مدار منطقی

۱-۱- کامپیوتر و سیستم های دیجیتالی

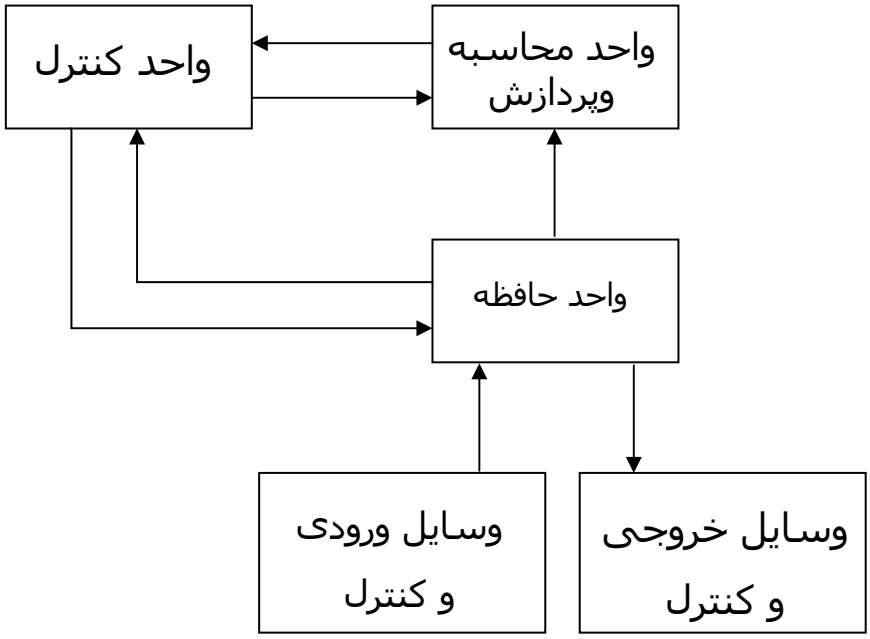
کامپیوتر های دیجیتال بسیاری از ، پیشرفت های علمی ، صنعتی و تجاری را که به صورت دیگر قابل دسترس نبودند ممکن ساخته اند . کامپیوتر ها در محاسبات علمی ، پردازش داده های تجاری ، کنترل ترافیک هوایی ، هدایت فضایی ، زمینه های فرهنگی و موارد بسیار دیگری مورد استفاده قرار گفته اند . کامپیوتر می تواند از مجموعه ای دستورالعمل های بنام برنامه که روی داده های مفروض عمل می کنند تبعیت نماید . استفاده کننده قادر است تغییرات گوناگونی را در برنامه ، داده ها و یا هر دوی آنها ، بر حسب نیاز ایجاد کند . به دلیل این انعطاف پذیری می توان نتیجه گرفت که کامپیوتر دیجیتال همه منظوره قادر هستند وظایف پردازش اطلاعات را در یک محدوده وسیع و متنوع به انجام برساند .

یک کامپیوتر دیجیتال همه منظوره ، شناخته شده ترین نمونه از یک دستگاه دیجیتال است . مشخصه یک سیستم دیجیتال ، توانایی اش در دستکاری اجزای گسسته اطلاعاتی است . کامپیوتر های دیجیتال اولیه بیشتر برای محاسبات عددی مورد استفاده قرار می گرفتند . در این حالت اجزاء گسسته ، ارقام هستند . عبارت کامپیوتر دیجیتال هم از همین کاربرد ناشی شده است . سیستم پردازش اطلاعات گسسته می تواند نام مناسبتری برای یک کامپیوتر دیجیتال باشد .

اجزاء گسسته اطلاعات در یک سیستم دیجیتال را کمیت هایی فیزیکی به نام سیگنال می سازند ، که سیگنالهای الکتریکی مثل ولتاژ و جریان های معمول ترین هستند . سیگنال ها در تمام سیستمهای دیجیتال الکترونیکی امروز ، تنها دو مقدار مجزا داشته و دودویی نامیده می شوند. به دلیل قابلیت اعتماد کمی که مدارهای

الکترونیکی چند مقداره دارا هستند ، طراح یک سیستم دیجیتال به استفاده از سیگنالهای دودویی مقید است . به عبارت دیگر می توان با استفاده از ده ولتاژ مختلف یک مدار ده حالته را طراحی کرد اما این مدار از لحاظ عملیاتی دارای قابلیت اعتماد کمی می باشد . بر عکس ، یک مدار ترانزیستوری خاموش یا روشن دارای دو مقدار سیگنال بوده و می تواند با قابلیت اعتماد زیادی ساخته شود .

بلوک دیاگرام کامپیوتر دیجیتال در شکل (۱-۱) نشان داده شده است . واحد حافظه ، برنامه ها ، داده های ورودی ، خروجی و داده های واسطه را ذخیره می کند . واحد پردازشگر یا پردازنده وظیفه اجرای عملیات ریاضی و دیگر وظایف پردازش داده های را آنطوری که در برنامه مشخص شده است بعهده دارد . واحد کنترل بر جریان اطلاعات بین قسمت های گوناگون نظارت می کند . این واحددستورات را یک به یک از برنامه ای که در حافظه ذخیره شده است بازیابی کرده و برای هر دستورالعمل ، پردازنده را مطلع می نماید تا عملیات مشخص شده در آن دستور را اجرا کند .



شکل (۱-۱) بلوک دیاگرام یک کامپیوتر دیجیتال

برنامه ها و داده هایی که توسط استفاده کننده تهیه شده اند بوسیله یک دستگاه ورودی مثل صفحه کلیدبه واحد حافظه منتقل می گردند . یک دستگاه خروجی مثل چاپگر نتایج محاسبات را دریافت کرده و نتایج چاپ شده را در اختیار استفاده کننده قرار می دهد . دستگاههای ورودی و خروجی ، سیستم های دیجیتال بخصوصی هستند که با قسمت های الکترو مکانیکی راه اندازی شده و بوسیله مدارهای الکترونیکی دیجیتال کنترل می شوند.

همانطوری که قبلاً اشاره شد کامپیوتر های دیجیتال روی اجزای گسسته اطلاعات عمل می کنند و این اطلاعات به شکل دودویی نمایش داده می شوند . عملوندهای مورد استفاده در محاسبات ممکن است در دستگاه اعداد دودویی بیان شوند . اجزای گسسته دیگر مثل ارقام دهدهی به کدهای دودویی نمایش داده می شوند . پردازش داده ها با استفاده از اجزای منطقی دودویی که از سیگنالهای دودویی استفاده می کنند انجام می شود و مقادیر در المان های حافظه دودویی ذخیره می شوند .

۲-۱- اعداد دودودیی

یک عدد در مبنای ده مثل ۷۳۹۲ مقداری معادل ۷ هزارتایی و به اضافه ۳ صدتایی به اشافه ۹ ده تایی به اضافه ۲ یکی را نشان می دهد . هزارگان ، صدگان و غیره توانهایی از ده هستند که دلالت بر مکان ضرایب می کنند . به منظور دقت بیشتر عدد ۷۳۹۲ بهتر است به صورت زیر نوشته شود :

$$7\times10^3+3\times10^2+9\times10^1+2\times10^0$$

بهر حال قرار داد این است که فقط ضرایب را بنویسیم و با توجه به مکان آنها توانهای ده را استنتاج نماییم . بطور کلی یک عد با نقطه اعشار در مبنای ده بوسیله ضرایب به صورت زیر نمایش داده می شوند :

$a_5\,a_4\,a_3\,a_2\,a_1\,a_0, a_{-1}\,a_{-2}\,a_{-3}$

که ضریب a_j یکی ارقام 0 تا ۹ بوده و مقدار اندیس j ارزش مکانی آن رقم ولذا توان دهی که ضریب بایستی در آن ضرب شود را می دهد .

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

بنا به تعریف گفته می شود که سیستم اعداد اعشاری از مبنا یا پایه ۱۰ می باشد چرا که ده رقم در آن استفاده می شود و ضرایب نیز در توانهایی از ده ضرب می گردند ، دستگاه دودویی سیستم دیگری از اعداد است . ضرایب دستگاه اعداد دودودیی دارای دو ارزش ممکن می باشند : ۰ و ۱ هر ضریب a_j ضریب در 2^j می شود .

برای مثال معادل مبنای ده عدد دودویی ۱۱۰۱۰,۱۱ همانطور که در زیر نشان داده شده ریال عدد ۲۶,۷۵ از ضریب توانایی از ۲ در ضرایب بدست می آید .

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

بطور کلی یک عدد در مبنای r به صورت حاصلضرب توانهای r در ضرایب مربوطه اش بیان می شود .

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

که ضرایب a_j بین ۰ تا -۱ هستند .

در مبنای r کمتر از ۱۰ می باشد ، مرسوم است که r رقم مورد نیاز برای یک عدد از دستگاه دهدهی گرفته می شود .وقتی مبنای عدد بزرگتر از ده است از حروف الفبا برای تکمیل ارقام دهدهی استفاده می گردد . در مبنای شانزده ، ده رقم اول از سیستم دهدهی گرفته شده و حروف F,E,D,C,B,A به ترتیب به جای اعداد

(۱۵,۱۴,۱۳,۱۲,۱۱,۱۰) بکار می روند . مثالی از یک عدد در مبنای شانزده بصورت زیر است ،

$$(B65F)_{10} = 11\times16^3 + 6\times16^2 + 5\times16 + 15 = (46687)_{10}$$

شانزده عدد اول دستگاه اعداد شانزده تایی ، هشت تایی ، دودویی و دهدهی در جدول (۱-۱) آمده است .

جدول (۱-۱) اعداد با مبناهای متفاوت

شانزده تایی (پایه ۱۶)	هشتایی (پایه ۸)	دودویی (پایه ۲)	دهدهی (پایه ۱۰)
۰	۰۰	۰۰۰۰	۰۰
۱	۰۱	۰۰۰۱	۰۱
۲	۰۲	۰۰۱۰	۰۲
۳	۰۳	۰۰۱۱	۰۳
۴	۰۴	۰۱۰۰	۰۴
۵	۰۵	۰۱۰۱	۰۵
۶	۰۶	۰۱۱۰	۰۶
۷	۰۷	۰۱۱۱	۰۷
۸	۱۰	۱۰۰۰	۰۸
۹	۱۱	۱۰۰۱	۰۹
A	۱۲	۱۰۱۰	۱۰
B	۱۳	۱۰۱۱	۱۱
C	۱۴	۱۱۰۰	۱۲
D	۱۵	۱۱۰۱	۱۳
E	۱۶	۱۱۱۰	۱۴
F	۱۷	۱۱۱۱	۱۵

اعمال ریاضی با مبنای r از همان قواعدی که برای اعداد دهدهی حاکم است پیروی می کند . وقتی از مبنای غیر از ۱۰ استفاده می شود می بایست دقت کرد تا فقط r رقم مجاز آن مبنا مورد استفاده قرار گیرد .مثالهای از جمع ، تفریق و ضرب دو عدد دودویی در زیر نشان داده شده است :

۱۰۱۱	مضروب	۱۰۱۱۰۱	مفروق	۱۰۱۱۰۱	مضاف
۱۰۱*	مضروب فیه	۱۰۰۱۱۱-	مفروق منه	۱۰۰۱۱۱+	مضاف الیه
۱۰۱۱		۰۰۰۱۱۰	باقیمانده	۱۰۱۰۱۰۰	حاصل جمع
۰۰۰۰					
۱۰۱۱					
۱۱۰۱۱۱	حاصل ضرب				

مجموع دو عدد دودویی طبق همان قوانین دستگاه دهدهی محاسبه می شود ، بجز اینکه ارقام با ارزش حاصل جمع در تمام مکان های با معنی فقط می تواند ۰ یا ۱ باشند . هر رقم نقلی بدست آمده در مکانی مفروض بوسیله جفت رقم های مرتبه بالاتر مورد استفاده قرار می گیرد . عمل تفریق کمی پیچیده تر است . قوانین باز هم همان قانونهای دهدهی هستند ، بجز اینکه رقم قرضی با ارزش مکانی داده شده ۲ واحد به رقم مفروق اضافه می کند . (یک رقم قرضی از دستگاه دهدهی ، ۱۰ واحد به رقم مفروق اضافه می کند) عمل ضرب بسیار ساده است . ارقام مضروب فیه همیشه ۱ یا ۰ هستند . بنابراین حاصل ضرب های جزئی یا ۰ و یا مساوری مضروب می باشند .

۳-۱- تبدیل مبنای اعداد

یک عدد دودویی به وسیله جمع کردن توانهایی از ۲ که مقدار ضرایبشان یک است به صورت دهدهی آن تبدیل می شود . برای مثال :

$$(1010.011)_2 = 2^3 + 2^1 + 2^{-2} + 2^{-3} = (10.375)_{10}$$

در زیر مثالی از تبدیل مبنای هشت به ده آمده است :

$$(630.4)_3 = 6\times 8^2 + 3\times 8 + 4\times 8^{-1} = (408.5)_{10}$$

در تبدیل مبنای ده به دو یا به هر مبنای دیگر راحت تر است که قسمت صحیح و قسمت اعشاری عدد را جدا کرده و هر کدام را به طور جداگانه تبدیل کنیم .

مثال ۱-۱- عدد ۴۱ را به دودویی تبدیل کنید .

ابتدا ۴۱ بر حسب ۲ تقسیم شده تا خارج قسمت ۲۰ و باقیمانده ۱/۲ بدست آید . خارج قسمت مجدداً تقسیم شده تا خارج قسمت و باقیمانده جدیدی حاصل گردد . این روال به همین صورت تا زمانی ادامه می یابد که خارج قسمت صحیح به دست آمده صفر شود . ضرایب عدد دودویی مطلوب به صورت زیر از باقیمانده ها بدست می آیند .

خارج قسمت صحیح		باقیمانده	ضریب عدد دودویی
$\frac{41}{2}=20$	+	$\frac{1}{2}$	$a_0=1$
$\frac{20}{2}=10$	+	۰	$a_1=0$
$\frac{10}{2}=5$	+	۰	$a_2=0$
$\frac{5}{2}=2$	+	$\frac{1}{2}$	$a_3=1$
$\frac{2}{2}=1$	+	۰	$a_4=0$
$\frac{1}{2}=0$	+	$\frac{1}{2}$	$a_5=1$

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

روال ریاضی فوق می تواند بصورت مناسبتری بصورت زیر عمل شود :

خارج قسمت صحیح	باقیمانده
۴۱	
۲۰	۱
۱۰	۰
۵	۰
۲	۱
۱	۰
۰	۱

جواب = ۱۰۱۰۰۱

تبدیل اعداد صحیح دهدهی به مبنای r شبیه به مثال مذکور است بجز اینکه تقسیم می بایست به جای ۲ بر r صورت گیرد .

مثال : ۲-۱- : عدد ۱۵۳ را به مبنای هشت ببرید.

۱۵۳	
۱۹	۱
۲	۳
۰	۲

$= (231)_8$

مثال : ۳-۱- عدد $(0.6875)_{10}$ را به مبنای دو ببرید .

	صحیح		کسری	ضریب
$0.6875\times 2=$	۱	+	۰,۳۷۵۰	$a_{-1}=1$
$0.3750\times 2=$	۰	+	۰,۷۵۰۰	$a_{-2}=0$
$0.7500\times 2=$	۱	+	۰,۵۰۰۰	$a_{-3}=1$
$0.5000\times 2=$	۱	+	۰,۰۰۰۰	$a_{-4}=1$

جواب : $(0.6875)_{10}=(0.a_{-1}a_{-2}a_{-3}a_{-4})_2=(0.1011_2$

برای تبدیل یک عدد کسری از مبنای ده به یک عدد در پایه r ، روش مشابهی انجام می شود .

فقط به جای ضرب در ۲ ، ضرب در r انجام می گردد و ضرایب حاصل از قسمتهای صحیح می توانند به جای ۰ و ۱ در محدوده بین ۰ تا ۱-r باشند .

مثال : ۱-۴ : عدد $(0.513)_{10}$ را به مبنای هشت ببرید .

$0.413\times 8 = 4.104$
$0.104\times 8 = 0.832$
$0.832\times 8 = 6.656$
$0.656\times 8 = 5.248$
$0.248\times 8 = 1.984$
$0.984\times 8 = 7.872$

جواب تا هفت رقم با معنی ، از قسمتهای صحیح حاصل ضرب ها بدست می آید .

$$(0.513)_{10} = (0.406517...)_8$$

تبدیل اعداد دهدهی که دارای هر دو قسمت صحیح و کسری هستند به این صورت انجام میگیرد که هر قسمت بطور مجزا تبدیل شده سپس جوابها با هم ترکیب می شوند . با استفاده از نتایج مثال ۱-۱ و ۱-۳ داریم :

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

از مثالهای ۱-۲ و ۱-۴ نیز داریم :

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

۱-۴ اعداد مبنای هشت و شانزده

تبدیل از مبنای دو به مبنای هشت و شانزده و بالعکس نقش مهمی در کامپیوترهای دیجیتال دارد . چون $2^3=8$ و $2^4=16$ است ، هر رقم در مبنای هشت مطابق سه رقم دودویی و هر رقم بر مبنای شانزده ، چهار رقم دودویی است . تبدیل مبنای دو به هشت به سادگی با تقسیم عدد دودویی به دسته های سه تایی از نقطه اعشاری

دودویی به سمت چپ و راست صورت می گیرد و به هر دسته از این اعداد یک رقم در مبنای هشت نسبت داده می شود . مثال زیر نشان دهنده روند مربوطه است :

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

تبدیل از مبنای دو به مبنای شانزده نیز مشابه با روند بالا است ، با این تفاوت که عدد دودویی به دسته های چهارتایی از ارقام تقسیم بندی می شوند.

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

هر عدد در مبنای شانزده (هشت ۹ متناسب با هر دسته از ارقام دودویی ، بعد از مطالعه مقادیر ثبت شده در جدول (۱-۱) به سادگی تعیین می شود .

تبدیل از مبنای هشت یا شانزده به مبنای دو با روشی عکس روش بالا صورت می گیرد . هر رقم در مبنای شانزده به چهار رقم معادل در مبنای دو تبدیل می شود . بطور مشابه هر رقم در مبنای شانزده به معادل دودویی چهار رقمی خود تبدیل می گردد . این مطلب در مثال زیر تشریح شده است :

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

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

کارکردن با اعداد دودویی به دلیل اینکه تعداد ارقامشان سه یا چهار برابر عدد معادلشان در مبنای ده می باشد مشکل است . (مثلاً عدد دودویی ۱۱۱۱ ۱۱۱۱ ۱۱۱۱ معادل عدد دهدهی ۴۰۹۵ است . با این وجود کامپیوترهای دیجیتال از اعداد دودویی استفاده می کنند و گاهی نیز لازم است که اپراتور و یا استفاده کننده مستقیماً به وسیله اعداد دودویی با ماشین ارتباط برقرار کند . یک راه برای نگهداری سیستم دودویی در کامپیوتر که ضمناً تعداد ارقام را نیز کاهش می دهد ، این است

که از ارتباط بین سیستم اعداد دودویی و سیستم هشت تایی یا شانزده استفاده شود . با این روش انسان می تواند بر حسب اعداد مبنای شانزده یا هشت تایی فکر کرده و در مواقعی که ارتباط مستقیم با ماشین لازم است تبدیل لازمه را با بازدید کردن این اعداد انجام دهد . به این ترتیب عدد دودویی ۱۱۱۱ ۱۱۱۱ ۱۱۱۱ که دارای دوازده رقمن است در مبنای هشت به صورت چهار رقم VVVV بیان می شود و یا در مبنای شانزده به صورت سه رقم FFF خواهد بود .در ارتباطات بین مردم (در مورد اعداد اردودویی در کامپیوتر) نمایش اعداد در مبناهای هشت و شانزده مطلوب تر است زیرا که در این مبناها اعداد به صورت کوچکتری با ۱/۳ یا ۱/۴ تعداد ارقام معادلشان در دودویی قابل نمایش هستند .

۱-۵- مکمل ها

مکمل ها در کامپیوترهای دیجیتال برای ساده کردن عمل تفریق و یا عملیات منطقی به کار می روند . در هر مبنای r دو نوع مکمل برای هر سیستم وجود دارد : یکی مکمل مبنا یا پایه و دیگری مکمل مبنا یا پایه کاهش یافته است . فرم اول به مکمل r و دومی به مکمل (r -۱) موسوم است . وقتی مقدار پایه را جایگزین کنیم ، برای اعداد دودویی مکمل های ۲ و ۱ برای اعداد مکمل های ۱۰ و ۹ را خواهیم داشت .

مکمل در پایه کاهش یافته

برای عددی مانندN در مبنا یا پایه r که دارای n رقم است ، مکمل (r-۱) مربوط به N بصورت $(r^n-1)-N$ تعریف می شود . برای اعدادی با r = ۱۰ و r -۱= ۹ ، مکمل ۹ برای عدد N برابر است با $(10^n-1)-N$. عدد ۱۰^n برابرست با یک عدد ۱ که n عدد ۰ بدنبال آن آمده است . بهمین ترتیب ۱۰^n-۱ برابرست با n عدد ۹ . مثلاً اگر

n=۴ باشدداریم $10^4=10000$ و $10^4-1=9999$ دیده می شود که مکمل ۹ یک عدد دهدهی از تفریق هر رقم آن از ۹ حاصل می شود . به چند مثال عددی توجه کنید .

مکمل ۹ عدد ۵۴۶۷۰۰ برابرست با

۹۹۹۹۹۹-۵۴۶۷۰۰=۴۵۳۲۹۹

مکمل ۹ عدد ۰۱۲۳۹۸ برابرست با

۹۹۹۹۹۹-۰۱۲۳۹۸=۹۸۷۶۰۱

برای اعداد دودویی ، r=۲ و r-۱=۱ است ، لذا مکمل ۱ عدد n برابرست با $(۲^n-۱)$ – n . مجدداً ۲^n از یک عدد دودویی متشکل از یک ۱ و تعدادی ۰ بدنبال آن است . ۲^n - ۱ نیز بوسیله n عدد ۱ نشان داده می شود . مثلاً اگر n=۴ باشد داریم $(10000)_2=2^4$ و $2^4-1=1111$.

مکمل ۱ یک عدد دودویی از تبدیل ۱ها به ۰ها و به ۱ حاصل می شود .

مکمل ۱ عدد ۱۰۱۱۰۰۰ برابرست با ۰۱۰۰۱۱۱

مکمل ۱ عدد ۰۱۰۱۱۰۱ برابرست با ۱۰۱۰۰۱۰

مکمل (r-۱) اعداد مبنای هشت و شانزده به ترتیب از تفریق ارقام از V یا F (معادل ۱۵ دهدهی) حاصل می شود .

مکمل پایه r

مکمل r یک عدد n رقمی مانند N در مبنای r بصورت r^n-N به ازای $N \neq 0$ و بصورت ۰ به ازای N=۰ تعریف می شود . با مقایسه این نوع مکمل با مکمل (r-۱) ملاحظه می شود که مکمل r از جمع ۱ با مکمل (r-۱) حاصل می شود . زیرا $r^n-N=[(r^n-1)-N]+1$ است . بنابراین مکمل ۱۰ یک عدد دهدهی مانند ۲۳۸۹ برابراست با ۷۶۱۱=۱+۷۶۱۰ و با افزودن ۱ به مقدار مکمل ۹ حاصل گردیده است .

مکمل عدد دودویی ۱۰۱۱۰۰ برابر است با ۰۱۰۱۰۰=۱+۰۱۰۰۱۱ و از جمع ۱ با مکمل ۱ عد حاصل شده است .

چون 10^n عددی است که با یک ۱ و n عدد ۰ بدنبال آن ساخته شده است ، مکمل عدد N یعنی $10^n - N$ نیز با تغییر ندادن ۰ های کم ارزشتر و کسر اولین رقم غیر صفر از ۱۰ و تفریق تمام ارقام با ارزشتر از ۹ حاصل می شود .

مکمل ۱۰ عدد ۰۱۲۳۹۸ برابرست با ۹۸۷۶۰۲

مکمل ۱۰ عدد ۲۴۶۷۰۰ برابرست با ۷۵۳۳۰۰

مکمل ۱۰ اولین عدد از تفریق ۸ از ۱۰ در کم ارزش ترین مکان و تفریق بقیه ارقام از ۹ حاصل شده است . مکمل ۱۰ دومین عد بدین فرم حاصل شده که دو عدد ۰ با ارزش کمتر بدون تغییر مانده درحالیکه ۷ از ۱۰ و سه رقم دیگر از ۹ کسر شده است .

بطور مشابه مکمل ۲ دمی تواند با بدوت تغییر گذاردن ۰ های کم ارزش تر و اولین ۱ پس از آنهاو جایگزینی ۱ها یا ۰ ها با ۱ در سایر ستون های ارقام با ارزشتر بدست آید .

مکمل ۲ عدد ۱۱۰۱۱۰۰برابرست با ۰۰۱۰۱۰۰

مکمل ۲ عدد ۰۱۱۰۱۱۱برابرست با ۱۰۰۱۰۰۱

مکمل ۲ اولین عدد با بدون تغییر گذاشتن دو ۰ با ارزش کمتر و نیز اولین ۱ پس از آنها و سپس جایگزینی ۱ ها با ۰ و ۰ ها با ۱ در چهار ستون باقیمانده حاصل شده است . دومین مکمل ۲ با تغییر ندادن کم ارزش تری ۱ و مکمل نمودن بقیه ارقام بدست آمده است .

اگر عدد اولیه N دارای ممیز باشد باید آن را موقتا حذف و مکمل های r و (r-۱) را بدست آورد . سپس آن را به همان مکانی نسبی عدد مکمل بازگرداند . همچنین ذکر

این نکته که مکمل مربوط به مکمل یک عدد همان اولیه را نتیجه می دهد مفید بنظر می رسد . مکمل r عدد N برابرست با r^n - N مکمل مربوط به این مکمل برابرست با $r^n-(r^n-N)=N$ که همان عدد اولیه است .

تفریق به کمک مکمل ها

تفریق دو عدد n رقمی بدون علامت M-N در پایه r بطریق زیر صورت می گیرد .

١- مفروق M را به مکمل r مفروق منه N اضافه کنید یعنی

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

٢- اگر $M \geq N$ باشد ، جمع یک رقم نقلی نهایی r^n تولید می کند که چشم پوشی می شود ، آنچه باقی می ماند M-N است .

٣- اگر $M < N$ باشد ، جمع هیچگونه رقم نقلی نهایی تولید ننموده وجواب $r^n-(N-M)$ می باشد که مکمل r عدد (M-N) است . برای یافتن جواب بفرم معمول ، مکمل r حاصل جمع را بدست آورده و یک علامت منفی در جلو آن قرار می دهیم .

مثال ١-٥ : با استفاده از مکمل ١٠، ٣٢٥٠ – ٧٢٥٣٢ را بدست آورید .

M =	٧٢٥٣٢
مکمل ١٠ عدد N =	+٩٦٧٥٠
حاصل جمع =	١٦٩٢٨٢
حذف رقم نقلی 10^5 =	-١٠٠٠٠٠
جواب =	٦٩٢٨٢

دقت کنید که M دارای پنج رقم ولی N فقط دارای چهار رقم است . چون هر دو عدد باید دارای تعداد ارقام برابر باشند ، پس باید بصورت ٠٣٢٥٠ نوشته می شود .

مثال ۶-۱ : با استفاده از مکمل ۱۰ ، ۳۲۵۰-۷۲۵۳۲ را بدست آورید .

M = ۰۳۲۵

N عدد ۱۰ مکمل = +۲۷۴۶۸

حاصل جمع = 30718

رقم نقلی وجود ندارد

جواب : -(مکمل ۱۰ عدد ۳۰۷۱۸) =-۶۹۲۸۲

توجه کنید چون ۷۲۵۳۲ > ۳۲۵۰ است ، جواب منفی است .

مثال ۷-۱ : با فرض دود عدد دودویی X=۱۰۱۰۱۰۰ و Y = ۱۰۰۰۰۱۱ ، تفریق های :

(الف) X-Y و (ب) Y-X را با استفاده از مکمل ۲ بدست آورید .

X = ۱۰۱۰۱۰۰

Y عدد ۲ مکمل = +۰۱۱۱۱۰۱

حاصل جمع = ۱۰۰۱۰۰۰۱

رقم نقلی حذف شده $۲^{۷}$ = -۱۰۰۰۰۰۰۰

جواب X-Y = ۰۰۱۰۰۰۱

Y = ۱۰۰۰۰۱۱

X عدد ۲ مکمل = +۰۱۰۱۱۰۰

حاصل جمع = ۱۱۰۱۱۱۱

رقم نقلی وجود ندارد

جواب : Y-X = -(مکمل ۲ عدد ۱۱۰۱۱۱۱) = -۰۰۱۰۰۰۱

تفریق اعداد بدون علامت می تواند با استفاده از مکمل (R-۱) نیز انجام شود . بخاطر بیاورید که مکمل (R-۱) یکی کمتر از مکمل r است . به این علت ، نتیجه جمع مفروق به مکمل مفروق منه حاصل جمعی تولید می کند که یکی کمتر از تفاضل صحیح بهنگام رخداد رقم نقلی نهایی است . حذف رقم نقلی نهایی و افزودن آن به حاصل جمع بنام رقم نقلی چرخشی خوانده می شود .

مثال ۸-۱ : مثال ۷-۱ را با استفاده از مکمل ۱ تکرار کنید .

(الف)

X-Y=۱۰۱۰۱۰۰-۱۰۰۰۰۱۱

۱۰۱۰۱۰۰ = X

۰۱۱۱۱۰۰ + = مکمل ۱ عدد Y

۱۰۰۱۰۰۰۰ = حاصل جمع

۱+ = رقم نقلی چرخشی

۰۰۱۰۰۰۱ = X-Y : جواب

(ب)

Y-X=۱۰۰۰۰۱۱-۱۰۱۰۱۰۰

۱۰۰۰۰۱۱ = Y

۰۱۰۱۰۱۱ + = مکمل ۱ عدد X

۱۱۰۱۱۱۰ = حاصل جمع

رقم نقلی وجود ندارد

-۰۰۱۰۰۰۱ = (مکمل ۱ عدد ۱۱۰۱۱۱۰)-= Y-X: جواب

توجه کنید که نتیجه منفی پس از گرفتن مکمل ۱ از حاصل جمع بدست آمده است . زیرا مکمل ۱ در بالا بکار رفته است . روش رقم نقلی چرخشی برای تفریق اعداد دهدهی بدون علامت با مکمل ۹ نیز قابل استفاده است .

۱-۶ اعداد دودویی علامت دار

بعلت محدودیت سخت افزار ، کامپیوتر ها باید هر چیزی را با ارقام دودیی نشان دهند ، که معمولاً این ارقام بیت نامیده می شوند . معمول است که سمت چپ ترین بیت عدد را به علامت اختصاص می دهند . قرار این است که اعداد مثبت را با گذاشتن ۰ و اعداد منفی را با گذاشتن ۱ در محل بیت مزبور نشان دهند .

مثلاً ، رشته بیت های ۰۱۰۰۱ می تواند بعنوان ۹ (دودویی بدون علامت) و یا ۹+ (دودویی علامت دار) در نظر گرفته شود زیرا سمت چپترین بیت ۰ است . رشته بیت های ۱۱۰۰۱ ، هرگاه بعنوان عدد بدون علامت در نظر گرفتنه شود برابر ۲۵ تو بهنگام علامت دار بودن برابر ۹ را نشان می دهد . مکان عدد رقم ۱ وجود دارد که بیانگر منفی بودن عدد و بقیه چهار بیت عدد ۹ را نشان می دهد . معمولاً اگر نوع عدد مشخص باشد هیچگونه اشتباهی در تشخیص وجود نخواهد داشت .

نمایش اعداد علامت دار در آخرین مثال فوق ، نمایش مقدار – علامت نامیده می شود . در ین نامگذاری عدد شامل مقدار و یک نماد (+ یا -) یا یک بیت (۰ یا ۱) برای مشخص نمودن علامت است . این روش مورد استفاده اعداد علامت دار در ریاضیات معمولی است . وقتی که اعمال ریاضی در یک کامپیوتر پیاده سازی می شوند ، بهتر است از روش دیگری بنام سیستم مکمل – علامت برای ارائه اعداد منفی استفاده شود . در این سیستم ، یک عدد منفی بوسیله مکمل آن مشخص می شود . در

حالیکه سیستم مقدار – علامت ، عد را با تغییر علامتش منفی می نماید ، سیستم مکمل – علامت با مکمل سازی ، منفی آن را تهیه می نماید . چون اعداد مثبت همواره با ۰ (مثبت) در سمت چپشان شروع می شوند ، مکمل آنها همیشه با ۱ آغاز خواهند شد ، این نشانگر عدد منفی است . سیستم مکمل – علامت می تواند از مکمل ۱ یا ۲ استفاده نماید . ولی مکمل ۲ مرسوم تر است .

بعنوان مثال ، فرض کنید عدد ۹ بصورت دودویی با هشت بیت نشان داده شده باشد . ۹+ بوسیله یک ۰ در سمت چپ ترین امکان از هشت بیت و بدنبال آن معادل دودویی ۹ ، نشان داده می شود و نتیجه ۰۰۰۰۱۰۰۱ خواهد بود . توجه داشته باشید که تمام هشت بیت باید مقدار داشته باشد ، بنابراین ۰ ها از محل علامت تا اولین ۱ از سمت چپ وارد شده اند . هر چند که فقط یک راه برای نمایش ۹+ وجود دارد ، برای نمایش ۹- با هشت بیت سه روش موجود است :

در نمایش مقدار – علامت ۱۰۰۰۱۰۰۱

در نمایش مکمل۱- علامت ۱۱۱۱۰۱۱۰

در نمایش مکمل۲- علامت ۱۱۱۱۰۱۱۱

در سیستم مقدار – علامت ، ۹- از ۹+ و با تغییر بیت علامت در سمت چپ ترین مکان از ۰ به ۱ حاصل می شود . در سیستم مکمل ۱- علامت ، ۹- را با مکمل کردن تمام بیت های ۹+ از جمله بیت علامت بدست می آوریم . در سیستم مکمل ۲- علامت ، ۹- را از مکمل ۲ عدد مثبت و از جمله بیت علامت بدست می آوریم .

سیستم مقدار – علامت در ریاضی معمولی بکار می رود ، ولی وقتی کامپیوتر بکار رود مشکلاتی به همراه دارد . بنابراین معمولاً درکامپیوتر روش مکمل – علامت بکار گرفته می شود . مکمل ۱ نیز مشکلاتی را ایجاد می نماید و بندرت برای اعمال

ریاضی، بجز در کامپیوتر های قدیمی استفاده می شود . مکمل ۱ برای اعمال منطقی مفید است چون تبدیل ۰ به ۱ و یا به ۱ به ۰ معادل با یک مکمل سازی منطقی است که در فصل بعدی نشان داده خواهد شد .. روش مشابهی به سیستم مکمل ۱- علامت قابل اعمال است و در ان رقم نقلی چرخشی ، همچون اعداد بدون علامت ، نیز منظور می شود .

جمع حسابی

جمع دو عدد در سیستم مقدار – علامت از قوانین معمولی ریاضی تبعیت می نماید . اگر علامتها یکسان باشند ، دو مقدار را به هم اضافه می کنیم تا مجموع با علامت مشترک را بدهد . اگر علامتها مختلف باشند ما مقدار کوچکتر را از بزرگتر کم می کنیم و علامت مقدار را بر می گزنیم مثلاً ،

$$(+25)+(-37)=-(37-25)=-12$$

و بدین ترتیب انجام شده که مقدار کوچکتر ۲۵ از ۳۷ کم شده و علامت ۳۷ بعنوان علامت جواب بکار رفته است . این روند به مقایسه علامتها و سپس اجرای جمع یا تفریق نیاز دارد . روش مشابهی به اعداد دودویی در فرم مقدار – علامت قابل اعمال است . برعکس ، قانون جمع در سیستم مکمل – علامت مقایسه یا تفریقی را احتیاج ندارد بلکه فقط جمع مورد نیاز است .

جمع دو عدد دودویی علامت دار با اعداد منفی که بفرم مکمل ۲ نشان داده شده اند از جمع دو عدد حاصل می شود که بیت علامتشان نیز منظور می گردد . رقم منفی در ابتدا بصورت مکمل ۲ می باشند و حاصل جمع اگر منفی باشد بصورت مکمل ۲ است .

۱۱۱۱۱۰۱۰	-۶	۰۰۰۰۰۱۱۰	+۶
۰۰۰۰۱۱۰۱	+۱۳	۰۰۰۰۱۱۰۱	
+۰۱۳			
۰۰۰۰۰۱۱۱	+۷	۰۰۰۱۰۰۱۱	+۱۹
۱۱۱۱۱۰۱۰	-۶	۰۰۰۰۰۱۱۰	+۶
۱۱۱۱۰۰۱۱	-۱۳	۱۱۱۱۰۰۱۱	-۱۳
۱۱۱۰۱۱۰۱	-۱۹	۱۱۱۱۱۰۰۱	-۷

برای یافتن یک جواب صحیح ، ما باید مطمئن باشیم که برای جا سازی حاصل جمع تعداد کافی بیت وجود دارد . اگر با دو عدد n بیت آغاز کنیم و جمع n+۱ بیت را اشغال کند گوییم سرریز رخ داده است . سرریز کامپیوتر یک مسئله است زیرا تعداد بیت هایی که عدد را نگه می دارند محدود است و اگر جواب به اندازه ۱ واحد از حداکثر مقدار قابل نگهداری در n بیت تجاوز کند قابل جای دهی نخواهد بود .

تفریق حسابی

تفریق دود عدد علامت دار ، وقتی که بصورت مکمل ۲ باشد بسیار ساده است و بصورت زیر بیان می گردد .

مکمل ۲ مفروق منه را بدست آورید (با بیت علامت) و آن را با مفروق (با بیت علامت) جمع کند . رقم نقلی از مکان بیت علامت حذف می گردد .

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

اما تبدیل یک عد مثبت به منفی به سادگی با یافتن مکمل ۲ آن امکان پذیر است . عکس مطلب نیز صحیح است زیرا مکمل یک عدد منفی بفرم مکمل ، یک عد مثبت

تولید می نماید . تفریق ۷+ = (۳-) – (۶-) را ملاحظه کنید . در دودویی با هشت بیت ، این تفریق بصورت ۱۱۱۱۰۰۱۱ – ۱۱۱۱۱۰۱۰ نوشته می شود و عمل تفریق با بدست آوردن مکمل ۲ مفروض منه (۱۳-) بصورت (۱۳+) در می آید . در دودویی برابرست با ۱۰۰۰۰۰۱۱۱ = ۰۰۰۰۱۱۰۱ – ۱۱۱۱۱۰۱۰ . با حذف رقم نقلی نهایی پاسخ صحیح ۰۰۰۰۰۱۱۱ که همان (۷+) است بدست می آید .

۱-۷ کدهای دودویی

یک عدد دودویی n رقمی را می توان با یک مدار که دارای n جزء دودویی است و هر کدام دارای یک سیگنال خروجی معادل ۰ و یا ۱ هستند ، نشان داد . سیستم های دیجیتال نه تنها اعداد دودویی بلکه بسیاری از اجزاء گسسته اطلاعاتی دیگر را نیز نمایش می دهند و روی آنها عمل می کنند . هر عنصر گسسته مستقل اطلاعاتی در میان یک گروه از مقادیر را می توان با استفاده از کد دودویی نشان داد . کدها باید بصورت دودویی باشند زیرا کامپیوتر ها قادر به نگهداری ۰ ها و ۱ ها می باشند .

یک بیت ، طبق تعریف یک رقم دودویی است . وقتی که به همراه یک کد بکار می رود بهتر است که آن را به یک کمیت دودویی برابر با ۰ یا ۱ ها تصور می کنیم . نمایش یک گروه از ۲^n عنصر به صورت کد ، به حداقل n بیت نیاز دارد ، زیرا n بیت را می توان به 2^n طریق مجزا در کنار هم قرار داد . به عنوان مثال هشت عنصر نیازمند یک کد سه بیتی است که هر جزء آن فقط و فقط به یکی از ترکیبات ۰۰۰ ، ۰۰۱ ، ۰۱۰ ، ۰۱۱ ، ۱۰۰ ، ۱۰۱ ، ۱۱۰ و ۱۱۱ نسبت داده می شود . مثالهای فوق نشان می دهند که ترکیبات یک کد n بیتی را می توان با شمارش دودویی از صفر تا $(۲^n-۱)$ به دست آورد . وقتی که تعداد اجزای یک گروه اطلاعاتی دقیقاً معادل توانی از ۲ نباشد تعدادی از ترکیبات کدها را بلااستفاده باقی می گذاریم . ارقام ۰ ، ۱ ، ... ، ۹ در دستگاهی

دهدهی مثالی از چنین گروهی است . چهار بیت می تواند شانزده ترکیب مجزا را به وجود آورد اما از آنجایی که ده رقم را بیشتر نمی خواهیم کد گذاری کنیم شش ترکیب باقی مانده دیگر به کار گرفته نشده وبلا استفاده می ماند .

اگر چه برای کد کردن ۲^n مقدار مشتمل ، مینیمم تعداد بیتها لازم n تاست . ولی مقدار ماکزیمم برای تعداد بیتهای مورد استفاده وجود ندارد . مثلاً ده رقم دهدهی را می توان با ده بیت به این صورت کد کرد که هر رقم دهدهی را به رقم دودویی نسبت بدهیم که ۹ تا صفر و یک ۱ دارد . در این کد گذاری ویپه رقم ۶ با این ترکیب بصورت ۰۰۰۱۰۰۰۰۰۰ نمایش داده می شود .

کدهای دهدهی

کدهای دودویی برای ارقام دهدهی حداقل چهار بیت لازم دارند . از کنار هم قرار دادن چهار بیت یا بیشتر در ده ترکیب مستقل ممکن ، کدهای متعددی می توان به دست اورد . در جدول (۲-۱) تعدادی از این حالات ممکن نشان داده شده است .

کد BCD کدی است که در آن از معادل دودویی اعداد در مبنای ده مستقیماً استفاده می شود . به بیتهای دودویی بر طبق مکانشان می توان وزن یا ارزشی نسبت داد . این روش در کد BCD ، ۱،۲،۴،۸ است . مثلاً کد ۰۱۱۰ برحسب ارزش بیتها نشان دهنده رقم ۶ دهدهی است : چون $0\times8+1\times4+1\times2+0\times1=6$ همچنین می توان ارزشهای منفی به صورت ۱-،۲-، ۴ ، ۸ را به کد دهدهی تخصیص داد . در این حالت ترکیب ۰۱۱۰ ، عدد ۲ تفسیر می شود و بطریق زیر محاسبه می گردد :

$$0\times8+1\times4+1\times(-2)+0\times(-1)=2$$

دو کد وزین دیگر که در جدول نشان داده شده اند ، ۲۴۲۱ و ۵۰۴۳۲۱۰ هستند که دهدهی که در کامپیوتر های قدیمی به کار می رفته کد افزونی ۳- بوده است . این یک کد غیر وزین است ، و از جمع عدد ۳ با مقدار BCD آن به دست می آید .

جدول (۲-۱) کدهای دودویی برای ارقام دهدهی

رقم دهدهی	(BCD) ۸۴۲۱	افزونی -۳	۸۴-۲-۱	۲۴۲۱	دوپنجی ۵۰۴۳۲۱۰
۰	۰۰۰۰	۰۰۱۱	۰۰۰۰	۰۰۰۰	۰۱۰۰۰۰۱
۱	۰۰۰۱	۰۱۰۰	۰۱۱۱	۰۰۰۱	۰۱۰۰۰۱۰
۲	۰۰۱۰	۰۱۰۱	۰۱۱۰	۰۰۱۰	۰۱۰۰۱۰۰
۳	۰۰۱۱	۰۱۱۰	۰۱۰۱	۰۰۱۱	۰۱۰۱۰۰۰
۴	۰۱۰۰	۰۱۱۱	۰۱۰۰	۰۱۰۰	۰۱۱۰۰۰۰
۵	۰۱۰۱	۱۰۰۰	۱۰۱۱	۱۰۱۱	۱۰۰۰۰۰۱
۶	۰۱۱۰	۱۰۰۱	۱۰۱۰	۱۱۰۰	۱۰۰۰۰۱۰
۷	۰۱۱۱	۱۰۱۰	۱۰۰۱	۱۱۰۱	۱۰۰۰۱۰۰
۸	۱۰۰۰	۱۰۱۱	۱۰۰۰	۱۱۱۰	۱۰۰۱۰۰۰
۹	۱۰۰۱	۱۱۰۰	۱۱۱۱	۱۱۱۱	۱۰۱۰۰۰۰

اعداد در کامپیوتر دیجیتال به صورت دودویی و یا به صورت دهدهی توسط یک کد دودویی نمایش داده می شوند . مثلاً وقتی که عدد ۳۹۲ به دودویی تبدیل می شود عدد ۱۱۰۰۰۱۰۱۱ به دست می آید که شامل ۹ رقم دودویی است . همان عدد وقتی به فرم BCD نمایش داده می شود برای هر رقم دهدهی چهار بیت اشغال می گردد که جمعاً دوازد بیت خواهد شد ، یعنی ۰۰۱۱۱۰۰۱۰۱۰۱ .

درک اختلاف تبدیل یک عدد دهدهی به دودویی و کد گذاری دودویی همان عدد دهدهی امر مهمی است . در هر حالت نتیجه نهایی مجموعه ای از بیتها است . کد BCD به عنوان کدی که به دو صورت مورد استفاده قرار می گیرد ، انتخاب شده است . مادامی که عدد بین ۰ الی ۹ باشد اطلاعاتی دودویی ، نتیجه تبدیل مستقیم اعداد فوق به صورت دودویی است ولی اگر عدد بیش از ۹ باشد دیگر این تبدیل

مفهومی ندارد و در این صورت تبدیل و کد گذاری با یکدیگر اختلاف دارند . این مفهوم آنقدر اهمیت دارد که تکرار یک مثال دیگر در مورد آن ارزشمند است . معادل عدد دهدهی ۱۳ به دودویی عدد ۱۱۰۱ است و کد آن در BCD ، ۰۰۰۱۰۰۱۱ می باشد .

از میان پنج کد فهرست شده در جدول (۲-۱) به نظر می رسد که BCD طبیعی ترین کد برای استفاده بوده و در حقیقت معمولیترین آنهاست . کدهای دیگر چهار بیتی یک مشخصه مشترک دارند که در BCD یافت نمی شود . کد افزونی -۳ و ۲،۲،۴،۱ و کد ۱- ، ۲- ، ۴ ، ۸ کدهای خود مکمل هستند ، به این مفهوم که مکمل ۹ عدد دهدهی به سادگی با تبدیل ۰ها به ۱ها و ۱ ها به ۰ ها بدست می آید . مثلاً عدد ۳۹۵ در مد ۱،۲،۴،۲ به شکل ۰۰۱۱۱۱۱۱۱۰۱۱ است . مکمل ۹ این عدد یعنی ۶۰۴ با ۱۱۰۰۰۰۰۰۰۱۰۰ نمایش داده می شودکه به سادگی از جایگزینی ۱ ها با ۰ ها و ۰ ها با ۱ ها به دست می آید . این خاصیت زمانی که اعمال محاسباتی کامپیوتر با اعداد دهدهی (در کد دودویی) صورت می گیرد و عمل تفریق با استفاده از مکمل ۹ انجام می شود ، سودمند است .

کد دو پنجی که در جدول (۲-۱) نشان داده شده است مثالی از یک کد هفت بیتی با خاصیت آشکار سازی خطا است . هر رقم دهدهی ، شامل پنج ۰ و دو ۱ ، که در ستونهای وزین مربوطه جای گرفته اند می باشد . خاصیت آشکار سازی خطای این کد زمانی قابل درک است که بدانیم سیستمهای دیجیتال ۱ و ۰ دودویی را با دو سطح ولتاژ یا جریان مستقل از هم نشان می دهند . در طول انتقال این سطح ولتاژ یا سیگنالها ، از یک محل به محل دیگر ، خطایی ممکن است اتفاق افتد و یک یا چند بیت احتمالاً تغییر ارزش بدهد . یک کدار در مقصد قادر است وجود دو و ۱ یا کمتر در

کد دو پنجی را آشکار کند . اگر ترکیب بیتهای رسیده با ترکیب مجاز در کد یکسان نباشد ، یک خطا محسوب شده و اطلاع داده می شود .

کد های آشکار سازی خطا

اطلاعات دودویی ممکن است از یک مکان به مکان دیگر بکمک وسایل ارتباطی مثل سیمها یا موجهای رادیویی انتقال یابند . هر پارازیت خارجی که وارد وسایل فیزیکی شود ارزش بیتها را از ۰ به ۱ و یا برعکس تغییر می دهد . معمول ترین روش خطایابی ، استفاده از بیت توازن است . یک بیت توازن ، بیتی است اضافی که جزئی از پیام است سبب می شود که تعداد کل ۱ ها در پیان زوج یا فرد گردد یک پیغام چهار بیتی به همراه بیت توازن P در جدول (۳-۱) نشان داده شده است . اگر بیت توازن فرد انتخاب شده باشد P طوری انتخاب می گردد که مجموع ۱ها در پنج بیت فرد باشد و در توازن زوج P طوری انتخاب شده تا مجموع همه ۱ ها زوج باشد .

جدول (۳-۱) بیت توازن

پیام (a)	P (فرد)	پیام (b)	P (زوج)
۰۰۰۰	۱	۰۰۰۰	۰
۰۰۰۱	۰	۰۰۰۱	۱
۰۰۱۰	۰	۰۰۱۰	۱
۰۰۱۱	۱	۰۰۱۱	۰
۰۱۰۰	۰	۰۱۰۰	۱
۰۱۰۱	۱	۰۱۰۱	۰
۰۱۱۰	۱	۰۱۱۰	۰
۰۱۱۱	۰	۰۱۱۱	۱
۱۰۰۰	۰	۱۰۰۰	۱
۱۰۰۱	۱	۱۰۰۱	۰
۱۰۱۰	۱	۱۰۱۰	۰
۱۰۱۱	۰	۱۰۱۱	۱
۱۱۰۰	۱	۱۱۰۰	۰
۱۱۰۱	۰	۱۱۰۱	۱
۱۱۱۰	۰	۱۱۱۰	۱
۱۱۱۱	۱	۱۱۱۱	۰

نحوه خطایابی بدون شرح است . یک بیت توازن زوج در مبدا برای هر پیام تولید می شود . بیت توازن همراه با پیام به سمت مقصد ارسال می شود . توزان در مقصد چک می گردد . زوج نبودن داده رسیده به معنی این است که حداقل یک بیت در ضمن انتقال تعویض شده است . این روش قادر است هر ترکیب فردی از تعداد خطا مانند تغییر یک ، سه و ... بیت را در هر پیام انتقال یافته مشخص نماید . روش های تشخیص خطای اضافی دیگری برای یافتن خطاهای زوج لازم است .

کد گری (انعکاسی)

سیستمهای دیجیتال فقط برای پردازش داده های گسسته طراحی می شوند . بسیاری از دستگاههای فیزیکی داده خروجی پیوسته تولید می کنند .. اطلاعات پیوسته یا آنالوگ بوسیله مبدل آنالوگ به دیجیتال به فرم دیجیتال تبدیل می شوند . گاهی اوقات استفاده از کد گری نشان داده شده در جدول (۴-۱) ، جهت نمایش داده های دیجیتال تبدیل شده از داده های آنالوگ معمولتر است .

مزیت کد گری نسبت به اعداد دودویی محض این است که وقتی از یک عدد به عدد بعدی می رویم فقط یک مزیت بیت تغییر می کند . مثلاً در رفتن از ۷ به ۸ ، کد گری از ۰۱۰۰ به ۱۱۰۰ تغییر می یابد . دیده می شود که فقط سمت چپ ترین بیت از ۰ به ۱ تغییر یافته و سه بیت بقیه یکسانند . وقتی مطلب را با اعداد دودویی مقایسه کنیم ، تغییر از ۷ به ۸ سبب تغییر هر چهار بیت ، یعنی از ۰۱۱۱ به ۱۰۰۰ می گردد .

کد گری در کاربردهایی که رشته معمولی اعداد دودویی امکان تولید خطا دارند بکار می رود . بهنگام تغییر از ۰۱۱۱ به ۱۰۰۰ ، اگر تغییر سمت راست ترین بیت از سه بیت دیگر بیشتر طول بکشد یک عدد میانه ای مانند ۱۰۰۱ تولید می شود . کد گری

این مشکل را حذف می نماید زیرا بهنگام انتقال بین دو عدد فقط یک تغیر رخ می دهد .

نمونه ای از کاربرد کد گری هنگامی است که داده آنالوگ بوسیله تغییر پیوسته شفت نمایش داده می شود . دور شفت به قطعاتی تقسیم شده ، و به هر قطعه عددی تخصیص یافته است . اگر قطعات مجاور بوسیله کد گری مرتبط شوند ، ابهام در تفکیک دو ناحیه مجاور که در حال احساس شدن است کاهش می یابد .

جدول (۴-۱) کد گری ۴ بیتی

معادل دهدهی	کد گری
۰	۰۰۰۰
۱	۰۰۰۱
۲	۰۰۱۱
۳	۰۰۱۰
۴	۰۱۱۰
۵	۰۱۱۱
۶	۰۱۰۱
۷	۰۱۰۰
۸	۱۱۰۰
۹	۱۱۰۱
۱۰	۱۱۱۱
۱۱	۱۱۱۰
۱۲	۱۰۱۰
۱۳	۱۰۱۱
۱۴	۱۰۰۱
۱۵	۱۰۰۰

کد های ASCII

در بسیاری از کاربردهای کامپیوتر های دیجیتال نه تنها نیاز به دستکاری روی داده های عددی بلکه روی حروف نیز می باشد . یک کاراکتر الفبا عددی عبارت از یک کد دودویی مربوط به عنصری از یک مجموعه که شامل ۱۰ رقم دهدهی ، ۲۶ حروف الفبا و تعداد معینی از علائم مخصوص است . چنین مجموعه ای بین ۳۶ تا ۶۴ عنصر برای

حروف بزرگ و یا بین ۶۴ تا ۱۲۸ عنصر با حروف بالا و پایین هر کلید دارد . در حالت اول به شش بیت و در حالت دوم به هفت بیت نیاز است .

کد دودویی استاندارد برای کاراکتر های الفبا عددی ASCII است . این کد از هفت بیت برای کد نمودن ۱۲۸ کاراکتر استفاده می کند . هفت بیت با b_1 تا b_7 مشخص شده اند که b_7 با ارزشترین بیت را تشکیل می دهد . مثلاً ، حرف A در ASCII بصورت ۱۰۰۰۰۰۱ (ستون ۱۰۰ سطر ۰۰۰۱) می باشد . کد ASCII دارای ۹۴ کد شامل ۲۶ کاراکتر مربوطه به حروف بزرگ (A تا Z) ، ۲۶ کاراکتر حروف کوچک (a تا z) ، ۱۰ عدد (۰ تا ۹) و ۳۲ کاراکتر مخصوص چاپ نشدنی مانند % ، * و $ است .

کد همینگ

جهت تشخیص و تصحیح خطا بکار می رود. اگر M پیام ارسالی m بیتی باشد.

$$M : n_1 n_2 n_3 n_4$$

k تعداد بیتهای توازن که اضافه می شود و از رابطه زیر تبعیت می کند. $k + m \le 2^k - 1$

بیتهای توزان درمحلهای 2^n $(2^0\ 2^1\ 2^2 .....)$ قرار می گیرد . $p_1 p_2 m_3 p_4 m_5 m_6 m_7$

بیتهای توازن بدینصورت بدست می آیند:

$$P_1 = m_3 \oplus m_5 \oplus m_7$$
$$P_2 = m_3 \oplus m_6 \oplus m_7$$
$$P_4 = m_5 \oplus m_6 \oplus m_7$$

$m_3 \to 0\ \ 1\ \ 1$
$m_5 \to 1\ \ 0\ \ 1$
$m_6 \to 1\ \ 1\ \ 0$
$m_7 \to 0\ \ 1\ \ 1$

محدوده بیت های پیام	تعداد بیت های توزان k
2-4	3
5-11	4
12-26	5
27-57	6

$M: 1011 \qquad \Rightarrow p_1 p_2 1 p_4 011$

$P_1 = m_3 \oplus m_5 \oplus m_7 = 1 \oplus 0 \oplus 1 = 0$

$P_2 = m_3 \oplus m_6 \oplus m_7 = 1 \oplus 1 \oplus 1 = 1$

$P_4 = m_5 \oplus m_6 \oplus m_7 = 0 \oplus 1 \oplus 1 = 0$

تشخیص خطا:

کد خطایی از روی اطلاعات دریافتی ایجاد می شود.

$C: C_4 C_2 C_1$

$C_1 = P_1 \oplus m_3 \oplus m_5 \oplus m_7$

$C_2 = P_2 \oplus m_3 \oplus m_6 \oplus m_7$

$C_4 = P_4 \oplus m_5 \oplus m_6 \oplus m_7$

اگر C=0 خطایی رخ نداده است اگر $C \neq 0$ خطا رخ داده و مقداری c مکان خطا خواهد بود .

$0\ 1\ 1\ 0\ 0\ 1\ 1 \rightarrow 0\ 0\ 1\ 0\ 0\ 1\ 1$

$C_1 = 0$

$C_2 = 1 \quad \Rightarrow C = 0\ 1\ 0 = 2$

$C_3 = 0$

تشخیص و تصحیح یك خطا

$m = 5 \qquad k = 4$

$p_1\ p_2\ m_3\ p_4\ m_5\ m_6\ m_7\ p_8\ m_9$

$C_1 = P_1 \oplus m_3 \oplus m_5 \oplus m_7 \oplus m_9$

$C_2 = P_2 \oplus m_3 \oplus m_6 \oplus m_7$

$C_4 = P_4 \oplus m_5 \oplus m_6 \oplus m_7$

$C_8 = m_9$

تعریف minimum distance

حداقل فاصله عبارتست از حداقل تعداد بیت هایی که باید در یک کد تغییر یابد تا کد مجاز دیگری از همان سیستم کد گذاری به دست آید .

۱ = کد گری M . D.

۱ = کد 2421 M . D.

در صورتیکه MD = 3 باشد و یک بیت دچار خطا شود هم قادر به تشخیص خطا و هم توانایی تصحیح آن را داریم .

$$M = C + d + 1$$

d : تعداد بیت هاي خطاي قابل تشخیص

c : تعداد بیت هاي خطاي قابل تصحیح

به کد همینگ بیت توازن دیگری اضافه می کند که با تمام بیتهای توزان زوج برقراری می کند .

M	d	c
1	0	0
2	1	0
3	2	0
	1	1
4	3	0
	2	1
5	4	0
	3	1
	2	2

$P_1 p_2 m_3 p_4 m_5 m_6 m_7 p_8 m_9 , p_0$

در مقصد که $C : C_8 C_4 C_2 C_1$ و توزان دیگری با نام p را ایجاد می کنند .

$$P = P_1 \oplus p_2 \oplus \dots m_9 \oplus p_0$$

خطایی رخ نداده است $C = 0$, $P = 0$

یک خطا رخ داده و قابل تصحیح $C \neq 0$, $P = 1$

دو خطا رخ داده و فقط قابل تشخیص $C \neq 0$, $P = 0$

خود P دچار خطا شده است $C = 0$, $P = 1$

۲-۱ تعریف اصولی جبر بول

در سال ۱۸۵۴ جورج بول روش اصولی برای منطق معرفی نمود و بدین طریق یک سیستم جبری را پایه ریزی کرد که امروز جبر بول نامیده می شود . برای تعریف مستدل جبر بول ، ما اصول فرموله شده بوسیله هانتینگتون در ۱۹۰۴ را به کار خواهیم برد . این اصول برای تعریف جبر بول منحصر به فرد نیستند و اصول دیگری نیز در آن بکار رفته اند .

جبر بول یک ساختار جبری است که با عناصر مجموعه B همراه با دو عملگر (+) و (.) تعریف شده و دارای اصول زیر (اصول هانتینگتون) باشد :

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

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

۲- (a) عنصر خنثی در مجموعه برای (+) برابر با ۰ باشد .

$$x+0=0+x=x$$

۳- (b) عنصر خنثی در مجموعه برای (.) برابر ۱ باشد .

$$x.1=1.x=x$$

۴- (a) مجموعه نسبت به (+) دارای خاصیت جابجایی باشد .

$$x+y=y+x$$

(b) مجوعه نسبت به (.) دارای خاصیت جابجایی باشد :

$$x.y=y..x$$

۴- (a) (.) روی (+) دارای خاصیت پخشی است . $x.(y+z)=(x.y)+(x.z)$

(b) (+) روی (.) دارای خاصیت پخشی است . $x+(y.z)=(x+y).(x+z)$

۵- به ازای هر عنصر $x \in B$ عنصری مثل $x \in B$ وجود داشته باشد (این عنصر مکمل خوانده می شود) بطوری که :

$$x + x' = 1 \qquad (a)$$

$$x.x' = 0 \qquad (b)$$

۶- حداقل دو عنصر مانند $x, y \in B$ موجود باشند بطوریکه : $x \neq y$ ازمقایسه جبر بول با ریاضیات جبری معمولی (میدان اعداد حقیقی) اختلافات زیر ملاحظه می گردند :

۱- اصول هانتینگتون شامل اصل اشتراک پذیری نیستند . این قانون برای جبر بول نیز وجود دارد و می توان آن را برای هر دو عملگر از سایر اصول بدست آورد .

۲- قانون توزیع پذیری (+) و (.) اختلاف بعدی است . رابطه :

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

برای جبر بول معتبر ولی برای جبر معمولی قابل قبول نیست .

۳- جبر بول معکوس جمع و ضرب را ندارد ، بنابراین تفریق و تقسیم مفهوم نخواهند داشت .

۴- اصل ۵ عملگرد دیگری بنام مکمل را معرفی می نماید که در جبر معمولی وجود ندارد .

۵- جبر معمولی در مورد اعداد حقیقی است که بی نهایت عنصر را شامل می شود . جبر بول با عناصری از مجموعه B که البته تا کنون معرفی نشده اند سرو کار داشت ولی درجبر بول دو ارزشی یا دو مقداری که در زیر تعریف شده ، B یک مجموعه دو عنصری است که این دو عنصر ۰ و ۱ می باشند .

۲-۲ قضیه های اصلی و خواص جبر بول

اصول هانتیگتون بصورت جفت جفت لیست و با قسمت های (a) و (b) مشخص شد . هر یک از این دو را می توان از دیگری بدست آورد بشرط اینکه عملگرها و نیز عناصر خنثی تعویض شوند. این خاصیت مهم درجبر بول به اصل دوگانگی معروف است و بیان می دارد که هر عبارت جبری منتجه از اصول جبر بول حتی با تعویض عملگر ها و عناصر خنثی باز هم معتبر می باشد . در جبر بول دو ارزشی عناصر خنثی و خود عناصر محموعه B یکسانند : ۱و۰ اصل دوگانگی کاربردهای فراوانی دارد . اگر دو گان یک عبارت جبری ، مورد نظر باشد تنها کافی است عملگرهای OR و AND تعویض شده و ۰ ها به ۱ ها و همچنین ۱ها به ۰ ها تبدیل گردند .

تئوری های اساسی

جدول (۱-۲) شش تئوری و چهار اصل از جبر بول را در بر دارد . در سمت چپ روابط ، شماره اصول بکار رفته نوشته شده است .

جدول (۱-۲) اصول و قضایای جبر بول)

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

۲-۳ توابع بول

یک متغییر دودویی می تواند یکی از دو مقدار ۰ یا ۱ را اختیار کند . یک تابع بول عبارتی است که از متغیرهای دودویی ، عملگر های OR ، AND ، NOT پرانتز ها و

علامت تساوی تشکیل شده است . به ازای مقادیر مفروضی از متغیرها تابع فقط می تواند ۰ یا ۱ باشد . مثلاً تابع بول $F_1 = xyz'$ را در نظر بگیرید. تابع f_1 برابر با ۱ است بشرطی که x=1 ، y=1 و z´=1 باشد ، در غیر این صورت $F_1 = 0$ خواهد بود . برای نمایش یک تابع بفرم جدول درستی نیاز به 2^n ترکیب از ۱ ها و ۰ ها مربوط به n به تغییر دودویی و ستونی یکه در آن مقدار تابع برابر ۰ یا ۱ است ، داریم . از جدول (۲-۲) دیده می شود وکه برای سه متغیر ۸ حالت جدا می توان در نظر گرفت . در جدول (۲-۲) ، چهار ردیف آخر مساوی ۱ و xy در ردیفهای ۰۰۱ و ۱۰۱ برابر ۰۱ است . ترکیب آخری دلالت بر x=۱ نیز دارد .بنابراین برای F_2=۱ پنج حالت وجود دارد .

جدول (۲-۲) جدول درستی برای

$F_1 = xyz \quad F_2 = x + y'z \;,\; F_3 = x'y'z + x'yz + xy \quad , \; F_4 = xy' + xz$

x	y	Z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	0	0
1	1	1	1	1	0	0

عملیات جبری

لیترال ، یک متغیر با پریم یا بدون پریم است . وقتی که یک تابع بوسیله گیت منطقی پیاده شود هر لیترال در تابع معروف یک ورودی به یک گیت و هر جمله منطقی نیز توسط یک گیت ساخته می شود . می نیمم کردن تعداد متغیرها و جملات ، نتیجه اش ساخت دستگاهی با قطعات کمتر است .

البته همیشه ممکن نیست که هر دو را با هم کاهش داد . فعلاً ما می نیمم سازی را فقط به متغیرها محدود میکنیم تعداد متغیر ها در تابع بول می تواند با یک سری اعمال جبری می نیمم گردد ، متاسفانه قوانین مشخص و معینی که تضمین کننده فرم نهایی باشد وجود ندارد . تنها روش موجود سعی در کاهش مدار و تداوم این عمل با استفاده از اصول اولیه ، تئوری های اصلی و هر روش عملیاتی دیگری ، که ضمن عمل با آنها حاصل می گردد ، می باشد .

مثال ۱-۲ : تابع بولی زیر را از نظر تعداد متغیرها می نیمم کنید .

$$1.\ x + x'y = (x + x')(x + y) = 1.(x + y) = x + y$$

$$2.\ x(x' + y) = xx' + xy = 0 + xy = xy$$

$$3.\ x'y'z + x'yz + yz = x'z(y' + y) + xy' = x'z + xy'$$

$$4.\ xy + x'z + yz = xy + x'z + yz(x + x')$$
$$= xy + x'z + xyz + x'yz$$
$$= xy(1 + z) + x'z(1 + y)$$
$$= xy + x'z$$

$$5.\ (x + y)(x' + z)(y + z) = (x + y)(x' + z)$$

با توجه به دو گانه بودن تابع ۴

تابع ۱ و ۲ دوگان یکدیگرند و عبارت دوگانی رادر مراحل مربوط به خود بکار می برند . تابع ۳ هم ارزی توابع F_4, F_3 بحث شده در قبل را نشان می دهد . چهارمین تابع روشنگر این واقعیت است که افزایش در تعداد متفیرها گاهی اوقات سبب ساده تر

شدن عبارت نهایی می گردد . تابع ۵۴ مستقیماً ساده نشده است ولی با استفاده از دوگان مراحل رمربوط به تابع ۴ می تواند حاصل گردد .

مکمل یک تابع

مکمل یک تابع F تابعی است مانند F َ که با تعویض ۰ ها به ۱ها و ۱ ها به ۰ ها در مقدار F حاصل می گردد . مکمل یک تابع ممکن است با استفاده از تئوری دمورگان نیز بدست می آید . زوج قوانین دمورگان برای دو متغییر در جدول (۱-۲) لیست شده اند . تئوری های دمورگان قابل تعمیم برای سه متغیر و یا بیشتر از آن نیز هستند . فرم سه متغیر تئوری اول دمورگان در زیر آمده است . اصول و تئوری های بکار رفته همان همایی هستند که در جدول (۱-۲) آورده شده اند .

با فرض B + C = A $(A+B+C)'=(A+X)'$

با توجه به تئوری ۵- (a) دمورگان $=A'X'$

با جایگزینی B + C = X $=A'.(B+C)'$

با توجه به تئوری ۴- (a) شرکت پذیری $=A'B'C'$

تئوری های دمورگان برای هر تعداد از متغیر ها ابتد به شکل دو متغیره در آمده و سپس با جایگزینی های متوالی ، مشابه با آنچه در فوق دیده شد ، نتیجه نهایی حاصل می گردد .

این تئوریها می تواند به صورت زیر عمومیت داده شوند .

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

فرم های کلی تئوری دمورگان بیان می کند که مکمل هر تابع با تعویض عملگرهای AND و OR و مکمل نمودن هر متغیر حاصل می شود .

مثال ۲-۲ : مکمل توابع $F_1 = x'yz' + x'y'z$, $F_2 = x(y'z' + yz)$ را بدست آورید .

تئوری دمورگان را هر چند بار که لازم باشد بکار ببرید . مکمل ها بفرم زیر حاصل می گردند

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

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

روش ساده تری برای بدست آوردن مکمل یک تابع این است که ابتدا دوگان آنرا بدست آورده و سپس متغیرهایش را مکمل نماییم . این روش با توجه به فرم کلی تئوری دمورگان نتیجه می شود . بخاطر داشته باشید که دوگان یک تابع با تبدیل عملگر AND و OR وتبدیل ۱ ها و ۰ ها به یکدیگر بدست می آید .

مثال ۲-۳ : مکمل های توابع F_1 , F_2 مثال ۲-۲ را باتوجه به دوگان آنها و مکمل کردن هر متغیر بدست آورد .

1. $F_1 x'yz' + x'y'z$

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

پس از مکمل کردن هر متغیر داریم $(x + y' + z)(x + y + z') = F_1'$

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

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

پس از مکمل کردن هر متغیر داریم $x' + (y + z)(y' + z') = F_2'$

۲-۴ حالات متعارف و استاندارد

یک متغیر دودویی ممکن است بفرم معمولی (x) یا مکملش ($\bar{x}$) ظاهر شود . حال فرض کنیم که متغیرهای دودویی x و y بوسیله عملگر AND با یکدیگر ترکیب شوند . چون هر متغیر ممکن است به هر یک از دو شکل فوق ظاهر گردد چهار ترکیب برای آن دو متغیر وجود دارند $xy', x'y, x'y'$ و $xy`$. هر یک از این چهار جمله نشان دهنده یک ناحیه در دیاگرام ون ، شکل (۲-۱) بوده و مینترم نامیده می شود . بروشی مشابه ، n متغییر می تواند روشی مشابه یا آنچه در جدول (۲-۳) برای سه متغیر حاصل شده بدست آیند . اعداد دودویی از صفر تا 2^n-1 برای n متغیر در زیر ستون متغیر ها در جدول نوشته می شوند . هر مینترم از اجزای عملگر AND روی n متغیر بدست می آید و هر متغیر در آن با مقدار ۰ با علامت پریم و با مقدار ۱ بدون پریم خواهد بود . سمبل مینترم نیز درجدول بفرم m_j آورده شده است که j معادل دهدهی جمله مربوطه می باشد .

جدول (۲-۳) مینترم و ماکسترم ها برای سه متغییر دودویی

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

بطریق مشابهی n متغیر در یک جمله OR ، که هر یک می توانند با پریم و یا بدون پریم باشند ، 2^n ترکیب ممکن را ایجاد می نمایند که هر یک از آنها ماکسترم نامیده می شود . هشت جمله ماکسترم مربوط به سه متغیر با سمبل آنها درجدول (۳-۲) لیست شده اند . هر 2^n جمله ماکسترم برای n متغیر مشابهی تعیین می شوند . هر ماکسترم از یک جمله OR مربوط به n متغیر دارای متغیر های بدون پریم است بشرطی که آن متغیرها ۰ باشند ولی هر گاه مقدار متغیر ۱ باشد در اینصورت آن متغیر پریم دار نمایش داده می شود . توجه داشته باشید که هر جمله ماکسترم مکمل مینترم مربوطه اش می باشد و بالعکس .

یک تابع بول می تواند با استفاده از جدول درستی بفرم جبری با در نظر گرفتن مینترم هایی که تابع به ازای آنها برابر ۱ است و اجرای عملگر OR روی آنها تشکیل گردد . مثلاً تابع F_1 در جدول (۴-۲) بدین طریق معین می شود که ۰۰۱ و۱۰۰ و ۱۱۱ را بفرم $xy'z, x'y'z$ و xyz نشان داده و سپس با یکدیگر ترکیب کنیم . چون هر یک از این مینترم ها برابر ۱ است باید رابطه زیر را داشته باشیم :

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

بطور مشابه بسادگی می توان نشان داد که :

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

این مثالها نشان دهده یک خاصیت مهم جبر بول می باشند که عبارتست از : هر تابع بول می تواند بصورت مجموع مینترم ها (در اینجا بمعنی OR است) بیان شود . حال مکمل یک تابع بول را در نظر بگیرد .این مکمل را می توان با استفاده از جدول وبکارگیری جملات مینترم که در جدول برای تابع ۰ هستند و اعمال عملگرد OR روی آنها بوجود آورد . لذا مکمل تابع f_1 برابر خواهد بود با .

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

اگر ما مکمل f_1 را پیدا کنیم نتیجه همان تابع f_1 خواهد شد .

$$f_1 = (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z)$$
$$= M_0.M_2.M_3.M_5.M_6$$

بطور مشابه عبارت مربوط به f_2 را با توجه به جدول می توان نوشت :

$$f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$$
$$= M_0.M_1.M_2.M_4$$

این مثالها بیانکر دومین خاصیت مهم جبر بول می باشند . یعنی هر تابع می تواند بصورت حاصلضرب ماکسترم ها (حاصلضرب به معنی اعمال عملگرد AND می باشد) نوشته شود . روش بدست آوردن مستقیم حاصلضرب ماکسترم ها با استفاده از جدول ربطریق زیر است : ابتدا جملات ماسکترمی که از ترکیب متغیرها تشکیل شده و برای تابع تولید ۰ می نماید انتخاب شده و سپس با اجرای عملگر AND روی تمام آنها می توان به نتیجه مورد نظر رسید . هر گاه توابع بول بصورت مجموعمینترم ها یا حاصلضرب ماکسترم ها در آیند گویند به شکل متعارف می باشند .

مجموع مینترم ها

قبلاً گفته شده که برای n متغیر 2^n مینترم مستقل بدست آورده و هر تابع بول را میتوان بصورت محموع آنها بیان کرد . تابع بول از مجموع مینترم هایی که مقدارشان در جدول درستی برابر ۱ است تشکیل می گردد. چون مقدار هر مینترم می تواند ۱ یا ۰ باشد و نیز 2^n . گاهی اوقات بهتر است که تابع را بصورت مجموع مینتم ها نشان داد . چنانچه تابع به این شکل نباشد می توان آن را با اجرای اعمال زیر بفرم مورد نظر در آورد . ابتدا مجموعه جملات AND شده را بدست می آوریم و سپس جملات را از

نظر وجود کلیه متغیر ها مورد بازرسی قرار می دهیم . در صورت عدم وجود برخی متغیرها ، باید آنها رادر عباراتی مانندَ x+x و غیره AND کرد . که x یکی از متغیرهایی است که در جمله وجود ندارد . مثال زیر مطلب را روشن میکند :

مثال ۴-۲- : تابع $F = A + B'C$ را بصورت مجوع مینترم نشان دهید .

تابع دارای سه متغیر A,B,C می باشد . در اولین جمله دو متغیر وجود ندارد ، بنابراین

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

هنوز هم یک متغیر کسر است .

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

جمله دوم $B'C$ یک متغیر کسر دارد .

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

از ترکیب نتایج فوق داریم

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

از طرفی $AB'C$ دوباره تکرار شده است و بر طبق تئوری ۱ ، $(x = x + x)$ می توان یکی از آنها را حذف کرد . با مرتب نمودن مینترم ها بترتیب صعودی چنین نتیجه می شود .

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

هنگامی که تابع بول بفرم مجموع مینترم ها است مناسب تر است تا آن بفرم خلاصه زیر نشان دهیم .

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

سمبل $\sum$ به معنی اجرای عملگر OR روی جملات است . حروفی که در داخل پرانتز قرار دارند لیست متغیرهای بکار رفته را بهنگام تشکیل جملات مینترم و جمع آنها معین می کنند . روش دیگری برای تشکیل مینترم های تابع بول تهیه جدول درستی تابع مستقیماً از عبارت جبری است که از روی آن مینترم ها خوانده می شوند . تابع بول مثال ۴-۲ را در نظر بگیرد :

$$F=A+B'C$$

جدول درستی شکب (۵-۲) مستقیماً از عبارت جبری با هشت ترکیب دودویی متغیر های A , B , C حاصل می شود که برای مینترم هایی که در آنها $A=1$, $BC=01$ باشد ۱ قرار می دهیم . سپس با توجه به حجدول درستی پنج مینترم تابعی را که ۱،۴،۵،۶،۷ می باشند می خوانیم.

ضرب ماکسترم ها

هر یک از 2^{2^n} تابع متشکل از n متغیر را همچنین می توان بصورت حاصلضرب ماکسترم ها بیان داشت . برای چنین فرمی باید اول جمله های OR را تشکیل داد . این عمل را می توان با استفاده از قانون توزیع ÷ذیری $x+yz=(x+y)(x+z)$ نیز انجام داد . سپس هر متغیر غایب در هر جمله OR با xx ، OR می شود . این روش با مثال زیر واضحتر خواهد شد :

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

A	B	C	F
۰	۰	۰	۰
۰	۰	۱	۱
۰	۱	۰	۰
۰	۱	۱	۰
۱	۰	۰	۱
۱	۰	۱	۱
۱	۱	۰	۱
۱	۱	۱	۱

مثال ۵-۲ : تابع $F = xy + x'z$ را بصورت حاصلضرب ماکسترم بنویسید :

ابتدا با استفاده از قانون توزیع پذیری تابع را به صورت جملا OR در می آوریم :

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

تابع دارای سه متغیر z,y,x است . هر جمله OR فاقد یک متغیر است . بنابراین .

$$\begin{aligned} x' + y &= x' + y + zz' = (x' + y + z)(x' + y + z') \\ x + z &= x + z + yy' = (x + y + z)(x6y' + z) \\ y + z &= -y + z + xx' = (x + y + z)(x' + y + z) \end{aligned}$$

با ترکیب عبارت فوق و حذف آنهایی که بیش از یکبار تکرار شده اند خواهیم داشت :

$$\begin{aligned} F &= (x + y + z)(x + y' + z)(x' + y + z)(x' + y + z') \\ &= M_0 M_2 M_4 M_5 \end{aligned}$$

روش مناسب تری برای نمایش تابع بقرار زیر است :

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

سمبل ضرب ، $\prod$ بیانگر حاصلضرب ماکسترم ها می باشد و اعداد ، شماره جملات ماکسترم را مشخص می سازد .

تبدیل فرمهای متعارف به یکدیگر

مکمل یک تابع که بصورت مجموع منیترم ها نشان داده شده برابر است با مجموع مینترم هایی که در فرم اصلی تابع وجود ندارد . زیرا تابع اصلی از جملات مینترمی تشکیل شده که تابع را برابر ١ می نماید ، در حالیکه مکمل آن تابع به ازای جملاتی برابر ١ است که تابع اصلی به ازای آنها ٠ می باشد . بعنوان مثال تابع زیر را در نظر بگیرید :

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

مکل این تابع به شکل زیر است :

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

حال ، اگر مکمل `F را با توجه به تئوری دمورگان بدست آوریم فرم جدیدی برای F بدست می آید.

$$F=m_0+m_2+m_3)=m_0.m_2.m_3=M_0M_2M_3=\prod(0,2,3)$$

آخرین تبدیل در رابطه فوق نتیجه تعاریف جدول (٣-٢) می باشد . با توجه به جدول درستی رابطه زیر مسلم است .

$$m'_j=M_j$$

یعنی ، جمله ماکسترم با اندیس j مکمل جمله مینترم با همان اندیس است و بالعکس .

آخرین مثال ، تبدیل یک تابع بصورت مجموع مینترم ها بیان شده به معادل آن که بصورت حاصلضرب ماکسترم ها است را بیان می دارد . بحث مشابهی نشان می دهد که تبدیل حاصلضرب ماکسترم ها به مجموع مینترم ها بطریق فوق است . حال یک روش کلی را برای تبدیل بیان می کنیم :

برای تبدیل یک فرم متعارف به دیگری سمبل های $\sum$, $\prod$ را با یکدیگر عوض نموده و جملاتی که در تابع اصلی وجود ندارد را نیز لیست می نماییم . برای یافتن جملات گم شده باید بیاد بیاوریم که تعداد کل جملات 2^n است که در آن n تعداد متغیرها در تابع است .

یک تابع بول بفرم عبارت جبری بوسیله جدول درستی و روش تبدیل متعارف قابل تبدیل به ضرب ماکسترم ها است . مثلاً عبارت بول زیر را در نظر بگیرید :

$$F = xy + x'z$$

ابتدا جدول درستی را بدست می آوریم ،شکل (۶-۲) . ۱ های زیر ستون F از ترکیب متغیرها با x=11 و xz=01 حاصل می شود مینترم های تابع از روی جدول درستی عبارتند از ۱،۳،۶،۷ . تابع بر حسب مینترم ها برابرست با

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

چون جمعاً هشت مینترم یا ماکسترم در یک تابع سه متغیره وجود دارد ، ما جملات غیر موجود در فوق را می یابیم که عبارتند از ۰ ، ۲ ، ۴ و ۵ . تابع بر حسب ضرب ماکسترم ها چنین خواهد شد .

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

این همان مثالی است که در مثال ۵-۲ دیدیم .

جدول (٢-۶) جدول درستی برای $F = xy + x'z$

x	y	z	f
0	0	0	0
0	0	1	1
۰	١	۰	۰
۰	١	١	١
١	۰	۰	۰
١	۰	١	۰
١	١	۰	١
١	١	١	١

فرم های استاندارد

دو فرم متعارف جبر بول ، فرم هایی ابتدایی هستند که هر کس می تواند با توجه به جدول درستی به آنها دسترسی پیدا کند . این فرم ها معمولاً دارای حداقل متغیرها نیستند ، زیرا هر مینترم یا ماکسترم بایستی بنا به تعریف دارای تمام متغیرها اعم از مکمل و غیر مکمل باشند . راه دیگری برای بیان تابع بول ، فرم استاندارد است . در این فرم ، جمله هایی که تابع را تشکیل می دهند ممکن است یک یا دو یا هر تعدادی از متغیرها را داراباشند .دو نوع فرم استاندارد وجود دارد . یکی جمع حاصلضرب ها و دیگری ضرب حاصل جمع ها .

جمع حاصلضرب ها ، یک عبارت بول است که شامل جملات AND (با نام جملات حاصلضرب)از یک یا چندمتغیر می باشد .کلمه جمع دراینجا به معنی عملگر OR روی این جملات است .

مثالی از این نوع بقرار زیر می باشد :

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

عبارت دارای سه جمله حاصلضرب از یک ، دو و سه متغیر است . جمع آنها در واقع اجرای عمل OR است که جمع نامیده می شوند . هر جمله هر تعداد متغیر را ممکن

است دارا باشد .ضرب بیانگر عملگر AND روی آنها است . مثالی از یک تابع که بصورت ضرب حاصل جمع ها بیان شده عبارتست از :

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

این عبارت به ترتیب دارای سه جمله ، با یک ، دو چهار متغیر است . ضرب آنها در واقع اجرای عمل AND می باشد . کاربرد کلمه ضرب و جمع بیانگر شباهت AND با ضرب و عملگر OR با جمع در حساب می باشد .

یک تابع بول ممکن است بفرم غیر استاندارد نیز بیان شود . بعنوان مثال تابع :

$$F_3 = (AB+CD)(A'B'+C'D')$$

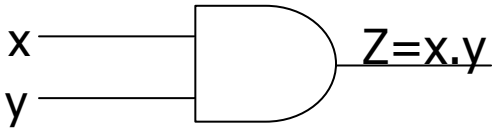
نه بشکل جمع حاصلضرب ها و نه بشکل ضرب حاصل جمع ها است . البته می توان با استفاده از قانون توزیع پذیری آن را بفرم استاندارد در آورد :

$$F_3 = A'B'CD+ABC'D'$$

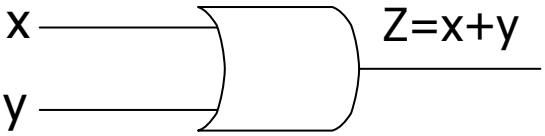
۲-۵ گیت های منطقی دیجیتال

مدارهای دیجیتال الکترونیکی ، مدارهای منطقی نیز نامیده می شوند . زیرا اینگونه مدارهای در مقابل ورودی مناسبی ، تولید کننده یک سری اعمال منطقی می باشند . هر گونه اطلاعات محاسباتی یا کنترلی مورد نظر را می توان با عبور سیگنال های دودویی از میان دسته های متفاوت مدارهای منطقی مورد استفاده قرار داد ، که هر سیگنال نشان دهنده یک متغیر بوده و یک بیت از اطلاعات راحمل می کند . مدارهای منطقی که اعمال منطقی AND و OR و NOT را اجرا می کند به همراه سمبل های مربوطه در شکل (۱-۲) نشان داده شده اند . این مدارها که گیت نامیده می شوند بلوکهای سخت افزاری هستند که با ورودی منطقی مناسبی در خروجی

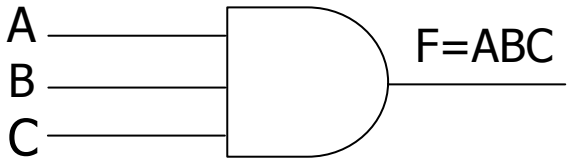
خود ۰ یا ۱ منطقی تولید می کنند . توجه کنید که چهار نام مختلف برای این مدارها بکار رفته است . مدارهای دیجیتال ، مدارهای سوئیچینگ ، مدارهای منطقی و گیت ها . همه این اساس بطور گسترده ای استفاده میشوند ولی بهتر است ما این مدارهای را AND و OR و NOT گاهی مدار وارونگر یا معکوس کننده نیز نامیده می شود زیرا سیگنال دودویی را معکوس می کند .



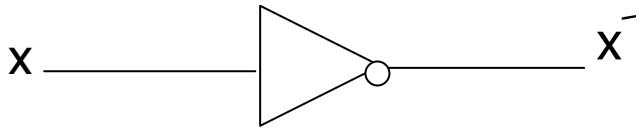
(الف) گیت AND دو ورودی



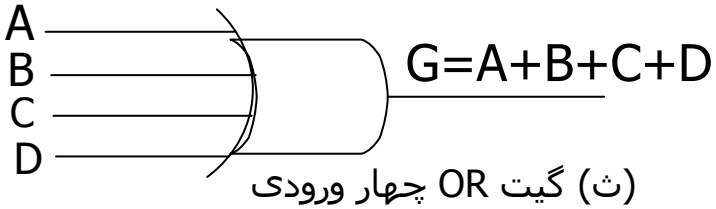
(ب) گیت OR دو ورودی



(ت) گیت AND سه ورودی



(پ) گیت NOT برای معکوس کردن (واروگر)



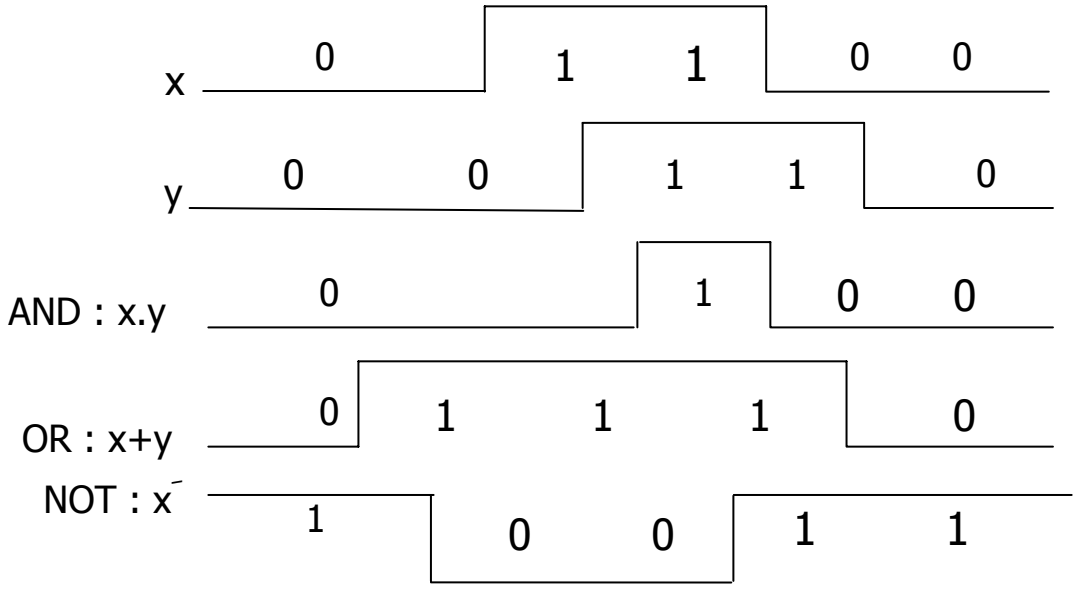
(ث) گیت OR چهار ورودی

شکل (۱-۲): گیتهای NAND، NORو NOT

سیگنال های ورودی X و Y در گیت هایی با دو ورودی ، طبق شکل (۲-۲) مُی توانند به یکی از چهار حالت ممکن ۰۰ ، ۰۱ ، ۱۰ ، ۱۱ باشند . این سیگنال های ورودی به همراه سیگنال های خروجی شان برای گیت های AND و OR در شکل (۲-۲) نشان داده شده اند . نمودار زمانی شکل (۲-۲) پاسخ هر مدار را به هر یک از چهار ترکیب ممکن ورودی نشان می دهد . دلیل انتخاب نام وارونگر برای گیت NOT از مقایسه پالس X (ورودی وارونگر) و X (خروجی وارونگر) بخوبی آشکار می شود .

گیت های AND و OR ممکن است بیش از دو ورودی داشته باشند . یک گیت AND با سه ورودی و یک گیت AND با سه ورودی و یک گیت OR با چهار ورودی در شکل (۱-۲) نشان داده شده اند . گیتAND سه ورودی ، بشرطی در خروجی خود دارای پاسخ ۱ منطقی باشد ، خروجی گیت ۰ منطقی است . گیت OR با چهار ورودی دارای خروجی ۱ منطقی است بشرطی که حداقل یک ورودیها ،۱ منطقی باشد و اگر همه سیگنال های ورودی ۰ منطقی باشند خروجی ۰ منطقی خواهد بود .

فرم ریاضی منطق دودویی ، اغلب جبر بول و یا جبر سوئیچینگ خوانده می شود . این جبر برای تشریح عملیات شبکه های پیچیده در مدارهای دیجیتال استفاده می گردد . طراحان سیستمهای دیجیتال از جبر بول برای تبدیل اشکال مدارها به عبارت جبری و بالعکس استفاده می کنند .



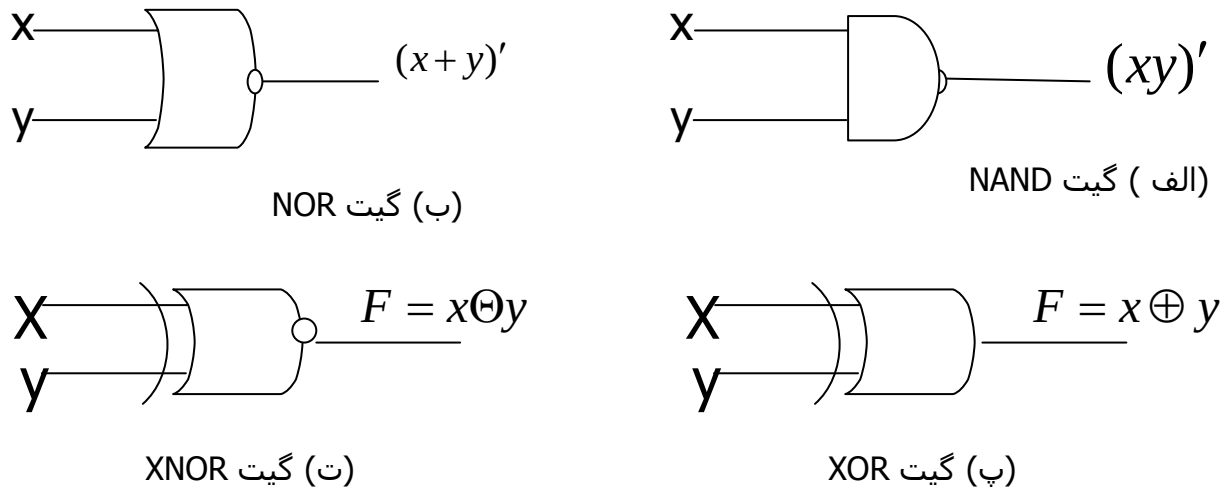
شکل (۲-۲) سیگنالهای ورودی – خروجی برای گیت های (الف) (ب) (پ) از شکل (۱-۲)

گیتهای دیگری یعنی بعنوان گیت های استاندارد در طراحی سیستم های دیجیتال بکار می روند .این گیتها عبارتند از: NAND ، NOR ، XOR ، XNOR

.

تابع NAND ، مکمل AND می باشد و متشکل از یک سمبل AND که بدنبال آن دایره کوچکی قرار گرفته است . تابع NOR مکمل تابع OR بوده و بوسیله سمبل OR که بدنبال آن دایره کوچک نمایش داده می شود . گیت های NAND و NOR بسادگی بوسیله مدارات ترانزیستوری قابل تولید بوده و می توان براحتی توابع بول را با آنها پیاده نمود .

گبت XOR دارای سمبل مشابهی با OR می باشد ، بجز یک خط منحنی که در سمت ورودی اش کشیده شده است . گیت XNOR مکمل XOR است و لذا یک دایره کوچک اضافی در سمت خروجی سمبل آن وجود دارد .



(الف) گیت NAND

(ب) گیت NOR

(پ) گیت XOR

(ت) گیت XNOR

شکل (3-۲) گیت های NAND ، NOR ، XOR ، XNOR

گسترش ورودی گیت ها

گیت های نشان داده شده بجر معکوس کننده و بافر ، قابل گسترش به حالتی بیش از دو ورودی هستند بشرط اینکه عمل دودویی ارائه شده بوسیله آنها خواص جابجایی و شرکت پذیری را داشته باشد . اعمال AND و OR که در جبر بول تعریف شدند دارای این دو خاصیت هستند . برای تابع OR داریم :

جابجایی $x+y=y+x$

و شرکت پذیری $(x+y)+z=x+(y+z)=x+y+z$

این روابط بیانگر آنند که ورودی قابل تعویض بوده و بنابراین تابع OR قابل گسترش به سه متغیر و بیشتر است .

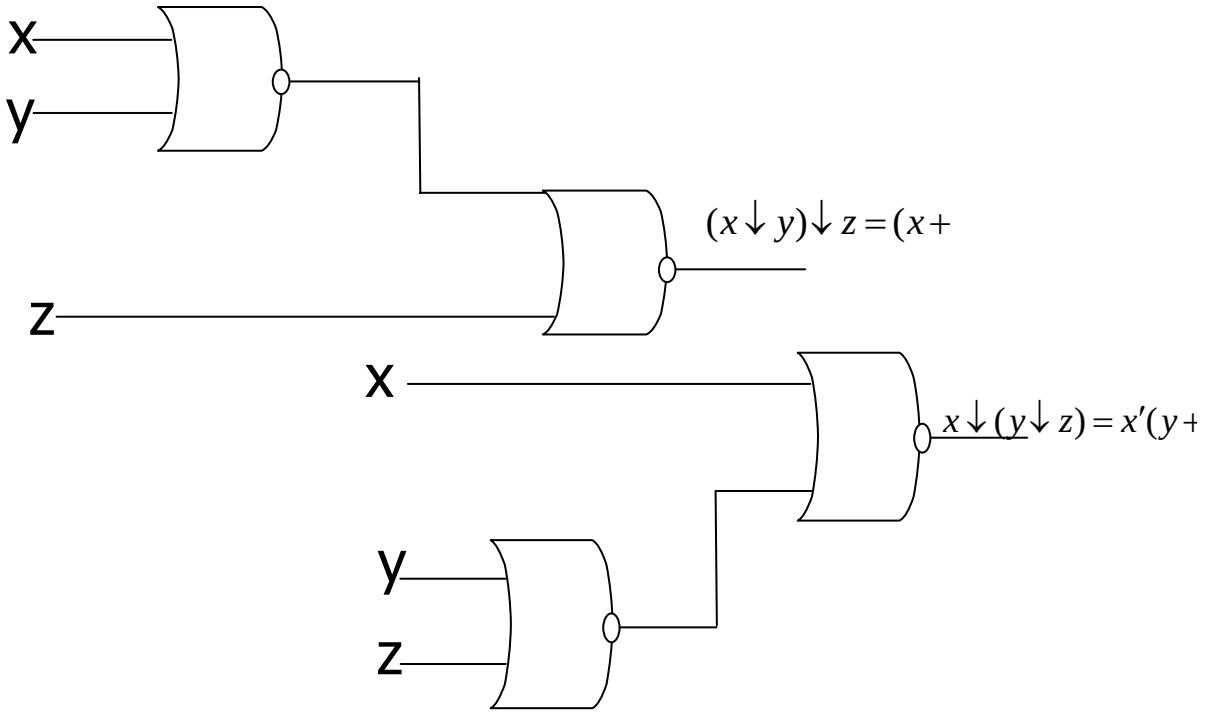
توابع NAND و NOR خاصیت جابجایی دارند و ورودی گیت آنها قابل گسترش است ، بشرط اینکه تعریف عمل آنها تصحیح شود. مشکل این است که عملگرهای NAND ، NOR شرکت پذیری نیستند . یعنی :

$$(x\downarrow y)\downarrow z\neq x\downarrow(y\downarrow z)$$

زیرا طبق شکل (۲-4) داریم :

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

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



شکل (۲-4) نمایش شرکت پذیر نبودن NOR، $(x\downarrow y)\downarrow z\neq x(y\downarrow z)$

برای غلبه براین مشکل گیت های NOR (NAND) چند ورودی را بعنوان مکمل OR (AND) آن تعریف می کنیم ، بنابراین داریم :

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

سمبل های گرافیکی برای گیت های سه ورودی در شکل (۲-5) نشان داده شده اند . در نوشتن متوالی اعمال NOR و NAND بایستی پرانتزها بفرم صحیح انتخاب شوند ، تا بیانگر ترتیب صحیح گیت ها باشند . برای نمایش این مطلب مدار شکل (5-۲-پ) راملاحظه کنید . تابع بول برای این مدر بایستی بفرم زیر نوشته شود :

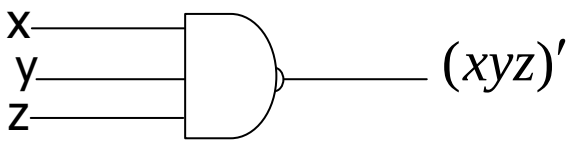
$$F = [(ABC)'(DE)']' = ABC + DE$$

دومین عبارت از رابطه دمورگان نتیجه شده است . این رابطه همچنین بیانگر آنست که جمع حاصلضرب ها قابل پیاده شدن بوسیله گیت ها NAND است .

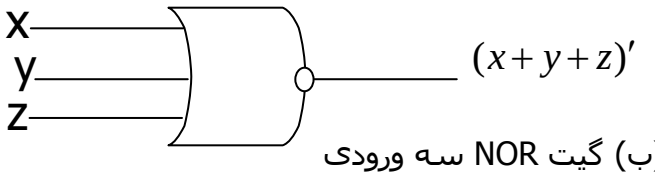
گیت ها XOR و XNOR هر دو دارای خواص جابجایی و شرکت پذیری بوده ، ورودی شان قابل توسعه به بیشتر از دو می باشد . معهذا مدارهای XOR با چند ورودی ، از نقطه نظر سخت افزاری متداول نیستند .در واقع حتی فرم دو ورودی آن نیز معمولاً از سایر گیت ها ساخته می شود . علاوه بر این تعریف این توابع بایستی بهنگام گسترش ورودی آنها تصحیح گردد .تابع XOR یک تابه فرد است یعنی هرگاه ورودی ها تعداد غردی ۱ را دارا باشند این تابع برابر ۱ خواهد بود . ساختمان یک گیت XOR با سه ورودی در شکل (۲-۶) دیده می شود .این مدار معمولاً با گیت های دو ورودی تهیه می گردد . شکل (الف) فرم گرافیکی آن را با گیت سه ورودی نیز می توان نشان داد ، شکل ب) جدول درستی در (پ) بطور آشکار مشخص می نماید که خروجی F

برابر ۱ خواهد بود ، اگر فقط یکی از ورودی ها و یا هر سه ورودی برابر باشد . به بیان دیگر وقتی تعداد ۱ ها درورودی فرد است F مساوی ۱ است .

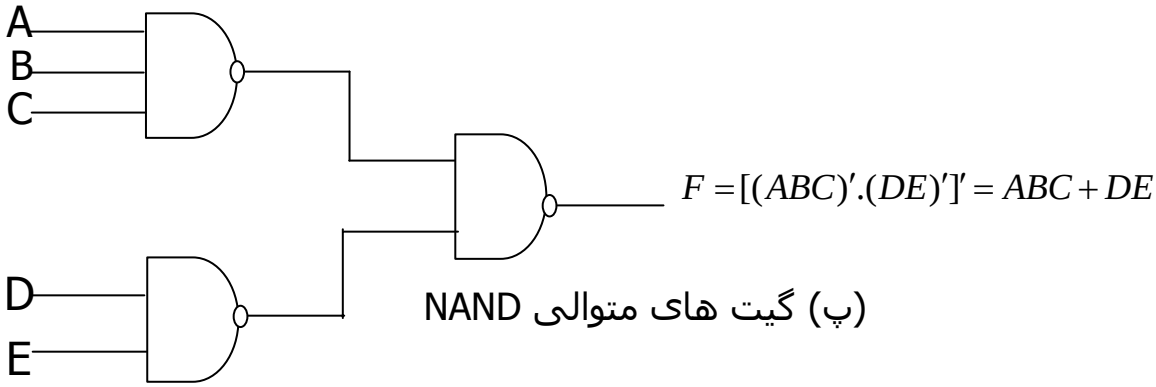
اضافه می نماید که تابع NOR یک تابع زوج است . یعنی هرگاه تعداد ۰ ها در ورودی زوج باشد این تابع مساوی ۱ است .



(الف) گیت NOR سه ورودی

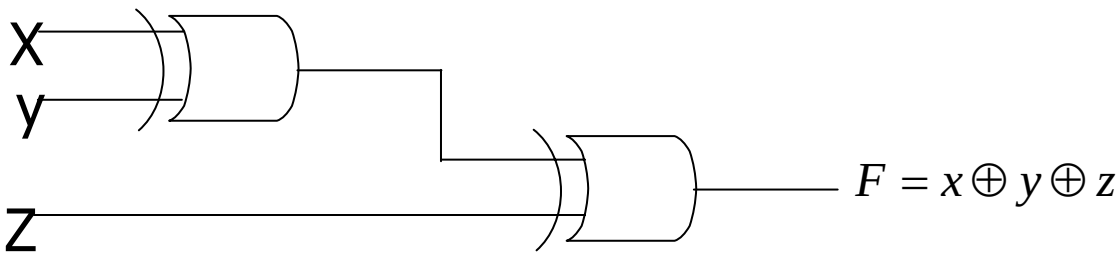


(ب) گیت NOR سه ورودی

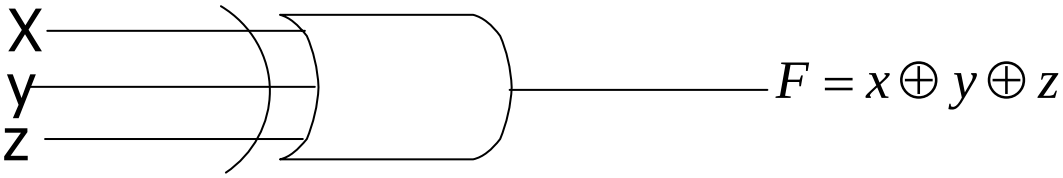


(پ) گیت های متوالی NAND

شکل (2-5) گیت های NAND و NOR پشت سر هم و چند ورودی



(الف) با گیت های دو ورودی

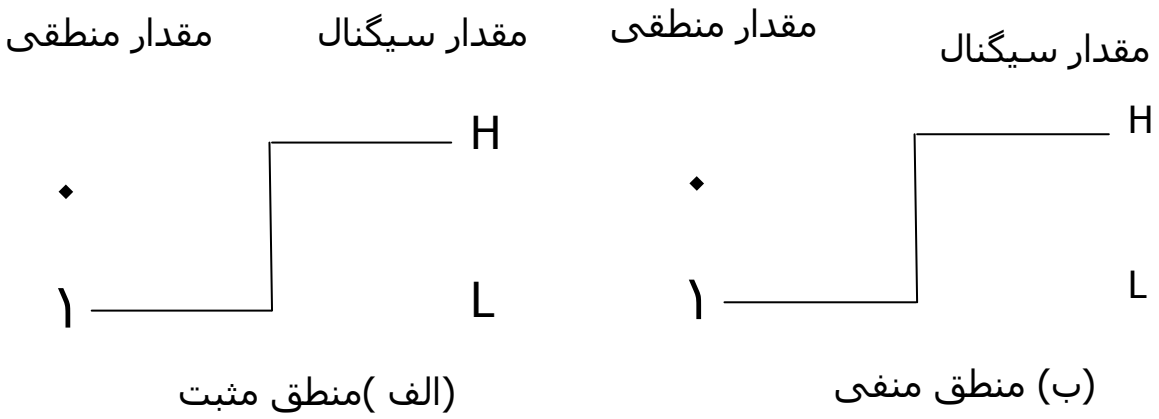


(ب) یک گیت سه ورودی

شکل (۲-۶) گیت XOR سه ورودی

منطق مثبت و منفی

سیگنال دودویی در ورودی یا خروجی هر گیت یکی از دو مقدار را بجز در حالت گذرا ، دارد . یک مقدار سیگنال منطق -١ و دیگری منطق -٠ را نمایش می دهد . چون دو مقدار سیگنال متعلق به دو ارزش منطقی است ، لذا دو انتساب متفاوت برای دو ارزش منطقی می توان اختیار کرد ، شکل (٢-٧) انتخاب سطح بالاتر H برای نمایش منطق ١ مطابق شکل (٢-٧-الف) ، سیستم منطق مثبت را معرفی می نماید و انتخاب سطح پایین L بعنوان منطق ١.



(ب) منطق منفی (الف)منطق مثبت

شکل (٢-٧) علامت دامنه سیگنال و نوع منطق

ساده سازی توابع بول

نقشه ، دیاگرام متشکل از تعدادی مربع است . هر مربع نشان دهنده یک مینترم می باشد و جون هر تابع بول را میتوان بصورت مجموع مینترم ها نمایش داد ، لذا یک تابع بول را می توان بصورت مصور با در نظر گرفتن نواحی اشغال شده بوسیله مربع هایی که مینترم آنها درتابع وجود دارد مشخص نمود . در واقع نقشه یک دیاگرام از کلیه روشهای ممکن برای ارائه استاندارد یک تابع می باشد . با استخراج الگوهای مختلفی از جدول ، استفاده کننده می تواند عبارت جبری معادل ولی ظاهراً متفاوتی را برای یک تابع بدست آورد و از بین آنها ساده ترین را انتخاب کند . ما فرض خواهیم کرد که ساده ترین عبارت جبری در میان جمع حاصلضرب ها یا ضرب حاصلجمع ها ، عبارتی است که تعداد متغیرهای آن کمترین باشد .

یک نقشه دو متغیره در شکل (۱-۳) نشان داده شده است که دارای چهار مینترم برای دو متغیر است . بنابراین نقشه شامل چهار مربع بوده و هر مربع مربوطه به یک مینترم است . برای نشان دادن ارتباط بین دو متغیر و مربعها در قسمت (ب) نقشه دوبار کشیده شده است . ۰ ها و ۱ هایی که برای هر سطح و ستون گذاشته شده مشخص کننده مقادیر متغیر x و y است . توجه کنید که x در سطر ۰ با پریم و در سطح ۱ بدون پریم ظاهر شده است . y ستون ۰ ، با پریم ، و در ستون ۱ بدون پریم آمده است .

اگر مربعهایی که از مینترم آنها متعلق به تابع مفروضی است با علائمی مشخص کنیم روش مفید دیگری جهت نمایش هر یک از ۱۶ تابع ممکن از دو متغیر بدست می آید. بعنوان مثال ، تابع xy در شکل (۲-۳ الف) نشان داده شده است . از آنجا که xy برابر با m_3 می باشد در مربع مربوط به m_3 ، ۱ قرار گرفته است . بطور مشابه تابع x+y نیز در نقشه شکل (۲-۳ب) بوسیله سه مربعی که با ۱ پر شده اند مشخص شده است . این مربعها از مینترم های تابع بدست آمده اند :

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

همچنین می توان سه مربع را از اشتراک متغیر x در سطر دوم و متغیر y در ستون دوم که ناحیه متعلق به x یا y را در بر می گیرد بدست آورد .

m_0	m_1
m_2	m_3

(الف)

x \ y	0	1
0	$x'y'$	$x'y$
1	xy'	xy

(ب)

x \ y	0	1
0		
1		1

xy (الف)

x \ y	0	1
0		1
1	1	1

x+y (ب)

یک نقشه سه متغیره در شکل (۳-۳) نشان داده شده است . هشت مینترم برای سه متغیر دودویی وجود دارد ، بهمین جهت نقشه دارای هشت مربع است . توجه

کنید که مینترم ها بر اساس ترتیب دودویی مرتب نشده اند بلکه ترتیبشان بر اساس کد گری فهرست شده در جدول (۴-۱) است . خاصیت این ترتیب این است که از هر مربع به مربع دیگر فقط یک بیت از ۰ به ۱ و یا از ۱ به ۰ تغییر می کند . نقشه ای که در قسمت (ب) کشیده شده با شماره هایی برای هر سطر و هر ستون علامت گذاری شده است تا ارتباط مربعها و سه متغیر را نشان بدهد . مثلاً مربعی که به m_5 نسبت داده شده به سطر ۱ و ستون ۰۱ مربوط است . وقتی این دو عدد به هم ملحق می شوند ، عدد دودیی ۱۰۱ را می سازد که معادل عدد ۵ است .از دید دیگری می توان مربع $m_5 = xy'z$ را مورد توجه قرار دارد ، به این شکل که بگوییم m_5 در سطر مربوط به x و ستون متعلق به z' قرار گرفته است . (س تون ۰۱) توجه کنید که هر متغیر در چهار مربع ۰ . چهار مربع دیگر ۱ است . بخاطر سهولت ، متغیردر خانه های ۱ بدون پریم و در خانه های ۰ با پریم ظاهر می شود . برای سادگی ، اسیم متغیر را با سمبل حرفی اش در زیر خانه هایی که بدون پریم هستند می نویسیم .

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(الف)

yz / x	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

(ب)

شکل (۳-۳) نقشه سه متغیره

جهت درک فایده نقشه در ساده سازی توابع بول می بایست خاصیت مربع های همجوار را مشخص کنیم . تنها اختلاف بین هر دو مربع در یک متغیر می باشد.تابع

$F(x,y,z)=\sum(5,7)$ را در نظر بگیرید. در نقشه کارنو m_5 , m_7 را ۱ و سایر خانه ها با ۰ پر می گردند. با توجه به اصول جبر بول نتیجه میگیریم که می توان جمع دو مینترم در مربع های همجوار را به AND با دو متغیر ساده کرد به منظور روشن شدن مطلب به جمع دو مربع همجوار m_5 , m_7 توجه کنید .

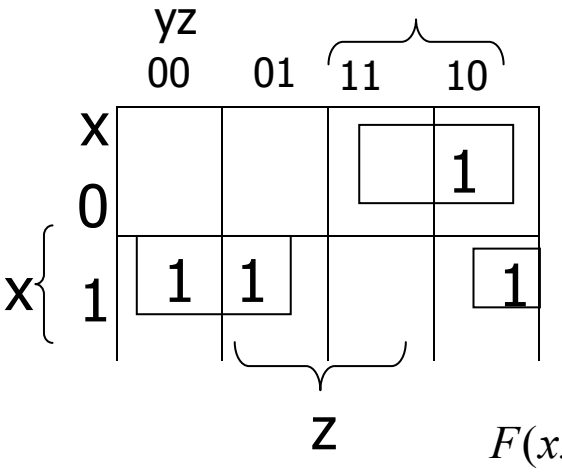
$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

در اینجا دو مربع در متغیر y اختلاف دارند که می توان بهنگام جمع آن را حذف کرد . بنابراین هر دو مینترم در دو مربع همجوار که با هم OR شده اند سبب حذف متغیری می گردند که در آن دو مینترم ، متفاوت است . مثالی زیر روالی را برای می نیمم کردن یک تابع بول بوسیله جدول بیان می کند .

مثال (۱-۳) : تابع بول زیر را ساده کنید .

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

ابتدا در خانه هايي كه مينترم هاي آن در تابع وجود دارد ۱ می گذاریم . این کار در شکل (۴-۳) که در آن مربع هاي مربوط به مينترم هاي ۰۱۰ ، ۰۱۱ ، ۱۰۰، ۱۰۱، با ۱ علامت زده شده اند قدم بعدي يافتن مربع هاي همجوار است . این کار در نقشه با مربع مستطيلي كه دو عدد ۱ را در بر مي گيرد صورت گرفته است . مستطيل بالاي سمت راست ناحيه اي را كه زير پوشش $x'y$ است شامل مي شود. بطور مشابه مستطيل پايين سمت چپ جمله ضرب xy' نشان می دهد. (سطر دوم نشان دهنده x و دو ستون سمت نیز نتیجه خواهد داد و در نتیجه $F=x'y+xy$



شکل (۳-۴) نقشه مثال ۳-۱

$$F(x.y.z)=\sum(2.3.4.5)=x'y+xy'$$

حالاتی وجود دارند که در آنها دو مربع مجاورند حتی اگر بهم نچسبیده باشند . در شکل (۳-۳) ، m_0 مجاور m_2 و m_4 مجاور m_6 است زیرا مینترم ها تنها با یك تغییر با هم حل اختلاف دارند .

این مطلب بصورت جبري قابل اثبات است .

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

در نتیجه ما باید تعریف همجواري مربع هاي را براي منظور نمودن مورد فوق یا موارد مشابه دیگر تصحیح کنیم . این تصحیح بدین صورت انجام مي گیرد که نقشه کشیده شده در یك سطح ، از دو لبه سمت چپ و راست مجاور تصور مي شوند.

مثال ۳-۲ : تابع زیر را ساده کنید .

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

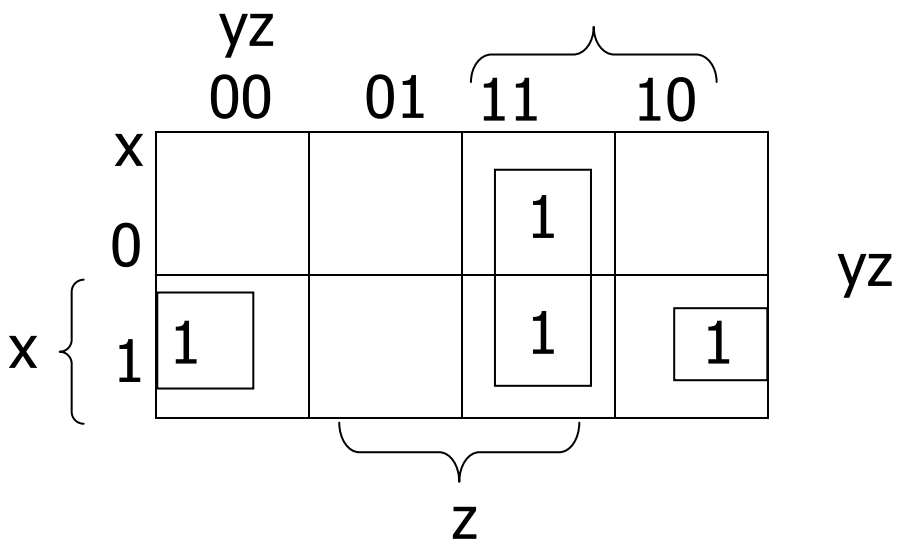
نقشه این تابع در شکل (۳-۵) نشان داده شده است . چهار مربع که هر کدام مربوط بهیک مینترم از تابع می باشد با ۱ پر می شود . دو مربع همجوار در ستون سوم با هم ترکیب شده اند تا عبارت xy را بوجود آورند . ضمناً دو مربع باقیمانده که دارای ۱

هستند با توجه به تعریف جدید همجوار می باشند و در دیاگرام بوسیله یک جفت نیم مستطیل مشخص شده اند . این دو مربع پس ازترکیب ، تابع بولی ساده شده عبارتست از :

$$F = yz + xz'$$

حال به ترکیب چهار مربع همجوار در یک نقشه سه متغیره توجه کنید . چنین ترکیبی نشان دهنده جمع چهار مینترم همجوار است و نتیجه این ترکیب فقط یک عبارت یک متغیره خواهد بود . بعنوان مثال جمع چهار همجوار ۰،۲،۴،۶، به عبارت z' تقلیل می یابد .

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



شکل (۵-۳) نقشه براي ۲-۳

$$F(x.y.z) = \sum(3.4.6.7) = yz + xz = yz + xz'$$

تعداد مربعات همجواری که ممکن است ترکیب شوند همواره برابر عددی که توانی از دو است ، مانند ۸،۲،۴،۱ که هر چه تعداد بیشتری از مربعات همجواری ترکیب شوند جمله حاصلضرب منتجه دارای تعداد کمتری متغیر است .

یک مربع که یک مینترم را نمایش می دهد دارای سه متغیر است .

دو مربع همجوار نشان دهنده یک جمله یا دو متغیر است .

چهار مربع همجوار نشان دهنده یک جمله با یک متغیر است .

هشت مربع همجوار که تمام نقشه را در بر می گیرند همواره تابع ۱ راتولید می نماید .

مثال ۳-۳: تابع بول زیر را ساده کنید .

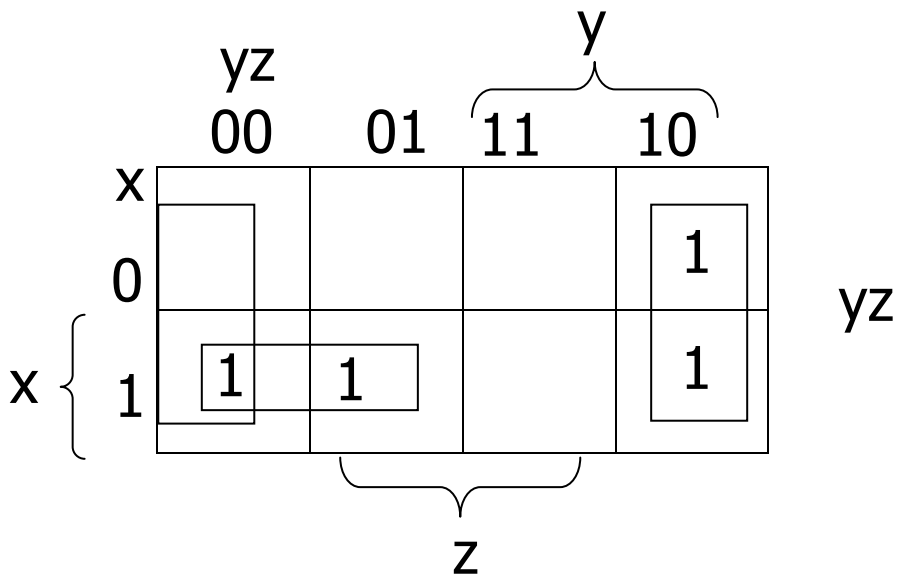
$$F(x.y.z) = \sum 0,2,4,5,6)$$

نقشه تابع f در شکل (۳-۶) نشان داده شده است . ابتدا ، ما چهار مربع مجاور را در اولین و آخرین ستون ترکیب می نماییم تا جمله z' از آن حاصل شود . تنها مینترم باقیمانده که متعلق به مینترم ۵ است با مربع مجاورش که قبلاً ترکیب شده است . این نه تنها مجاور است بلکه مفید نیز هست . چون دو مربع مجاور جمله دومتغیره xy' رابدست می دهد در حالیکه یک مربع تنها به جمله سه متغیره $xy'z$ متعلق است . تابع ساده شده عبارتست از

$$F = z' + xy'$$

اگر تابعی بصورت جمه مینترم ها بیان نشده باشد ، می توان از نقشه برای تهیه مینترم ها استفاده کرد و سپس تابع را بمنظور کاهش به حداقل متغیرها ساده نمود . البته لازم است که عبارت جبری حتماً بصورت جمع حاصلضرب ها باشد . هر جمله

ضرب قابل نشان دادن در یک ، دو یا چند مربع است . سپس مینترم های تابع مستقیماً از جدول استنتاج می گردند .



شکل (۳-۶) نقشه مثال ۳-۳ $F(x.y.z) = \sum (0.2.4.5.6) = z' + xy'$

مثال ۳-۴ : تابع بول مفروض زیر را :

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

الف) بصورت مجموع مینترم ها نماشی دهید .

ب) تابع می نیمم را بصورت جمع حاصلضرب بدست آورید .

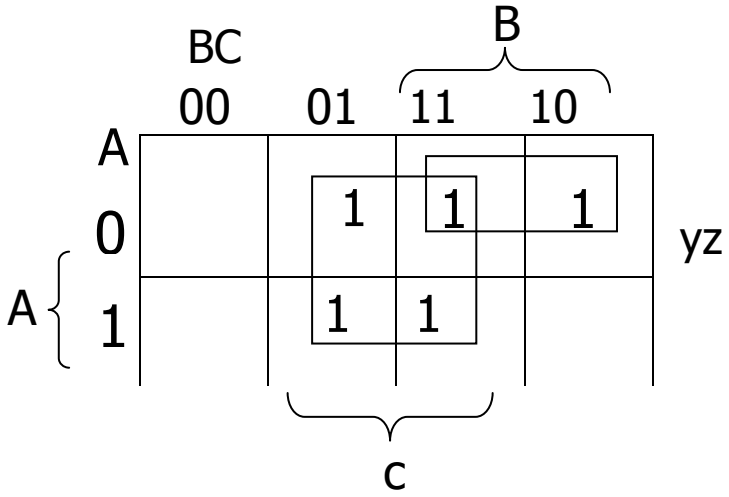
سه جمله ضرب در عبارت دارای دو متغیر بوده و در هر نقشه هر یک بوسیله دو مربع نشان داده شده اند . دو مربع مربوط به اولین جمله $A'C$ در شکل (۳-۷) از تلاقی A َ (اولین سطر) با C (دو ستون وسط) یافت می شوند . که مربع های ۰۰۱ و ۰۱۱ خواهندبود .

توجه کنید که وقتی داخل مربع ها را با ۱ علامت می گذارید احتمال یافتن یک ۱ ، حاصل از جمله قبل در آن وجود دارد . این در دومین جمله یعنی $A'B$ ملاحظه می گردد که یک ۱ در حاصل ۰۰۱ و ۰۱۰ قرار دارد ولی مربع ۰۰۱ با $A'C$ مشترک است

لذا تنها ۱ علامت گذاری می شود . ادامه کار بترتیب فوق نشان می دهد که $AB'C$ مربوط به مربع ۰۱۰ است که متعلق به مینترم ۵ می باشد و جمله BC نیز دارای دو ۱ در مربعات ۰۰۱ و ۱۱۱ است . پس تابع کلاً دارای پنج مینترم است که در نقشه شکل با پنج ۱ مشخص گردیده است . مینترم ها که مستقیماً از نقشه خوانده می شوند و عبارتند از ۵,۳,۲,۱ و ۷ . تابع را می توان بر حسب مجموع مینترم ها نشان داد .

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

بنابراین عبارت مجموع حاصلضرب های اولیه دارای تعداد قابل ملاحظه ای مینترم است . می توان همانطور که در نقشه دیده می شود آن را ساده کرد بطوری که فقط دو متغیر داشته باشد .



شکل (۳-۷) نقشه براي مثال ۳-۴ $A'C+A'B+BC=C+A'B$

۳-۳ نقشه چهار متغیره

در شکل (۳-۸) نقشه مربوط به توابع بول با چهار متغیر مشاهده می شود .در (الف) شانزده جمله مینترم ، فهرست گردیده و به هر کدام یک مربع نسبت داده شده است . در حالت (ب) نقشه دو مرتبه رسم شده تا بیانگر ارتباط بین چهار متغیر

باشد . دریفها و ستونها بر اساس ترتیب کد گری شماره گذاری شده اند ، که بین دو سطر و یا دو ستون همجوار یک تغییر رقم وجود دارد . مینترم مربوط ستون دوم (۰۱) که وقتی به هم ملحق شوند حاصل عدد دودویی ۱۱۰۱ است و معادل عدد ۱۳ دهدهی می باشد .بنابراین مربع ردیف سوم و ستون دوم عبارت m_{13} را نشان میدهد . یک مربع که یک جمله مینترم را نمایش می دهد دارای چهار متغیر است .دومربع همجوار نشان دهنده یک عبارت با سه متغیر است .چهار مربع همجوار نشان دهنده یک عبارت با دو متغیر است . هشت مربع همجوار یک عبارت با یک متغیر را نشان می دهد . شانزده مربع همجوار نشان دهنده تابعی معادل ۱ است .

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
M_{12}	M_{13}	M_{15}	M_{14}
M_8	M_9	M_{11}	M_{10}

(الف)

wx \ yz	00	01	11	
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$w'x'yz$	$w'x'yz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

y (روی ستون 11)

w (روی سطرهای 11 و 10)

z (روی ستون 01 و 11)

(ب)

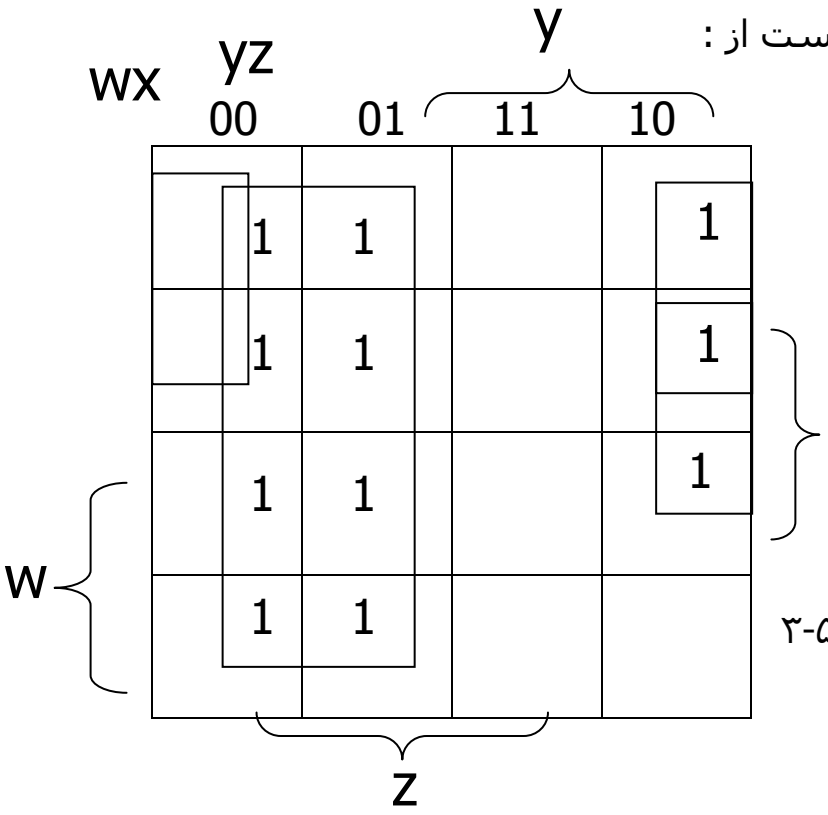
شکل (۸-۳): نقشه چهار متغیره

مثال ۵-۳ : 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 تولیدمیکنند . 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 یاین 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 ودو ستون آخر می باشند عبارت xz' خواهد بود .در نهایت تابع ساده شده عبارتست از :

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



شکل (۹-۳) نقشه مثال ۵-۳

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

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

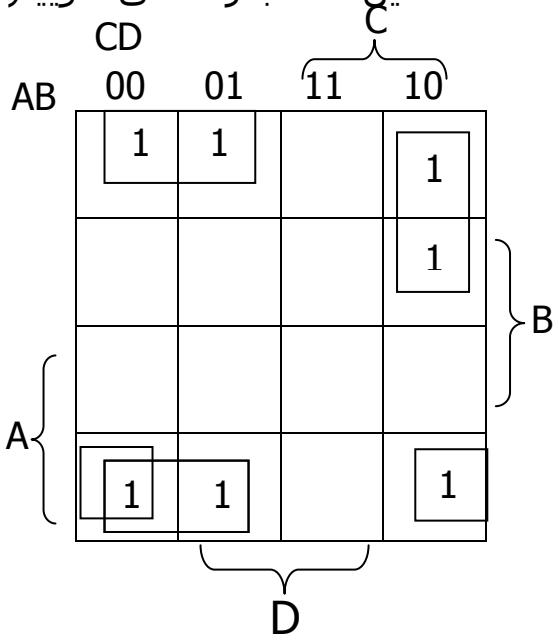
ناحیه ای از نقشه که بوسیله این تابع پوشیده می شود شامل مربعهایی است که در شکل (۱۰-۳) با عدد ۱ پر شده اند . این تابع چهار متغیر دارد و همانطور که ملاحظه شد شامل سه جمله هر یک با سه متغیر و یک جمله با چهار متغیر است . جملات سه متغیره در نقشه با دو مربع نشان داده شده اند . مثلاً $A'B'C'$ بوسیله مربعهای 0000 و 0001 مشخص شده است . تابع را می توان بوسیله ترکیب ۱ های چهار گوشه نقشه ساده کرد کو 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 همجوارند . 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 درعبارت 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 درنقشه 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 گوییم .



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 هشتایی 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 (١١-٣ ب) تمام راههای ممکنی 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 .

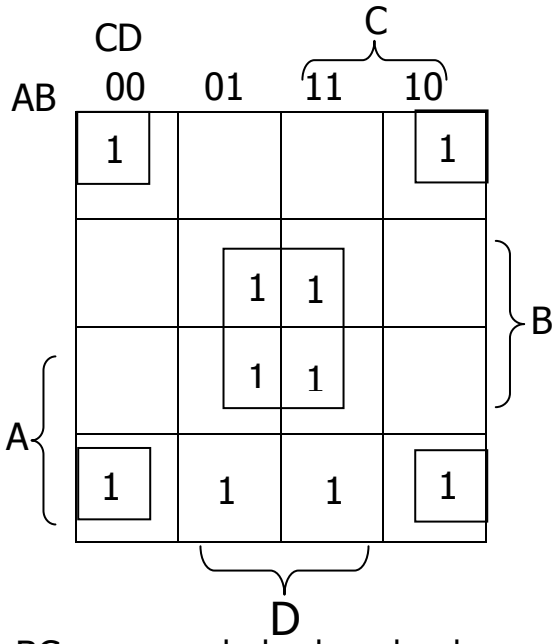
عبارت ساده شده از جمع منطقی دو نخستین انتخاب اصلی و هر دو نخستین انتخابی که مینترم های m_3،m_9 و m_{11} را پوشش دهد ، حاصل می گردد . چهار روش برای بیان تابع با چهار جمله ضرب که هر یک دارای دو متغیرند وجود دارد :

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

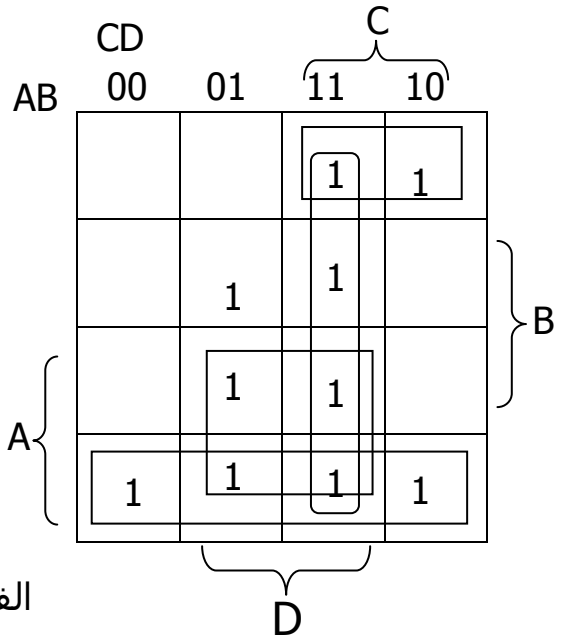
مثال فوق نشان داد که شناخت نخستین انتخاب ها در نقشه به یافتن صور مختلف تابع کمک موثری مینمایند .

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 هاي اصلي $B'D'$ و BC



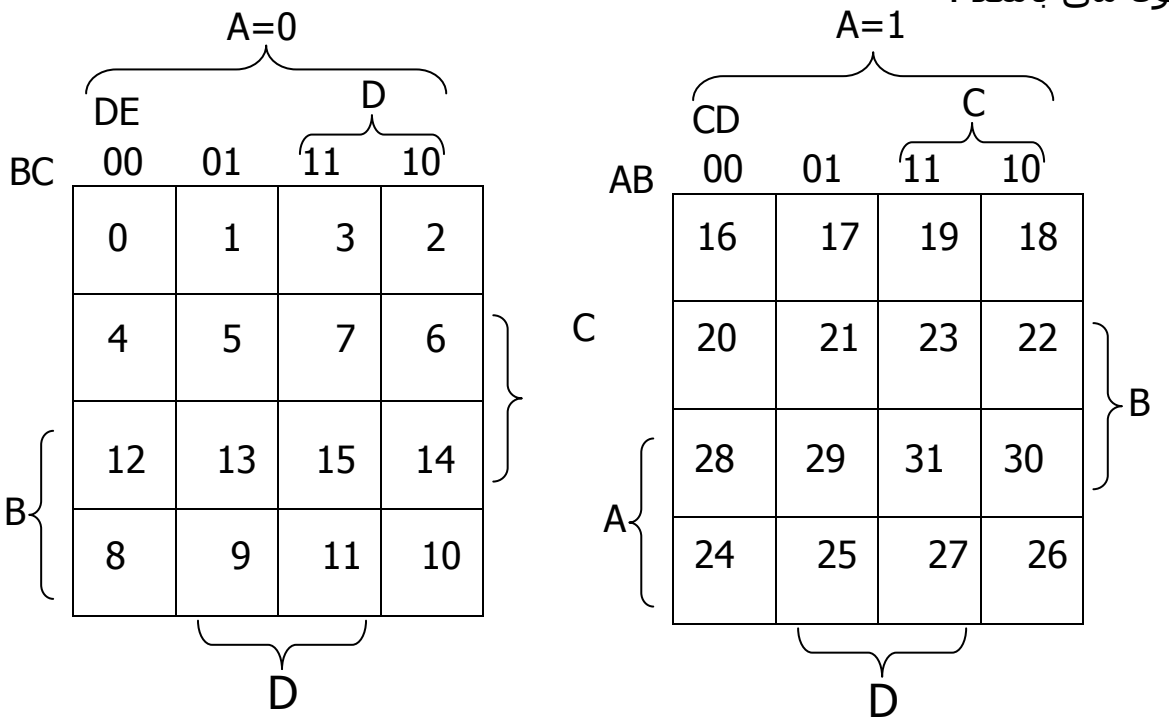
ب) نخستن انتخاب هاي $B'D'$ ، CD ، AD و AB

۳-۴ 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 (۳-۱۲) 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 جچهار متغیره وقتی که

جداگانه در نظر گرفته ، همجواری تعریف شده قبلی خود را حفظ می کند . بعلاوه هر مبع از نقشه A=0 با مربع متناظرش در A=1 همجوار است . مثلاً مینترم ۴ با مینترم ۲۰ مجاور است و مینتمر 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 تبدیا 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 نیازبه تعداد بی شماری مربع داشته و غیر کاربردی هستند . روش دیگر بکار بردن بنرامه 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 $k = 0,1,2,...,\ n$ 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 (۱-۳) 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	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

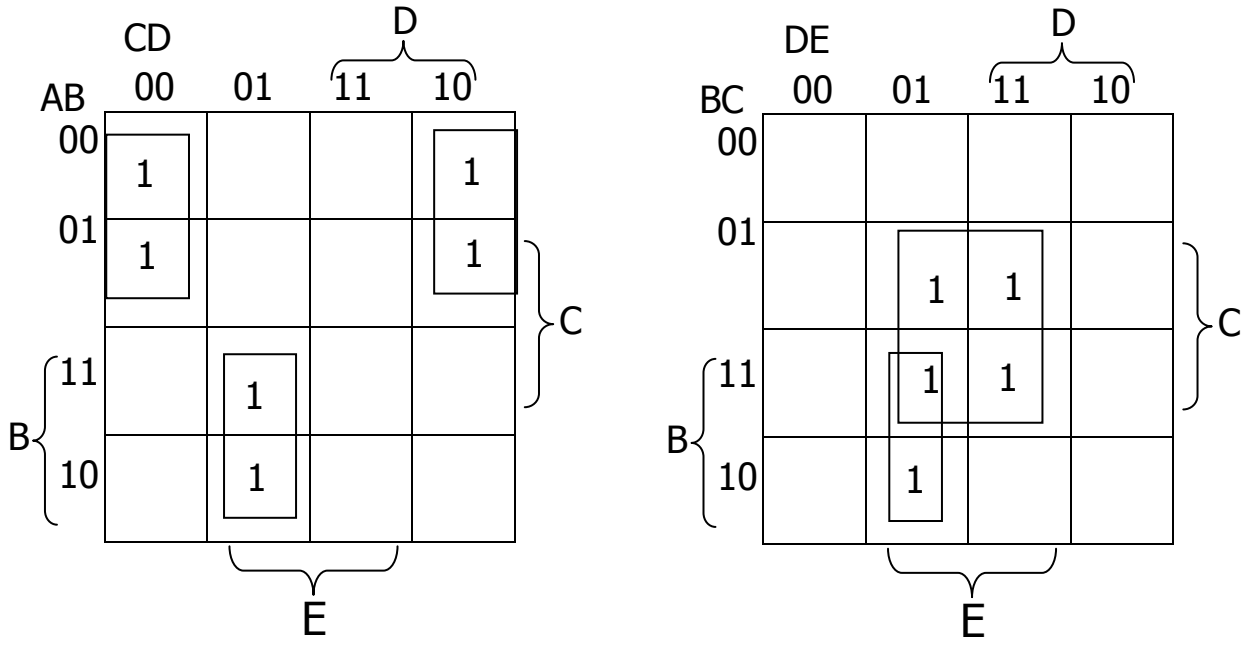
مثال ۷-۳: تابع بول زیر را ساده کنید :

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

نقشه پنج متغیره برای این تابع در شکل (۱۳-۳) دیده 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 (۳-۱۳) نقشه براي مثال ۳-۷ $F = A'B'E' + BD'E + ACE$

۳-۵ 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 تغیر کوچک می توان فرم ضرب حاصلجمع ها را نیز برای آنها بدست آورد .

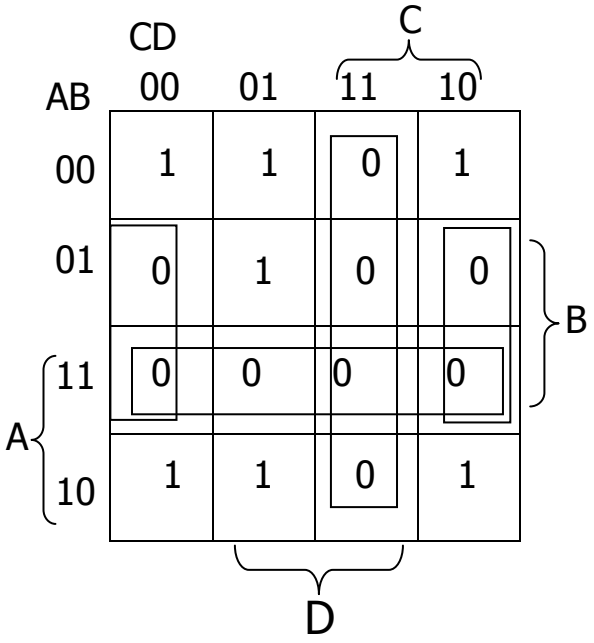
یافتن روالی برای بدست آوردن یک تابع می نیمم بر حسب ضرب حاصلجمع ها نیازمند دانستن خواص اساسی توابع بول است . وجود ۱ ها در مربعهای نقشه ، بیانگر مینترم های تابع است . مینترم هایی که در تابع نیستند بیانگر مکمل تابع می باشند . با توجه به این مطلب مشاهده می کنیم که مکمل یک تابع بوسیله مربعهایی که فاید ۱ هستند نشان داده می شود . اگردر 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 ۸-۳: 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 موجوددر نقشه شکل (۱۴-۳) نشان دهنده تمام مینترم های تابع است . مربعهای دارای ۰ بیانگر مینترم هایی هستند که در تابع F نبستند ، بنابراین بر مکمل F دلالت می کنند . از ترکیب مربعهای ۱ ، تابع ساده شده به فرم جمع حاصلضرب ها بدست می آید.



شکل (۱۴-۳) نقشه مثال ۸-۳

الف) $$F = B'D' + B'C' + A'CD'$$

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

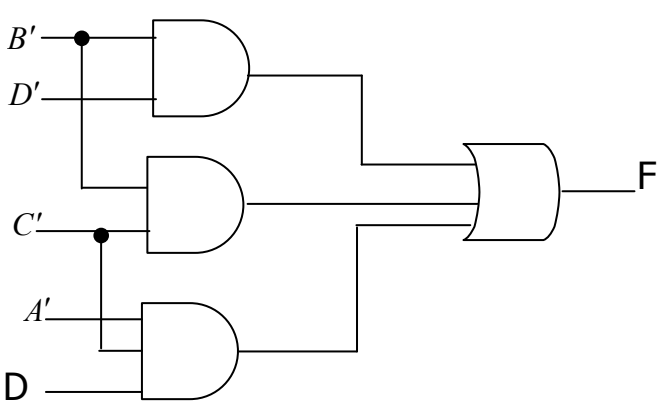
$$F' = AB + CD + BD'$$

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

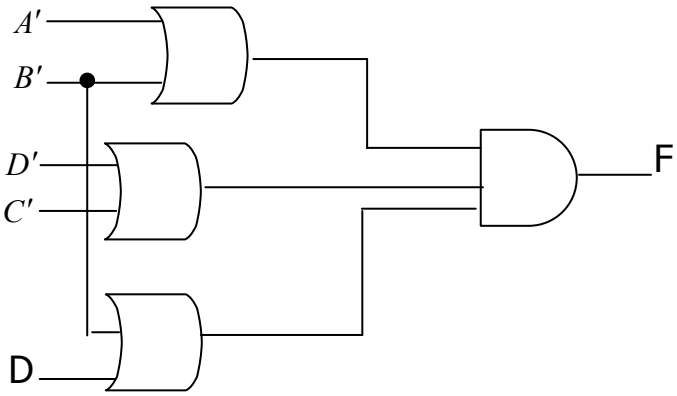
ب) $$F = (A' + B')(C' + D')(B' + D)$$

پیاده سازی عبارت ساده شده حاصل از مثال ۸-۳ در شکل (۱۵-۳) نشان داده شده است . عبارت جمع حاصلضرب ها در شکل (۱۵-۳) ، 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 رورودی 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 (۱۵-۳) یک سری روشهای کلی هستند ک 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

پیکره 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 .



$F = B'D' + B'C' + A'C'D'$ (الف)



$F = (A' + B')(C' + D')'(B' + D)$ (ب)

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

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

و در ضرب ماکسترم ها بصورت زیر است :

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

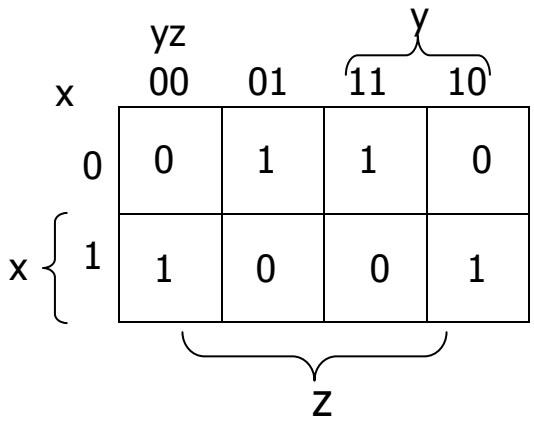
به عبارت دیگر ۱ های تابع نشان دهنده جملات مینترم و ۰ های آن بیانگر جملات ماکسترم هستند . نقشه این تابع در شکل (۱۶-۳) رسم شده است . برای ساده کردن این تابع ابتدا می توان در مربع مربوط به هر جمله مینترم که تابع به ازای آن مقدار ۱ دارد عدد ۱ گذاشت و مربعهای باقیمانده را با ۰ پر می کرد . از طریق دیگر اگر

تابع به فرم ضرب ماکسترم ها داه شده باشد در ابتدا می توان در مربعهایی که تابع مشخص می کند ۰ گذاشت و مربعهای باقیمانده را با ۱ پر کرد . هنگامی که ۱ ها و ۰ ها در جدول گذاشته شدند ، تابع می تواند باقیمانده را با ۱ پر کرد . هنگامی که ۱ ها و ۰ ها در جدول گذاشته شدند ، تابع می تواند به یکی از فرمهای استاندارد ساده شود . برای جمع حاصلضرب ها ۱ ها را با ترکیب می کنیم و خواهیم داشت :

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

جدول (۲-۳) 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



شکل (۱۶-۳) نقشه تابع جدول (۲-۳)

براي ضرب حاصلجمع ها ، ۰ 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 ارزي مي باشد . (بخش ۶-۲) با مکمل کردن F' در حقیقت تابع ساده شده به فرم ضرب حاصلجمع ها خواهیم داشت :

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

براي وارد کردن يك تابع در يك نقشه که بر حسب ضرب حاصلجمع ها بیان شده است مي بايست مکمل تابع را بدست آوردد و با استفاده از آن ، مربعهاي مربوطه را با ۰ پر کرد . برای مثال تابع

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

و سپس مربعهایی که عبارات می نیمم f رانشان مي دهند با ۰ مربعهاي باقيمانده را با ۱ پر مي کنيم .

۳-۶ پیاده سازي بوسيله کيت هاي NAND و NOR

مدارهاي ديجيتال اغلب بجاي اينکه با کيت هاي AND و OR ساخته شوند با کيت هاي NAND و NOR ساخته مي شوند .ساختن گیت های NAND و NOR بااجزای الکترونکی 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 .

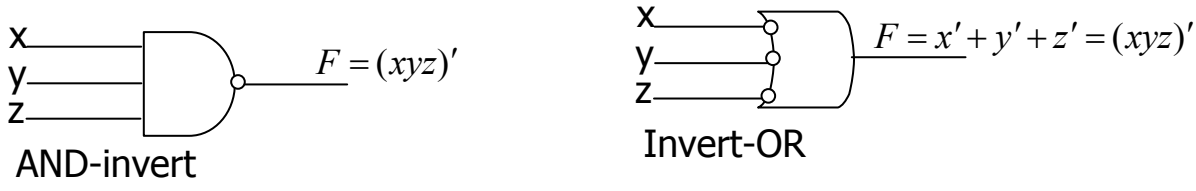
برای سادگی تبدیل به منطق NAND و NOR بهتر است دو سمبل دیگر را برای این گیت ها تعریف کنیم .دو سمبل برای گیت NAND در شکل (۳-۱۷) نشان داده شده

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 ودر نظر گرفتن این قرار داد که دوایر کوچک مکمل کردن می باشند بدست می آید .

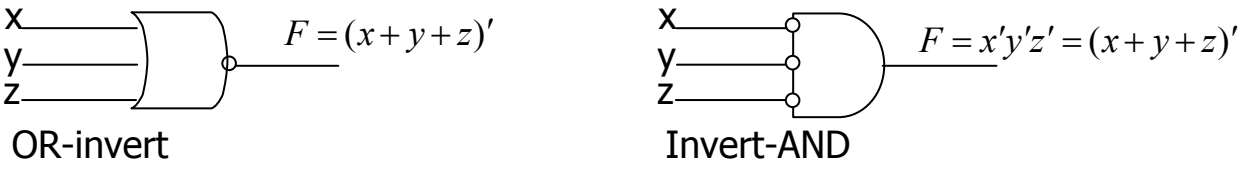
بطور مشابه ، دو سمبل نیز برای گیت NOR یک سمیل قراردادی است وAND-INVERT نیز شکل مناسب دیگری استه قضیه دمورگان و قرار داد دایره های کوچک به معنی مکمل کردن را مورد استفاده قرار می دهد .

یک گیت nand یا nor با یک ورودی مثل معکوس کننده عمل می کند ، نتیجتاً یک گیت معکوس کننده رامی توان به یکی از سه حالت مختلف که در شکل (٣-١٧ پ) نشان داده شده است نمایش داد .دایره های کوچک درهر سمبل معکوس کننده را می توان بدون تغییر منطق گیت به پایه ورودی آن انتقال داد .

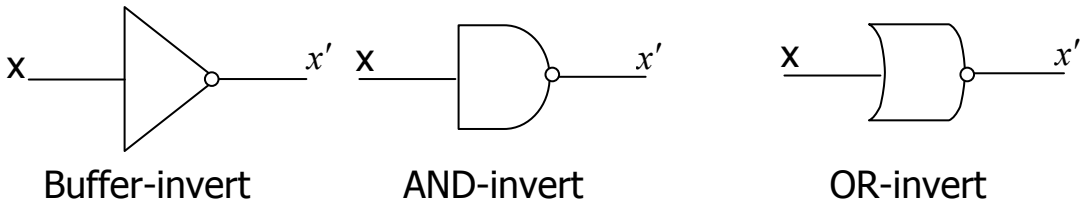
باید خاطر نشان کرد که سمبلهای معادل دیگر برای گیت های NAND و NOR را می توان با جایگزینی مثلث های کوچک بجای دایره های در همه پایانه های ورودی کشید . یک مثلث کوچک نشانه یک منطق منفی است ، بنابراین وجود مثلهای کوچک روی پیایه های ورودی در سمبل گرافیگی یک گیت ، دلالت برا منفی بودن منطقی ورودی های آن دارد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



(الف) دو سمبل گرافیکی برای گیت NAND



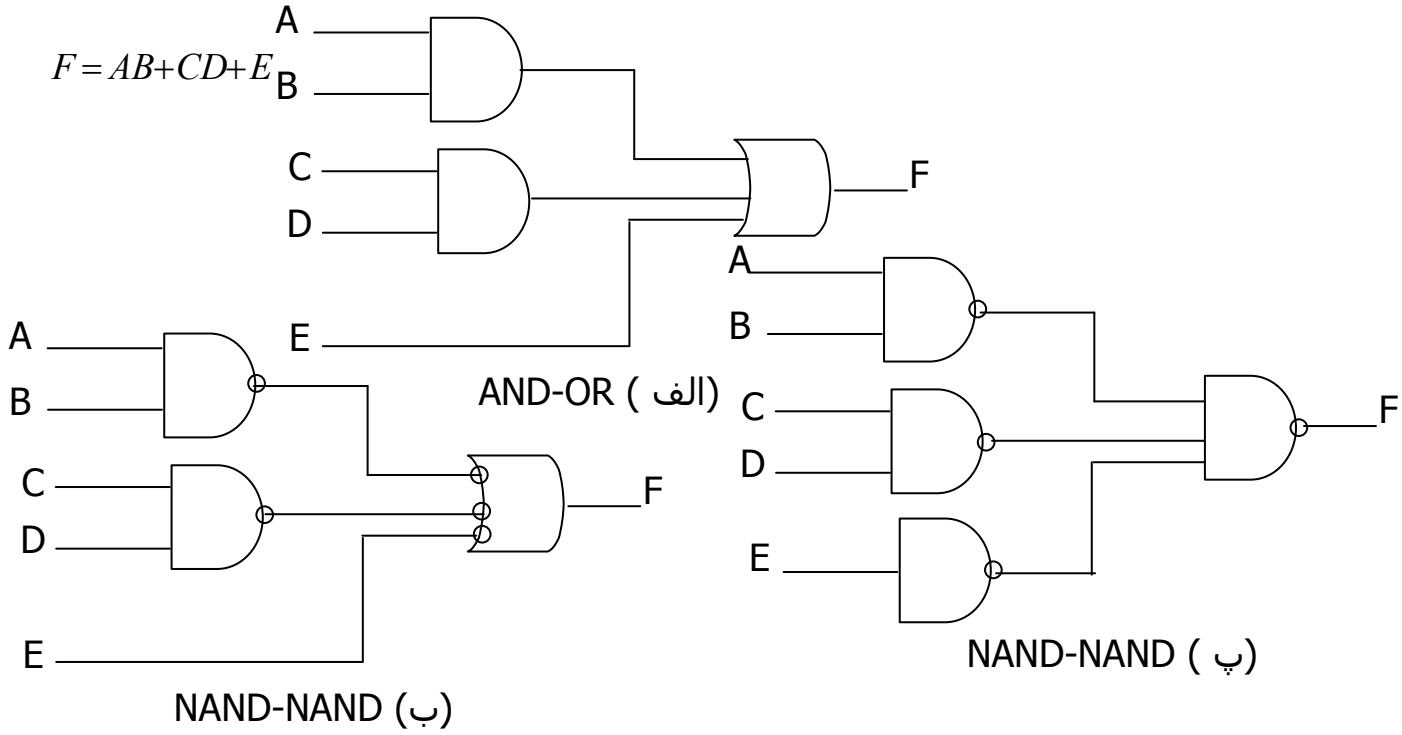
(ب) دو سمبل گرافیکی برای گیت NOR



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

پیاده سازی با گیت NAND

پیاده سازی یک تابع بول باگیت های NAND مستلزم این است که تابه 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 باشیدکه دایره کوچک دال بر مکمل سازی است . بنابراین دو دایره کوچک در یک مسیر نشان دهنده دوبار مکمل سازی بوده وم یتواند 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 درگیت های شکل (۱۸-۳ب) ف مدار (الف) راتولید می کند . بنابرای 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 است که این نیز به نوبه خودمعادل (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 بااستفاده از قانون دمورگان به فرم جمع حاصلضرب تبدیل گردد .

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

از تبدیل گام به گام در شکل (۳-۱۸) نتیجه می گیریم که یک تابع بول می تواند به دو طبقه از گیتهاNAND 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 گرافییکی AND-invert یا invert –OR استفاده کنید) .

٤- 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 بردذ .

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 بیانشده 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 دراین صورت متددوم ارجح می باشد .

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

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

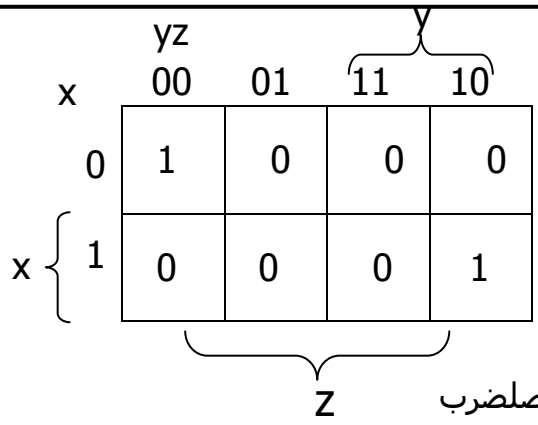
اولین قدم ساده کردن تابع به فرم جمه حاصلضرب ها است که طبق نقشه نشان داده شده در شکل (۱۹-۳ الف) انجام می شود .در نقطه فقط دو ۱ موجود است که نمی توانند با هم ترکیب شوند . بنابراین تابع ساده شده به فرم جمع حاصلضرب ها عبارتست از :

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

پیاده سازی با دو طبق گیت NAND در شکل (۱۹-۳ 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 .

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

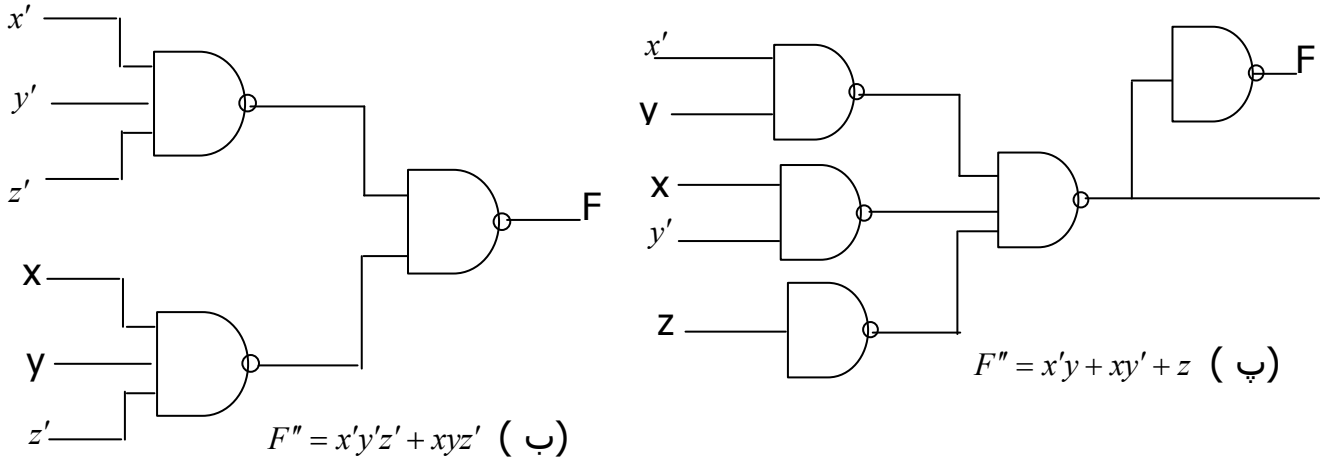
در شکل (۱۹-۳ پ) دو طبقه گیت NAND برای تولید F' نشان داده شده است .اگر در خروجی به F نیاز باشد یک گیت NAND با یک ورودی نیز اضافه شود تا تابع را معکوس کند که در این صورت پیاده سازی در سه طبقه خواهد بود . اگر متغیرها فقط به یک فرم قابل دسترسی باشند می بایست در ورودی معکوس کننده هایی منظور کنیم که باعث اضافه 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 .



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

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

(الف) ساده سازی نقشه در جمع حاصلضرب



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

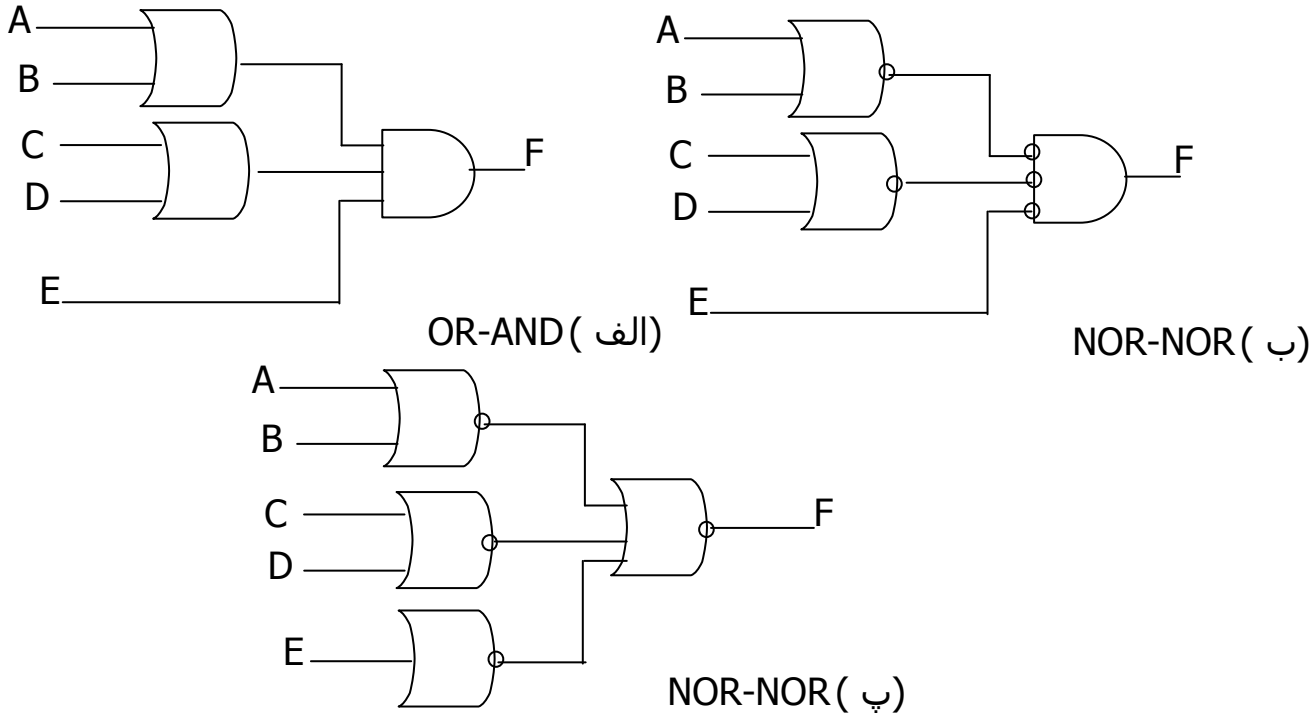
پیاده سازی بوسیله گیت NOR

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 دیاگرام NOR-NOR در شکل (۲۰-۳) 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 یگ 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 ازمکمل 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 ۱۰-۳: تابع مثال ۹-۳ را با گیت های NOR پیاده کنید .

نقشه این تابع در شکل (۱۹-۳ 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 برحسب ضرب حاصلجمع ها است و برای پیاده سازی بوسیله NOR لازم است بدست آورید .

پیاده سازی دو طبقه با گیت های NOR در شکل (۲۱-۳ 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 (۲۱-۳ب) 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 (۳-۳) روشهای پیاده سازی NOR یا NAND خلاصه شده است . چیزی که نباید فراموش شود این است که همیشه هدف از ساده کردت 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 (۳-۳) قوانین پیاده سازی NOR و NAND

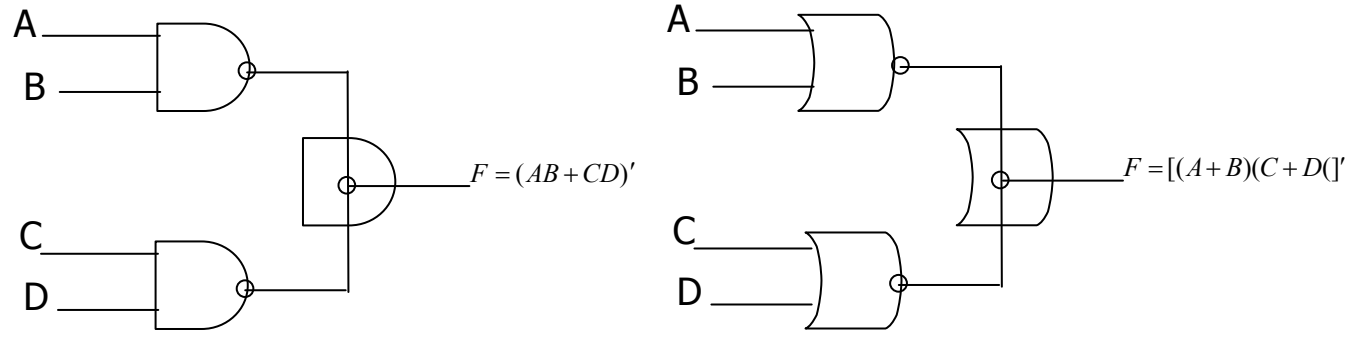
حالت	تابع جهت ساده سازی	فرم استاندارد جهت استفاده	نحوه بدست آوردن	پیاده سازی با	تعداد طبقات تا F
(الف)	F	جمع حاصلضربها	۱ها را در نقشه ترکیب کنید	NAND	۲
(ب)	F'	جمع حاصلضربها	۰ها را در نقشه ترکیب کنید	NAND	۳
(پ)	F	ضرب حاصلضربها	F' را در (ب) مکمل کنید	NOR	۲
(ت)	F'	ضرب حاصلضربها	F را در(الف) مکمل کنید	NOR	۳

۳-۷- سایر پیاده سازی های دو طبقه

گیت های موجود درمدارهای 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 منطقAND اتصالی را انجام می دهند . منطق AND اتصالی که بوسیله دو گیت NAND انجام شده در شکل (۳-۲۲ 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 (۳-۲۲ الف) پیاده سازی شده عبارتست از :

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



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

(ب)OR اتصالی در گیتهای ECL (AND-OR-INVERT)

شکل (۳-۲۲) 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 (۳-۲۲ 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 گدد . چون فقط از اتصال سیمها بوجود آمده است . ولی ، به هنگام بحث مدارهای شکل (۲۲-۳) 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 :

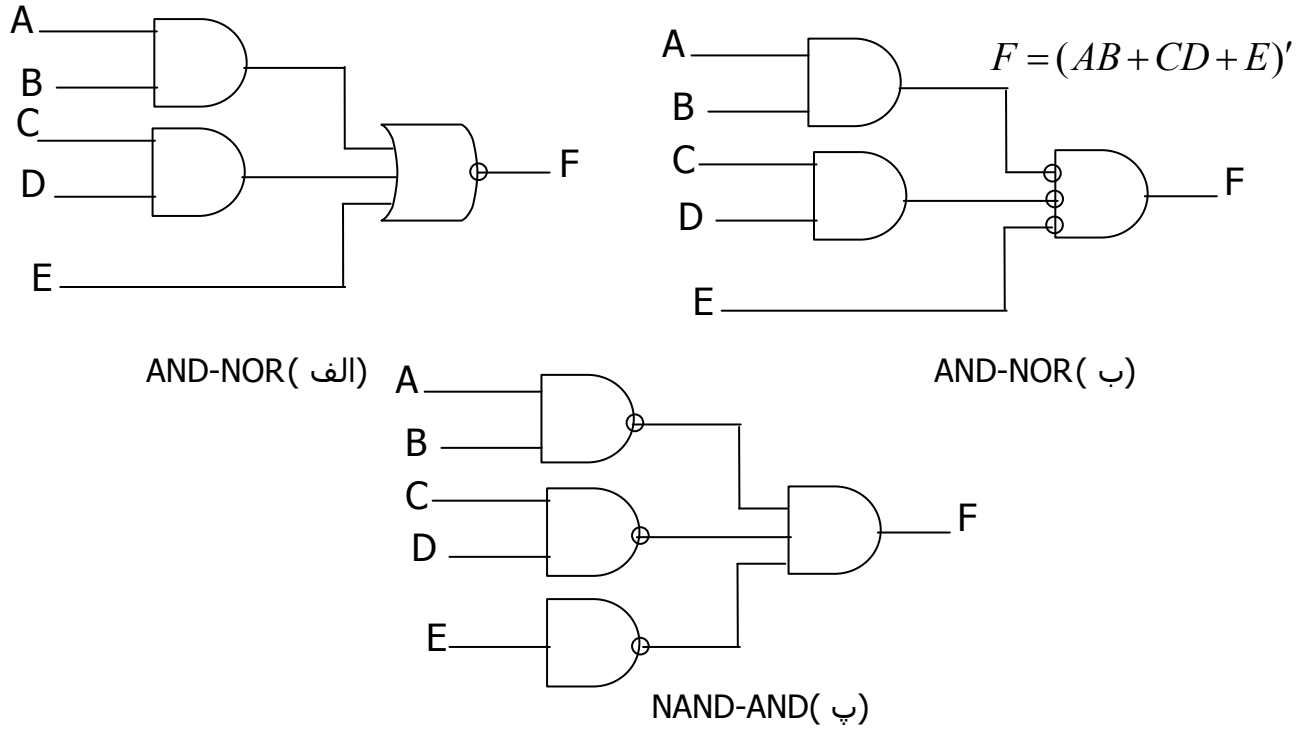
OR-AND	AND-OR
NOR-NOR	NAND-NAND
NAND-AND	NOR-OR
AND-NOR	OR-NAND

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 کید 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 ۵-۳ بحث شدند . همچنین فرمهای NAND-NAND و NOR-NOR در بخش ۶-۳ 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 یکدیگرندو می توان آنها را هم شرح داد . هر دوی آنها عمل AND-OR-INVERT را همانطور که در شکل (۲۳-۳) 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 :



شکل (۲۳-۳) مدارهای AND-OR-INVERT

بااستفاده 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.

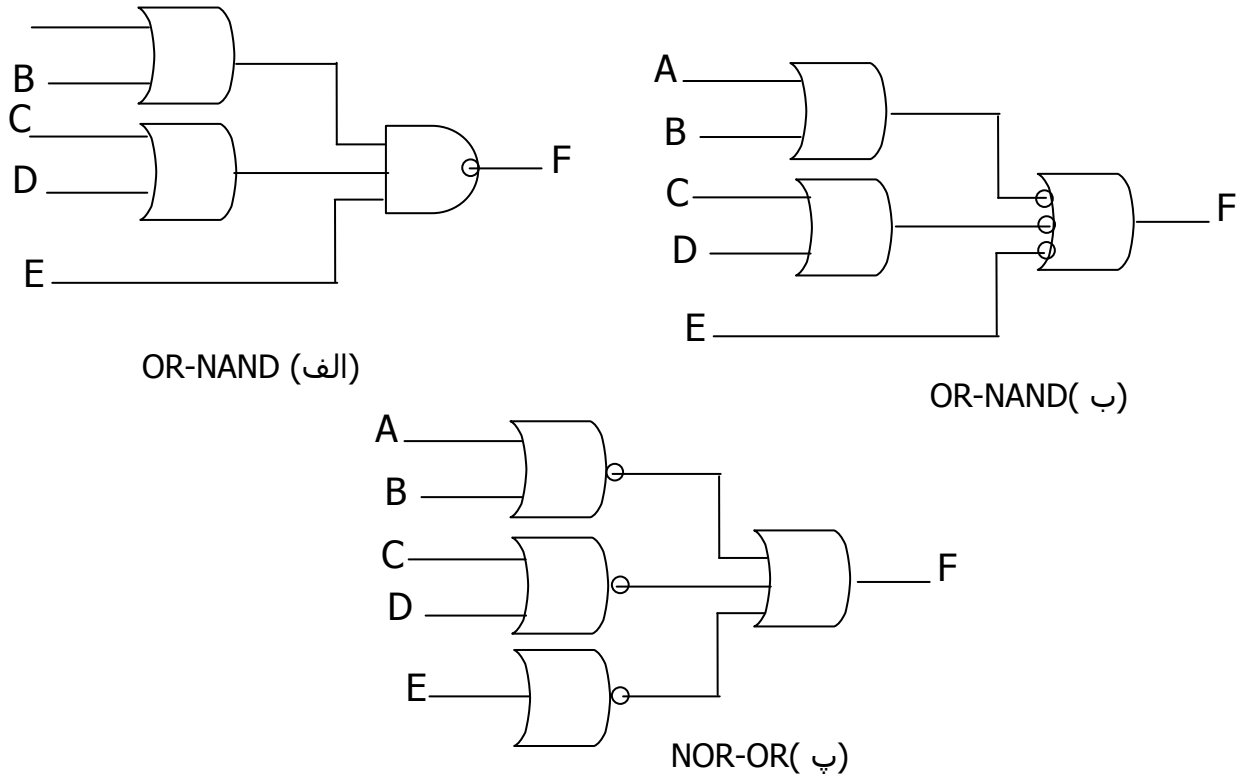
برای پیاده سازی یک AND-OR باید عبارتی ، بر حسب جمع حاصلضرب ها موجود باشد . پیاده سازی AND-OR-INVERT نیز به استثنای معکوس کردنش شبیه AND-OR است . بنابراین اگر مکمل تابع برحسب حاصلضرب ها ساده شود (بوسیله ترکیب ۰ها در نقشه) میتوان F' را بوسیله AND-OR پیاده سازی کرد و وقتی F' از قسمت معکوس کننده عبور کند تابع F تولید می شود . پیاده سازی AND-OR-INEVRT 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 وتابع زیر را پیاده می کند .

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

بااستفاده 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 A بر سحب ضرب حاصلجمع ها نیاز داریم . اگر مکمل تابع بر حسب ضرب حاصلجمع 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 .



شکل (۳-۲۳) مدارهای OR-AND-INVERT

خلاصه مطلب و مثال

در جدول (۳-۴) روشهای پیاده سازی یک تابع بول به هر چهارفرم 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 درخروجی 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 ۰ ها در جدول ساده شده عبارتست از :

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

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

خروجی از	ساده کردن F' بفرم	پیاده سازی تابع	معادل فرم مفید	
			(b)*	(a)
F	مجموع حاصلضرب ها بوسیله ترکیب ۰ ها درنقشه	AND-OR-INVERT	NAND-AND	AND-NOR
F	ضرب حاصلجمع ها با ترکیب ۱ها در نقشه و سپس مکمل سازی	ORND-INVERT-A	NOR-OR	OR-NAND

* 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 zدر 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 نیازدارد که برای بدست آوردن این عبارت می بایت ۱ ها را در نقشه با هم ترکیب کرد :

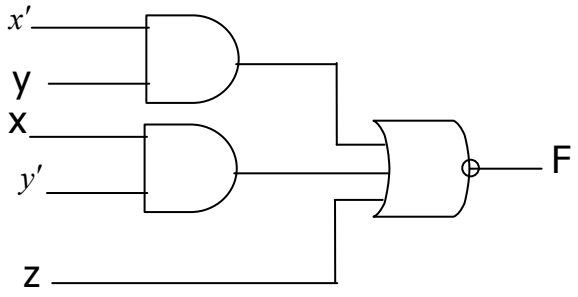
$$F = x'y'z' + xyz'$$

و سپس مکمل تابع را بدست آورد :

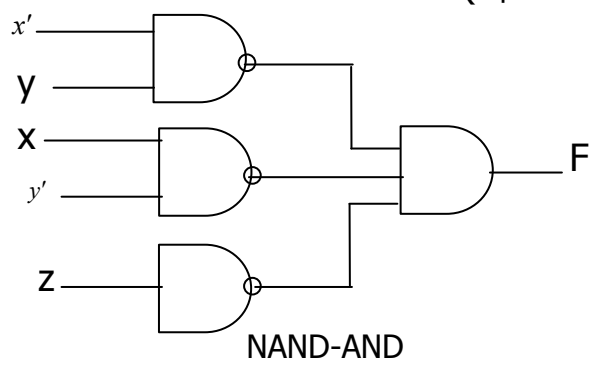
$$F' = (x + y + z)(x' + y' + z)$$

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 (۳-۲۵ ب) نشان داده شده است .

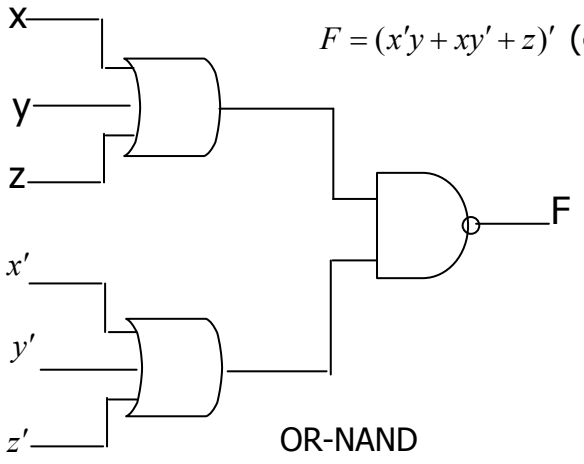


AND-NOR

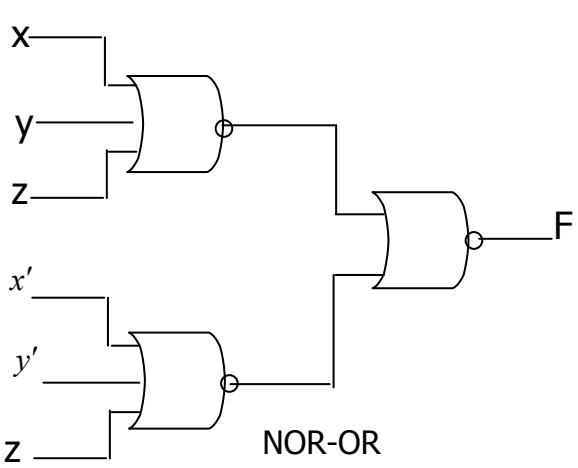


NAND-AND

$F = (x'y + xy' + z)'$ (الف)



OR-NAND



NOR-OR

$F = [(x + y + z)(x' + y' + z)]'$ (ب)

۳-۸- 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 ودر سایر مکانهای نقشه مقدارتابع 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 دهدهی 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 جه 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 نماش می دهیم . بنابراین یک X بیانگر این واقعیت ایت 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 نماییم . در ساده

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

مثال : ۳-۱۲ : 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 (۳-۲۶) 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 .

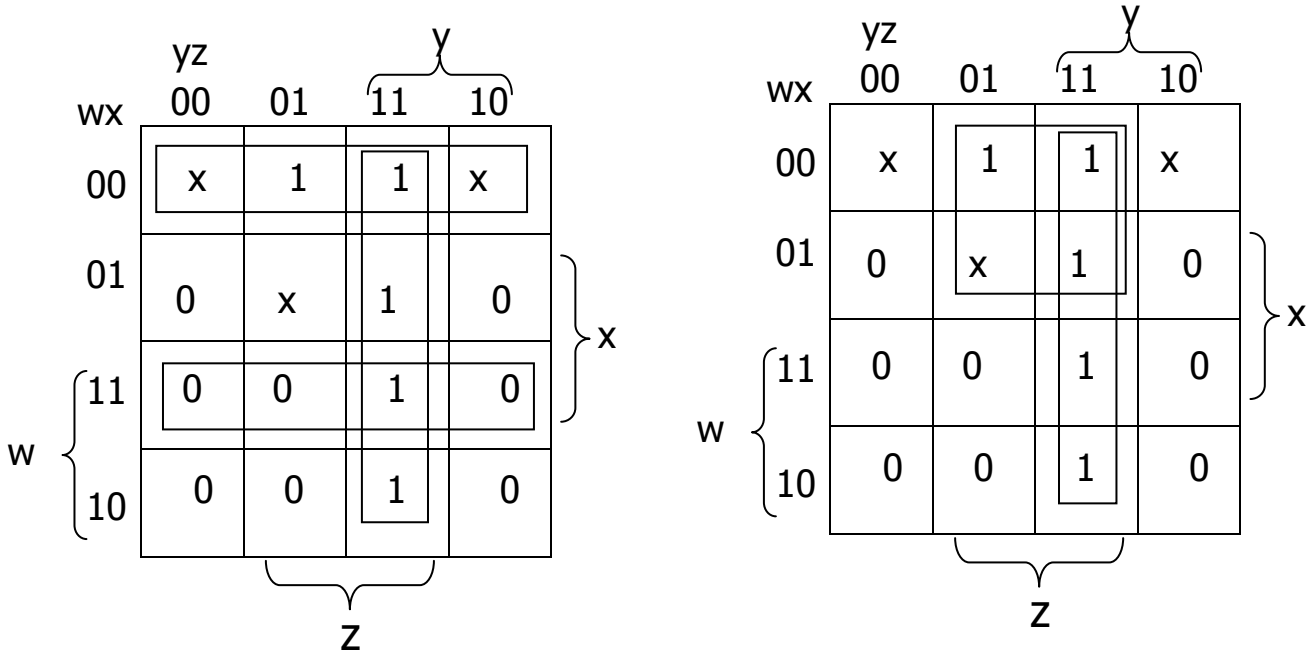
$$F = yz + w'z$$

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 بین۰ 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 .

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

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



(الف) $F = yz + w'z'$ (ب) $F = yz + w'z$

شکل (۲۶-۳) مثال مربوط به حالات بي اهميت

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 (۲۶-۳) بدست آورد .در اینحالت ، تنها راه ترکیب ۰ ها این است که مینترم های ۰ و ۲را 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) :

mi	w	x	y	z
0	0	0	0	0
11	0	0	0	1
2	0	0	1	0
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
14	0	1	1	0

$(0,1)$	$000-$
$(0,2)$	$00-0$
$\underline{(0,8)}$	$\underline{-000}$
$(2,10)$	-010
$\underline{(8,10)}$	$\underline{10-0}$
$(10,11)$	$101-$
$(10,14)$	$1-10$

$$\left.\begin{matrix}(0,2,8,10)-0-0\\(0,8,2,10)-0-0\end{matrix}\right\} x'z'$$

$$f = x'z' + w'x'y' + wx'y + wy'z$$

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
8	1	0	0	0
10	1	0	1	0
11	1	0	1	1
14	1	0	1	0
15	1	1	1	1

(0,1)	(1)
(0,2)	(2)
(0,8)	(8)
(2,8)	(8)
(8,10)	(2)

(10,11)	(1)
(10,14)	(4)
(11,15)	(4)
(14,15)	(1)

۳- 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)
(0,8)	(8)
(2,10)	(8)
(8,10)	(2)
(10,11)	(1)
(10,14)	(2)
(11,15)	(4)
(14,15)	(1)

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

	w	x	y	z	
10	1	0	1	0	
11	1	0	1	1	$= wy$
14	1	1	1	0	
15	1	1	1	1	

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

	w	x	y	z	
0	0	0	0	0	
2	0	0	1	0	$= x'z'$
8	1	0	0	0	
10	1	0	1	0	

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)
4	(4,6)	(2)
8	(8,9)	(1)
6	(8,10)	(2)
9	(8,10)	(1)
10	(9,11)	(2)
7	(10,11)	(1)
11	(7,15)	(8)
15	(11,15)	(4)

(8,	0,	10,	11)	(1,2)
(8,	10,	9,	11)	(2,1)
0	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

$$PI_1 = wx'$$
$$PI_1 = x'y'z$$
$$PI_3 = w'xz'$$
$$PI_4 = w'xy$$
$$PI_5 = xyz$$
$$PI_6 = wyz$$

1	(1,9)	(8)
4	(4,6)	(2)
8	(8,9)	(1)
6	(8,10)	(2)
9	(8,10)	(1)
10	(9,11)	(2)
7	(10,11)	(1)
11	(7,15)	(8)
15	(11,15)	(4)

(8,	0,	10,	11)	(1,2)
(8,	10,	9,	11)	(2,1)
0	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

$PI_1 = wx'$
$PI_1 = x'y'z$
$PI_3 = w'xz'$
$PI_4 = w'xy$
$PI_5 = xyz$
$PI_6 = wyz$

روش پیدا کردن اسامیها :

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

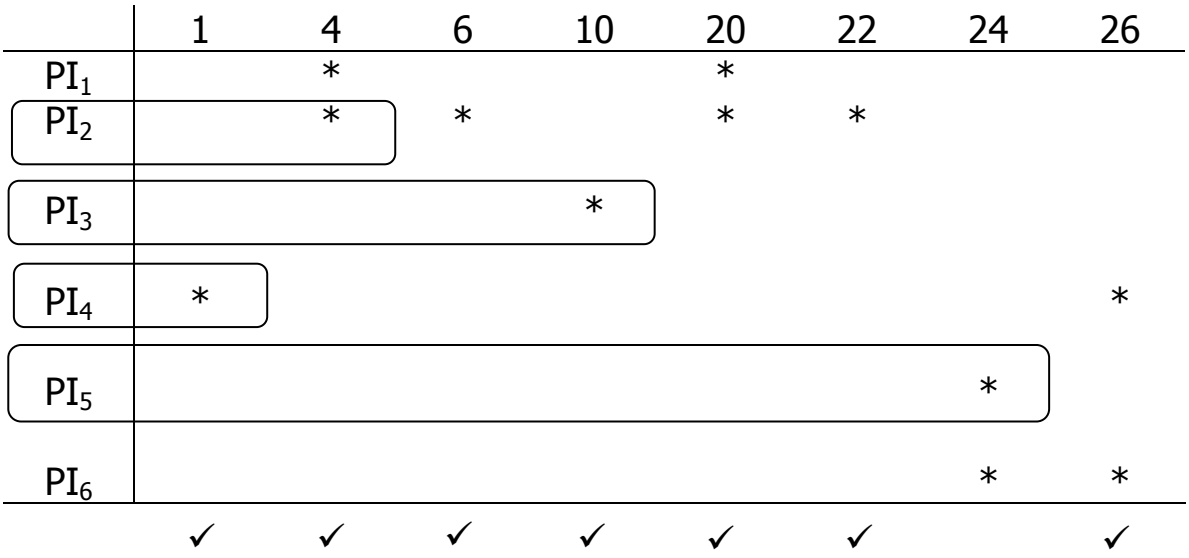
	1	4	6	7	8	9	10	11	15
PI_1					*	*	*	*	
PI_2	*					*			
PI_3		*	*						
PI_4		*	*						
PI_5			*						*
PI_6								*	*
	✓	✓	✓	□	✓	✓	✓	✓	□

	7	15
PI_4	*	
PI_5	*	*
PI_6		*

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

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

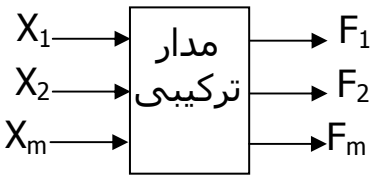


در ماکسترم : درماکسترم بر حسب تعداد صفرها بررسی می شود در ترکیب دو دسته ای باید ماکسترم بالایی بزرگتر باشد . در نوشتن PI ها ما جملات حاصل جمع داشته و قوانین ماکسترم ها را رعایت می کنیم نهایتاً f همان حاصلضرب PI ها می شود .

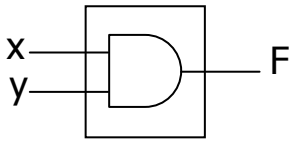
تمرین : تابع مقابل را ساده نمایید. $F = \pi M\,(0,1,2,3,4,6,10,11,12,15)$

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- SSI (Small Scale Integrated Circuits)

مدارهای ترکیبی مقیاس کوچک مانند :

AND OR NOT XOR

۲- MSI (Medium OR NOT XOR)

مدارهای ترکیبی مقیاس متوسط مانند :

- مدارهای مبدل کد ها
- مدارهای جمع کننده / تفریق کننده
- مالتی پلکسرها
- دیکودرها

۳- LSI (Large Scale Integrated Circuits)

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

طراحی مدارهای ترکیبی

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

۱- نحوه ي 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

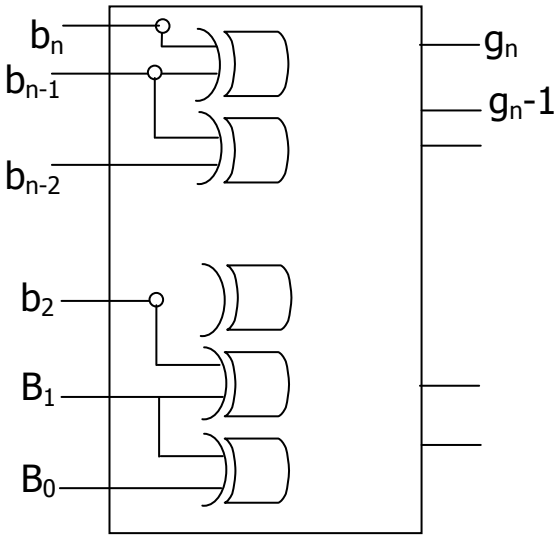
۷- پیاده سازی

دسته بندی مدارهای ترکیبی

الف) مدارهای مبدل کد :

جهت تبدیل کدهای مختلف به یکدیگر

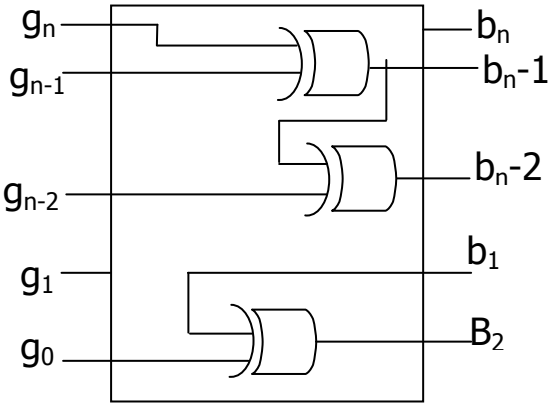
۱- **مدار مبدل کد باینری به گری** :



$$g_n = b_n \qquad g_i = b_i \oplus b_{i+1} \qquad 0 \le i \le n-1$$

تاخیر به اندازه n-1 طبقه

۲- مدار مبدل گری به باینری :

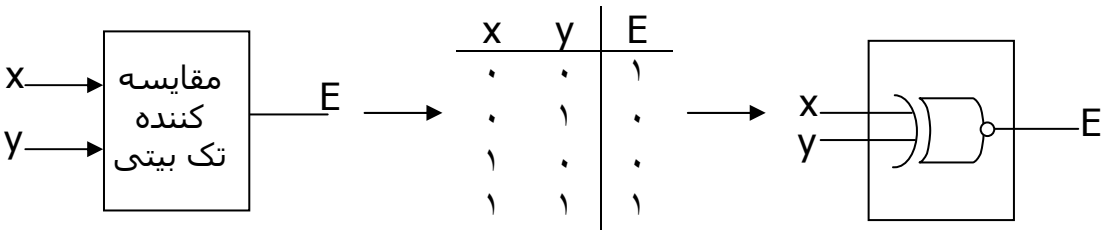


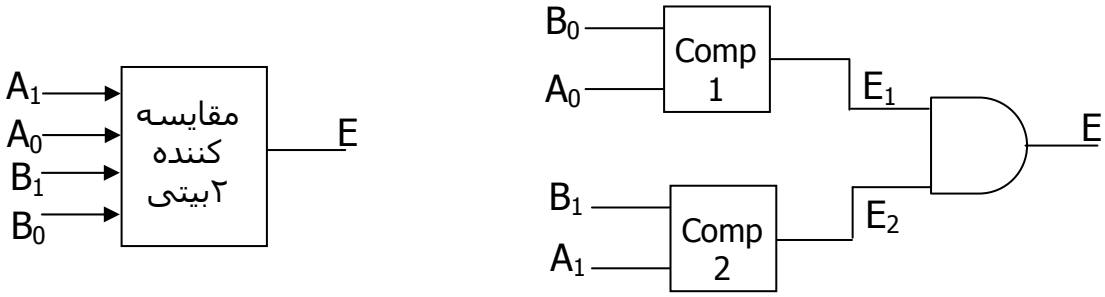
$$b_n = g_n \qquad b_i = g_i \oplus b_{i+1} \qquad 0 \le i \le n-1$$

« تاخیر تجمعی به اندازه n طبقه »

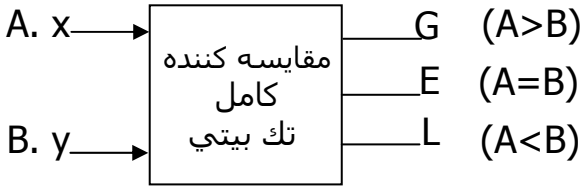
ب) مدارهای مقایسه کننده :

برای مقایسه اطلاعات ورودی به کار می رود .





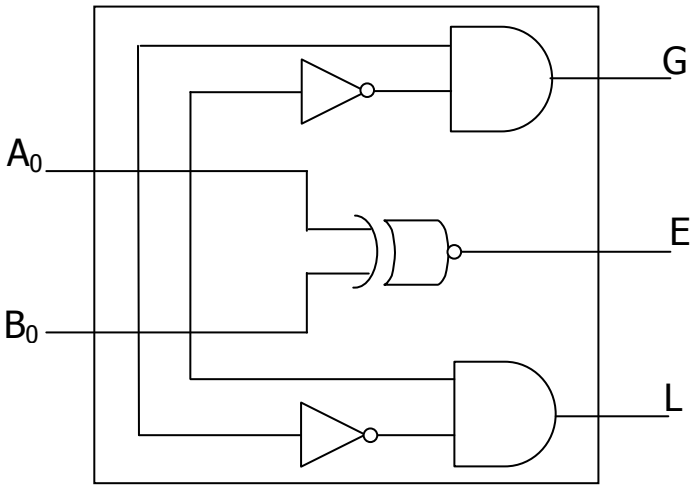
مدار مقایسه کننده کامل : ۱- ابتدا عملکرد براي يك بيت بررسي مي شود.

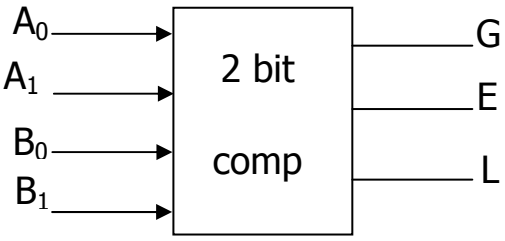


۲- بعد جدول صحت را تنظیم کنید .

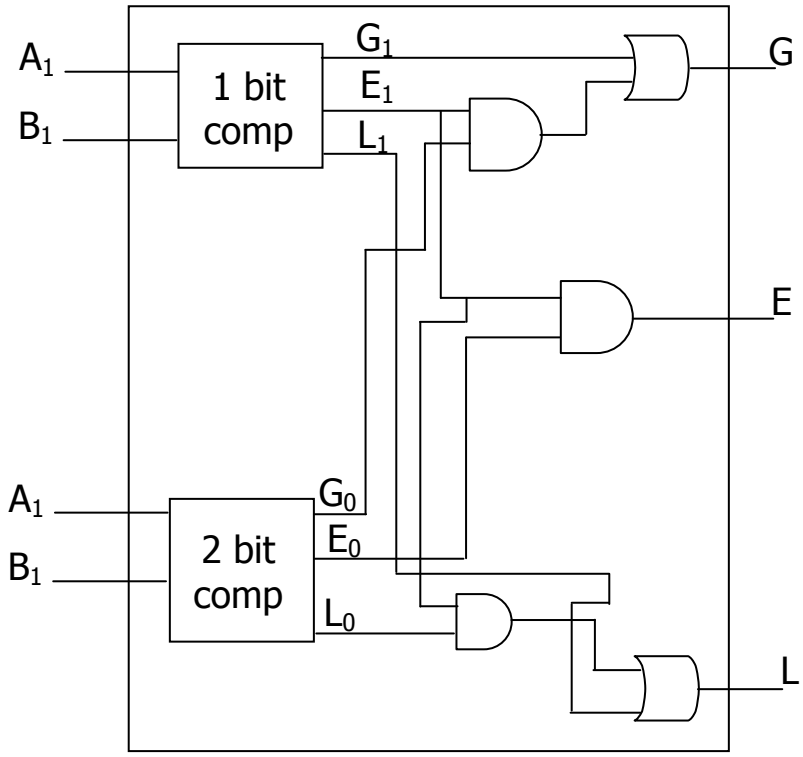
A_0	B_0	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

۳- مدار را ترسیم کنید .

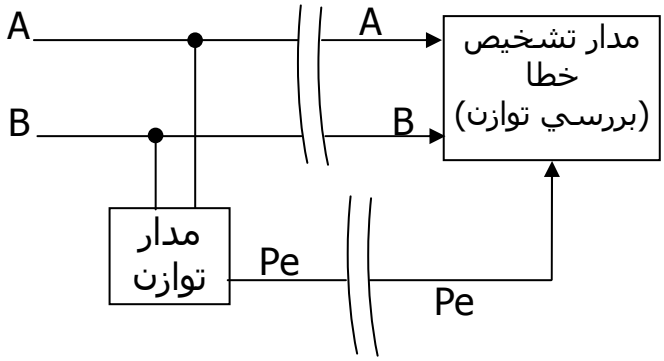




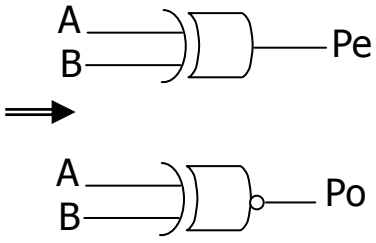
$$A: A_1\ A_0$$
$$B: B_1\ B_1$$
$$E = (A_1 \oplus B_1)\,o(A_0 \oplus B_0)$$
$$G = (A_1 > B_1) + (A_1 = B_1)\,o(A_0 > B_0)$$
$$\downarrow$$
$$G = A_1 B_1' + (A_1 \oplus B_1)\,o(A_0 B_0') = (G_1 + E_1 G_0$$
$$L = (A_1 < B_1) + (A_1 = B_1)\,o(A_0 < B_0)$$
$$\downarrow$$
$$L = (A_1' B_1 + (A_1 \oplus B_1)(A_0' B_0) = L_1 + E_1\,L_0$$



ب) مدارهاي توليد توازن و تشخيص:



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

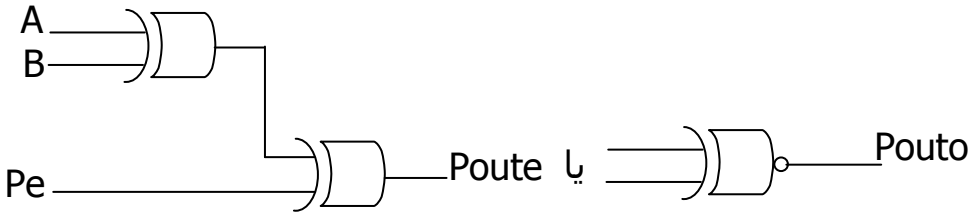


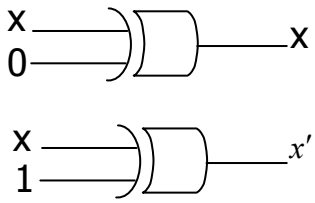
A	B	Pe	Poute
۰	۰	۰	۰
۰	۰	۱	۱
۰	۱	۰	۱
۰	۱	۱	۰
۱	۰	۰	۱
۱	۰	۱	۰
۱	۱	۰	۰
۱	۱	۱	۱

A \ Bpe	۰	۱	۰	۱
۰				
۱				

$$Poute = A \oplus B \oplus p_e$$

$$Poute = [(A \oplus B) \oplus p_e]' = (A \oplus B) \oplus P$$ ۱





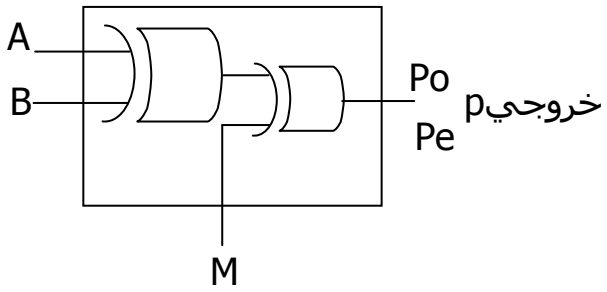
$$F = x.1 + x'.0 = x$$

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

نكته :

هرگاه يكي از ورودیهاي xor ، ۰ باشد خروجي برابر با ورودي ديگر است و اگر ۱ باشد برابر با متمم خروجي ديگر است .

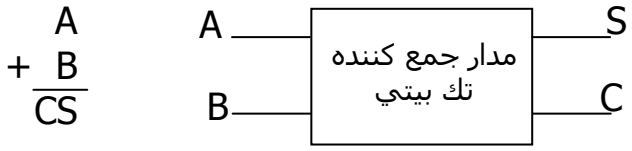
هدف : طراحي مداري كه با يك بيت كنترل توازن زوج يا فرد ايجاد مي كند .



$$iF \quad M=0 \, then \, P=pe \;,\; P=A\oplus B$$
$$iF=M=1 \, then \, P=p0 \;,\; P=A\oplus B$$

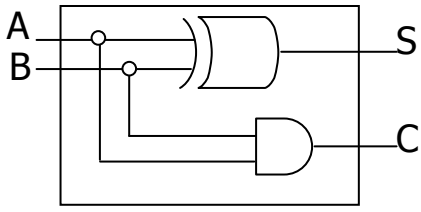
پ) مدارهاي جمع كننده / تفريق :

۱- نيم-جمع كننده



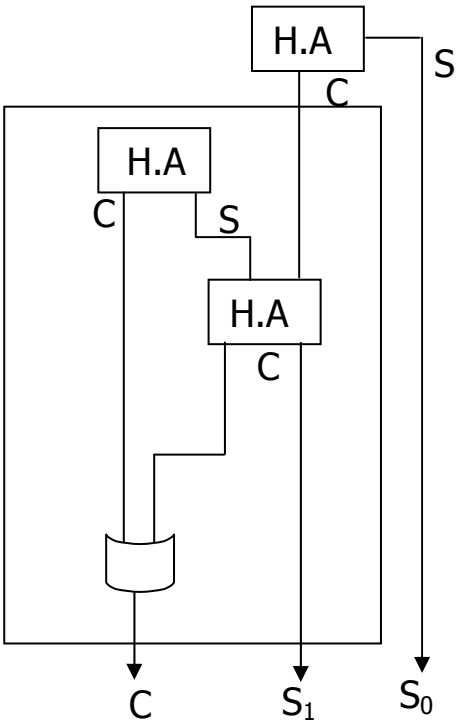
A	B	C	S
۰	۰	۰	۰
۰	۱	۰	۱
۱	۰	۰	۱
۱	۱	۱	۰

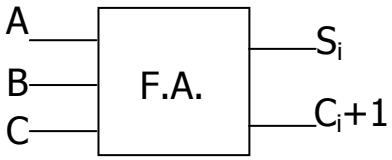
$$\begin{cases} S=A\oplus B \\ C=A.B \end{cases}$$



Half Addes (H.A) یا نیم جمع کننده

۲- مدار تمام- جمع كننده





۰۱
+ ۱۱
۱۰۰
$C_2\ S_1\ S_1$

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	۱

A
B
+ C
CS

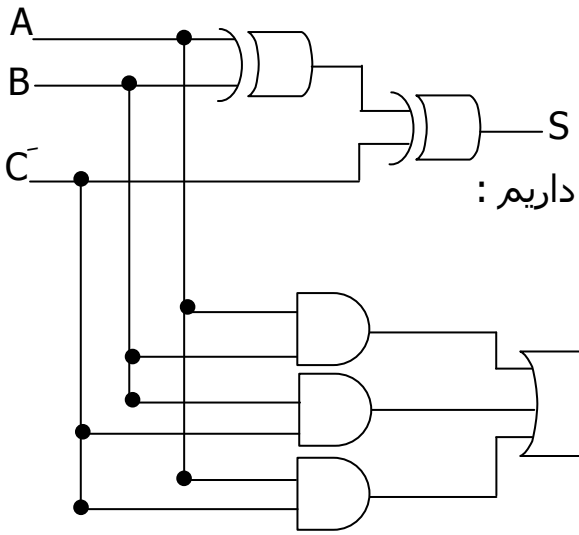
A \ BC	۰۰	۰۱	۱۱	۱۰
۰	۰	۱	۰	۱
۱	۱	۰	۱	۰

(S)

A \ BC	۰۰	۰۱	۱۱	۱۰
۰	۰	۰	۱	۰
۱	۰	۱	۱	۱

(C)

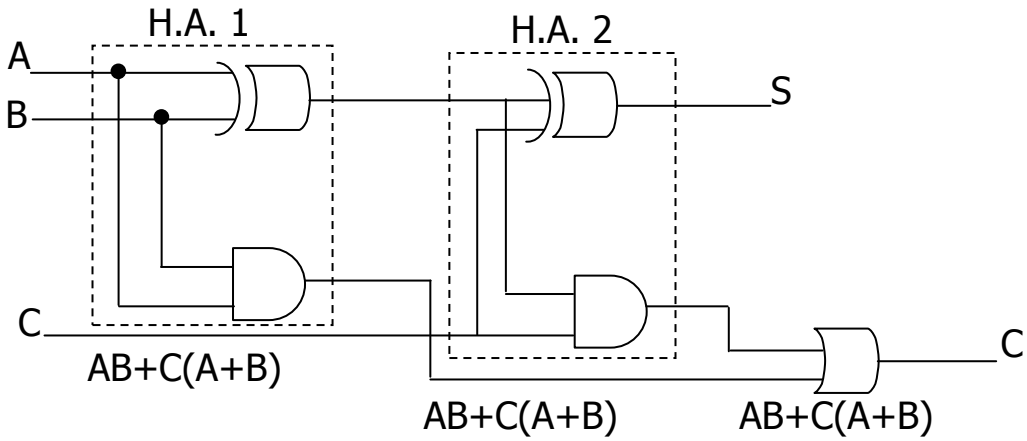
$$\Longrightarrow \begin{aligned} S &= \sum m(1,2,4,7) = A \oplus B \oplus C \\ C &= \sum m(3,5,6,7) = AB \oplus BC \oplus AC \end{aligned}$$



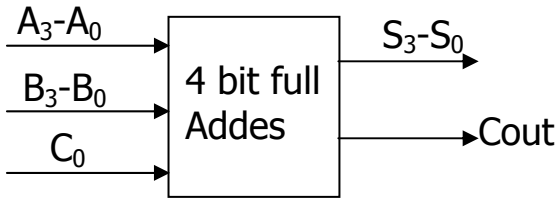
به جاي اين مدار شلوغ مدار ساده تر زير را داريم :

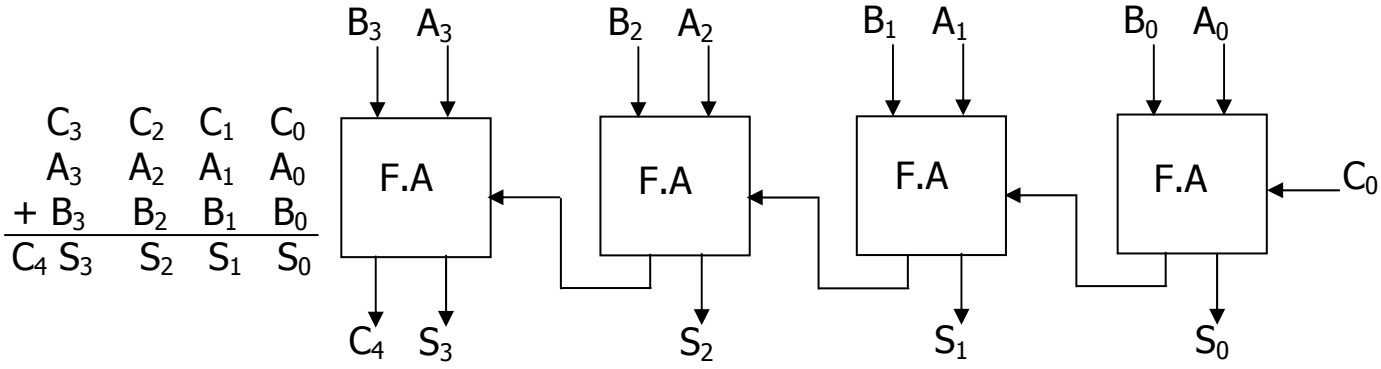
$$\begin{aligned} Cout &= AB + BC + AC \\ &= AB + C(A + B) \end{aligned}$$

لذا :

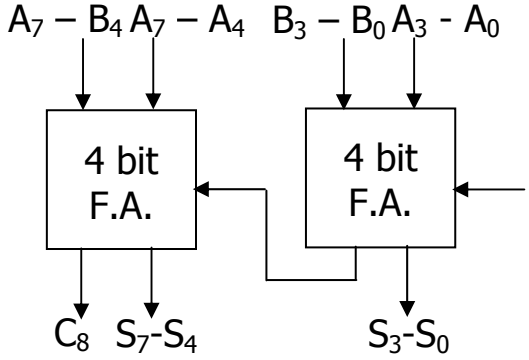


۳- جمع کننده کامل ۴ بيتي :

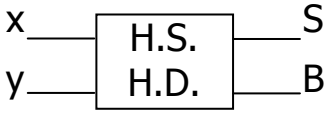




: 8 bit Full Adder



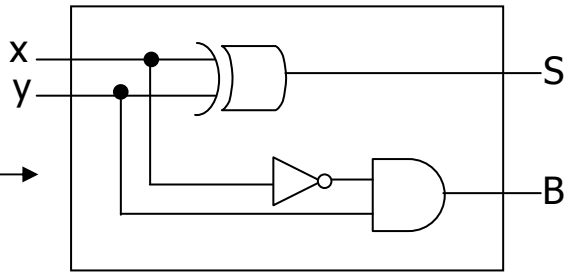
۴- مدار تفریق کننده :



x
-y
BS

X	Y	B	S
۰	۰	۰	۰
۰	۱	۱	۱
۱	۰	۰	۱
۱	۱	۰	۰

$S = x \oplus y$
$B = x'y$



X	Y	Z	B	S
۰	۰	۰	۰	۰
۰	۰	۱	۱	۱
۰	۱	۰	۱	۱
۰	۱	۱	۱	۰
۱	۰	۰	۰	۱
۱	۰	۱	۰	۰
۱	۱	۰	۰	۰
۱	۱	۱	۱	۱

x
-y
-z
BS

S =

x \ yz	۰۰	۰۱	۱۱	۱۰
۰	۰	۱	۰	۱
۱	۱	۰	۱	۰

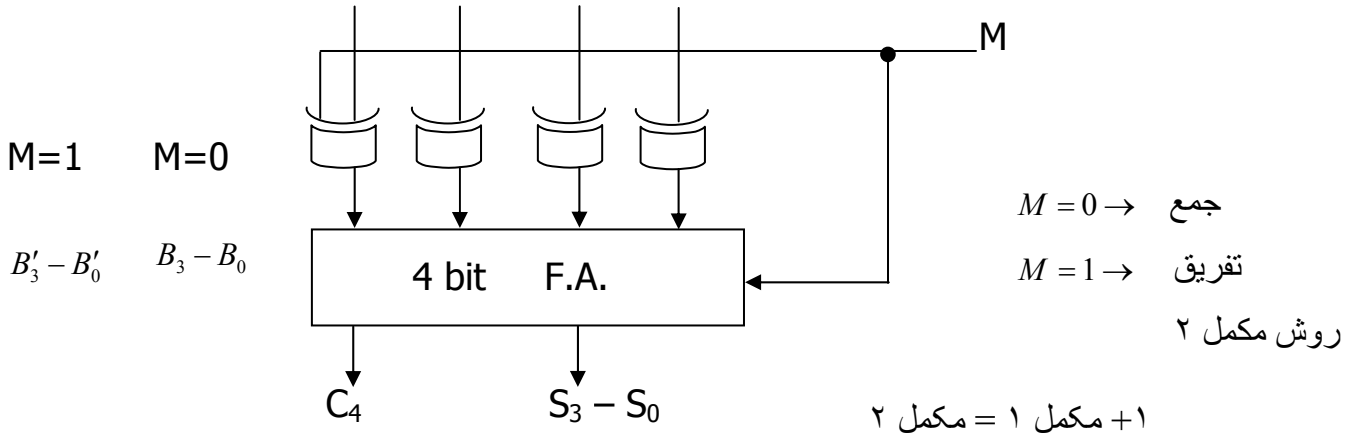
B =

x \ yz	۰۰	۰۱	۱۱	۱۰
۰	۰	۱	۱	۱
۱	۰	۰	۱	۰

$S = x \oplus y \oplus z$

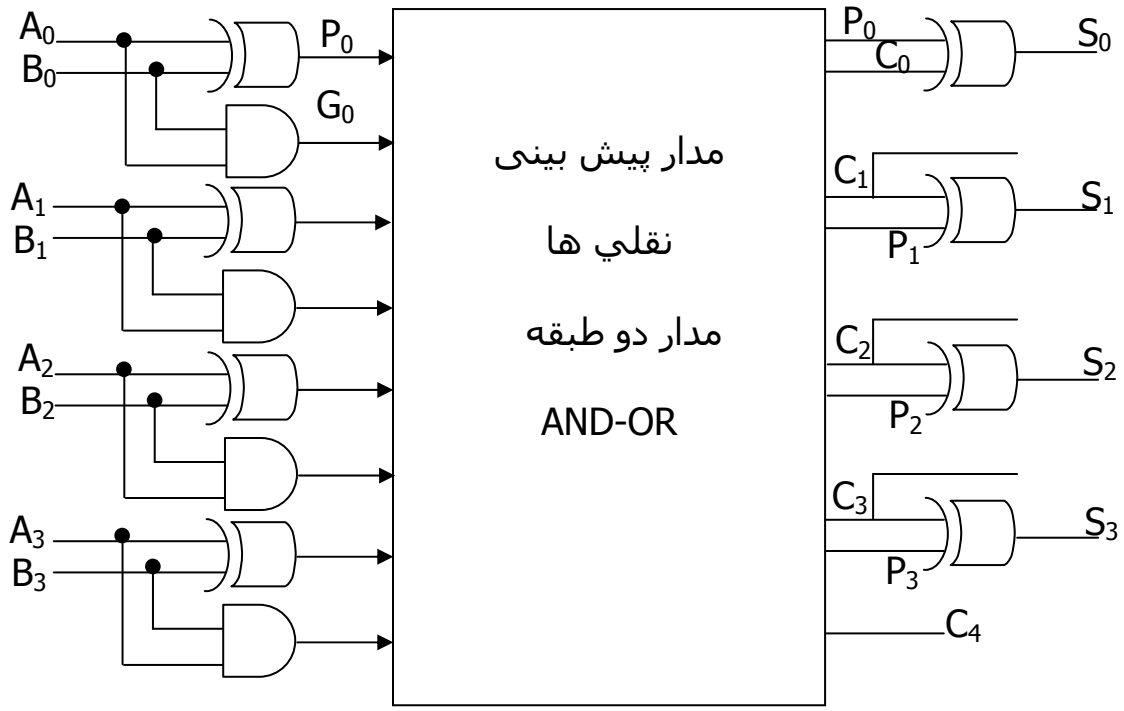
$B = x'z + x'y + yz$

۵- طراحی مدار جمع کننده / تفریق کننده ی ۴ بیتی با روش مکمل ۲ با خط کنترل M:



نکته : در مدارهای جمع کننده تاخیر تجمعی داریم که با افزایش تعداد بیت های ورودی افزایش می یابد . برای مدار جمع کننده ی ۲ بیتی ۵ طبقه تاخیر داشتیم پس این مدار مشکل دارد پس مدار جمع کننده با نقلی پیش بینی شده طراحی شد .

۶- مدار جمع کننده با نقلی پیش بینی شده (Look Ahead Carry Generator) :



$$S_0 = G_0 \oplus P$$
$$C_1 = G_0 + C_0 P_0$$
$$S_1 = C_1 \oplus P_1$$
$$C_2 = G_1 + C_1 P_1$$
$$C_2 = G_1 + P_1 G_0 + P_1 P_2 C_0$$
$$S_2 = G_2 \oplus P_2$$
$$C_3 = G_2 + C_2 P_2$$
$$C_3 = G_2 + P_2 G_1 + P_1 P_2 G_1 + P_1 P_2 P_0 G_0$$

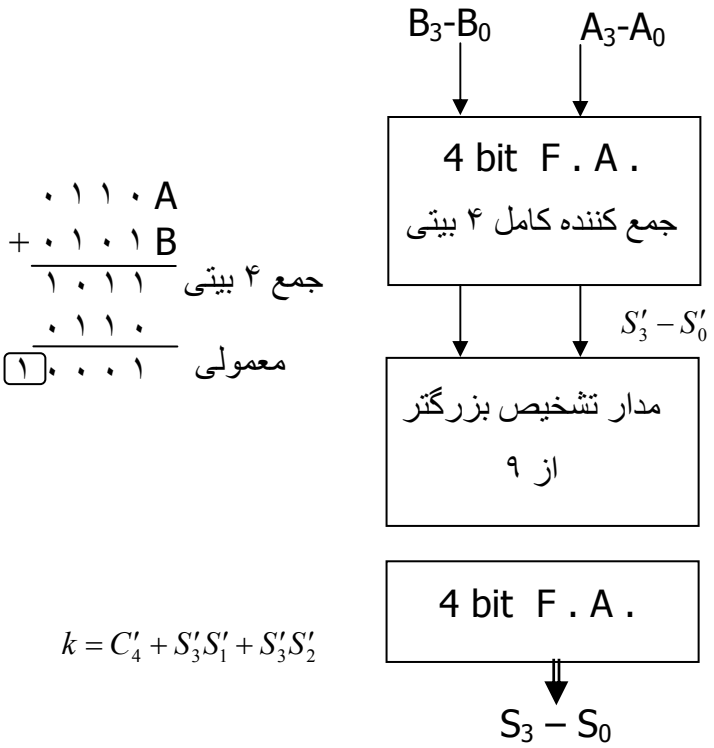
$$P_i = A_i \oplus B_i$$
$$B_i = A_i \,.\, B_i$$
$$S_i = P_i \oplus C_i$$
$$C_i = G_{i-1} + C_{i-1} P_{i-1}, C_0$$

داریم

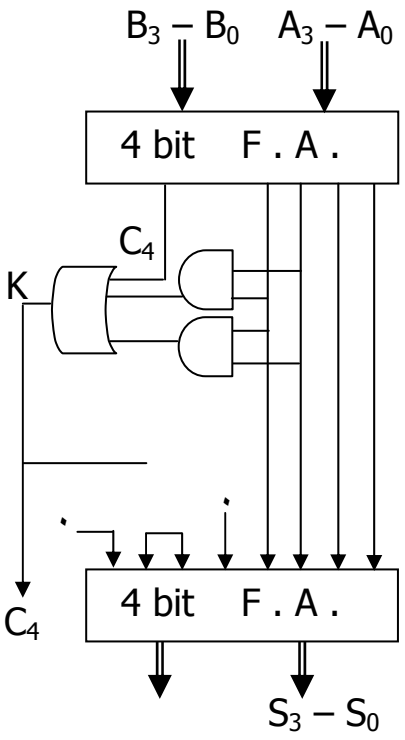
نکته : این مدار و کلیه مدارهای توسعه یافته (6,8) و ... بیتی) می توانند با این روش با 4 طبقه تاخیر طراحی شوند .

۷- مدار جمع کننده BCD :

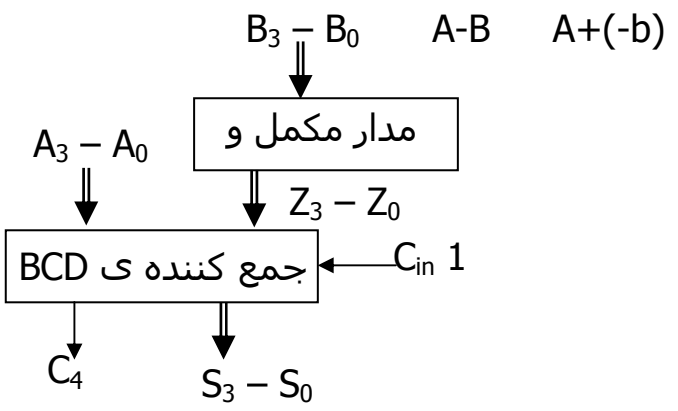
C_4'	S_3'	S_2'	S_1'	S_0'	K
۰	۰	۰	۰	۰	۰
۰	۰	۰	۰	۱	۰
۰	۰	۰	۱	۰	۰
۰	۱	۰	۰	۱	۰
۰	۱	۰	۱	۰	۱
۰	۱	۰	۱	۱	۱
۱	۰	۰	۰	۰	۱
۱	۰	۰	۰	۱	۱
۱	۰	۰	۱	۰	۱
۱	۰	۰	۱	۱	*



پیاده سازی :



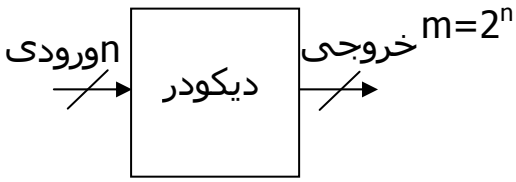
۸- مدارهای تفریق کننده ی BCD :

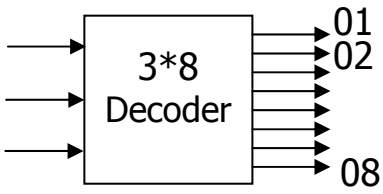


B3	B2	B1	B0	Z3	Z2	Z1	Z0
۰	۰	۰	۰	۱	۰	۰	۱
۰	۰	۰	۱	۱	۰	۰	۰
۰	۰	۱	۰	۰	۱	۱	۱
۰	۰	۱	۱	۰	۱	۱	۰
۰	۱	۰	۰	۰	۱	۰	۱
۰	۱	۰	۱	۰	۱	۰	۰
۰	۱	۱	۰	۰	۱	۱	۱
۰	۱	۱	۱	۰	۰	۱	۰
۱	۰	۰	۰	۰	۰	۰	۱
۱	۰	۰	۱	۰	۰	۰	۰

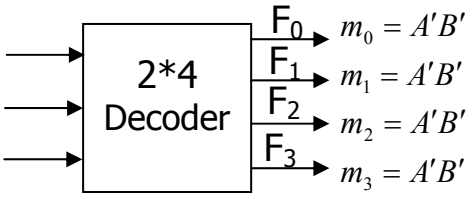
۹- مدارهای دیکودر (Decoder) :

مدار دیکودر برای n متغیر ورودی 2^n حالت آنرا ایجاد می نماید.

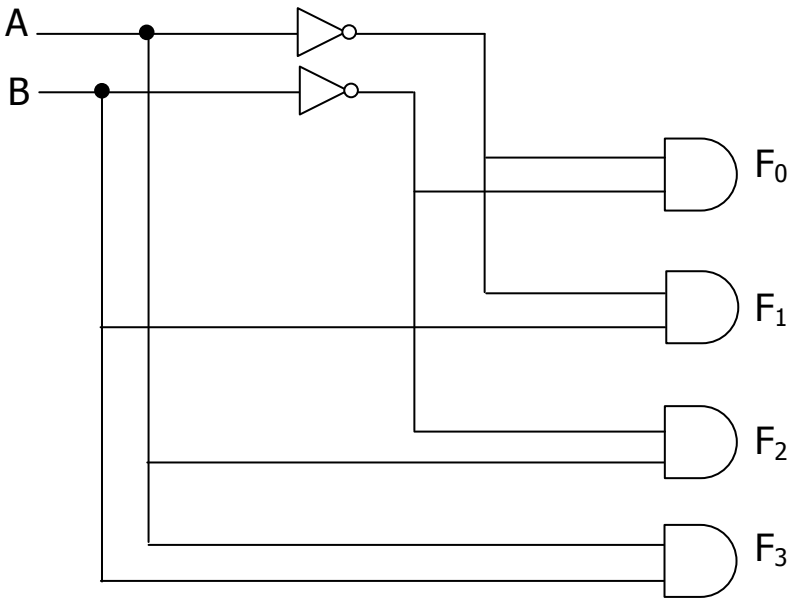




نکته : حالات مختلف متغیرها را در خروجی می دهد .



A	B	F0	F1	F2	F3
۰	۰	۱	۰	۰	۰
۰	۱	۰	۱	۰	۰
۱	۰	۰	۰	۱	۰
۱	۱	۰	۰	۰	۱



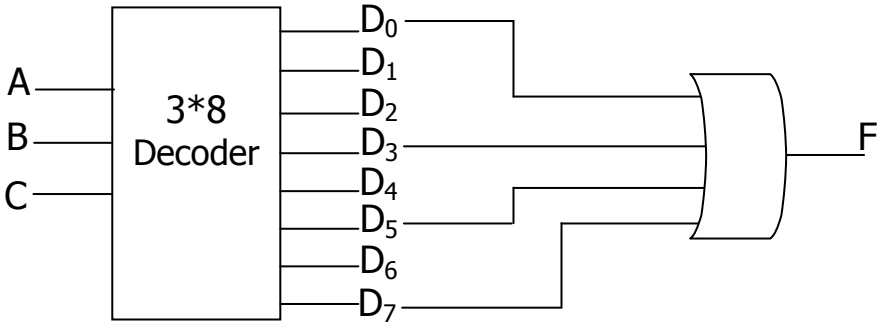
نکته : اگر به جای AND از NAND استفاده کنیم آنگاه Maxterm ها خواهیم داشت :

کاربردها :

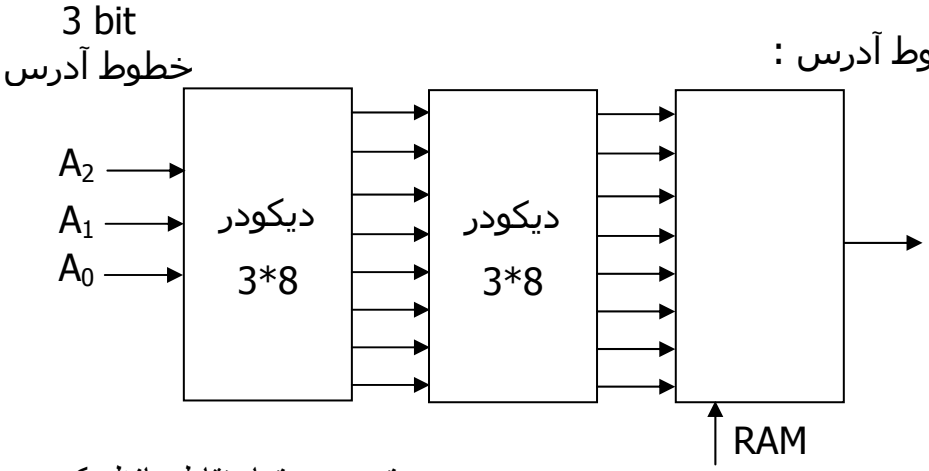
۱- در پیاده سازی توابع :

$$F = (A,B,C) = \sum m(0,3,5,7)$$

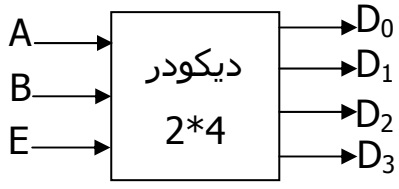
تابع را به کمك يك ديكودر 3*8 پياده سازی نمائيد .

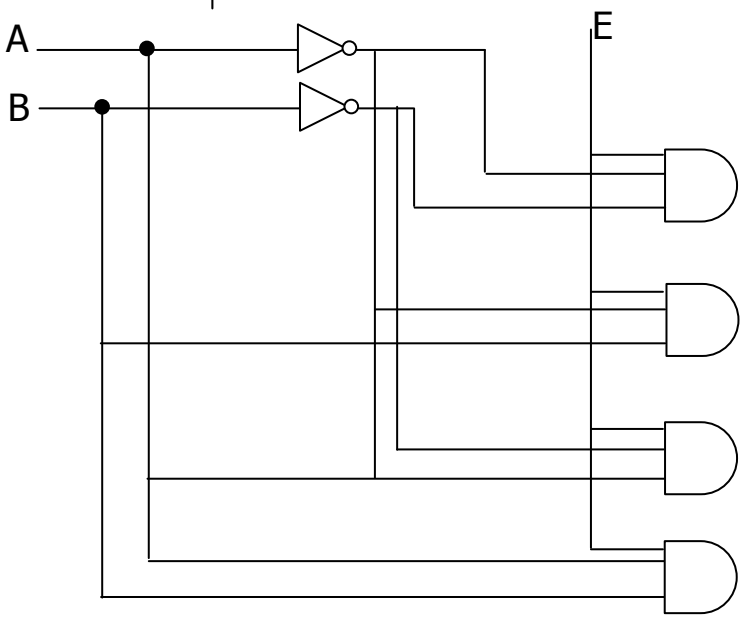


۲- در رمز گشايي خطوط آدرس :



دسترسي به تمام نقاط حافظه كه در اين مثال 8 حافظه است .





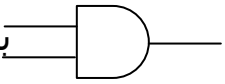
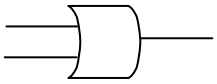
دیکودر عمل نمي كند E=0

Enable

Active High enable

E	A	B	D_0	D_1	D_2	D_3
۰	*	*	۰	۰	۰	۰
۱	۰	۰	۱	۰	۰	۰
۱	۰	۱	۰	۱	۰	۰
۱	۱	۰	۰	۰	۱	۰
۱	۱	۱	۰	۰	۰	۱

نکته : در حالت ماکسترمی نیز درصورتی کـه E=0 ، B,A نیـز Den't care مـی باشـد .

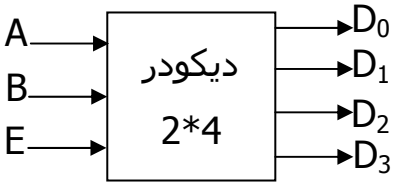
در صورتی که به جای  ،  بگـذاریم جـدول زیـر را خـواهیم داشـت .

E	A	B	D_0	D_1	D_2	D_3
1	*	*	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

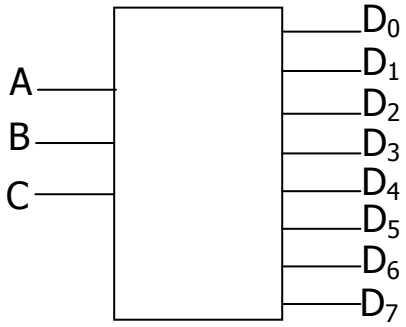
Enable

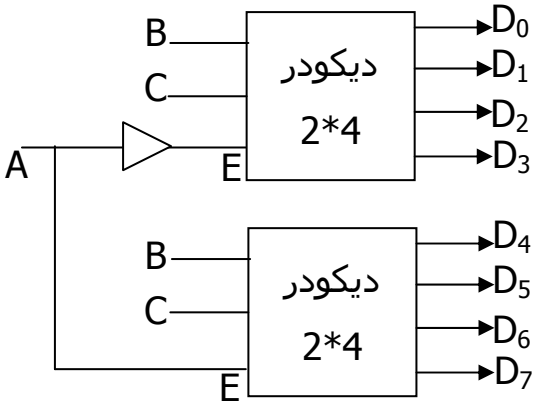
Active High enable

يعني Enable بايد NOT شـود به صورت زير



با كمك ديكودر هاي 2*4 ، ديكودر 3*8 بسازيد .



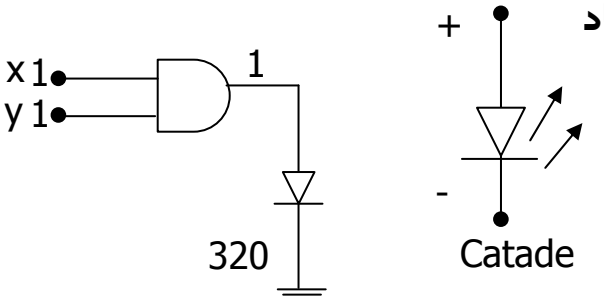


E	A	B	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
۰	۰	۰	۱	۰	۰	۰	۰	۰	۰	۰
۰	۰	۱	۰	۱	۰	۰	۰	۰	۰	۰
۰	۱	۰	۰	۰	۱	۰	۰	۰	۰	۰
۰	۱	۱	۰	۰	۰	۱	۰	۰	۰	۰
۱	۰	۰					۱			
۱	۰	۱						۱		
۱	۱	۰							۱	
۱	۱	۱								۱

دیکودر BCD به 7.seg (Seven Segment)

قطعه الکترونیکي LED : (دیود نوري) با نماد

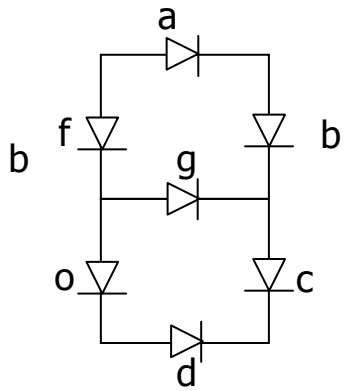
(Light Emitting Diode)



اگر [v]2 به دو سر LED اختلاف پتانسیل اعمال شود روشن مي شود .

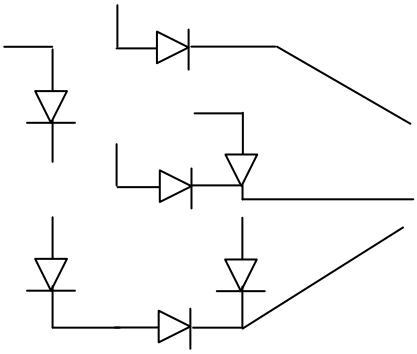
در نتیجه :

7.seg صفحه نمایش یا خروجي یك سیستم دیجیتال مي توانند باشد .



نکته :

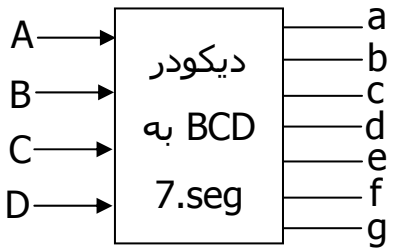
اگر کاتدها را به هم وصل کنیم و از طریق آنها فرمان بدهیم مي گویند (کاتد مشترك 7.seg) و اگر آن را به هم متصل و در کل به جز اي که مي خواهیم روي آن کار انجام دهیم وصل کنیم و از طریق کاتدها فرمان دهیم (آنرا مشترك 7.seg)



کاتد مشترک

A	B	C	D		a	b	c	d	e	f	g
۰	۰	۰	۰	0	۱	۱	۱	۱	۱	۱	۰
۰	۰	۰	۱	1	۰	۱	۱	۰	۰	۰	۰
۰	۰	۱	۰	2	۱	۱	۰	۱	۱	۰	۱
۰	۰	۱	۱	3	۱	۱	۱	۱	۰	۰	۱
۰	۱	۰	۰	4	۰	۱	۱	۰	۰	۱	۱
۰	۱	۰	۱	5	۱	۰	۱	۱	۰	۱	۱
۰	۱	۱	۰	6	۰	۰	۱	۱	۱	۱	۱
۰	۱	۱	۱	7	۱	۱	۱	۰	۰	۰	۰
۱	۰	۰	۰	8	۱	۱	۱	۱	۱	۱	۱
۱	۰	۰	۱	9	۱	۱	۱	۱	۰	۱	۱

باقي مانده Don't car هستند

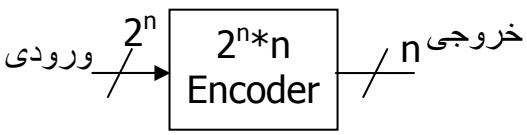


کاتد مشترک است

نکته :

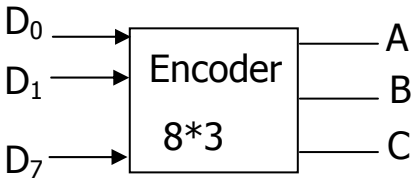
۱۰- مدارهای Encoder یا رمز گذار :

مدار انکودر برای رمزگذاری یا فشرده سازی متغیرهای ورودی بکار می رود.

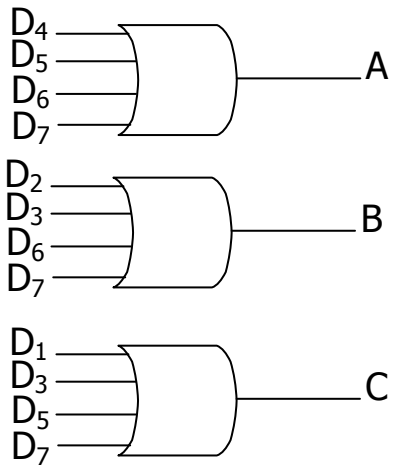


ورودی	خروجی
7	3
11	4
16	4

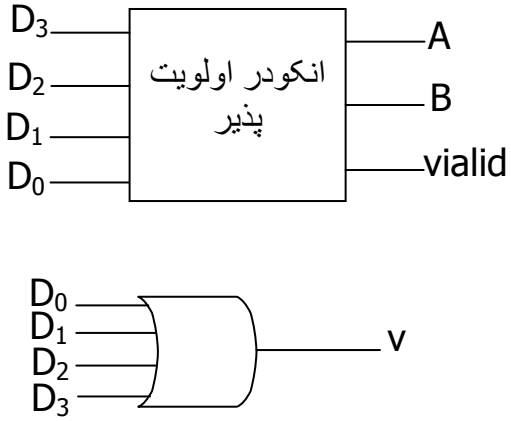
حالت متعارف



D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1								۰	۰	۰
	1							۰	۰	۱
		1			0			۰	۱	۰
			1					۰	۱	۱
				1				۱	۰	۰
		0			1			۱	۰	۱
						1		۱	۱	۰
							1	۱	۱	۱



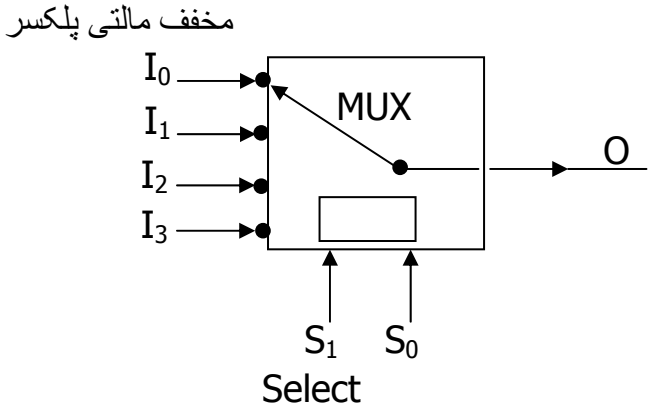
D_3	D_2	D_1	D_0	A	B	V
۱	*	*	*	۱	۱	۱
۰	۱	*	*	۱	۰	۱
۰	۰	۱	*	۰	۱	۱
۰	۰	۰	۱	۰	۰	۱
۰	۰		۰	۰	۰	۰



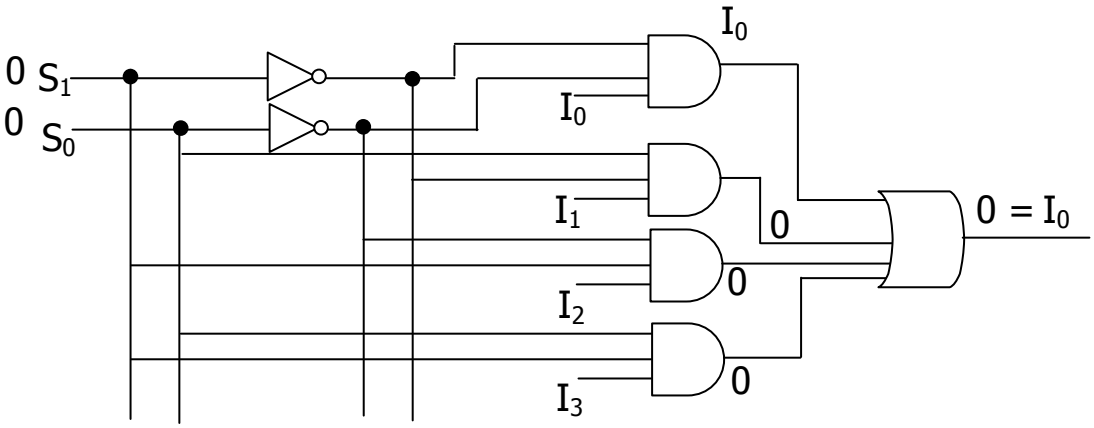
$$A = D_3 + D_2 + D_3' = (D_3 + D_2)(D_3 + D_3') = D_3 + D_2$$
$$B = D_3 + D_3'D_2'D_1 = D_3 + D_2'D_1$$

کاربرد : اولویت گذاري Interrupt مانند وقفه کیبورد ، سدی درایو ، کارت صوتي و VGA

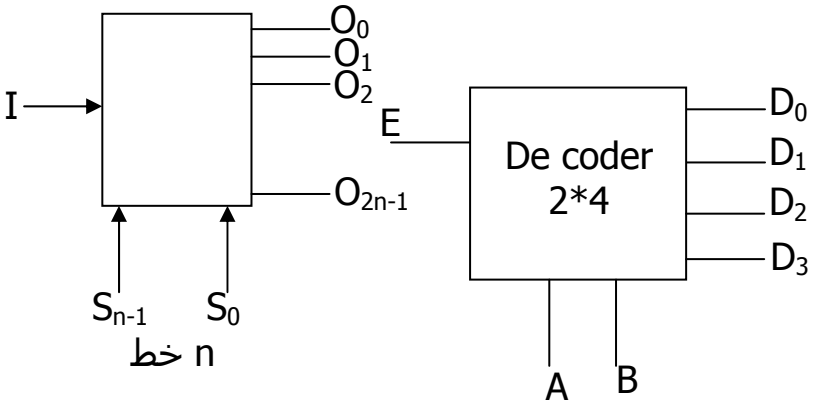
Multiplexer : با کمك n انتخاب 2^n ورودی را روی یک خروجی می فرستد .



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



۱۱- مدارهای ترکیبی – دی مالتی پلکسر Demulti plexer :



شبیه سازی Demax

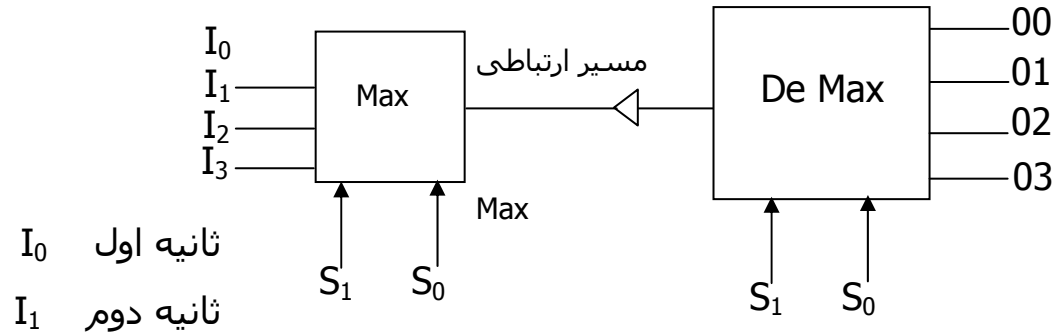
A	B	C	D_0	D_1	D_2	D_3
۰	۰	۰	۰	۰	۰	۰
۰	۰	۱	۰	۰	۰	۰
۰	۱	۰	۰	۰	۰	۰
۰	۱	۱	۰	۰	۰	۰
۱	۰	۰	۱	۰	۰	۰
۱	۰	۱	۰	۱	۰	۰
۱	۱	۰	۰	۰	۱	۰
۱	۱	۱	۰	۰	۰	۱

لذا مدار Demax همان دیکودر با یک ورودی Enable است.

کاربرد :

در ارتباط مخابراتی مانند تلفنهای منازل یک منطقه :

به این روش ارسال Time Domain Multiplexing (TDM) گویند .



I_0 ثانیه اول

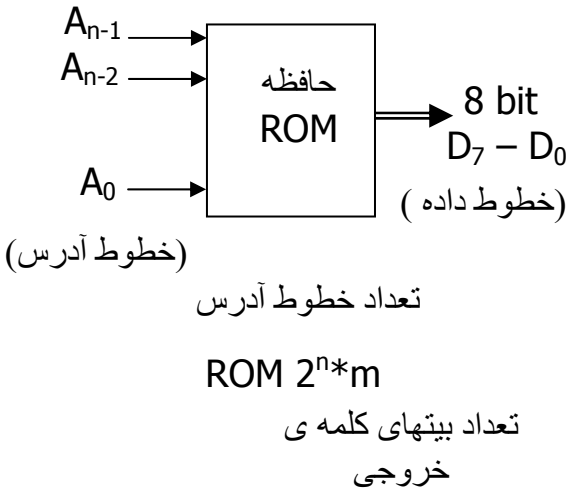
I_1 ثانیه دوم

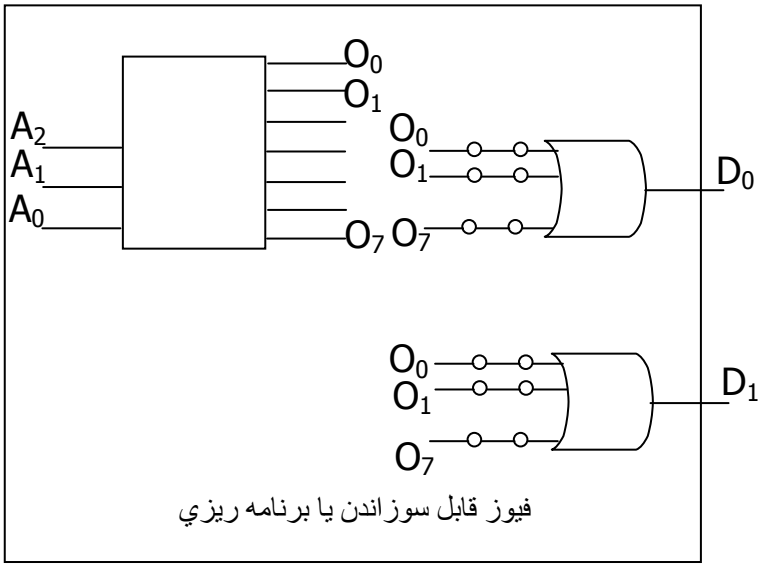
I_2 ثانیه سوم

I_3 ثانیه چهارم

۱۲- مدارهای ترکیبی – قطعات قابل برنامه ریزی (ROM)،(PLA) ،(CAL)،(PLD) و ... :

ROM یا Read Only Memory در واقع مدارهای ترکیبی هستند .





ROM 8 * 4

ROM : 8 خانه حافظه – هر خانه 4 بيتي يا 4 bit word

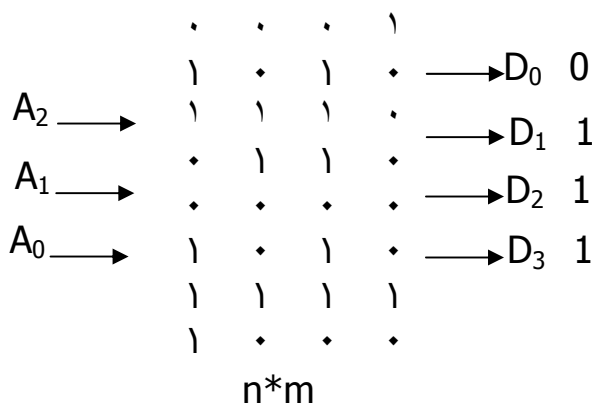
نكته : فيوز هاي 4,3,0 در D_3 سوزانده مي شوند .

نكته : ما در ROM تمام فينترمها را داريم و با or هاي قابل برنامه ريزي و با توجه به اطلاعات مشخص مي شود براي هر خروجي كدام فيوز ها سوزانده شود .

كاربرد ROM :

۱- پياده سازی توابع

۲- يک المان حافظه



استفاده از Rom به عنوان پیاده سازی توابع :

مداري تركيبي داريم كه n ورودي و m خروجي دارد و بدين منظور مي توان از 2^n*k Rom كه $k \geq m$ استفاده نمود.

نكته :

چون فیوزهای سوخته شده دیگر قابل باز یافت نیستند لذا Rom ها فقط یکبار قابل خواندن مي باشد . براي رفع اين مشكل EPROM طراحي شده است . كه با اشعه ي ماوراء بنفش پاك مي شود و با مدارهاي الكتريكي برنامه ريزي مي شوند .

ولي EPROM ها مدت زيادي را براي طراحي پاك شدن مصرف مي كنند (حدود 15-16 دقيقه) لذا EEPROM را طراحي كردند كه بوسيله ي مدارهاي الكتريكي پاك مي شوند كه بسيار كم زمان مي برند (ميلي ثانيه).

مدارهاي ترتيبی :

مدارهای منطقی ترتیبی :

خروجی علاوه براینکه به ورودیهای مدار بستگی دارد به خروجیهای قبلی مدار (و در نتجه به ورودیهای قبلی) بستگی دارد . پس می توان گفت مدار ترتیبی حافظه دار است .

انواع مدارهای ترتیبی :

۱- سنکرون یا همزمان : همزمان با سیگنالی بنام کلاک تغییرات در خروجی مدار و در اثر ورودی صورت می گیرد .

۲- آسنکرون یا غیر همزمان : تغییرات خروجی بدون سیگنال با تغییر ورودی امکان پذیر است .

موضوعات :

۱- آشنایی با انواع فلیپ فلاپها

۲- آشنایی با برخی مدارهای ترتیبی

۳- طراحی مدارهای منطقی ترتیبی

فليپ فلاپها (Flip-Flop یا FF) :

مدار منطقي است كه مي تواند يك بيت اطلاعات را براي ما نگه دارد .

انواع FF :

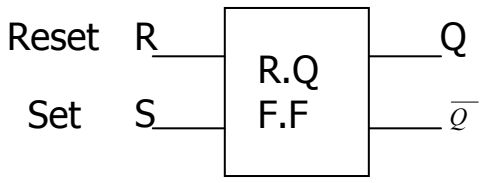
۱- Rs-FF

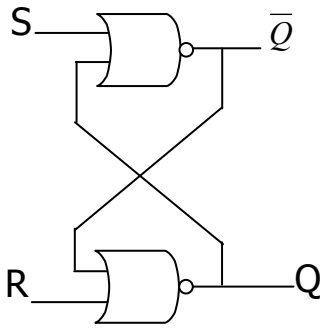
۲- D-FF

۳- jk-FF

T-FF -۴

Rs-FF :



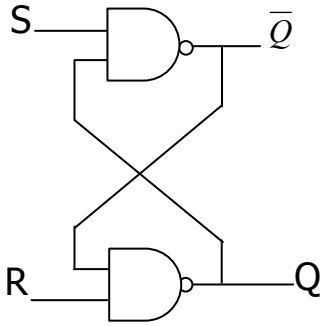


R	S	Q	$\overline{Q}$
۱	۰	۰	۱
۰	۰	۰	۱
۰	۱	۱	۰
۰	۰	۱	۰
۱	۱	۰	۰

یا Set – Reset FF

$\overline{Q}$ = NOT = Q

از خروجي Q يك Feed book يا باز خورد زديم به ورودي NOR



R	S	Q_{n-1}	$\overline{Q}_{n+1}$	وضعيت (عمل)
۱	۱	Q_n	$\overline{Q}_n$	وضعيت قبلي(Hold)
۰	۱	۰	۱	Reset
۱	۱	۰	۱	Hold
۱	۰	۱	۰	Set
۱	۱	۱	۰	Hold
۰	۰	۱	۱	غير مجاز

نكته : هرگاه ورودي را صفر كنيم تاثير منفي مي گذارد.

نكته : خروجي وضعيت قبلي به صورت يك ورودي به طور منفي منظور مي گردد.

جدول صحت (براي ساختار NOR) :

R	S	Q_n	Q_{n-1}		$\overline{Q}_{n+1}$
۰	۰	۰	۰		۱
۰	۰	۱	۱		۰
۰	۱	۰	۱		۰
۰	۱	۱	۱		۰
۱	۰	۰	۰		۱
۱	۰	۱	۰		۱
۱	۱	۰	*	Don't care	*
۱	۱	۱	*	Don't care	*

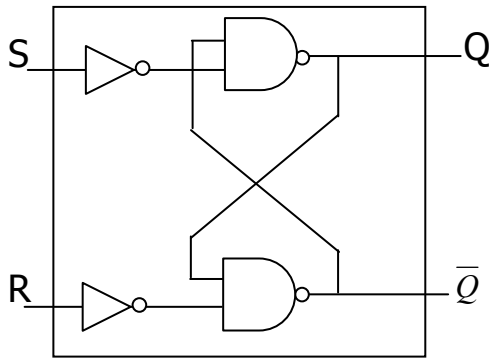
RQn / S	Q_{n-1}			
۰	۰	۱	۰	۰
۱	۱	۱	*	*

RQn / S	$\overline{Q}_{n-1}$			
۰	۱	۰	۱	۱
۱	۰	۰	*	*

$$Q_{n+1} = S + R'Q_n$$
$$Rs = 0$$
$$\overline{Q}_{n+1} = R + SQ_n$$

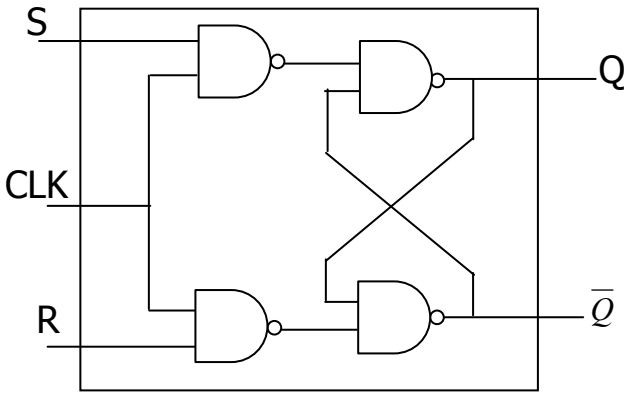
S,R با هم ترکیب نمي شود .

High Active : يك موثر است .



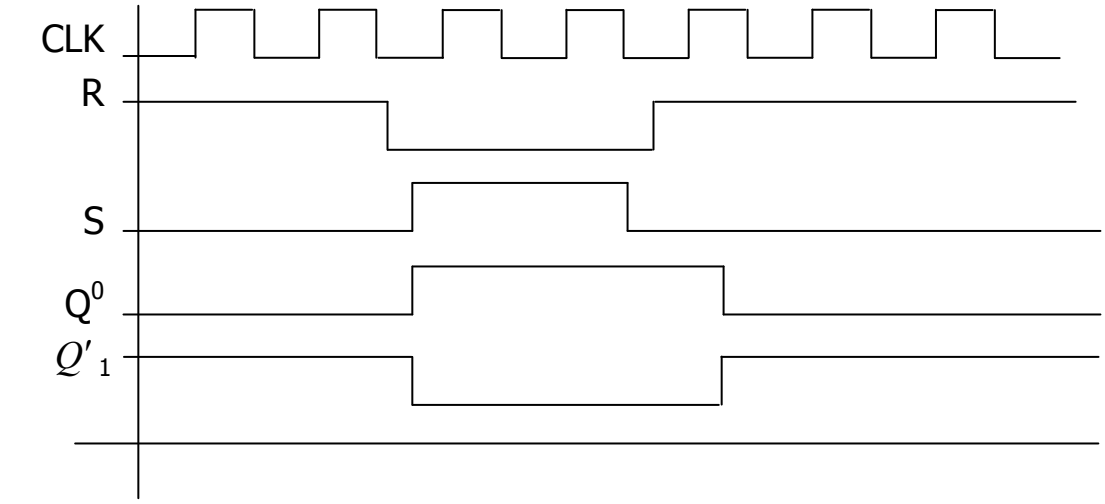
S	R	Q	Q'
۰	۰	Hold	
۱	۰	۱	۰
۰	۱	۰	۱
۱	۱	غیر مجاز	

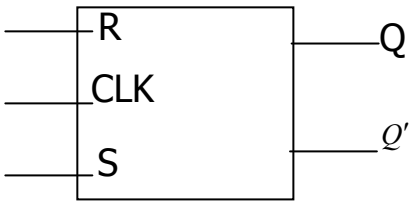
فیلیپ فلاپ با کلاك :



وضعیت	Q'_{n+1}	Q_{n-1}	S	R	CLK
Hold	Q'_n	Q_n	*	*	۰
Set	۰	۱	۱	۰	۱
Reset	۱	۰	۰	۱	۱
Hold	Q'_n	Q_n	۱	۱	۱
غیرمجاز	؟	۱	۱	۱	۱

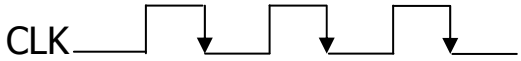
نکته : زماني خروجي تغيير مي كند كه Clock بيايد در غير اينصورت Hold يعني بايد Clock يك شود تا بعد R و S را بررسي كنيم در غير اينصورت R و S هر چه باشند Hold داريم .



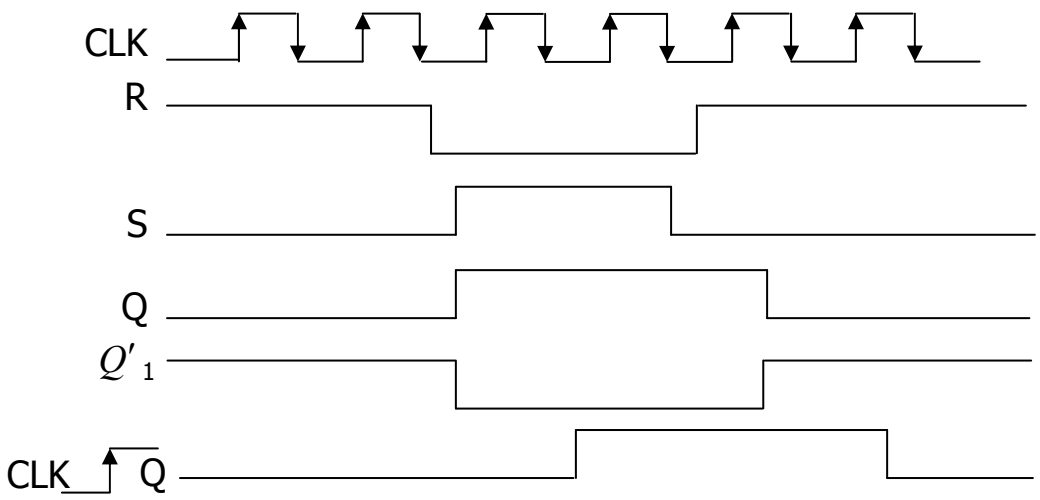


ورودي كلاك حساس به سطح يا FF يا حساس به سطح كلاك

F.F حساس به لبه كلاك : مدار داخلي از كتاب مانول ، زماني خروجي تغيير مي كند كه به لبه كلاك بيايد .



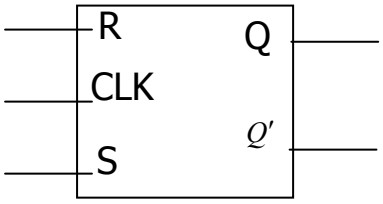
دو حالت مي تواند پيش آيد : حساس به لبه مثبت كلاك (بالا رونده ، يا حساس به لبه منفي كلاك (پايين رونده)



نكته : فقط در لحظه لبه ورودي هاي R,S را بررسي مي كند و در غير از آن هيچ كار نمي كند (Hold) حتي براي پايين رونده نيز هيچ كاري نمي كند .

نكته : F.F حساس به لبه در مدارهاي ترتيبي سنكرون استفاده مي شود و F.F هاي حساس به سطح در مدارهاي ترتيبي آسنكرون استفاده مي شود .

نمايش سمبليك :



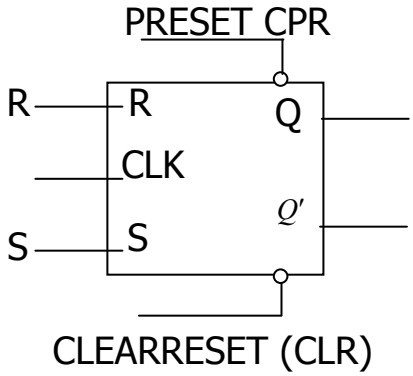
ترسیم جدول :

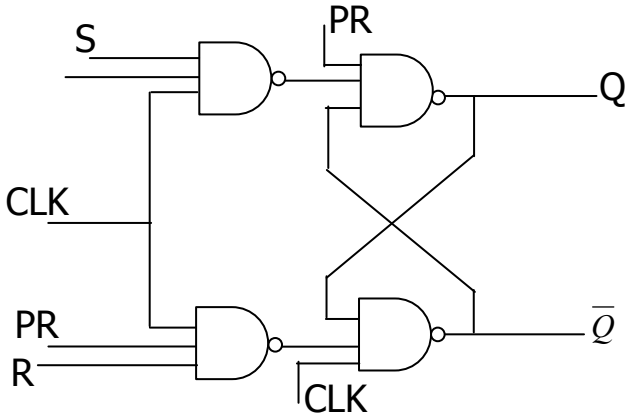
CLK	R	S	Q_{n-1}	Q'_{n+1}
۰	*	*	Q_n	Q'_n
۱	۰	۱	۱	۰
۱	۱	۰	۰	۱
۱	۱	۰	۰	۱
۱	۱	۱	۱	۱

نکته : کلاك يك سيگنال پريوديك مي باشد.

نکته : خروجي به خروجي هاي قبلي و به ورودي ها و هم به سيگنال زمانبندي بستگي دارد .

توسعه فيليپ فلاپ با كلاك :

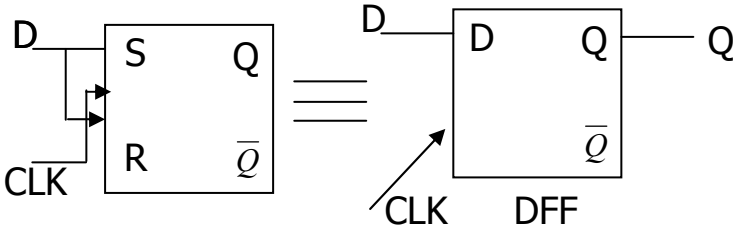




CLK	PR	CLK	R	S	Q_{n-1}	Q'_{n+1}
*	۰	۱	*	*	۱	۰
*	۱	۰	*	*	۰	۱
*	۰	۰	*	*	۱	۱
۰	۱	۱	*	*	Q_n	Q'_n
⎍	۱	۱	۰	۰	Q_n	Q'_n
⎍	۱	۱	۰	۱		
⎍	۱	۱	۱	۰		
⎍	۱	۱	۱	۱		

نکته : ورودی آسنکرون PR (CLR) بدون در نظر کلاک (زمانبندی) یا به صورت غیر همزمان روی خروجی تاثیر می گذارد .

D-FF : کاربرد در طراحی مدارهای ترتیبی و خصوصاً در رجیسترها :

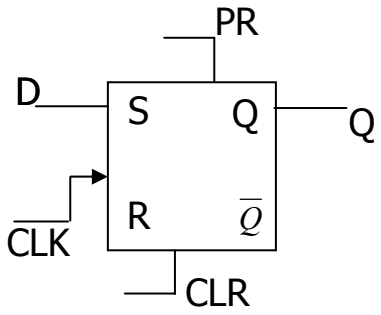


CLK	D	Q_{n-1}	وضعیت
L	*	Q_n	خط وضعیت
⎍	0	۰	Reset
⎍	1	۱	Set
⎍	D	Q_n	Q_{n+1}=D

مزایا :

۱- برای Set و Reset و ... از یک ورودی استفاده می کنیم .

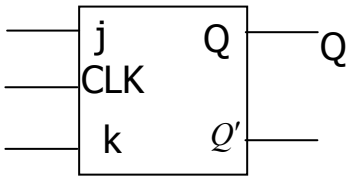
۲- حالت غیر مجاز را از بین برده است .

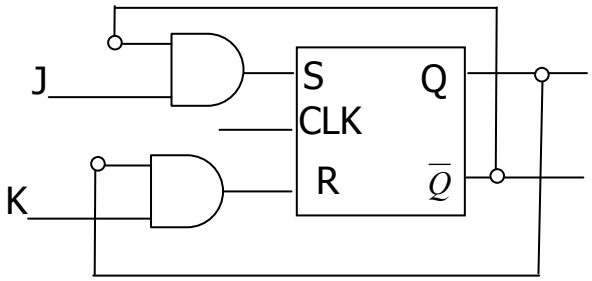


PR	CLK	CLD	D	
۰	۱	*	*	۱
۱	۰	*	*	۰
۰	۰	*	*	* غیر مجاز
۱	۱	⎍	D	D
۱	۱	۱	*	Q_n

Jk-ff : کاربرد در طراحي مدارهاي ترتيبي و مخصوصاً در شمارنده ها .

PR	CLK	CLD	Q_{n+1}	Q'_{n-1}	وضعیت
L	*	*	Q_n	Q'_n	Hold
⎍	1	0	1	0	Set
⎍	0	1	0	1	Reset
⎍	0	0	Q_n	Q'_n	Hold
⎍	1	1	Q'_n	Q_n	Toggle (NOT)

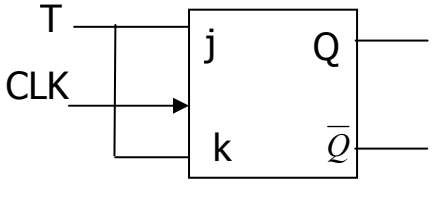


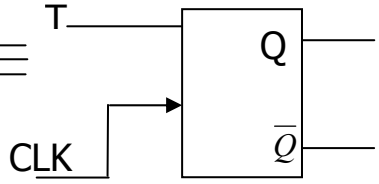


J \ KQn	۰۰	۰۱	۱۱	۱۰
۰	۰	۱	۰	۰
۱	۱	۱	۰	۱

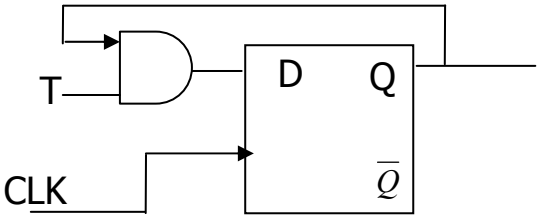
$$Q_{n+1} = Q_n k' + JQ'_n$$
$$Q'_{n+1} = J'Q'_n + kQ_n$$

T-FF : JK-FF اي است كه j و K آن به هم به هم متصل است .كاربرد و شمارنده ها .





CLK	T	Q_{n+1}	وضعيت
۰	x	Q_n	Hold
⎍	۰	Q_n	Hold
⎍	۱	Q'_n	toggle



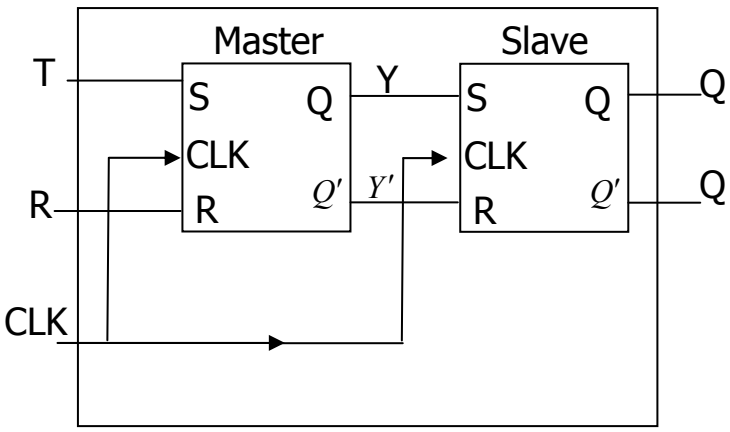
$$Q_{n+1} = T'Q_n + TQ'_n = T \oplus Q_n$$

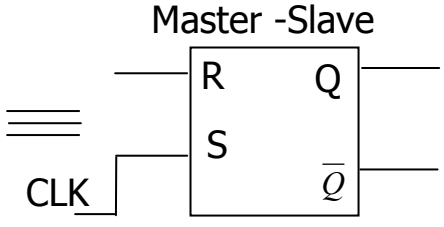
نكته : در سطح مادامي كه CLK وجود دارد و k و j هـر دو يـك باشـند و JKFF حـساس به سطح باشد خروجي نوسان مي كند براي رفع اين مشكل :

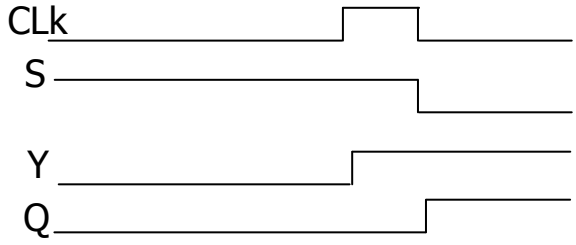
۱- استفاده از JK-FF حساس به لبه

۲- با استفاده از Master-Slave . F.F آنرا JK تبديل به حساس به لبه مي كند .

: Master-Slave . F.F

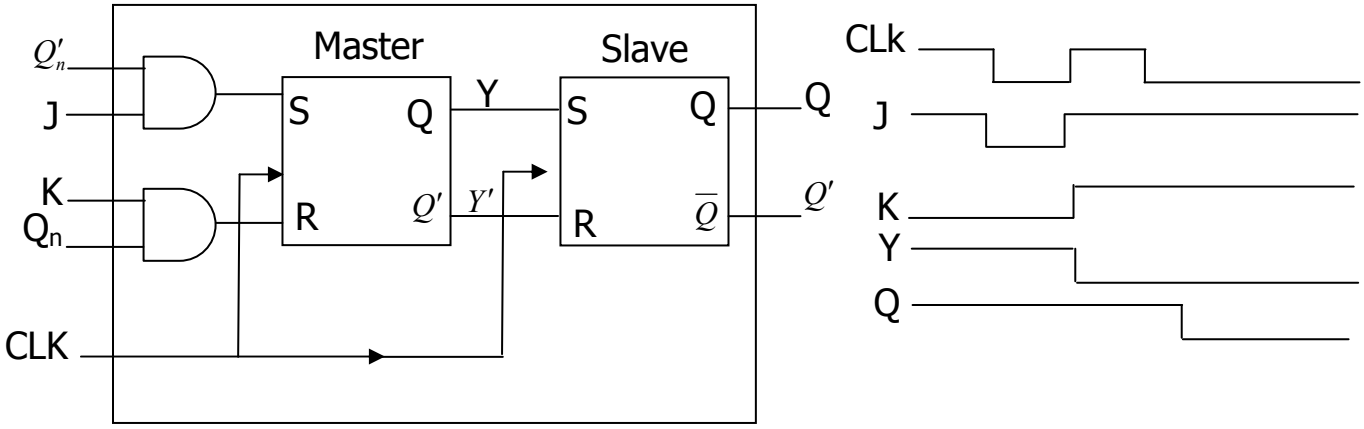






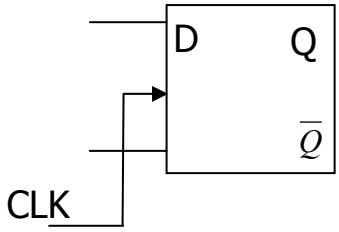
CLK	Master	Slave
Low	disActive	Active
High	Active	disActive

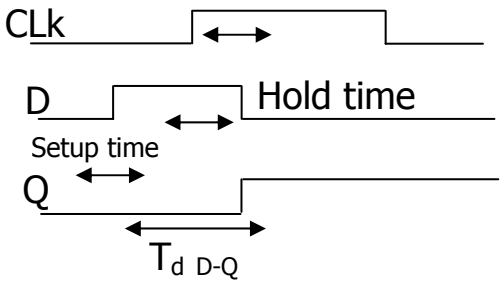
R = "C"



و بدین ترتیب مشکل حل شد .

تاخیر



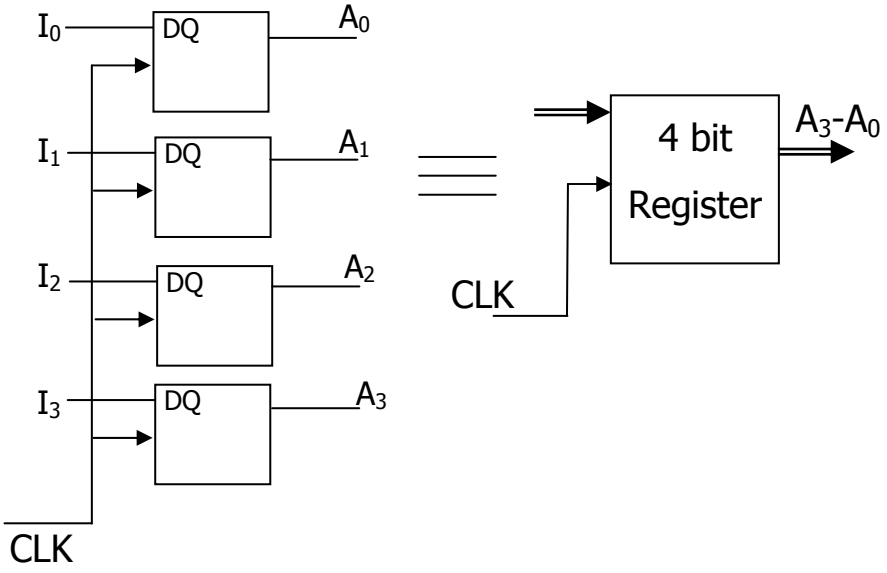


نکته : باید کمی قبل از آمدن CLK «1» شـود تـا بتواننـد تغییـرات لازم را در مـدار بدهـد (آماده باشـد) که به این زمان Setup time گویند (حداقل زمان کمی می بایسـت قبـل از آمدن به کلاک ورودی (Stable بماند) همچنین بـرای انجـام تغییـرات لازم در مـدارات داخلی D نباید همزمان با آمدن CLK صفر شود بلکه باید کمی بگذرد که بـه ایـن مـدت

Hold time گویند (= مدت زمانی که لازم است تا ورودی D پس از آمدن به کلاک ثابت بماند تا تغییر مناسب در خروجی صورت گیرد .

رجیستر ها یا ثباتها :

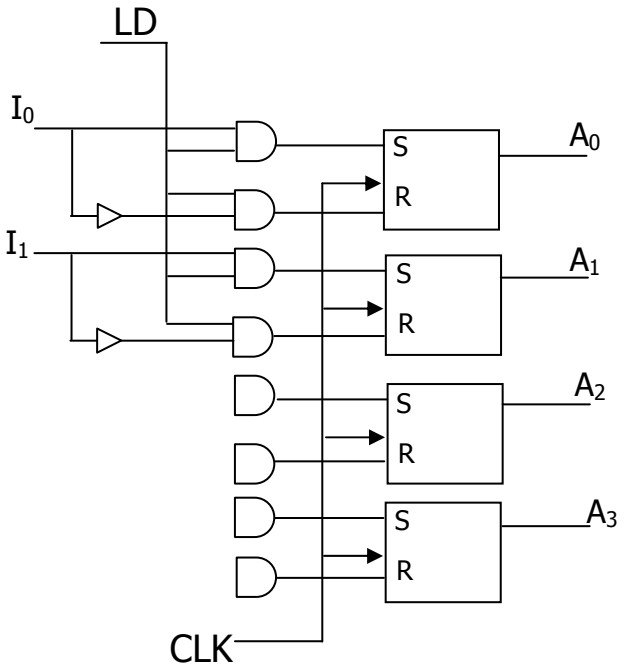
به مجموعه ای از FF ها که با کلاک مشترک کار می کنند و برای ثبت اطلاعات دودویی بکار می روند ثبات یا ریجستر گویند .

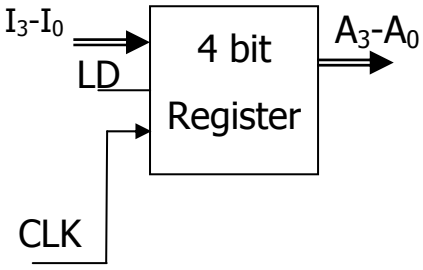


CLK	in puts	Out puts
L	x	خروجيهاي قبلي
↓	I3-I0	I3-I0

نکته : اگر D-FF حساس به لبه باشد کلاک ورودی ، CLK نام دارد و اگر حساس به سطح باشد ورودی کلاک G (گیت) نام دارد .

نکته : چون بایستی CLK به کلاک سیستم ، هیچ کنترلی روی ثبات ندایم (زیرا کلاک سیستم تواماً می آید) لذا از یک ورودی دیگر به نام LD (Laod) استفاده می کنیم که به چند تو مایه های Enable است .

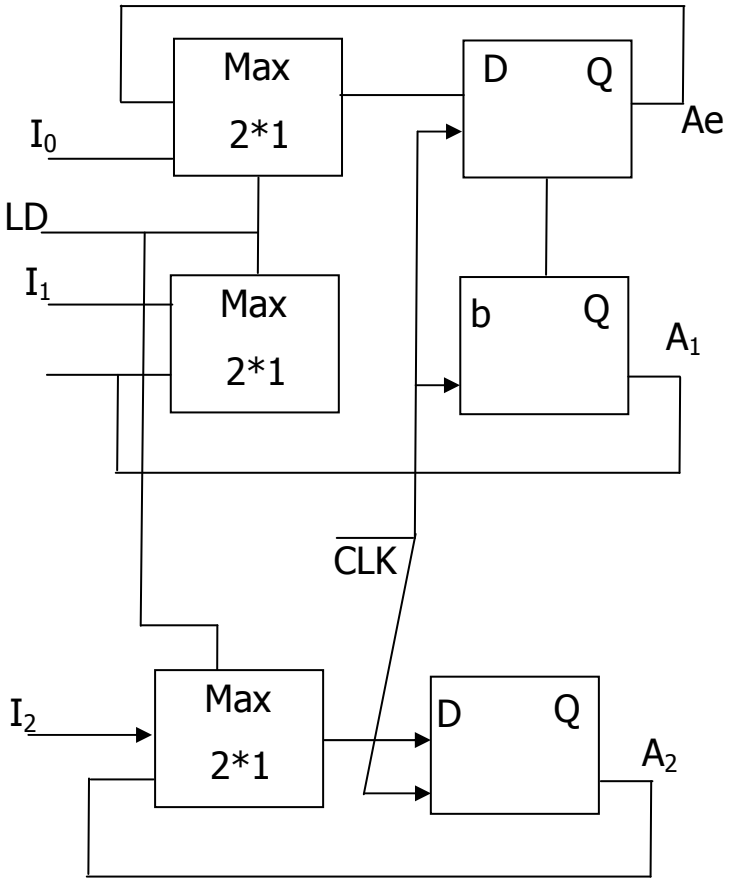


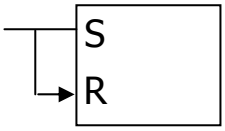


LD	CLK	in puts	Out puts
o	x	x	Rfgd
۱	↓	I_3-I_0	I_3-I_0

LD	I_0	CLK	A_0
۰	x	↓	قبلي
۱	۰	↓	۰
۱	۱	↓	۱

اگر ثبات ۴ بیتی با کمک D-FF و با ورودی LD :





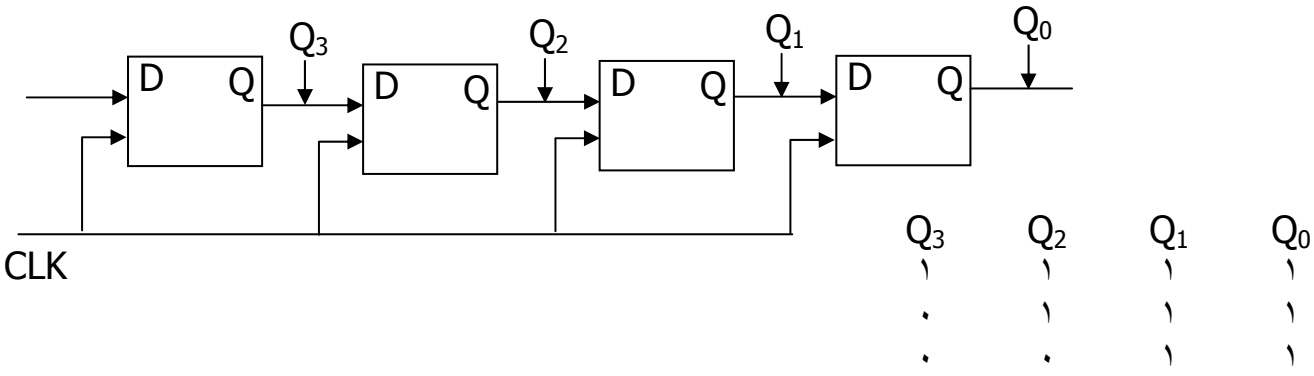
شیفت رجیسترها :

برای جابه جایی اطلاعات باینتری (شیفت) به سمت چپ یا راست.

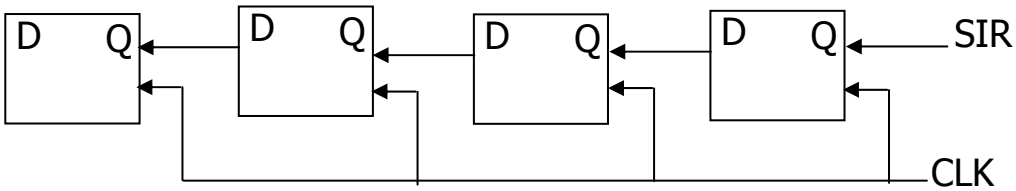
کاربرد : ضرب ، تقسیم ، جابه جائی اطلاعات .

۱۰	1	0	1	0
۵	0	1	0	1
۱۰	1	0	1	0

مثال :



مدار شیفت رجیستر شیفت به چپ :



تقسیم بندی با توجه به ورودی و خروجی :

۱- ورودی سری – خروجی سری SI/SO

SI → | 1 | 0 | 1 | 0 |

۲- ورودی سری – خروجی موازی SI/PO

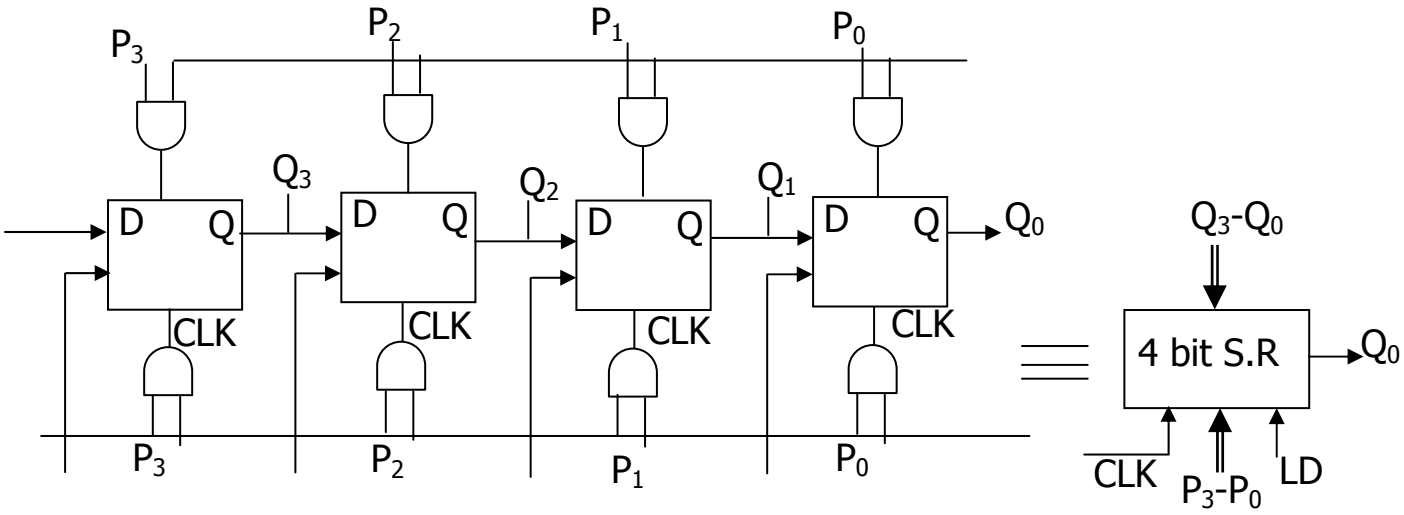
SI → | 0 | 1 | 0 | 1 |

۳- وودی موازی – خروجی سری PI/SO

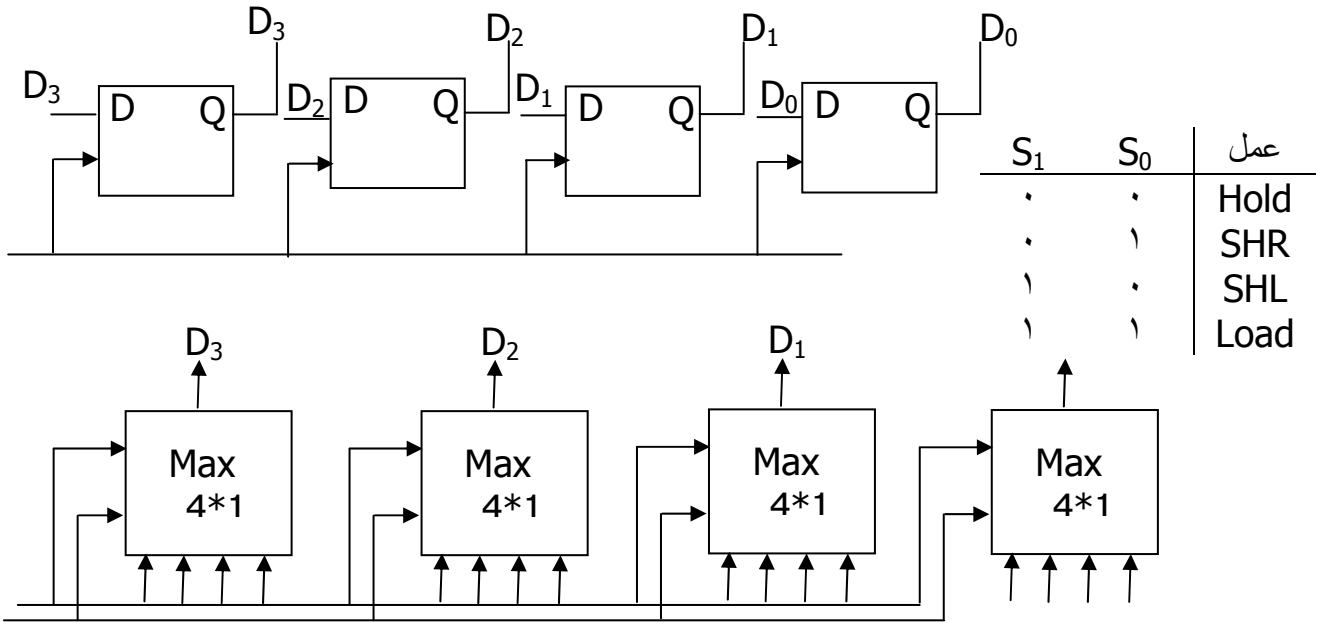
→ | 1 | 0 | 1 | 0 | →

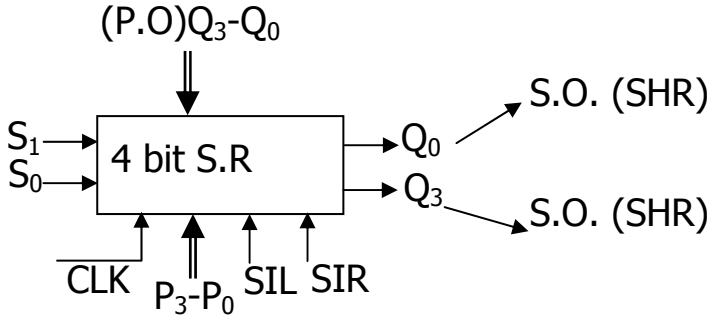
۴- ورودی موازی – خروجی موازی PI/PO

| 0 | 1 | 0 | 1 |



هدف : طراحی شیفت رجیستری که با دو خط انتخاب S_1 , S_A اعمال باردهی ، جابه جایی به چپ یا راست به همراه نگهداری اطلاعات را انجام می دهد . که به آن شیفت رجیستری عمومی یا همه منظوره گفته می شود .



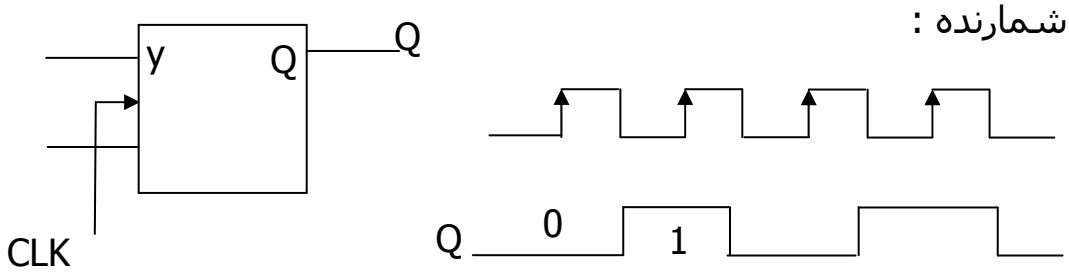


شمارنده ها :

درحالت کلی : ۱- سنکرون یا همزمان : تغییر خروجی تمام F.F ها با تاخیر ثابتی نسبت به CLK صورت می گیرد .

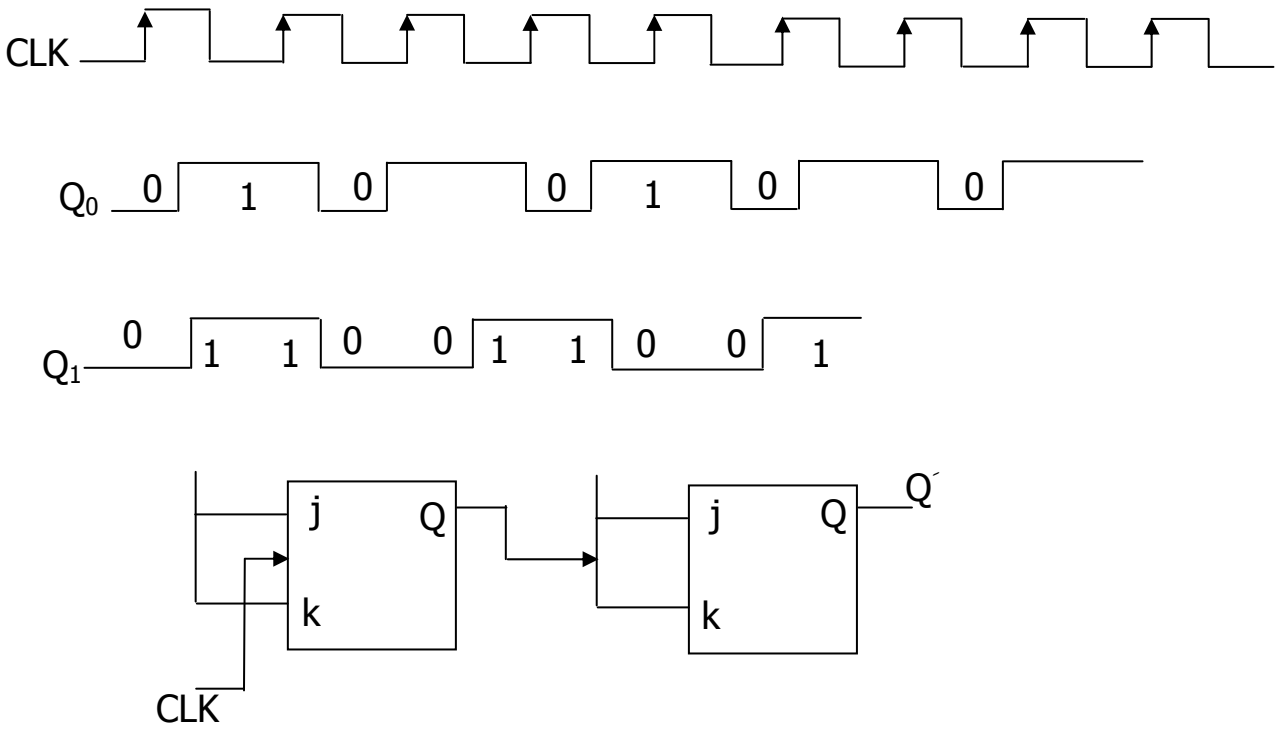
۲- آنسکرون یا غیر همزمان یا ضربان : تغییر خروجی تمام F.F ها با تاخیر متغیر (ثبت به CLK) صورت می گیرد .

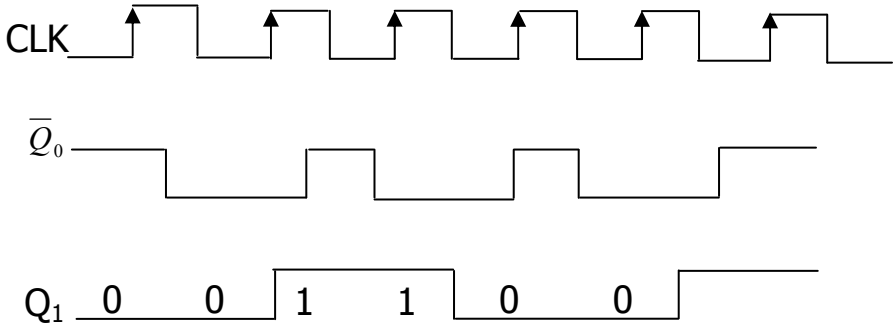
ساده ترین شمارنده :

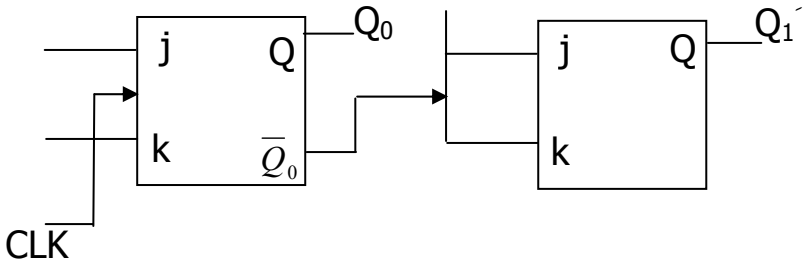


خروجی تقسیم بر 2 $F_Q = \frac{f_{CLK}}{2}$

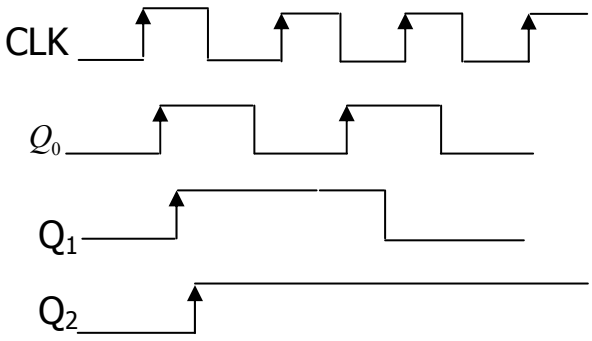
توسعه شمارنده :



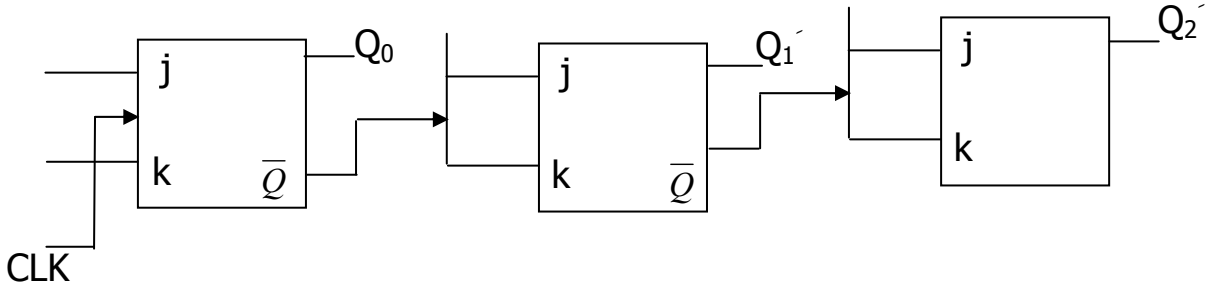


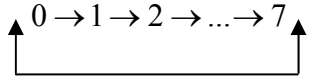


شمارنده آسنکرون

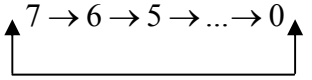


$Q_1 - Q_0$	$0 \to 1$	$2 \to 3$
0	0	
0	1	
1	0	
1	1	



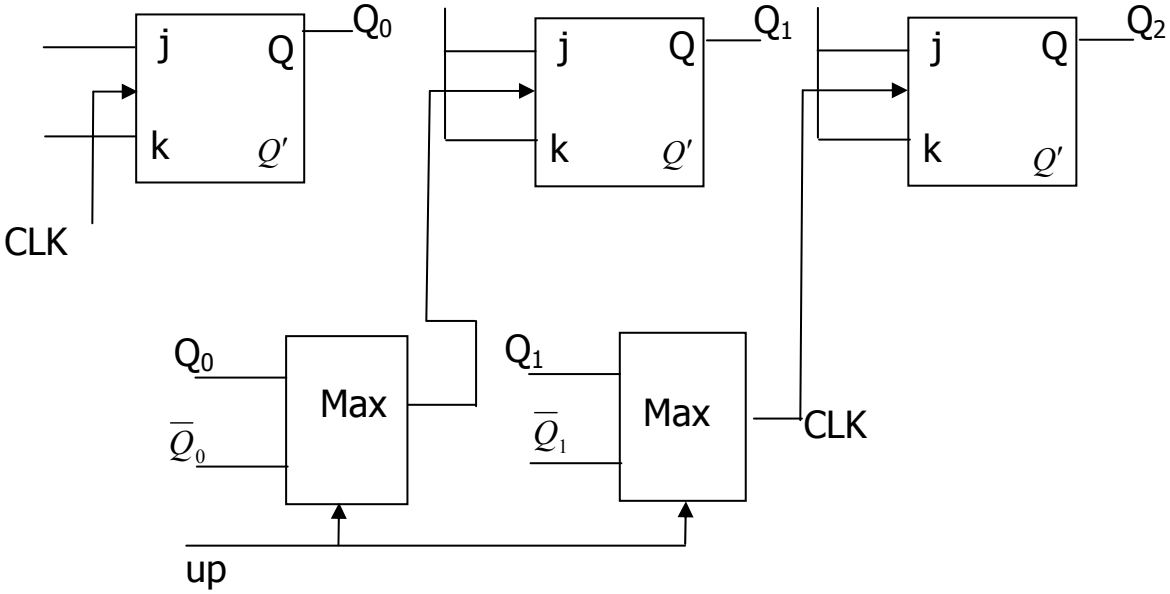


بالا رو :



پایین رو :

مثال : شمارنده آسنکرونی طراحی کنید که با یک خط کنترل به سمت بالا یا پایین شمارش کند .



نکته :

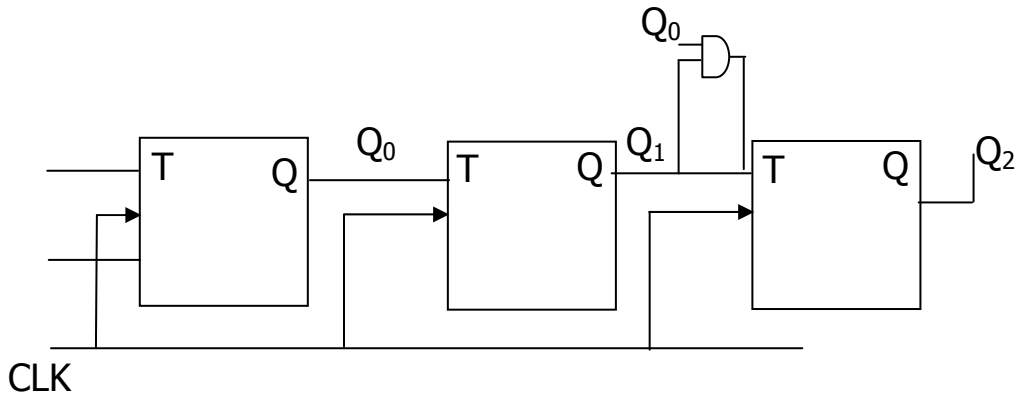
- در 2^n لزومی ندارد که ورودیهای k,j حتماً به ۱ وصل می شود .

- برای شمارنده های آنسکرون غیر 2^n روش خاصی وجود ندارد .

نحوه شمارش	خروجیها	ورودیهای
پایین رو	Q ها	Q ها
بالا رو	$\overline{Q}$ ها	Q ها
بالا رو	Q ها	$\overline{Q}$ ها
پایین رو	$\overline{Q}$ ها	$\overline{Q}$ ها

شمارنده سنکرون :

Q_2	Q_1	Q_0	
۰	۰	۰	۰
۰	۰	۱	۱
۰	۱	۰	۲
۰	۱	۱	۳
۰	۰	۰	۴
۱	۰	۱	۵
۱	۱	۰	۶
۱	۱	۱	۷
۰	۰	۰	۸



نکته :

فقط تاخیر کلاک تا خروجی را داریم . چون این AND قبل از آمدن کلاک خروجی اش را آماده می کند .

$$\text{Max } F_{CLK} = \frac{۱}{\text{تاخیر AND} + \text{تاخیر FF}}$$

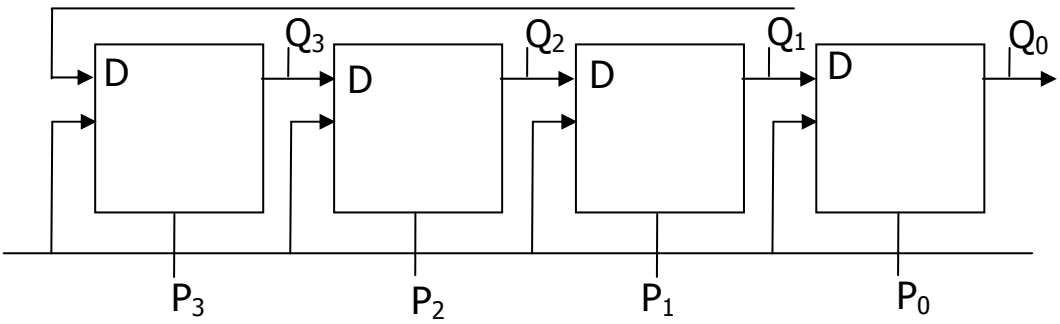
ولی برای آنسکرون : $Max\,F_{CLK} = \frac{1}{ntd}$

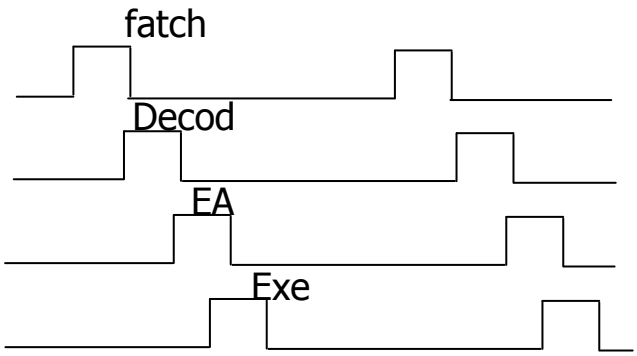
مثال : اگر $FF_{td} = 20_{ns}$ باشد آنگاه بیشینه ی F_{CLK} چقدر است ؟ (برای شمارنده ی آنسکرون ۴ بیتی)

$$td_{tot} = 4 * td = 80_{ns} \qquad \rightarrow F_{\max} = 12.5\,MH$$

شمارنده های خاص :

۱- شمارنده حلقوی

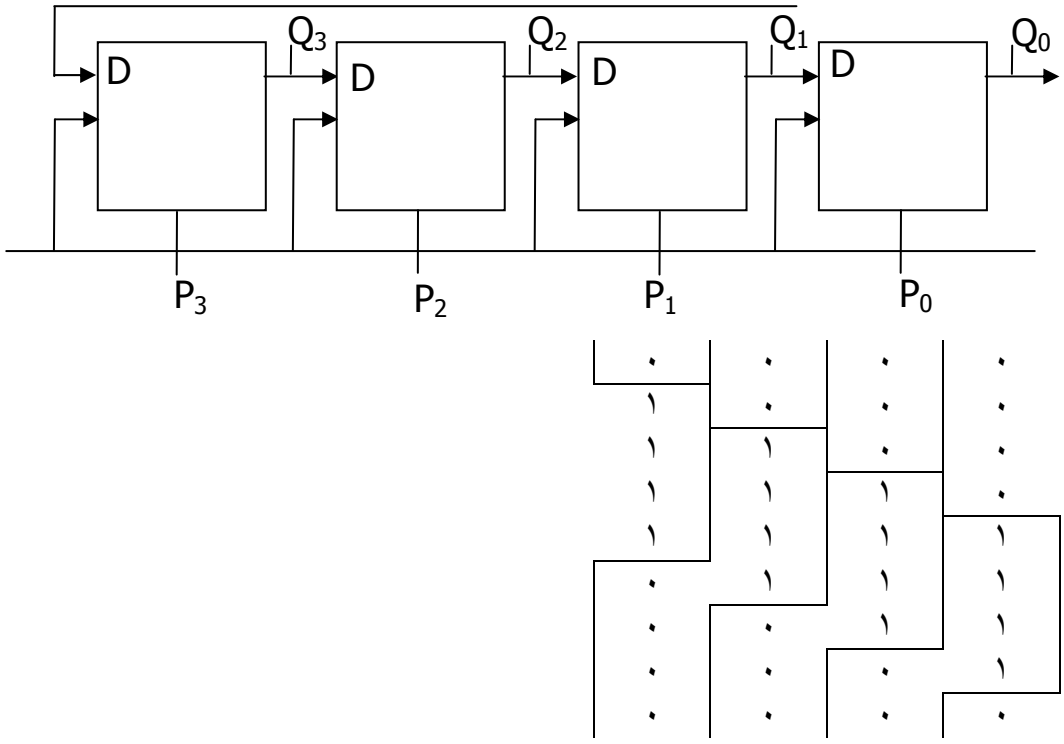




	P_3	P_2	P_1	P_0
عمل Load	۱	۰	۰	۰

	Q_3	Q_2	Q_1	Q_0
ترتیب شمارش :	۱	۰	۰	۰
	۰	۱	۰	۰
	۰	۰	۱	۰
	۰	۰	۰	۱
	۱	۰	۰	۰

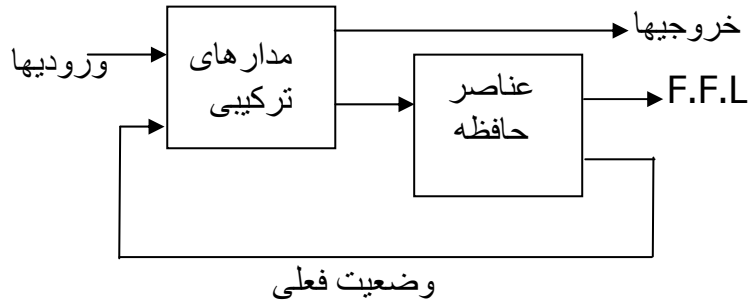
۲- شمارنده ی جانسون :



P_3	P_2	P_1	P_0
۰	۰	۰	۰
۱	۰	۰	۰
۱	۱	۰	۰
۱	۱	۱	۰
۱	۱	۱	۱
۰	۱	۱	۱
۰	۰	۱	۱
۰	۰	۰	۱
۰	۰	۰	۰

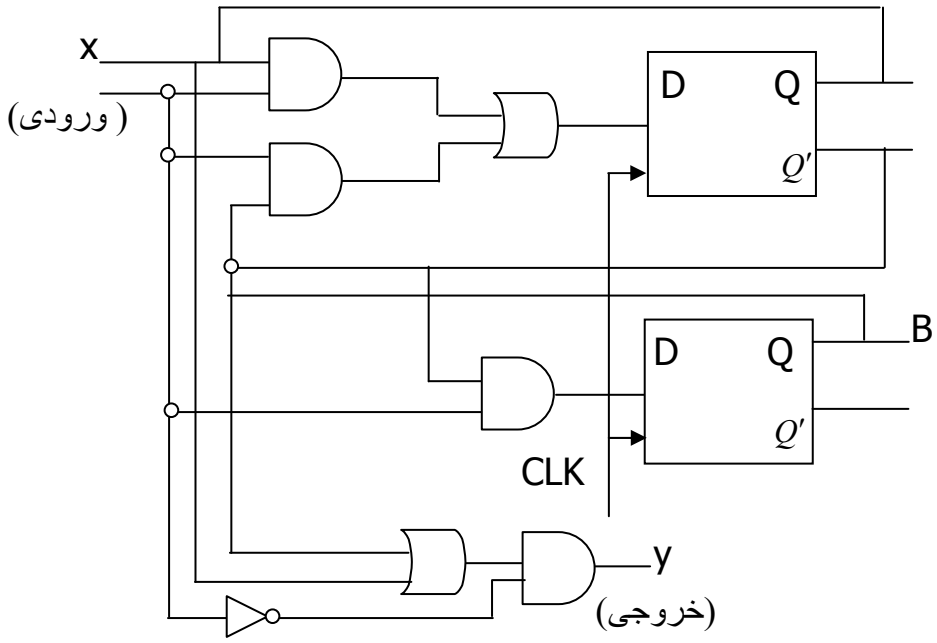
طراحی مدارهای منطقی ترتیبی :

ساختار کلی مدارهای ترتیبی :



مثال از مدار منطقی ترتیبی :

$$A(t+1) = A(t)x(t) + B(t)x(t)$$
$$B(t+1) = A'(x)x(t)$$
$$y = (A+B)x'$$
$$\downarrow \quad x'(t)$$
$$A(t)$$



جدول حالت :

ترتیب زمانی ورودیها ، خروجیها و وضعیت FF ها را در جدولی بنام جدول حالات می توان بیان نمود . این جدول شامل قسمتهای حالت فعلی ، ورودی ، حالت بعدی و خروجی است .

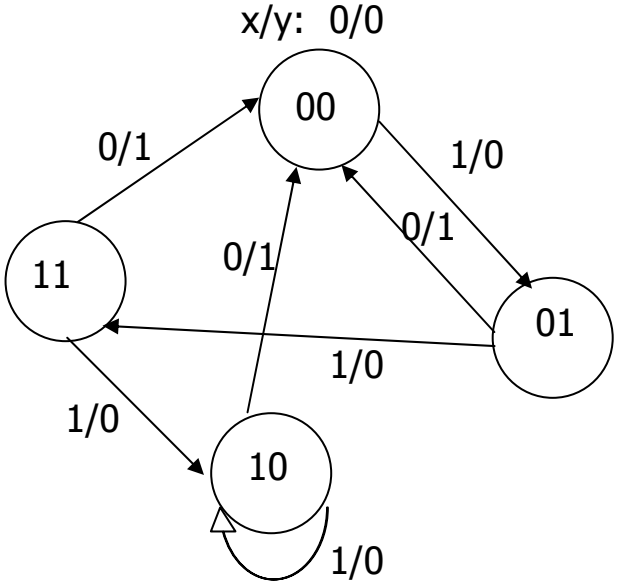
جدول حالت مثال بالا :

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B	x	A	B	y
۰	۰	۰	۰	۰	۰
۰	۰	۱	۰	۱	۰
۰	۱	۰	۰	۰	۱
۰	۱	۱	۱	۱	۰
۱	۰	۰	۰	۰	۱
۱	۰	۱	۱	۰	۰
۱	۱	۰	۰	۰	۱
۱	۱	۱	۱	۰	۰

طراحی جدول به روش دیگر :

A	B	x		A		B		y	
		x=0		x=1		x=0		x=1	
۰	۰	۰	۰	۰	۱	۰		۰	
۰	۱	۰	۰	۱	۱	۱		۰	
۱	۰	۰	۰	۱	۰	۱		۰	
۱	۱	۰	۰	۱	۰	۱		۰	

دیاگرام حالت (State Diagram) :

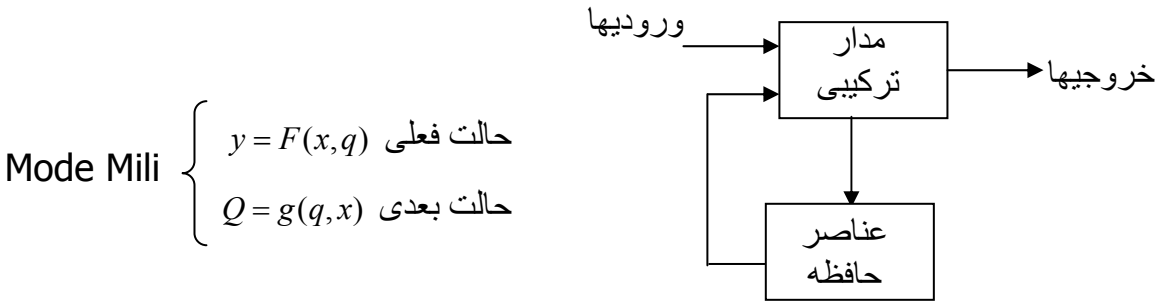


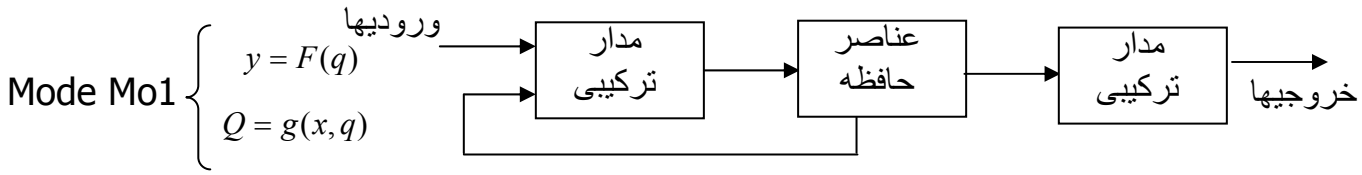
انواع مدارهای ترتیبی :

مدهای میلی و مور :

۱- مد میلی : خروجی بر اساس حالت فعلی ورودیها مشخص می شود .

۲- مد مور : خروجی فقط از روی حالات فعلی مشخص می شود ، به صورت مستقیم به ورودی ربط ندارد یا در رابطه ی خروجی ورودیها دیده نمی شود .

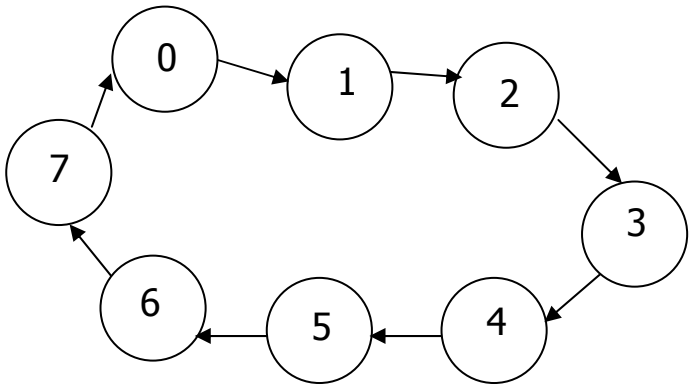




نکته : در مد میلی اگر ورودی تغییر کند حتی بدون آمدن CLK ممکن است که خروجی تغییر کند ولی در مد مور چنین نیست لذا برای رفع این مشکل ورودیهای مد میلی را با CLK همزمان می کنند .

نکته : مدار صفحه ی قبلی مثالی از یک مد و شمارنده های مثالی از مد مور هستند .

نکته : در مد مور نمایش State Diagram تنها برحسب ورودی است یعنی :



جدول تحریک F.F ها :

S.R را چه در نظر بگیریم تا به حالت Q ها برسیم .

Q(t+1)	Q(t+1)	S	R	D	J	K	T
۱	۰	۰	x	۰	۰	x	۰
۰	۱	۱	۰	۱	۱	x	۱
۱	۰	۰	۱	۰	x	۱	۱
۱	۱	x	۰	۱	x	۰	۰

مراحل طراحی مدارهای ترتیبی :

۱- توصیف لفظی عملکرد مدارد : ۱- دیاگرام حالت ۲- دیاگرام زمانی

۲- با توجه به اطلاعات مفروض در مورد مدار جدول حالات تنظیم شود .

۳- تعداد حالات را در صورت امکان کاهش دهید .

۴- اگر درجدول حالات ، سمبل های حرفی وجود دارد ، آنها را با مقادیر دودویی جایگزین کنید (تشخیص حالت)

۵- تعداد F.F ها را مشخص کنید (با توجه به جدول حالات) و به هر کدام سمبل حرفی تخصیص دهید .

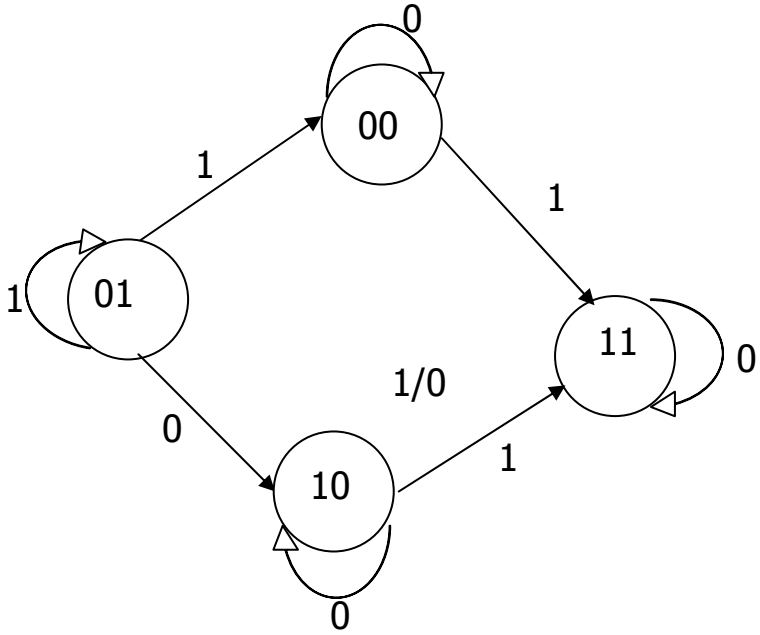
۶- نوع F.F ها را مشخص کنید (معمولاً در صورت مساله)

۷- باتوجه به جدول حالات ، جداول تحریک و خروجی را بدست آورید .

۸- ساده سازی روابط خروجی ، ورودیهای تحریک از روی جدول خروجی و تحریک با کمک روشهای متعارف مثل روش کارنو .

مثال : دیاگرام حالت :

دیاگرام حالت زیر مفروض است . مدار ترتیبی لازم را به کمک jkff رسم کنید .



	بعدی		ورودی	فعلی	
	B	A	x	B	A
3 امکان ندارد	۰	۰	۰	۰	۰
	۱	۰	۱	۰	۰
4 نداریم	۰	۱	۰	۱	۰
	۱	۰	۱	۱	۰
5 دوتا FF	۰	۱	۰	۰	۱
	۱	۰	۱	۰	۱
JKFF 6	۱	۱	۰	۱	۱
	۰	۰	۱	۱	۱

J_A	K_A	J_B	K_B
۰	x	۰	x
۰	x	۱	x
۱	x	x	۱
۰	x	x	۰
x	۰	۰	x
x	۰	1	x
x	۰	x	۰
x	۱	x	۱

A \ AB

۰	۰	۰	۱
x	x	x	x

$: j_A = Bx'$

A \ Bx

x	x	x	x
۰	۰	۱	۰

$: k_A = Bx$

A \ ox

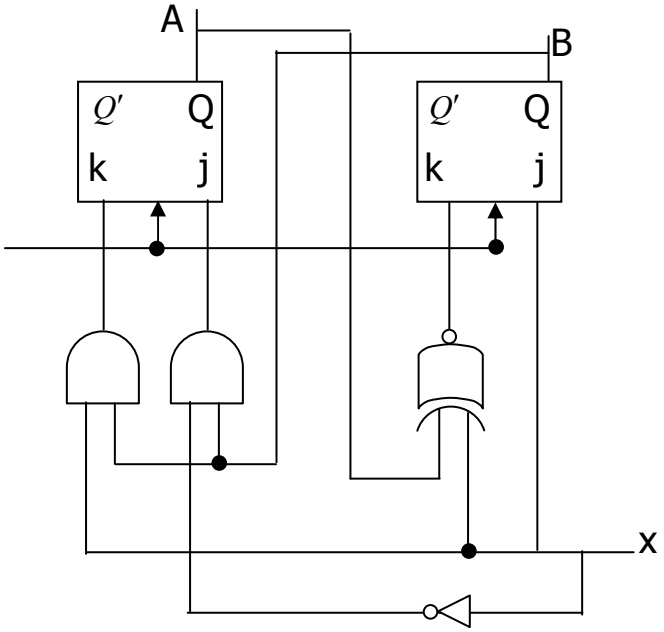
۰	۱	x	x
۰	۱	x	x

$: j_B = x$

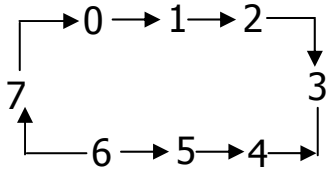
A \ Bx

x	x	۰	۱
x	x	۱	۰

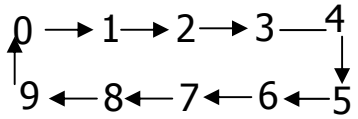
$: k_B = (A \oplus x)'$



تمرین : شما رنده اي طراحي كنيد كه ترتيب زير بشما رد با كمك T.F.F



مثال : شما رونده BCD با كمك T.FF :



(فعلي)				(بعدي)			
Q	Q	Q	Q	Q	Q	Q	Q
8	4	2	1	8	4	2	1
۰	۰	۰	۰	۰	۰	۰	۱
۰	۰	۰	۱	۰	۰	۱	۰
۰	۰	۱	۰	۰	۰	۱	۱
۰	۰	۱	۱	۰	۱	۰	۰
۰	۱	۰	۰	۰	۱	۰	۱
۰	۱	۰	۱	۰	۱	۱	۰
۰	۱	۱	۰	۰	۱	۱	۱
۰	۱	۱	۱	۱	۰	۰	۰
۱	۰	۰	۰	۱	۰	۰	۱
۱	۰	۰	۱	۰	۰	۰	۰

جدول تحريك :

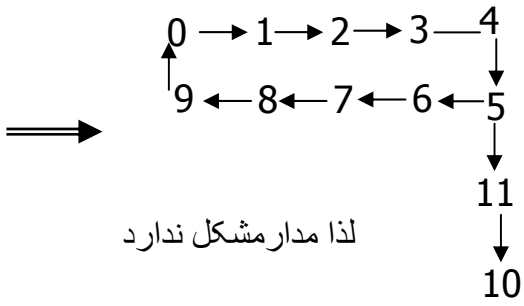
T_8	T_4	T_2	T_1
۰	۰	۰	۱
۰	۰	۱	۱
۰	۰	۰	۱
۰	۱	۱	۱
۰	۰	۰	۱
۰	۰	۱	۱
۰	۰	۰	۱
۱	۱	۱	۱
۰	۰	۰	۱
۱	۰	۰	۱
*	*	*	*

$$T_{Q_8} = Q_{D_8} Q_{A_1} + Q_{C_4} Q_{B_2} Q_{A_1}$$
$$T_{Q_4} = Q_2 Q_1$$
$$T_{Q_2} = Q_2 Q_1$$
$$T_{Q_1} = 1$$

نکته : اگر بعد از بستن مدار مقدار اولیه ۱۲ یا ۱۳ و ... بود چون پیش بینی نشده اند لذا مدار به درستی عمل نخواهد کرد لذا باید این حالات را بررسی کرد .

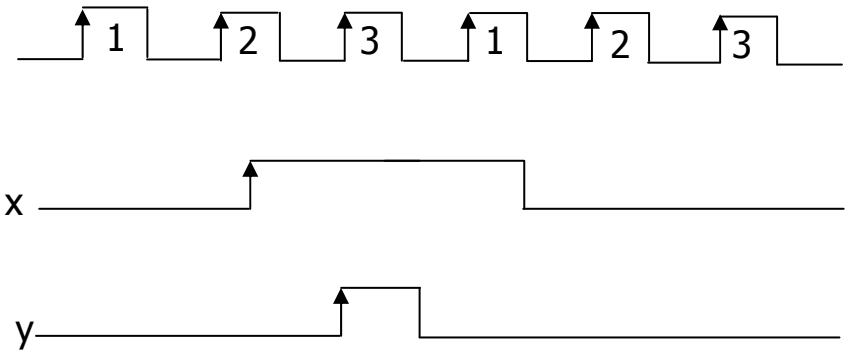
Q_8	Q_4	Q_2	Q_1
۱	۰	۱	۰
۱	۰	۱	۱
۱	۱	۰	۰
۱	۱	۰	۱
۱	۱	۱	۰
۱	۱	۱	۱

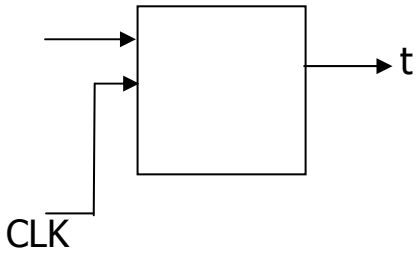
Q_8	Q_4	Q_2	Q_1
۱	۰	۱	۱
۰	۱	۱	۰
۱	۱	۰	۱
۰	۱	۰	۰

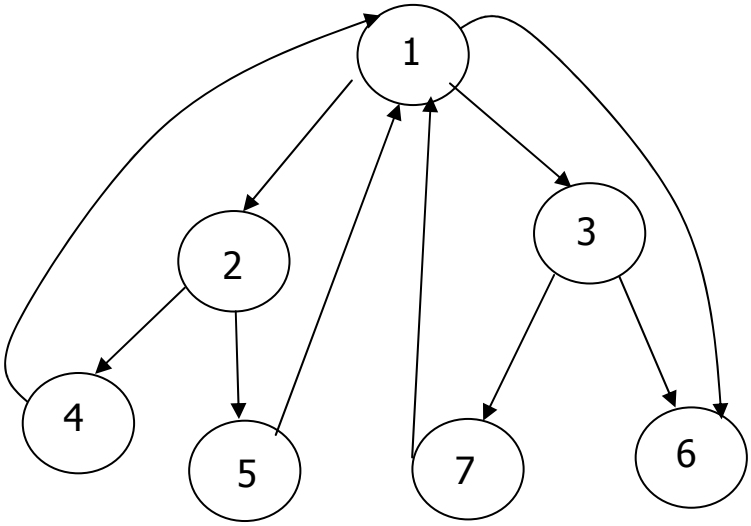


نکته : می توانستم از همان ابتدا مدار را بر این مبنا طراحی کنیم که اعداد Dont Care به صفر بروند .

مثالی از آخر : مداری طراحی کنید که به صورت پریودیک پالس ساعت ورودی x را بررسی و در صورت فرود بودن تعداد یکهای دریافتی ، خروجی یک شود .







جدول حالات :

حالتهاي مشابه :

به ازاي ورودیهاي یکسان

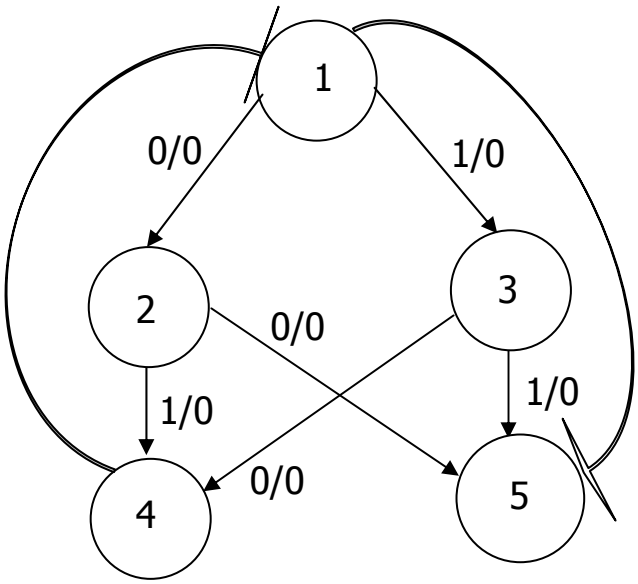
حالت بعدي و خروجي

حالت فعلي	حالت بعدي		خروجي	
	x=0	x=1	x=0	x=1
1	2	3	0	0
2	5	4	0	0
3	6	7	0	0
4	1	1	1	0
5	1	1	0	1
6	1	1	1	0
7	9	1	0	1

یکسان نتیجه شود.

لذا 4 با 6 و 5 با 7 مشابه هستند در نتیجه مي توان گفت 6 و 7 زيادي هستند .

: Up dateed State Diagram



تخصيص حالات : سعي مي شود از يك حالت به حالت ديگر مجاور منطقي در جدول كار نو باشند .

	حالت فعلي			بعدي		خروجي	
				x=0	x=1	x=0	x=1
1	۰	۰	۰	۰۰۱	۰۱۱	۰	۰
2	۰	۰	۱	۱۱۱	۱۰۱	۰	۰
3	۰	۱	۱	۱۰۱	۱۱۱	۰	۰
4	۱	۰	۱	۰۰۰	۰۰۰	۱	۰
5	۱	۱	۱	۰۰۰	۰۰۰	۰	۱
	A	B	C				

با كمك T.F.F

خلاصه ...

$$T_A = C$$
$$T_B = AB + \overline{C}x + \overline{A}C\overline{x}$$
$$T_C = A + \overline{C}$$
$$y = A\overline{B}\overline{x} + ABx$$