطح پایین‌تری در طراحی ارجاع می‌دهند، و کپی‌هایی اساسی (یا نمونه) از آن قطعات در مدول سطح بالاتر ایجاد می‌نمایند. بنابراین مدولی که یک گیت را در توصیف خود به کار می‌برد گیت را **ذکر** کرده است.

اکنون دو مثال از مدل‌سازی سطح گیت را نمایش می‌دهیم. هر دو مثال از عرض چند بیتی که بردار نامیده می‌شود، استفاده می‌کنند. یک بردار در داخل یک جفت کروشه مشخص می‌شود که از دو عدد و دو نقطه در بین آنها تشکیل شده است. کد زیر دو بردار را نشان می‌دهد:

```
output [0:3]D;
wire [7:0]SUM;
```

عبارت اول یک بردار خروجی D را با چهار بیت 0 تا 3 نشان می‌دهد. دومین عبارت یک بردار SUM سیمی (wire) با هشت بیت که از 0 تا 7 شماره‌گذاری شده‌اند را نشان می‌دهد. اولین عدد لیست شده باارزش‌ترین بیت بردار است. بیت‌های خاص در داخل کروشه ذکر می‌گردند، بنابراین D[2]، بیت 2 از D را نشان می‌دهد. همچنین ممکن است بخش‌هایی از یک بردار را ذکر کند. مثلاً SUM[2 : 0] سه بیت کم‌ارزش‌تر بردار SUM را بیان می‌کند.

مثال ۱-۴ HDL توصیف سطح گیت یک دیکدر 2 به 4 را نشان می‌دهد. این مدار دو ورودی داده A و B و یک ورودی فعال‌ساز E دارد. چهار خروجی با بردار D تعریف شده‌اند. wire برای اتصالات درونی به کار می‌رود. گیت‌های not متمم ورودی‌ها و چهار گیت nand خروجی D را تهیه می‌نماید. به یاد داشته باشید که همواره در لیست گیت، خروجی ابتدا ذکر می‌شود و سپس ورودی‌ها نوشته می‌شوند. این مثال دیکدر شکل ۱۹-۴ را توصیف می‌کند و رویه‌های بیان شده در بخش ۹-۳ را دنبال می‌نماید.

مثال ۱-۴، HDL

```
//Gate-level description of a 2-to-4-line decoder
//Figure 4-19
module decoder_gl (A,B,E,D);
   input A,B,E;
   output [0:3]D;
   wire Anot,Bnot,Enot;
   not
      n1 (Anot,A),
      n2 (Bnot,B),
      n3 (Enot,E);
   nand
      n4 (D[0],Anot,Bnot,Enot),
      n5 (D[1],Anot,B,Enot),
      n6 (D[2],A,Bnot,Enot),
      n7 (D[3],A,B,Enot);
endmodule
```

```
//Gate-level hierarchical description of 4-bit adder
// Description of half adder (see Fig 4-5b)
module halfadder (S,C,x,y);
   input x,y;
   output S,C;
//Instantiate primitive gates
   xor (S,x,y);
   and (C,x,y);
endmodule

//Description of full adder (see Fig 4-8)
module fulladder (S,C,x,y,z);
   input x,y,z;
   output S,C;
   wire S1,D1,D2; //Outputs of first XOR and two AND gates
//Instantiate the halfadder
    halfadder HA1 (S1,D1,x,y),
              HA2 (S,D2,S1,z);
    or g1(C,D2,D1);
endmodule

//Description of 4-bit adder (see Fig 4-9)
module _4bit_adder (S,C4,A,B,C0);
   input [3:0] A,B;
   input C0;
   output [3:0] S;
   output C4;
   wire C1,C2,C3;  //Intermediate carries
//Instantiate the fulladder
   fulladder  FA0 (S[0],C1,A[0],B[0],C0),
              FA1 (S[1],C2,A[1],B[1],C1),
              FA2 (S[2],C3,A[2],B[2],C2),
              FA3 (S[3],C4,A[3],B[3],C3);
endmodule
```

توجه کنید که کلمات کلیدی not و nand فقط یک بار نوشته می‌شوند و لزومی ندارد که برای هر گیت تکرار شوند، ولی ویرگول‌ها باید در انتهای هر سری گیت‌ها گذاشته شوند به جز در آخرین عبارت که با نقطه ویرگول پایان می‌یابد.

دو یا چند مدول را می‌توان با هم ترکیب نموده و توصیف سلسله مراتبی یک طرح را ساخت. دو روش طراحی اساسی وجود دارد: یکی بالا به پایین و دیگری پایین به بالا. در طراحی بالا به پایین، بلوک سطح بالا تعریف شده و سپس بلوک‌های جزیی لازم برای ساخت بلوک سطح بالا مشخص می‌شوند. در طراحی پایین به بالا، بلوک‌های ساختاری ابتدا مشخص شده و سپس برای ساخت بلوک بالایی ترکیب

می‌شوند. مثلاً جمع‌کننده شکل ۹-۴ را در نظر بگیرید. می‌توان آن را بلوکی تصور کرد که با چهار بلوک جمع‌کننده کامل ساخته شده است، در حالی که هر جمع‌کننده کامل از دو بلوک نیم‌جمع‌کننده تشکیل می‌شود. در طراحی بالا به پایین، ابتدا جمع کننده 4 بیت تعریف می‌شود و سپس بلوک‌های دو نیم‌جمع کننده توصیف می‌شوند. در طراحی پایین به بالا ، نیم جمع کننده‌ها اول تعریف می‌شوند، سپس جمع کننده 4 بیت از آن ساخته می‌شود.

توصیف سلسه مراتبی (پایین - بالا) جمع‌کننده 4 بیت در مثال ۲-۴ HDL نشان داده شده است. نیم‌جمع‌کننده با ذکر گیت‌های اصلی تعریف می‌شود. مدول بعدی با ذکر دو نیم جمع کننده، جمع کننده کامل را ذکر کرده است. مدول سوم یک جمع‌کننده 4 بیت را با ذکر چهار جمع‌کننده کامل توصیف می‌نماید. (دقت کنید که شناسه‌ها نمی‌توانند با یک عدد آغاز شوند ولی می‌توانند با یک زیر خط تیره در پایین شروع شوند، بنابراین نام مدول 4bitadder_ خواهد بود.) ذکر هم با نام مدولی که یک مجموعه نام پورت جدید یا قبلی را بکار می‌برد انجام می‌شود. مثلاً، HA1 در داخل مدول جمع‌کننده با x، y و D_1 ، S_1 ذکر، می‌شود. این ذکر یک نیم‌جمع‌کننده با خروجی‌های D_1، S_1 و ورودی‌های y و x را تولید می‌نماید.

توجه کنید که در Verilog ، تعریف یک مدول نمی‌تواند در توصیف دیگری قرار گیرد. به بیان دیگر، یک مدول نمی‌تواند در بین یک کلمه کلیدی module و endmodule واقع شود. تنها راه تعریف یک مدول در دیگری، ذکر آن است. بنابراین مدول‌ها در مدول‌های دیگر برای توصیف سلسله مراتبی طرح ذکر می‌شوند. همچنین توجه کنید که نام‌ها باید هنگام تعریف مدول‌ها ذکر شوند. (مثل FA0 برای اولین جمع‌کننده کامل در سومین جدول)، ولی ذکر نام در گیت‌های اصلی اختیاری است.

گیت‌های سه حالته

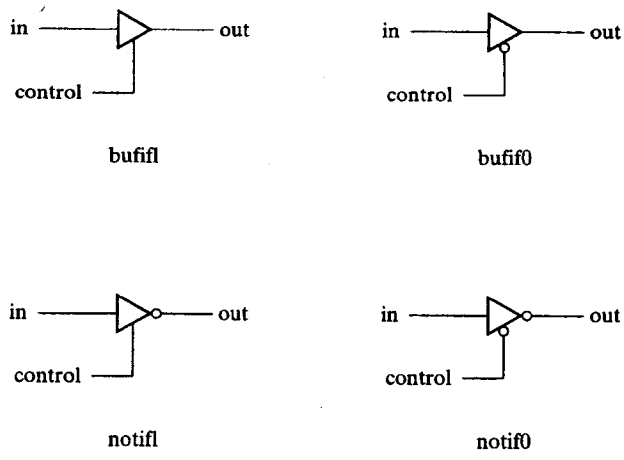
همانطور که در بخش ۱۰-۴ ذکر شد، هر گیت سه حالته دارای یک ورودی کنترل است که می‌تواند آن را به حالت امپدانس بالا ببرد. این حالت در HDL با سمبل z مشخص می‌شود. چهار نوع گیت سه حالته طبق شکل ۳۱-۴ وجود دارد. گیت bufif1 مانند یک بافر معمولی عمل می‌کند به شرطی که control = 1 باشد. وقتی control = 0 است، خروجی به حالت امپدانس بالای z می‌رود. گیت bufif0 رفتاری مشابه دارد با این تفاوت که امپدانس بالا در control = 1 رخ می‌دهد. دو گیت not به همین ترتیب کار می‌کنند با این تفاوت که وقتی در امپدانس بالا نیستند، خروجی آنها متمم ورودی است. گیت‌ها با عبارت زیر ذکر می‌شوند:

gate name (output, input, control);

gate name می‌تواند یکی از چهار گیت سه حالته انتخاب گردد. خروجی می‌تواند 0، 1 یا z باشد دو مثال از ذکر گیت در زیر آمده است:

```
bufif1  (OUT,A,control);
notif0  (Y,B,enable);
```

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



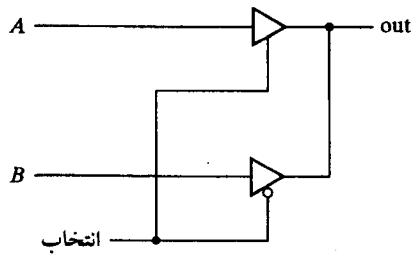
شکل ۳۱-۴ . گیت‌های سه حالته

باشد خروجی out به z می‌رود. در مثال دوم وقتی enable = 1 است Y = z خواهد شد و خروجی Y = B′ در enable = 0 رخ می‌دهد.

خروجی گیت‌های سه حالته می‌توانند برای تشکیل یک خروجی مشترک بهم وصل شوند. برای شناخت چنین اتصالی، HDL از کلمه کلید tri (برای tristate) استفاده می‌کند تا بیان نماید که خروجی دارای قابلیت راه‌اندازی چندگانه است. مثلاً مولتی پلکسر 2 به 1 را در شکل ۳۲-۴ با گیت‌های سه حالته در نظر بگیرید.

توصیف HDL باید برای خروجی از داده نوع tri استفاده کند.

```
module muxtri (A,B,select,OUT);
   input A,B,select;
   output OUT;
   tri OUT;
   bufif1 (OUT,A,select);
   bufif0 (OUT,B,select);
endmodule
```



شکل ۳۲-۴ . مولتی پلکسر 2 به 1 با بافرهای سه حالته

دو بافر سه حالته دارای خروجی مشترک‌اند. برای این که نشان دهیم آنها اتصال مشترک دارند، باید خروجی OUT را با کلمه کلیدی tri همراه کنیم.

کلمات کلیدی wire و tri مثال‌هایی از داده نوع net می‌باشند. Net ها اتصال بین دوالمان یا عنصر را نشان می‌دهند. خروجی آنها با خروجی وسیله‌ای که نشان می‌دهند مرتباً راه‌اندازی می‌شود. کلمه net (شبکه) یک کلمه کلیدی نیست و کلاسی از انواع داده مانند wire ، wor ، wand ، tri ، supply1 و supply0 را نشان می‌دهد. اعلان wire مکرّر به کار برده می‌شود. مدل‌های wor پیاده‌سازی سخت‌افزاری آرایش wired–OR را نشان می‌دهند. مدل‌های wand هم برای آرایش wired–AND در نظر گرفته شده‌اند (شکل ۲۸-۳). شبکه‌های supply1 و supply0 منبع تغذیه و زمین هستند. از آنها در توصیف سطح- سوئیچ استفاده می‌شود. (بخش ۱۰-۱۰ ملاحظه گردد).

مدل‌سازی روند داده

مدل‌سازی روند داده از تعدادی عملگر روی عملوندها استفاده می‌کند تا خروجی موردنظر را تولید کند. Verilog HDL حدود 30 عملگر در اختیار می‌گذارد. جدول ۱۰-۴ بخشی از این عملگرها، سمبل‌های آنها و عملی که اجرا می‌کنند را نشان می‌دهند. لیست کامل عملگرها را در جدول ۱-۸ در بخش ۲-۸ ملاحظه کنید. لازم است تا بین اعمال حسابی و منطقی تفکیک به عمل آید، بنابراین برای هر یک، از سمبل جداگانه‌ای استفاده شده است. سمبل (+) برای جمع حسابی به کار رفته و AND منطقی از سمبل & استفاده می‌کند. برای OR ، NOT و XOR سمبل‌های خاصی وجود دارد. سمبل برابری از دو علامت مساوی و بدون فضا در بین آنها استفاده می‌کند تا با علامت برابری به کار رفته با عبارت تخصیص اشتباه نشود. عملگر ادغام مکانیزمی است که برای ضمیمه کردن چند عملوند استفاده می‌شود. مثلاً دو عملوند دو بیتی را می‌توان برای ایجاد یک عملوند چهار بیتی در هم ادغام کرد. عملگر شرطی بعد همراه با مثال ۶-۴ HDL توضیح داده شده است.

جدول ۱۰-۴. عملگرهای Verilog HDL

Symbol	Operation
+	binary addition
−	binary subtraction
&	bit-wise AND
|	bit-wise OR
∧	bit-wise XOR
~	bit-wise NOT
==	equality
>	greater than
<	less than
{ }	concatenation
? :	conditional

مثال ۳-۴ HDL

```
//Dataflow description of a 2-to-4-line decoder
//See Fig. 4-19
module decoder_df (A,B,E,D);
   input A,B,E;
   output [0:3] D;
   assign D[0] = ~(~A & ~B & ~E),
          D[1] = ~(~A & B & ~E),
          D[2] = ~(A & ~B & ~E),
          D[3] = ~(A & B & ~E);
endmodule
```

مدل‌سازی روند داده از تخصیص‌های مداوم و کلمه کلیدی assign استفاده می‌کند. یک تخصیص مداوم عبارتی است که یک مقدار را به یک net تخصیص می‌دهد. *net* نوع داده در HDL برای نمایش اتصالات بین عناصر مدار به کار می‌رود. یک net خروجی یک گیت را که با عبارت output یا wire بیان شده، تعریف می‌کند. مقدار تخصیص داده شده به net با یک عبارت که عملگر و عملوند را استفاده می‌کند بیان می‌گردد. به عنوان مثال با فرض اعلان متغیرها، یک مولتی پلکسر 2 به 1 با ورودی‌های داده A و B، ورودی انتخاب S و خروجی Y با عبارت مداوم زیر تعریف می‌گردد:

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

عبارت با کلمه کلیدی assign و به دنبال آن خروجی موردنظر Y و یک علامت مساوی شروع می‌شود. به دنبال علامت مساوی، یک عبارت بول آورده شده است. در سخت‌افزار، این عبارت معادل با اتصال خروجی گیت OR (|) به سیم Y است.

مثال‌های بعدی مدل‌های روند داده دو مثال سطح گیت قبلی را نشان می‌دهند. توصیف روند داده یک دیکدر 2 به 4 در مثال ۳-۴ HDL نشان داده شده است. مدار با توجه به عبارات بولی با چهار عبارت تخصیص مداوم تعریف شده است که هر یک متعلق به یک خروجی است. توصیف روند داده یک جمع کننده 4 بیت در مثال ۴-۴ HDL آورده شده است. منطق جمع با یک عبارت و استفاده از عملگرهای جمع و ادغام بیان شده است. سمبل جمع (+) بیانگر جمع دودویی 4 بیت A با چهار بیت B و یک رقم نقلی Cin است. خروجی موردنظر ادغام نقلی خروجی Cout و چهار بیت SUM است.

مثال ۴-۴ HDL

```
//Dataflow description of 4-bit adder
module binary_adder (A,B,Cin,SUM,Cout);
   input [3:0] A,B;
   input Cin;
   output [3:0] SUM;
   output Cout;
   assign {Cout,SUM} = A + B + Cin;
endmodule
```

مثال ۵-۵، HDL

```
//Dataflow description of a 4-bit comparator.
module magcomp (A,B,ALSB,AGTB,AEQB);
   input [3:0] A,B;
   output ALTB,AGTB,AEQB;
   assign ALTB=(A < B),
          AGTB = (A > B),
          AEQB = (A == B);
endmodule
```

ادغام علموندها در پرانتز بیان شده و یک ویرگول آنها را از هم جدامی‌سازد. بنابراین پنج بیت {Cout, SUM} ، نتیجه عمل جمع را به نمایش می‌گذارد.

مدل‌سازی روند داده امکان توصیف مدارهای ترکیبی را با تابع به جای ساختار گیتی‌اش فراهم می‌سازد. برای ملاحظه چگونگی انجام طراحی دیجیتال با روند داده، مقایسه‌گر مقدار چهار بیتی توصیف شده در مثال ۵-۴ HDL را در نظر بگیرید. مدل دو گروه ورودی چهار بیت A و B و سه خروجی را مشخص می‌کند. اگر A کوچکتر از B باشد خروجی (ALTB) در منطق 1 و اگر A بزرگتر از B باشد خروجی (AGTB) در منطق 1 قرار می‌گیرد. ضمن این که اگر A برابر با B است خروجی (AEQB) وجود داشته باشد. توجه کنید که تساوی با دو علامت مساوی تعریف می‌شود. کامپایلر طراحی Verilog HDL می‌تواند این توصیف مدول را به عنوان ورودی بپذیرد و یک خروجی netlist از یک مدار معادل با شکل ۱۷-۴ را فراهم سازد.

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

Condition ? true– expression: false–expression;

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

مثال ۶-۴ HDL مولتی پلکسر 2 به 1 خط را با عملگر شرطی توصیف می‌نماید. تخصیص مداوم

```
assign OUT = select ? A : B ;
```

شرط زیر را بیان می‌کند.

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

مثال ۶-۴، HDL

```
//Dataflow description of 2-to-1-line multiplexer
module mux2x1_df (A,B,select,OUT);
   input A,B,select;
   output OUT;
   assign OUT = select ? A : B;
endmodule
```

مثال ۴-۷، HDL

```
//Behavioral description of 2-to-1-line multiplexer
module mux2x1_bh(A,B,select,OUT);
   input A,B,select;
   output OUT;
   reg OUT;
   always @ (select or A or B)
         if (select == 1) OUT = A;
         else OUT = B;
endmodule
```

مدل‌سازی رفتاری

مدل‌سازی رفتاری مدارهای دیجیتال را در سطح الگوریتمی و عملیاتی نمایش می‌دهد. این مدل اغلب در توصیف مدارهای ترتیبی به کار برده می‌شود، ولی قابل استفاده در توصیف مدارهای ترکیبی هم می‌باشد. در اینجا دو مثال از مدارهای ترکیبی ساده برای معرفی موضوع ارائه می‌گردد. این بحث پس از مطالعه مدارهای ترتیبی در بخش ۵-۵ مورد بحث بیشتری قرار گرفته است.

توصیف‌های رفتاری از کلمه کلیدی always و به دنبال آن لیستی از عبارات تخصیص اجرایی (رویه‌ای) استفاده می‌کنند. خروجی موردنظر این عبارات باید نوع داده reg باشد. برخلاف داده نوع wire که خروجی موردنظر یک تخصیص ممکن است مرتباً به روز شود، داده نوع reg مقدارش را تا تخصیص مقدار جدید حفظ می‌کند.

مثال ۴-۷ HDL توصیف رفتاری مولتی پلکسر 2 به 1 را نشان می‌دهد (آن را با مثال ۴-۶ HDL مقایسه کنید). چون OUT یک خروجی موردنظر یا مقصد است، باید علاوه بر اعلان output ، به صورت داده reg هم اعلام گردد. عبارات تخصیص اجرایی در داخل بلوک always هر زمان که تغییری در هر متغیر بعد از علامت @ رخ دهد دوباره اجرا می‌گردد. توجه کنید که در انتهای عبارت always علامت (;) وجود ندارد. در این حال، آنها عبارتند از A ، B و select . توجه کنید که کلمه کلیدی or در بین متغیرها به جای عملگر منطقی OR ، "|" ، استفاده شده است. عبارت شرطی if– else تصمیمی را که مبتنی بر مقدار ورودی selectاست فراهم می‌کند. عبارت if را می‌توان بدون ذکر سمبل کمیت نیز نوشت:

```
if (select) OUT = A ;
```

و به این معنی است که select برای منطق 1 چک می‌شود.

مثال ۴-۸ HDL یک مولتی پلکسر 4 به 1 را توصیف می‌نماید. ورودی select به صورت یک بردار 2 بیت توصیف شده و خروجی y هم با داده reg اعلان شده است. عبارت always دارای یک بلوک ترتیبی در بین کلمات کلیدی case و endcase است. این بلوک هر وقت که هر ورودی بعد از علامت @ تغییر کند، اجرا خواهد شد. عبارت case یک انشعاب شرطی چند مسیری است. عبارت case (select) با مقادیر لیست عباراتی که به دنبالش می‌آیند ارزیابی و مقایسه می‌شود. اولین مقداری که با شرط

مثال ۸-۴، HDL

```
//Behavioral description of 4-to-1- line multiplexer
//Describes the function table of  Fig. 4-25(b).
module mux4x1_bh (i0,i1,i2,i3,select,y);
   input i0,i1,i2,i3;
   input [1:0] select;
   output y;
   reg y;
   always @ (i0 or i1 or i2 or i3 or select)
            case (select)
               2'b00: y = i0;
               2'b01: y = i1;
               2'b10: y = i2;
               2'b11: y = i3;
            endcase
endmodule
```

صحیح تطبیق کند اجرا می‌گردد. چون select یک عدد دوبیتی است، می‌تواند 00، 01، 10 و یا 11 باشد. اعداد دودویی با حرف b و قبل از آن یک علامت پریم مشخص می‌گردند. سایز عدد ابتدا نوشته می‌گردد و سپس مقدار آن ذکر می‌شود. بنابراین 2' b 0 1 به معنی عدد دودویی دو رقمی است که مقدارش 01 است. اعداد را می‌توان به دهدهی، هشت هشتی یا شانزده شانزدهی و به ترتیب با 'd، 'o و 'h مشخص کرد. اگر مبنای عدد مشخص نباشد، پیش‌فرض دهدهی خواهد بود. اگر سایز عدد نامشخص باشد، سیستم سایز را 32 بیت فرض خواهد کرد.

در اینجا مثال‌های ساده‌ای را از توصیف‌های رفتاری مدارهای ترکیبی نشان دادیم. مدل‌سازی رفتاری و عبارات تخصیص اجرایی دانش مدارهای ترتیبی را لازم دارد و در بخش ۵-۵ به طور مشروح ارائه شده است.

نوشتن یک برنامه تست ساده

یک برنامه تست (T.B) برنامه‌ای HDL است که برای اعمال محرک به طرح HDL برای تست و مشاهده پاسخ شبیه‌ساز آن به کار می‌رود. T.B می‌تواند بسیار پیچیده و طولانی باشند تا حدی که ساخت آن از طرح مورد تست بیشتر طول بکشد. با این وجود، برنامه‌ای که در اینجا بررسی می‌شود نسبتاً ساده است زیرا ما فقط مایل به تست مدارهای ترکیبی هستیم. مثال‌ها برای نمایش توصیف‌های نمونه مدول‌های محرک HDL ارائه شده‌اند.

علاوه بر عبارت always، برنامه تست از عبارت initial برای تهیه محرک به مدار تحت تست استفاده می‌کند. عبارت always حلقه‌ای را به صورت تکراری اجرا می‌کند. عبارت initial فقط یک بار با شروع از t = 0 شبیه‌سازی را انجام داده و ممکن است هر عملی را با تأخیری که مضربی از واحدهای زمانی است و با سمبل # مشخص شده ادامه دهد. مثلاً بلوک initial زیر را ملاحظه نمایید.

```
initial
    begin
            A = 0; B= 0;
       #10  A = 1;
       #20  A = 0; B=1;
    end
```

بلوک، بین begin و end محصور شده است. در $t = 0$، A و B در 0 قرار گرفته‌اند. 10 واحد زمان بعد، A به 1 تغییر یافته است، پس از 20 واحد زمانی، (در $t = 30$) A به 0 و B به 1 تغییر پیدا می‌کند. ورودی‌ها به جدول درستی 3 بیتی می‌توانند با بلوک initial تولید شوند:

```
initial
    begin
            D = 3'b000;
            repeat (7)
       #10  D = D + 3'b001;
    end
```

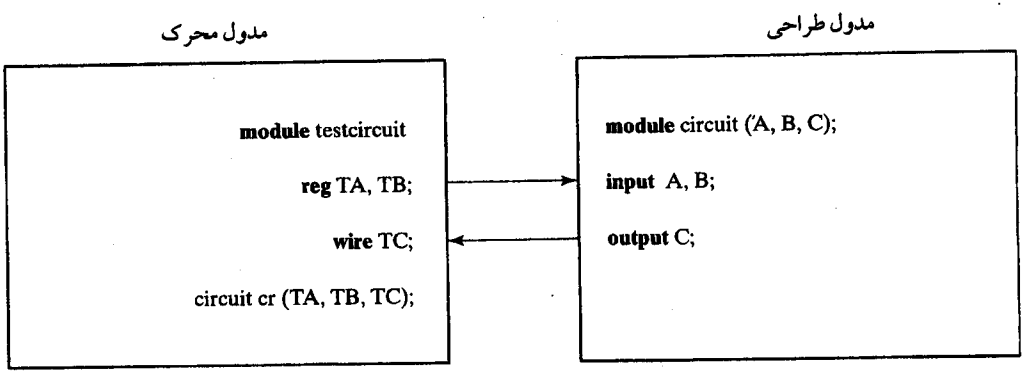
بردار 3 بیت D در $t = 0$ با 000 مقداردهی اولیه می‌شود. کلمه کلیدی repeat یک عبارت حلقه‌ای را تداعی می‌کند: یعنی عدد 1 هفت بار به D اضافه شده و این کار هر 10 واحد زمانی یک بار تکرار شده است. نتیجه یک رشته اعدادی از 000 تا 111 خواهد بود.

یک مدول محرک در یک برنامه HDL ساختار زیر را دارد.

نام **module** تست
اعلان شناسه‌های **reg** و **wire** محلی
ذکر مدول طراحی مورد تست
ایجاد عبارات محرک **initial** و **always**
نمایش پاسخ خروجی
endmodule

یک مدول تست معمولاً ورودی یا خروجی ندارد. سیگنال‌هایی که به عنوان ورودی به مدول طراحی برای شبیه‌سازی اعمال می‌شوند معمولاً در مدول محرک به عنوان نوع داده reg محلی اعلان می‌گردند. خروجی‌های مدول طراحی که برای تست نمایش داده می‌شوند در مدول محرک به عنوان داده نوع wire اعلان می‌شوند. آنگاه مدول تحت تست با به‌کارگیری شناسه‌های محلی ذکر می‌گردد. شکل ۳۳-۴ این ارتباط را نشان می‌دهد. مدول محرک ورودی‌ها را برای مدول طراحی با اعلان شناسه‌های TA و TB به عنوان نوع reg تولید می‌کند و خروجی طرح را با شناسه نوع wire، یعنی TC چک می‌نماید. سپس شناسه‌های محلی برای ذکر مدول زیر تست به کار می‌روند.

پاسخ به محرک تولید شده با بلوک‌های intial و always در خروجی محرک به صورت نمودار زمان‌بندی ظاهر می‌شود. و نیز می‌توان با استفاده از Verilog system tasks خروجی عددی نیز تولید



شکل ۳۳ ۴ مدول‌های محرک و طراحی محاوره‌ای

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:

یک بار نمایش مقدار متغیر یا رشته‌هایی با بازگشت از انتهای خط	$ display
مثل $ display ولی بدون رفتن به خط بعد	$ write
هر وقت در حین اجرای شبیه‌سازی متغیری تغییر کند، آن را نمایش می‌دهد	$ monitor
زمان شبیه‌سازی را نشان می‌دهد	$ time
شبیه‌سازی را پایان می‌دهد	$ finish

قاعده نوشتن $ display ، $write و $ monitor به شکل زیر است.

Task– name (format specification, argument list);

و یا

نام تکلیف (لیست آرگومان و مشخصات قالب);

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 سبمل‌های %b ، %d ، %o و %h نشان داده می‌شوند. مثلاً عبارت:

```
$display (%d %b %b, C,A,B);
```

به این معنی که C به دهدهی و A و B به دودویی نمایش داده شوند. توجه کنید که در مشخصات قالب، علامت ویرگول وجود ندارد ولی برای جداسازی مشخصات قالب و لیست آرگومان و نیز بین متغیرهای لیست آرگومان، ویرگول وجود دارد. مثالی که یک رشته را داخل علامت کوتیشن یا نقل قول محصور کند مشابه زیر است:

```
$display ("time = %0d A = %b B = %b", $time,A,B);
```

و نمایش زیر را تولید خواهد کرد

```
time = 3  A = 10  B = 1
```

```
//Stimulus for mux2x1_df.
module testmux;
  reg TA,TB,TS;  //inputs for mux
  wire Y;        //output from mux
  mux2x1_df mx (TA,TB,TS,Y);  // instantiate mux
     initial
        begin
              TS = 1; TA = 0; TB = 1;
          #10 TA = 1; TB = 0;
          #10 TS = 0;
          #10 TA = 0; TB = 1;
        end
     initial
      $monitor("select = %b A = %b B = %b OUT = %b time = %0d",
               TS, TA, TB, Y, $time);
endmodule

//Dataflow description of 2-to-1-line multiplexer
//from Example 4-6
module mux2x1_df (A,B,select,OUT);
   input A,B,select;
   output OUT;
   assign OUT = select ? A : B;
endmodule

Simulation log:

select = 1 A = 0 B = 1 OUT = 0 time = 0
select = 1 A = 1 B = 0 OUT = 1 time = 10
select = 0 A = 1 B = 0 OUT = 0 time = 20
select = 0 A = 0 B = 1 OUT = 1 time = 30
```

که (time =)، (A =) و (B =) بخشی از رشته مورد نمایش‌اند. قالب %0d، %b و %b به ترتیب مبنای $ time، A و B را مشخص می‌کنند. هنگام نمایش مقادیر تابع، بهتر است قالب %0d را به جای %d به کار ببریم. این کار رقم‌های باارزش‌تر را بدون فضای خالی تولید می‌نماید (%d حدود 10 فضای خالی را تولید می‌کند زیرا زمان با عدد 32 بیت تولید می‌گردد).

مثالی از مدول محرک در مثال HDL ۴-۹ نشان داده شده است. مدار مورد تست یک مولتی پلکسر 1 × 2 است که در مثال ۶-۴ توصیف گردید مدول testmux پورت ندارد. ورودی‌های mux با کلمه کلیدی reg و خروجی‌ها با wire مشخص می‌شوند. mux با متغیرهای محلی ذکر شده است. بلوک initial رشته‌ای از اعداد دودویی را که در حین شبیه‌سازی اعمال می‌گردند، مشخص می‌کند. پاسخ خروجی با تکلیف monitor $ چک می‌شود. هر بار یک متغیر تغییر مقدار دهد، شبیه‌ساز ورودی‌ها، خروجی و زمان را نمایش می‌دهد. نتیجه شبیه‌سازی در مثال زیر تیتر simulation ذکر شده است.

```
//Gate-level description of circuit of Fig. 4-2
module analysis (A,B,C,F1,F2);
   input   A,B,C;
   output  F1,F2;
   wire    T1,T2,T3,F2not,E1,E2,E3;
   or  g1 (T1,A,B,C);
   and g2 (T2,A,B,C);
   and g3 (E1,A,B);
   and g4 (E2,A,C);
   and g5 (E3,B,C);
   or  g6 (F2,E1,E2,E3);
   not g7 (F2not,F2);
   and g8 (T3,T1,F2not);
   or  g9 (F1,T2,T3);
endmodule

//Stimulus to analyze the circuit
module test_circuit;
  reg [2:0]D;
  wire F1,F2;
  analysis fig42(D[2],D[1],D[0],F1,F2);
  initial
    begin
     D = 3'b000;
     repeat(7)
      #10 D = D + 1'b1;
    end
  initial
     $monitor ("ABC = %b F1 = %b F2 =%b ",D, F1, F2);
endmodule


Simulation log:

ABC = 000 F1 = 0 F2 =0
ABC = 001 F1 = 1 F2 =0
ABC = 010 F1 = 1 F2 =0
ABC = 011 F1 = 0 F2 =1
ABC = 100 F1 = 1 F2 =0
ABC = 101 F1 = 0 F2 =1
ABC = 110 F1 = 0 F2 =1
ABC = 111 F1 = 1 F2 =1
```

می‌بینیم که وقتی $S = 1$ باشد $OUT = A$ و وقتی $S = 0$ باشد $OUT = B$ است، که بدین ترتیب عملکرد مولتی پلکسر را تأیید می‌کند.

شبیه‌سازی منطقی، روشی سریع و دقیق در تحلیل مدارهای ترکیبی جهت اطمینان از عملکرد

صحیح آنهاست. دو نوع تصدیق وجود دارد: عملیاتی و زمانی. در تصدیق عملیاتی، ما عملکرد مدار را جدا از ملاحظات زمانی مورد بررسی قرار می‌دهیم. این کار با تهیه جدول درستی مدار ترکیبی انجام می‌شود. در تصدیق زمانی عملکرد مدار را با احتساب آثار تأخیر در گیت‌ها مطالعه می‌کنیم. این کار با مشاهده امواج در خروجی گیت‌ها وقتی به یک ورودی مفروض پاسخ می‌دهند، صورت می‌گیرد.
مثالی از یک مدار با تأخیر در گیت‌ها در بخش ۹-۳ و با مثال ۳-۳ HDL ملاحظه شد. اکنون یک مثال HDL، که جدول درستی یک مدار ترکیبی را تولید می‌کند، ملاحظه می‌شود.

تحلیل مدارهای ترکیبی در بخش ۲-۴ ملاحظه شد. یک مدار جمع‌کننده کامل چند طبقه تحلیل شد و جدول درستی آن بدست آمد. توصیف سطح گیت این مدار در مثال ۱۰-۴ HDL آورده شده است. مدار دارای سه ورودی دو خروجی و نه گیت است. به دنبال توصیف مدار، اتصالات درونی بین گیت‌ها طبق نمودار شماتیک ۲-۴ آمده است. محرک مدار در مدول دوم ذکر شده است. ورودی‌ها برای شبیه‌سازی مدار با بردار D سه بیتی reg مشخص شده‌اند. D[2] با ورودی A و D[1] با ورودی B، و D[0] با ورودی C معادل است. خروجی‌های مدار F_1 و F_2 بوده و از نوع wire اعلان می‌شوند. این رویه مراحل مشخص شده در شکل ۳۳-۴ را دنبال می‌کند. حلقه repeat هفت عدد دودویی را پس از 000 برای جدول درستی تهیه می‌کند. نتایج شبیه‌سازی جدول درستی خروجی را نمایش می‌دهد. جدول درستی لیست شده نشان می‌دهد که مدار یک جمع‌کننده کامل است.

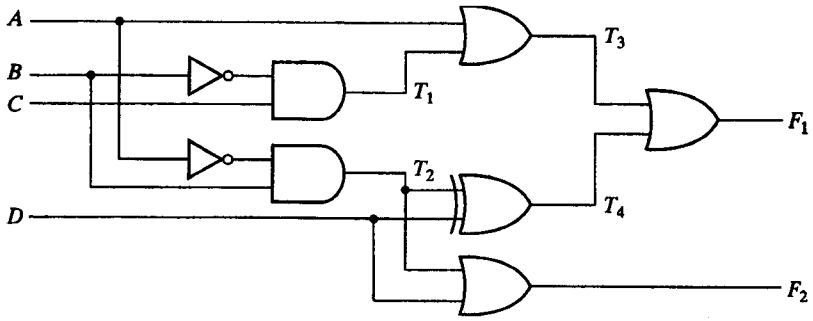
مسائل

۱-۴ مدار ترکیبی شکل (م ۱-۴) را در نظر بگیرید.

(الف) عبارات بولی را برای T_1 تا T_4 بدست آورید. خروجی‌های F_1 و F_2 را به عنوان یک تابع چهار ورودی ارزیابی کنید.

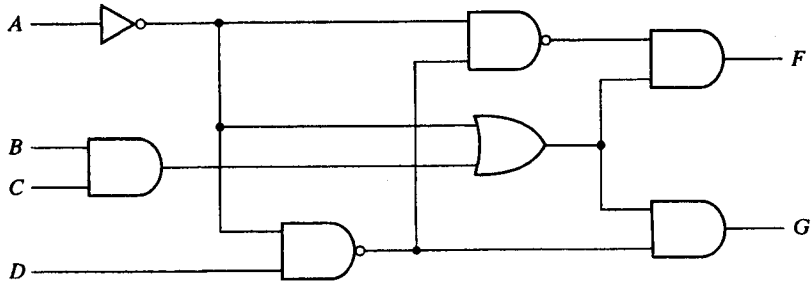
(ب) جدول درستی را با 16 ترکیب ورودی تشکیل دهید. آنگاه مقادیر دودویی T_1 تا T_4 و خروجی‌های F_1 و F_2 را در جدول لیست نمایید.

(پ) توابع بولی خروجی حاصل در بخش (ب) را روی نقشه‌ها رسم کنید و نشان دهید که عبارات بولی ساده شده معادل با آنهایی هستند که در بخش (الف) بدست آمد.



شکل (م ۱-۴)

۴-۲ عبارات بولی ساده شده را برای خروجی F و G برحسب متغیرهای ورودی در مدار شکل (م ۲-۴) بدست آورید.



شکل (م ۲-۴)

۴-۳ برای مدار شکل ۲۶-۴ (بخش ۱۰-۴)

(الف) توابع بولی را برای چهار خروجی برحسب متغیرهای ورودی بنویسید.

(ب) اگر مدار در جدول درستی لیست شده باشد، چند سطر و ستون در جدول وجود دارد؟

۴-۴ یک مدار ترکیبی با سه ورودی و یک خروجی طراحی کنید.

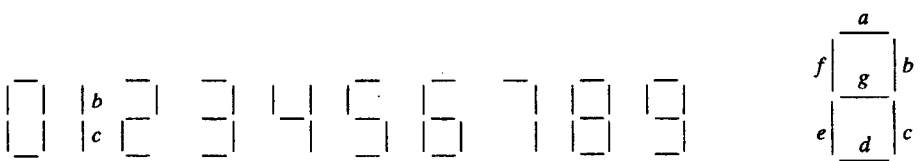
۴-۵ یک مدار ترکیبی با سه ورودی x و y و z و سه خروجی A و B و C طراحی کنید. وقتی ورودی دودویی 0 ، 1 ، 2 یا 3 است، خروجی دودویی یکی بیشتر از ورودی است. وقتی ورودی دودیی 4 ، 5 ، 6 و یا 7 است خروجی دودویی یکی کمتر از ورودی است.

۴-۶ یک مدار اکثریت مداری ترکیبی است که اگر متغیرها بیش از 0 ، مقدار 1 را داشته باشند خروجی‌اش 1 می‌گردد. در غیر این صورت خروجی 0 است. یک مدار اکثریت 3 بیت طراحی کنید.

۴-۷ یک مدار ترکیبی که چهار بیت از کد گری (جدول ۶-۱) را به 4 بیت عدد دودویی تبدیل نماید، طراحی کنید. مدار با XOR طراحی گردد.

۴-۸ یک مبدل کد که یک رقم دهدهی از کد 1- ، 2- ، 4 و 8 را به BCD (جدول ۵-۱) تبدیل کند، طراحی نمایید.

۴-۹ یک دیکدر BCD به هفت قسمتی مداری ترکیبی است که یک رقم دهدهی در BCD را به کد مناسبی جهت انتخاب قطعات در یک نمایشگر تبدیل می‌کند تا ارقام دهدهی به صورت مفهوم قابل مشاهده باشند. هفت خروجی دیکدر (g, f, e, d, c, b, a) قطعات مربوطه را طبق شکل (م ۹-۴ الف) انتخاب می‌کنند. عدد مورد نمایش در شکل (م ۹-۴ ب) دیده می‌شود. یک دیکدر BCD به هفت قسمتی با حداقل گیت طراحی کنید. شش حالت نامعتبر باید به صورت خاموش ظاهر شوند.



(الف) علامت سگمنت (ب) علامت عدد برای نمایش گر

شکل (م ۹-۴)

۴-۱۰ یک مدار ترکیبی مولد متمم 2 چهار بیت طراحی کنید. (خروجی، متمم 2 عدد ورودی را تولید می‌کند) نشان دهید که مدار می‌تواند با گیت‌های XOR ساخته شود. آیا می‌توانید توابع خروجی یک مولد متمم 2 پنج بیت را حدس بزنید.

۴-۱۱ یک مدار ترکیبی افزایش‌گر 4 بیت طراحی نمایید (مداری که 1 را به عدد دودویی 4 بیت اضافه می‌کند.) این مدار را می‌توان با چهار نیم جمع‌کننده طراحی کرد.

۴-۱۲ (الف) یک مدار نیم تفریق‌گر با ورودی‌های x و y و خروجی‌های D و B طراحی کنید. مدار تفریق x−y را انجام می‌دهد سپس تفاضل را در D و رقم قرضی را در B قرار می‌دهد.
(ب) یک مدار تفریق‌گر کامل با سه ورودی x، y و z و دو خروجی D و B طراحی نمایید. مدار تفریق x−y−z را اجرا می‌کند، که z قرضی ورودی، B قرضی خروجی و D تفاضل باقیمانده است.

۴-۱۳ مدار جمع- تفریق‌گر شکل ۴-۱۳ دارای مقادیر زیر برای ورودی مد M و ورودی‌های داده A و B است. در هر حال، مقادیر چهار خروجی SUM، رقم نقلی C و سرریز V را معین کنید.

	M	A	B
(a)	0	0111	0110
(b)	0	1000	1001
(c)	1	1100	1000
(d)	1	0101	1010
(e)	1	0000	0001

۴-۱۴ فرض کنید که گیت XOR دارای یک تأخیر انتشار 20ns و گیت‌های AND و OR دارای تأخیر انتشار 10ns هستند. زمان کل تأخیر انتشار در جمع‌کننده 4 بیت شکل ۴-۱۲ چقدر است؟

۴-۱۵ عبارت بولی دو طبقه را برای نقلی خروجی C_4 در مولد نقلی پیش‌بینی شونده شکل ۴-۱۲ بدست آورید.

۴-۱۶ نشان دهید که نقلی خروجی در یک مدار جمع‌کننده کامل را می‌توان به فرم AND−OR−INVERT

$$C_{i+1} = G_i + P_iC_i = (G_i'P_i + G_i'C_i')'$$

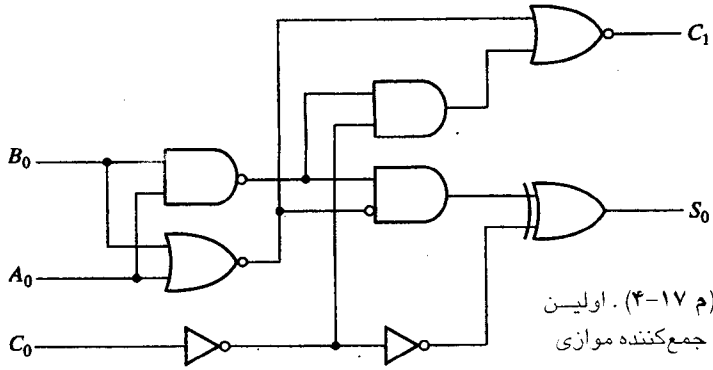
نشان داد. آی‌سی 74182 یک مدار مولد نقلی پیش‌بینی شونده است که نقلی‌ها را با گیت‌های AND−OR−INVERT تولید می‌کند (بخش ۳-۷ ملاحظه شود). فرض بر این است که پایانه‌های ورودی متمم‌های G و P و C_1 را دارند. توابع بول را برای نقلی‌های پیش‌بینی شونده C_2، C_3 و C_4 در این آی‌سی، بدست آورید. (توجه: از روش جایگزینی در بدست آوردن نقلی‌ها برحسب C'_1 استفاده کنید).

۴-۱۷ تولید و انتشار نقلی را به صورت زیر تعریف کنید:

$$P_i = A_i + B_i$$
$$G_i = A_iB_i$$

نشان دهید که نقلی خروجی و جمع خروجی یک جمع‌کننده کامل به صورت زیر است:

$$C_{i+1} = (C_i'G_i' + P_i')'$$
$$S_i = (P_iG_i') \oplus C_i$$



شکل (م ۱۷-۴) . اولین طبقه یک جمع‌کننده موازی

نمودار منطقی اولین طبقه یک جمع‌کننده موازی 4 بیت با استفاده از آی‌سی 74283 در شکل (م ۱۷-۴) نشان داده شده است. پایانه‌های P'_1 ، G'_1 را شناسایی کنید و نشان دهید که مدار یک جمع‌کننده کامل را پیاده‌سازی می‌نماید.

۱۸-۴ یک مدار ترکیبی طراحی کنید تا متمم 9 یک رقم BCD را تولید نماید.

۱۹-۴ یک مدار جمع‌کننده کامل BCD بسازید. جمع‌کننده BCD شکل ۱۴-۴ و متمم 9 مسئله ۱۸-۴ را به کار ببرید. برای قطعات از نمودارهای بلوکی استفاده کنید.

۲۰-۴ یک ضرب‌کننده دودویی که دو عدد 4 بیت را در هم ضرب می‌کند طراحی نمایید.

۲۱-۴ یک مدار ترکیبی که دو عدد 4 بیت را برای مساوی بودن مقایسه می‌کند طراحی کنید. خروجی مدار هنگامی 1 است که دو عدد مساوی باشند. در غیر این صورت 0 است.

۲۲-۴ یک دیکدر افزونی 3 به دودویی با استفاده از ترکیبات به‌کار نرفته بی‌اهمیت طراحی کنید.

۲۳-۴ نمودار منطقی یک دیکدر 2 به 4 با فقط گیت NOR بسازید. یک ورودی فعال‌ساز در آن لحاظ کنید.

۲۴-۴ یک دیکدر BCD به دهدهی با ترکیبات بی‌اهمیت کد BCD طراحی کنید.

۲۵-۴ یک دیکدر 5 به 32 خط با دیکدرهای 3 به 8 خط و فعال‌ساز و یک دیکدر 2 به 4 طراحی نمایید. از نمودار بلوکی برای عناصر استفاده کنید.

۲۶-۴ یک دیکدر 4 به 16 خط با پنج دیکدر 2 به 4 همراه با فعال‌ساز بسازید.

۲۷-۴ یک تابع ترکیبی با سه تابع بولی زیر تعریف شده است.

$$F_1(A, B, C) = \Sigma(2, 4, 7)$$
$$F_2(A, B, C) = \Sigma(0, 3)$$
$$F_3(A, B, C) = \Sigma(0, 2, 3, 4, 7)$$

مدار را با یک دیکدر که ازگیت‌های NAND ساخته شده بسازید (مشابه شکل ۱۹-۴). گیت‌های NAND یا AND به خروجی‌های دیکدر متصل‌اند. از نمودار بلوکی برای دیکدر استفاده نمایید. تعداد ورودی‌ها را در گیت‌های بیرونی حداقل نمایید.

۴-۲۸ یک مدار ترکیبی با سه تابع بولی زیر تعریف شده است.

$$F_1 = x'y'z' + xz$$
$$F_2 = xy'z' + x'y$$
$$F_3 = x'y'z + xy$$

مدار را با دیکدر و گیت‌های بیرونی بسازید.

۴-۲۹ یک انکدر اولویت 4 ورودی با ورودی‌هایی مثل جدول ۸-۴ طراحی کنید، ولی D_0 بالاترین اولویت و D_3 پایین‌ترین اولویت را داشته باشد.

۴-۳۰ جدول درستی یک انکدر اولویت هشت هشتی به دودویی را معین کنید. یک خروجی V برای مشخص نمودن این‌که حداقل یکی از ورودی‌ها وجود دارد، فراهم نمایید. ورودی‌که بزرگترین عدد را در اندیس دارد بالاترین اولویت را داراست. اگر ورودی‌های D_5 و D_3 به طور همزمان 1 شوند، مقدار چهار خروجی چیست؟

۴-۳۱ یک مولتی پلکسر 1 × 16 با دو مولتی پلکسر 1 × 8 و یک مولتی پلکسر 1 × 2 بسازید. نمودارهای بلوکی را به‌کار ببرید.

۴-۳۲ تابع بولی زیر را با یک مولتی پلکسر پیاده سازی کنید.

$$F(A, B, C, D) = \Sigma(0, 1, 3, 4, 8, 9, 15)$$

۴-۳۳ یک جمع‌کننده کامل را با دو مولتی پلکسر 1 × 4 پیاده کنید.

۴-۳۴ یک مولتی پلکسر 1 × 8 دارای ورودی‌های A، B و C متصل به ورودی‌های انتخاب S_2، S_1 و S_0 است. ورودی‌های داده I_0 تا I_7 به قرار زیراند: $I_1 = I_2 = I_7 = 0$؛ $I_3 = I_5 = 1$؛ $I_0 = I_4 = D$ و $I_6 = D'$. تابع بولی که با آن مولتی پلکسر پیاده شده را معین نمایید.

۴-۳۵ تابع بولی زیر را با یک مولتی‌پلکسر 1 × 4 و گیت‌های بیرونی پیاده سازی کنید. ورودی‌های A و B را به خطوط انتخاب وصل کنید. ورودی‌های چهار خط داده تابعی از متغیرهای C و D است. این مقادیر با بیان F به عنوان تابعی از C و D برای هر چهار حالت وقتی AB = 00 ، 01 ، 10 ، 11 باشد، بدست می‌آید. این توابع را می‌توان با گیت‌های بیرونی پیاده‌سازی کرد.

$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$$

۴-۳۶ یک توصیف HDL در سطح گیت برای مدار انکدر اولویت شکل ۲۳-۴ بنویسید.

۴-۳۷ یک توصیف سلسله مراتبی HDL در سطح گیت برای جمع- تفریق‌گر 4 بیت با اعداد بی‌علامت بنویسید. مدار مشابه شکل ۱۳-۴ است ولی خروجی V در آن وجود ندارد. می‌توانید جمع‌کننده کامل 4 بیت مثال ۲-۴ HDL را ذکر کنید.

۴-۳۸ یک توصیف روند داده را برای مولتی پلکسر 2 به 1 چهارتایی با فعال‌ساز بنویسید (شکل ۲۶-۴).

۴-۳۹ یک توصیف رفتاری HDL از یک مقایسه‌گر 4 بیت با 6 خروجی Y[5 : 0] بنویسید. بیت 5 از Y برای مساوی، بیت 4 برای نامساوی، بیت 3 بزرگتر از، بیت 2 برای کمتر از، بیت 1 برای بزرگتر یا مساوی، و بیت 0 برای کوچکتر یا مساوی در نظر گرفته شده است.

۴-۴۰ یک توصیف روند داده برای جمع - تفریق‌گر 4 بیت برای اعداد بی‌علامت بنویسید. از مقایسه‌گر شرطی (: ?) استفاده کنید.

۴-۴۱ مسئله ۴-۴۰ را با مدل‌سازی رفتاری تکرار نمایید.

۴-۴۲ (الف) یک توصیف HDL سطح گیت مدار مبدل BCD به افزونی -3 مطابق شکل ۴-۴ بنویسید.

(ب) یک توصیف HDL روند داده برای مبدل BCD به افزونی -3 با استفاده از عبارات بولی لیست شده در شکل ۴-۳ بنویسید.

(پ) یک توصیف رفتاری برای مدار مبدل BCD به افزونی -3 بنویسید.

(ت) یک برنامه تست برای شبیه‌سازی مدار مبدل BCD به افزونی -3 بنویسید تا صحت جدول درستی تحقیق شود. هر سه مدار را چک کنید.

۴-۴۳ عمل مدار را با توصیف HDL زیر توضیح دهید.

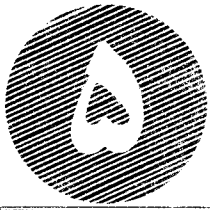
```
module Prob438 (A,B,S,E,Q);
   input  [1:0] A, B;
   input  S, E;
   output [1:0] Q;
   assign Q = E ? (S ? A : B) : 'bz;
endmodule
```

۴-۴۴ یک توصیف مداری HDL برای یک مدار حسابی- منطقی 4 بیت (ALU) بنویسید. مدار دو عمل حسابی و دو عمل منطق را اجرا می‌کند که با ورودی 2 بیت انتخاب می‌شوند. چهار عمل عبارتند از جمع، تفریق، AND و OR.

۴-۴۵ یک توصیف رفتاری HDL برای یک انکدر اولویت 4 بیتی بنویسید. از یک بردار 4 بیت برای ورودی‌های D و یک بلوک always با عبارت if –else استفاده کنید. فرض نمایید که ورودی D[3] دارای بالاترین اولویت باشد.

مراجع

1. DIETMEYER, D. L. 1988. *Logic Design of Digital Systems.* 3rd ed. Boston: Allyn Bacon.
2. GAJSKI, D. D. 1997. *Principles of Digital Design.* Upper Saddle River, NJ: Prentice Hall.
3. HAYES, J. P. 1993. *Introduction to Digital Logic Design.* Reading, MA: Addison-Wesley.
4. KATZ, R. H. 1994. *Contemporary Logic Design.* Upper Saddle River, NJ: Prentice Hall.
5. MANO, M. M. and C. R. KIME. 2000. *Logic and Computer Design Fundamentals,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.
6. NELSON V. P., H. T. NAGLE, J. D. IRWIN and B. D. CARROLL. 1995. *Digital Logic Circuit Analysis and Design.* Upper Saddle River, NJ: Prentice Hall.
7. ROTH, C. H. 1992. *Fundamentals of Logic Design.* 4th ed. St. Paul: West.
8. WAKERLY, J. F. 2000. *Digital Design, Principles and Practices.* 3rd ed. Upper Saddle River, NJ: Prentice Hall.
9. BHASKER, J. 1997. *A Verilog HDL Primer.* Allentown, PA: Star Galaxy Press.
10. BHASKER, J. 1998. *Verilog HDL Synthesis.* Allentown, PA: Star Galaxy Press.
11. CILETTI, M. D. 1999. *Modeling, Synthesis, and Rapid Prototyping with Verilog HDL.* Upper Saddle River, NJ: Prentice Hall.
12. PALNITKAR, S. 1996. *Verilog HDL: A Guide to Digital Design and Synthesis.* SunSoft Press (A Prentice Hall Title).

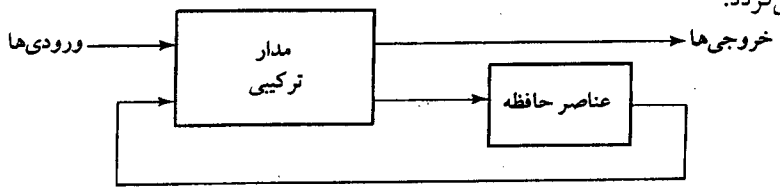


مدارهای منطقی ترتیبی همزمان

۵-۱ مدارهای ترتیبی

مدارهای دیجیتالی که تاکنون بررسی شدند از نوع ترکیبی بودند. در این مدارها خروجی‌ها همه به ورودی‌های جاری وابسته‌اند. گرچه به نظر می‌رسد که هر سیستم دیجیتال دارای مدارهای ترکیبی است، بسیاری از سیستم‌هایی که در عمل با آن مواجه هستیم حاوی عناصر حافظه هم می‌باشند و بنابراین لازم است تا این سیستم‌ها برحسب منطق ترتیبی مورد بررسی قرار گیرند.

نمودار بلوکی یک مدار ترتیبی در شکل ۵-۱ نشان داده شده است. این مدار متشکل از مداری ترکیبی است که عناصر حافظه برای ایجاد یک مسیر پسخورد به آن وصل شده‌اند. عناصر حافظه قطعاتی هستند که می‌توانند اطلاعات دودویی را ذخیره کنند. اطلاعات دودویی ذخیره شده در این عناصر در هر لحظه از زمان حالت مدار ترتیبی در آن زمان است. مدار ترتیبی اطلاعات دودویی را از ورودی‌های بیرونی دریافت می‌کند. این ورودی‌ها همراه با حالت فعلی عناصر حافظه، مقدار دودویی خروجی‌ها را معین می‌نماید. آنها شرط تغییر حالت در عناصر حافظه را نیز معین می‌سازند. نمودار بلوکی نشان می‌دهد که خروجی‌های یک مدار ترتیبی نه فقط تابعی از ورودی‌ها هستند، بلکه به حالت فعلی عناصر حافظه نیز وابسته می‌باشند. حالت بعدی عناصر حافظه نیز تابعی از ورودی‌های بیرونی و حالت فعلی است. بنابراین یک مدار ترتیبی با ترتیب زمانی ورودی‌ها، خروجی‌ها و حالات داخلی مشخص می‌گردد.

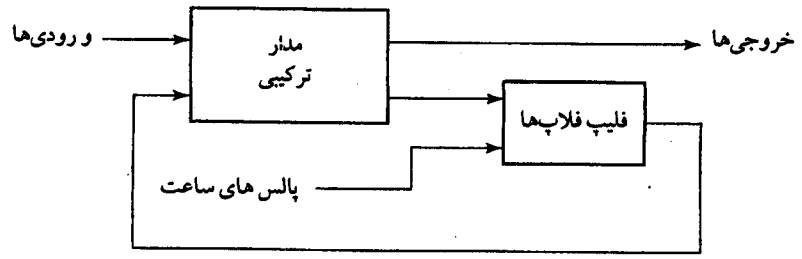


شکل ۵-۱ نمودار بلوکی یک مدار ترتیبی

دو نوع مدار ترتیبی وجود دارد که دسته‌بندی آنها به زمانبندی سیگنال آنها وابسته است. مدار ترتیبی همزمان یا همگام سیستمی است که رفتار آن با توجه به دانش و آگاهی از سیگنال‌هایش در هر لحظه گسسته‌ای از زمان قابل تعریف می‌باشد. رفتار یک مدار ترتیبی غیرهمزمان به ترتیب تغییر سیگنال‌های ورودی آن که می‌توانند در هر لحظه از زمان روی مدار تأثیر کنند وابسته می‌باشد. عناصر حافظه‌ای که به طور معمول در مدارهای ترتیبی غیرهمزمان به کار می‌روند نوعی وسایل تأخیر زمانی هستند. قابلیت نگهداری یک وسیله تأخیر زمانی به زمان انتشار سیگنال در وسیله بستگی دارد. در عمل تأخیر انتشار در گیت‌های منطقی درونی برای ایجاد تأخیر کفایت می‌کند بنابراین واحد تأخیر واقعی می‌تواند مورد نیاز نباشد. در سیستم‌های غیرهمزمان نوع گیتی، عناصر حافظه متشکل از گیت‌های منطقی است که در واقع تأخیر انتشار آنها عمل ذخیره‌سازی را تداعی می‌نماید. بنابراین در چنین مواقعی یک مدار ترتیبی غیرهمزمان را می‌توان مداری ترکیبی با پسخورد دانست. به دلیل وجود پسخورد در بین گیت‌های منطقی، هر مدار ترتیبی غیرهمزمان هر لحظه ممکن است ناپایدار شود. مسئله بی‌ثباتی حاکم مشکلات عدیده‌ای را بر طراح تحمیل خواهد کرد. مدارهای ترتیبی غیرهمزمان در فصل ۹ مورد بحث قرار گرفته‌اند.

با توجه به تعریف، یک مدار ترتیبی همزمان سیگنال‌هایی را مورد استفاده قرار می‌دهد که فقط در لحظات گسسته‌ای از زمان روی عناصر حافظه‌اش اثر می‌گذارد. در این مدارها همزمانی با وسیله‌ای به نام مولد ساعت تحقق می‌یابد و طی آن رشته متناوبی از پالس ساعت به وسیله این دستگاه تولید می‌گردد. پالس‌های ساعت در سرتاسر سیستم توزیع می‌گردند به نحوی که عناصر حافظه تنها هنگام رسیدن هر پالس تحت تأثیر ورودی خود قرار می‌گیرند. در عمل پالس‌های ساعت به همراه دیگر پالس‌ها که تغییرات لازم را در حافظه ایجاد می‌کنند همراه هستند. مدارهای ترتیبی همزمانی که پالس‌های ساعت را در ورودی عناصر ذخیره‌ساز خود به کار می‌برند، مدارهای ترتیبی ساعت‌دار خوانده می‌شوند. ما غالباً در عمل با مدارهای ترتیبی ساعت‌دار مواجه هستیم. آنها مشکل ناپایداری را ندارند و موضوع زمانبندی در آنها به راحتی به مراحلی گسسته و مستقل شکسته می‌شود. هر یک از این مراحل یا برش‌های زمانی مستقلاً قابل بررسی می‌باشند.

عناصر ذخیره‌سازی در مدارهای ترتیبی ساعت‌دار را فلیپ فلاپ می‌گویند. فلیپ فلاپ یک وسیله ذخیره‌سازی دودویی بوده و قادر است یک بیت از اطلاعات را در خود ذخیره نماید. یک مدار ترتیبی ممکن است در صورت لزوم تعداد قابل توجهی از این فلیپ فلاپ‌ها را به کار ببرد. نمودار بلوکی یک مدار ترتیبی ساعت‌دار همزمان در شکل ۲-۵ دیده می‌شود. خروجی‌ها می‌توانند از یک مدار ترکیبی، یا از فلیپ فلاپ‌ها و یا هر دو حاصل شوند. فلیپ فلاپ‌ها ورودی‌های خود را از مدار ترکیبی و نیز از سیگنال ساعت که با فواصل زمانی رخ می‌دهند، طبق نمودار زمانی دریافت می‌کنند. حالت فلیپ فلاپ‌ها تنها هنگام تغییر وضعیت یک پالس ساعت عوض می‌شود. وقتی یک پالس ساعت فعال نیست، حلقه پسخورد قطع می‌شود زیرا حتی اگر خروجی‌های مدار ترکیبی که ورودی آنها را تغذیه می‌کند عوض شود خروجی‌های فلیپ فلاپ تغییر نمی‌نمایند. بنابراین تغییر وضعیت از یک حالت به بعدی فقط در فواصل زمانی دیکته شده به وسیله پالس‌های ساعت امکان‌پذیر است.



(الف) نمودار بلوکی



(ب) نمودار زمان بندی پالس‌های ساعت

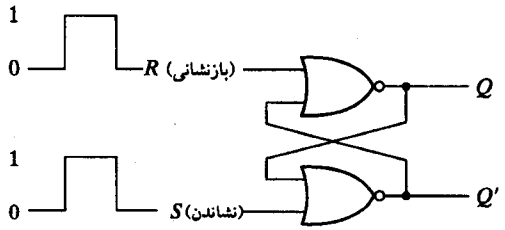
شکل ۲-۵ . مدار ترتیبی ساعت‌دار همزمان

۲-۵ لچ‌ها

یک فلیپ فلاپ می‌تواند یک حالت دودویی را مادامی که تغذیه به مدارش اعمال شود، تا مدتی نامحدود نگهدارد. تفاوت عمده بین انواع فلیپ فلاپ‌ها، در تعداد ورودی‌ها و نحوه تأثیر آنها در تغییر حالت دودویی است. ساده‌ترین انواع فلیپ فلاپ‌ها که با سطوح سیگنال عمل می‌کنند، لچ نامیده می‌شوند. لچ‌ها (یا نگهدارها) مدارهای مبنایی هستند که همه فلیپ فلاپ‌ها با آنها ساخته می‌شوند. گرچه لچ‌ها برای ذخیره اطلاعات دودویی و طراحی مدارهای ترتیبی غیرهمزمان (بخش ۳-۹) مفیدند، ولی عملاً در مدارهای ترتیبی همزمان به کار نمی‌روند. انواع فلیپ فلاپ‌هایی که در مدارهای ترتیبی مورد استفاده قرار می‌گیرند در بخش بعد معرفی شده‌اند.

لچ SR

لچ SR مداری با دو گیت NAND یا NOR است که به طور متقاطع به هم وصل شده‌اند. این مدار دو ورودی دارد که با S به معنی نشاندن (set) و R برای بازنشانی (Reset) نام‌گذاری شده‌اند. لچ SR ساخته شده از دو گیت NOR در شکل ۳-۵ دیده می‌شود. لچ دارای دو حالت مفید است. وقتی



(الف) نمودار منطقی

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(after $S = 1, R = 0$)
0	1	0	1	
0	0	0	1	(after $S = 0, R = 1$)
1	1	0	0	

(ب) جدول تابع

شکل ۳-۵ . لچ SR با گیت‌های NOR

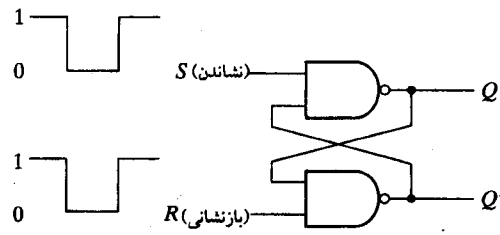
خروجی $Q = 1$ و $Q' = 0$ باشند گوییم که لچ در حالت نشانده (منطق 1) است. اگر $Q = 0$ و $Q' = 1$ باشد گوییم در حالت بازنشانی (منطق 0) است. خروجی‌های Q و Q' متمم یکدیگرند. با این وجود، وقتی هر دو ورودی به طور همزمان 1 شوند، حالت تعریف نشده 0 برای دو خروجی رخ می‌دهد.

تحت شرایط معمولی، هر دو ورودی در 0 نگهداری می‌شوند مگر این که بخواهیم حالت لچ را عوض کنیم. اعمال یک لحظه 1 به ورودی S موجب می‌شود که لچ به حالت 1 برود. قبل از این که حالت تعریف نشده‌ای رخ دهد، ورودی S باید به 0 بازگردد. طبق جدول تابع در شکل ۳-۵ (ب)، دو حالت از ورودی موجب می‌شود تا مدار در حالت 1 قرار گیرد. اولین حالت ($S = 1$ ، $R = 0$) نقشی است که طی آن ورودی S، مدار را به حالت 1 می‌برد. حذف ورودی فعال از S، مدار را در همان حالت باقی می‌گذارد. پس از بازگشت هر دو ورودی به 0، امکان رفتن به حالت 0 میسر خواهد شد، به این ترتیب که برای یک لحظه یک 1 به R اعمال می‌گردد. سپس می‌توان 1 را از R حذف کرد و در این حال مدار در حالت 0 باقی خواهد ماند. بنابراین وقتی هر دو ورودی S و R برابر 0اند، بسته به این که کدام ورودی اخیراً 1 شده است، لچ می‌تواند در حالت 1 یا 0 قرار گیرد.

اگر به طور همزمان به هر دو ورودی R و S، 1 دودویی را اعمال کنیم، هر دو خروجی به 0 می‌روند. این ورودی‌ها حالت تعریف نشده‌ای را در خروجی ایجاد می‌کنند، زیرا حالت بعدی پیش‌بینی نشده‌ای را به هنگام بازگشت دو ورودی به 0 نتیجه می‌دهد. در حالت کار معمولی لچ، با اطمینان از اعمال نشدن همزمان 1 به ورودی‌ها، این وضعیت پرهیز می‌گردد.

لچ SR با دو گیت NAND متقاطع در شکل ۴-۵ مشاهده می‌شود. این مدار به طور معمول با 1 در هر دو ورودی‌اش کار می‌کند مگر این که بخواهیم حالت لچ را تغییر دهیم. اعمال 0 به S موجب می‌شود Q به 1 برود، و لچ را به حالت نشانده وا دارد. وقتی که ورودی S به 1 بازگردد، مدار در همان حالت 1 باقی می‌ماند. پس از بازگشت هر دو ورودی به 1، ما مجاز به تغییر حالت لچ با استقرار 0 در R هستیم. این موجب می‌شود تا مدار به حالت بازنشانی برود و حتی پس از بازگشت هر دو ورودی به 1، لچ در همان حال بماند. حالتی که برای لچ NAND غیر مجاز است، هنگامی است که هر دو ورودی به طور همزمان در 0 باشند. بنابراین از وقوع این حالت باید ممانعت کرد.

با مقایسه لچ NOR با NAND مشاهده می‌شود که سیگنال‌های ورودی برای NAND متمم

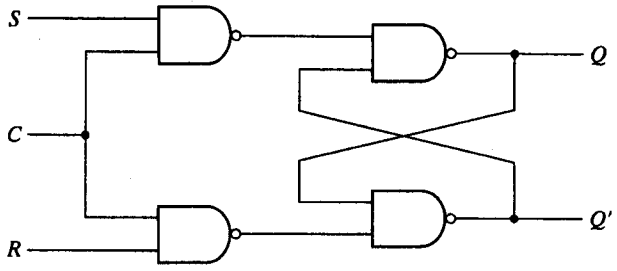


(الف) نمودار منطقی

S	R	Q	Q′	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(ب) جدول تابع

شکل ۴-۵ . لچ SR با گیت‌های NAND



(الف) نمودار منطقی

C	S	R	حالت بعدی Q
0	X	X	بلاتغییر
1	0	0	بلاتغییر
1	0	1	حالت بازنشانی; $Q = 0$
1	1	0	حالت نشاندن ; $Q = 1$
1	1	1	نامعین

(ب) جدول درستی

شکل ۵-۵. لچ SR با ورودی کنترل

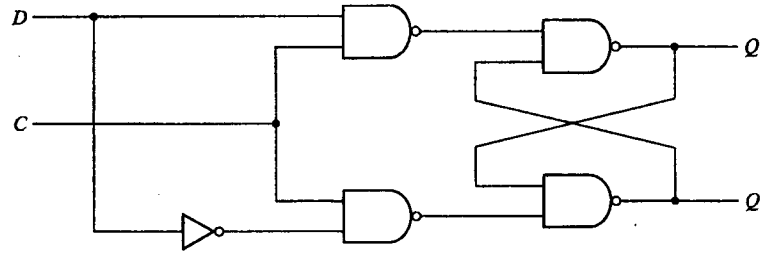
ورودی‌های لچ NOR است. چون لچ NAND برای تغییر حالت خود به سیگنال 0 نیاز دارد، گاهی آن را لچ $S'-R'$ می‌خوانند. علامت پریم یا خط بار بر روی حروف، بیانگر این حقیقت است که ورودی‌ها باید در حالت متمم خود باشند تا مدار را فعال کنند.

عملکرد لچ SR با افزودن یک ورودی کنترل برای تعیین زمان تغییر حالت لچ اصلاح می‌گردد. یک لچ کنترل‌دار در شکل ۵-۵ مشاهده می‌شود. این مدار شامل یک لچ SR پایه و دو گیت NAND اضافی است. ورودی کنترل C به عنوان یک سیگنال فعال‌ساز برای دو ورودی عمل می‌کند. مادامی که ورودی کنترل در 0 باقی بماند، خروجی گیت‌های NAND در سطح منطقی 1 باقی می‌مانند. این وضعیت حالت سکون برای لچ SR است. حالت نشاندن با $S = 1$ ، $R = 0$ و $C = 1$ حاصل می‌گردد. برای تغییر وضعیت باید $S = 0$ ، $R = 1$ و $C = 1$ باشد. در هر حال، وقتی که C به 0 بازگردد، مدار در حالت جاری باقی می‌ماند. در ورودی کنترل با اعمال 0 به C ، مدار غیرفعال می‌شود، به نحوی که عدم تغییر حالت مستقل از مقادیر S و R ، نیز می‌باشد. به علاوه وقتی $C = 1$ باشد، و هر دو ورودی S و R برابر 0 باشند، باز هم حالت مدار تغییر نخواهد کرد. این حالات در جدول تابع در کنار نمودار، ملاحظه می‌شوند.

حالت نامعین هنگامی رخ می‌دهد که هر سه ورودی برابر 1 باشند. این وضعیت، مقدار 0 را روی هر دو ورودی لچ SR پایه قرار می‌دهد، که این ورودی‌ها حالت نامعین را برقرار می‌نمایند. وقتی که ورودی کنترل به 0 بازمی‌گردد، نمی‌توان حالت بعدی را معین کرد زیرا بستگی دارد که کدام یک از دو ورودی S و R زودتر به 0 بروند. این حالت نامعین موجب می‌گردد تا اداره مدار مشکل باشد و بنابراین به ندرت به کار گرفته می‌شود. با این وجود، مدار از اهمیت لازم برخوردار است زیرا دیگر لچ‌ها و فلیپ فلاپ‌ها با آن ساخته می‌شوند.

لچ D

یکی از راه‌های حذف حالت نامطلوب یعنی حالت نامعین یا غیر مجاز در لچ SR این است که مطمئن شویم S و R هرگز به طور همزمان به 1 نمی‌روند. این کار با لچ D شکل ۶-۵ میسراست. این لچ تنها دو ورودی دارد: D (داده) و C (کنترل). ورودی D مستقیماً به ورودی S و متمم آن به ورودی R وصل می‌شود. مادامی که ورودی کنترل در 0 قرار دارد، لچ SR متقاطع دارای 1 در هر دو ورودی بوده و



(الف) نمودار منطقی

C	D	حالت بعدی Q
0	X	بلاتغییر
1	0	حالت بازنشانی; Q = 0
1	1	حالت نشاندن; Q = 1

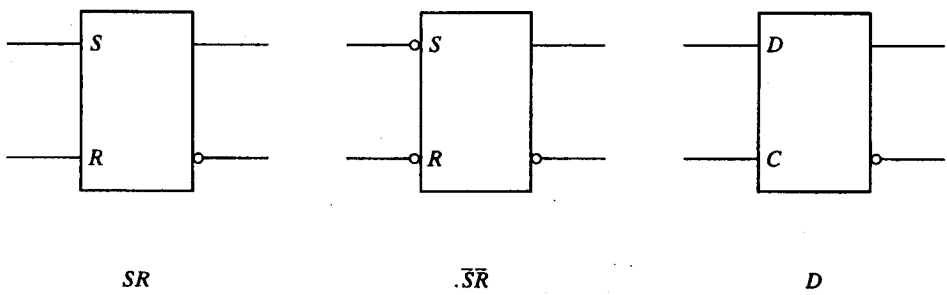
(ب) جدول درستی

شکل ۶-۵. لچ D

مدار نمی‌تواند تغییر حالت دهد. در واقع مقدار D هم نقشی ندارد. وقتی C = 1 باشد ورودی D نمونه‌برداری می‌شود. اگر D = 1 باشد خروجی Q به 1 می‌رود، به این ترتیب مدار در حالت نشانده است. اگر D = 0، خروجی Q به 0 رفته و مدار را به حالت بازنشانی می‌برد.

لچ D نامش را از قابلیت نگهداری داده در درون دریافت کرده است. این لچ برای ذخیره موقت اطلاعات دودویی بین یک محیط و یک واحد مناسب است. اطلاعات دودویی حاضر در ورودی داده لچ D هنگامی که ورودی کنترل فعال شود، به خروجی Q منتقل می‌گردد. مادامی که ورودی کنترل فعال است، خروجی تغییرات ورودی را دنبال می‌کند. این وضعیت مسیری از D به خروجی ایجاد می‌کند، و به این دلیل مدار را لچ شفاف هم می‌خوانند. وقتی ورودی کنترل غیرفعال شود، اطلاعات دودویی حاضر قبلی در ورودی، در خروجی Q باقی می‌ماند تا دوباره ورودی کنترل فعال گردد.

سمبل گرافیک برای انواع لچ در شکل ۷-۵ آمده است. لچ با یک بلوک مستطیلی مشخص می‌شود، که در آن ورودی‌ها در سمت چپ و خروجی‌ها در سمت راست نشان داده می‌شوند. یکی از خروجی‌ها، خروجی معمولی و دیگری متمم خروجی معمولی را نشان می‌دهد. نمودار گرافیک لچ SR دارای ورودی‌های S و R می‌باشد که در داخل بلوک ذکر شده‌اند. در لچ گیت NAND به ورودی‌ها حباب‌هایی اضافه می‌شود که بیانگر نشانده شدن و بازنشانی با سیگنال منطقی 0 است. نمودار گرافیکی برای ورودی‌های D دارای ورودی‌های D و C است که در داخل بلوک مشخص شده‌اند.



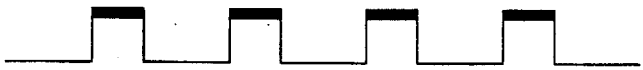
شکل ۷-۵. سمبل‌های گرافیکی لچ

۵-۳ فلیپ فلاپ‌ها

حالت یک لچ یا یک فلیپ فلاپ با تغییر در ورودی کنترل عوض می‌شود. این تغییر لحظه‌ای را ژیگرگویند و انتقال مربوط به آن را تریگر کردن فلیپ فلاپ خوانند. لچ D با پالس‌ها در ورودی کنترلش اساساً یک فلیپ فلاپ است که هر زمان پالس به سطح منطقی 1 برود تریگر می‌شود. مادامی که پالس ورودی کنترل در این سطح بماند هر تغییری در ورودی داده، خروجی و حالت لچ را عوض خواهد کرد.

همانطور که از نمودار بلوکی شکل ۲-۵ ملاحظه می‌شود، یک مدار ترتیبی از خروجی‌های فلیپ فلاپ به ورودی‌های مدار ترکیبی دارای مسیر پسخورد است. در نتیجه ورودی‌های فلیپ فلاپ ممکن است از خروجی همان و یا دیگر فلیپ فلاپ‌ها راه‌اندازی شوند. وقتی که لچ‌ها به عنوان عناصر مورد استفاده قرار گیرند، مشکلی اساسی به وجود می‌آید. به محض تغییر پالس ساعت به منطق 1، انتقال حالت لچ‌ها آغاز می‌شود. در حالی که پالس ساعت هنوز فعال است، حالت جدید لچ در خروجی ظاهر می‌گردد. این خروجی به ورودی لچ‌ها از طریق مدار ترکیبی وصل می‌شود. اگر پالس ساعت در منطق 1، باشد و ورودی اعمال شده به لچ‌ها تغییر کند، لچ به مقادیر جدید واکنش نشان داده و خروجی جدیدی رخ خواهد داد. نتیجه این واکنش وضعیت پیش‌بینی نشده‌ای است زیرا حالت لچ‌ها ممکن است با قرار داشتن پالس ساعت در سطح فعال همچنان به تغییر خود ادامه دهد. به دلیل این عملکرد غیرمطلوب، خروجی یک لچ وقتی همه لچ‌ها به منبع ساعت مشترکی وصلند نمی‌تواند مستقیماً و یا از طریق یک مدار منطقی به همان لچ یا دیگر لچ‌ها وصل شود.

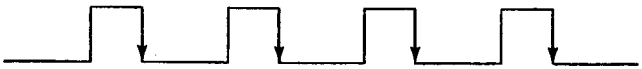
فلیپ فلاپ‌ها طوری ساخته می‌شوند که وقتی بخشی از نوع مدار ترتیبی‌اند و از ساعت مشترکی استفاده می‌کنند، عملکردشان صحیح باشد. مشکل لچ این است که به سطح پالس ساعت پاسخ می‌دهد. همانطور که در شکل ۸-۵ (الف) مشاهده می‌شود، وقتی که پالس ساعت در منطق 1 قرار دارد، هر تغییر مثبت در ورودی کنترل موجب می‌شود تا به ازاء هر تغییر در ورودی D، تغییری در خروجی به وجود



(الف) پاسخ به سطح مثبت



(ب) پاسخ به لبه مثبت



(پ) پاسخ به لبه منفی

شکل ۸-۵. پاسخ ساعت در لچ و فلیپ فلاپ

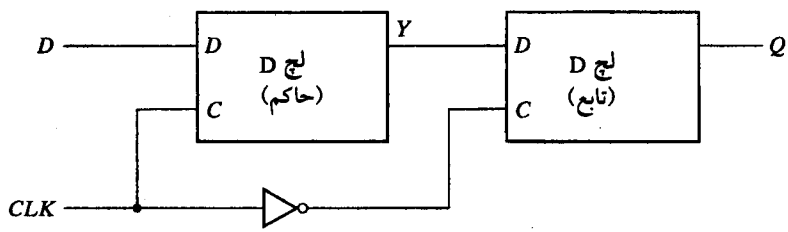
آید. نکته کلیدی در یک عملکرد صحیح فلیپ فلاپ‌ها تریگر شدن آنها در زمان گذر سیگنال است. پالس ساعت از دو انتقال 1 به 0 و 0 به 1 گذر می‌کند. طبق شکل ۸-۵ گذر مثبت به عنوان لبه مثبت و گذر منفی به نام لبه منفی شناخته می‌شود. برای اصلاح یک لچ به یک فلیپ فلاپ، دو راه وجود دارد. یکی از این روش‌ها استفاده از دو لچ با آرایشی خاص است که خروجی فلیپ فلاپ را در حین تغییر ورودی، از آن جدا می‌سازد. راه دیگر تهیه فلیپ فلاپی است که فقط در حین گذر سیگنال تریگر می‌شود (از 0 به 1 یا از 1 به 0) و در بقیه لحظات پالس ساعت غیرفعال است. اکنون هر دو روش را مطالعه می‌کنیم.

فلیپ فلاپ D حساس به لبه

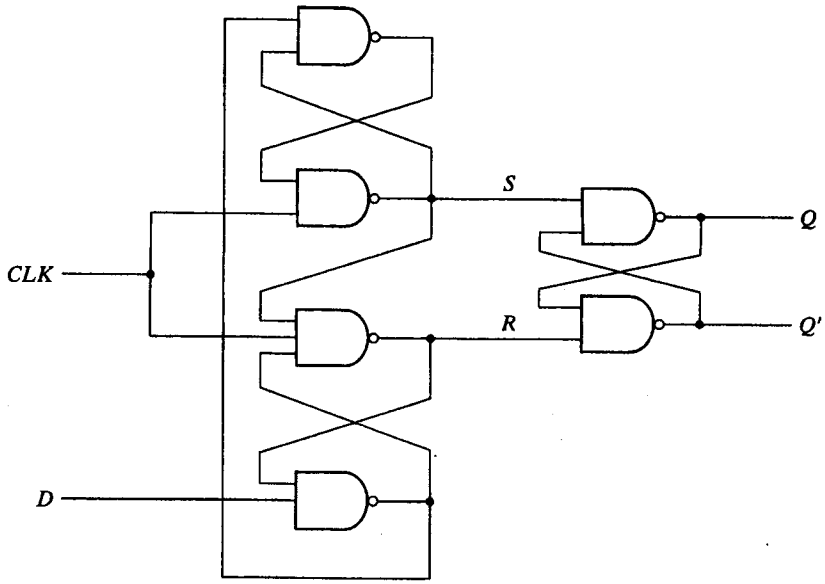
ساخت یک فلیپ فلاپ D با دو لچ D و یک وارونگر در شکل ۹-۵ ملاحظه می‌گردد. اولین لچ را حاکم و دومی را تابع می‌گویند. مدار، ورودی D را نمونه‌برداری کرده و خروجی Q را فقط در لبه منفی پالس کنترل ساعت (CLK) تغییر می‌دهد. وقتی که پالس ساعت در 0 است، خروجی وارونگر 1 می‌باشد. لچ تابع فعال شده و خروجی آن، Q، برابر با خروجی حاکم یعنی Y خواهد شد. لچ حاکم غیرفعال است زیرا CLK = 0 می‌باشد. وقتی که پالس ساعت ورودی به سطح 1 تغییر وضعیت می‌دهد، داده از ورودی بیرونی D به حاکم منتقل می‌گردد. در این حال، مادامی که ساعت در سطح 1 بماند، تابع غیرفعال خواهد بود زیرا ورودی C آن برابر 0 است. هر تغییر در ورودی، خروجی Y را عوض می‌کند، ولی نمی‌تواند خروجی تابع را عوض کند. وقتی که پالس ساعت به 0 بازگردد، حاکم غیرفعال شده و از ورودی D جدا می‌شود. در همان زمان تابع فعال شده و مقدار Y به خروجی فلیپ فلاپ در Q انتقال می‌یابد. بنابراین خروجی فلیپ فلاپ فقط در حین گذر پالس ساعت از 1 به 0 تغییر می‌کند.

رفتار فلیپ فلاپ حاکم - تابع که در بالا توصیف شد نشان می‌دهد که خروجی فقط در لبه منفی پالس ساعت تغییر می‌نماید. این تغییر را می‌توان در لبه مثبت پالس ساعت هم انجام داد. این کار به این ترتیب صورت می‌گیرد که یک وارونگر اضافی بین پایانه CLK و اتصال بین وارونگر دیگر و ورودی C لچ حاکم قرار گیرد. چنین فلیپ فلاپی با لبه منفی پالس عمل کرده و به این ترتیب لبه منفی حاکم و لبه مثبت نیز تابع و پایانه خروجی را عوض می‌کند.

نمونه دیگری از فلیپ فلاپ D حساس به لبه از سه لچ SR، مطابق شکل ۱۰-۵ استفاده می‌کند. دو



شکل ۹-۵. فلیپ فلاپ D حاکم- تابع

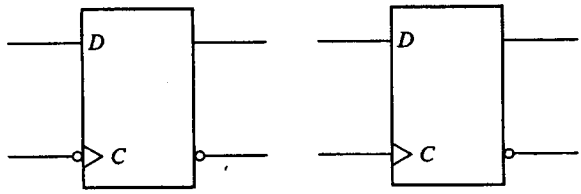


شکل ۱۰-۵. فلیپ فلاپ D حساس به لبه مثبت

لچ موجود در این شکل به ورودی‌های بیرونی D (داده) و CLK (ساعت) پاسخ می‌دهند. لچ سوم خروجی را برای فلیپ فلاپ تهیه می‌کند. ورودی‌های S و R لچ خروجی در CLK = 0 در سطح منطق 1 نگهداری می‌شوند. این موجب می‌شود تا خروجی در حالت فعلی خود باقی بماند. ورودی D ممکن است برابر 0 یا 1 باشد. اگر هنگام 1 شدن CLK ، D = 0 برقرار باشد، R به 0 تغییر می‌کند. یعنی فلیپ فلاپ به حالت بازنشان رفته و در آن Q = 0 می‌گردد. اگر در زمان CLK = 1 تغییری در ورودی رخ دهد پایانه R در 0 می‌ماند. بنابراین فلیپ فلاپ علیرغم تغییر در ورودی خود به حالت قفل باقی خواهد ماند. وقتی که ساعت به 0 بازگردد، R به 1 می‌رود و لچ خروجی در وضعیت ساکن و بدون تغییر در خروجی باقی می‌ماند. به طور مشابه وقتی CLK از 0 به 1 می‌رود، اگر D = 1 باشد، S به 0 تغییر می‌کند. این موجب می‌شود تا مدار به حالت 1 رفته و Q = 1 گردد. هر تغییر در D ، مادامی که CLK = 1 است، نمی‌تواند خروجی را تحت تأثیر قرار دهد.

به طور خلاصه، وقتی ساعت ورودی در فلیپ فلاپ حساس به لبه مثبت یک انتقال مثبت انجام دهد، مقدار D به Q منتقل می‌شود. یک لبه منفی از 1 به 0 تأثیری بر روی خروجی ندارد. به همین ترتیب سطح منطق 1، و نیز سطح منطق 0 هم خروجی را عوض نمی‌کنند. از این رو این نوع فلیپ فلاپ تنها به لبه 0 به 1 و لاغیر پاسخ می‌دهد.

هنگام استفاده از فلیپ فلاپ حساس به لبه باید زمانبندی پاسخ فلیپ فلاپ تحت بررسی قرار گیرد. در این زمانبندی، حداقل زمانی به نام زمان برپایی وجود دارد که طی آن قبل از وقوع گذر ساعت، ورودی باید در مقدار ثابت خود نگهداری شود. به همین ترتیب حداقل زمانی به نام زمان نگهداری وجود دارد



(الف) لبه مثبت (ب) لبه منفی

شکل ۱۱-۵ . سمبل گرافیکی فلیپ فلاپ D حساس به لبه

که طی آن ورودی D نباید پس از اعمال لبه مثبت ساعت، تغییر کند. تأخیر انتشار به صورت فاصله زمانی بین لبه تریگر شدن و تثبیت خروجی در حالت جدید تعریف می‌گردد. این و دیگر پارامترها در برگه‌های اطلاعاتی سازندگان برای هر خانواده منطقی ارائه می‌شوند.

سمبل گرافیکی فلیپ فلاپ D حساس به لبه در شکل ۱۱-۵ مشاهده می‌شود. این سمبل مشابه با سمبل لچ D است به جز این که در جلو حرف C علامت فلشی وجود دارد که دینامیکی بودن ورودی را نشان می‌دهد. نشانگر دینامیک به این معنی است که فلیپ فلاپ به گذر لبه ساعت حساس است. وجود یک حباب در ورودی دینامیکی به معنی نیاز به لبه منفی ساعت است. عدم وجود حباب پاسخ به لبه مثبت را نشان می‌دهد.

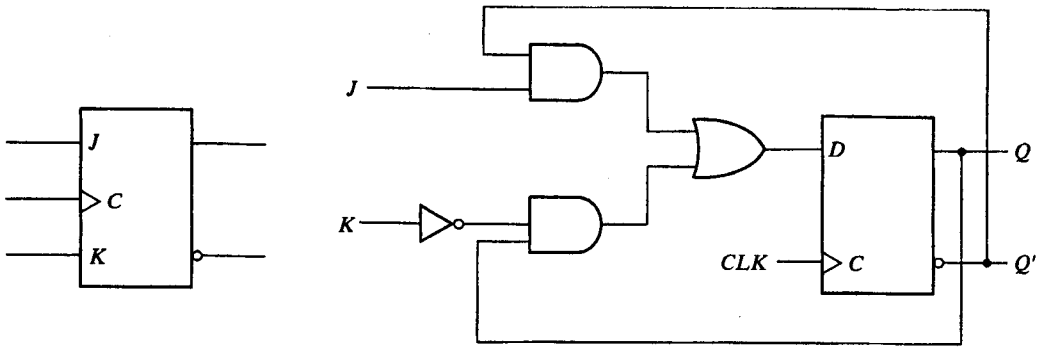
دیگر فلیپ فلاپ‌ها

مدارهای مجتمع VLSI حاوی هزاران گیت در داخل بسته اند. در ساخت یک سیستم دیجیتال انواع گیت‌های درون بسته به هم مرتبط می‌شوند. اقتصادی‌ترین و بهترین فلیپ فلاپ قابل ساخت، نوع D حساس به لبه می‌باشد که به تعداد کمتری گیت نیاز دارد. دیگر فلیپ فلاپ‌ها را می‌توان با فلیپ فلاپ D و مقداری مدار بیرونی به وجود آورد. دو فلیپ فلاپ رایج در طراحی سیستم‌های دیجیتال عبارتند از: فلیپ فلاپ JK و T .

با یک فلیپ فلاپ سه عمل را می‌توان انجام داد: نشاندن در 1 ، بازنشانی در 0 و متمم شدن خروجی. فلیپ فلاپ JK هر سه کار را انجام می‌دهد. نمودار مدار یک فلیپ فلاپ JK که از یک فلیپ فلاپ D ساخته شده است، در شکل ۱۲-۵ (الف) دیده می‌شود. ورودی J، فلیپ فلاپ را در 1 ، ورودی K ، آن را در 0 می‌نشاند، و وقتی هر دو ورودی در 1 قرار گیرند خروجی متمم می‌شود. صحت این مطلب را می‌توان با بررسی مداری که به ورودی D اعمال شده تحقیق کرد:

$$D = JQ' + K'Q$$

وقتی $J = 1$ و $K = 0$ است، $D = Q' + Q = 1$ بوده و بنابراین لبه ساعت بعدی خروجی را در 1 می‌نشاند. وقتی که $J = 0$ و $K = 1$ باشد، لبه پالس بعدی خروجی را به 0 باز می‌نشاند. وقتی هر دو



(ب) سمبل گرافیکی

(الف) نمودار مدار

شکل ۱۲-۵. فلیپ فلاپ D

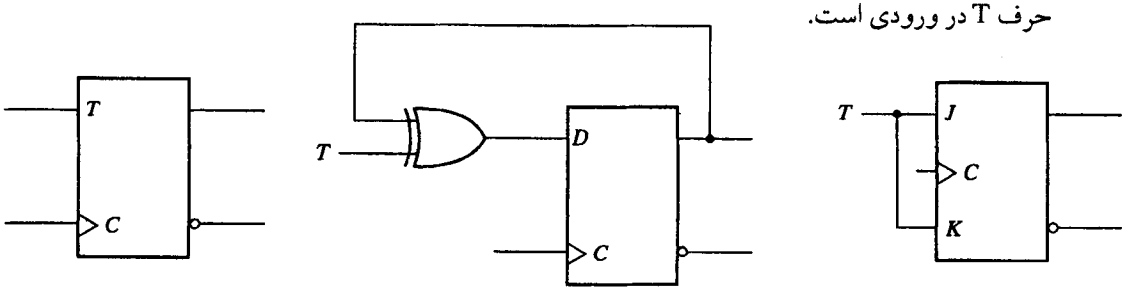
ورودی $J = K = 1$ باشد، $D = Q'$ است و بنابراین لبه ساعت بعدی خروجی را متمم می‌کند. هنگامی که $J = K = 0$ باشد، $D = Q$ است و لبه پالس ساعت بعدی خروجی را بدون تغییر رها خواهد کرد. سمبل گرافیکی برای فلیپ فلاپ JK در شکل ۱۲-۵ (ب) ملاحظه می‌گردد. این سمبل مشابه فلیپ فلاپ D است به جز این که اکنون ورودی‌ها با J و K نام گذاری شده‌اند.

فلیپ فلاپ T (دگر وضع) یک فلیپ فلاپ متمم‌ساز است و می‌توان آن را با گره زدن دو ورودی J و K ایجاد کرد. این عمل در شکل ۱۳-۵ (الف) نشان داده شده است. وقتی $T = 0$ باشد، $(J = K = 0)$، لبه ساعت، خروجی را عوض نمی‌کند. وقتی که $T = 1$ است $(J = K = 1)$، لبه ساعت، خروجی را متمم می‌نماید. فلیپ فلاپ متمم‌ساز در طراحی شمارنده‌های دودویی بسیار مورد توجه است.

یک فلیپ فلاپ T را می‌توان با یک فلیپ فلاپ D و یک گیت XOR مطابق شکل ۱۳-۵ (ب) ساخت. عبارت ورودی D در این حالت برابر است با:

$$D = T \oplus Q = TQ' + T'Q$$

وقتی $T = 0$ است، آنگاه $D = Q$ می‌باشد و بنابراین تغییری در خروجی رخ نمی‌دهد. وقتی $T = 1$ باشد، آنگاه $D = Q'$ بوده و خروجی متمم می‌گردد. سمبل گرافیکی برای این نوع فلیپ فلاپ دارای حرف T در ورودی است.



(پ) سمبل گرافیکی

(ب) با فلیپ فلاپ *D*

(الف) با فلیپ فلاپ *JK*

شکل ۱۳-۵. فلیپ فلاپ D

جدول مشخصه

جدول مشخصه خواص منطقی یک فلیپ فلاپ را تعریف می‌کند و بدین ترتیب عملکرد آن به صورت جدول توصیف می‌گردد. جداول مشخصه سه نوع فلیپ فلاپ در جدول ۱-۵ نشان داده شده است. آنها حالت بعدی را به صورت تابعی از ورودی‌ها و حالت فعلی تعریف می‌نمایند. Q(t) به معنی حالت فعلی و یا حالت قبل از اعمال لبه ساعت است. Q(t+1) ، حالت بعدی پس از اعمال ساعت می‌باشد. توجه کنید ورودی لبه ساعت در جدول مشخصه ذکر نشده است ولی فرض بر این است که بین t و t+1 رخ می‌دهد.

جدول مشخصه فلیپ فلاپ JK نشان می‌دهد که حالت بعدی برابر است با حالت فعلی، به شرطی که J = K = 0 باشد. این وضع را می‌توان به صورت Q(t+1) = Q(t) نشان داد و بیان می‌دارد که تغییری در حالت آن ایجاد نمی‌شود. وقتی که K = 1 و J = 0 باشد، ساعت فلیپ فلاپ را به 0 بازنشانی می‌کند و بنابراین Q(t+1)=0 خواهد شد. اگر J = 1 و K = 0 گردد فلیپ فلاپ به Q(t+1)= 1 می‌رود. وقتی که هر دو ورودی J و K برابر 1 شوند، حالت بعدی متمم حالت فعلی خواهد بود و می‌توان آن را با Q(t+1) = Q′(t) نشان داد.

حالت بعدی فلیپ فلاپ فقط به ورودی D بستگی دارد و مستقل از حالت فعلی است. این حالت را با Q(t+1) = D نشان می‌دهیم. این بدان معنی است که مقدار حالت بعدی برابر با مقدار فعلی (قبل از لبه پالس ساعت) ورودی D است. البته باید توجه کرد که فلیپ فلاپ D حالت بی‌تغییر را دارا نیست. ولی این کار با غیرفعال کردن ساعت و یا با اتصال خروجی به ورودی D انجام می‌شود و طی آن خروجی یا حالت فلیپ فلاپ همواره بی‌تغییر خواهد ماند.

جدول درستی فلیپ فلاپ T فقط دو حالت دارد. وقتی T = 0 باشد، لبه ساعت حالت را تغییر نمی‌دهد. وقتی T = 1 باشد، لبه ساعت حالت فلیپ فلاپ را متمم می‌کند.

جدول ۱-۵ . جداول مشخصه فلیپ فلاپ

فلیپ فلاپ *JK*

J	*K*	*Q*(*t* + 1)	
0	0	*Q*(*t*)	بلاتغییر
0	1	0	بازنشانی
1	0	1	نشاندن
1	1	*Q*′(*t*)	متمم

فلیپ فلاپ *T*

T	*Q*(*t* + 1)	
0	*Q*(*t*)	بلاتغییر
1	*Q*′(*t*)	متمم

فلیپ فلاپ *D*

D	*Q*(*t* + 1)	
0	0	بازنشانی
1	1	نشاندن

معادلات مشخصه

خواص منطقی یک فلیپ فلاپ که در جدول مشخصه ملاحظه شد را می‌توان به صورت معادله مشخصه هم بیان کرد. برای فلیپ فلاپ D، این معادله به صورت زیر است:

$$Q(t + 1) = D$$

این رابطه بیان می‌کند که حالت بعدی خروجی برابر با مقدار ورودی D در حال حاضر است. معادله مشخصه برای فلیپ فلاپ JK را از جدول مشخصه و یا از مدار شکل ۱۲-۵ می‌توان بدست آورد. یعنی

$$Q(t + 1) = JQ' + K'Q$$

که Q مقدار خروجی فلیپ فلاپ قبل از اعمال یک پالس ساعت است. معادله مشخصه برای فلیپ فلاپ T از شکل ۱۳-۵ حاصل می‌شود.

$$Q(t + 1) = T \oplus Q = TQ' + T'Q$$

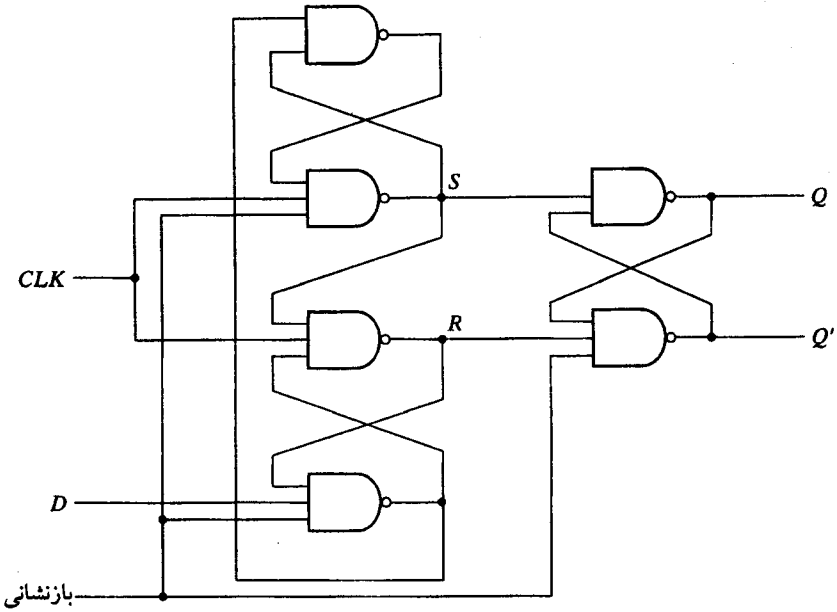
ورودی‌های سیستم

بعضی از فلیپ فلاپ‌ها دارای ورودی‌های غیرهمزمان برای واداشتن آن به یک حالت خاص مستقل از پالس ساعت می‌باشند. ورودی که فلیپ فلاپ را در 1 می‌نشاند، پیش تنظیم یا تنظیم مستقیم می‌نامند. ورودی که فلیپ فلاپ را به 0 پاک می‌کند، ورودی پاک یا بازنشان مستقیم (غیر همزمان) می‌خوانند. وقتی تغذیه در یک سیستم دیجیتال روشن شود، حالت فلیپ فلاپ نامعلوم است. ورودی‌های مستقیم در استقرار همه فلیپ فلاپ‌های سیستم به یک حالت آغازین معلوم، قبل از اعمال پالس ساعت مفید هستند.

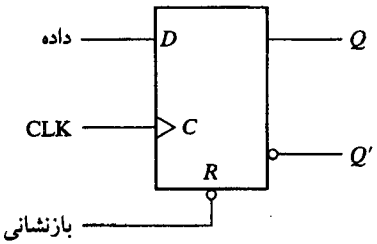
یک فلیپ فلاپ D حساس به لبه مثبت با بازنشانی غیرهمزمان R در شکل ۱۴-۵ ملاحظه می‌شود. نمودار مدار مشابه شکل ۱۰-۵ است با این تفاوت که یک ورودی بازنشانی اضافی، به سه گیت NAND متصل شده است. وقتی که این ورودی در 0 است، Q' را به ماندن در 1 وا می‌دارد، و این به نوبه خود به معنی پاک شدن خروجی Q به 0 است و بنابراین فلیپ فلاپ بازنشانی می‌شود. دو اتصال دیگر از ورودی بازنشان بقاء سومین لچ SR را در منطق 1 تضمین می‌کند. این وضع هنگامی رخ می‌دهد که ورودی بازنشانی، بدون توجه به مقادیر D و CLK، در 0 باشد.

سمبل گرافیکی فلیپ فلاپ D با یک ورودی بازنشان مستقیم دارای یک ورودی اضافی است که با R علامت‌گذاری شده است. وجود حباب در ورودی به این معنی است که بازنشانی با سطح منطق 0 فعال می‌گردد. فلیپ فلاپ‌هایی که از نشاندن مستقیم استفاده می‌کنند از سمبل S در ورودی نشاندن غیرهمزمان استفاده می‌کنند.

جدول تابع، عملکرد مدار را مشخص می‌کند. وقتی $R = 0$ باشد، خروجی به 0 بازنشانی می‌شود. این حالت مستقل از D و C است. هنگامی مدار می‌تواند به روند عادی خود بازگردد که ورودی بازنشانی به 1 برود. ساعت در C با یک فلش رو به بالا، که به معنی عملکرد فلیپ فلاپ در لبه مثبت ساعت می‌باشد، نشان داده شده است. مقدار D با هر لبه مثبت سیگنال ساعت، به شرطی که $R = 1$ باشد، به خروجی Q منتقل می‌گردد.



(الف) نمودارمدار



(ب) سمبل گرافیکی

R	C	D	Q	Q′
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(پ) جدول درستی

شکل ۱۴-۵. فلیپ فلاپ D با بازنشانی غیرهمزمان

۵-۴ تحلیل مدارهای ترتیبی ساعت‌دار

رفتار یک مدار ترتیبی ساعت‌دار با ورودی‌ها، خروجی‌ها و حالت فلیپ فلاپ‌ها مشخص می‌گردد. خروجی‌ها و حالت بعدی هر دو تابعی از ورودی‌ها و حالت فعلی‌اند. تحلیل یک مدار ترتیبی به معنی تهیه جدول یا نموداری از رشته زمانی ورودی‌ها، خروجی‌ها و حالات درونی است. می‌توان عبارات بول را نوشت و به وسیله آنها رفتار مدار را توصیف کرد. این عبارات باید رشته زمانی لازم را چه مستقیماً و چه غیرمستقیم مشخص کند.

یک نمودار منطقی، وقتی دارای فلیپ فلاپ‌ها با ورودی‌های ساعت باشد، مدار ترتیبی ساعت‌دار

خوانده می‌شود. فلیپ فلاپ‌ها می‌توانند از هر نوع، و نمودار منطقی هم ممکن است شامل گیت‌های ترکیبی باشد یا نباشد. در این بخش، ما یک نمایش جبری را برای تعیین حالت بعدی برحسب حالت فعلی و ورودی‌ها ارائه می‌کنیم. آنگاه برای توصیف رفتار یک مدار ترتیبی، یک جدول حالت و یک نمودار حالت ارائه می‌شود. یک عبارت جبری دیگر هم برای مشخص کردن نمودار منطقی مدارهای ترتیبی بیان می‌گردد. برای تشریح روال‌های مختلف، مثال‌های خاصی آورده شده است.

معادلات حالت

رفتار مدار ترتیبی ساعت‌دار را می‌توان با معادلات حالت توصیف کرد. یک معادله حالت (که به آن معادله گذر هم می‌گویند) حالت بعدی را برحسب تابعی از حالات فعلی و ورودی‌ها بیان می‌نماید. مدار شکل ۱۵-۵ را ملاحظه نمایید. این مدار از دو فلیپ فلاپ A و B نوع D و یک ورودی x و یک خروجی y تشکیل شده است. چون ورودی D یک فلیپ فلاپ، مقدار حالت بعدی را معین می‌کند، می‌توان مجموعه معادلاتی را به صورت زیر برای مدار نوشت:

$$A(t + 1) = A(t)x(t) + B(t)x(t)$$
$$B(t + 1) = A'(t)x(t)$$

یک معادله حالت معادله‌ای است که شرایط گذر حالت را برای یک فلیپ فلاپ بیان می‌کند. سمت چپ معادله با (t+1) حالت بعدی فلیپ فلاپ را پس از یک لبه ساعت معین می‌نماید. سمت راست معادله عبارتی است بولی که حالت فعلی و وضعیت ورودی‌هایی را مشخص می‌نمایند که در قبال آنها حالت بعدی 1 می‌گردد. چون همه متغیرها در عبارت بول تابعی از حالت فعلی هستند، ما از نوشتن (t) پس از متغیر صرف‌نظر کرده و معادلات حالت را به صورت فشرده‌تری مطابق زیر می‌نویسیم:

$$A(t + 1) = Ax + Bx$$
$$B(t + 1) = A'x$$

عبارات بولی برای معادلات حالت مستقیماً از گیت‌های تشکیل‌دهنده بخش ترکیبی در مدار ترتیبی بدست می‌آیند، زیرا مقادیر D در مدار ترکیبی حالت بعدی را تعیین می‌کنند. به طور مشابه مقدار فعلی خروجی نیز قابل ارائه به صورت جبری زیر است:

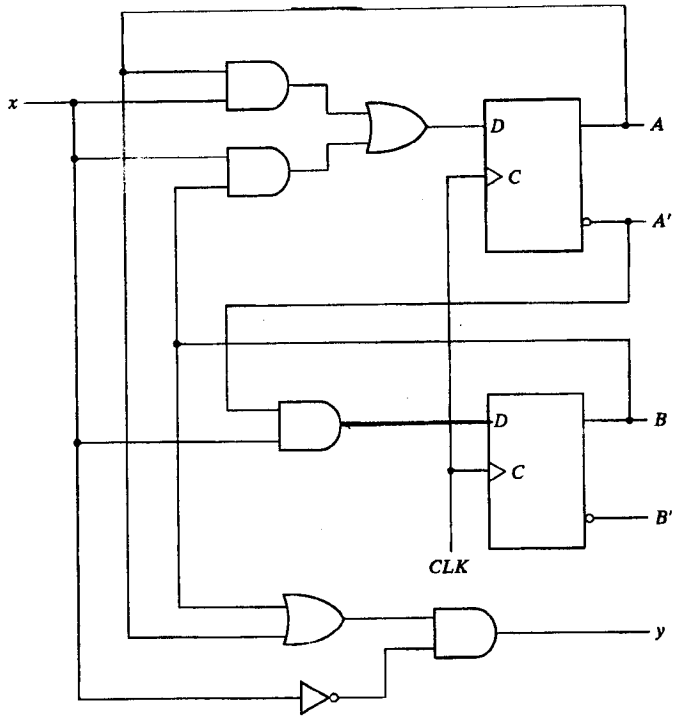
$$y(t) = [A(t) + B(t)]x'(t)$$

با حذف سمبل (t) از مقدار فعلی، معادله بولی خروجی زیر بدست می‌آید:

$$y = (A + B)x'$$

جدول حالت

رشته‌های زمانی ورودی‌ها، و خروجی‌ها و حالات فلیپ فلاپ را می‌توان در یک جدول حالت (به آن جدول گذر هم می‌گویند) جمع‌آوری کرد. جدول حالت برای مدار شکل ۱۵-۵ در جدول ۲-۵ دیده می‌شود. جدول متشکل از چهار بخش با نام‌های حالت فعلی، ورودی، حالت بعدی و خروجی است.



شکل ۱۵-۵. مثال مدار ترتیبی

بخش حالت فعلی، حالت فلیپ فلاپ‌های A و B را در هر لحظه از زمان t نشان می‌دهد. بخش ورودی مقدار x را برای هر حالت فعلی ممکن بدست می‌دهد. بخش حالت بعدی، وضعیت فلیپ فلاپ‌ها را یک سیکل ساعت بعد، در زمان t+1 بیان می‌دارد. بخش خروجی مقدار y را در هر زمان t در قبال هر حالت فعلی با توجه به شرایط ورودی، مشخص می‌کند.

تهیه جدول حالت به لیستی از همه ترکیبات دودویی حالت فعلی و ورودی‌ها نیاز دارد. در این حال،

جدول ۲-۵. جدول حالت برای شکل ۱۵-۵

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B	x	A	B	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

ما هشت ترکیب دودویی 000 تا 111 را داریم. سپس مقادیر حالت بعدی از نمودار منطقی یا از معادلات حالت بدست می‌آیند. حالت بعدی فلیپ فلاپ A باید در معادله زیر صدق کند.

$$A(t + 1) = Ax + Bx$$

بخش حالت بعدی در جدول حالت در زیر ستون A دارای سه عدد 1 است که در قبال آنها حالت فعلی و مقدار ورودی شرایطی را که حالت فعلی A و ورودی x هر دو برابر 1 هستند و یا حالت فعلی B و ورودی x هر دو برابر 1 می‌باشند برآورده می‌سازند. به طور مشابه حالت بعدی فلیپ فلاپ B از معادله حالت زیر حاصل می‌گردد.

$$B(t + 1) = A'x$$

و هنگامی برابر 1 است که حالت فعلی $A = 0$ و ورودی $x = 1$ باشد. ستون خروجی از معادله زیر حاصل می‌شود:

$$y = Ax' + Bx'$$

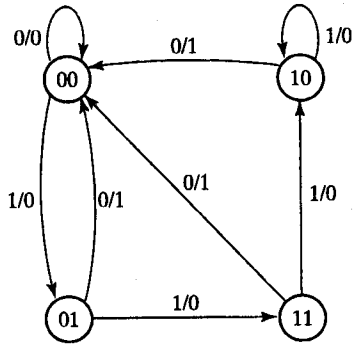
جدول حالت یک مدار ترتیبی با فلیپ فلاپ‌های نوع D با روال مشابهی بدست می‌آید. به طور کلی، یک مدار ترتیبی با m فلیپ فلاپ و n ورودی نیاز به 2^{m+n} سطر در جدول حالت دارد. اعداد دودویی از 0 تا $2^{m+n}-1$ در زیر ستون‌های حالت فعلی و ورودی لیست شده‌اند. بخش حالت بعدی دارای m ستون، یعنی یک ستون در ازاء هر فلیپ فلاپ، می‌باشد. مقادیر دودویی برای حالت بعدی مستقیماً از معادلات حالت حاصل می‌گردند. بخش خروجی دارای ستونهایی به تعداد خروجی‌هاست. مقدار دودویی این بخش مشابه با جدول درستی از مدار یا تابع بولی آن بدست می‌آید. گاهی بهتر است تا جدول حالت را با کمی تغییر نشان دهیم. در آرایشی دیگر، جدول حالت تنها سه بخش دارد که عبارتند از: حالت فعلی، حالت بعدی و خروجی. حالات ورودی در زیر ستون حالت بعدی و ستون خروجی ذکر می‌شود. جدول حالت ۲-۵ با توجه به این روش به جدول ۳-۵ تبدیل شده است. برای هر حالت فعلی، بسته به مقدار ورودی، دو حالت ممکن برای حالت بعدی و خروجی وجود دارد. بسته به نوع کاربرد هر یک از دو روش فوق بر دیگری ارجحیت دارد.

نمودار حالت

اطلاعات موجود در جدول حالت را می‌توان به صورت گرافیکی با نمودار حالت نشان داد. در این

جدول ۳-۵ . فرم دیگر جدول حالت

حالت فعلی	حالت بعدی		خروجی	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
AB	AB	AB	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0



شکل ۱۶-۵. نمودار حالت مدار شکل ۱۵-۵

نوع نمودار، یک حالت با یک دایره نشان داده می‌شود و گذر در بین حالات با خطوط جهت‌داری که دو دایره را به هم وصل می‌کنند نمایش داده می‌شود. نمودار حالت مدار ترتیبی شکل ۱۵-۵ در شکل ۱۶-۵ ملاحظه می‌گردد. نمودار حالت همان اطلاعات جدول حالت را بیان می‌کند که مستقیماً از جدول ۲-۵ یا ۳-۵ بدست می‌آید. عدد دودویی داخل هر دایره حالت فلیپ فلاپ‌ها را بیان می‌نماید. خطوط جهت‌دار با دو عدد که با یک خط مورب از هم جدا شده‌اند، برچسب خورده‌اند. مقدار ورودی در حالت فعلی در سمت چپ این خط و عدد پس از خط مورب، خروجی را در حالت فعلی در قبال ورودی مربوطه‌اش نشان می‌دهد. باید توجه داشت مقدار خروجی ذکر شده در کنار خطوط جهت‌دار، در حین حالت فعلی و ورودی مربوطه رخ می‌دهد و هیچ ارتباطی به حالت بعدی ندارد. مثلاً خط واصل جهت‌دار که از حالت 00 به 01 می‌رود با 1/0 برچسب خورده است و به این معنی است که وقتی مدار ترتیبی در حالت فعلی 00 است ورودی 1 و خروجی 0 می‌باشد، در پالس ساعت بعدی، مدار به حالت بعدی 01 می‌رود. اگر ورودی به 0 تغییر یابد، آنگاه خروجی 1 می‌گردد ولی اگر ورودی در 1 باقی بماند، خروجی در 0 خواهد ماند. این اطلاعات از نمودار حالت و خطوط جهت‌دار که از دایره 01 سرچشمه گرفته، حاصل شده است. یک خط جهت‌دار که دایره را به خودش وصل کند، به معنی عدم وجود تغییر در حالت است.

بین جدول حالت و نمودار حالت تفاوتی به جز نحوه ارائه وجود ندارد. جدول درستی به راحتی از یک معادله حالت و نمودار منطقی حاصل می‌گردد. نمودار حالت مستقیماً از جدول حالت بدست می‌آید. نمودار حالت تصویری از گذر حالات را مجسم می‌کند و برای تفسیر عملکرد مدار مناسب‌تر است. مثلاً، نمودار حالت شکل ۱۶-۵ به وضوح نشان می‌دهد که، با شروع از حالت 00 ، مادامی که ورودی در 1 باشد خروجی برابر 0 است. اولین ورودی 0 بعد از رشته‌ای از 1 ، خروجی 1 را تولید کرده و مدار را به 00 اولیه باز می‌گرداند.

معادلات ورودی فلیپ فلاپ

نمودار منطقی یک مدار ترتیبی متشکل از فلیپ فلاپ‌ها و گیت‌هاست. اتصالات میان گیت‌ها

مدار ترکیبی را می‌سازند و ممکن است با عبارات بولی نشان داده شوند. آگاهی از نوع فلیپ فلاپ‌ها و لیست عبارات بولی مدار ترکیبی، اطلاعات لازم را برای ترسیم نمودار منطقی مدار ترتیبی فراهم می‌سازد. بخشی از مدار ترکیبی که خروجی‌های بیرونی را تولید می‌کند و به صورت توابع بولی توصیف می‌گردند معادلات خروجی نامیده می‌شوند. بخشی از مدار که ورودی‌های فلیپ فلاپ‌ها را تولید می‌کنند با توابع بولی به نام معادلات ورودی فلیپ فلاپ نام‌گذاری شده‌اند (گاهی به آنها معادلات تحریک هم می‌گویند). ما از سمبل ورودی فلیپ فلاپ برای نام‌گذاری متغیر معادله ورودی و نام خروجی فلیپ فلاپ‌ها به عنوان اندیس استفاده خواهیم کرد. مثلاً معادله ورودی زیر یک گیت OR، با ورودی‌های x و y که به ورودی D از فلیپ فلاپ متصلند و خروجی آن با Q نام‌گذاری شده است را نشان می‌دهد.

$$D_Q = x + y$$

مدار ترتیبی شکل ۱۵-۵ متشکل از دو فلیپ فلاپ A و B از نوع D، یک ورودی x و یک خروجی y است. نمودار منطقی مدار می‌تواند به صورت جبری با دو معادله ورودی و یک معادله خروجی بیان شود:

$$D_A = Ax + Bx$$
$$D_B = A'x$$
$$y = (A + B)x'$$

سه معادله فوق اطلاعات لازم را برای ترسیم نمودار منطقی مدار ترتیبی فراهم می‌سازند. سمبل D_A یک فلیپ فلاپ D با نام A را مشخص می‌نماید. به همین ترتیب D_B فلیپ فلاپ B از نوع D است. عبارات بولی مربوط به این دو متغیر و عبارت خروجی y، بخش ترکیبی مدار ترتیبی را معین می‌کنند.

معادلات ورودی فلیپ فلاپ‌ها فرم جبری مناسبی را برای نمودار منطقی یک مدار ترتیبی تشکیل می‌دهند. آنها نوع فلیپ فلاپ را با توجه به سمبل فلیپ فلاپ مشخص می‌نمایند و مدار ترکیبی که فلیپ فلاپ‌ها را راه می‌اندازند هم با آنها مشخص می‌شود. توجه کنید که عبارت معادله ورودی با عبارت مربوط به معادله حالت یکی است. دلیل این است که معادله مشخصه با مقدار ورودی به D برابر است: یعنی $Q(t+1) = D_Q$.

تحلیل با کمک فلیپ فلاپ‌های D

در اینجا روال تحلیل یک مدار ترتیبی متشکل از فلیپ فلاپ‌های D را با یک مثال ساده خلاصه می‌کنیم. مداری که برای این هدف در نظر گرفته شده با معادله ورودی زیر توصیف گردیده است.

$$D_A = A \oplus x \oplus y$$

سمبل D_A یک فلیپ فلاپ نوع D با خروجی A را بیان می‌کند. متغیرهای x و y، ورودی‌ها به مدار هستند. هیچ معادله خروجی مشخص نشده، بنابراین خروجی مدار از خروجی فلیپ فلاپ اخذ شده است. نمودار منطقی از معادله ورودی حاصل و در شکل ۱۷-۵ (الف) رسم شده است.

جدول حالت برای حالت فعلی یک ستون داشته و متعلق به فلیپ فلاپ A است، دو ستون هم برای ورودی‌ها و یک ستون برای حالت بعدی A لازم است. اعداد دودویی زیر ستون Axy از 000 تا 111