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

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

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

مثال ۱۰-۳

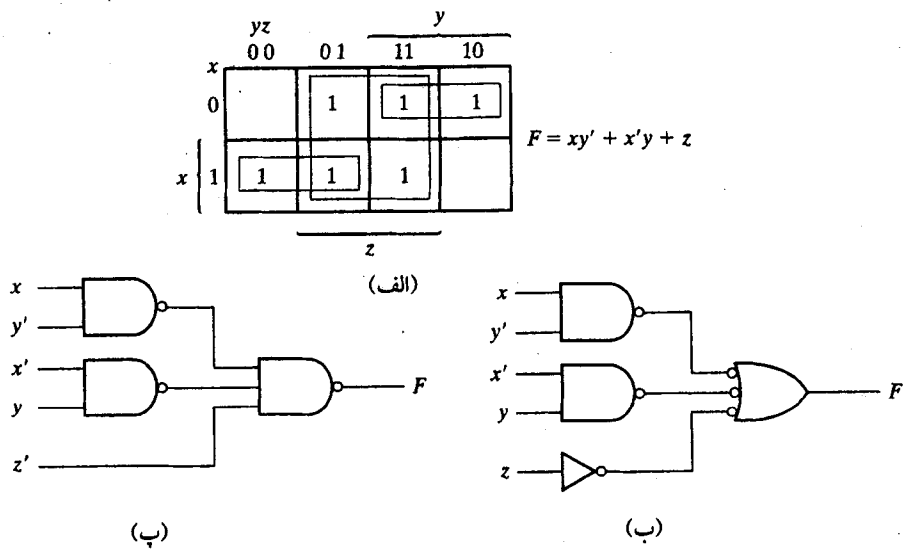
تابع بول زیر را با گیت‌های NAND پیاده کنید.

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

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

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

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



شکل ۲۱-۳. حل مثال ۱۰-۳

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

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

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

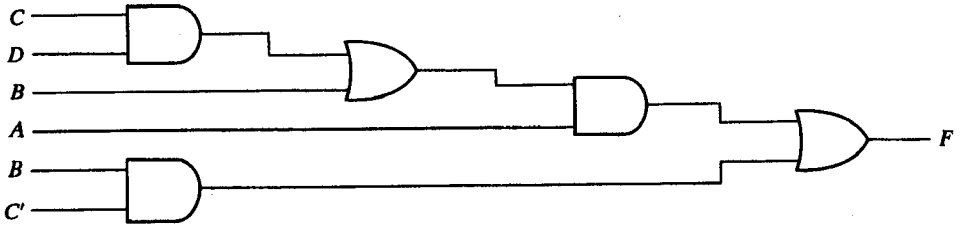
مدارهای NAND چند سطحی

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

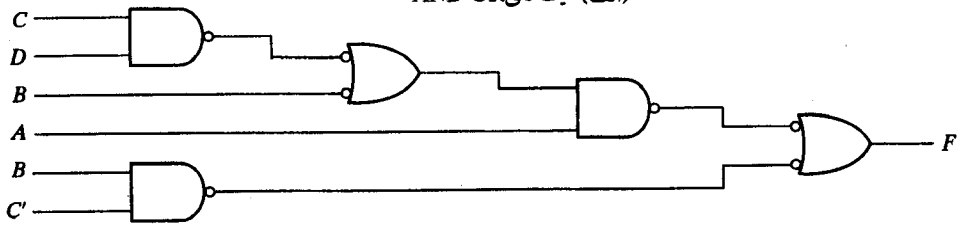
$$F = A(CD + B) + BC'$$

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

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



(الف) گیت‌های AND-OR



(ب) گیت‌های NAND

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

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

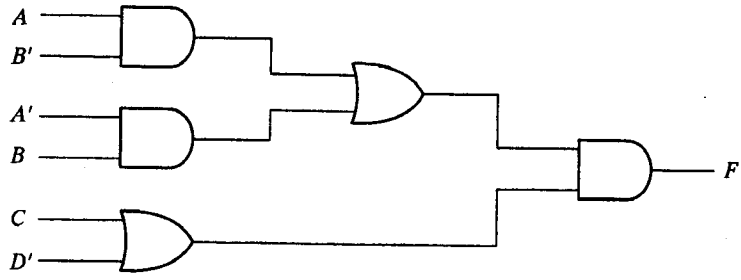
به عنوان مثالی دیگر تابع بول چند سطحی زیر را ملاحظه کنید.

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

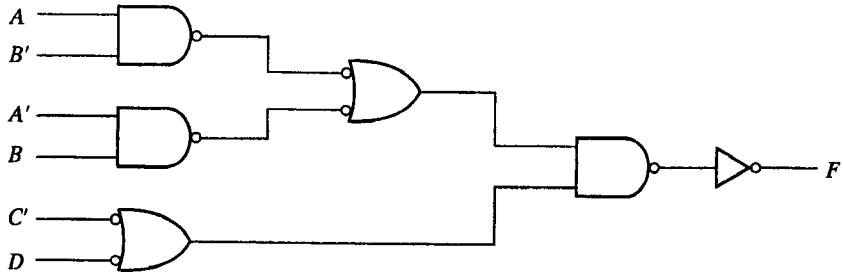
پیاده‌سازی AND–OR تابع در شکل ۲۳-۳ (الف) ملاحظه می‌شود که در آن سه طبقه گیت به کار رفته است. در بخش (ب) نمودار فرم تبدیل شده به NAND آن با علائم مخلوط دیده می‌شود. دو حباب اضافی مربوط به ورودی‌های C و D' موجب متمم شدن آنها به C' و D می‌گردد. حباب موجود در گیت NAND خروجی، مقدار خروجی را متمم می‌کند بنابراین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

پیاده‌سازی NOR

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

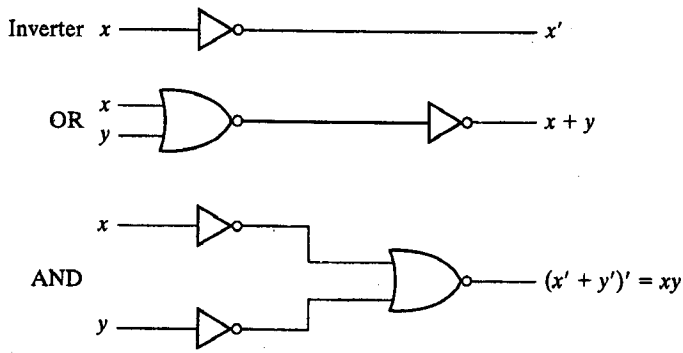


(الف) گیت‌های AND-OR



(ب) گیت‌های NAND

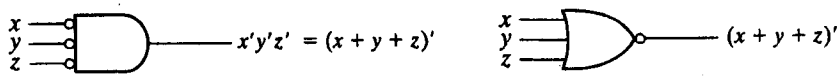
شکل ۲۳-۳. پیاده‌سازی $F = (AB' + A'B)(C + D')$



شکل ۲۴-۳. عملیات منطقی با گیت‌های NOR

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

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



Invert-AND (ب)　　OR-invert (الف)

شکل ۲۵-۳. دو سمبل گرافیکی برای گیت NOR

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

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

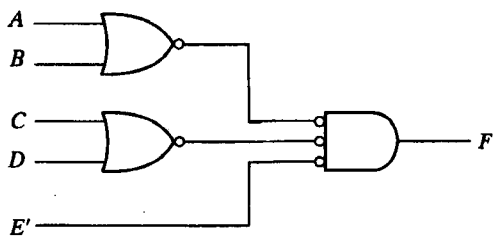
الگوی OR–AND را به سادگی با حذف حباب‌ها در طول هر مسیر می‌توان شناسایی کرد. متغیر E برای جبران سومین حباب در ورودی گیت سطح دوم متمم شده است.

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

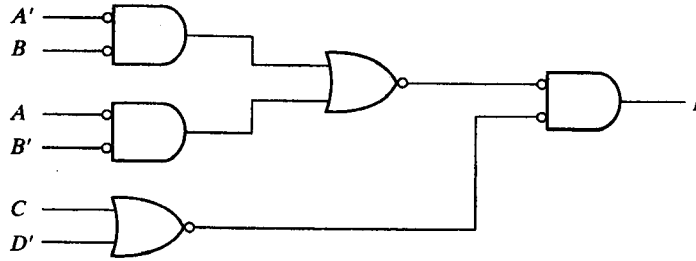
تبدیل نمودار AND-OR شکل ۲۳-۳(الف) به یک نمودار NOR در شکل ۲۷-۳ نشان داده شده است. تابع بول برای این مدار به شکل زیر است

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

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



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



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

۳-۷ دیگر پیاده‌سازی‌های دو سطحی

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

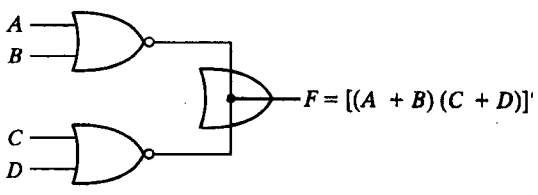
$$F = (AB)' \cdot (CD)' = (AB + CD)'$$

و به آن تابع AND–OR–INVERT می‌گویند.

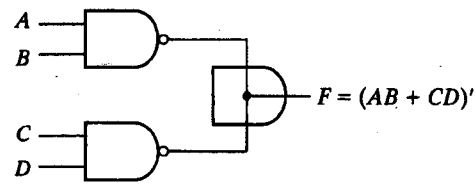
به طور مشابه خروجی NOR گیت‌های ECL برای اجرای یک تابع OR سیمی به هم گره زده می‌شوند. تابع منطقی پیاده‌سازی شده با مدار شکل ۲۸-۳ (ب) چنین است:

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

و به آن تابع OR–AND–INVERT می‌گویند.



(ب) OR سیمی در گیت‌های ECL



(الف) AND سیمی در گیت‌های TTL NAND کلکتور باز

شکل ۲۸-۳. منطق سیمی

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

فرم‌های مفید گیت‌ها

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

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

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

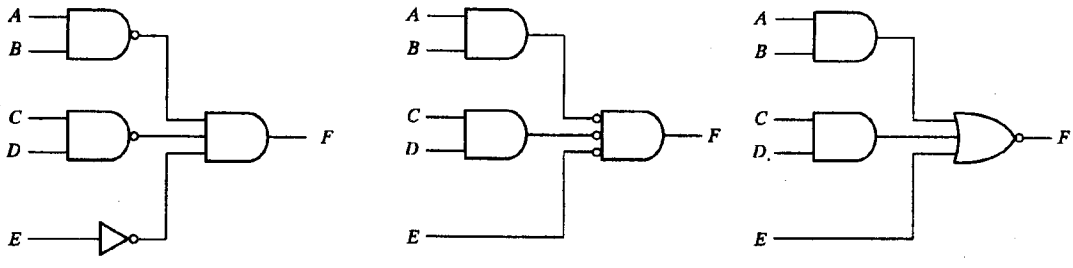
فرم‌های AND–OR و OR–AND، فرم‌های دو سطح اصلی بحث شده در بخش ۴-۳ می‌باشند. فرم‌های NAND–NAND و NOR–NOR در بخش ۶-۳ ارائه شدند. چهار فرم باقیمانده نیز در این بخش بررسی می‌شوند.

پیاده‌سازی AND–OR–INVERT

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

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

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



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

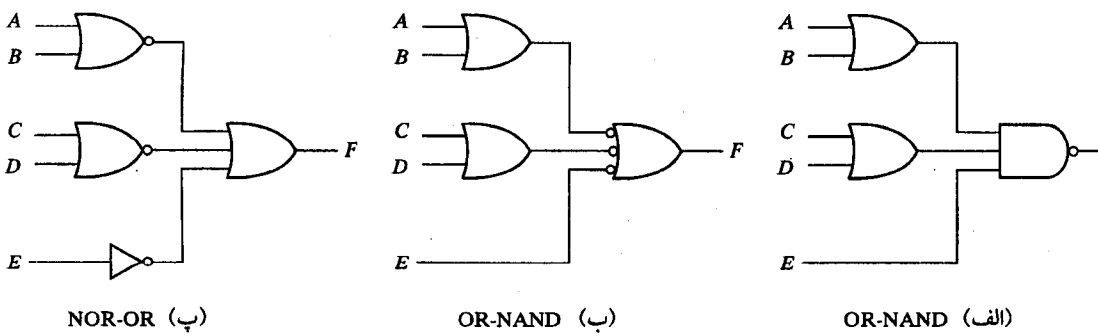
اول انتقال می‌دهیم. برای جبران هر حباب یک وارونگر در ازاء هر متغیر لازم است. به همین ترتیب می‌توان وارونگر را حذف کرد به شرطی که E متمم شود. مدار شکل ۲۹-۳ (پ)، فرم NAND–AND است و برای پیاده‌سازی تابع AND–OR–INVERT در شکل ۲۸-۳ نشان داده شده است.

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

پیاده‌سازی OR–AND–INVERT

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

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



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

جدول ۳-۳. پیاده‌سازی با دیگر فرم‌های دو طبقه

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

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

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

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

خلاصه جدول وار و مثال

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

مثال ۱۱-۳

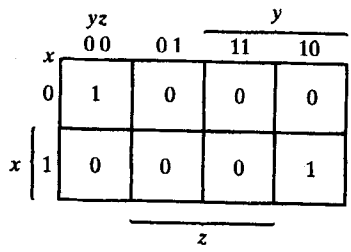
تابع شکل ۳۱-۳ (الف) را به فرم دو سطحی لیست شده در جدول ۳-۳ پیاده‌سازی کنید.

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

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

خروجی نرمال این تابع می‌تواند به صورت زیر بیان شود.

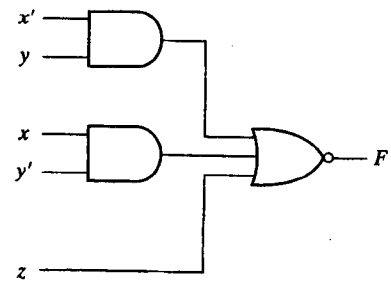
$$F = (x'y + xy' + z)'$$



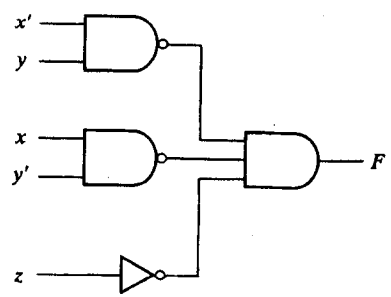
$F = x'y'z' + xyz'$

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

(الف) ساده سازی نقشه در جمع حاصلضرب ها

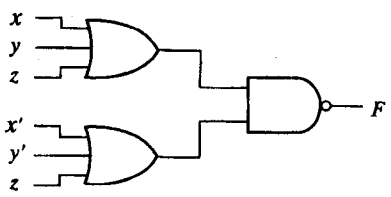


AND-NOR

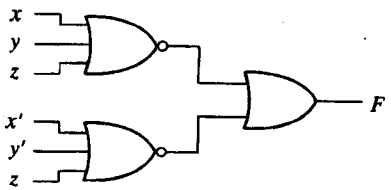


NAND-AND

$F = (x'y + xy' + z)'$ (ب)



OR-NAND



NOR-OR

$F = [(x + y + z)(x' + y' + z)]'$ (پ)

شکل ۳۱-۳. دیگر پیاده‌سازی‌های دو طبقه

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

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

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

آنگاه متمم تابع را بدست می‌آوریم.

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

خروجی نرمال F اکنون به فرم زیر نوشته می‌شود.

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

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

❑

۳-۸ تابع OR انحصاری

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

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

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

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

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

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

روابط زیر در مورد OR انحصاری معتبرند

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

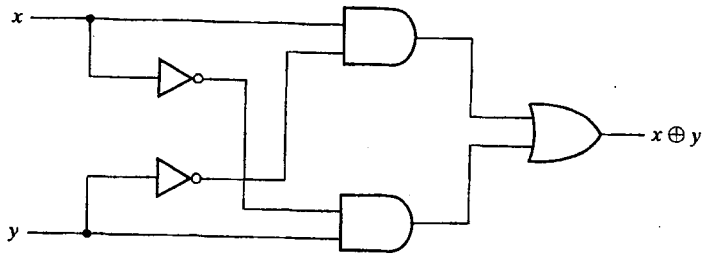
هر یک از این روابط می‌تواند با به کارگیری جدول درستی و جایگزینی ⊕ با عبارت بولی هم‌ارزی ثابت شود. و نیز می‌توان نشان داد که عمل OR انحصاری خاصیت جابجایی و شرکت‌پذیری را دارد. یعنی:

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

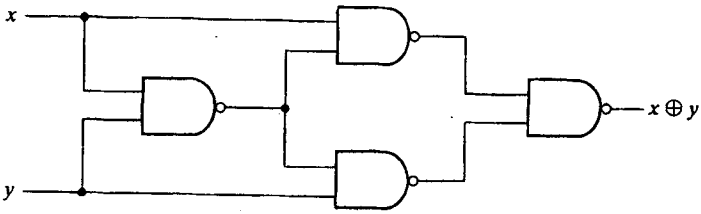
و

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

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



(الف) با گیت‌های AND-OR-NOT



(ب) با گیت‌های NAND

شکل ۳۲-۳. پیاده‌سازی XOR

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

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

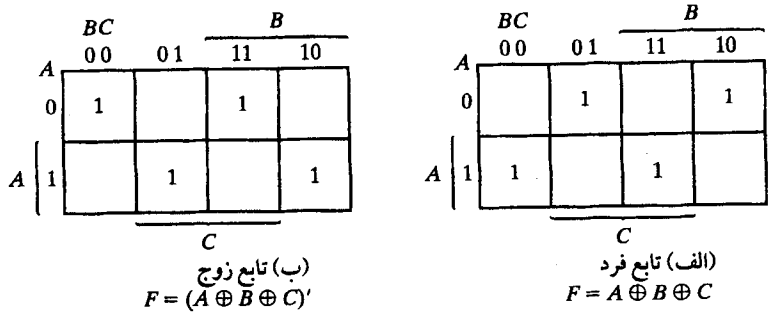
تنها توابع بولی محدودی برحسب OR انحصاری بیان می‌شوند. با این وجود این تابع به کرّات ضمن طراحی سیستم‌های دیجیتال به کار گرفته می‌شود. خصوصاً در عملیات حسابی و خطایابی و تصحیح خطا بسیار مفید است.

تابع فرد

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

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

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



شکل ۳۳-۳. نقشه تابع XOR سه متغیر

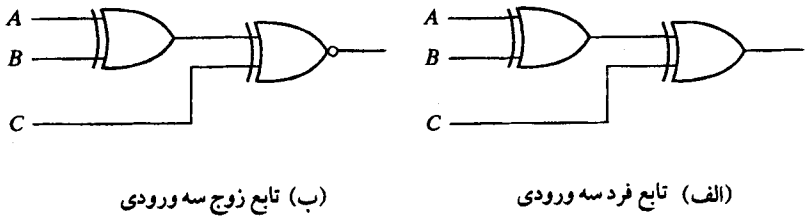
یک متغیر 1 باشد و یا هر سه متغیر برابر 1 باشند. برخلاف حالت دو متغیره، که فقط یک متغیر باید برابر 1 می‌بود، در حالت سه یا چند متغیر، نیاز این است که تعداد فردی از متغیرها برابر 1 باشند. در نتیجه عمل XOR چند متغیره را تابع فرد می‌خوانند.

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

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

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

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



شکل ۳۴-۳. نمودار منطقی توابع فرد و زوج