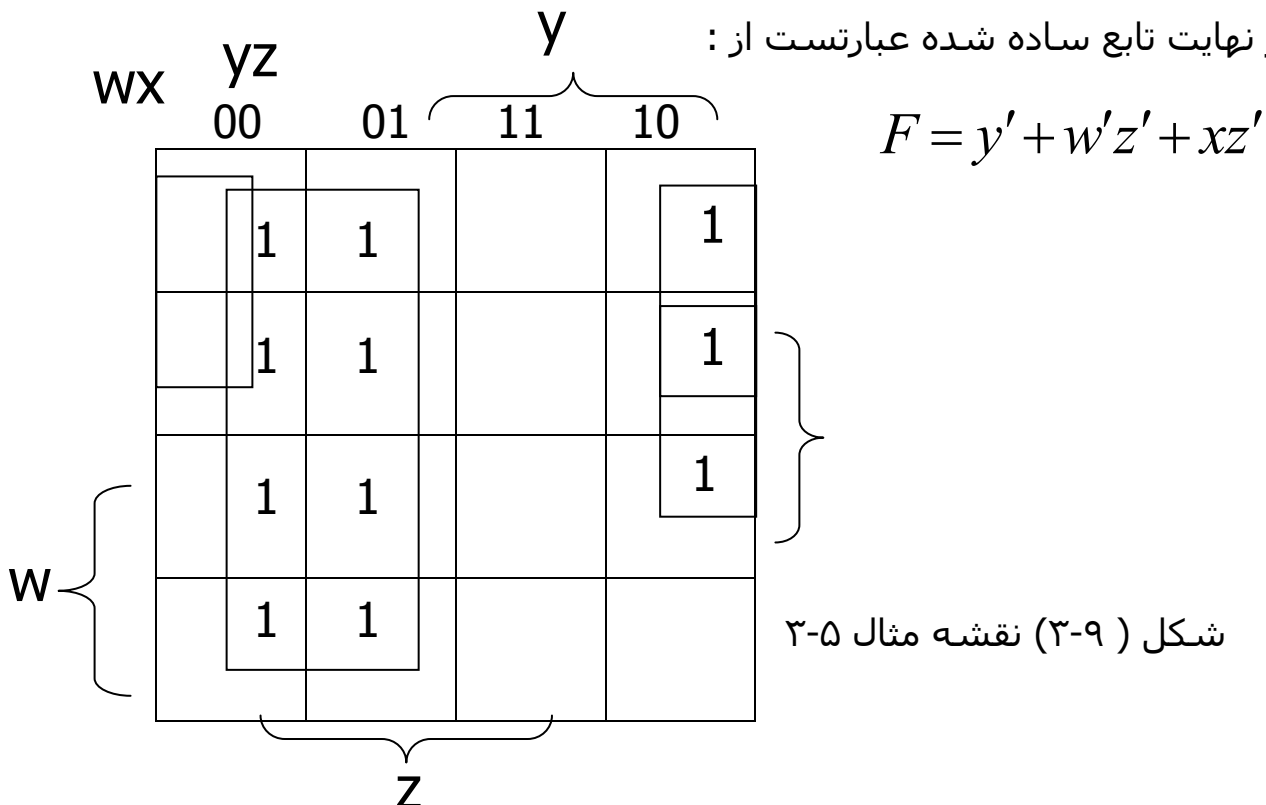


مثال ۳-۵ : تابع بول زیر را ساده کنید :

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

چون که تابع چهار متغیره دارد یک نقشه چهار متغیره باید بکار رود . هشت مینترم سمت چپ با هم ترکیب شده تا عبارت تک متغیره  $y'$  نتیجه شود . سه ۱ باقیمانده در سمت راست میتوانند با هم ترکیب شوند تا عبارت ساده تری حاصل گردد : آنها می بایست بصورت و یا چهار مربع همجوار ترکیب شوند . نتیجه افزایش تعداد مربعهای همجوار ترکیب شده و عبارت  $w'z'$  را تولید میکنند . یادآوری می شود که استفاده از یک مربع بیش از یک بار مجاز است . حال یک مربع دارای ۱ در سطر دوم و ستون چهارم باقی می ماند ، ( مربع ۱۱۱۰ ) . بجای اینکه تنها یابین مربع را در نظر بگیریم ( که عبارت با چهار متغیر را تولید می کند ) آن را با مربع هایی که قبلاً برای تشکیل ناحیه ای با چهار مربع همجوار بکار رفته بود ترکیب می کنیم . حاصل ترکیب این مربعها که شامل دو سطر وسطی و دو ستون آخر می باشند عبارت  $xz'$  خواهد

بود . در نهایت تابع ساده شده عبارتست از :



مثال ۶-۳: تابع بولی زیر را ساده کنید .

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

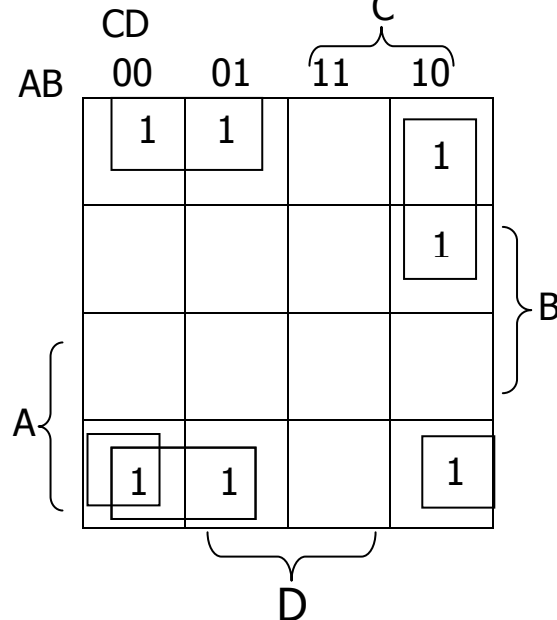
ناحیه ای از نقشه که بوسیله این تابع پوشیده می شود شامل مربعهایی است که در شکل (۱۰-۳) با عدد ۱ پر شده اند . این تابع چهار متغیر دارد و همانطور که ملاحظه شد شامل سه جمله هر یک با سه متغیر و یک جمله با چهار متغیر است . جملات سه متغیره در نقشه با دو مربع نشان داده شده اند . مثلاً  $A'B'C'$  بوسیله مربعهای 0001 و 0000 مشخص شده است . تابع را می توان بوسیله ترکیب ۱ های چهار گوشه نقشه ساده کرد که عبارت  $B'D'$  را بدست آورد . این عمل مجاز است چون وقتی که نقشه را در سطحی فرض کنیم که لبه های چپ و راست و لبه های بالا و پایین آن بهم متصلند ، این چهار مربع همجواریند . دوتا ۱ در سمت چپ از ردیف اول با دو تا ۱ از سمت آخر ترکیب شده و عبارت  $B'C'$  را بدست آورد . ۱ های باقیمانده را می توان به صورت دو مربع ترکیب کرد و عبارت  $A'CD'$  را بدست آورد . تابع ساده شده عبارتست از :

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

### انتخاب های نخستین

هنگام انتخاب مربع های مجاور در نقشه ما باید مطمئن باشیم که تمام مینترم های تابع ضمن ترکیب پوشش داده شده اند . همچنین لازم است که تعداد مینترم ها در عبارت حداقل شده و از جملات مانده ای که مینترم هایشن قبلاً بوسیله سایر جملات پوشش یافته نیز پرهیز گردد . گاهی اوقات هم دو یا سه عبارت وجود دارند که بر عمل ساده سازی صحه می گذرند . روش ترکیب مربع ها در نقشه ممکن است سیستماتیک تر شود ، بشرطی که ما مفهوم جملاتی مانند نخستین انتخاب اصلی

را بدانیم . یک نخستین انتخاب جمله حاصلضربی است که از ترکیب حداکثر ممکن از مربع ها همجوار در نقشه بدست آید . اگر مینترمی در یک مربع بوسیله فقط یک نخستین انتخاب پوشش یابد ، آن نخستین انتخاب را اصلی گوئیم .



نخستین انتخاب یک تابع از یک نقشه با ترکیب حداکثر تعداد ممکن مربع ها بدست می آید . این بدان معنی است که یک ۱ تنها در یک نقشه اگر با هیچ ۱ دیگری مجاور نیست یک نخستین انتخاب را بدست میدهد . دو ۱ مجاور هم یک نخستین انتخاب را تشکیل می دهند بشرطی که در یک گروه هشتتایی مجاور نباشند والی آخر . نخستین انتخابهای اصلی با نظاره بر هر مربع که با ۱ علامت زده شده و چک نمودن تعداد نخستین انتخابهایی که آنرا می پوشاند یافت می شود . یک نخستین انتخاب ، اصلی است اگر که تنها نخستین انتخابی باشد که مینترم را پوشش می دهد . تابع بول چهار متغیره زیرا را ملاحظه کنید .

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

مینترم های تابع با ۱ ها در نقشه شکل (۱۱-۳) علامت زده شده اند . قسمت (الف) از شکل ، دو نخستین انتخاب اصلی را نشان میدهد . چون  $m_0$  تنها در یک گروه مربع

چهار تایی می تواند باشد پس یک جمله اصلی وجود دارد . این چهار مربع  $B'D'$  را تعریف می نمایند . بطور مشابه تنها برای ترکیب  $m_5$  با چهار مربع مجاورش تنها یک راه وجود دارد و این دومین جمله  $BD$  را خواهد داد . دو نخستین انتخاب اصلی هشت مینترم را پوشش می دهند . سه مینترم باقیمانده  $m_3$  ،  $m_9$  و  $m_{11}$  در زیر بررسی میشوند .

شکل (۳-۱۱ ب) تمام راههای ممکن که سه مینترم می تواند با نخستین انتخاب ها پوشش یابد را نشان می دهد . مینترم  $m_3$  می تواند بوسیله نخستین انتخاب  $CD$  یا  $B'C'$  پوشش یابد . مینترم  $m_9$  بوسیله  $AD$  یا  $AD'$  پوشش می یابد. مینترم  $m_{11}$  با هر یک از چهار نخستین انتخاب قابل پوشش است .

عبارت ساده شده از جمع منطقی دو نخستین انتخاب اصلی و هر دو نخستین انتخابی که مینترم های  $m_3, m_9, m_{11}$  را پوشش دهد ، حاصل می گردد . چهار روش برای بیان تابع با چهار جمله ضرب که هر یک دارای دو متغیرند وجود دارد :

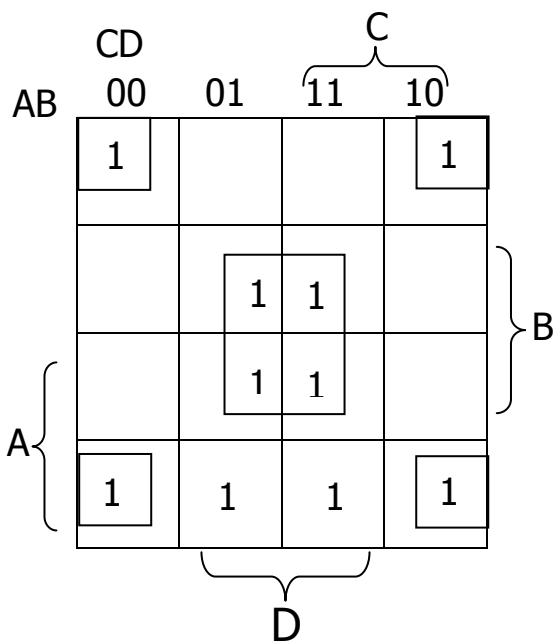
$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

مثال فوق نشان داد که شناخت نخستین انتخاب ها در نقشه به یافتن صور مختلف تابع کمک موثری مینمایند .

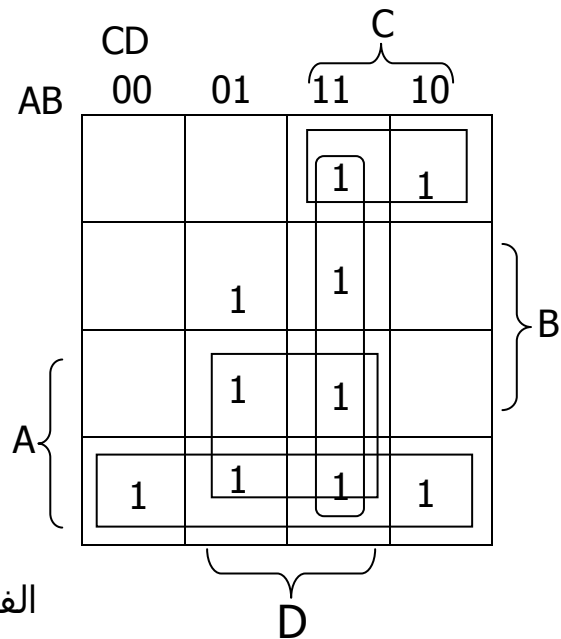
روش یافتن عبارت ساده شده از نقشه نیاز دارد که ابتدا تمام نخستین انتخاب های اصلی را معین کنیم . عبارت ساده شده از جمع منطقی تمام نخستین انتخاب های اصلی را بعلاوه سایر نخستین می آید . در نتیجه ممکن است بیش از یک راه برای

ترکیبات مربعات وجود داشته باشد که هر ترکیب خود عبارت ساده شده ای را تولید

نماید .



الف) نخستین انتخاب های اصلی  $B'C$  و  $B'D'$



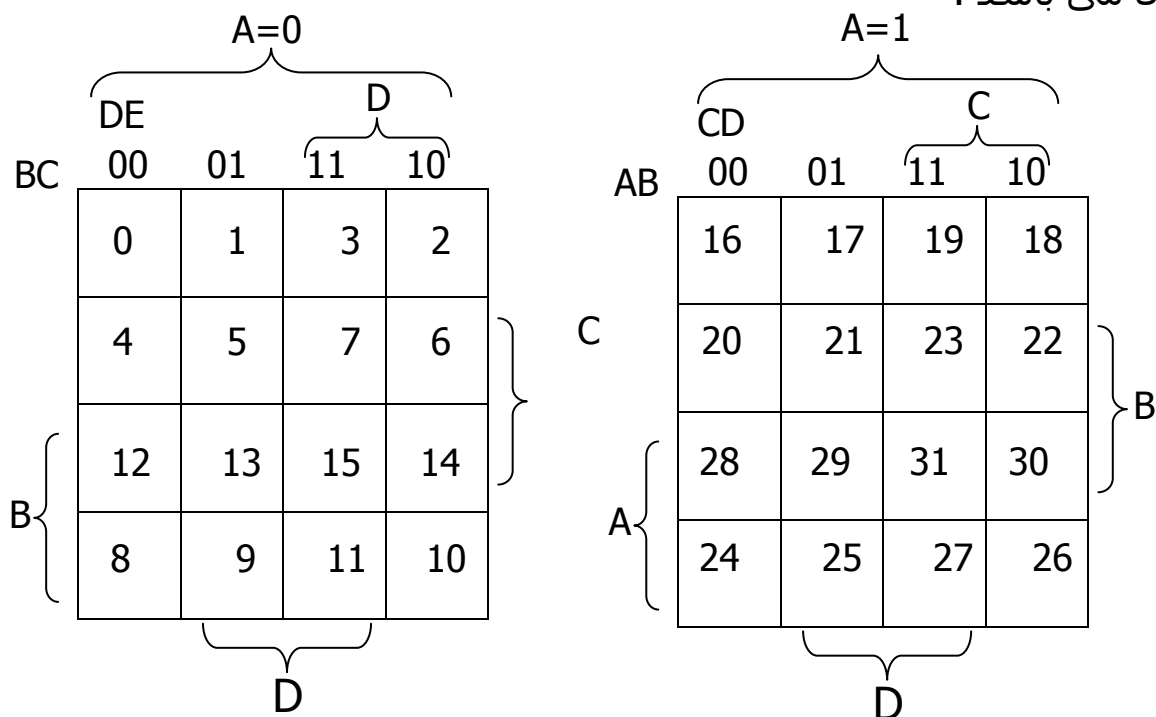
ب) نخستین انتخاب های  $B'D'$  ،  $CD$  ،  $AD$  و  $AB$

## ۲-۲ نقشه پنج متغیره

کار با نقشه هایی که بیش از چهار متغیر دارند ساده نیست . یک نقشه پنج متغیره ۳۲ مربع و یک نقشه شش متغیره ۶۴ مربع دارد . وقتی که تعداد متغیرها زیاد شود تعداد مربعه ها هم بطور بی رویه ای افزایش می یابد و یافتن مربعات همجوار بیشتر به شکل هندسی نقشه وابسته می گردد .

نقشه پنج تغیره در شکل (۲-۱۲) نشان داده شده است . این شکل شامل دو نقشه چهار متغیره با متغیرهای  $E, D, C, B, A$  می باشد . متغیر  $A$  دو نقشه را ، همانطور که در بالای جداول دیده می شود ، از یکدیگر تفکیک می نماید . نقشه چهار متغیره سمت چپ شانزده مربعی را که در آنها  $A=0$  است نشان می دهد ، و نقشه چهار متغیره دیگر مربع هایی که در آنها  $A=1$  است را در بر دارد . مینترم های ۰ تا ۱۵ متعلق به  $A=0$  و مینترم ۱۶ الی ۳۱ به  $A=1$  وابسته اند . هر نقشه چهار متغیره وقتی که

جداگانه در نظر گرفته ، همجواری تعریف شده قبلی خود را حفظ می کند . بعلاوه هر مربع از نقشه  $A=0$  با مربع متناظرش در  $A=1$  همجوار است . مثلاً مینترم ۴ با مینترم ۲۰ مجاور است و مینترم ۱۵ نیز با ۳۱ همجوار می باشد . بهترین راه تشخیص این قانون جدید برای مربع های همجوار اینست که تصور کنیم که دو نیم نقشه با قرار گرفتن روی هم تبدیل به یکی گردیده اند . هر دو مربعی که روی دیگری بیفتد با آن مجاور است . با دنبال کردن روشی که برای نقشه پنج متغیره بکار رفت ، می توان نقشه شش متغیره ای با ۴ نقشه متغیره ساخت تا ۶۴ مربع مورد نیاز بدست آید . نقشه هایی با شش یا تعداد بیشتری متغیر نیاز به تعداد بی شماری مربع داشته و غیر کاربردی هستند . روش دیگر بکار بردن برنامه های کامپیوتر خاص جهت سازی توابع بول می باشد .



شکل (۱۲-۳) نقشه شش متغیره

با بررسی و در نظر گرفتن تعریف جدید همجواری مربع های ، میتوان نشان داد که  $2^k$  مربع همجواری با ازای  $k = 0, 1, 2, \dots, n$  در یک نقشه  $n$  متغیره ، ناحیه ای را مشخص می کنند که نمایش دهنده یک جمله  $n-k$  متغیره است . برای تکمیل مفهوم

عبارت بالا ، می بایست  $n$  از  $k$  بزرگتر باشد . وقتی  $n=k$  باشد تمام سطوح نقشه با هم ترکیب می شوند و حاصل ترکیب ، تابع ثابت ۱ است .

جدول (۳-۱) ارتباط بین تعداد مربعهای همجوار و تعداد متغیرها در یک جمله را نشان می دهد . مثلاً هست مربع همجوار در نقشه پنج متغیره مساحتی را ترکیب می کنند تا یک جمله با و متغیر بدست آید .

	$2^k$	$n=2$	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$
0	1	2	3	4	5	6	7
1	2	1	2	3	4	5	6
2	4	0	1	2	3	4	5
3	8		0	1	2	3	4
4	16			0	1	2	3
5	32				0	1	2
6	64					0	1

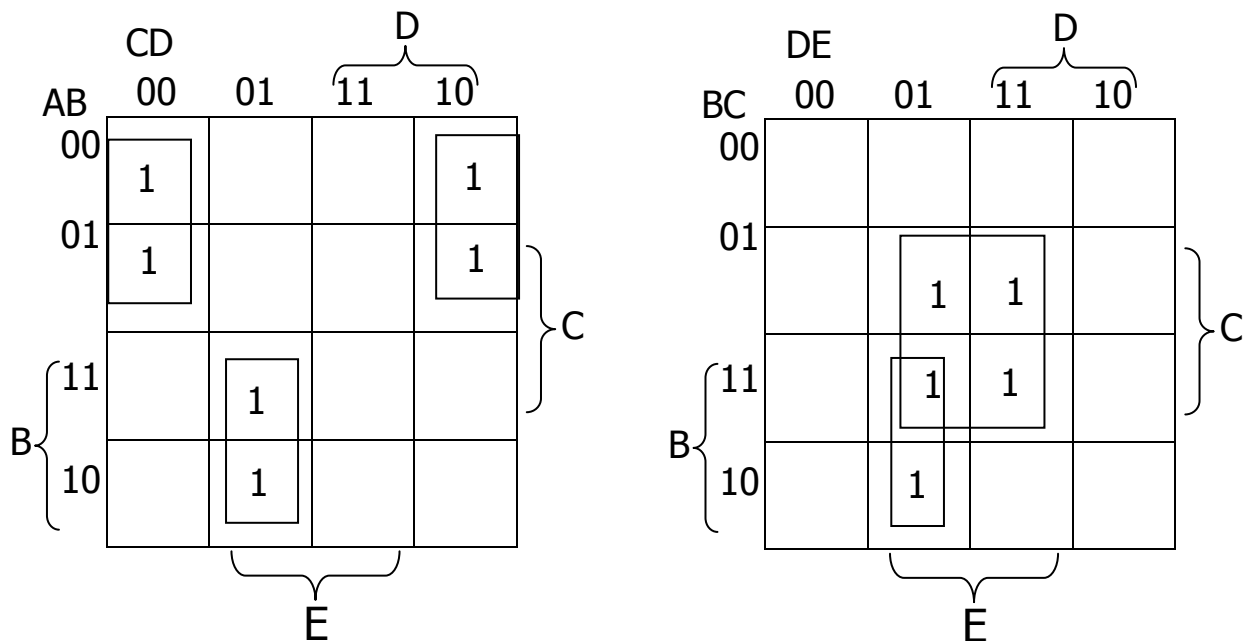
مثال ۳-۷: تابع بول زیر را ساده کنید :

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

نقشه پنج متغیره برای این تابع در شکل (۳-۱۲) دیده می شود . به بخشی از نقشه که در آن  $A=0$  است شش مینترم تابع ، از ۰ تا ۱۵ ، تعلق دارد . پنج مینترم دیگر به بخش  $A=1$  متعلق است . چهار مربع مجاور در نقشه  $A=0$  با هر ترکیب شده تا جمله سه متغیره  $A'B'E'$  را بدهد . دقت کنید که لازم است  $A'$  را در جمله منظور کنیم زیرا تمام مربع های متعلق به  $A=0$  می باشند . دو مربع در ستون ۰۱ و

دو سطر آخر در هر دو قسمت نقشه مشترکند . بنابراین چهار مربع مجار را تشکیل داده و جمله سه متغیره  $BD'E$  را می سازند . متغیر  $A$  در اینجا آورده نشده زیرا مربع های مجاور متعلق به هر دو  $A=0$  و  $A=1$  می باشند . جمله  $ACE$  از چهارمربع همجواری حاصل شده که در نقشه  $A=1$  قرار دارند . تابع ساده شده جمع منطقی سه جمله است :

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



شکل (۳-۱۲) نقشه برای مثال ۲-۷  $F = A'B'E' + BD'E + ACE$

### ۲-۵ ساده سازی با استفاده از ضرب حاصلجمع ها

در تمام مثالهای قبل از توابع بول حاصل از نقشه ها به فرم جمع حاصلضرب بیان شده بودند . با یک تغییر کوچک می توان فرم ضرب حاصلجمع ها را نیز برای آنها بدست آورد .



یافتن روالی برای بدست آوردن یک تابع می نیمم بر حسب ضرب حاصلجمع ها نیازمند دانستن خواص اساسی توابع بول است . وجود ۱ ها در مربعهای نقشه ، بیانگر مینترم های تابع است . مینترم هایی که در تابع نیستند بیانگر مکمل تابع می باشند . با توجه به این مطلب مشاهده می کنیم که مکمل یک تابع بوسیله مربعهایی که فاقد ۱ هستند نشان داده می شود . اگر در مربعهای خالی ۰ بگذاریم و آنها را با روش مربعهای همجوار ترکیب کنیم عبارت ساده شده ای از مکمل تابع ، یعنی  $F'$  تابع  $F$  را به ما بر می گرداند . بعلت عمومیت تئوری دمورگان تابع حاصل خودبخود بصورت ضرب حاصلجمع ها بدست می آید . بهترین روش برای تشریح این مطلب ارائه یک مثال است .

مثال ۳-۸: تابع بول زیر را ساده کنید :

الف) بر حسب جمع حاصلضرب ها ( ب ) بر حسب ضرب حاصلجمع ها

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

۱ ها موجود در نقشه شکل (۳-۱۴) نشان دهنده تمام مینترم های تابع است . مربعهای دارای ۰ بیانگر مینترم هایی هستند که در تابع  $F$  نیستند ، بنابراین بر مکمل  $F$  دلالت می کنند . از ترکیب مربعهای ۱ ، تابع ساده شده به فرم جمع حاصلضرب ها بدست می آید.

AB	CD			
	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

شکل (۳-۱۴) نقشه مثال ۳-۸

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

اگر مربعهای دارای ۰ همانطوری که در شکل نشان داده شده است ترکیب شوند تابع ساده شده زیر را خواهیم داشت :

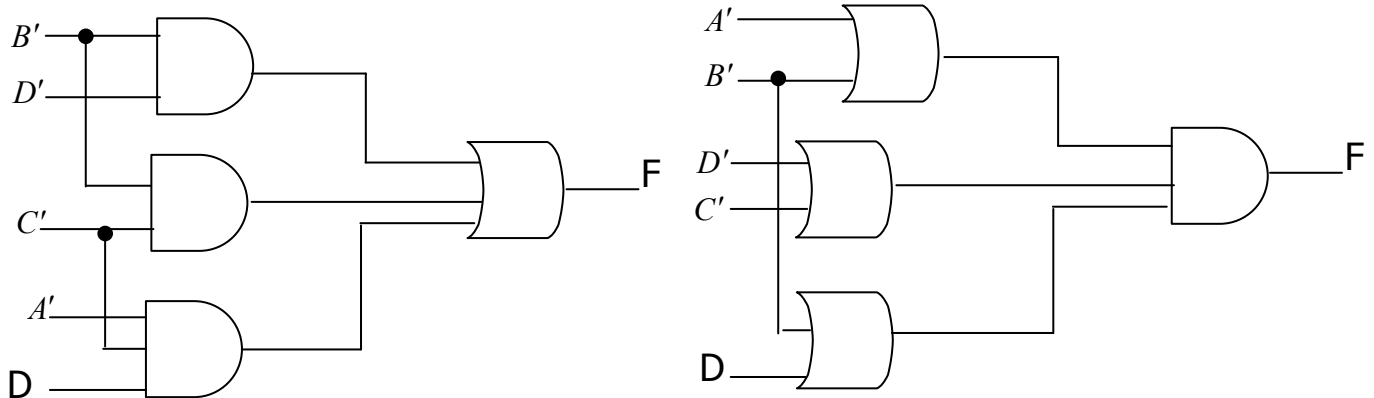
$$F' = AB + CD + BD'$$

با بکار بردن قضیه مورگان ( با استفاده از دوگانگی و مکمل کردن هر متغیر طبق آنچه در بخش ۲-۴ شرح داده شده ) تابع ساده شده را به فرم ضرب حاصلجمع تنها بدست می آوریم .

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

پیاده سازی عبارت ساده شده حاصل از مثال ۳-۸ در شکل (۳-۱۵) نشان داده شده است . عبارت جمع حاصلضرب ها در شکل (۳-۱۵) ، با مجموعه ای از گیت های AND که هر گیت برای یک جمله AND می باشد پیاده سازی شده است و خروجی گیت های AND نیز به ورودی گیت OR متصل گردیده است . همان عبارت بصورت ضرب حاصلجمع ها در شکل (ب) با تعدادی گیت OR که هر کدام برای یک جمله OR می باشد پیاده سازی شده و خروجی آنها به یک AND منتهی گردیده است . در هر حالت فرض شده که مکمل متغیرهای ورودی مستقیماً در دسترس است ، بنابراین به معکوس کننده ها نیازی نیست . الگوهای ایجاد شده در شکل (۳-۱۵) یک سری روشهای کلی هستند که بوسیله آنها هر تابع بول استاندارد ، قابل پیاده سازی است . در جمع حاصلضرب ها ، گیت های AND به یک گیت OR متصل شده و در ضرب حاصلجمع ها گیت های OR به یک گیت AND وصل می شوند . هر یک از دو

بیکره بندی فوق دارای دو طبقه از گیت ها می باشند . به همین دلیل پیاده سازی یک تابع به فرم استاندارد ، پیاده سازی دو طبقه ای نامیده می شود .



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

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

مثال ۸-۳ روالی برای مجاسبه فرم ساده شده یک تابع بر حسب ضرب حاصلجمع ها ، وقتی که تابع ابتدا بر حسب جمع مینترم ها بیان شده باشد را نشان داد . این روال هنگامی که تابع در آغاز بر حسب ضرب ماکسترم ها بیان شود نیز معتبر است . برای مثال به جدول درستی (۳-۲) که تابع F تعریف می کند توجه کنید در جمع مینترم ها این تابع چنین بیان می شود :

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

و در ضرب ماکسترم ها بصورت زیر است :

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

به عبارت دیگر ۱ های تابع نشان دهنده جملات مینترم و ۰ های آن بیانگر جملات ماکسترم هستند . نقشه این تابع در شکل (۳-۱۶) رسم شده است . برای ساده کردن این تابع ابتدا می توان در مربع مربوط به هر جمله مینترم که تابع به ازای آن مقدار ۱ دارد عدد ۱ گذاشت و مربعهای باقیمانده را با ۰ پر می کرد . از طریق دیگر اگر

تابع به فرم ضرب ماکسترم ها داه شده باشد در ابتدا می توان در مربعهایی که تابع مشخص می کند  $\cdot$  گذاشت و مربعهای باقیمانده را با  $1$  پر کرد . هنگامی که  $1$  ها و  $\cdot$  ها در جدول گذاشته شدند ، تابع می تواند باقیمانده را با  $1$  پر کرد . هنگامی که  $1$  ها و  $\cdot$  ها در جدول گذاشته شدند ، تابع می تواند به یکی از فرمهای استاندارد ساده شود . برای جمع حاصلضرب ها  $1$  ها را با ترکیب می کنیم و خواهیم داشت :

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

جدول (۳-۲) جدول درستی

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

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

z

شکل (۳-۱۶) نقشه تابع جدول (۳-۲)

برای ضرب حاصلجمع ها ،  $\cdot$  ها را با هم ترکیب می کنیم تا مکمل تابع ساده شده بصورت زیر بدست آید :

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

این رابطه نشان می دهد که تابع XOR ، مکمل تابع هم ارزی می باشد . ( بخش ۶-  
 ۲) با مکمل کردن  $F'$  در حقیقت تابع ساده شده به فرم ضرب حاصلجمع ها خواهیم  
 داشت :

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

برای وارد کردن یک تابع در یک نقشه که بر حسب ضرب حاصلجمع ها بیان شده است  
 می بایست مکمل تابع را بدست آورد و با استفاده از آن ، مربعهای مربوطه را با ۰ پر  
 کرد . برای مثال تابع

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

و سپس مربعهایی که عبارات می نیمم f را نشان می دهند با ۰ مربعهای باقیمانده را  
 با ۱ پر می کنیم .

### ۶-۲ پیاده سازی بوسیله کیت های NAND و NOR

مدارهای دیجیتال اغلب بجای اینکه با کیت های AND و OR ساخته شوند با کیت  
 های NAND و NOR ساخته می شوند . ساختن گیت های NAND و NOR با اجزای  
 الکترونیکی ساده تر بوده و بعنوان گیت های پایه در تمام خانواده های آی سی های  
 منطقی بکار میروند . به دلیل مزیت گیتهای NAND NAND و NOR در طراحی مدارهای  
 دیجیتال ، اصول و قواعدی برای تبدیل توابع بول بیان شده بر حسب AND و OR و  
 NOT به دیاگرام منطقی NAND و NOR معادل بوجود آمده است . در این بخش ، روال  
 پیاده سازی دو طبقه نشان داده شده است .

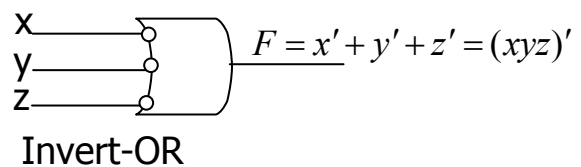
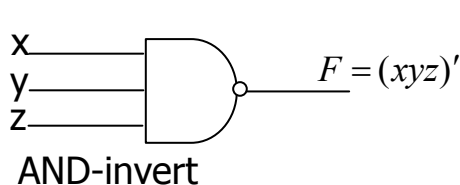
برای سادگی تبدیل به منطق NAND و NOR بهتر است دو سمبل دیگر را برای این  
 گیت ها تعریف کنیم . دو سمبل برای گیت NAND در شکل (۱۷-۳) نشان داده شده

است . سمبل AND-INVERT قبلاً تعریف شده که شامل سمبل AND و بدنبال آن یک دایره کوچک است . بجای آن می توان یک گیت NAND را با یک سمبل OR که وایر کوچکی در تمام ورودی های آن کشیده شده است نشان داد . سمبل AND-INVERT برا گیت NAND با توجه به قضیه دمورگان و در نظر گرفتن این قرار داد که دوایر کوچک مکمل کردن می باشند بدست می آید .

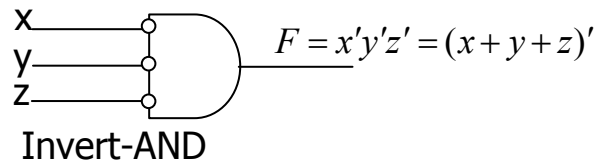
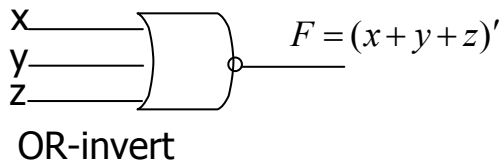
بطور مشابه ، دو سمبل نیز برای گیت NOR یک سمیل قراردادی است و AND-INVERT نیز شکل مناسب دیگری است که قضیه دمورگان و قرار داد دایره های کوچک به معنی مکمل کردن را مورد استفاده قرار می دهد .

یک گیت nand یا nor با یک ورودی مثل معکوس کننده عمل می کند ، نتیجتاً یک گیت معکوس کننده را می توان به یکی از سه حالت مختلف که در شکل (۱۷-۳ پ ) نشان داده شده است نمایش داد . دایره های کوچک در هر سمبل معکوس کننده را می توان بدون تغییر منطق گیت به پایه ورودی آن انتقال داد .

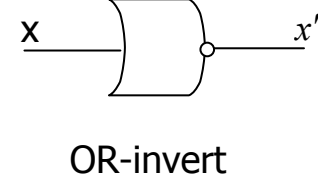
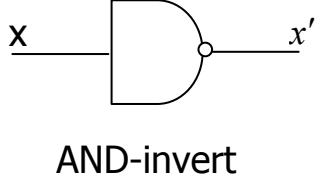
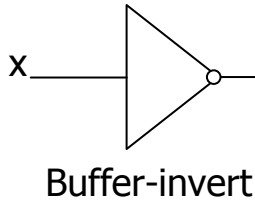
باید خاطر نشان کرد که سمبلهای معادل دیگر برای گیت های NAND و NOR را می توان با جایگزینی مثلث های کوچک بجای دایره های در همه پایانه های ورودی کشید . یک مثلث کوچک نشانه یک منطق منفی است ، بنابراین وجود مثلثهای کوچک روی پایه های ورودی در سمبل گرافیکی یک گیت ، دلالت بر منفی بودن منطقی ورودی های آن دارد . اما به خروى یک گیت ( که دارای مثلث نیست ) یک منطق مثبت منسوب شده است .



(الف) دو سمبل گرافیکی برای گیت NAND



(ب) دو سمبل گرافیکی برای گیت NOR

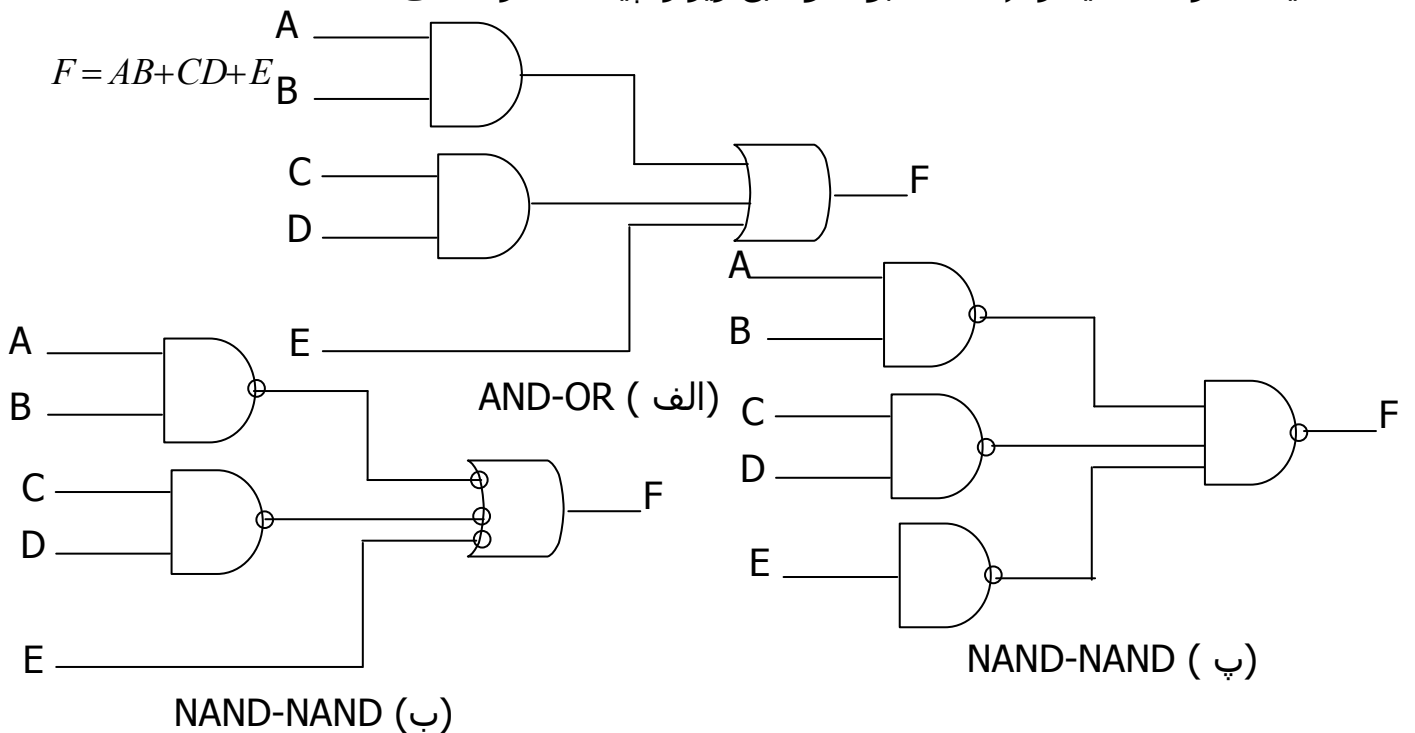


شکل (۳-۱۷) سه سمبل گرافیکی برای معکوس کننده

### پیاده سازی با گیت NAND

پیاده سازی یک تابع بول با گیت های NAND مستلزم این است که تابع به فرم جمع حاصلضرب ها ساده شده باشد. برای درک ارتباط بین عبارت جمع حاصلضرب ها و معادل پیاده شده NAND آن به دیاگرام منطقی کشیده شده در شکل (۳-۱۸) توجه کنید.

هر سه دیاگرام معادل بوده و تابع زیر را پیاده سازی می کنند.



شکل (۳-۱۸) سه راه مختلف پیاده سازی  $F = AB + CD + E$

تابع در شکل ( ۱۸-۳ الف) به فرم جمع حاصلضرب ها با استفاده از گیت های AND و OR پیاده شده است . در شکل (ب) گیت های AND با گیت های NAND و گیت OR نیز بوسیله یک گیت NAND با سمبل AND-INVERT مشخص گردیده جایگزین شده است . متغیر E مکمل شده و به طبقه دوم یعنی گیت INVERT-OR اعمال شده است . بخاطر داشته باشید که دایره کوچک دال بر مکمل سازی است . بنابراین دو دایره کوچک در یک مسیر نشان دهنده دوبار مکمل سازی بوده و می تواند حذف شوند . مکمل E از میان یک دایره کوچک که متغیر را مجدداً مکمل می نماید عبور کرده و تا مقدار طبیعی E را تولید می کند . حذف دایره های کوچک در گیت های شکل (۱۸-۳ ب) ف مدار (الف) را تولید می کند . بنابراین هر دو دیاگرام یک تابع را پیاده سازی کرده و معادل هستند .

در شکل ( ۱۸-۳ پ ) گیت NAND طبقه خروجی دوباره با سمبل قراردادی کشیده شده است . گیت NAND با یک ورودی ، متغیر E را مکمل می کند . این معکوس کننده را می توان حذف کرد و مستقیماً برای گیت NAND سطح دوم ، E را بکار برد . دیاگرام شکل ( پ ) معادل با (ب) است که این نیز به نوبه خود معادل ( الف) می باشد . به شباهت بیت دیاگرام های (الف) و ( پ ) توجه کنید گیت های AND و OR به گیت های NAND تغیر یافته اند ، اما یک گیت NAND در ورودی E اضافه شده است به هر حال در رسم دیاگرام منطقی متشکل از NAND ، هر دو مدار نشان داده شده در ( ب ) یا (پ) قابل قبول هستند . با این وجد دیاگرام (ب) دارای ارتباط مستقیم بیشتری با عبارت بول پیاده شده است .



صحت پیاده سازی بوسیله گیت های NAND در شکل ( ۳-۱۸ پ ) می تواند بصورت جبری بازنگری شود . تابع NAND پیاده شده می توانند بسادگی با استفاده از قانون دمورگان به فرم جمع حاصلضرب تبدیل گردد .

$$F = [(AB)' \cdot (CD)' \cdot E']' = AB + CD + E$$

از تبدیل گام به گام در شکل ( ۳-۱۸ ) نتیجه می گیریم که یک تابع بول می تواند به دو طبقه از گیتها NAND پیاده سازی شود . قاعده بدست آوردن دیاگرام منطقی NAND از یک تابع بول شده بشرح زیر است :

۱- تابع را ساده کرده و آن را به فرم جمع حاصلضرب ها بنویسید .

۲- برای هر جمله ضرب موجود درتابع که حداقل دارای دو متغیر است یک گیت NAND بکشید . ورودی های هر گیت NAND متغیرهای آن جمله هستند . این مجموعه گیت های طبقه اول را تشکیل می دهند .

۳- در طبقه دوم ، یک گیت NAND با ورودیهایی که ازخروجی های طبقه اول می آیند ، بکشد . ( از سمبل گرافیکی AND-invert یا OR-invert استفاده کنید ) .

۴- یک جمله تک متغیری در طبقه او نیازمند یک معکوس کننده است . همچنین میتوان مکمل آن را بعنوان ورودی برای گیت NAND طبقه دوم بکار برد .

قبل از بکار گیری این روش در یک مثال خاص باید خاطر نشان کرد که راه دومی نیز برای پیاده سازی توابع بول با گیت های NAND موجود است . بخاطر بیاورید که اگر در یک نقشه ۰ ها را ترکیب کنیم عبارت ساده شده مکمل آنها تابع را به فرم جمع حاصلضرب ها بدست می آوریم . مکمل تابع رامی توان با دو طبقه ازگیت های NAND و با استفاده از قواعدی که در بالا بیان شده پیاده کرد . اگر خروجی به فرم طبیعی مورد نظر باشد لزوم است تا یک گیت NAND با یک ورودی یا گیت معکوس کننده برای تولید

فرم واقعی تابع خروجی منظور شود . گاهی ممکن است طراح بخواهد مکمل یکتابع را تولید کند که در این صورت متددوم ارجح می باشد .

مثال ۳-۹: تابع زیر را با گیت های NAND پیاده کنید :

$$F(x, y, z) = \sum(0, 6)$$

اولین قدم ساده کردن تابع به فرم جمه حاصلضرب ها است که طبق نقشه نشان داده شده در شکل ( ۳-۱۹ الف ) انجام می شود . در نقطه فقط دو ۱ موجود است که نمی توانند با هم ترکیب شوند . بنابراین تابع ساده شده به فرم جمع حاصلضرب ها عبارتست از :

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

پیاده سازی با دو طبق گیت NAND در شکل (۳-۱۹ ب) نشان داده شده است . در قدم بعدی سعی می کنیم مکمل تابعی را به فرم جمع حاصلضرب ها ساده کنیم که با ترکیب ۰ ها در نقشه میسر است .

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

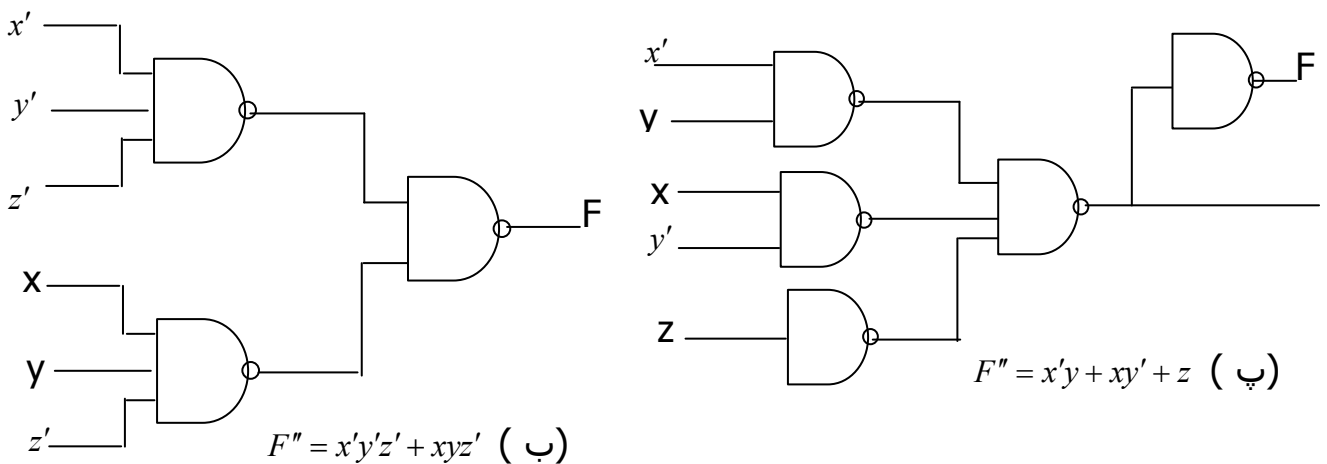
در شکل (۳-۱۹ پ) دو طبقه گیت NAND برای تولید  $F'$  نشان داده شده است . اگر در خروجی به  $F$  نیاز باشد یک گیت NAND با یک ورودی نیز اضافه شود تا تابع را معکوس کند که در این صورت پیاده سازی در سه طبقه خواهد بود . اگر متغیرها فقط به یک فرم قابل دسترسی باشند می بایست در ورودی معکوس کننده هایی منظور کنیم که باعث اضافه شدن طبقه دیگری به مدار خواهند شد . گیت NAND با یک ورودی مربوط به متغیر  $Z$  می تواند حذف شود ، بشرط آنکه آن متغیر به  $Z'$  تبدیل گردد .

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

Z

( الف ) ساده سازی نقشه در جمع حاصلضرب

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

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


شکل (۳-۱۹) پیاده سازی تابع مثال ۳-۹ با استفاده از گیت های NAND

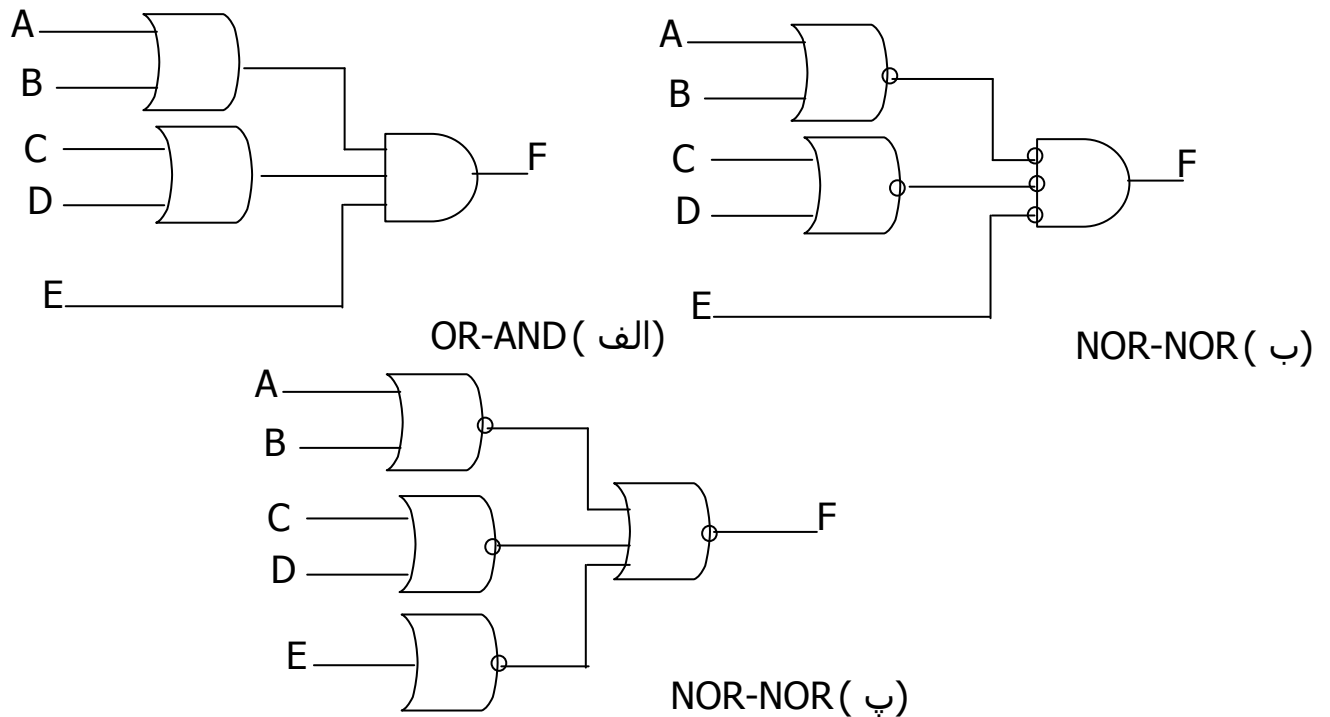
### پیاده سازی بوسیله گیت NOR

تابع NOR دوگان تابع NAND است . به همین دلیل همه روالها و قواعد منطق NAND دوگان روالها وقواعد منطق NAND می باشد .

برای پیاده سازی یک تابع بول با گیت NOR باید به فرم ضرب حاصلجمع ها ساده شود . یک عبارت به صورت ضرب حاصلجمع ها ، مجموعه ای از گیت های OR برای جملات جمع و بدنبال آن یک AND برای تولید ضرب است . تبدیل گام به گام از OR-AND به دیگرام NOR-NOR در شکل (۳-۲۰) نشان داده شده است . این مراحل شبیه مراحل تبدیل به NAND است با این تفاوت که در اینجا از عبارت ضرب حاصلجمع ها استفاده می کنیم .

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

قانون بدست آوردن دیاگرام منطقی NOR از یک تابع بول می تواند از تبدیل مشتق شود که شبیه به قانون سه مرحله ای NAND است ، با این تفاوت که عبارت ساده شده می بایست به فرم ضرب حاصلجمع ها باشد و جملات طبقه اول گیت ها NOR جملات جمع باشند . یک جمله تک متغیری به یک NOR با یک ورودی با یک گیت معکوس کننده نیاز دارد و یا اینکه می توان آن را مکمل کرده و مستقیماً در طبقه دوم گیت NOR بکار برد.



شکل (۳-۲۰) سه روش پیاده سازی تابع  $F = (A+B)(C+D)E$

روش دوم پیاده سازی یک تابع بوسیله گیت های NOR ، استفاده از مکمل تابع بر حسب ضرب حاصلجمع ها است . این روش پیاده سازی دو طبقه را برای  $F'$  و پیاده سازی سه طبقه را در صورت نیاز برای  $F$  نتیجه می دهد .

برای بدست آوردن ضرب حاصلجمع های ساده شده از یک جدول ، لازم است تا ۰ ها را در جدول ترکیب کرده و سپس تابع حاصل را مکمل کنیم به منظور بدست آوردن

عبارت ساده شده ضرب حاصلجمع ها برای مکمل تابع ، می بایست ۱ ها را در جدول ترکیب کرده و سپس تابع را مکمل نماییم . مثال زیر روال پیاده سازی بوسیله NOR را نشان می دهد .

مثال ۳-۱۰: تابع مثال ۳-۹ را با گیت های NOR پیاده کنید .

نقشه این تابع در شکل (۳-۱۹ الف) کشیده شده است . در این نقشه ابتدا ۰ ها را با هم ترکیب می کنیم تا عبارت زیر بدست آید :

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

این عبارت ، مکمل تابع بر حسب جمع حاصلضرب ها است . سپس  $F'$  را مکمل می نمایم تا تابعی که بر حسب ضرب حاصلجمع ها است و برای پیاده سازی بوسیله NOR لازم است بدست آورید .

پیاده سازی دو طبقه با گیت های NOR در شکل (۳-۲۱ الف) نشان داده شده است . جمله ای که فقط دارای حرف  $z'$  است نیازمند به گیت NOR با یک ورودی و یا یک گیت معکوس کننده می باشد . این گیت می تواند حذف شده و ورودی  $z$  مستقیماً به ورودی گیت NOR طبقه دوم متصل گردد .

با استفاده از مکمل تابع بر حسب ضرب حاصلجمع ها پیاده سازی به روش دیگری نیز مقدور است . در این حالت ابتدا ۱ ها را در جدول ترکیب و عبارت زیر را بدست می آوریم .

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

این عبارت فرم ساده تابع بر حسب جمع حاصلضرب ها می باشد. سپس تابعی را مکمل می کنیم تا مکمل آن را بر حسب ضرب حاصلجمع ها به صورتی بدست آوریم که برای پیاده سازی توسط NOR لازم است :

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

در شکل (۲۱-۲) پیاده سازی دو طبقه برای  $F'$  نشان داده شده است . اگر خروجی  $F$  مورد نظر باشد ، میتوان با استفاده از یک معکوس کننده آن رادر طبقه سوم تولید کرد .

در جدول (۳-۲) روشهای پیاده سازی NOR یا NAND خلاصه شده است . چیزی که نباید فراموش شود این است که همیشه هدف از ساده کردن یک تابع کاهش تعداد گیت های آن در زمان پیاده سازی است . فرمهای استاندارد از روشهای ساده سازی مستقیماً کاربرد دارند و هنگامی که هدف ، بکارگیری NAND یا NOR باشد بسیار مفید هستند.

جدول (۳-۲) قوانین پیاده سازی NOR و NAND

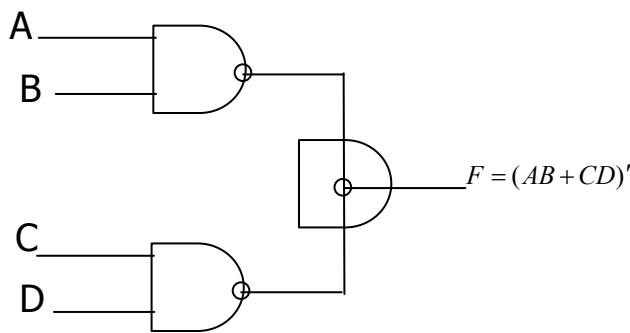
حالت	تابع جهت ساده سازی	فرم استاندارد جهت استفاده	نحوه بدست آوردن	پیاده سازی با	تعداد طبقات تا
(الف)	F	جمع حاصلضربها	۱ها را در نقشه ترکیب کنید	NAND	۲
(ب)	$F'$	جمع حاصلضربها	۰ها را در نقشه ترکیب کنید	NAND	۲
(پ)	F	ضرب حاصلضربها	$F'$ را در (ب) مکمل کنید	NOR	۲
(ت)	$F'$	ضرب حاصلضربها	F را در (الف) مکمل کنید	NOR	۲

### ۲-۷- سایر پیاده سازی های دو طبقه

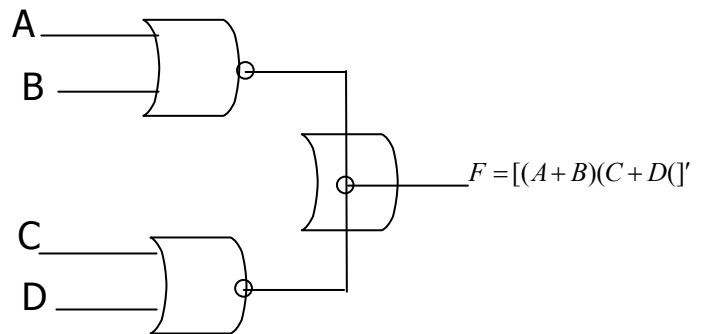
گیت های موجود در مدارهای مجتمع اغلب از نوع NAND و NOR هستند . به همین دلیل عملاً پیاده سازی منطقی با NOR و NAND بسیار اهمیت دارد . در بعضی از گیت های NOR یا NAND نه همه آنها این امکان وجود دارد که با اتصال یک سیم بین

خروجی های دو گیت ، یک تابع منطقی مشخص تولید کرد که این منطق ، منطق اتصالی نامیده می شود . مثلاً وقتی خروجی گیت های NAND - از نوع کلکتور باز TTL - به هم متصل شوند عمل منطق AND اتصالی را انجام می دهند . منطق AND اتصالی که بوسیله دو گیت NAND انجام شده در شکل (۲-۲۲ الف) ترسیم شده است . برای تفکیک گیت AND اتصالی از گیت های قراردادی ، آن را بخطهایی که تا مرکز آن امتداد دارد مشخص می کنیم . گیت AND اتصالی یک گیت فیزیکی نیست بلکه فقط سمبلی است برای توصیف یگ تابع که از اتصال سیمها بدست می آید . تابع منطقی که بوسیله مدار (۲-۲۲ الف) پیاده سازی شده عبارتست از :

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



الف) AND اتصالی در گیت های TTL NAND کلکتور باز (AND-OR-INVERT)



ب) OR اتصالی در گیت های ECL (AND-OR-INVERT)

شکل (۲-۲۲) منطق اتصالی

که تابع AND-OR-INVERT نامیده می شود .

بطور مشابه خروجی NOR در گیت های ECL رامی توان به هم گره زد و تابع OR اتصالی را ایجاد نمود . تابع منطقی که بوسیله مدار (۲-۲۲ ب) پیاده سازی شده عبارتست از :

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

که تابع OR-AND-INVERT نامیده میشود .

یک گیت منطقی اتصالی بعنوانگیت طبقه دوم تلقی نمی گردد . چون فقط از اتصال سیمها بوجود آمده است . ولی ، به هنگام بحث مدارهای شکل (۲-۲۲) را به فرم مدارهای دو طبقه می نگریم .

اولین طبقه ، شامل گیت های NAND ( یا NOR ) و دومین طبقه فقط دارای یک گیت AND ( یا OR ) است .

### ترکیبات مفید گیت ها

از نقطه نظر تئوری دانستن ترکیبات ممکن گیت ها در دو طبقه آموزنده است . در اینجا چهار نوع گیت را بررسی می کنیم : NOR-NAND-OR-AND . اگر به هر طبقه یک نوع گیت را نسبت دهیم در می یابیم که شانزده ترکیب ممکن به فرم دو طبقه وجود دارد ، ( می توان گیت های یکسانی را درطبقات اول و دوم بکار برد مانند پیاده سازی NAND-NAND ) هشت ترکیب از ترکیبات فوق زائد نامیده می شوند چون در حقیقت یک همل ساده منطقی را انجام می دهند . این نکته در مواردی که طبقه های اول و دوم هر دو دارای گیت های AND هستند بخوبی دیده می شود .

خروجی مدار ، صرفاً تابع AND روی همه متغیرهای ورودی است . هشت فرم مفید باقیمانده هر کدام پیاده سازی را بصورت جمع حاصلضرب ها و یا ضرب حاصلجمع ها تولید می کنند . این هشت فرم مفید عبارتند از :

AND-OR                      OR-AND

NAND-NAND                NOR-NOR

NOR-OR                     NAND-AND

OR-NAND                    AND-NOR



در هر یک از فرمهای فوق اولین گیت ذکر شده تشکیل دهنده طبقه اول در پیاده سازی است و دومین گیت به صورت یک گیت منفرد در طبقه دوم قرار می گیرد . توجه کنید هر دو فرمی که در یک سطر آمده اند دوگان یکدیگرند .

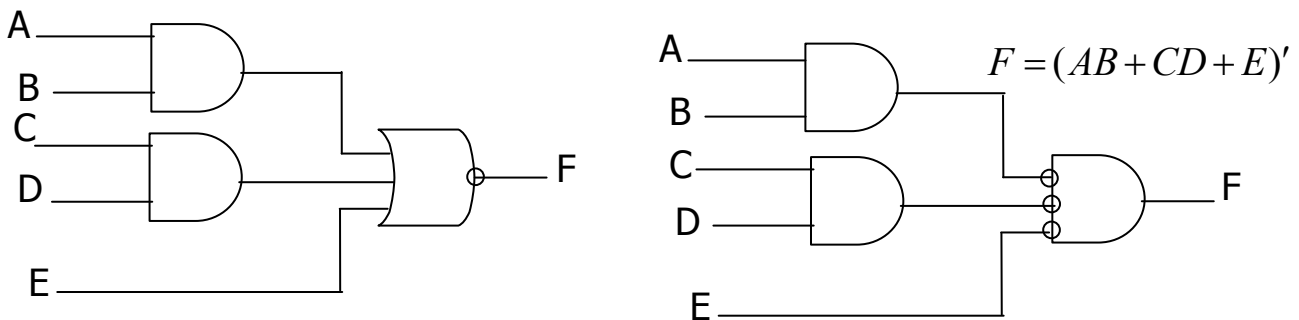
فرمهای OR-AND و AND-OR فرمهای اولیه و طبقه هستند که در بخش ۳-۵ بحث شدند . همچنین فرمهای NAND-NAND و NOR-NOR در بخش ۳-۶ معرفی گردیدند .

چهار فرم باقیمانده نیز در این بخش بررسی می شوند.

### پیاده سازی AND-OR-INVERT

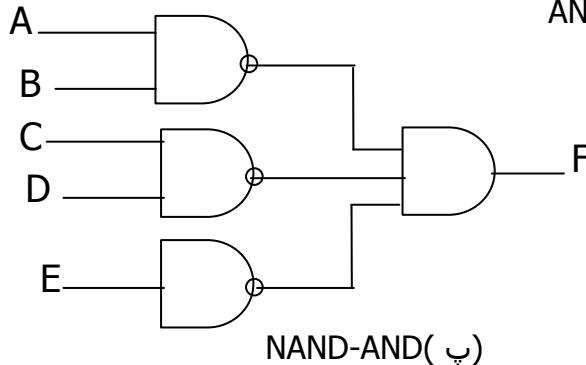
دو فرم NAND-AND و AND-NOR معادل یکدیگرند و می توان آنها را هم شرح داد . هر دوی آنها عمل AND-OR-INVERT را همانطور که در شکل (۳-۲۳) نشان داده شده انجام می دهند . فرم AND-NOR با یک عمل معکوس سازی توسط یک دایره کوچک در خروجی گیت NOR فرم AND-OR را شبیه سازی کرده و تابع را بصورت زیر پیاده

سازی می نماید :



AND-NOR ( الف )

AND-NOR ( ب )



NAND-AND ( پ )

شکل (۳-۲۳) مدارهای AND-OR-INVERT

با استفاده از سمبل گرافیکی معادل دیگری برای گیت NOR ، دیاگرام شکل (۳-۲۳) الف) را خواهیم داشت . دقت کنید که متغیر E مکمل نشده چون تنها تغییر صرفاً در سمبل گرافیکی گیت NOR بوده است . حال دایره ها را از پایانه های ورودی در گیت طبقه دوم به پایانه های خروجی طبقه اول منتقل می کنیم . در نهایت یک معکوس کننده برای متغیر E بخاطر نگهداشتن دایره لازم است . و یا می توان معکوس کننده ها را حذف کرد و ورودی E را بصورت مکمل در نظر گرفت . مدار شکل (۳-۲۳) پ) به فرم NAND-AND می باشد که در شکل (۳-۲۲) برای پیاده سازی عمل AND-OR-INVERT نشان داده شده است.

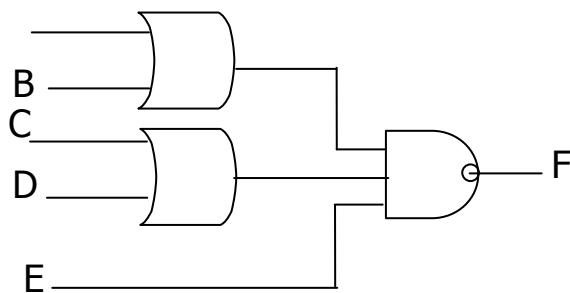
برای پیاده سازی یک AND-OR باید عبارتی ، بر حسب جمع حاصلضرب ها موجود باشد . پیاده سازی AND-OR-INVERT نیز به استثنای معکوس کردنش شبیه AND-OR است . بنابراین اگر مکمل تابع بر حسب حاصلضرب ها ساده شود ( بوسیله ترکیب ها در نقشه ) می توان  $F'$  را بوسیله AND-OR پیاده سازی کرد و وقتی  $F'$  از قسمت معکوس کننده عبور کند تابع F تولید می شود . پیاده سازی AND-OR-INVERT بعداً با یک مثال نشان داده خواهد شد .

### پیاده سازی OR-AND-INVERT

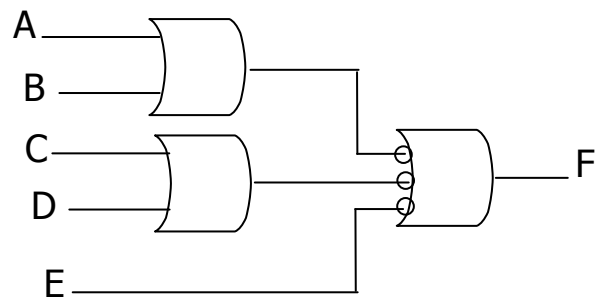
فرمهای NOR-OR و OR-NAND ، عمل OR-AND-INVERT را اجرا می نمایند ، که در شکل (۳-۲۴) نشان داده شده است . فرم OR-NAND ، فرم OR-AND را به استثنای معکوس کردن که بوسیله دوایر در خروجی گیت NAND انجام می شود .- شبیه سازی نموده و تابع زیر را پیاده می کند .

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

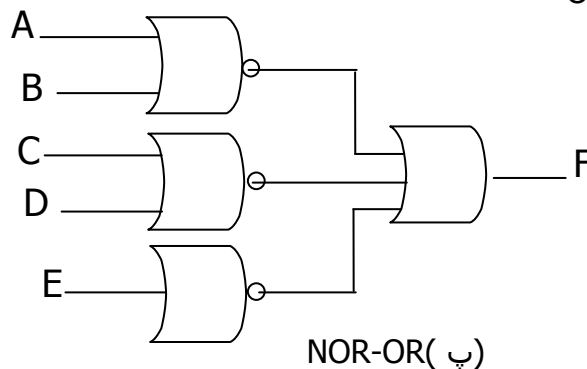
با استفاده از سمبل گرافیکی معادل دیگری برای گیت NAND دیاگرام شکل (۳-۲۴)، بدست می آید . مدار در قسمت (پ) بوسیله انتقال دایره های کوچک از ورودی های گیت طبقه دوم به خروجی گیت های طبقه اول حاصل می شود . مدار شکل (۳-۲۴) پ ( یک مدار به فرم NOR-OR است که جهت پیاده سازی عمل OR-AND-INVERT در شکل (۳-۲۲) نشان داده شده است . برای پیاده سازی OR-AND-INVERT به عبارتی بر سحبه ضرب حاصل جمع ها نیاز داریم . اگر مکمل تابع بر حسب ضرب حاصل جمع ها ساده شود میتوان  $F'$  را با قسمت OR-AND پیاده کرد و وقتی  $F'$  از یک معکوس کننده عبور کند ، مکمل  $F$  یا همان  $F$  را در خروجی آن خواهیم داشت .



OR-NAND (الف)



OR-NAND( ب)



NOR-OR( پ)

شکل (۳-۲۲) مدارهای OR-AND-INVERT

### خلاصه مطلب و مثال

در جدول (۳-۴) روشهای پیاده سازی یک تابع بول به هر چهار فرم و طبقه خلاصه شده است . به دلیل وجود قسمت معکوس کننده در هر حالت ، مناسب است تا از ساده

سازی  $F'$  ( مکما تابع ) استفاده شود . زیرا وقتی ک  $F'$  به یکی از فرمهای پیاده شود . در حقیقت مکمل تابع به فرم AND-OR یا OR-AND بدست می آید. چهار فرم دو طبقه ، این تابع را معکوس کرده و مکمل  $F'$  را در خروجی بدست می دهند که در حقیقت همان  $F$  است .

مثال ۳-۱۱ : تابع شکل (۳-۱۹ الف) را به چهار فرم دو طبقه که در جدول (۳-۴) آمده پیاده سازی کنید . مکمل تابع برحسب جمع حاصلضرب ها بوسیله ترکیب  $\cdot$  ها در جدول ساده شده عبارتست از :

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

جدول (۳-۴) پیاده سازی با سایر فرم های دو طبقه

معادل فرم مفید	پیاده سازی تابع	ساده کردن $F'$ بفرم	خروجی
( a )	( b)*		از
AND-NOR	NAND-AND	AND-OR-INVERT	F مجموع حاصلضرب ها بوسیله ترکیب $\cdot$ ها در نقشه
OR-NAND	NOR-OR	ORND-INVERT-A	F ضرب حاصلجمع ها با ترکیب ۱ها در نقشه و سپس مکمل سازی

\* فرم b برای جملات یک متغیره نیاز به یک NOR و یا NAND یک ورودی ( معکوس کننده ) دارد .

می توان خروجی طبیعی این تابع را بصورت زیر بیان کرد :

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

که به فرم AND-OR-INVERT است . پیاده سازی های AND-NOR و NAND-AND در شکل (۳-۲۵ الف) نشان داده شده است . توجه کنید که یک NAND تک ورودی یا

یک گیت معکوس کننده در پیاده سازی NAND-AND لازم است که در حالت AND-NOR چنین نیست . اگر متغیر  $z'$  را به فرم جای  $z$  در ورودی بکار ببریم ، می توان معکوس کننده را حذف کرد . فرم OR-AND-INVERT به عبارت ساده شده مکمل تابع بر حسب ضرب حاصلجمع ها نیاز دارد که برای بدست آوردن این عبارت می بایت ۱ ها را در نقشه با هم ترکیب کرد :

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

و سپس مکمل تابع را بدست آورد :

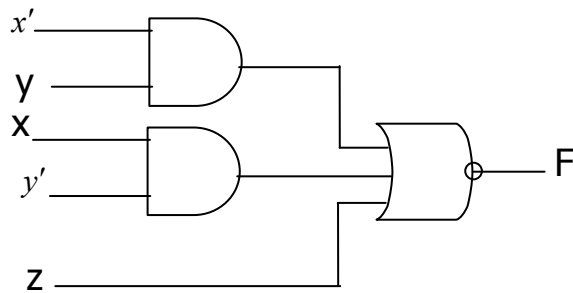
$$F' = (x + y + z)(x' + y' + z)$$

خروجی طبیعی  $F$  را می توان به فرم زیر بیان کرد :

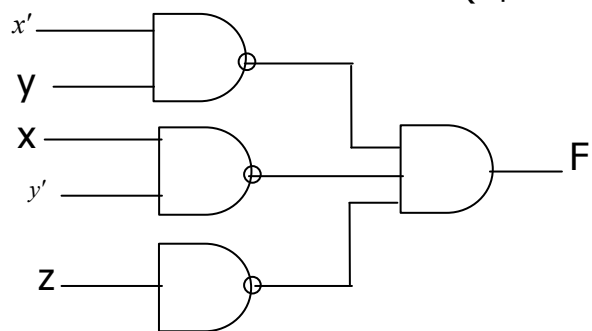
$F = [(x + y + z)(x' + y' + z)]'$  که به فرم OR-AND-INVERT است . با استفاده از این

عبارت می توان تابع را به فرمهای OR-NAND , NOR-OR پیاده سازی کرد که در شکل

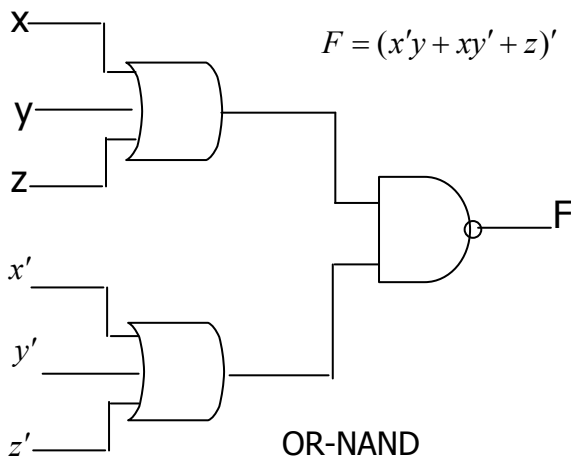
(۲۵-۳ ب) نشان داده شده است .



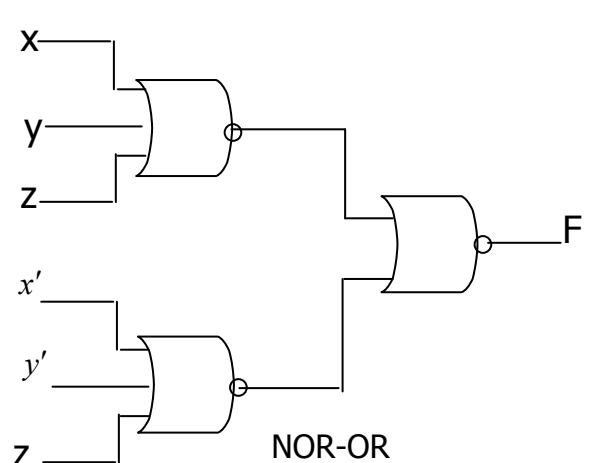
AND-NOR



NAND-AND



OR-NAND



NOR-OR

### ۸-۲- حالات بی اهمیت

۱ ها و ۰ های نقشه بیانگر ترکیبی از متغیرهاست که بترتیب تابع را ۱ و یا ۰ می نامند . معمولاً ترکیبات حاصل از جدول درستی حالاتی هستند که تحت آنها تابع برابر ۱ می باشد و در سایر مکانهای نقشه مقدراتابع ۰ فرض می گردد . این فرض همیشه صحیح نیست ، زیرا در بعضی کاربردها ترکیبات معینی از متغیر های ورودی هرگز وجود ندارد بعنوان مثال یک کد دهندهی چهار بیتی دارای شش ترکیب بالا استفاده است . تابعی که خروجی های نامشخص یا بلااستفاده ، در ازای برخی از ترکیبات ورودی را دارد به توابع ناقص معروفند . در اغلب کاربردهای ما درواقع به اینکه تابع درازای مینترم های نامعین چه مقداری دارد توجهی نمی کنیم . به این دلیل مینترم نامعین را در تابع ، حالات بی اهمیت نام می گذاریم . این حالات بی اهمیت در نقشه برای ساده سازی بیشتر عبارت بول بکار می روند .

باید توجه داشت که یک مینترم بی اهمیت ترکیبی از متغیرهاست که مقدار منطقی آن نامشخص است . به همین دلیل است که نمی توان یک حالت بی اهمیت را در نقشه با ۱ نشان داد زیرا این عمل به این معنی است که تابع برای این ترکیب خاص ورودی ها همواره برابر ۱ است . بطور مشابه گذاشتن ۰ در مربع های نقشه به معنی ۰ بودن همیشگی تابع است . لذا برای تشخیص ۰ ها و ۱ ها واقعی تابع ، حالت بی اهمیتی را با X نمایش می دهیم . بنابراین یک X بیانگر این واقعیت است که ما به ازای مینترم خاصی به ۰ یا ۱ شدن F اهمیتی نمی دهیم .

به هنگام انتخاب مربع های همجوار درنقشه برای ساده نمودن تابع ، با این ایده که ساده ترین عبارت حاصل گردد ، X ها را برابر ۰ و یا ۱ فرض می نمایم . در ساده

سازگی تابع می توانیم با توجه به ساده ترین فرم ممکن برای تابع به حالات بی اهمیت ۰ یا ۱ بدهیم .

مثال : ۱۲-۲ : تابع بول زیر را ساده کنید .

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

حالات بی اهمیت عبارتند از :

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

مینترم های تابع  $F$  ، ترکیبی از متغیرهاست که تابع را ۱ می کند . مینترم  $d$  ترکیبات بی اهمیتی هستند که ممکن است ۰ یا ۱ باشند . ساده سازی در شکل (۲۶-۳) نشان داده شده است . مینترم های  $F$  با ۱ و جملات  $d$  یا  $X$  مشخص شده اند و بقیه مربع ها با ۰ پر شده اند . برای بدست آوردن عبارت ساده شده بصورت جمع حاصلضرب ، ما باید هر پنج ۱ موجود در نقشه را در نظر بگیریم ، ولی ممکن است  $X$  ها را در نظر بگیریم ، و یا نگیریم و این به راه ساده سازی تابع وابسته است . جمله  $yz$  چهار مینترم را در ستون سوم پوسس می دهد . مینترم باقیمانده  $m_1$  می تواند با مینترم  $m_3$  ترکیب شده و جمله سه متغیره  $w'x'z'$  را بدهد . با این وجود ، با منظور کردن یکیا دو  $x$  همجوار ، ما می توانیم چهار مربع مجاور را ترکیب کنیم تا جمله دو متغیره حاصل گردد . در بخش (الف) از دیاگرام ، مینترم های بی اهمیت ۰ و ۲ با ۱ ها ترکیب شده اند که حاصل آن تابع ساده شده زیر است :

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

در بخش (ب) مینترم بی اهمیت ۵ با ۱ جایگزین شده و تابع ساده چنین است .

$$F = yz + w'z$$

هر یک از عبارات فوق خواسته های مثال را برآورده می سازد .

مثال فوق نشان داد که مینترم های بی اهمیت در نقشه ابتدا با X علامت گذاری می شوند و بعداً به ۰ یا ۱ بدل می گردند . انتخاب بین ۰ و ۱ به راهی که تابع غیر کامل یا ناقص ساده می شوند وابسته است . هر گاه انتخاب صورت گیرد ، تابع ساده شده حاصل متشکل از یک مجموع از مینترم ها و از جمله آنهایی است که ابتدا معین نشده بود ولی بعداً با ۱ جایگزین شده اند. دو عبارت ساده شده در مثال ۱۲-۳ را ملاحظه نمایید .

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

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

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

Z

$$F = yz + w'z' \text{ (الف)}$$

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

Z

$$F = yz + w'z \text{ (ب)}$$

شکل (۲-۲۶) مثال مربوط به حالات بی اهمیت

هر دو عبارت مینترم های 15.11.7.3.1 که تابع F را ۱ می کند ، دارا هستند . با مینترم های بی اهمیت ۰،۲،۵ بصورت متفاوتی درهر عبارت برخورد شده است . اولین عبارت شامل مینترم های ۰ و ۲ با مقدار ۱ و مینترم های ۵ با ۰ می باشد.



دومین عبارت شامل مینترم ۵ برابر با ۱ و مینترم های ۰ و ۲ برابر ۰ است . دو عبارت دو تابعی را نشان می دهند که بصورت جبری برابر نیستند .

هر دو ، مینترم های مشخص شده را پوشش می دهند ، ولی مینترم های بی اهمیت در آنها فرق دارد . هر دو عبارت قابل قبول است زیرا اختلاف آنها فقط در مقدار F به ازای مینترم های بی اهمیت است .

می توان عبارت حاصلضرب مجموعی نیز برای تابع شکل (۳-۲۶) بدست آورد . در اینحالت ، تنها راه ترکیب ۰ ها این است که مینترم های ۰ و ۲ را ۰ گرفته و مکمل تابع را بدست آوریم .

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

با گرفتن مکمل از F ، عبارت ساده بصورت حاصلضرب مجموع بدست می آید .

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

در این حالت مینترم های ۰ و ۲ با ۰ و مینترم ۵ با ۱ مقدار می گیرند .

**روش جدول بندی ( کوئین - مک کلاسکی ) :**

روش سیستماتیک برای ساده سازی عبارات منطقی : ( مثال :

$$f = \sum m(0,1,2,8,10,11,14)$$

۱- دسته بندی مینترمها بر اساس تعداد یکها ( صعودی ) :

<i>mi</i>	<i>w</i>	<i>x</i>	<i>y</i>	<i>z</i>	(0,1) 000-		
0	0	0	0	0	(0,2) 00-0		
11	0	0	0	1	(0,8) -000	(0,2,8,10)-0-0	} $x'z'$
2	0	0	1	0	(2,10) -010	(0,8,2,10)-0-0	
8	1	0	0	0	(8,10) 10-0		
10	1	0	1	0	(10,11) 101-		
11	1	0	1	1	(10,14) 1-10		$f = x'z' + w'x'y' + wx'y + wy'z$
14	0	1	1	0			

۲- هر دو مینترمهایی که در یک متغیر اختلاف داشته باشند قابل ترکیبند . ( تشکیل دسته های دوتایی ) هر مینترم در هر دسته با جملات دسته بعدی که پایین تر از خود قابل ترکیب است و مینترمهای شرکت کننده را تیک می زنیم .

۳- تشکیل دسته های چهار تایی : بایستی مینترمها نظیر به نظیر افزایش باشند .

$a > a', b > b' \leftarrow \begin{matrix} a & , & b \\ a' & , & b' \end{matrix}$  و باز هم تیک می زنیم آنهایی که تیک نخورده اند پوشش داده

نشده اند .

مثال :  $f = \sum m(0,1,2,8,10,11,14,18)$

**روش اصلاح یافته روش جدول بندی :**

۱- همان روش ۲- مینترمهایی را که در  $2^n$  تفاوت دارند ترکیب می کنیم که نیاز به نوشتن معادل باینری نیست و میزان اختلاف ( $2^n$ ) در یک پرانتز نوشته شود .

0	0	0	0	0	
1	0	0	0	1	
2	0	0	1	0	(0,1) (1)
8	1	0	0	0	(0,2) (2) (10,11) (1)
10	1	0	1	0	(0,8) (8) (10,14) (4)
11	1	0	1	1	(2,8) (8) (11,15) (4)
14	1	0	1	0	(8,10) (2) (14,15) (1)
15	1	1	1	1	

۲- برای دسته های چهارتایی دسته های دوتایی با هم ترکیب می شوند که علاوه

بر شرط افزایش اعداد داخل پرانتز آنها یکی باشد .  $a' - a = b' - b = \overline{2^n} \leftarrow \begin{matrix} a & b \\ a' & b' \end{matrix}$

(0,1) (1)								
(0,2) (2)	(0,2,8,10) (2,8)	→ معادلند	10	1	0	1	0	= wy
(0,8) (8)	(0,8,2,10) (8,2)	→	11	1	0	1	1	
(2,10) (8)	(10,11,14,15) (1,4)	→ معادلند	14	1	1	1	0	
(8,10) (2)	(10,14,11,15) (4,1)	→	15	1	1	1	1	
(10,11) (1)			w	x	y	z		
(10,14) (2)			0	0	0	0	0	= x'z'
(11,15) (4)			2	0	0	1	0	
(14,15) (1)			8	1	0	0	0	
			10	1	0	1	0	

$\Rightarrow F = wy + x'z' + w'z'y'$

اعدادی در پرانتز باقی مانند  $2^n$  هستند که n ها مکان متغیرهای حذف شده را نشان می دهد سپس مینترمها را می نویسیم و اسامیها را می یابیم .

مثال :

$$F = \sum m(1,4,6,7,8,9,10,11,15)$$

1	(1,9) (8)	(8, 0, 10, 11) (1,2)	$PI_1 = wx'$
4	(4,6) (2)	(8, 10, 9, 11) (2,1)	$PI_1 = x'y'z$
8	(8,9) (1)	0 1 0 0 0	$PI_3 = w'xz'$
6	(8,10) (2)	9 1 0 0 1	$PI_4 = w'xy$
9	(8,10) (1)	10 1 0 1 0	$PI_5 = xyz$
10	(9,11) (2)	11 1 0 1 1	$PI_6 = wyz$
7	(10,11) (1)		
11	(7,15) (8)		
15	(11,15) (4)		

1	(1,9)	(8)							
4	(4,6)	(2)							
8	(8,9)	(1)	(8, 0, 10, 11)	(1,2)					$PI_1 = wx'$
<u>6</u>	<u>(8,10)</u>	<u>(2)</u>	<u>(8, 10, 9, 11)</u>	<u>(2,1)</u>					$PI_1 = x'y'z$
9	(8,10)	(1)	0	1	0	0	0		$PI_3 = w'xz'$
10	(9,11)	(2)	9	1	0	0	1		$PI_4 = w'xy$
<u>7</u>	<u>(10,11)</u>	<u>(1)</u>	10	1	0	1	0		$PI_5 = xyz$
11	(7,15)	(8)	11	1	0	1	1		$PI_6 = wyz$
<u>15</u>	<u>(11,15)</u>	<u>(4)</u>							

**روش پیدا کردن اسامیها :**

مینترمهایی که تنها توسط یک PI تیک خورده اند را علامت می زنیم .

	1	4	6	7	8	9	10	11	15
PI <sub>1</sub>					*	*	*	*	
PI <sub>2</sub>	*					*			
PI <sub>3</sub>		*	*						
PI <sub>4</sub>		*	*						
PI <sub>5</sub>			*						*
PI <sub>6</sub>								*	*
	✓	✓	✓	□	✓	✓	✓	✓	□

	7	15
PI <sub>4</sub>	*	
PI <sub>5</sub>	*	*
PI <sub>6</sub>		*

مثال :  $F(A,B,C,D,E) = \sum(1,4,6,10,20,22,24,26) + \sum d(0,11,16,17)$

0	(0,1)	(7)		
1	(0,4)	(4)		
4	(0,16)	(16)	(0,4,16,20)	(4,16)
16	(4,6)	(2)	(0,16,4,20)	(16,4)
6	(4,20)	(16)	(4,6,20,22)	(2,16)
10	(16,20)	(4)	(4,20,6,22)	(16,2)
20	(16,24)	(8)	(10,11,26,27)	(1,16)
24	(6,22)	(16)	(10,26,11,27)	(16,1)
11	(10,11)	(1)		
22	(10,26)	(16)		
26	(20,22)	(2)		
27	(24,26)	(2)		
	(11,27)	(16)		
	(26,27)	(1)		

	1	4	6	10	20	22	24	26
PI <sub>1</sub>		*			*			
PI <sub>2</sub>		*	*		*	*		
PI <sub>3</sub>				*				
PI <sub>4</sub>	*							*
PI <sub>5</sub>							*	
PI <sub>6</sub>							*	*
	✓	✓	✓	✓	✓	✓		✓

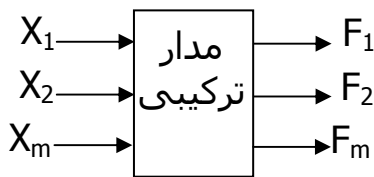
**در ماکسترم :** در ماکسترم بر حسب تعداد صفرها بررسی می شود در ترکیب دو دسته ای باید ماکسترم بالایی بزرگتر باشد . در نوشتن PI ها ما جملات حاصل جمع داشته و قوانین ماکسترم ها را رعایت می کنیم نهایتاً f همان حاصل ضرب PI ها می شود .

$$F = \pi M (0,1,2,3,4,6,10,11,12,15)$$

تمرین : تابع مقابل را ساده نمایید.

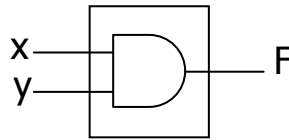
**مدارهای منطقی ترکیبی :**

مدار منطقی ترکیبی مداری است که در آن خروجیها در هر لحظه به ورودیها در همان



لحظه بستگی دارند .

مثال :



$$F = x.y$$

مدار  
ترکیبی

**مدارهای منطقی ترکیبی :**

۱- ( Small Scale Integrated Circuits ) SSI

مدارهای ترکیبی مقیاس کوچک مانند :

AND OR NOT XOR

۲- ( Medium OR NOT XOR ) MSI

مدارهای ترکیبی مقیاس متوسط مانند :

- مدارهای مبدل کد ها

- مدارهای جمع کننده / تفریق کننده

- مالتی پلکسرها

- دیکودرها

۲- ( Large Scale Integrated Circuits ) LSI

۴- ( Very Large scale Integrated Circuits ) VLSI همچون ریزپردازنده ها

**طراحی مدارهای ترکیبی**

برای طراحی یک مدار منطقی باید مراحل زیر را دنبال نمود:

۱- نحوه ی کار

۲- تعریف عملکرد مدار

۳- تعیین ورودیها و تعداد آنها

۴- تعیین خروجیها

۵- نوشتن روابط خروجیها برحسب ورودیها - جدول صحت (

۶- ساده سازی عبارات خروجیها برحسب ورودیها

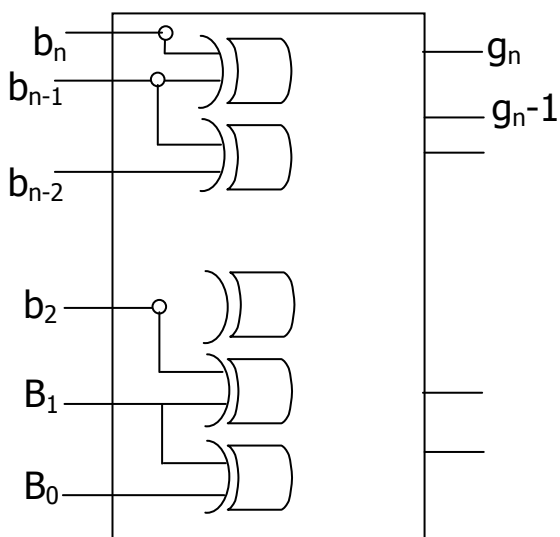
۷- پیاده سازی

### دسته بندی مدارهای ترکیبی

#### الف ( مدارهای مبدل کد :

جهت تبدیل کدهای مختلف به یکدیگر

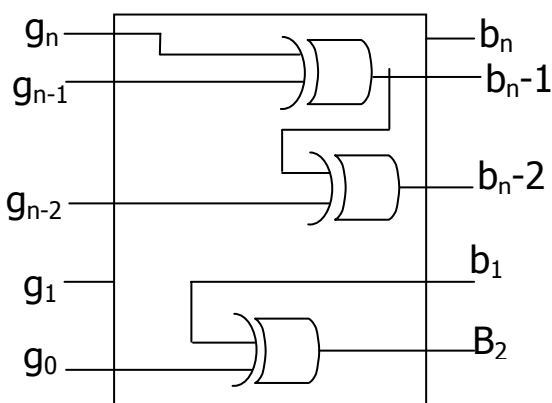
##### ۱- مدار مبدل کد باینری به گری :



$$g_n = b_n$$

$$g_i = b_i \oplus b_{i+1} \quad 0 \leq i \leq n-1$$

تاخیر به اندازه n-1 طبقه



##### ۲- مدار مبدل گری به باینری :

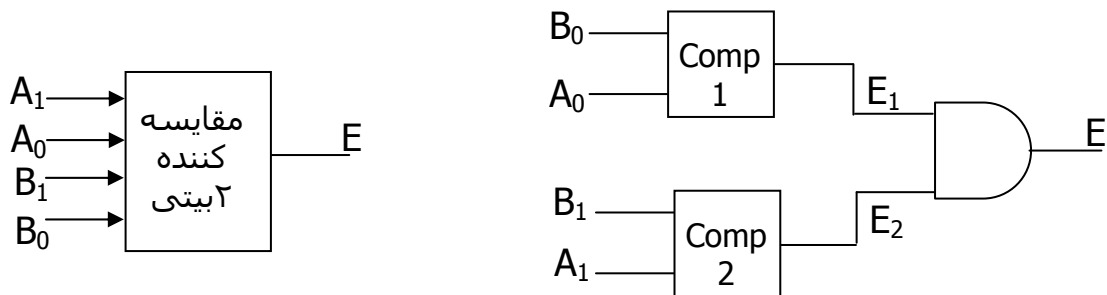
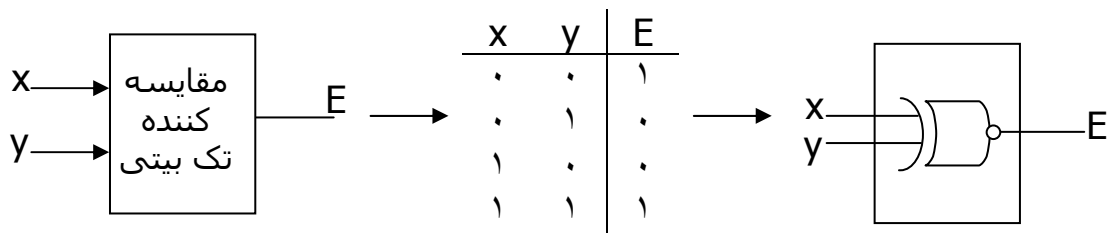
$$b_n = g_n$$

$$b_i = g_i \oplus b_{i+1} \quad 0 \leq i \leq n-1$$

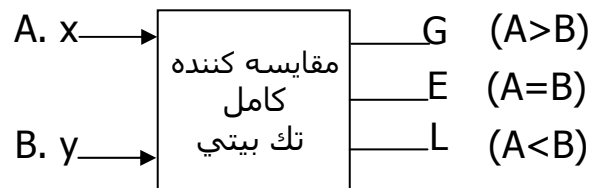
« تاخیر تجمعی به اندازه n طبقه »

**ب) مدارهای مقایسه کننده :**

برای مقایسه اطلاعات ورودی به کار می رود .



مدار مقایسه کننده کامل : ۱- ابتدا عملکرد برای یک بیت بررسی می شود.

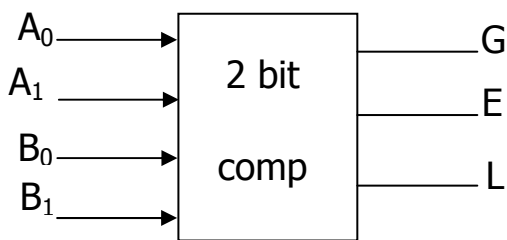
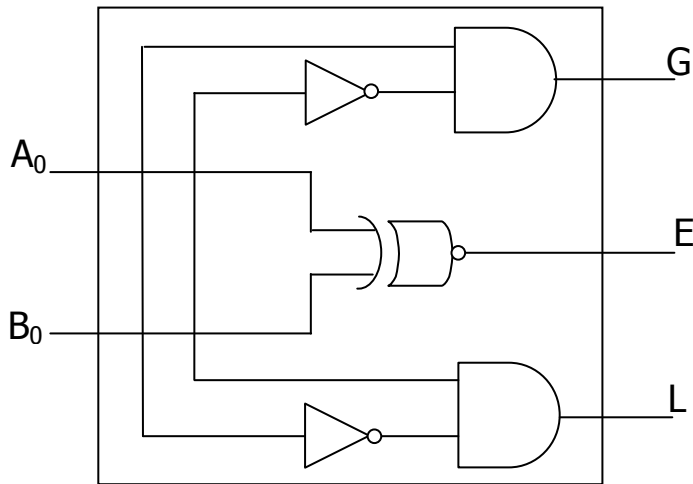


۲- بعد جدول صحت را تنظیم کنید .

$A_0$	$B_0$	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

۳- مدار را ترسیم کنید .





$$A: A_1 A_0$$

$$B: B_1 B_0$$

$$E = (A_1 \oplus B_1) \circ (A_0 \oplus B_0)$$

$$G = (A_1 > B_1) + (A_1 = B_1) \circ (A_0 > B_0)$$

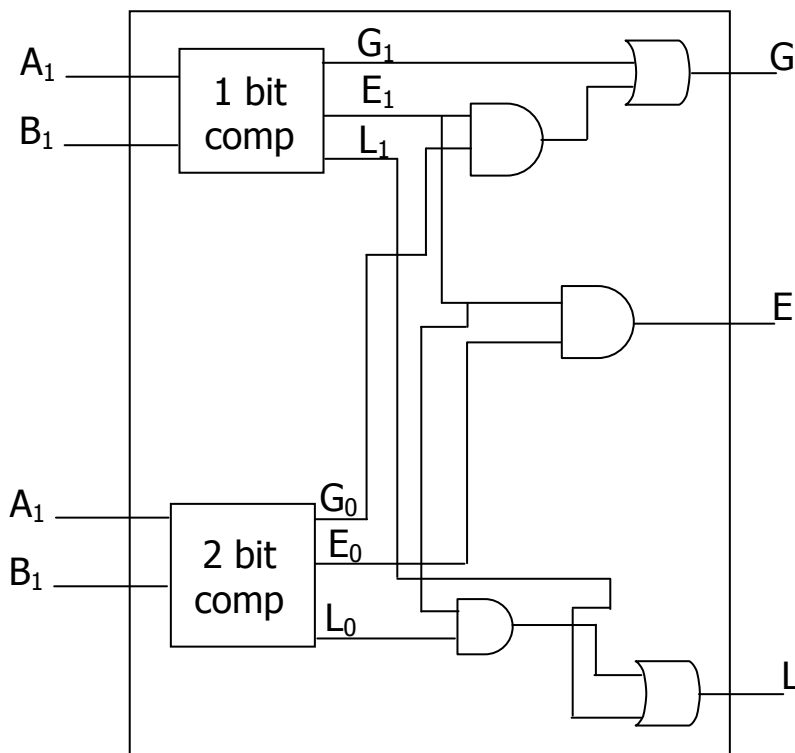
↓

$$G = A_1 B_1' + (A_1 \oplus B_1) \circ (A_0 B_0') = (G_1 + E_1 G_0)$$

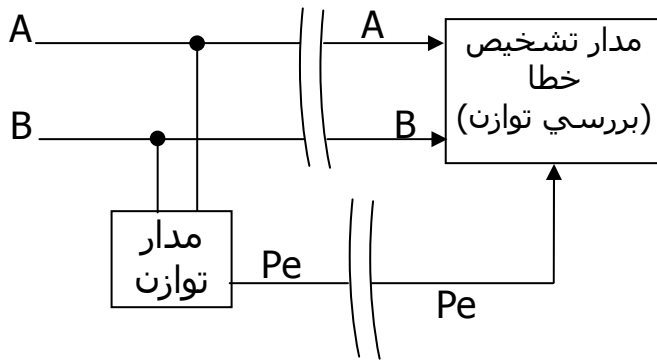
$$L = (A_1 < B_1) + (A_1 = B_1) \circ (A_0 < B_0)$$

↓

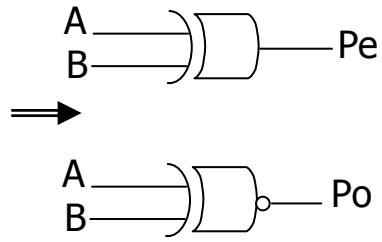
$$L = (A_1' B_1 + (A_1 \oplus B_1) \circ (A_0' B_0)) = L_1 + E_1 L_0$$



**(ب) مدارهای تولید توازن و تشخیص:**

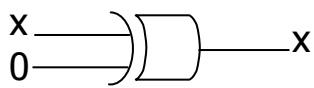
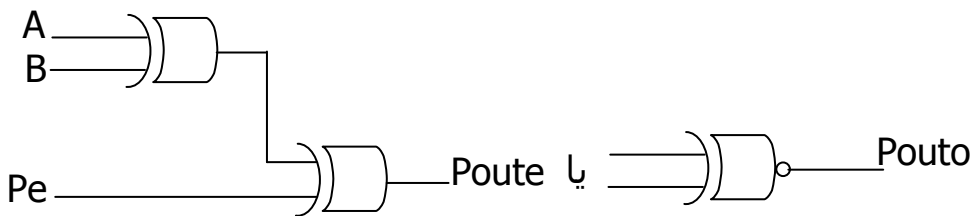


زوج		فرد					
A	B	Pe	Po	A	B	Pe	Poute
۰	۰	۰	۱	۰	۰	۰	۰
۰	۱	۱	۰	۰	۱	۱	۱
۱	۰	۱	۰	۱	۰	۱	۰
۱	۱	۰	۱	۱	۱	۰	۱

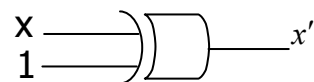


A	B	Pe	Poute
۰	۱	۰	۱
۱	۰	۱	۰

$$\Rightarrow \begin{aligned} Poute &= A \oplus B \oplus p_e \\ Poute &= [(A \oplus B) \oplus p_e]' = (A \oplus B) \oplus P \end{aligned}$$



$$F = x.1 + x'.0 = x$$

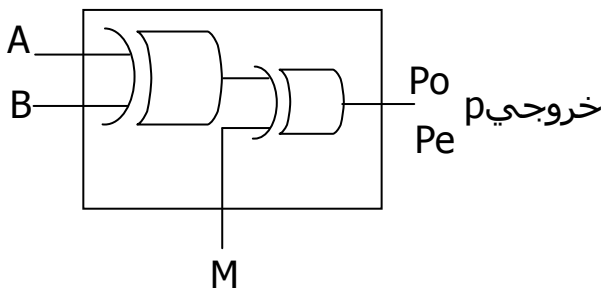


$$F = x.0 + x'.1 = x'$$

نکته :

هرگاه یکی از ورودیهای XOR ، ۰ باشد خروجی برابر با ورودی دیگر است و اگر ۱ باشد برابر با متمم خروجی دیگر است .

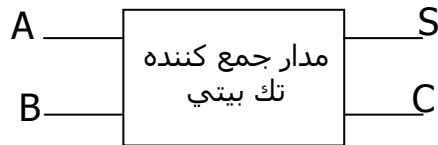
هدف : طراحی مداری که با یک بیت کنترل توازن زوج یا فرد ایجاد می کند .



$iF = M = 0 \text{ then } P = p_0, P = A \oplus B$   
 $iF = M = 1 \text{ then } P = p_1, P = A \oplus B$

پ) مدارهای جمع کننده / تفریق :

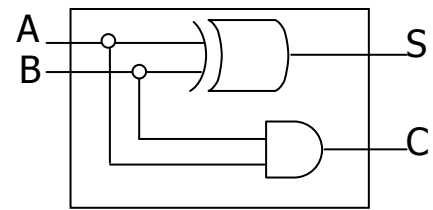
$$\begin{array}{r} A \\ + B \\ \hline CS \end{array}$$



۱- نیم-جمع کننده

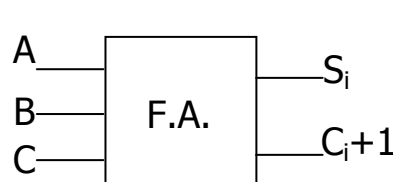
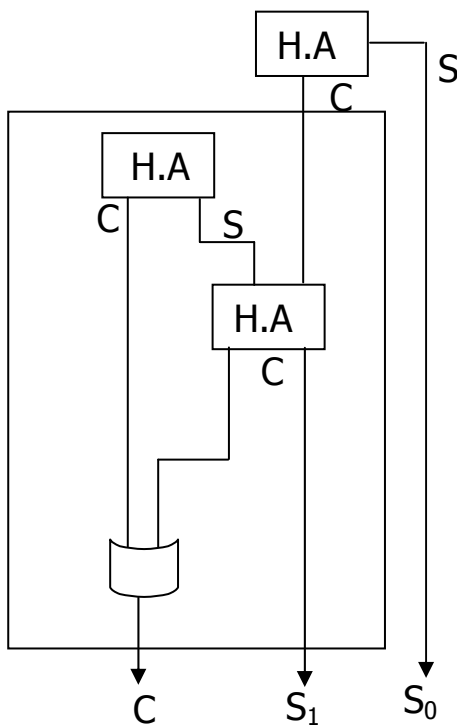
A	B	C	S
۰	۰	۰	۰
۰	۱	۰	۱
۱	۰	۰	۱
۱	۱	۱	۰

$$\begin{cases} S = A \oplus B \\ C = A.B \end{cases}$$



Half Adder (H.A) یا نیم جمع کننده

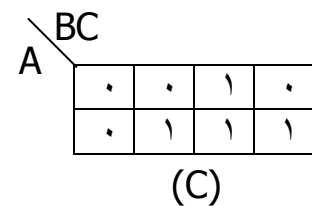
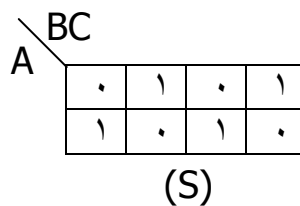
۲- مدار تمام-جمع کننده



$$\begin{array}{r} ۰۱ \\ + ۱۱ \\ \hline ۱۰۰ \\ C_2 S_1 S_1 \end{array}$$

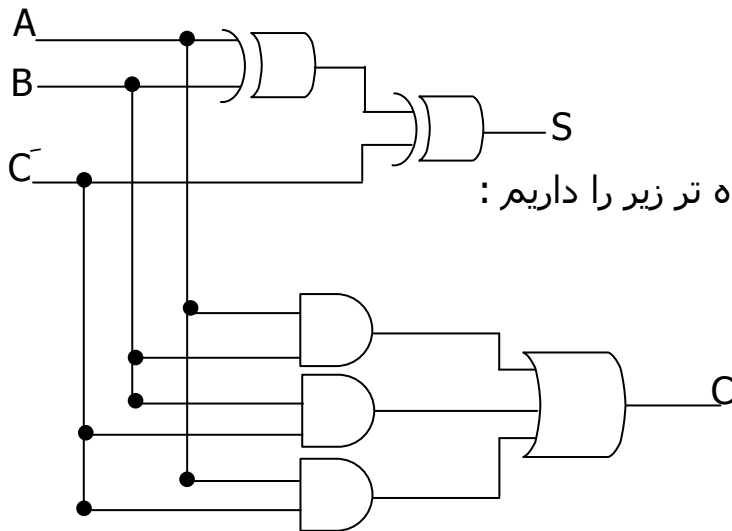
A	B	C	C	S
۰	۰	۰	۰	۰
۰	۰	۱	۰	۱
۰	۱	۰	۰	۱
۰	۱	۱	۱	۰
۱	۰	۰	۰	۱
۱	۰	۱	۱	۰
۱	۱	۰	۱	۰
۱	۱	۱	۱	۰

$$\begin{array}{r} A \\ B \\ + C \\ \hline CS \end{array}$$



$$\Rightarrow S = \sum m(1,2,4,7) = A \oplus B \oplus C$$

$$C = \sum m(3,5,6,7) = AB \oplus BC \oplus AC$$

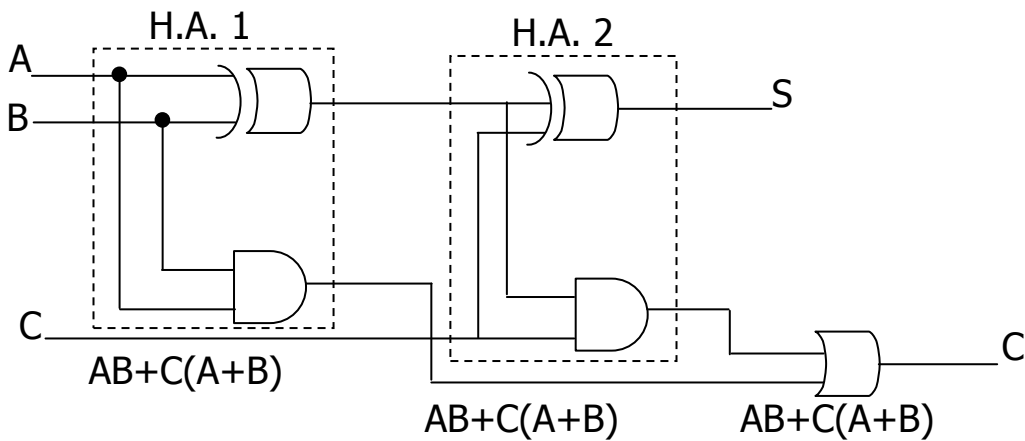


به جای این مدار شلوغ مدار ساده تر زیر را داریم :

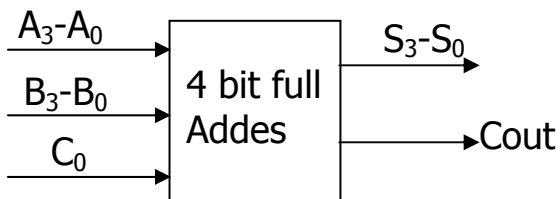
$$C_{out} = AB + BC + AC$$

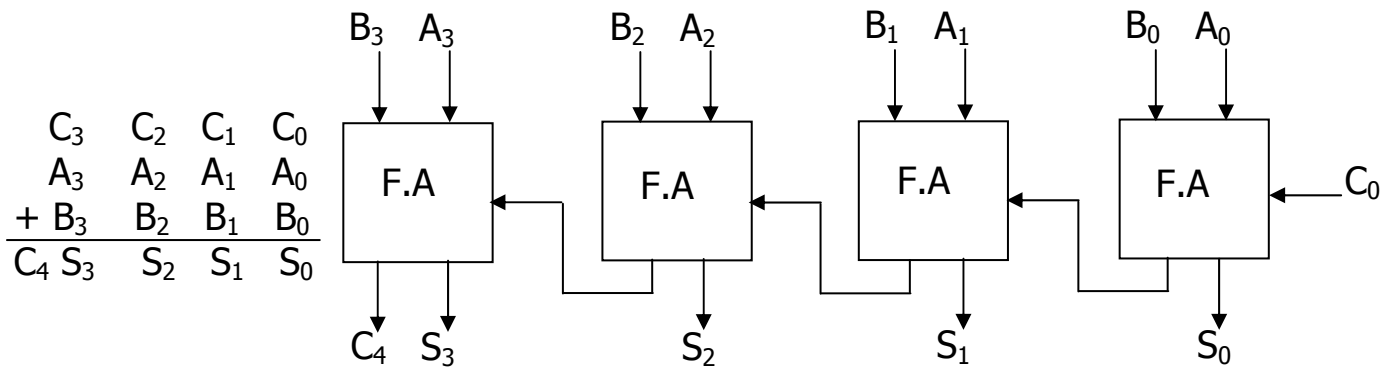
$$= AB + C(A + B)$$

لذا :

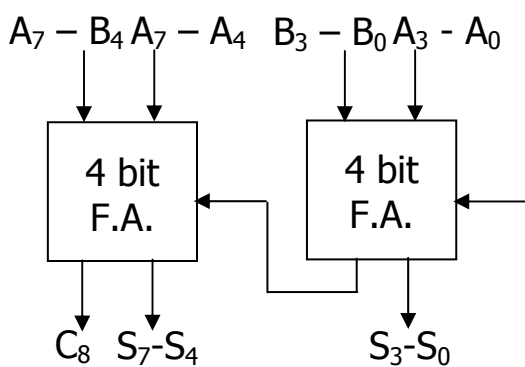


۲- جمع کننده کامل ۴ بیتی :



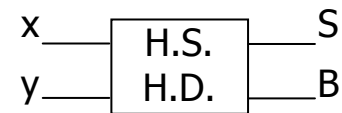


$$\begin{array}{r} C_3 \quad C_2 \quad C_1 \quad C_0 \\ A_3 \quad A_2 \quad A_1 \quad A_0 \\ + B_3 \quad B_2 \quad B_1 \quad B_0 \\ \hline C_4 \quad S_3 \quad S_2 \quad S_1 \quad S_0 \end{array}$$



8 bit Full Adder :

۲- مدار تفریق کننده :

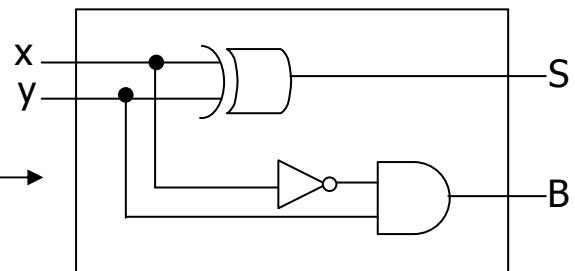


$$\begin{array}{r} x \\ -y \\ \hline BS \end{array}$$

X	Y	B	S
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

$$S = x \oplus y$$

$$B = x'y$$



X	Y	Z	B	S
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

$$S = \begin{array}{c|cccc} & yz & & & \\ \hline x & . & \cdot & . & \cdot \\ & \cdot & . & \cdot & . \end{array}$$

$$B = \begin{array}{c|cccc} & yz & & & \\ \hline x & . & \cdot & \cdot & \cdot \\ & . & . & \cdot & . \end{array}$$

$$\begin{array}{r} x \\ -y \\ -z \\ \hline BS \end{array}$$

$$S = x \oplus y \oplus z$$

$$B = x'z + x'y + yz$$