سخت افزار (۲)

رشته کامپیوتر- مهندسی نرم افزار

دانشگاه آزاد اسلامی

سازمان سما

آموزشکده فنی و حرفه‌ای سما اندیشه

تألیف : رامین رضوانی

فصل اوّل

سیستم‌های دودویی

سیستم اعداد و مبناهای عددی

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- آشنایی مختصر با سیستم‌های دیجیتال
- آشنایی با سیستم اعداد
- شناخت مبناهای عددی مختلف از جمله مبناهای ۲، ۸، ۱۰، ۱۶
- نحوه تبدیل مبناهای اعداد غیراعشاری به یکدیگر
- نحوه تبدیل مبناهای اعداد اعشاری به یکدیگر

سیستم‌های دیجیتال

مشخصه اصلی سیستم‌های دیجیتال، قدرت آن‌ها در کار با اعداد و ارقام، حروف الفبا و به طور کلی هر مجموعه متشکل از تعداد متناهی از عناصر گسسته اطلاعاتی می‌باشد؛ کامپیوترها نیز دسته خاصی از سیستم‌های دیجیتال هستند و چون کامپیوترهای اولیه برای محاسبات عددی به‌کار می‌رفتند، نام دیجیتال یا رقمی برای آن‌ها گذاشته شده است.

سیستم‌های دیجیتال عمدتاً از مبنای ۲ برای نمایش ارقام، اعداد و انجام اعمال محاسباتی استفاده می‌کنند؛ زیرا پیاده سازی فیزیکی آن‌ها با استفاده از مدارات الکترونیکی و ترانزیستوری از سایر مبناها ساده‌تر است.

بیت یک رقم دودویی است و می تواند دو مقدار ۰ و ۱ را داشته باشد؛ یک کد دودویی بسته به تعداد عناصر مجموعه‌ای که می‌خواهند کدگذاری شوند، می‌تواند از چندین بیت تشکیل شود.

مبناهای عددی- اعداد دودویی

جهت درک هرچه بیش‌تر مبناهای عددی، ابتدا عدد ۶۳۵۴ در مبنای ۱۰ را در نظر می‌گیریم؛ این عدد را می‌توان به شکل زیر نمایش داد:

$$۶\times۱۰۰۰+۳\times۱۰۰+۵\times۱۰+۴\times۱$$

این عدد به صورت ضرب سمبل‌های عددی آن در توان‌هایی از ۱۰ نمایش داده شده است و می‌توان آن را به شکل زیر نیز نمایش داد :

$$۶\times۱۰^{۳}+۳\times۱۰^{۲}+۵\times۱۰^{۱}+۴\times۱۰^{۰}$$

ارزش مکانی : هر رقم بنا به موقعیت خود در یک عدد وزنی دارد. مثلاً ارزش ۶ در ۶۳۵۴ برابر ۱۰۰۰ واحد و ارزش ۵ برابر ۱۰ واحد است.

به طور کلی هر عدد در مبنای مفروض r را به صورت حاصل‌ضرب توان‌های r در سمبل‌های مربوطه‌اش بیان می‌گردد. اعداد اعشاری نیز به همین روش می‌توانند نمایش داده شوند.

سایر مبناها :

سمبل‌های عددی در هر مبنای مفروض r می‌توانند مقادیری بین 0 تا r-1 داشته باشند. به‌عنوان مثال به مبناهای زیر توجه کنید :

در مبنای ۱۰ سمبل ها عبارتند از	۰ , ۱ , ۲ , ۳ , ۴ , ۵ , ۶ , ۷ , ۸ , ۹
در مبنای ۸ سمبل ها عبارتند از	۰ , ۱ , ۲ , ۳ , ۴ , ۵ , ۶ , ۷
در مبنای ۲ سمبل ها عبارتند از	۰ , ۱

قرارداد۱ : مبنای ۱۰ مبنای پایه بوده و برای نمایش اعداد در هر مبنای دیگر، آن‌ها را داخل پرانتز قرار داده و مبنا را به صورت اندیس جلوی آن‌ها می‌نویسیم.

مثال : $(۷۶۰)_{۸}$ $(۲۰۳۴)_{۵}$

قرارداد۲ : برای نمایش سمبل‌ها مبناهای زیر ۱۰ از ارقام استفاده می‌کنیم؛ اما در مبناهای بزرگ‌تر از ۱۰ مانند ۱۶، تا ضریب ۹ از ارقام استفاده می‌کنیم و به جای سایر سمبل‌ها، حروف الفبای لاتین را قرار می‌دهیم؛ بنابراین در مبنای ۱۶ سمبل‌های عددی عبارتند از :

۰،۱،۲،۳،۴،۵،۶،۷،۸،۹،A،B،C،D،E،F

مبنای ۲ : در این مبنا که معمولا برای نمایش اعداد در سیستم‌های دیجیتال استفاده می‌شود، سمبل‌های عددی فقط می‌توانند مقادیر ۰ و ۱ را اختیار کنند.

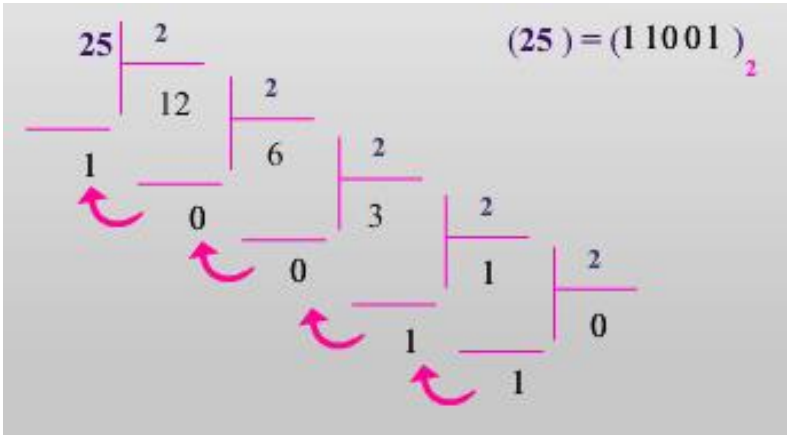
مثال : $(11001)_2$ $(1001101)_2$

ارزش مکانی بیت‌ها توان‌هایی از دو می باشد. در جدول زیر، توان‌های دو برای اعداد صفر تا ۱۰ مشاهده می‌شوند؛

	0	1	2	3	4	5	6	7	8	9	10
2	1	2	4	8	16	32	64	128	256	512	1024

تبدیل مبنای اعداد

تبدیل از مبنای ۱۰ به سایر مبناها : برای این منظور از تقسیمات متوالی استفاده می‌کنیم. به‌عنوان مثال برای تبدیل عدد ۲۵ در مبنای ۱۰ به معادل آن در مبنای ۲ به شکل زیر دقت نمایید:

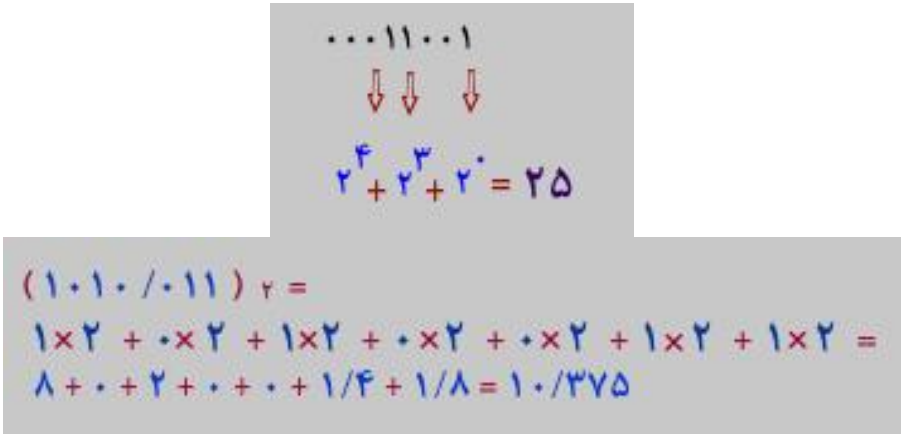


همان‌گونه که در شکل نیز مشاهده می‌شود، در هر مرحله عدد را به مبنای خواسته شده تقسیم می‌کنیم. سپس باقیمانده را نگه‌داشته و با خارج قسمت جدید مراحل را تکرار می‌کنیم. شرط اتمام عملیات در این حالت، کوچک‌تر شدن خارج قسمت جدید از مبنای خواسته شده و یا ۰ شدن آن است.

توجه : اولین باقیمانده‌ی حاصل، کم‌ارزش‌ترین رقم در مبنای خواسته شده است.

تبدیل مبنای اعداد اعشاری : برای این منظور، قسمت صحیح عدد را به روش تقسیمات متوالی به مبنای خواسته شده می‌بریم و برای تبدیل قسمت اعشاری، از روش ضرب‌های متوالی استفاده می‌کنیم، به این صورت که در هر مرحله، قسمت اعشاری عدد را در مبنای خواسته شده ضرب می‌کنیم. قسمت صحیح حاصل را نگه داشته و مجدداً قسمت اعشاری را در مبنای مقصد ضرب می‌کنیم. شرط پایان عملیات، ۰ شدن قسمت اعشاری و یا رسیدن به یک دقت مناسب است. به مثال زیر دقت نمایید :

تبدیل از سایرمبناها به ۱۰ : برای این منظور ارزش مکانی رقم‌ها را با هم جمع می کنیم، با توجه به‌این‌که ارزش مکانی رقم‌های اعشاری توان‌های منفی از مبنا هستند. به عبارت دیگر می‌توان عدد داده شده در هر مبنا را به صورت حاصلضرب سمبل های عدد در توان‌های مبنا نوشته و سپس جملات حاصلضرب را با یکدیگر جمع نمود. به مثال زیر توجه کنید :



تبدیل از مبناهای غیر ۱۰ به یکدیگر : روش اصولی وسیستماتیک برای این عمل، تبدیل از مبنای اول به مبنای ۱۰ و سپس از مبنای ۱۰ به مبنای خواسته شده می‌باشد.

تبدیل از مبناهای ۸ و ۱۶ به مبنای ۲ و بالعکس : تبدیل از مبنای ۲ به ۸ و ۱۶ نقش عمده‌ای در کامپیوترهای دیجیتال بازی می‌کند. برای انجام این تبدیل مبناها نیازی به مبنای میانی ۱۰ نیست و می‌توان مستقیماً این عمل را با توجه به رابطه‌ی توانی که بین عدد ۲ با اعداد ۸ و ۱۶ برقرار است ($۸ = ۲^۳$ و $۱۶ = ۲^۴$)، انجام داد. به عبارت دیگر، هر رقم در مبنای ۸ برابر است با ۳ رقم در مبنای ۲ و هر رقم در مبنای ۱۶ برابر است با ۴ رقم در مبنای ۲.

جداول زیر رابطه میان اعداد در مبناهای ۸، ۱۶، و ۲ را بهتر بیان می‌کنند؛

جدول مبنای ۱۶

مبنای ۱۰	مبنای ۲	مبنای ۱۶
۰	۰۰۰۰	۰
۱	۰۰۰۱	۱
۲	۰۰۱۰	۲
۳	۰۰۱۱	۳
۴	۰۱۰۰	۴
۵	۰۱۰۱	۵
۶	۰۱۱۰	۶
۷	۰۱۱۱	۷
۸	۱۰۰۰	۸
۹	۱۰۰۱	۹
۱۰	۱۰۱۰	A
۱۱	۱۰۱۱	B
۱۲	۱۱۰۰	C
۱۳	۱۱۰۱	D
۱۴	۱۱۱۰	E
۱۵	۱۱۱۱	F

جدول مبنای ۸

مبنای ۸	مبنای ۱۰	مبنای ۲
۰	۰	۰۰۰
۱	۱	۰۰۱
۲	۲	۰۱۰
۳	۳	۰۱۱
۴	۴	۱۰۰
۵	۵	۱۰۱
۶	۶	۱۱۰
۷	۷	۱۱۱

برای تبدیل یک عدد از مبنای ۱۶یا ۸ به مبنای ۲ کافیست از جداول فوق معادل مبنای ۲ هر رقم را پیدا کرده و به جای آن قرار دهیم.

$$(۵۲۱F)_{۱۶} = (۰۱۰۱،۰۰۱۰،۰۰۰۱،۱۱۱۱)_{۲}$$

$$(۳۲۷)_{۸} = (۰۱۱،۰۱۰،۱۱۱)_{۲}$$

برای تبدیل یک عدد از مبنای ۲ به مبنای ۸ از دو سمت نقطه اعشار شروع می‌کنیم و به دو سمت راست و چپ بیت ها را به صورت دسته‌های سه‌تایی جدا می‌کنیم. اگر تعداد ارقام اعشار و یا صحیح یک عدد مضربی از ۳ نباشد، به ترتیب به صفرهای کم ارزش بعد از ممیز و صفرهای پر ارزش قبل از ممیز اضافه می‌کنیم. سپس معادل مبنای ۸ هر دسته از بیت‌ها را می‌نویسیم.

برای تبدیل یک عدد از مبنای ۲ به مبنای ۱۶ از دو سمت نقطه اعشار شروع می‌کنیم و به دو سمت راست و چپ بیت ها را به صورت دسته‌های چهارتایی جدا می‌کنیم. اگر تعداد ارقام اعشار و یا صحیح یک عدد مضربی از ۴ نباشد، به ترتیب صفرهای کم ارزش بعد از ممیز و صفرهای پر ارزش قبل از ممیز اضافه می‌کنیم. سپس معادل مبنای ۱۶ هر دسته از بیت‌ها را می‌نویسیم.

$$(۳۲۲/۳۲۴)_{۸} = (۰۱۱,۰۱۰,۰۱۰/۰۱۱,۰۱۰,۱۰۰)_{۲} = (۱۱۰۱۰۰۱۰/۰۱۱۰۱۰۱)_{۲}$$

$$(۲CA/۹E۶)_{۱۶} = (۰۰۱۰,۱۱۰۰,۱۰۱۰/۱۰۰۱,۱۱۱۰,۰۱۱۰)_{۲} = (۱۰۱۱۰۰۱۰۱۰/۱۰۰۱۱۱۱۰۰۱۱)_{۲}$$

توجه : مبناهای هشت و شانزده اغلب جهت اختصار و کم شدن حجم اعداد مبنای دو استفاده می‌شوند و کم‌تر به‌عنوان مبنای محاسبات می‌باشند و در ضمن تبدیل مبنای هشت به مبنای شانزده و بر عکس از طریق مبنای دو ساده‌تر خواهد بود.

اعداد دودویی علامت‌دار و حساب دودویی

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- آشنایی با متمم‌ها (متمم‌ها)
- طرز متمم گرفتن از اعداد
- اعداد دودویی علامت دار
- انجام اعمال حسابی برای اعداد دودویی علامت دار
- خطای سرریز و تشخیص آن
- خلاصه درس
- آزمون

متمم‌ها(متمم‌ها)

متمم‌ها در کامپیوترهای دیجیتال برای ساده کردن برخی عملیات مانند عمل تفریق و عملیات منطقی به کار می‌روند. با بهره‌گیری از متمم‌ها، پیاده سازی سخت‌افزاری عملیات تفریق در کامپیوترها به سادگی و کمک به مدارات جمع‌کننده امکان‌پذیر خواهد بود.

در هر مبنای مفروض r دو نوع متمم برای هر عدد تعریف می‌شود :

۱. متمم مبنا (متمم r)

۲. متمم مبنای کاهش یافته (متمم r-1)

فرمول کلی متمم‌ها : عدد N دارای n رقم صحیح و m رقم اعشار در مبنای r مفروض است. در اینصورت :

مکمل r-۱ عدد N ← $r^n - N - r^{-m}$

مکمل r عدد N ← $r^n - N$

راه حل ساده‌تر : برای به دست آوردن متمم‌ها در مبنای ۲ می‌توان به‌صورت‌های زیر عمل نمود :

متمم ۱ یک عدد در مبنای ۲ : تمام بیت های عدد را از ۱ کم می‌کنیم؛ به عبارت دیگر تمام بیت های ۰ را به ۱ وتمام بیت های ۱ را به ۰ تبدیل می‌کنیم.

متمم ۲ یک عدد در مبنای ۲ : از سمت راست عدد شروع می‌کنیم. صفرهای کم ارزش و اولین یک کم ارزش را دست‌نخورده باقی می‌گذاریم. سپس تمام بیت‌های ۰ را به ۱ وتمام بیت های ۱ را به ۰ تبدیل می‌کنیم.

از میان مبناها مبحث متمم‌ها در مبنای ۲ اهمیت بیش‌تری دارد و در ادامه با مثال‌هایی به نحوه به‌دست‌آوردن متمم‌ها در این مبنای ۲ می‌پردازیم.

مثال:

11001001 $\xrightarrow{\text{متمم ۲}}$ 00110111

11110000 $\xrightarrow{\text{متمم ۲}}$ 00010000

00000000 $\xrightarrow{\text{متمم ۲}}$ 00000000

10000000 $\xrightarrow{\text{متمم ۲}}$ 10000000

10101110 $\xrightarrow{\text{متمم ۲}}$ 01010010

11111111 $\xrightarrow{\text{متمم ۲}}$ 00000001

اعداد دودویی علامت‌دار

اعداد صحیح مثبت و صفر را می‌توان با اعداد بی‌علامت نشان داد. اما برای ذخیره اعداد منفی بر روی کامپیوترها، روش های مختلفی پیشنهاد شده است. ۳ روش برای نمایش و ذخیره اعداد دودویی در کامپیوترها وجود دارد :

۱. روش مقدار- علامت دار : در این روش، چپ‌ترین بیت برای تعیین علامت استفاده می‌شود که به آن (بیت علامت) می‌گویند؛ در سایر بیت‌ها قدر مطلق عدد نمایش داده می‌شود. این سیستم عدد را با تغیر علامتش منفی می‌کند. اما دو روش دیگر، عدد را با متمم‌سازی منفی می‌کنند.

۲. روش متمم ۱ علامت دار : در این روش، برای منفی کردن یک عدد، ابتدا معادل مثبت آن را نوشته و سپس از آن متمم ۱ می‌گیریم.

۳. روش متمم ۲ علامت دار : در این روش، برای منفی کردن یک عدد، ابتدا معادل مثبت آن را نوشته و سپس از آن متمم ۲ می‌گیریم.

توجه ۱ : در تمامی روش‌های نمایش اعداد دودویی علامت‌دار، اعداد مثبت به یک شکل نمایش داده می‌شوند؛ تفاوت در نحوه نمایش اعداد منفی است.

توجه ۲ : در تمامی روش های نمایش اعداد دودویی علامت‌دار، چپ‌ترین بیت به عنوان بیت علامت استفاده می‌شود. این بیت در تمام اعداد مثبت برابر صفر و در تمام روش‌ها برای اعداد منفی برابر ۱ است.

از میان روش‌های فوق، روش اول در کامپیوترها کاربرد ندارد. متمم ۱ نیز مشکلاتی به بار می‌آورد و بیش‌تر برای اعمال منطقی مفید است. معمولا در اکثر سیستم‌های کامپیوتری از سیستم نمایش متمم ۲ برای نمایش اعداد علامت‌دار و محاسبات استفاده می‌شود.

قرارداد : از این پس فرض بر این است که اعداد علامت‌دار به روش متمم ۲ نمایش داده می‌شوند.

عملیات حسابی : معمولاً برای انجام عمل تفریق در مبنای ۲، از عمل جمع استفاده می شود؛ زیرا پیاده‌سازی سخت‌افزاری مدارات تفریق کننده به صرفه نمی‌باشد؛ از طرفی انجام عملیات تفریق به روش متمم ۲ دارای سخت‌افزار ساده‌تری است. بنابراین به کمک متمم‌ها و عمل جمع، می‌توان تفریق را انجام داد.

تفریق اعداد دودویی بدون علامت : برای تفریق دو عدد بدون علامت M-N ابتدا متمم ۲ عدد N را بدست آورده و سپس با M جمع می‌کنیم.

اگر $M \geq N$ باشد، عمل جمع یک رقم نقلی انتهایی تولید می‌کند که نشانگر مثبت بودن حاصل است و باید چشم‌پوشی شود؛

اگر M<N باشد، هیچ رقم نقلی انتهایی تولید نشده و نشانگر این است که حاصل، یک عدد منفی به صورت متمم ۲ می‌باشد.

مثال:

با فرض دو عدد دودویی $x = 84$ $(x = 1010100)$ و $y = 67$ $(y = 1000011)$ تفریق‌های زیر را به روش متمم ۲ انجام دهید:

(الف) X-Y ، (ب) Y-X

جواب (الف):

عدد x را با متمم ۲ عدد y جمع می‌کنیم:

$$x = 1010100$$
$$+\ 0111101$$
$$0010001$$

رقم نقلی ← 1 رقم نقلی را حذف می‌کنیم.

$x - y = 0010001 = 17$

جواب (ب):

رقم نقلی انتهایی وجود ندارد و بیت سمت چپ نتیجه برابر ۱ است. پس حاصل یک عدد منفی می‌باشد.
از حاصل متمم ۲ می‌گیریم. برابر ۱۷ خواهد شد. در نتیجه عدد نتیجه ۱۷- بوده است.

عدد y را با متمم ۲ عدد x جمع می‌کنیم:

$$1000011$$
$$+\ 0101100$$
$$1101111$$

$1101111 \xrightarrow{\text{متمم ۲}} 0010001$

$-۱۷ \qquad +۱۷$

جمع اعداد دودویی علامت‌دار : جمع دو عدد دودویی علامت‌دار که در آن، اعداد منفی به فرم متمم ۲ هستند، از جمع تمام بیت های دو عدد منجمله بیت‌های علامت آن‌ها حاصل می‌شود و از رقم نقلی حاصل از بیت علامت باید چشم‌پوشی شود.

مثال:

ابتدا اعداد ۶+ و ۱۳+ را در ۸ بیت نمایش داده و سپس از روی اعداد حاصل، اعداد ۶- و ۱۳+ را نمایش دهید. در ادامه چهار عمل جمع زیر را انجام دهید.

$+13 : 00001101 \xrightarrow{\text{متمم ۲}} -13 : 11110011$

11111010	(-6)
+ 00001101	+(+13)
00000111	+7

نقلی را نادیده می‌گیریم.

00000110	(+6)
+11110011	+(-13)
11111001	-7

حاصل یک عدد منفی است زیرا بیت سمت چپ برابر ۱ است که این عدد ۷- می‌باشد.

$+6 : 00000110 \xrightarrow{\text{متمم ۲}} -6 : 11111010$

00000110	(+6)
+ 00001101	+(+13)
00010011	+19

حاصل یک عدد مثبت می‌باشد.

11111010	(-6)
+11110011	+(-13)
11101101	-19

نقلی را نادیده می‌گیریم.
حاصل عدد ۱۹- به فرم متمم ۲ می‌باشد.

تمرین : عملیات زیر را در مبنای ۲ انجام دهید :

$-96 + 70$ $\quad -96 + (-70)$ $\quad 96 + (-70)$ $\quad 96 + 70$

تفریق اعداد دودویی علامت‌دار : تفریق دو عدد دودویی علامت‌دار که در آن، اعداد منفی به فرم متمم ۲ هستند، به این صورت انجام می‌شود : متمم ۲ عدد دوم را به دست آورده و حاصل را به عدد اول می‌افزاییم. در این حالت نیز از رقم نقلی حاصل از بیت علامت باید چشم‌پوشی شود.

نحوه انجام این عمل با روابط زیر بیان می‌شود:

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

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

خطای سرریز (OverFlow):

هرگاه دو عدد n رقمی با هم جمع شوند و حاصل جمع n+1 رقم را اشغال کند، سرریز رخ خواهد داد. سرریز می‌تواند در جمع دو عدد دودویی علامت‌دار یا بدون علامت رخ دهد؛ در جمع دو عدد بی علامت، یک سرریز از نقلی با ارزش‌ترین مکان تشخیص داده می‌شود.

سرریز در جمع اعداد علامت‌دار زمانی رخ می‌دهد که حاصل جمع دو عدد منفی، مثبت یا حاصل جمع دو عدد مثبت، منفی شود که در این حالت رقم نقلی انتهایی هیچ سرریزی را مشخص نمی‌کند. سرریز زمانی که یکی از اعداد مثبت و دیگری منفی باشد، رخ نخواهد داد.

نحوه تشخیص سرریز در جمع اعداد دودویی علامت‌دار : وضعیت سرریز را می‌توان با وجود رقم نقلی به بیت علامت و نقلی خروجی از بیت علامت مشاهده کرد. اگر این دو نقلی یکی نباشند، یک سرریز رخ داده است؛ به این معنی که علامت نتیجه عوض شده است. مثال زیر یک نمونه خطای سرریز را نشان می‌دهد:

۷۵ + ۷۵ =

$(۰۱۰۰۱۰۱۱)_۲ + (۰۱۰۰۱۰۱۱)_۲ = (۱۰۰۱۰۱۱۰)_۲$

در مثال فوق، بیت یکِ سمت چپ نشانگر این است که عدد حاصل منفی است، درصورتی که دو عدد مثبت با هم جمع شده است.

راه حل برطرف کردن خطای سرریز افزایش تعداد بیت‌های سیستم است. برای مثال عدد بالا را ده بیتی می‌کنیم تا مثبت شود ؛ یعنی بیت سمت چپ آن صفر می‌شود.

۷۵ + ۷۵ =

$(۰۰۰۱۰۰۱۰۱۱)_۲ + (۰۰۰۱۰۰۱۰۱۱)_۲ = (۰۰۱۰۰۱۰۱۱۰)_۲$

کدهای دودویی

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- آشنایی با کدهای دودویی
- انواع کدهای دودویی
- خواص کدهای دودویی
- آشنایی با مفاهیم تشخیص خطا
- آشنایی با بیت توازن و توان زوج و فرد
- خلاصه درس
- آزمون

کدهای دودویی

یک کد دودویی n بیتی، گروهی متشکل از n بیت است که ۲^n ترکیب ممکن از یک‌ها و صفرها را داراست و هر ترکیب یک عنصر از مجموعه کد شده را نمایش می‌دهد.

بنابراین، حداقل تعداد بیت‌های لازم برای ۲^n کد مجزا، برابر n است. همچنین، برای مجموعه‌هایی که تعداد عناصر آنها دقیقاً توانی از ۲ نمی‌باشد، برای یافتن مقدار n باید از رابطه زیر استفاده نمود :

$$2^{n-1} < m \leq 2^n$$

در رابطه فوق، m تعداد عناصر مجموعه ذکر شده می‌باشد. برای مثال، جهت کدگذاری مجموعه حروف الفبای لاتین به ۵ بیت نیاز داریم؛ زیرا :

$$۲^۴ < ۲۶ < ۲^۵ \longrightarrow ۱۶ < ۲۶ < ۳۲$$

انواع کدهای دودویی :

از آن‌جایی که بسیاری از انسان‌ها به سیستم دهدهی عادت دارند، بهتر است که اجرای همه محاسبات به دودویی و سپس تبدیل نتایج دودویی به دهدهی انجام شود. بنابراین در این روش باید اعداد دهدهی را در کامپیوتر ذخیره کرده تا بتوانند به اعداد دودویی تبدیل شوند. از طرفی کامپیوتر می‌تواند فقط با ارقام دودویی کار کند. بنابرین ارقامم دهدهی با کدی مرکب از صفر و یک نمایش داده می‌شوند.

کدهای دودویی برای ارقام دهدهی به حداقل ۴ بیت برای هر رقم نیاز دارند. با ایجاد ۱۰ ترکیب مختلف در ۴ بیت، کدهای دهدهی مختلف را می‌توان ایجاد کرد که هر کد تنها ۱۰ ترکیب بیتی از ۱۶ ترکیب ممکن در ۴ بیت را به کار می‌برد.

یکی از پرکاربردترین کدهای دهدهی مورد استفاده برای نمایش ارقام دهدهی، کد **BCD** است. **BCD:(Binary Coded Decimal)** به معنای دهدهی کد شده به دودویی می‌باشد. کد **BCD** معادل ارقام صفر تا نه در جدول زیر مشاهده می‌شود؛

دهدهی	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹
BCD	۰۰۰۰	۰۰۰۱	۰۰۱۰	۰۰۱۱	۰۱۰۰	۰۱۰۱	۰۱۱۰	۰۱۱۱	۱۰۰۰	۱۰۰۱

توجه ۱ : اعداد **BCD** اعداد دهدهی هستند، نه اعداد دودویی؛ هرچند که آنها در ساختارشان از بیت استفاده می‌کنند. تنها تفاوت بین یک عدد دهدهی و **BCD** در این است که در اعداد دهدهی از سمبل‌های صفر تا نه و در اعداد **BCD** از سمبل‌های **0000،0001،....،1001** استفاده می‌شود.

توجه ۲ : هرگاه عدد دهدهی در **BCD** بین صفر تا نه باشد، با عدد دودویی‌اش معادل است.

تبدیل اعداد دهدهی به BCD : برای این کار از جدول کدهای **BCD** ، معادل **BCD** هر یک از ارقام عدد را به دست آورده و کنار هم قرار می‌دهیم. **برای** نمایش یک عدد k رقمی در **BCD** به **4k** بیت نیاز داریم.

مثال :

$(236)_{10} = (0010,0011,0110)_{BCD}$
$(4185)_{10} = (0100,0001,1000,0101)_{BCD}$

به مثال‌های زیر توجه نمایید:

6 $\xrightarrow{BCD}$ 0110

12 $\xrightarrow{BCD}$ 00010010

753 $\xrightarrow{BCD}$ 011101010011

2948 $\xrightarrow{BCD}$ 0010100101001000

سایر کدهای دهدهی : در جدول زیر کد BCD به همراه ۳ کد دهدهی دیگر نشان داده شده‌اند. همانگونه که در جدول نیز مشخص است، در هر یک از کدها ۶ ترکیب بیتی به کار نرفته وجود دارد.

دهدهی	BCD	EX-3	8 4-2-1	2 4 2 1
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	0111	0100	0100
5	0101	1000	1011	1011
6	0110	1001	1010	1100
7	0111	1010	1001	1101
8	1000	1011	1000	1110
9	1001	1100	1111	1111

حال با مثالی، نحوه نمایش یک عدد را در هر یک از کدهای فوق مشخص می‌کنیم؛

$$792 = (\,0111\ \ 1001\ \ 0010\,)_{BCD}$$
$$= (\,1010\ \ 1100\ \ 0101\,)_{EX\text{-}3}$$
$$= (\,1001\ \ 1111\ \ 0110\,)_{8\ 4\text{-}2\text{-}1}$$
$$= (\,1101\ \ 1111\ \ 0010\,)_{2\ 4\ 2\ 1}$$

کدهای وزن‌دار : در یک کد وزن‌دار، به هر مکان از بیت‌ها وزنی اختصاص داده شده است. کدهای **BCD**، ۲۴۲۱ و ۸۴-۲-۱ از جمله کدهای وزین هستند.

کدهای خود متمم : در این کدها متمم ۹ عدد دهدهی مستقیماً از تغییر صفرها به یک و یک‌ها به صفر در کد حاصل می‌شود. کدهای ۲۴۲۱ و افزونی ۳ **(EX-3)** نمونه‌هایی از کدهای خود متمم هستند.

کد گِری (Gray Code) : گاهی اوقات بهتر است برای نمایش داده ها از کد گِری استفاده شود. مزیت کد گِری نسبت به کد دودویی این است که از هر کد به کد بعدی فقط یک بیت تغییر می‌یابد. کد گِری در کاربردهایی مورد استفاده است که رشته اعداد دودویی ممکن است در طول انتقال یا تبدیل از یک عدد به دیگری خطایی

تولید کنند. همچنین در جایی که شماره‌ها پشت سر هم ارسال شوند، این کد استفاده خوبی برای کنترل صحت اطلاعات خواهد داشت. کد گِری ۴ بیتی معادل اعداد صفر تا ۱۵ در جدول زیر مشاهده می‌شود؛

دهدهی	کد گِری
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

کدهای تشخیص خطا : برای تشخیص خطاها در مخابره یا پردازش داده‌ها، روش‌های مختلفی به کار می‌روند. یکی از این روش‌ها، استفاده از بیت توازن است؛ بیت توازن، بیتی اضافی است که حاوی پیامی بوده و طی آن تعداد یک‌های کل، زوج یا فرد خواهد شد.

بیت توازن (Parity bit) : بیت توازن، بیتی اضافی است که حاوی پیامی بوده و طی آن امکان کنترل خطا را به کد می دهد؛ یعنی اگر اشتباهاً یک به صفر یا برعکس تبدیل شده باشد قابل تشخیص خواهد بود. این بیت بیتی است که به کد اضافه می شود و تعداد یک‌ها را زوج یا فرد می‌کندکه به آن بیت توازن زوج یا فرد می‌گوییم. در نتیجه ما دو نوع توازن زوج و فرد خواهیم داشت؛

توازن زوج (Even Parity) : بیت توازن به نحوی صفر یا یک می‌شود که تعداد کل یک ها در پیام ارسالی همراه با بیت توازن، عدد زوجی شود.

توازن فرد (Odd Parity) : بیت توازن به نحوی صفر یا یک می‌شود که تعداد کل یک‌ها در پیام ارسالی همراه با بیت توازن، عدد فردی شود. جدول زیر نحوه تولید بیت‌های توازن زوج و فرد را برای کدهای ۴ بیتی نشان می-دهد؛

	پریتی فرد	پریتی زوج
کد ۴ بیتی	Po	Pe
0000	1	0
0001	0	1
0010	0	1
0011	1	0
0100	0	1
0101	1	0
0110	1	0
0111	0	1
1000	0	1
1001	1	0
1010	1	0
1011	0	1
1100	1	0
1101	0	1
1110	0	1
1111	1	0

فصل دوّم

جبر بول و گیت‌های منطقی

اصول، تئوری‌ها و خواص جبر بول

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- تعریف منطق دودویی، متغیرهای دودویی و عملیات منطقی
- درک ارتباط میان منطق دودویی و جبر بول دو ارزشی
- تعریف جبر بول و اصول هانتینگتون
- تفاوت‌های عمده میان جبر بول و جبر اعداد حقیقی
- تعریف جبر بول دو ارزشی
- اصول و تئوری‌های جبر بول واثبات آن‌ها
- روش‌های ساده‌سازی توابع منطقی

منطق دودویی : منطق دودویی شامل متغیرهای دودویی و عملیات منطقی است. متغیرهای دودویی با حروف الفبایی مانند A،B،C،x،y،z،.... نامگذاری می‌شوند و هر متغیر فقط دو مقدار 0 و 1 می‌تواند داشته باشد. سه نوع عملیات منطقی اصلی نیز وجود دارند که عبارتند از : AND ، OR و NOT

۱. **AND :** به وسیله یک "." نمایش داده می‌شود و x.y = z به معنی "z برابر با x AND y است" می‌باشد. در عمل AND، z=1 است اگر و فقط اگر x=1 و y=1 باشند. در غیر این‌صورت z=0 خواهد بود .

۲. **OR :** به وسیله یک "+" نمایش داده می‌شود و x+y = z به معنی "z برابر با x OR y است" می‌باشد. در عمل OR، z=1 است اگر x=1 یا y=1 و یا هر دو برابر 1 باشند. در غیر این‌صورت z=0 خواهد بود .

۳. **NOT :** عمل NOT یا متمم با یک علامت پریم یا یک خط بار نشان داده می‌شود و x’ = z به معنی "z برابر با NOT x است" می‌باشد. اگر x=0 باشد، z=1 و اگر x=1 باشد، z=0 خواهد شد.

در بخش‌های بعدی و مبحث جبر بول، عملیات منطقی و نحوه تعریف آن‌ها به طور مفصل شرح داده خواهد شد. منطق دودویی با تعریف ارائه شده در بالا، معادل با جبری به نام جبر بول است. منطق دودویی به روشی غیر مستدل تعریف شده است؛ اما در ادامه جبر بول به روش مستدل ریاضی بنا خواهد گردید. بیان غیر مستدل منطق دودویی، برای کاربرد جبر بول در مدارهای گیتی مفید است؛ اما روش مستدل جبر بول برای بیان و ایجاد تئوری‌ها و خواص سیستم‌های جبری به کار می‌رود.

جبر بول : جبر بول را می‌توان مانند هر سیستم ریاضی، به وسیله‌ی مجموعه‌ای از عناصر، یک مجموعه از الگوها و تعدادی اصول اثبات نشده یا بدیهیات تعریف نمود. پیش از پرداختن به تعریف جبر بول، ابتدا به یک سری تعاریف مقدماتی می پردازیم.

تعریف مجموعه : یک مجموعه از عناصر، کلکسیونی از اشیاء است که دارای خواص مشترکی باشند. در مجموعه‌ها دو تعریف عضویت و عدم عضویت مطرح است.

تعریف عملگر دودویی : یک عملگر دودویی روی یک مجموعه از عناصر مانند S، قانونی است که به هر جفت از عناصر S، یک عنصر منحصر به فرد از S را تخصیص دهد.

اصول به کار رفته یک سیستم ریاضی : مهم‌ترین اصول به کار رفته در فرموله کردن ساختارهای جبری عبارتند از :

- بسته بودن
- اصل شرکت‌پذیری
- اصل جابجایی
- عنصر شناسه
- معکوس
- اصل توزیع‌پذیری

تعریف جبر بول : جبر بول یک ساختار جبری است که با عناصر مجموعه B، همراه با دو عملگر دودویی (+) و (.) تعریف می‌شود به شرطی که اصول شش‌گانه زیر که به اصول هانتینگتون معروفند، در آن معتبر باشد :

۱	مجموعه نسبت به عملگر (+) بسته باشد.	مجموعه نسبت به عملگر (.) بسته باشد.
۲	یک عنصر شناسه ۰ برای (+) وجود داشته باشد.	یک عنصر شناسه ۱ برای (.) وجود داشته باشد.
۳	مجموعه نسبت به (+) دارای خاصیت جابجایی باشد. $X+Y=Y+X$	مجموعه نسبت به (.) دارای خاصیت جابجایی باشد. $X.Y=Y.X$
۴	(.) نسبت به (+) توزیع‌پذیر است. $x.(y+z) = (x.y) + (x.z)$	(+) نسبت به (.) توزیع‌پذیر است $x+(y.z) = (x+y) . (x+z)$
۵	برای هر عنصر عضو، عنصری مثل عضو وجود دارد به نحوی که :	
	A) $x + x' = 1$	B) $x . x' = 0$
۶	حداقل دو عنصر X و Y عضو وجود دارد به نحوی که : $x \neq y$	

از تعاریف فوق می‌توان تفاوت‌های زیر را میان جبر بول و جبر اعداد حقیقی بیان نمود :

۱. اصول هانتینگتون فاقد اصل شرکت‌پذیری است. اما این اصل برای جبر بول معتبر و برای هر دو عملگر قابل استنتاج است.
۲. اصل توزیع‌پذیری (+) روی (.) برای جبر بول معتبر است، ولی در جبر معمولی قابل قبول نیست.
۳. در جبر بول عملگرهای تفریق و تقسیم وجود ندارند.
۴. در جبر معمولی عملگر متمم تعریف نمی‌شود.
۵. جبر معمولی بر روی مجموعه اعداد حقیقی با بی‌نهایت عنصر تعریف می‌شود؛ اما جبر بول چنین نیست.

جبر بول دو ارزشی : بسته به انتخاب عناصر مجموعه B و قوانین عملیات، می‌توان چندین جبر بول را فرموله کرد.در ادامه درس مدارهای منطقی، تنها با جبر بول دو ارزشی متشکل از تنها دو عنصر سر و کار خواهیم داشت؛ جبر بول دو ارزشی روی مجموعه دو عنصری $B = \{0,1\}$ تعریف می‌شود. قوانین برای دو عملگر دودویی (+) و (.) در جدول های زیر مشاهده می‌شوند :

X AND Y

X	Y	XY
۰	۰	۰
۰	۱	۰
۱	۰	۰
۱	۱	۱

X OR Y

X	Y	X+Y
۰	۰	۰
۰	۱	۱
۱	۰	۱
۱	۱	۱

NOT X

X	X'
۰	۱
۱	۰

این قوانین دقیقاً منطبق با تعاریف اعمال AND، OR و NOT در ابتدای این بخش می‌باشند که در اشکال فوق به صورت خلاصه با استفاده از جدول درستی نمایش داده شده‌اند.

جدول درستی(Truth Table) : جدولی است متشکل از تمام ترکیبات ممکن متغیرها و بیانگر ارتباط بین مقادیر آن‌ها و نتایج حاصل از عمل مربوطه بر روی آن‌ها می‌باشد.

در ادامه این فصل خواهیم دید که تمامی عملیات منطقی توسط مدارات الکترونیکی و ترانزیستوری با نام دروازه-های منطقی (Gates) پیاده سازی می‌شوند.

اصول، تئوری‌ها و خواص جبر بول :

اصل دوگانی : این اصل بیان می‌دارد که هر عبارت جبری به دست آمده از اصول جبر بول، با تعویض عملگرها و شناسه‌ها باز هم معتبر باقی خواهد ماند. به عبارت دیگر، برای هر عبارت جبری یک دوگان وجود دارد و برای به دست آوردن آن کافیست عملگرهای AND و OR را به یکدیگر و 0 ها و 1 ها را نیز به یکدیگر تبدیل نماییم. با استناد به این اصل، تمام اصول و تئوری‌هایی که برای یکی از عملگرهای AND یا OR تعریف می‌شود، به راحتی قابل تعمیم به عملگر دیگر خواهد بود.

در جبر بول ۴ اصل و ۶ تئوری اساسی وجود دارد که در جدول های زیر مشاهده می‌شوند :

اصل ۲ : عنصر شناسه	1b) x.1= x	1a) x + 0 = x
اصل ۵ : عضو متمم	2b) x.x' = 0	2a) x + x' = 1
تئوری ۱	3b) x.x = x	3a) x + x = x
تئوری ۲	4b) x.0 = 0	4a) x + 1 = 1
تئوری ۳ : رجعت		5) (x')' = x
اصل ۳ : جابجایی	6b) x.y = y.x	6a) x + y = y + x
تئوری ۴ : شرکت‌پذیری	7b) x.(y.z) = (x.y).z	7a) x + (y+z) = (x + y) + z
اصل ۴ : توزیع‌پذیری یا پخش	8b) x + yz = (x+y)(x+z)	8a) x(y+z) = xy + xz
تئوری ۵ : جذب	9b) x (x+y) = x	9a) x + xy = x
تئوری ۶ : دمورگان	10b) (xy)' = x' + y'	10a) (x+y)' = x'y'

اصول جزو فرضیات هر ساختار جبری هستند و نیازی به اثبات ندارند. اما تئوری‌ها باید اثبات شوند.

تئوری‌های جبر بول به دو روش قابل اثبات هستند :

۱. با استفاده از جدول هم‌ارزی
۲. با استفاده از اصول و تئوری‌های اثبات شده قبلی

روش اول : در این روش برای اثبات تساوی طرفین یک عبارت جبری از مفهوم زیر استفاده می‌کنیم :

"دو عبارت جبری که جدول درستی یکسانی دارند؛ با هم معادلند."

در این روش تمام ترکیب‌های مختلف متغیرها را در جدول درستی می‌نویسیم تا نهایتاً تک تک عبارت‌های دو طرف رابطه تشکیل شوند؛ حال اگر این دو عبارت جدول درستی یکسانی داشته باشند، با هم معادلند.

مثال ۱ : درستی تئوری‌های دمورگان را از طریق جدول هم‌ارزی تحقیق نمایید.

X	Y	X+Y	(X+Y)'	X	Y'	X'Y'
۰	۰	۰	۱	۱	۱	۱
۰	۱	۱	۰	۱	۰	۰
۱	۰	۱	۰	۰	۱	۰
۱	۱	۱	۰	۰	۰	۰

X	Y	X .Y	(X .Y)'	X	Y'	X'+Y'
۰	۰	۰	۱	۱	۱	۱
۰	۱	۰	۱	۱	۰	۱
۱	۰	۰	۱	۰	۱	۱
۱	۱	۱	۰	۰	۰	۰

مثال۲ : درستی رابطه توزیع پذیری جمع نسبت به ضرب را از طریق جدول هم‌ارزی اثبات نمایید :

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

X	Y	Z	Y . Z	X+Y	X+Z	X+YZ	(X+Y) (X+Z)
۰	۰	۰	۰	۰	۰	۰	۰
۰	۰	۱	۰	۰	۱	۰	۰
۰	۱	۰	۰	۱	۰	۰	۰
۰	۱	۱	۱	۱	۱	۱	۱
۱	۰	۰	۰	۱	۱	۱	۱
۱	۰	۱	۰	۱	۱	۱	۱
۱	۱	۰	۰	۱	۱	۱	۱
۱	۱	۱	۱	۱	۱	۱	۱

روش دوم : در این روش برای اثبات درستی یک عبارت جبری، با استفاده از اصول جبر بول و تئوری‌های اثبات شده قبلی، یکی از طرفین تساوی را آن‌قدر تغییر می‌دهیم تا برابر با عبارت طرف دیگر شود. به کارگیری این روش مستلزم تسلط بر اصول و تئوری‌های جبر بول بوده و نیاز به تجربه کافی در به‌کارگیری آن‌ها در اثبات تساوی‌ها و ساده‌سازی‌ها دارد.

مثال ۱ : تئوری جذب را با استفاده از اصول و تئوری‌های اثبات شده‌ی قبلی اثبات نمایید.

$x+xy = x$
$x+xy = x.1 + xy$
$x+xy = x(1+y)$
$x+xy = x(y+1)$
$x+xy = x.1$
$x+xy = x$

مثال ۲ : درستی رابطه زیر را با استفاده از اصول و تئوری‌های اثبات شده‌ی قبلی اثبات نمایید.

$x + yz = (x+y)(x+z)$
$(x + y)(x + z) = xx + xz + xy + yz = x + xz + xy + yz = x(1 + y) + xz + yz =$
$x + xz + yz = x(1 + z) + yz = x + yz$

توجه : در ارزیابی عبارات جبر بول، تقدم اول با پرانتز، دوم با عمل متمم(NOT)، سوم با AND و چهارم با OR است. به بیان دیگر، عبارت داخل پرانتز باید قبل از سایر عملگرها ارزیابی شود.

به‌عنوان مثال در عبارت (X+Y)'، ابتدا داخل پرانتز ارزیابی می‌گردد و سپس نتیجه متمم می‌شود. یا در عبارت X'Y'، ابتدا متمم x و y ارزیابی شده و سپس حاصل AND می‌گردد.

توابع جبر بول، فرم‌های استاندارد و متعارف

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- تعریف تابع جبر بول
- روش‌های مختلف نمایش توابع به صورت‌های جدول درستی، عبارت‌های جبری و نمودار مداری
- روش‌های ساده‌سازی توابع منطقی
- کاربرد جبر بول در ساده‌سازی توابع (دست‌کاری جبری)
- متمم توابع و نحوهٔ به دست آوردن آن‌ها
- تعریف مینترم‌ها و ماکسترم‌ها
- نمایش توابع به صورت جمع مینترم‌ها و ضرب ماکسترم‌ها (فرم متعارف توابع)
- نحوهٔ تبدیل فرم‌های متعارف به یکدیگر
- نمایش توابع به صورت جمع حاصل ضرب‌ها و ضرب حاصل جمع‌ها (فرم استاندارد توابع)
- نحوه تعریف توابع مختلف برای ۲ متغیر و در حالت کلی n متغیر

توابع جبر بول : یک تابع جبر بول از متغیرهای دودویی، سمبل‌های عملیات منطقی و ثابت‌های 0 و 1 تشکیل می‌شود و به ازای هر ترکیب از متغیرهای دودویی تابع می‌تواند مقدار 0 یا 1 را برگرداند.

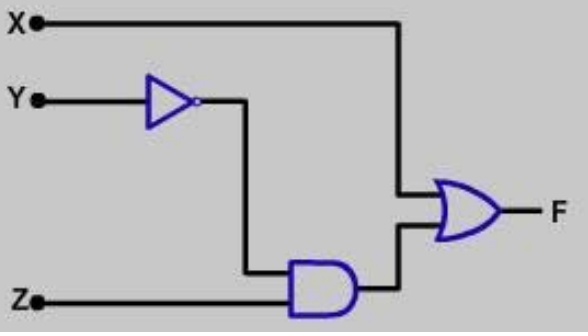
توصیف توابع جبر بول معمولاً از ۳ طریق زیر صورت می‌گیرد :

۱. جدول درستی : جدولی است متشکل از تمام ترکیبات ممکن برای متغیرهای دودویی و ستونی که مقدار نتایج را برای هر ترکیب نشان می‌دهد.
۲. عبارت جبری : از متغیرهای دودویی، سمبل‌های عملیات منطقی و ثابت‌های 0 و 1 تشکیل می‌شود.
۳. نمودار مداری : شکلی است گرافیکی متشکل از گیت های منطقی و خطوطی که اتصالات میان آن‌ها را مشخص می‌نماید. معادل هر عبارت جبری یک نمودار مداری وجود دارد و بالعکس.

مثال : تابع F به‌صورت عبارت جبری زیر بیان شده است. آن را در جدول درستی قرار داده و نمودار مداری آن را رسم نمایید.

F=X+Y'Z

X	Y	Z	F	F'
۰	۰	۰	۰	۱
۰	۰	۱	۱	۰
۰	۱	۰	۰	۱
۰	۱	۱	۰	۱
۱	۰	۰	۱	۰
۱	۰	۱	۱	۰
۱	۱	۰	۱	۰
۱	۱	۱	۱	۰



توجه مهم : برای نشان دادن هر تابع جبری به شکل جدول درستی تنها یک راه وجود دارد. اما برای نمایش تابع به صورت عبارت جبری، فرم‌های متفاوتی وجود دارد و معادل هر کدام نیز یک نمودار مداری.

عبارت‌های مختلف برای توابع جبر بول را می‌توان با استفاده از اصول و تئوری‌های جبر بول به دست آورده و به یکدیگر تبدیل نمود.

مثال : عبارت جبری تابع F در مثال فوق را ابتدا از روی جدول درستی به دست آورده و سپس با به کار بردن قوانین جبر بول، چند فرم جبری دیگر تابع را به دست آورید.

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

روش‌های ساده‌سازی توابع منطقی : در پیاده‌سازی مدارهای منطقی ، هدف از ساده‌سازی عبارت‌های جبری، رسیدن به عباراتی با کمترین تعداد ورودی‌ها و عملگرها است تا مدارات نهائی ساده‌تر و اقتصادی‌تر باشند.

روش‌های مختلف برای ساده‌سازی توابع جبر بول وجود دارند که مهم‌ترین آن‌ها به شرح زیرند :

۱. **قوانین جبر بول**
۲. **روش نقشه (جدول کارنو)**
۳. **برنامه‌های ساده‌سازی کامپیوتری**

روش ساده‌سازی با استفاده از قوانین جبر بول، یک روش سعی و خطا است که به دلیل کمبود قوانین خاص در پیش‌گویی مرحله‌ی بعدی ساده‌سازی، مشکل است. روش نقشه، روالی ساده و سیستماتیک برای ساده سازی ارائه می‌دهد و توابعی که تا ۵ متغیر دارند، قابل ساده‌سازی با این روش هستند؛ برای توابع پیچیده‌تر با بیش از ۵ متغیر، طراحان معمولاً از برنامه‌های ساده سازی کامپیوتری بهره می‌برند.

از میان روش‌های فوق، دو روش اول رایج‌تر می‌باشند. ساده سازی با استفاده از جبر بول را در این فصل شرح خواهیم داد و روش نقشه مبحث فصل بعدی خواهد بود.

کاربرد جبر بول در ساده‌سازی توابع (دست‌کاری جبری) : هدف از دست‌کاری جبری یک تابع منطقی، دستیابی به یک مدار نهایی ساده‌تر از طریق کاهش تعداد جملات یا تعداد لیترال‌ها (و یا هر دو) در عبارت جبری تابع می‌باشد. عمل ساده‌سازی با اعمال قوانین جبر بول در یک یا چند مرحله بر روی یک عبارت جبری انجام می‌گیرد.

به مثال‌های زیر توجه کنید:

مثال ۱ : $xyz + x'y + xyz' = xy(z + z') + x'y = xy.1 + x'y = xy + x'y = y(x + x') = y.1 = y$

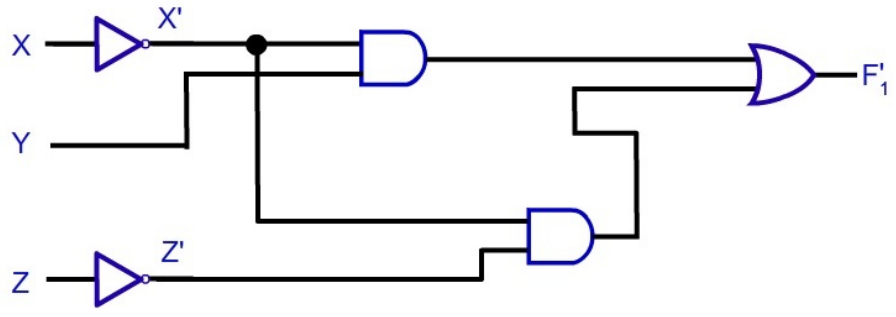
مثال ۲ : $A'C' + ABC + AC' = (A + A').C' + ABC = 1.C' + ABC = C' + (AB).C$
$= (C' + AB) . (C' + C) = (C' + AB).1 = C' + AB$

مثال ۳ : $(A' + C).(A' + C').(A + B + C'D) = (A' + CC').(A + B + C'D) = (A' + 0).(A + B + C'D)$
$= A'.(A + B + C'D) = AA' + A'B + A'C'D$
$= 0 + A'B + A'C'D = A'(B + C'D)$

تعریف لیترال : هر متغیر تک در یک جمله را لیترال می‌نامیم که ممکن است متمم شود یا نشود. به عنوان مثال تابع $F_1 = x + y'z$ دارای ۳ لیترال و تابع $F_2 = x.(y'z' + yz)$ دارای ۵ لیترال می‌باشند.

متمم توابع و نحوهٔ به‌دست‌آوردن آن‌ها : متمم تابع F را با F' نمایش می‌دهیم واز تعویض 0ها با 1ها در مقدار تابع F بدست می‌آید. روش اساسی به دست آوردن متمم تابع استفاده از تئوری‌های دمورگان چند متغیره است.

مثال : متمم تابع F_1 با ضابطه $F_1 = x + y'z$ را به دست آورده و نمودار مداری آن را رسم نمایید.



مثال : متمم تابع F_2 با ضابطه $F_2 = x.(y'z' + yz)$ را به دست آورید.

$F2' = [x.(y'z' + yz)]' = x' + (y'z' + yz)' = x' + (y'z')'.(yz)' = x' + (y+z).(y'+z')$

روال ساده‌تر برای به دست آوردن متمم یک تابع : این روش از تئوری دمورگان نتیجه شده و به این صورت است که ابتدا دوگان تابع و سپس متمم هر لیترال را به دست می‌آوریم.

مثال : متمم تابع F_2 با ضابطه $F_2 = x.(y'z' + yz)$ را با استفاده از دوگان‌ها و متمم‌های هر لیترال به‌دست آورید.

F2 = x.(y'z' + yz)

دوگان تابع ۲ : x + (y'+z').(y+z)

متمم هر لیترال : x' + (y+z).(y'+z')

فرم‌های استاندارد و متعارف توابع : همانگونه که پیش‌تر ذکر شد، با استفاده از اصول جبر بول می‌توان عبارت‌های جبری متفاوتی را برای یک تابع تعریف نمود. دسته‌ای از این عبارت‌ها به فرم‌های خاصی هستند و خواصی دارند که آن‌ها را از سایر نمایش‌های تابع متمایز می‌سازد. این فرم‌ها عبارتند از :

۱. فرم استاندارد توابع : در این فرم، جمله‌هایی که تابع را تشکیل می‌دهند ممکن است یک، دو یا هر تعدادی از متغیرها را دارا باشند. دو نوع فرم استاندارد برای توابع وجود دارد :

a) جمع حاصل ضرب‌ها (Sum of products) : یک عبارت بولی است شامل جملات حاصل ضرب (AND) که توسط عملگر جمع (OR) با یکدیگر ترکیب شده‌اند. هر یک از جملات می‌توانند شامل یک یا چند لیترال باشند. نمودار منطقی جمع حاصل ضرب‌ها متشکل از گروهی گیت AND است که به دنبال آن‌ها یک گیت OR می‌آید. مثال :

$F_1 = x + y'z$

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

b) ضرب حاصل جمع‌ها (Product of Sums) : یک عبارت بولی است شامل جملات حاصل جمع (OR) که توسط عملگر ضرب (AND) با یکدیگر ترکیب شده‌اند. هر یک از جملات می‌توانند شامل یک یا چند لیترال باشند. نمودار منطقی ضرب حاصل جمع‌ها متشکل از گروهی گیت OR است که به دنبال آن‌ها یک گیت AND می‌آید. مثال :

$F_3 = x.(y'+z).(x'+y+z')$ $F_4 = (x + z').(x + y').(y + z).x'$

۲. فرم متعارف توابع : در این فرم، جمله‌هایی که تابع را تشکیل می‌دهند باید تمام متغیرها را به شکل متمم و یا غیر متمم دارا باشند. دو نوع فرم متعارف برای توابع وجود دارد :

a) جمع مینترم‌ها b) ضرب ماکسترم‌ها

پیش از آنکه به تعریف این دو فرم توابع و خواص آن‌ها بپردازیم، ابتدا با مفهوم مینترم و ماکسترم برای توابع ۲ تا چند متغیره و نحوه به دست آوردن آن‌ها آشنا می‌شویم.

تعریف مینترم یا جمله ضرب استاندارد : از ترکیب n متغیر توسط عملگر AND، مینترم یا جمله ضرب استاندارد حاصل می‌شود. در جملات مینترم، هر متغیر می‌تواند با پریم یا بدون پریم ظاهر شود.

برای هر تابع n متغیره، 2^n مینترم وجود دارد. در هر جمله مینترم، تمام متغیرها وجود دارند و متغیر با مقدار 0 با پریم و متغیر با مقدار 1 بدون پریم ظاهر می‌شوند.

تعریف ماکسترم یا جمله جمع استاندارد : از ترکیب n متغیر توسط عملگر OR، ماکسترم یا جمله جمع استاندارد حاصل می‌شود. در جملات ماکسترم ، هر متغیر می‌تواند با پریم یا بدون پریم ظاهر شود.

برای هر تابع n متغیره، 2^n ماکسترم وجود دارد. در هر جمله ماکسترم ، تمام متغیرها وجود دارند و متغیر با مقدار 1 با پریم و متغیر با مقدار 0 بدون پریم ظاهر می‌شوند.

مینترم‌ها و ماکسترم‌ها برای توابع ۳ متغیره مطابق جدول زیر تعریف می‌شوند :

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

برای هر ترکیب از متغیرها، مینترم‌ها را با m_j و ماکسترم‌ها را با M_j نمایش می‌دهیم. J معادل دهدهی عدد دودویی مربوط به مینترم یا ماکسترم است.

نکته ۱ : هر تابع بولی را می‌توان به صورت جمع مینترم‌ها و ضرب ماکسترم‌ها درآورد. فرم‌های متعارف جمع مینترم‌ها و ضرب ماکسترم‌ها برای هر تابع جبری منحصر به فرد است.

نکته ۲ : بین هر مینترم m_j و ماکسترم متناظرش M_j رابطه زیر برقرار است : $M_j = m'_j$. به این معنی که هر مینترم متمم ماکسترم متناظرش است و بالعکس.

به دست آوردن جمع مینترم های تابع از روی جدول درستی : این عمل با تشکیل مینترم‌ها برای ترکیباتی از متغیرها که در تابع تولید 1 می‌کنند و سپس جمع آن‌ها توسط عملگر OR انجام می‌شود.

به دست آوردن ضرب ماکسترم های تابع از روی جدول درستی : این عمل با تشکیل ماکسترم‌ها برای ترکیباتی از متغیرها که در تابع تولید 0 می‌کنند و سپس ضرب آن‌ها توسط عملگر AND انجام می‌شود.

مثال : با توجه به جدول درستی زیر، تابع F را به صورت جمع مینترم‌ها و ضرب ماکسترم‌ها نمایش دهید.

X	Y	Z	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

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

$m_1 + m_4 + m_5 + m_6 + m_7 = \sum(1,4,5,6,7)$

$F = (x+y+z).(x+y'+z).(x+y'+z') = M_0.M_2.M_3 = \prod(0,2,3)$

در روابط فوق، روش‌های مختلف نمایش تابع به صورت جمع مینترم‌ها مشاهده می‌شوند. سمبل $\sum$ به معنی مجموع شماره مینترم های داخل پرانتز است.

در روابط فوق، روش‌های مختلف نمایش تابع به صورت ضرب ماکسترم‌ها مشاهده می‌شوند. سمبل $\prod$ به معنی ضرب شماره ماکسترم های داخل پرانتز است.

نحوه‌ی تبدیل فرم‌های متعارف به یکدیگر : برای تبدیل یک فرم متعارف به فرم متعارف دیگر، سمبل‌های $\sum$ و $\prod$ را با هم عوض می‌کنیم؛ سپس شماره‌های مفقود شده از فرم اصلی تابع را جایگزین شماره‌های قبلی می‌کنیم.

توجه داریم که برای یک تابع n متغیره، تعداد کل جملات برابر با 2^n (از 0 تا 2^n-1) است.

$F = \sum(1,4,5,6,7) \longrightarrow F = \prod(0,2,3)$

مثال :

نکته : اگر متمم تابع F را از روی جدول درستی تابع به صورت جمع مینترم‌ها به‌دست‌آورده و سپس از آن متمم بگیریم، تابع F به صورت ضرب ماکسترم‌ها به دست خواهد آمد. متناظراً، اگر متمم تابع F را از روی جدول درستی تابع به صورت ضرب ماکسترم‌ها به دست آورده و سپس از آن متمم بگیریم، تابع F به صورت جمع مینترم‌ها به دست خواهد آمد.

مثال : از روی جدول درستی زیر، ابتدا متمم تابع F را به صورت جمع مینترم‌ها به دست آورده و سپس با متمم‌گیری از آن، فرم ضرب ماکسترم‌های تابع F را به دست آورید.

X	Y	Z	F
۰	۰	۰	۱
۰	۰	۱	۱
۰	۱	۰	۰
۰	۱	۱	۰
۱	۰	۰	۱
۱	۰	۱	۰
۱	۱	۰	۰
۱	۱	۱	۱

$F' = x'yz' + x'yz + xy'z + xyz' = \sum(2,3,5,6) \rightarrow F = (x'yz' + x'yz + xy'z + xyz')'$

$= (x+y'+z).(x+y'+z').(x'+y+z').(x'+y'+z) = \prod(2,3,5,6)$

از مثال‌های فوق می‌توان روابطی را میان شماره مینترم‌ها و ماکسترم های یک تابع و متمم آن تابع نتیجه‌گیری نمود که با یک مثال به آن‌ها می‌پردازیم.

مثال : با استفاده از جدول درستی مثال قبل، تابع F و متمم آن را به صورت جمع مینترم‌ها و ضرب ماکسترم‌ها نمایش دهید.

X	Y	Z	F	F'
۰	۰	۰	۱	۰
۰	۰	۱	۱	۰
۰	۱	۰	۰	۱
۰	۱	۱	۰	۱
۱	۰	۰	۱	۰
۱	۰	۱	۰	۱
۱	۱	۰	۰	۱
۱	۱	۱	۱	۰

$F = \sum(۰,۱,۴,۷)$

$F = \prod(۲,۳,۵,۶)$

$F' = \sum(۲,۳,۵,۶)$

$F' = \prod(۰,۱,۴,۷)$

ارتباط میان فرم‌های استاندارد و متعارف توابع :

جمع مینترم ها، حالت خاصی از جمع حاصل ضرب‌ها است که در آن تمامی جملات حاصل ضرب ، مینترم‌های تابع هستند که تمام متغیرهای تابع در آن‌ها ظاهر شده است.

به همین ترتیب، ضرب ماکسترم‌ها ، حالت خاصی از ضرب حاصل جمع‌ها است که در آن تمامی جملات حاصل جمع ، ماکسترم‌های تابع هستند که تمام متغیرهای تابع در آن‌ها ظاهر شده است.

نکته ۱ : فرم‌های استاندارد و متعارف توابع، به نوع خاصی از پیاده‌سازی به نام مدارات دوسطحی یا دو طبقه منجر می‌شوند؛ در این نوع مدارات، سر راه ورودی های مدار تا خروجی ها تنها ۲ طبقه گیت AND-OR یا OR-AND وجود دارد که باعث می‌شود این‌گونه مدارات هنگام انتشار ورودی‌ها به سمت خروجی‌ها، کم‌ترین مقدار تاخیر را تولید کنند.

نکته ۲ : یک تابع بول می‌تواند به صورت غیر استاندارد و چند طبقه‌ای نیز بیان شود. هر چند این‌گونه عبارات با استفاده از اصول جبر بول از قبیل توزیع‌پذیری، به راحتی قابل تبدیل به فرم‌های استاندارد و متعارف معادل خواهند بود.

مثال : تابع F با عبارت جبری زیر را به فرم جمع حاصل ضرب‌ها تبدیل نمایید.

$F = AB + C(D+E) \rightarrow F = AB + CD + CE$

تبدیل عبارت‌های جبری به فرم‌های متعارف : یک عبارت جبری جهت تبدیل به فرم جمع مینترم‌ها ابتدا باید به شکل جمع حاصل ضرب‌ها درآید. هم‌چنین، جهت تبدیل یک عبارت به فرم ضرب ماکسترم‌ها، باید آن عبارت را به شکل ضرب حاصل جمع‌ها درآورد؛ سپس برای رسیدن به فرم مورد نظر، جملات را با استفاده از اصول جبر بول بسط می‌دهیم و متغیرهایی که در جملات نیستند را اضافه می‌کنیم تا فرم مورد نظر حاصل شود.

مثال : تابع $F_1 = (A' + B).(B' + C)$ را به فرم ضرب ماکسترم‌ها و تابع $F_2 = A(B+B') + (A+A')B'C$ را به فرم جمع مینترم‌ها تبدیل نمایید.

$$F_1 = (A' + B).(B' + C)$$
$$= (A' + B + CC')(B' + C + AA')$$
$$= (A'+B+C)(A'+B+C')(A+B'+C)(A'+B'+C)$$
$$= M_2.M_4.M_5.M_6 = \prod(2,4,5,6)$$

$$F_2 = A(B+B') + (A+A')B'C$$
$$= AB + AB' + AB'C + A'B'C$$
$$= AB(C + C') + AB'(C + C') + AB'C + A'B'C$$
$$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$$
$$= m_0 + m_1 + m_5 + m_6 + m_7 = \sum(1,4,5,6,7)$$

سایر اعمال منطقی : از آن‌جا که تابع در قبال هر مینترم می‌تواند 0 و یا 1 باشد و چون 2^n مینترم وجود دارد، لذا تعداد توابع ممکن که با n متغیر ایجاد می‌شوند، برابر با 2 به توان 2^n خواهد بود. به عنوان مثال برای ۲ متغیر، ۱۶ تابع مختلف تعریف می‌شود که در جدول زیر مشاهده می‌شوند.

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			AND					XOR	OR	NOR	XNOR					NAND	
																	

از میان ۱۶ تابع موجود در جدول فوق، ۸ تای آن‌ها به‌عنوان گیت‌های استاندارد در طراحی سیستم‌های دیجیتال به‌کارگرفته می‌شوند. پیاده‌سازی سایر توابع در قالب گیت‌های منطقی،از نظر اقتصادی و هزینه ساخت مقرون به صرفه نبوده و معمولاً آن‌ها را با استفاده از سایر گیت‌های منطقی می‌سازند. در زیر، جدول درستی، و شکل مداری این ۸ گتی منطقی را مشاهده می‌فرمایید.

AND

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

NAND

x	y	(x.y)'
0	0	1
0	1	1
1	0	1
1	1	0

OR

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

NOR

x	y	(x+y)'
0	0	1
0	1	0
1	0	0
1	1	0

XOR

x	y	x⊕y
0	0	0
0	1	1
1	0	1
1	1	0

XNOR

x	y	(x⊕y)'
0	0	1
0	1	0
1	0	0
1	1	1

NOT

x	x'
0	1
1	0

Buffer

x	F
0	0
1	1

تقویت ولتاژ و توان سیگنال

فصل سوّم

حداقل سازی در سطح گیت

روش نقشه یا جدول کارنو

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- مفهوم جدول کارنو و کاربرد آن در ساده‌سازی توابع جبر بول
- ارتباط جدول کارنو و جدول درستی
- تعریف همجواری در جدول کارنو
- نحوه به‌کارگیری قوانین جبر بول به صورت گرافیکی در جدول کارنو
- آشنایی با قواعد جدول کارنو
- کار با جداول کارنوی ۲ تا ۵ متغیره

روش نقشه (جدول کارنو) : همانگونه که در فصل قبل اشاره شد، روش ساده سازی با استفاده از قوانین جبر بول، به دلیل کمبود قوانین خاص در پیش‌گویی مرحله بعدی فرآیند ساده سازی، تضمینی برای به دست آوردن ساده‌ترین فرم تابع ندارد، اما روش نقشه یا جدول کارنو در واقع یک فرم گرافیکی و مصور از جدول صحت است و طوری مینترم‌ها را در کنارهم قرار می‌دهد که بتوان به سادگی آن‌ها را با هم ساده کرد و در صورتی‌که قوانین این روش را رعایت کنیم، ساده ترین فرم تابع به صورت تضمین شده حاصل خواهد شد.

روش نقشه که از این پس آن را جدول کارنو می‌نامیم، برای ساده سازی توابع ۲ تا ۵ متغیره بسیار مناسب است. در این روش، عبارات ساده شده حاصل از جدول کارنو، همیشه به یکی از دو فرم استاندارد جمع حاصل ضرب‌ها و یا ضرب حاصل جمع‌ها می‌باشد.

قرارداد : فرض بر این است که ساده‌ترین عبارت جبری، دارای حداقل جملات با کم‌ترین لیترال در هر جمله می‌باشد.

توجه مهم : ساده‌ترین عبارت به دست آمده از جدول کارنو منحصر به فرد نیست و ممکن است دو یا چند عبارت از جدول حاصل شوند که معیار حداقل‌سازی را برآورده سازند.

تعریف جدول کارنو : جدول کارنو نموداری است متشکل از مربعات که هر مربع، یک مینترم از تابع را نشان می‌دهد. در این جدول، ساده‌سازی با استفاده از مفهوم همجواری صورت می‌پذیرد.

تعریف مربع های همجوار : دو مربع را در جدول کارنو همجوار گوییم اگر تنها در یک متغیر اختلاف داشته باشند. به عبارت دیگر، تنها اختلاف دو مربع همجوار در جدول کارنو اینست که در یکی، متغیری با پریم و در دیگری بدون پریم ظاهر می‌شود، لذا با بهره گیری از اصول جبر بول می‌توان دو مربع همجوار را داخل یک پوشش قرار داده و متغیر غیر مشترک را در پوشش جدید حذف نمود.

قواعد جدول کارنو :

۱. در جدول کارنو، مینترم‌های همسایه فقط در یک متغیر با هم فرق دارند.
۲. در جدول کارنو، پوشش‌ها از خانه‌هایی تشکیل می‌شوند که مقدارتابع در آن‌ها برابر ۱ است.
۳. خانه‌های همجوار با یکدیگر می‌توانند یک پوشش را تشکیل دهند.
۴. هر پوشش باید شامل تعداد توان ۲ از خانه‌ها باشد.
۵. برای به‌دست‌آوردن تابع حداقل به ساده‌ترین فرم، باید پوشش‌های حداکثر با بیش‌ترین تعداد خانه ممکن را در نظر گرفت.
۶. هر خانه مورد نظر باید حداقل یک مرتبه پوشش داده شود، ولی به‌منظور ساده‌سازی بیش‌تر، می‌توان از یک خانه در بیش از یک پوشش استفاده نمود.
۷. هر چه تعداد بیش‌تری از مربعات همجوار در یک پوشش ترکیب شوند، جمله حاصل ضرب نتیجه، تعداد لیترال کم‌تری خواهد داشت.
۸. اگر در یک جدول کارنو، تمام خانه‌های جدول مقدار 1 داشته باشند، خروجی برابر تابع ثابت 1 خواهد بود.
۹. اگر در یک جدول کارنو، تمام خانه‌های جدول مقدار 0 داشته باشند، خروجی برابر تابع ثابت 0 خواهد بود.

ورودی‌ها و خروجی‌های جدول کارنو :

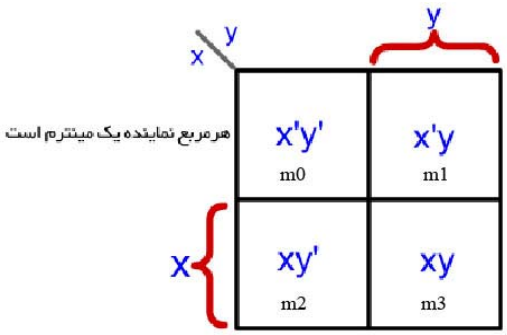
ورودی جدول کارنو می‌تواند به یکی از صورت‌های زیر باشد :

۱. جمع مینترم‌ها
۲. ضرب ماکسترم‌ها
۳. جمع حاصل ضرب‌ها
۴. ضرب حاصل جمع‌ها

عبارات ساده شده حاصل از جدول کارنو به یکی از صورت‌های زیر خواهند بود :

۱. جمع حاصل ضرب‌ها
۲. ضرب حاصل جمع‌ها

جدول کارنوی ۲ متغیره : برای توابع ۲ متغیره به‌کارمی‌رود و تعداد خانه‌های آن برابر بـا ۴(تعـداد مینتـرم‌هـای تابع ۲ متغیره) است. ساختار کارنوی ۲ متغیره در شکل مشاهده می‌شود.

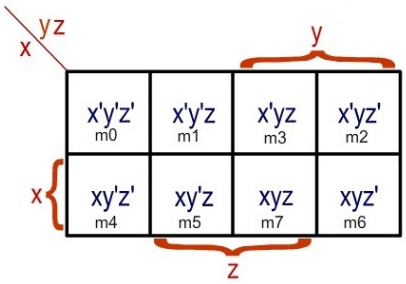


در کارنوی ۲ متغیره همجواری های زیر در خانه‌های جدول وجود دارند :

m_3 با m_2 - m_3 با m_1 - m_2 با m_1 - m_2 با m_0 - m_1 با m_0

جدول کارنوی ۳ متغیره : برای توابع ۳ متغیره به‌کارمی‌رود و تعداد خانه‌های آن برابر با ۸(تعـداد مینتـرم هـای تابع ۳ متغیره) است. ساختار کارنوی ۳ متغیره در شکل مشاهده می‌شود.

نکته مهم : در جداول کارنو با بیش از ۲ متغیر، متغیرها را در سـطرها و سـتون‌هـا بـه صـورت کـد گِـرِی تغییـر می‌دهیم تا همجواری میان مربع‌های همسایه در جدول به وجود آمده و خاصیت جدول کارنو در ساده‌سازی توابع حفظ شود.



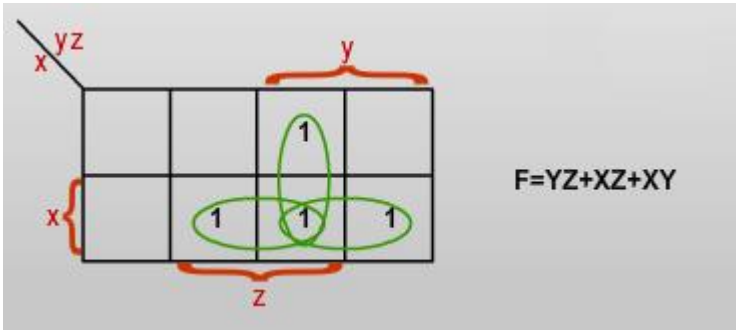
در کارنوی ۳ متغیره برخی همجواری ها به صورت زیر هستند :

```
m0,m1 : x'y'z' + x'y'z = x'y'
m0,m2 : x'y'z' + x'yz' = x'z'
m0,m4 : x'y'z' + xy'z' = y'z'
m1,m3 : x'y'z + x'yz = x'z
m0,m1, m4,m5 : x'y'z' + x'y'z + xy'z' + xy'z  = y'
m0,m2, m4,m6 : x'y'z' + x'yz' + xy'z' + xyz  = z'
m4,m5, m6,m7 : xy'z' + xy'z + xyz' + xyz  = x
```

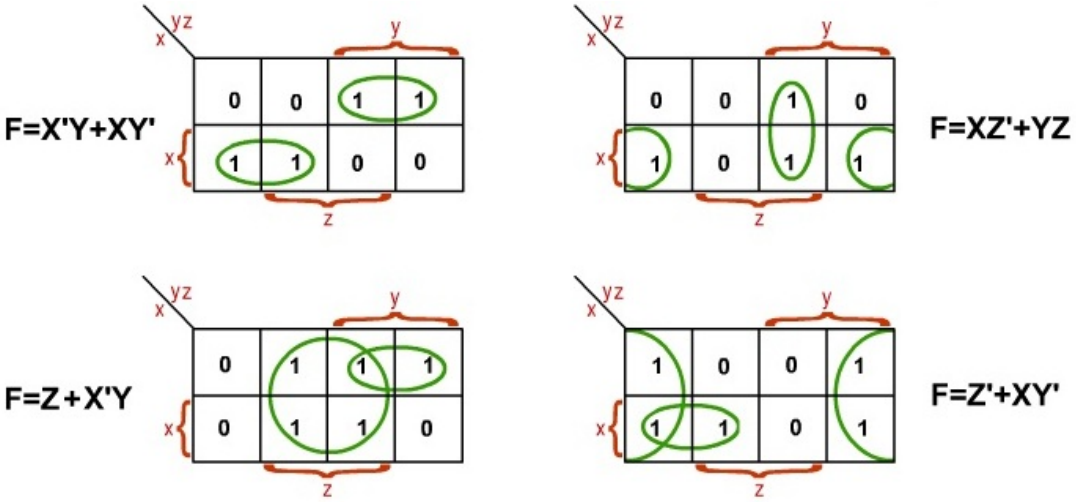
مثال ۱ : تابع زیر را با استفاده از جدول کارنو ساده نمایید.

$$F = X'YZ + XYZ + XYZ' + XY'Z$$

برای قرار دادن یک تابع به فرم جمع حاصل ضرب‌ها در جدول کارنو، محدوده هر جمله در تابع را در جدول مشخص می‌کنیم و در آن خانه‌ها 1 قرار می‌دهیم. در این مثال، ۴ جمله عبارت جبری تابع، هر کدام متناظر با یک مینترم تابع بوده و یک خانه در جدول کارنو را شامل می‌شوند.



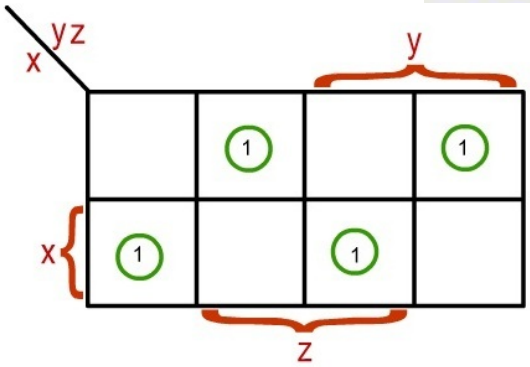
مثال ۲: ساده‌ترین عبارت تابع F را برای هر کدام از جداول کارنوی زیر به دست آورید.



مثال ۳: تابع زیر را با استفاده از جدول کارنو ساده نمائید.

$$F(x,y,z) = \sum(۱,۲,۴,۷)$$

جدول کارنو و عبارت تابع F به شکل زیر می باشند :



$$F = x'y'z + x'yz' + xy'z' + xyz$$
$$= x \oplus y \oplus z \quad (x \text{ XOR } y \text{ XOR } z)$$

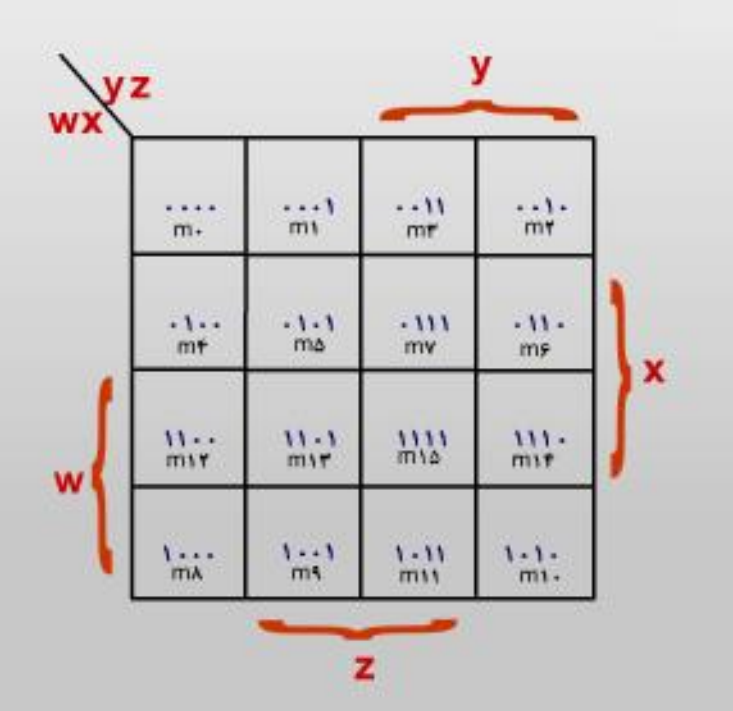
نکته : توابعی که مینترم‌های آن‌ها در جدول کارنو به‌صورت شطرنجی چیده می‌شوند، معمولاً از نوع توابع زوج یا فرد هستند و چیدمان شطرنجی آن‌ها با قوانین جبر بول ساده می‌شود و عبارت نهایی تابع برابر با XOR یا XNOR متغیرهای تابع خواهد بود.

تعریف تابع زوج : خروجی یک تابع زوج هنگامی 1 است که تعداد زوجی از متغیرهای ورودی مقدار 1 داشته باشند. تابع XNOR ۲ متغیره، ۳ متغیره و بالاتر، تابعی زوج است.

تعریف تابع فرد : خروجی یک تابع فرد هنگامی 1 است که تعداد فردی از متغیرهای ورودی مقدار 1 داشته باشند. تابع XOR ۲ متغیره، ۳ متغیره و بالاتر، تابعی فرد است.

نکته : هریک از توابع زوج و فرد، به ازای $2^n/2$ از مینترم‌ها برابر 1 خواهند شد. زیرا تعداد یک‌ها در نیمی از مینترم‌های تابع، زوج و در نیمی دیگر فرد می‌باشد.

جدول کارنوی ۴ متغیره : برای توابع ۴ متغیره به‌کارمی‌رود و تعداد خانه‌های آن برابر با ۱۶(تعداد مینترم های تابع ۴ متغیره) است. ساختار کارنوی ۴ متغیره در شکل مشاهده می‌شود.



در جدول کارنوی ۴ متغیره، خانه‌های ردیف های اول و آخر نظیر به نظیر با همجوار هستند. هم‌چنین، خانه‌های ستون اول و آخر نیز نظیر به نظیر با همجوار می‌باشند.

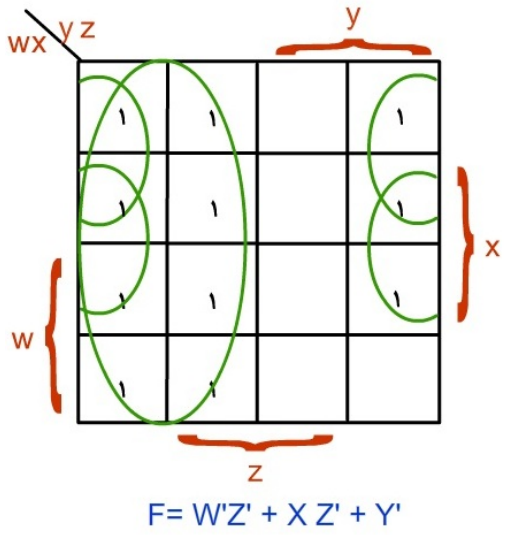
در کارنوی ۴ متغیره برخی همجواری ها به صورت زیرهستند :

m0,m2 : w'x'z'
m1,m9 : x'y'z
m12,m14 : wxz'
m8,m9, m10,m11 : wx'
m0,m4, m8,m12 : y'z'
m0,m2, m8,m10 : x'z'
m5,m7, m13,m15 : xz
m2,m3, m10,m11 : x'y
m4,m6, m12,m14 : xz'
m0,m1, m2,m3, m4,m5, m6,m7 : w'
m0,m1, m2,m3, m8,m9, m10,m11 : x'
m1,m3, m5,m7, m9,m11, m13,m15 : z
m0,m2, m4,m6, m8,m10, m12,m14 : z'

مثال ۱: تابع زیر را با استفاده از جدول کارنو ساده نمایید.

$$F(w,x,y,z) = \sum m(۰،۱،۲،۴،۵،۶،۸،۹،۱۲،۱۳،۱۴)$$

جدول کارنو و عبارت تابع به شکل زیر می باشند:

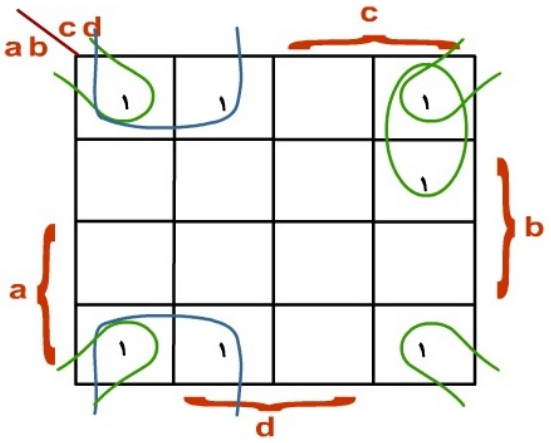


F= W'Z' + X Z' + Y'

مثال ۲: تابع زیر را ابتدا به فرم جمع مینترم‌ها درآورید و سپس با استفاده از جدول کارنو ساده نمایید.

$$F(A,B,C,D) = A'B'C'(D+D') + B'CD'(A+A') + A'BCD' + AB'C'(D + D')$$
$$= A'B'C'D + A'B'C'D' + AB'CD' + A'B'CD' + A'BCD' + AB'C'D + AB'C'D'$$
$$= m_1 + m_0 + m_{10} + m_2 + m_6 + m_9 + m_8 = \sum m(0,1,2,6,8,9,10)$$

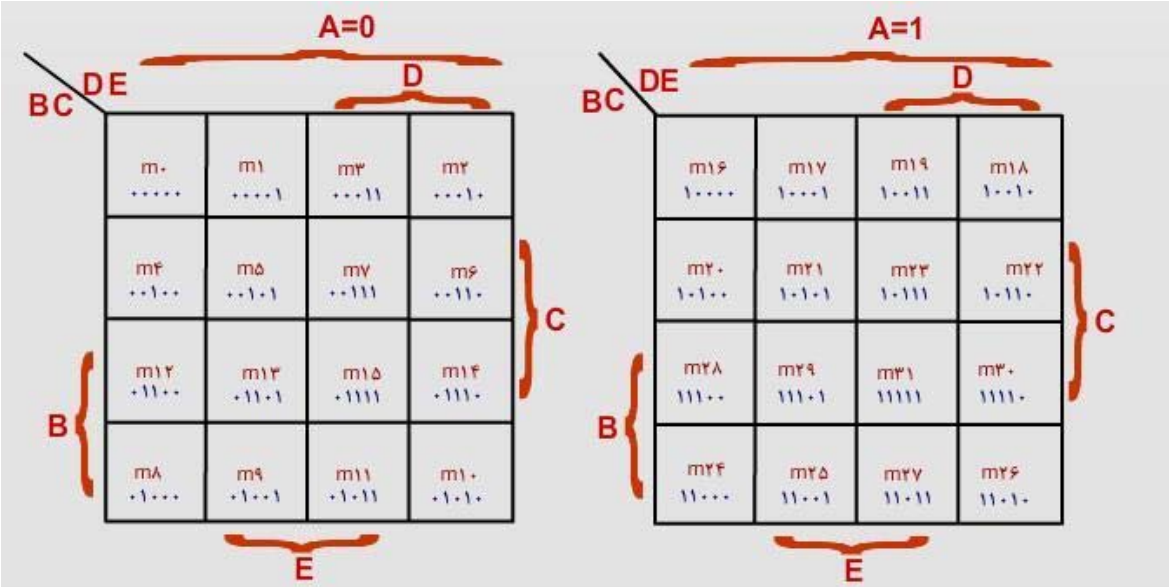
جدول کارنو و عبارت تابع F به شکل زیر می باشند:



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

جدول کارنوی ۵ متغیره : جدول کارنوی ۵ متغیره برای توابع ۵ متغیره به‌کارمی‌رود و تعداد خانه‌های آن برابر با ۳۲(تعداد مینترم‌های تابع ۵ متغیره) است. ساختار کارنوی ۵ متغیره در شکل مشاهده می‌شود. ترسیم کارنوی ۵ متغیره مستلزم سه بعد می باشد که ترسیم آن در صفحه ممکن نیست. برای این منظور از دو کارنوی ۴ متغیره در کنار هم استفاده می‌کنیم. در کارنوی سمت چپ، متغیر پر ارزش را صفر در نظر می‌گیریم و در کارنوی سمت راست متغیر پر ارزش را یک در نظر می گیریم. همانطور که در شکل ملاحظه می‌شود شماره مینترم‌های صفر تا ۱۵ در کارنوی سمت چپ و شماره مینترم‌های ۱۶ تا ۳۱ در کارنوی سمت راست قرار می گیرند.

در دو کارنوی ترسیم شده، خانه‌های دو جدول نظیر به نظیر باهم متناظرند.



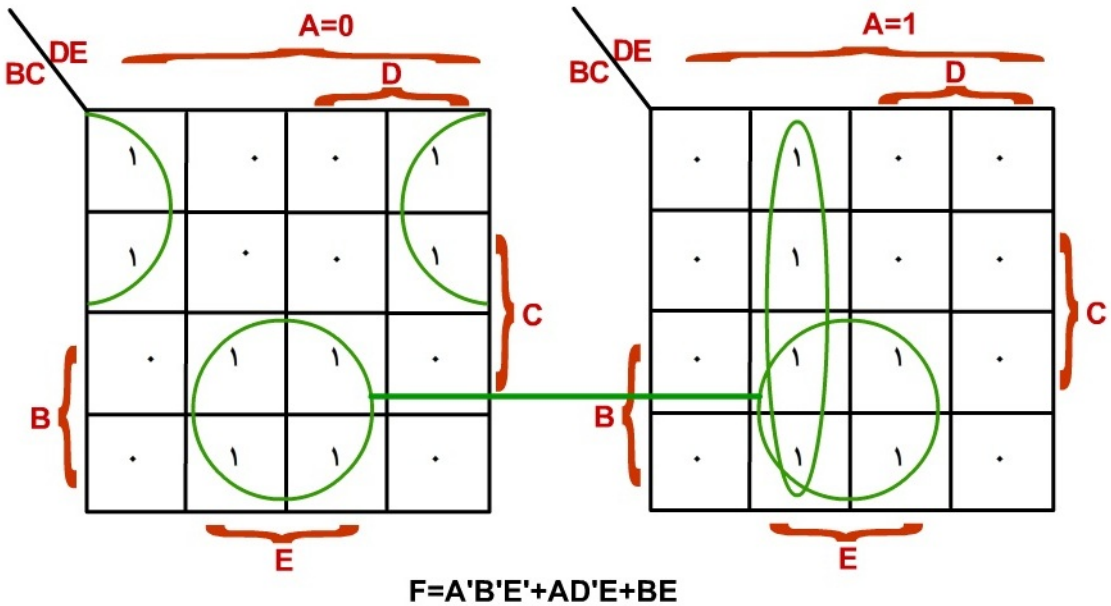
تعریف جدید همجواری در جدول کارنوی ۵ متغیره : در کارنوی ۵ متغیره، علاوه بر تعاریف همجواری در هر جدول که مشابه کارنوی ۴ متغیره است، خانه‌های متناظر در دو جدول نیز با هم همجوارند و تنها در متغیر اول تفاوت دارند. لذا پوششی که مینترم‌های آن در دو جدول قرار می‌گیرد، متغیر اول در جمله متناظر با آن ظاهر نخواهد شد.

در کارنوی ۵ متغیره برخی از همجواری‌های جدید به صورت زیرهستند :

m0,m16 : B'C'D'E'
m7,m23 : B'CDE
m8,m9, m24,m25 : BC'D'
m6,m14, m22,m30 : CDE'
m5,m7, m21,m23 : B'CE
m0,m1, m2,m3, m16,m17, m18,m19 : B'C'
m10,m11, m14,m15, m26,m27, m30,m31 : BD
m5,m7, m13,m15, m21,m23, m29,m31 : CE
m0,m2, m8,m10, m16,m18, m24,m26 : C'E'
m0,m1, m4,m5, m8,m9, m12,m13, m16,m17, m20,m21, m24,m25, m28,m29 : D'
m1,m3, m5,m7, m9,m11, m13,m15, m17,m19, m21,m23, m25,m27, m29,m31 : E
m4,m5, m6,m7, m12,m13, m14,m15, m20,m21, m22,m23, m28,m29, m30,m31 : C

مثال ۱: تابع زیر را با استفاده از جدول کارنو ساده نمایید.

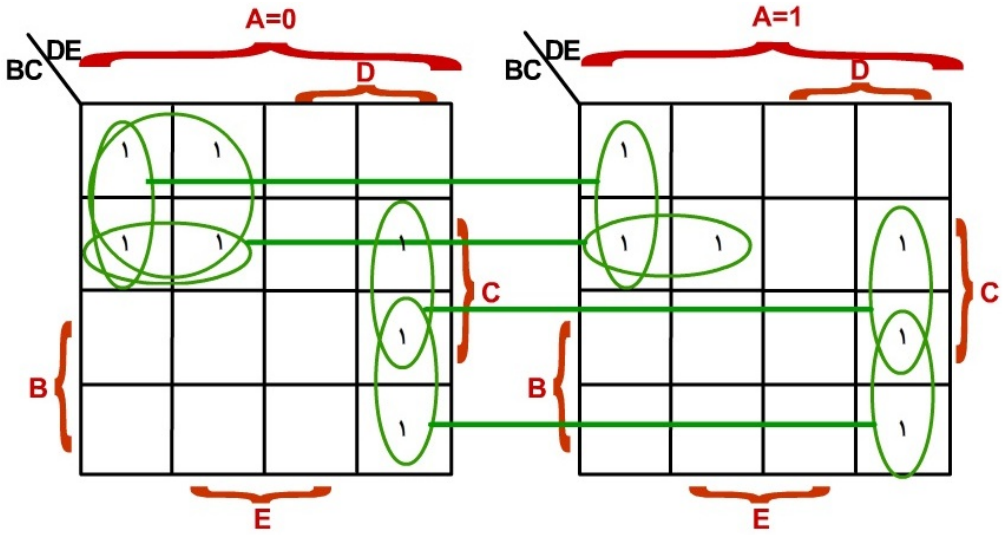
$$F(A,B,C,D,E) = \sum m(۰,۲,۴,۶,۹,۱۱,۱۳,۱۵,۱۷,۲۱,۲۵,۲۷,۲۹,۳۱)$$



F=A'B'E'+AD'E+BE

مثال ۲: تابع زیر را با استفاده از جدول کارنو ساده نمایید.

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



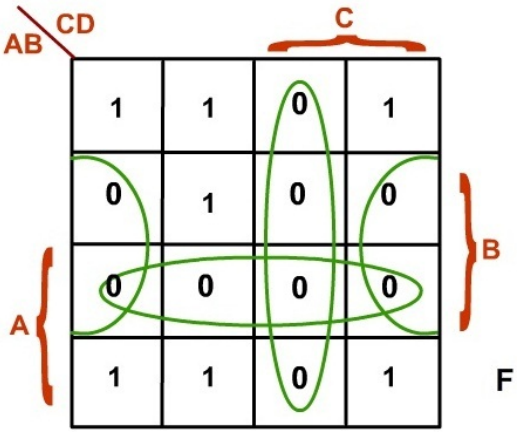
$$F=A'B'D'+B'CD'+B'D'E'+CDE'+BDE'$$

ارتباط میان تعداد خانه‌های پوشش و تعداد لیترال‌های جمله حاصل ضرب متناظر با پوشش : 2^k مربع همجوار به ازاء $k = (0,1,2,\ldots,n)$ در یک نقشه n متغیره ناحیه‌ای را مشخص می‌کند که نمایش دهنده جمله‌ای با $n - k$ لیترال است. اگر در جدولی، تمام خانه‌ها مقدار 1 داشته باشند، به این معنی است که تابع به ازای تمام مینترم‌ها خروجی 1 تولید می‌کند که همان تعریف تابع ثابت 1 است.

ساده‌سازی با ضرب حاصل جمع‌ها : برای این منظور از خواص جبر بول کمک می‌گیریم. یک های واقع در مربع های جدول کارنو، بیانگر مینترم‌های تابع است. اما مینترم‌هایی که در تابع ذکر نشوند، بیانگر متمم تابع هستند. لذا از این خاصیت استفاده کرده و پوشش هایی از صفرهای جدول تشکیل می‌دهیم. عبارت ساده شده متمم تابع(F') به صورت جمع حاصل ضرب‌ها تولید خواهد شد. با یک مرتبه متمم‌گیری از 'F، ساده‌ترین حالت تابع F به فرم ضرب حاصل جمع‌ها به دست خواهد آمد.

مثال : تابع زیر را با استفاده از جول کارنو به فرم ضرب حاصل جمع‌ها ساده نمایید.

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



$F' = CD + AB + BD'$

متمم

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

حالت های بی‌اهمیت در جدول کارنو، گیت های یونیورسال

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- آشنایی با مفهوم حالت‌های بی اهمیت در توابع جبر بول
- کاربرد حالت های بی اهمیت در ساده‌سازی بیشتر توابع
- آشنایی با گیت‌های یونیورسال و خواص گیت های NAND و NOR
- نحوه پیاده‌سازی توابع با گیت‌های تماماً NAND و NOR

حالتهای بی‌اهمیت : در یکی از شرایط زیر می‌گوییم که در تابع حالت بی‌اهمیت وجود دارد:

- به ازای برخی حالت‌ها ممکن است 0 یا 1 بودن تابع اهمیتی نداشته باشد.
- بعضی از حالت‌های ورودی اصلاً در تابع اتفاق نمی‌افتند.

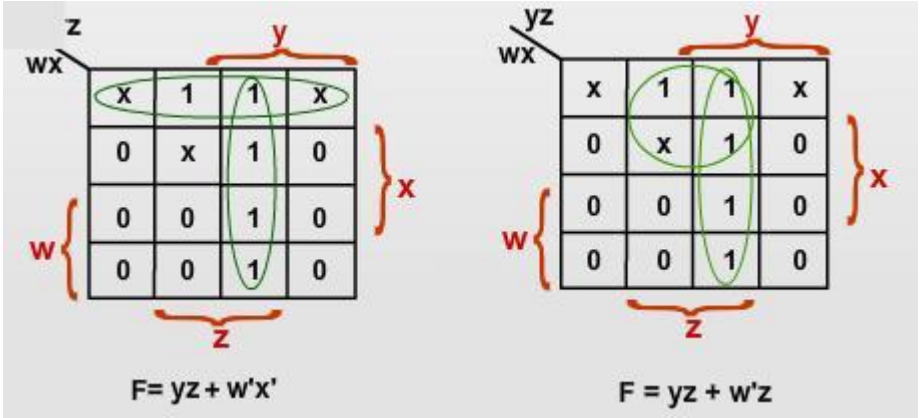
در هر ۲ حالت می‌گوئیم که در تابع حالت بی‌اهمیت (don't care) وجود دارد. در جدول درستی یا جدول کارنو، خروجی تابع را با X نمایش می‌دهیم.

کاربرد اصلی حالت‌های بی‌اهمیت در جدول کارنو، در ساده‌سازی بیشتر توابع می‌باشد. با توجه به ساختار جدول می توان این خانه‌ها را 0 یا 1 در نظر گرفته و سپس تابع را ساده نمود. حالت‌های بی‌اهمیت را با d و شماره مینترم‌های بی‌اهمیت در عبارت جبری تابع نشان می‌دهیم.

مثال ۱ : تابع زیر را با استفاده از حالت‌های بی‌اهمیت ساده نمائید.

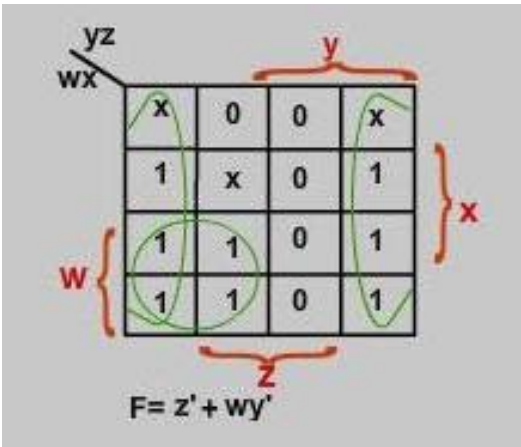
$$F(w,x,y,z) = \sum m(۱،۳،۷،۱۱،۱۵) + d\,(۰،۲،۵)$$

حل : بسته به اینکه کدام یک از مینترم‌های بی‌اهمیت را 0 و کدام یک را 1 در نظر بگیریم، دو عبارت ساده شده به صورت زیر می‌توان برای تابع به‌دست‌آورد :



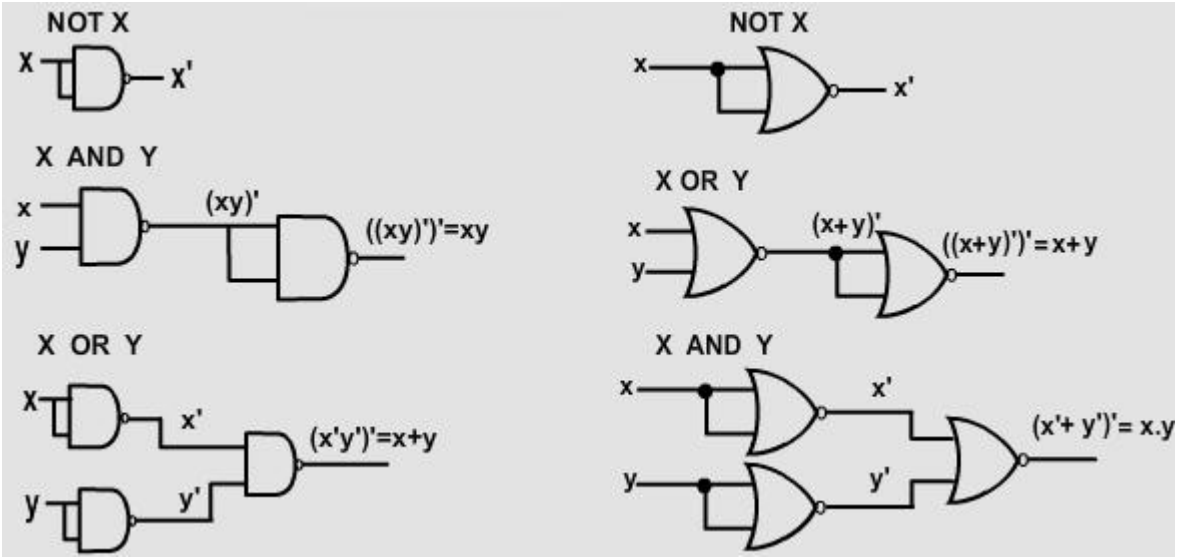
مثال ۲ : تابع زیر را با استفاده از حالت‌های بی‌اهمیت ساده نمایید.

$$F(w,x,y,z) = \sum m(۴،۶،۸،۹،۱۰،۱۲،۱۳،۱۴) + d\ (۰،۲،۵)$$



پیاده‌سازی مدارات منطقی با گیت‌های NAND و NOR : مدارهای دیجیتال اغلب به جای AND و OR با گیت‌های NAND و NOR ساخته می‌شوند. زیرا ساخت این‌گونه گیت‌ها با اجزاء الکترونیکی ساده‌تر بوده به‌عنوان گیت‌های پایه در تمام خانواده‌های IC‌های دیجیتال به کار می‌روند. به دلایل فوق قواعد و روال‌هایی برای تبدیل توابع منطقی بیان شده بر حسب گیت‌های منطقی AND، OR و NOT به نمودارهای منطقی معادل بر حسب گیت های NAND و NOR بیان شده است.

گیت‌های یونیورسال : گیت‌های NAND و NOR را به‌عنوان گیت‌های یونیورسال در تمام خانواده‌های منطقی می‌نامیم. زیرا هر سیستم دیجیتالی را می‌توان تماماً با گیتهای NAND و یا تماماً با گیت‌های NOR پیاده‌سازی نمود. برای اثبات این مطلب کافیست نشان دهیم که عملیات منطقی پایه شامل AND، OR و NOT را می‌توان با گیت‌های NAND یا NOR پیاده‌سازی نمود. این مطلب در شکل زیر برای گیت NAND و NOR اثبات شده است :



پیاده سازی توابع با گیت‌های تمام NAND : برای این منظور مراحل زیر را طی می‌کنیم :

۱. در جدول کارنوی تابع، 1 ها را پوشش می‌دهیم تا فرم AND-OR تابع حاصل شود.

۲. عبارت حاصل را ۲ بار NOT می‌کنیم.

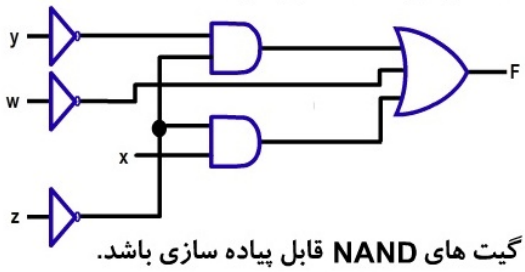
۳. در انتها یکی از NOT ها را به تابع اِعمال می‌کنیم؛ عبارت حاصل قابل پیاده‌سازی با گیت‌های NAND خواهد بود.

مثال : تابع زیر را با گیت‌های NAND پیاده‌سازی کنید.

مثال : تابع زیر را با گیت های NAND پیاده سازی کنید.

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

نمودار مداری تابع F با استفاده از گیت های AND ، OR و NOT در شکل زیر مشاهده می شود.



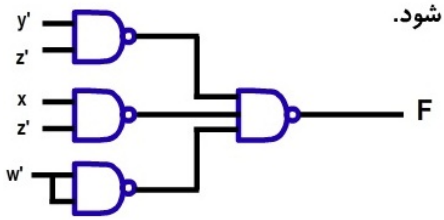
حال با دست کاری های جبری، تابع را به فرمی تبدیل می‌کنیم که با گیت های NAND قابل پیاده سازی باشد.

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

F = ((w'+y'z'+xz')')'

F = (w.(y'z')'.(xz')')'

نمودار مداری تابع F با استفاده از گیت‌های NAND در شکل زیر مشاهده می شود.



پیاده سازی توابع با گیت‌های تمام NOR : برای این منظور مراحل زیر را طی می‌کنیم :

۱. ابتدا باید صفرهای جدول را پوشش دهیم تا فرم AND-OR برای متمم تابع حاصل شود.

۲. عبارت حاصل را یک بار NOT می‌کنیم تا فرم OR-AND برای تابع اصلی حاصل شود.

۳. عبارت حاصل را ۲ بار NOT می‌کنیم.

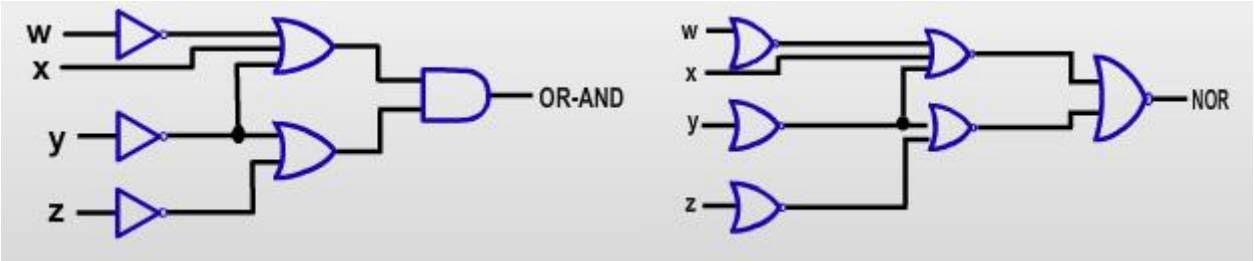
۴. در انتها یکی از NOT ها را به تابع اِعمال می‌کنیم؛ عبارت حاصل قابل پیاده‌سازی با گیت‌های NOR خواهد بود.

مثال : تابع زیر را با گیت‌های NOR پیاده‌سازی کنید.

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

حل : ابتدا متمم تابع را با پوشش صفرهای تابع در جدول درستی به‌دست می‌آوریم.

1 : F' = yz+wx'y
2 : F = (y'+z')(w'+x+y')
3 : F = (((y'+z').(w'+x+y'))') '
4 : F = ((y'+z')' + (w'+x+y')')'



فصل چهارم

مدارهای منطقی ترکیبی

تحلیل و طراحی مدارهای منطقی ترکیبی و پیاده‌سازی مدارهای محاسباتی و مبدل‌های کد

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

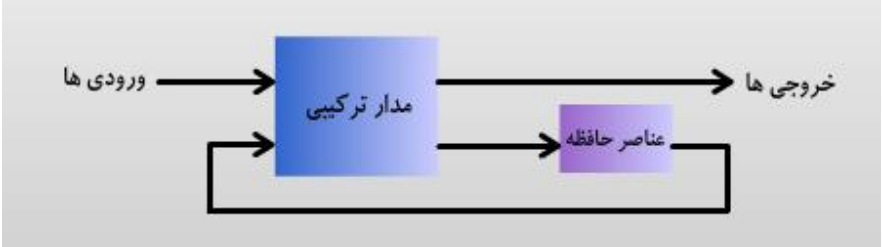
- شناخت انواع مدارهای منطقی و تعریف مدارهای منطقی ترکیبی و ترتیبی
- شناخت تفاوت‌های میان مدارهای ترکیبی و مدارهای ترتیبی و ویژگی‌های هر کدام
- نحوه تحلیل و آنالیز مدارهای منطقی ترکیبی و نمودارهای مداری متشکل از گیت‌ها
- مراحل طراحی مدارهای منطقی ترکیبی
- ساخت مدارهای محاسباتی جمع کننده، تفریق‌گر، جمع‌کننده BCD و ضرب‌کننده

انواع مدارهای منطقی : مدارهای منطقی در سیستم‌های دیجیتال می‌توانند از نوع ترکیبی یا ترتیبی باشند.

مدارهای منطقی ترکیبی : یک مدار ترکیبی متشکل از تعدادی گیت منطقی است که خروجی آن‌ها در هر لحظه از زمان مستقیماً به ورودی‌های همان لحظه بستگی داشته و به ورودی‌های قبلی بستگی ندارد. این نوع مدار پردازشی را انجام می‌دهد که با مجموعه‌ای از توابع بولی مشخص می‌گردد.

یک مدار ترکیبی می‌تواند دارای n متغیر ورودی و m متغیر خروجی باشد. برای هر ترکیب ممکن از ورودی ها، فقط یک مقدار برای هر خروجی موجود است. بنابراین، یک مدار ترکیبی با یک جدول درستی که مقادیر خروجی‌ها را برای هر ترکیب از متغیرهای ورودی لیست می‌نماید، نشان داده می‌شود. هر یک از خروجی‌های یک مدار ترکیبی را می‌توان توسط یک عبارت جبری نیز بیان نمود.

مدارهای منطقی ترتیبی : مدارهای ترتیبی علاوه بر گیت‌های منطقی، از عناصر حافظه نیز استفاده می‌کنند. خروجی‌های این مدارها در هر لحظه از زمان، تابعی از ورودی ها و حالت عناصر حافظه در آن لحظه است. لذا عملکرد مدار باید به وسیله حالات داخلی و ترتیب زمانی ورودی‌ها مشخص‌گردد.



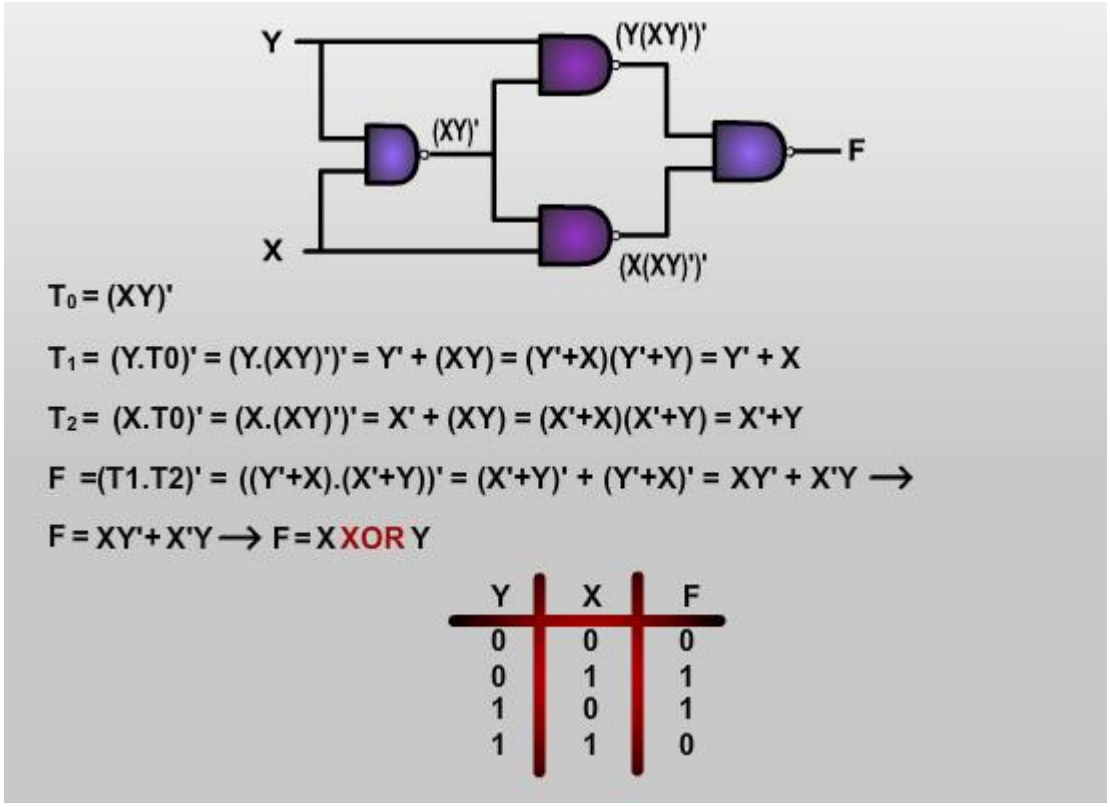
در این فصل به بررسی مدارهای ترکیبی، ویژگی ها و نحوه تحلیل و طراحی آن‌ها می‌پردازیم. مدارهای منطقی ترتیبی، مبحث فصل بعدی این درس خواهد بود.

تحلیل مدارهای ترکیبی : تحلیل یا آنالیز یک مدار ترکیبی با یک نمودار منطقی آغاز شده و با مجموعه‌ای از توابع بول، یک جدول درستی یا توضیحاتی از عملکرد مدار پایان می‌یابد.

مراحل تحلیل یک مدار ترکیبی و به دست آوردن توابع بولی خروجی :

۱. تعیین تابع خروجی گیت‌هایی که تابعی از ورودی ها هستند و نام گذاری آن‌ها.
۲. تعیین تابع خروجی گیت‌هایی که تابعی از متغیرهای ورودی و گیت‌های برچسب خورده قبلی هستند و نام گذاری آن‌ها.
۳. تکرار مرحله ۲ تا دستیابی به خروجی‌های مدار.
۴. به دست آوردن توابع بولی خروجی نهایی بر حسب متغیرهای ورودی اولیه با جایگزینی توابع به‌دست‌آمده در مراحل قبل.
۵. رسم جدول درستی برای خروجی‌های مدار.
۶. شرح عملکرد مدار با توجه به ارتباط میان مقادیر خروجی‌ها با ورودی ها در هر حالت.

مثال : جدول درستی مدار زیر را رسم نموده و عملکرد آن را آنالیز نمایید.



Y	X	F
0	0	0
0	1	1
1	0	1
1	1	0

طبق جدول ملاحظه می‌شود که عملکرد این مدار معادل گیت XOR دو متغیره است.

طراحی مدارهای ترکیبی : طراحی مدارهای ترکیبی با مشخصات مسئله آغاز و به فرم نمودار مدار منطقی یا مجموعه‌ای از توابع جبر بول برای خروجی‌ها پایان می‌یابد.

مراحل طراحی یک مدار ترکیبی به روش کلاسیک :

۱. تعیین تعداد ورودی ها و خروجی‌ها با استفاده از مشخصات مدار و تخصیص سمبل‌های حرفی به آن‌ها
۲. تشکیل جدول درستی مربوط به ورودی ها و خروجی‌های مدار
۳. به دست آوردن توابع جبری ساده شده برای هر خروجی به صورت تابعی از متغیرهای ورودی
۴. ترسیم نمودار منطقی مدار در صورت لزوم

روش های کلاسیک و غیر کلاسیک در طراحی مدارهای منطقی ترکیبی : در طراحی مدارهای ترکیبی به روش کلاسیک، با افزایش متغیرهای ورودی، ترسیم جدول درستی و جداول کارنو برای ساده‌سازی توابع خروجی به مراتب مشکل تر شده و ضریب اشتباه طراح را بالا خواهد برد.

اما توابع دیجیتالی نیز وجود دارند که ذاتاً دارای نظم درونی هستند و معمولاً با روال های الگوریتمی و روش های غیر کلاسیک و ساده‌تر مانند استفاده از توابع طراحی شده قبلی، قابل طراحی‌اند. در روش غیر کلاسیک، هر مسئله‌ای روش خاص خودش را دارد و از روال منظمی پیروی نمی‌کند.

در ادامهٔ درس، طراحی مدارهای مختلف ترکیبی به خصوص مدارهای محاسباتی- منطقی مورد استفاده در ALU و مدارات مبدل کد را مورد بررسی قرار می‌دهیم.

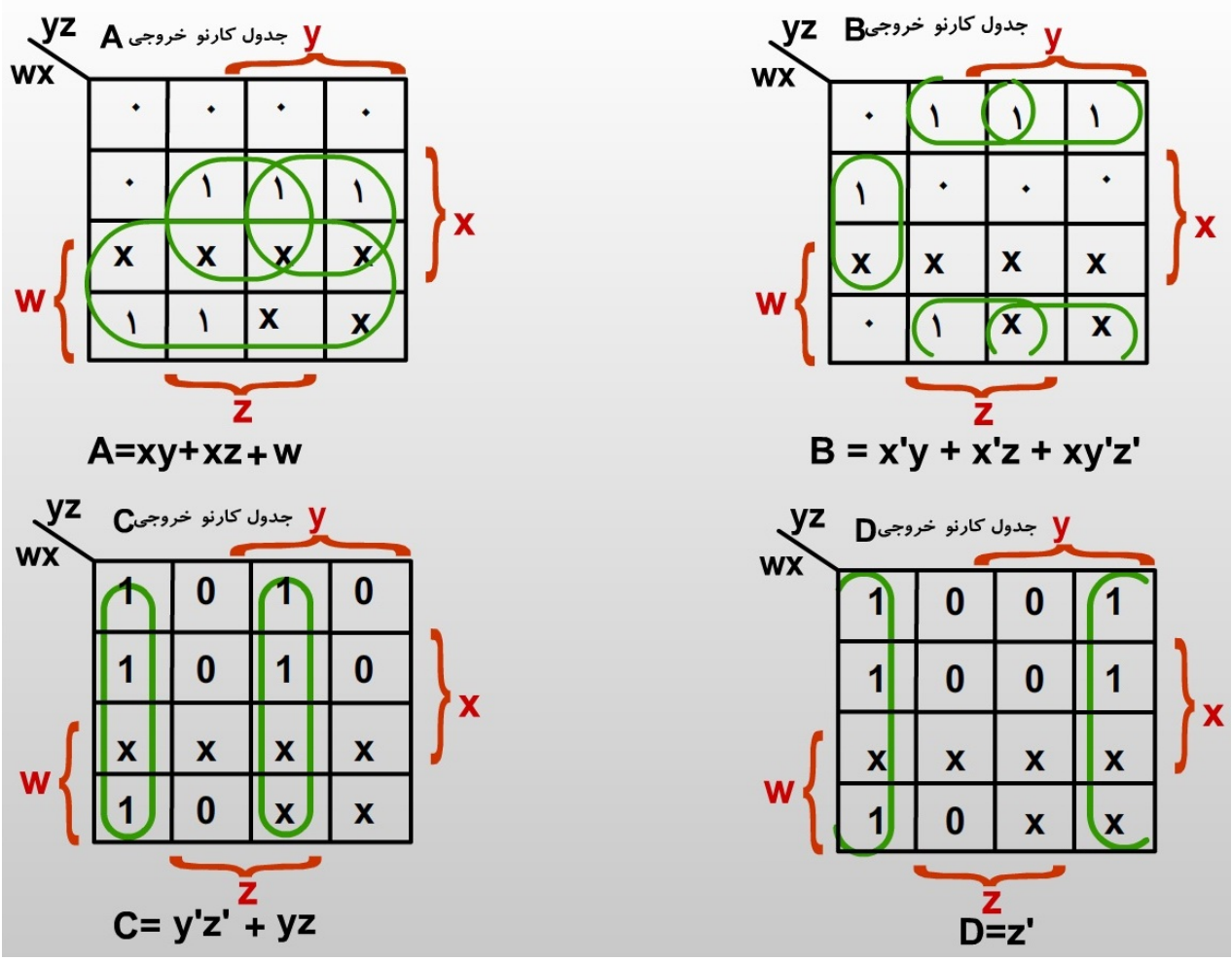
مدار مبدل کد BCD به افزونی ۳ (Excess-3) : سیستم‌های دیجیتال مختلف از کدهای متفاوتی برای ذخیره-سازی اطلاعات استفاده می‌کنند. برای برقراری ارتباط میان دو سیستم که از کدهای مختلفی برای بیان اطلاعات یکسان استفاده می‌کنند، یک مدار مبدل کد باید بین آن دو قرار داده شود.

بنابراین یک مدار مبدل کد، مداری است که دو سیستم را علیرغم به کارگیری کدهای دودویی متفاوت، با هم سازگار می‌سازد.

مثال : یک مدار مبدل کد BCD به افزونی ۳ طراحی نمایید.

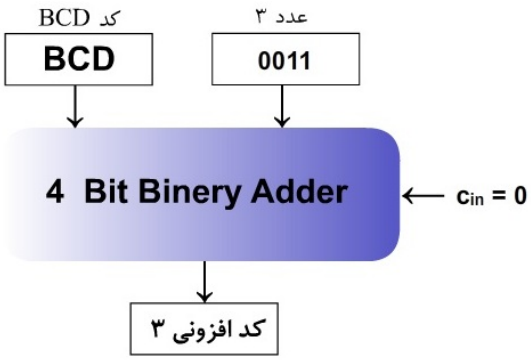
روش کلاسیک : در این روش، طبق مراحل ذکر شده در صفحات قبلی عمل می‌کنیم.

W	X	Y	Z	A	B	C	D
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



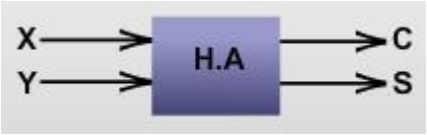
روش غیر کلاسیک : با توجه به عملکرد کد افزونی ۳ می توان نتیجه گرفت که اگر بتوانیم به هر کد ورودی BCD، ۳ واحد اضافه کنیم به مطلوب مسئله خواهیم رسید.

برای این منظور توسط یک جمع‌کننده چهار بیتی که در مثال های آتی طراحی خواهیم نمود، عدد ورودی را با یک عدد ثابت ۳ معادل باینری 0011 جمع کرده و رقم نقلی ورودی را صفر در نظر می گیریم.



توجه ۱ : همانطور که ملاحظه می شود روش غیر کلاسیک ما را مستقیماً به مدار می رساند ولی برای هر صورت مسئله، روش خاص خود را دارد. در حالی که روش کلاسیک برای تمام مسائل راه حل یکسانی دارد و مراحل آن مشخص می باشد.

جمع کننده بیتی یا نیم جمع کننده (Half Adder) : نیم جمع کننده مداریست که ۲ بیت را با یکدیگر جمع می‌کند. حاصل جمع ۲ بیت، عددی ۲ بیتی خواهد بود. برای طراحی مدار نیم جمع کننده، مراحل طراحی را دنبال می کنیم. به این ترتیب که شکل بلوکی، جدول درستی، جدول کارنو و در نهایت روابط منطقی را به دست می آوریم.



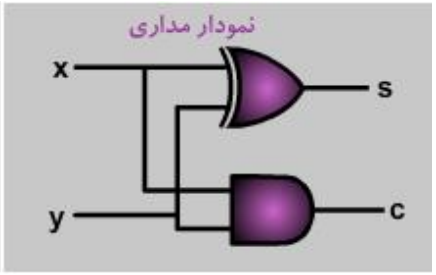
در این مدار، x و y ورودی‌های مدار و S و C خروجی‌های مدار هستند. S به معنی Sum یا حاصل جمع ۲ بیت و C به معنی Carry یا رقم نقلی خروجی است. جدول درستی مدار به صورت زیر است :

X	Y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

معادلات خروجی

$C=X.Y$

$S=X \oplus Y$



تمام جمع کننده (Full Adder) : در جمع دو عدد n بیتی به علت وجود رقم نقلی به جمع کننده‌ای که قابلیت جمع سه بیت با هم را داشته باشد، احتیاج داریم. پس طراحی جمع کننده سه بیت مورد نیاز است. جمع کننده-ای که قابلیت جمع سه بیت با هم را داشته باشد، تمام جمع کننده (Full Adder) نامیده می‌شود. حاصل جمع ۳ بیت، عددی ۲ بیتی خواهد بود.

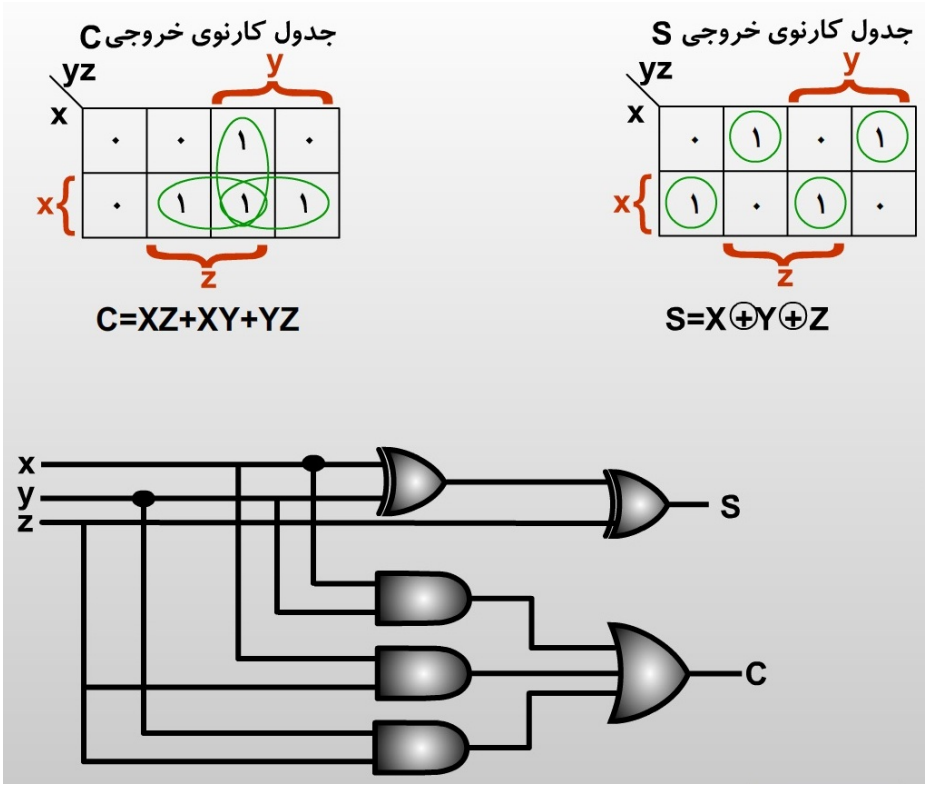
برای طراحی مدار تمام جمع کننده، مراحل طراحی را دنبال می‌کنیم. به این ترتیب که شکل بلوکی، جدول درستی، جدول کارنو و در نهایت روابط منطقی را به دست می آوریم.

در این مدار، x، y و z، ورودی‌های مدار و S و C خروجی‌های مدار هستند. S به معنی Sum یا حاصل جمع ۳ بیت و C به معنی Carry یا رقم نقلی خروجی است. جدول درستی مدار به صورت زیر است :

X	Y	Z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

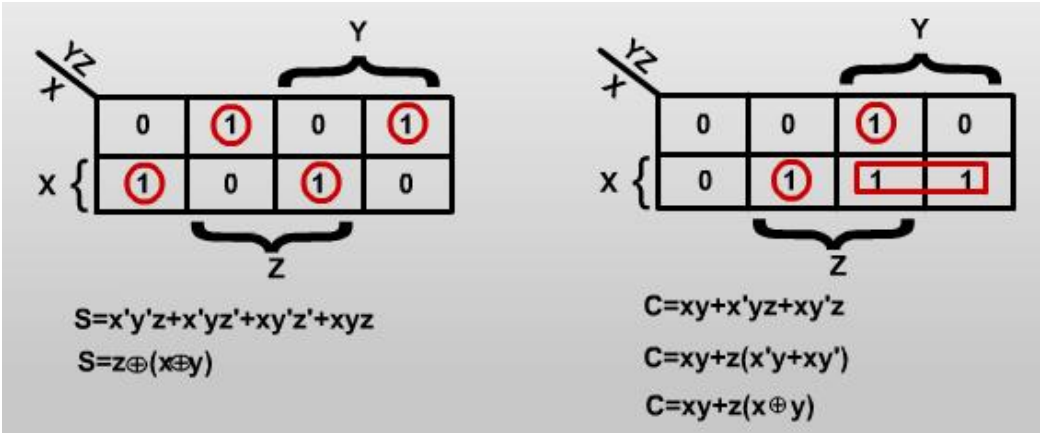
$C = \sum m(3,5,6,7) = X'YZ + XY'Z + XYZ' + XYZ = XZ + XY + YZ$
$S = \sum m(1,2,4,7) = X'Y'Z + X'YZ' + XY'Z' + XYZ = X \text{ XOR } Y \text{ XOR } Z$

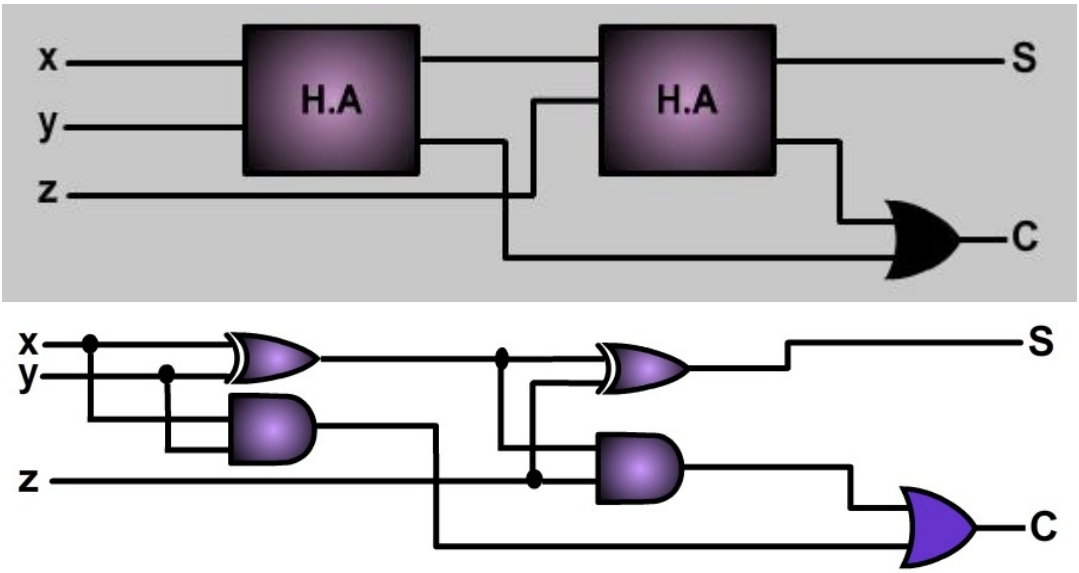
جدول کارنوی خروجی‌های توابع تمام جمع کننده به همراه نمودار منطقی آنها در زیر رسم شده‌اند :



پیاده‌سازی تمام جمع کننده با استفاده از نیم جمع کننده :

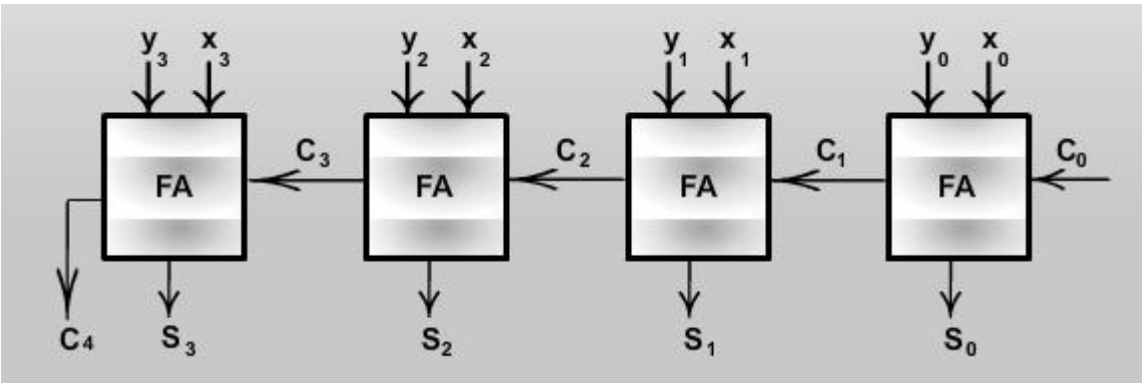
با استفاده از دو نیم جمع‌کننده و یک گیت OR می‌توان یک تمام جمع‌کننده ساخت. این عمل با یک دست کاری ساده در پوشش های جدول درستی به شکل زیر امکان‌پذیر خواهد بود :





جمع کننده دودویی n بیتی : در طراحی جمع کننده‌های n بیتی به روش کلاسیک، با افزایش تعداد بیت های اعداد، تعداد سطرهای جدول های درستی به شدت افزایش یافته و به دست آوردن توابع ساده شده خروجی به راحتی امکان پذیر نخواهد بود. به عنوان مثال، برای طراحی یک جمع کننده چهار بیتی به یک جدول درستی با $۲^{۹}=۵۱۲$ سطر نیاز خواهیم داشت. لذا به روش غیر کلاسیک عمل خواهیم کرد.

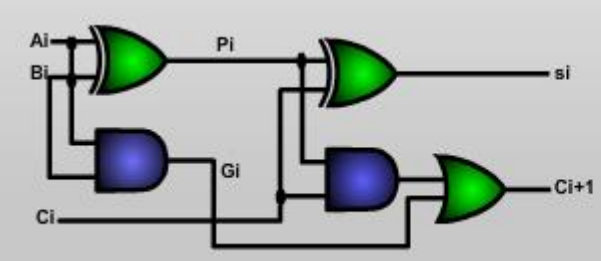
در این روش، برای جمع اعداد چند بیتی نیاز به جمع‌کننده های بزرگتر داریم. پس با استفاده از n عدد تمام جمع کننده، یک جمع‌کننده n بیتی می سازیم. به عنوان مثال، در یک جمع کننده چهار بیتی، چهار عدد تمام جمع کننده در کنار هم قرار می‌گیرند. لذا این مدار یک مدار تکرار پذیر است. در این جمع کننده، رقم نقلی به صورت موج گونه از تمام جمع کننده ها منتشر شده و عبور می کند. به این نوع جمع کننده، جمع کننده با رقم نقلی موج گونه گفته می شود.



این نوع جمع کننده را جمع کننده با رقم نقلی موج گونه می‌نامیم. اشکال عمدهٔ این روش طراحی جمع کننده، زمان انتشار طولانی مورد نیاز برای تولید رقم نقلی نهایی از بیت پر ارزش اعداد می‌باشد، زیرا برای تولید خروجی‌های نهایی در هر جمع کننده، ۲ طبقه گیت تاخیر وجود خواهد داشت و خروجی هر طبقه نیز به آماده

بودن ورودی طبقه ماقبل خود وابسته می‌باشد. لذا به عنوان مثال برای یک جمع کننده ۴ بیتی، ۸ طبقه گیت تاخیر وجود خواهد داشت. این تاخیر برای یک جمع کننده ۸ بیتی، برابر ۱۶ خواهد بود و مشاهده می‌شود که با افزایش تعداد بیت های اعداد، زمان لازم برای تولید خروجی نهایی به شدت افزایش خواهد یافت.

طراحی جمع کننده n بیتی با پیش بینی رقم نقلی : برای حل مشکل جمع کننده با رقم نقلی موج‌گونه، باید وابستگی نقلی خروجی هر طبقه به طبقات قبلی حذف شود، لذا جمع‌کننده جدیدی طراحی شده است که در آن، رقم نقلی ورودی هر جمع‌کننده بیتی، به رقم نقلی طبقه قبل وابسته نبوده و زودتر آماده می‌شود. در این حالت همه رقم‌های نقلی بطور همزمان آماده خواهند شد. به این ترتیب این نوع جمع کننده، تاخیر رقم های نقلی جمع‌کننده موج گونه را برای آماده شدن جواب نخواهد داشت.
برای این منظور با استفاده از عملیات جبری روابط جدیدی برای محاسبه رقم نقلی هر طبقه به دست آورده‌اند و برای طرح مدار مربوطه از تمام جمع کننده‌ها ایده گرفته شده است.



$P_i=A_i \text{ xor } B_i$
$G_i=A_i.B_i$
$S_i=A_i \text{ xor } B_i \text{ xor } C_i$
$C_{i+1}=A_i.B_i+C_i(A_i \text{ xor } B_i)=P_iC_i+G_i$

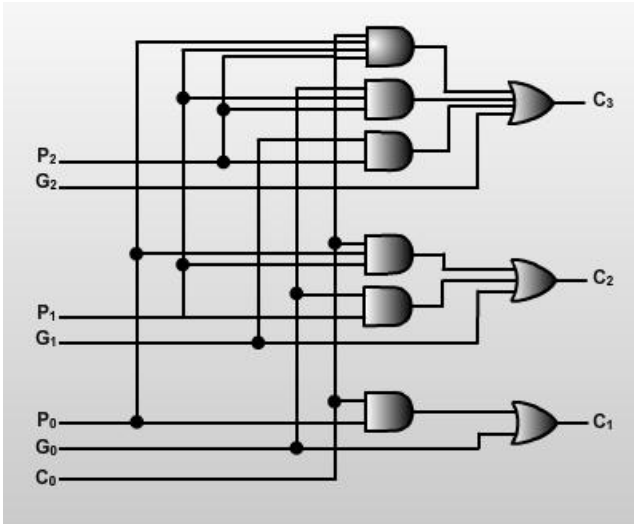
در مدارهای فوق، وابستگی رقم نقلی دیده می شود. اما با استفاده از دست کاری های جبری همان طور که در روابط زیر مشاهده می‌شود، رقم نقلی بیت های بعدی فقط به رقم نقلی اولی بستگی داشته و به سایر رقم‌های نقلی ارتباطی ندارد.

$S_i=P_i \oplus C_i$ $S_1=P_1 \oplus C_1$ $S_2=P_2 \oplus C_2$ $S_3=P_3 \oplus C_3$

$C_{i+1}=PC_i+G_i$ $C_2=P_1C_1+G_1$ $C_3=P_2C_2+G_2$ $C_4=P_3C_3+G_3$

$C_3=P_2P_1C_1+P_2G_1+G_2$ $C_4=P_3P_2P_1C_1+P_3P_2G_1+P_3G_2+G_3$

در شکل زیر مدار تولید رقم‌های نقلی برای یک جمع‌کننده ۴ بیتی رسم شده است. همان گونه که مشاهده می-شود، تاخیر انتشار لازم برای تولید رقم نقلی خروجی در تمام طبقات جمع‌کننده با هم یکسان بوده و برابر ۴ طبقه گیت است.



تفریق کننده بیتی یا نیم تفریق کننده (Half Subtractor) : برای تفریق دو عدد چند بیتی لازم است ابتدا تفریق دو بیت را انجام دهیم، لذا در این مرحله می‌خواهیم مدار تفریق دو بیت یا نیم تفریق کننده را طراحی کنیم.

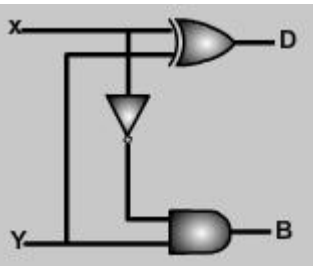
نیم تفریق کننده مداریست که ۲ بیت را از هم کم می‌کند. حاصل تفریق، عددی ۲ بیتی خواهد بود. این مدار عبارت زیر را محاسبه می کند که ممکن است یک بیت از بیت سمت چپ قرض گرفته شود : **X-Y** برای طراحی مدار نیم تفریق کننده، مراحل طراحی را دنبال می کنیم. به این ترتیب که شکل بلوکی، جدول درستی، جدول کارنو و در نهایت روابط منطقی را به دست می آوریم.

در این مدار، x و y ورودی‌های مدار و D و B خروجی‌های مدار هستند. D به معنی Difference یا حاصل تفریق ۲ بیت و B به معنی Borrow یا رقم قرض خروجی است. جدول درستی، معادلات خروجی‌ها و نمودار مداری نیم تفریق کننده به صورت زیر است :

X	Y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

D = X xor Y

B = X' Y



تمام تفریق کننده (Full subtractor) : در تفریق دو عدد n بیتی به علت وجود رقم قرض به تفریق کننده-ای نیاز است که قابلیت تفریق ۲ بیت همراه با رقم قرض احتمالی را از طبقه قبل داشته باشد. تفریق کننده‌ای که قابلیت تفریق ۳ بیت از هم را داشته باشد ، تمام تفریق کننده (Full Subtractor) نامیده می‌شود. در تمام تفریق کننده عبارت زیر محاسبه می شود :

X-Y-Z یا X-(Y+Z)

جدول درستی، جداول کارنو و در نهایت روابط منطقی خروجی‌های تمام تفریق کننده به شرح زیر به دست می‌آیند:

X	Y	Z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

جدول کارنوی خروجی D

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

$D=X\oplus Y\oplus Z$

جدول کارنوی خروجی B

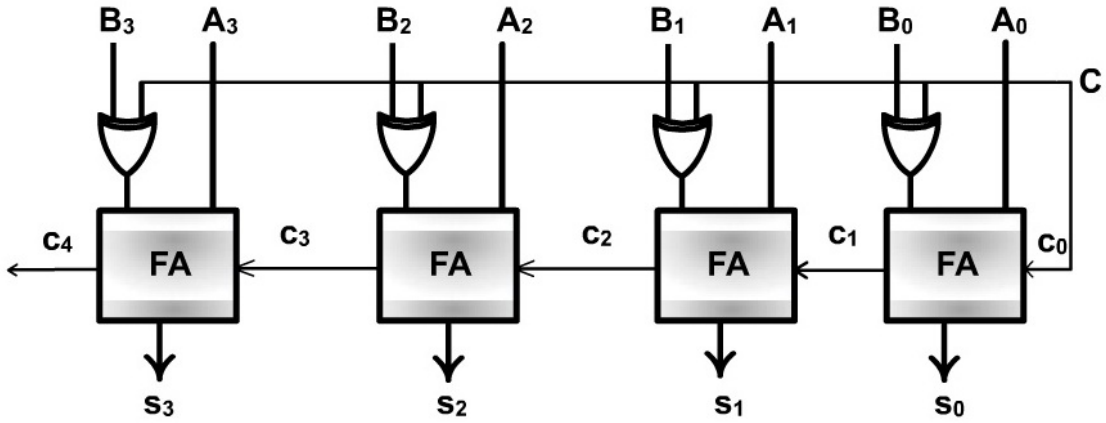
x \ yz	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$B=YX'+ZX'+YZ$

جمع و تفریق کننده n بیتی : برای تفریق اعداد n بیتی به دو روش زیر می‌توان عمل نمود :

۱. با استفاده از تمام تفریق کننده ها تفریق کننده n بیتی طراحی می‌کنیم. در این روش، مراحل کار مشابه مراحل طراحی جمع‌کننده های n بیتی با استفاده از تمام جمع‌کننده ها می‌باشد.

۲. در روش دوم با استفاده از جمع‌کننده ها و انجام عمل تفریق به روش متمم ۲، می‌توان مدارات جمع‌کننده تفریق کننده n بیتی ساخت. در این مدار با یک ورودی کنترل می‌توان تعیین نمود که دو عدد n بیتی با هم جمع شوند و یا از هم تفریق گردند.



همان گونه که در دیاگرام بلوکی فوق مشاهده می‌شود، بیت‌های عدد دوم (B) ابتدا وارد گیت های XOR شده و با یک ورودی کنترل یا فرمان XOR می‌شوند که این ورودی کنترل به عنوان نقلی ورودی، وارد مدار تمام جمع‌کننده نیز می‌شود. حال با توجه به مقادیر مختلف بیت کنترل (۰ یا ۱) این مدار می‌تواند عمل جمع و یا تفریق به روش متمم ۲ را انجام دهد.

C=0 ⟶ S = A+B+0=A+B(Sum)

C=1 ⟶ S=A+B'+1=A-B(Subtract)

جمع‌کننده BCD : جمع‌کننده BCD، مداریست که دو رقم BCD چهار بیتی را با یکدیگر جمع می‌کند. با اتصال چند جمع‌کننده BCD به یکدیگر می‌توان جمع اعداد BCD چند رقمی را انجام داد.

برای طراحی جمع‌کننده BCD دو روش کلاسیک و غیر کلاسیک وجود دارد:

۱. یا باید به روش کلاسیک عمل کنیم و مراحل جدول صحت ، جدول کارنو و به دست آوردن روابط را طی کرده تا به مدار برسیم.در این صورت یک کارنو با ۲۵۶ ردیف لازم داریم که پیاده‌سازی آن و به دست آوردن عبارت های ساده شده خروجی‌ها مشکل خواهد بود.

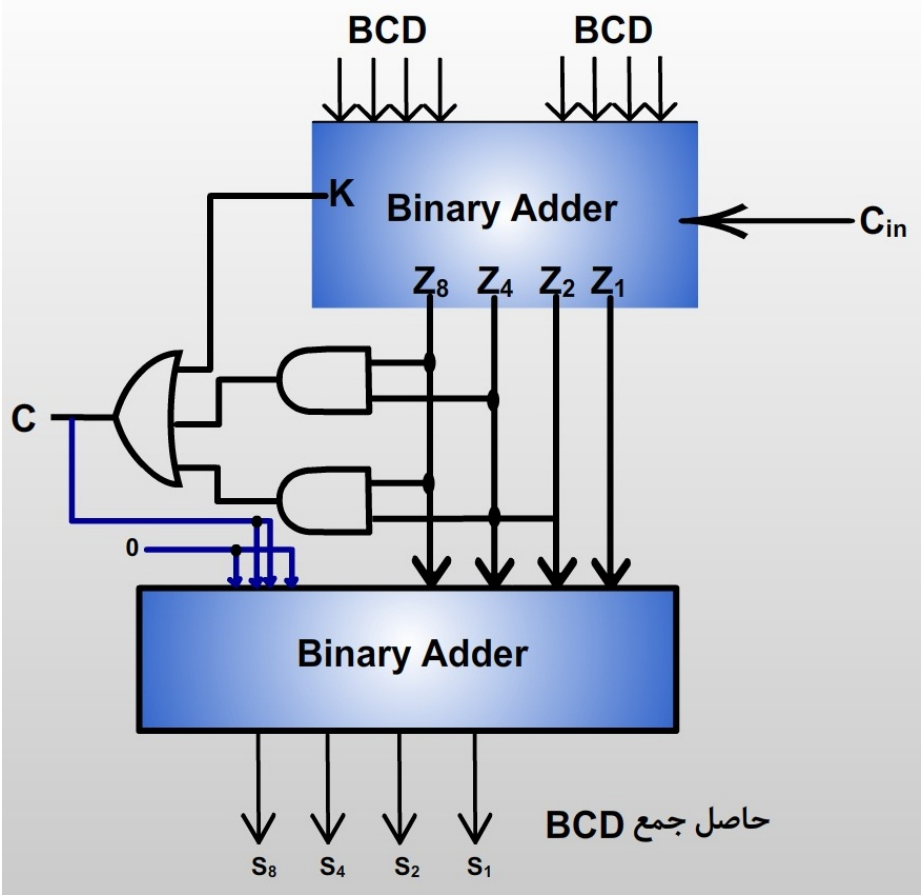
۲. در روش غیر کلاسیک می‌توانیم از جمع‌کننده دودویی استفاده کرده و جمع اعداد BCD را به صورت دودویی انجام دهیم. در این صورت خروجی جمع‌کننده دودویی چهار بیتی طبق جدول زیر خواهد بود.

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

این خروجی در برخی موارد مستلزم اصلاح است. به عبارت دیگر، هرگاه حاصل جمع دو عدد BCD بزرگتر از ۹ شود، نیاز به اصلاح داشته و حاصل باید با عدد ۶ دودویی $(0110)_2$ جمع شود. دلیل جمع با ۶ این است که اختلاف بین یک رقم نقلی در با ارزش‌ترین مکان بیتی حاصل از جمع دودویی و نقلی دهدهی برابر است با : 16-10 = 6. رابطه مدار اصلاح از جدول کارنو یا به روش سعی و خطا به دست می‌آید. در جدول فوق، C همان سیگنال اصلاح خروجی است که هرگاه برابر 1 شود، خروجی جمع‌کننده دودویی اول با یک ۶ دودویی $(0110)_2$ جمع خواهد شد. طبق جدول، عبارت جبری C برابر خواهد شد با :

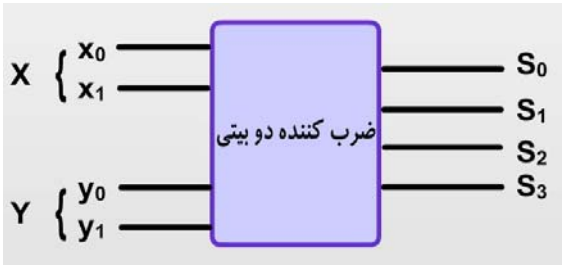
$$C = K + Z_8.Z_2 + Z_8.Z_4$$

شکل زیر دیاگرام بلوکی یک جمع‌کننده BCD چهار بیتی را نشان می‌دهد. در این مدار اگر حاصل مدار اصلاح 0 باشد، نتیجه با 0000 و اگر 1 باشد، نتیجه با 0110 جمع می‌شود.



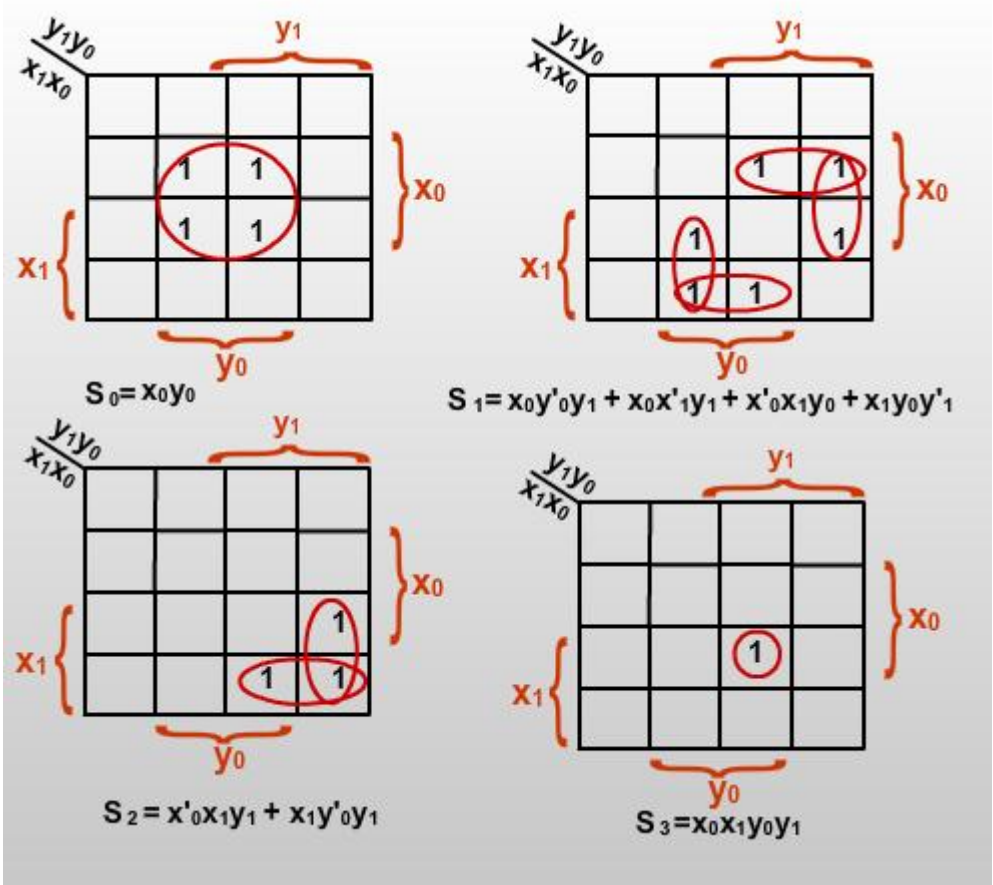
ضرب کننده ها : یکی دیگر از مدارات محاسباتی پر کاربرد در سیستمهای دیجیتال، ضرب کننده ها هستند؛ طراحی ضرب کننده ها را می‌توان به دو روش کلاسیک و غیر کلاسیک انجام داد. در ادامه یک ضرب کننده ۲ بیت در ۲ بیت را به هر دو روش طراحی خواهیم نمود.

۱. روش کلاسیک : در این روش ابتدا دیاگرام بلوکی مدار را رسم کرده و تعداد ورودی‌ها و خروجی‌های مدار را مشخص می‌کنیم.



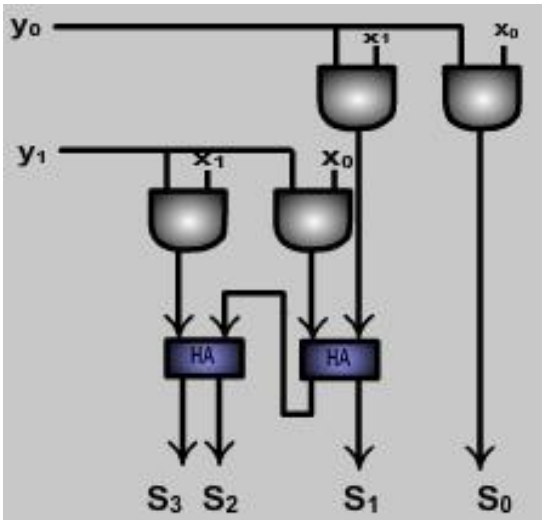
سپس برای پر کردن جدول صحت مدار، تمام ترکیبات ورودی را لیست کرده و به ازای هر ترکیب، حاصل ضرب دو بیت را در دو بیت دیگر بدست آورده و به صورت دودویی در ستون خروجی می نویسیم. در مرحله آخر نیز معادلات خروجی‌های مدار را به دست می‌آوریم.

X_1	X_0	Y_1	Y_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1



۲. روش غیر کلاسیک : در روش غیر کلاسیک، هر مساله راه حل خاص خود را دارد و هیچ راه حلی برای مسائل دیگر عمومیت ندارد. برای طراحی مدار ضرب کننده سعی می‌کنیم روش دستی ضرب را با ابزارهای منطقی پیاده سازی کنیم، بدون اینکه به مراحلی چون جدول صحت نیاز داشته باشیم. عمل ضرب دو بیت در ریاضی معادل عبارت x AND y در منطقی است. ضرب متعارف دستی به صورت زیر انجام می‌شود :

با الگو گرفتن از روش ضرب دستی و استفاده از مدارات جمع‌کننده و گیت های AND مدار ضرب کننده مورد نظر را طراحی می‌کنیم که دیاگرام بلوکی آن در شکل زیر مشاهده می‌شود:



با به‌کارگیری روش غیر کلاسیک در طراحی ضرب کننده ها، می‌توان ضرب کننده‌های n بیت در m بیت را بدون نیاز به جداول صحت با تعداد سطرهای زیاد به سادگی طراحی نمود.

مقایسه گرها، مدارهای مولد بیت توازن و مدارهای ترکیبی ماژولار

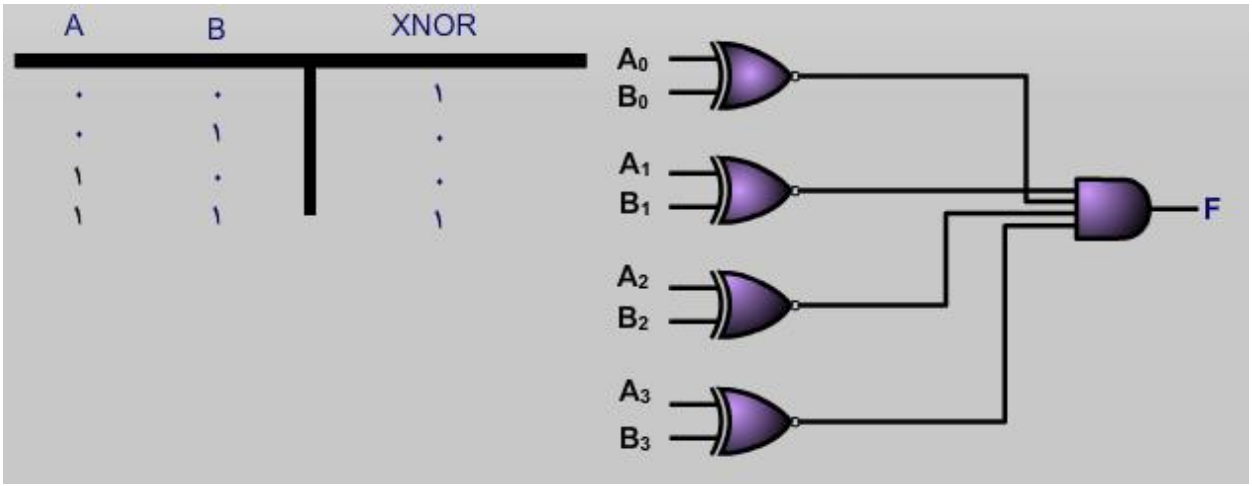
هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- آشنایی با مفهوم مقایسه گر مقدار و ساخت مقایسه گرهای یک بیتی و چند بیتی
- مدارهای تولید و تست خطای توازن
- آشنایی با مدارات ترکیبی ماژولار، خواص و کاربرد آنها

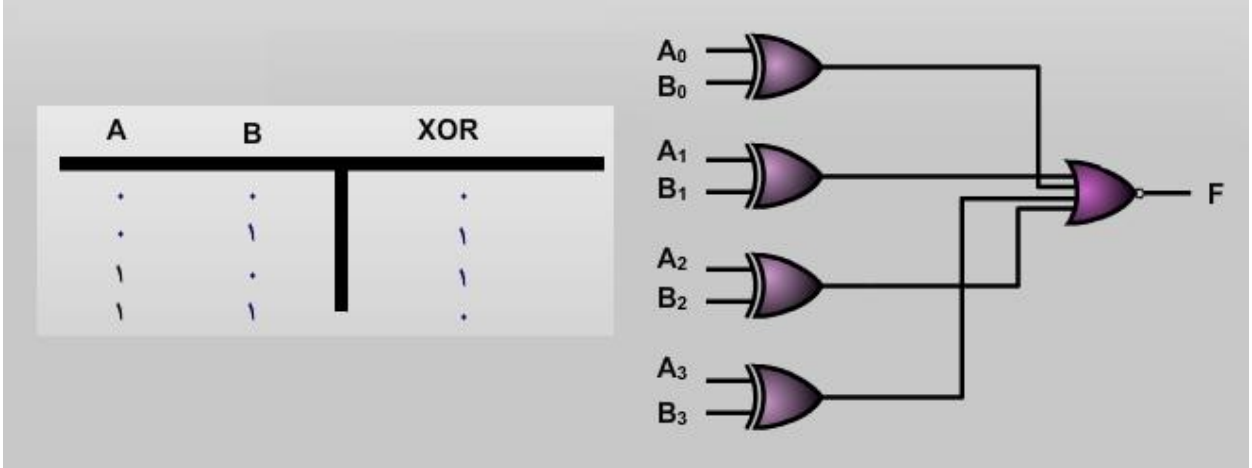
مقایسه‌گر مقدار: یک مقایسه‌گر مقدار، مداری ترکیبی است که دو عدد A و B را مقایسه می‌کند و اندازه نسبی آنها را تعیین می‌کند. نتیجه این مقایسه می‌تواند با سه متغیر خروجی (A=B)، (A>B) و (A<B) مشخص گردد. در برخی مدارات مقایسه کننده مقدار، تنها تحقیق تساوی دو عدد مطلوب است و نه روابط بزرگتر و کوچکتر بودن.

در طراحی مقایسه‌گرهای مقدار، معمولاً گیت های XOR و XNOR کاربرد زیادی دارند. در ادامه در نظر است یک مقایسه کننده چهاربیتی با استفاده از گیت های XOR و همچنین XNOR طراحی شود که دو عدد را با هم مقایسه کرده و در صورت مساوی بودن خروجی یک و در غیر این صورت خروجی صفر شود.

طراحی مقایسه کننده با XNOR :



طراحی مقایسه کننده با XOR :



مقایسه کننده ۴ بیتی کامل : می‌خواهیم یک مقایسه کننده ۴ بیتی طراحی کنیم که دارای خروجی‌های کوچکتر، مساوی و بزرگتر باشد. این مدار دارای ۳ خروجی E، G و L است که به ترتیب بیانگر حالت های (A=B)، (A>B) و (A<B) می‌باشند.

دیاگرام بلوکی و معادلات خروجی‌های مدار در زیر مشاهده می‌شوند. برای به دست آوردن معادلات خروجی نهایی، توابع میانی X_i تعریف شده‌اند که هر X_i از رابطه $X_i = A_i \text{ XNOR } B_i$ به دست آمده و تساوی دو بیت متناظر از اعداد A و B را تحقیق می‌کند.

$$L = A'_3 B_3 + X_3 A'_2 B_2 + X_3 X_2 A'_1 B_1 + X_3 X_2 X_1 A'_0 B_0$$

$$G = A_3 B'_3 + X_3 A_2 B'_2 + X_3 X_2 A_1 B'_1 + X_3 X_2 X_1 A_0 B'_0$$

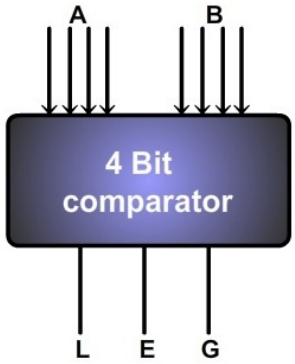
$$X_3 = A_3 \text{ XNOR } B_3$$

$$X_2 = A_2 \text{ XNOR } B_2$$

$$X_1 = A_1 \text{ XNOR } B_1$$

$$X_0 = A_0 \text{ XNOR } B_0$$

$$E = X_3 . X_2 . X_1 . X_0$$



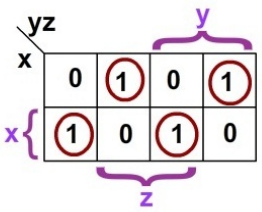
مدار تولید و تست خطای توازن : در طراحی این مثال قصد داریم برای یک کد n بیتی، مدار تولید بیت توازن زوج و همچنین مدار تست خطای توازن را طراحی نماییم. بیت توازن زوج، بیتی است که به بیت های کد ارسالی اضافه شده و تعداد یک ها را در کد حاصل زوج خواهد نمود.

در مقصد نیز گیرنده بیت های کد ارسالی را به همراه بیت توازن دریافت می کند. اگر تعداد کل یک های دریافتی زوج باشد، کد ارسالی درست است، در غیر این صورت خطائی در کد رخ داده است. این روش فقط وجود خطا را تشخیص داده و توان اصلاح آنرا ندارد.

در ادامه مدارات مذکور را برای یک کد ۳ بیتی طراحی می‌نمائیم. در شکل های زیـر، P بیـت تـوازن زوج تولیـد شده در فرستنده و E بیت تشخیص خطای توازن تولید شده در گیرنده می‌باشند.

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

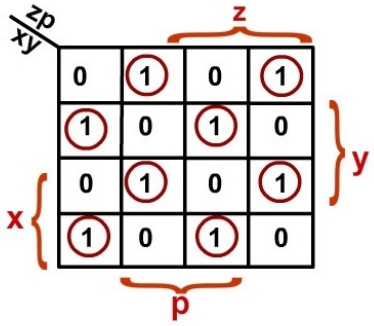
$$P = x \oplus y \oplus z$$



P=x XOR y XOR z

X	Y	Z	P	E
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

جدول کارنو و معادله را مشاهده می کنید.



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

توجه : توازن زوج NOT توازن فرد است، پس کافیست برای به دست آوردن توازن فرد، خروجی مدار توازن زوج را NOT کنیم.

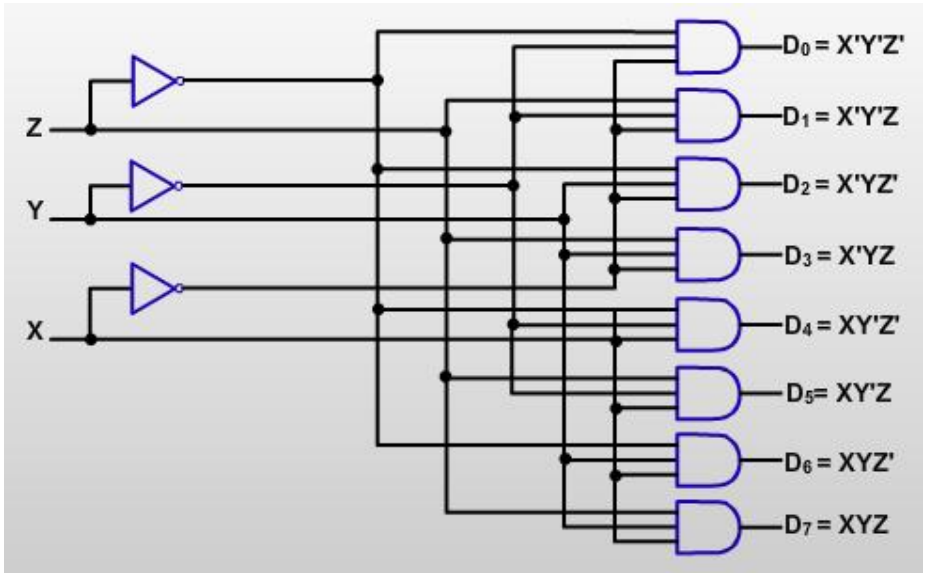
مدارهای ترکیبی ماژولار : گیت ها و مدارهای منطقی برای استفاده در طراحی‌ها به صورت قطعات مدارات مجتمع (IC) در اختیار طراحان قرار می‌گیرند. دسته‌ای از این مدارات که به مدارات ترکیبی ماژولار معروفند، کاربرد زیادی در طراحی سیستم‌های دیجیتال از جمله واحد کنترل CPU ها دارند. از جمله این مدارات می‌توان به دیکدرها، اِنکدرها، مالتی پلکسرها، دی‌مالتی پلکسرها و ابزارهای قابل برنامه ریزی اشاره نمود. در ادامه به معرفی دیکدرها و مالتی پلکسرها، ویژگی‌ها و کاربردهای آن ها خواهیم پرداخت.

مدار رمز گشا یا دیکدر : دیکدر مداری ترکیبی است که اطلاعات دودویی را از n خط ورودی به حداکثر 2^n خط خروجی منحصر به فرد تبدیل می‌کند. کاربرد دیکدرها، تولید 2^n مینترم از n متغیر ورودی است. در دیکدرها در هر زمان حداکثر یک خروجی فعال خواهد بود.

به عنوان مثال، یک دیکدر با ۳ ورودی، دارای ۸ خروجی است که جدول درستی آن در زیر مشاهده می‌شود.

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

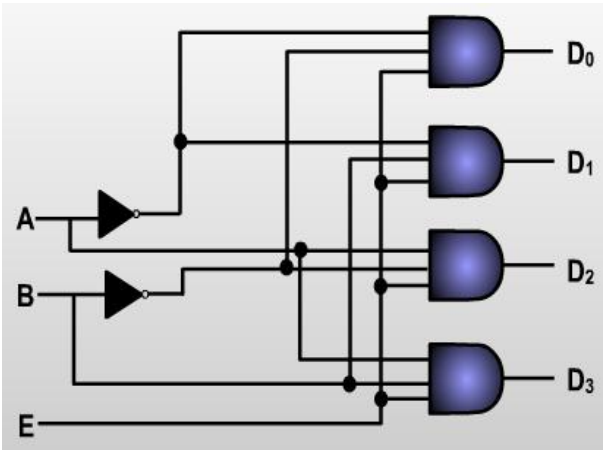
طراحی دیکدر ۸×۳ : در دیکدر در هر لحظه، به ازای هر ترکیب ورودی فقط یک خروجی فعال است. خروجی روابط خروجی‌ها و نمودار مداری یک دیکدر ۸×۳ در شکل زیر مشاهده می شود.



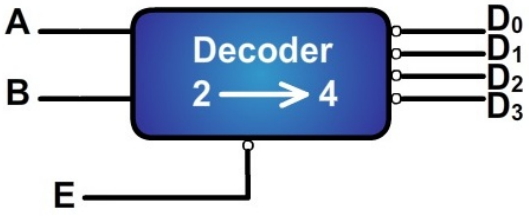
ورودی‌های فعال ساز : اکثر دیکدرها و مدارات ماژولار دارای یک یا چند ورودی فعال ساز هستند تا بتوان آن ها را داخل مدارات بزرگتر استفاده نموده و با سیگنال های کنترلی لازم، فعال یا غیر فعال نمود. این ورودی‌ها از دو نوع Active High و Active Low می‌باشند که نوع اول با 1 و نوع دوم با 0 فعال می‌شوند.

دیکدر ۲×۴ با ورودی کنترل Active High : در این مدار زمانی که ورودی فعال ساز برابر 1 است، دیکدر فعال شده و مینترم متناظر با ترکیب ورودی را در خروجی فعال خواهد نمود. دیاگرام بلوکی، جدول صحت دیکدر به همراه نمودار مداری آن در شکل مشاهده می‌شوند.

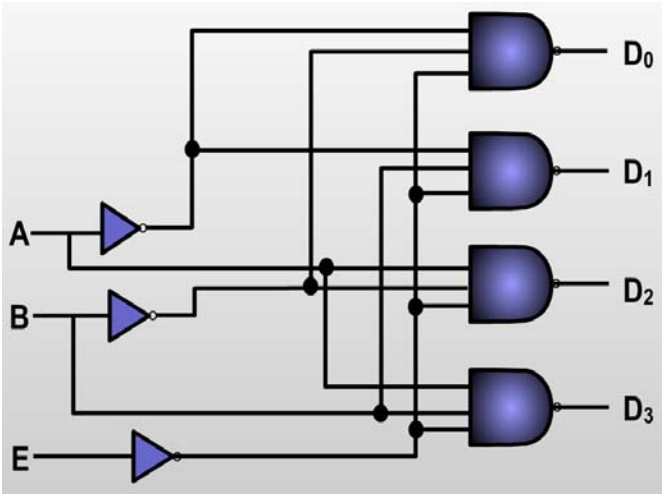
E	A	B	D3	D2	D1	D0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



دیکدر ۲×۴ با ورودی کنترل Active Low : در این مدار زمانی که ورودی فعال ساز برابر 0 است، دیکدر فعال شده و مینترم متناظر با ترکیب ورودی را در خروجی فعال خواهد نمود. دیاگرام بلوکی، جدول صحت دیکدر به همراه نمودار مداری آن در شکل مشاهده می‌شوند. توجه کنیدکه در این مدار، خروجی‌ها نیز به صورت Active Low طراحی شده‌اند؛ یعنی زمانی که خروجی فعال است، مقدار آن برابر 0 می‌باشد.



E	A	B	D3	D2	D1	D0
0	0	0	1	1	1	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	1
1	X	X	1	1	1	1

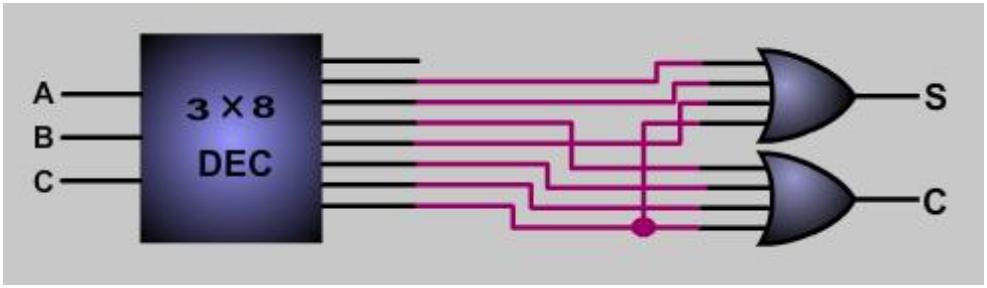


پیاده سازی توابع منطقی با استفاده از دیکدرها : هر تابع جبر بول را می‌توان بـا اسـتفاده از دیکـدر و گیـت های مناسب پیاده سازی نمود. اما ابتدا باید تابع به فرم استاندارد جمع حاصل ضرب‌ها تبدیل شود.

به عنوان مثال یک تمام جمع‌کننده را می‌توان با یک دیکدر ۳×۸ و دو عدد گیت OR پیاده سازی کرد.

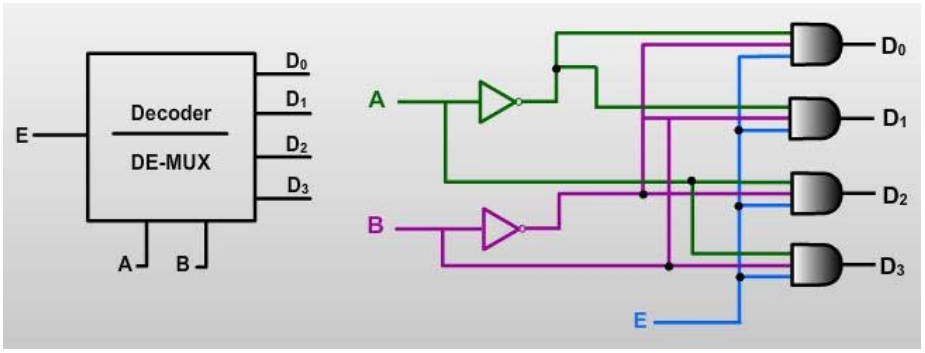
$S(x,y,z) = \sum m(1,2,4,7)$
$C(x,y,z) = \sum m(3,5,6,7)$

نمودار مداری تمام جمع‌کننده با استفاده از دیکدر در شکل زیر مشاهده می‌شود.

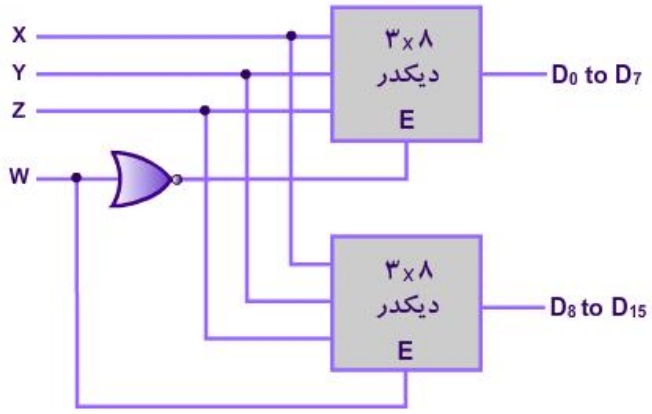


دی مالتی پلکسر: دی مالتی پلکسر مداری است که اطلاعات را از یک خط دریافت کرده و آن را به یکی از 2^n خط خروجی ممکن هدایت می‌نماید.انتخاب یک خروجی خاص با ترکیب n خط انتخاب صورت می‌گیرد.یک دیکدر با ورودی فعال‌ساز می‌تواند به‌عنوان یک دی مالتی پلکسر عمل کند. به‌عنوان مثال دیکدر شکل زیر را می‌توان به‌عنوان یک دی مالتی پلکسر 1 به 4 به‌کاربرد. تنها متغیر ورودی E مسیری به تمام جهار خروجی دارد، ولی اطلاعات ورودی تنها به یکی از خروجی‌ها هدایت می‌شود که این خروجی با دو خط انتخاب A و B انتخاب می‌شود. به‌عنوان مثال اگر خطوط انتخابی AB =10 باشند، خروجی D_2 مثل ورودی E خواهد بود، درحالی‌که دیگر خروجی‌ها در 0 نگه داشته خواهند شد.

چون عمل دیکدر و دی مالتی پلکسر با استفاده از یک مدار حاصل می‌شود، یک دیکدر با ورودی فعال‌ساز را دیکدر / دی مالتی پلکسر هم می‌خوانند.



ساخت دیکدرهای بزرگ‌تر: برای این عمل می‌توان دیکدرهای کوچک با ورودی‌های فعال ساز را به‌هم متصل کرد. به‌عنوان مثال برای ساخت یک دیکدر 4 به 16 می‌توان دیکدرهای کوچک‌تر با استفاده از دیکدرهای 3 به 8 مطابق شکل زیر عمل می‌کنیم.



در این‌گونه طراحی‌ها معمولاً ورودی پر ارزش را به ورودی‌های فعال‌ساز، دیکدرها و سایر ورودی‌ها را به ورودی‌های هرکدام از دیکدرها متصل می‌نماییم. با توجه به شکل فوق، وقتی w=0 است، دیکدر فوقانی فعال و دیکدر پایینی غیر فعال می‌شود. در این حالت خروجی‌های دیکدر پایینی همگی 0 خواهد بود و خروجی‌های بالایی، مینترم‌های 0 تا 7 را تولید می‌کنند. وقتی w=1 باشد، دیکدر بالایی غیر فعال شده و خروجی‌های پایینی، مینترم‌های 8 تا 15 را تولید می‌کنند.

انکدرها : انکدر مداری است که عکس عمل یک دیکدر را انجام می‌دهد. یک انکدر دارای 2^n خط ورودی و n خط خروجی است. خطوط خروجی، کد دودویی مربوط به مقدار دودویی ورودی را تولید می‌نماید. به‌عنوان مثال یک انکدر 8 به3 دارای جدول درستی و معادلاتی به شکل زیر است؛ در این ودار در هر لحظه از زمان تنها یک ورودی می‌توتند مقدار 1 را داشته باشد.

ورودی ها								خروجی ها		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

معادلات خروجی انکدر به‌صورت زیر می‌باشند؛ این انکدر با ۳ گیت OR قابل پیاده سازی است.

$$X=D_4+D_5+D_6+D_7$$
$$Y=D_2+D_3+D_6+D_7$$
$$Z=D_1+D_3+D_5+D_7$$

انکدر با ساختار فوق دارای دو محدودیت است؛ یکی این‌که در هر لحظه از زمان یک ورودی فعال می‌باشد. اگر دو ورودی به‌طور هم‌زمان فعال شوند، خروجی ترکیبات نامفهومی را تولید خواهد کرد.

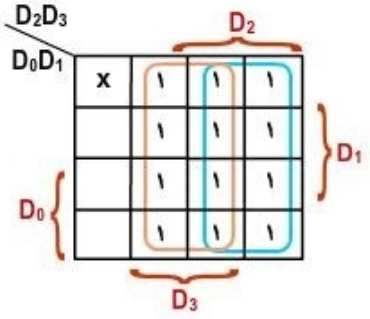
مشکل دیگر این است که به ازای حالتی که تمام ورودی‌ها 0 هستند، یک خروجی تمام 0 تولید می‌شود. این خروجی برابر با حالتی است که در آن $D_0=1$ می‌باشد. این ایراد را می‌توان با تهیه یک خروجی بیشتر حل کرد.

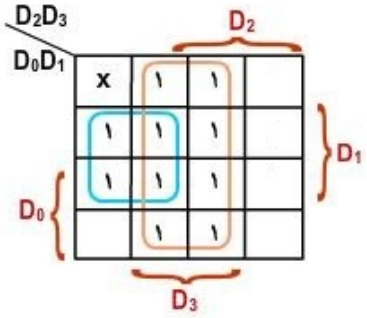
انکدر اولویت‌دار : مداری است که تابع اولویت‌دار را در خود دارد. در این نوع انکدر، اگر دو یا چند ورودی به‌طور هم‌زمان برابر 1 شوند، ورودی با اولویت بالاتر پیش خواهد افتاد. جدول درستی یک انکدر با اولویت چهار ورودی در زیر ملاحظه می‌گردد.

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

همان‌طور که در جدول ملاحظه می‌شود، مدار دارای یک خروجی V است که بیت معتبر خوانده می‌شود. اگر همه‌ی ورودی‌ها 0 باشند، این بیت نیز صفر شده و باعث می‌شود خروجی‌های دیگر حالت بی‌اهمیت داشته باشند و بررسی نشوند.

جدول کارنوی خروجی‌های X و Y در زیر مشاهده می‌شوند. توابع بولی ساده شده برای انکدر اولویت از جداول کارنو حاصل می‌گردند. شرط خروجی V یک تابع OR از همه‌ی متغیرهای ورودی است.



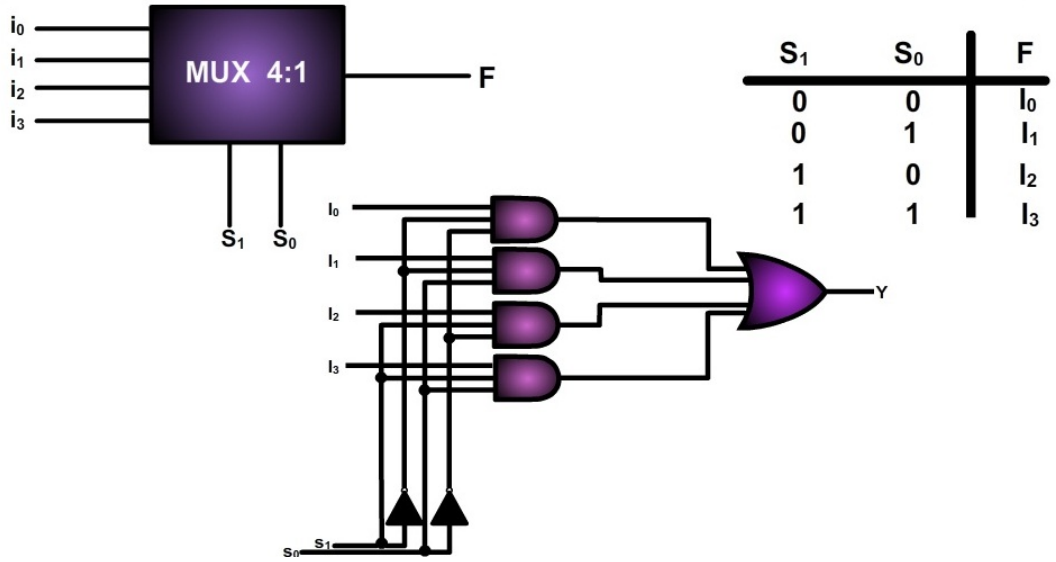


معادلات خروجی انکدر اولویت در زیر مشاهده می‌شوند :

$$X=D_2+D_3 \qquad Y=D_3+D_1D'_2 \qquad V=D_0+D_1+D_2+D_3$$

مدار تسهیم کننده یا مالتی پلکسر: مالتی پلکسر یک مدار انتخاب گر است که از چند ورودی فقط یکی از آنها را انتخاب کرده و به خروجی انتقال می‌دهد. یک مالتی پلکسر دارای n خط انتخاب، 2^n ورودی و یک خط خروجی می‌باشد که با توجه به مقدار ورودی‌های انتخاب در هر لحظه، یکی از ورودی‌ها انتخاب شده و در خروجی مالتی پلکسر ظاهر می‌شود.

دیاگرام بلوکی، جدول عملکرد و نمودار مدار یک مالتی پلکسر ۱×۴ را ملاحظه می‌فرمایید :



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

پیاده سازی توابع منطقی با استفاده از مالتی پلکسرها : هر تابع جبر بول را می‌توان با استفاده از مالتی پلکسر مناسب و گیت های لازم پیاده سازی نمود. برای این منظور بهتر است که تابع ابتدا در جدول درستی قرار داده شود. به عبارت دیگر می‌توان گفت که هر تابع n متغیره را به راحتی می‌توان با یک مالتی پلکسر با n-1 ورودی انتخاب پیاده سازی نمود. مالتی پلکسرهای مناسب برای پیاده سازی توابع ۳ تا ۵ متغیره را در زیر ملاحظه می‌فرمایید.

برای تابع سه متغیره : **MUX 4 :1** برای تابع چهار متغیره : **MUX 8 :1** برای تابع پنج متغیره : **MUX 16 :1**

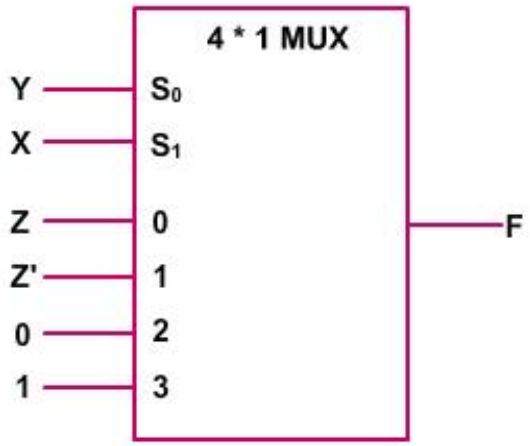
مثال : تابع زیر را با مالتی پلکسر مناسب پیاده سازی کنید.

$$F(X,Y,Z) = \sum (1,2,6,7)$$

الف) جدول درستی

X	Y	Z	F	
0	0	0	0	F = Z
0	0	1	1	
0	1	0	1	F = Z'
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

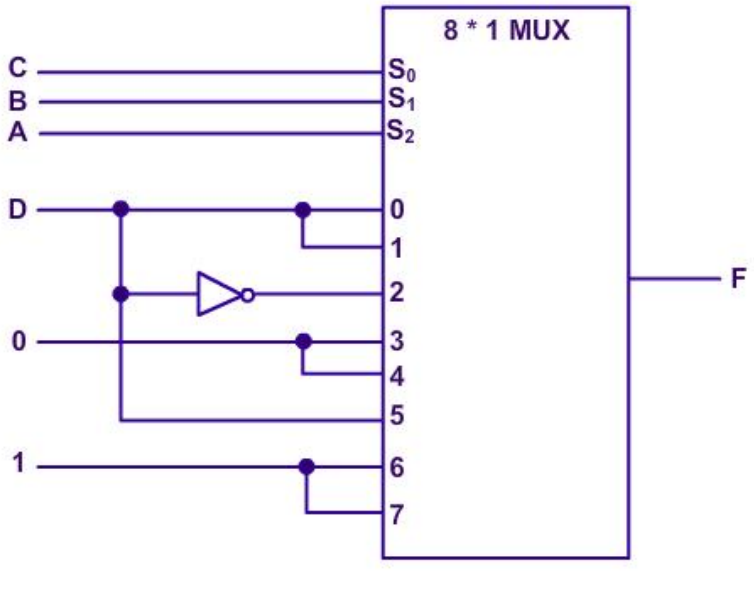
ب) پیاده سازی مالتی پلکسر



مثال : تابع زیر را با مالتی پلکسر مناسب پیاده سازی کنید.

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

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



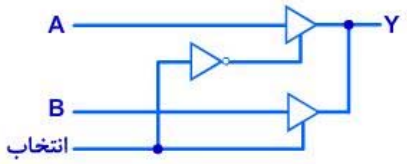
گیت‌های سه حالته : یک گیت سه حالته مداری دیجیتالی است که سه حالت را از خود به نمایش می‌گذارد. دو حالت، هم‌چون گیت‌های معمولی همان منطق 0 و 1 است. حالت سوم حالت امپدانس بالاست. حالت امپدانس بالا (High Impedance) مثل مدار باز عمل می‌کند و در این‌حالت خروجی از درون قطع بوده و مدار دارای مفهوم منطقی با ارزشی نیست.

گیت‌های سه حالته ممکن است به‌عنوان گیت‌های AND وNAND نیز عمل کنند. با این وجود اغلب به‌عنوان بافر استفاده می‌شوند. نمودار یک گیت بافر سه حالته در شکل مشاهده می‌شود :

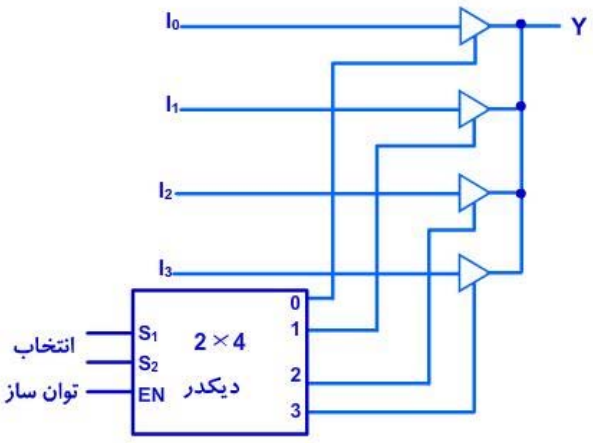
در این شکل، وقتی که ورودی کنتزل برابر 1 است، خروجی فعال شده و گیت مانند یک بافر معمولی عمل می‌کند. وقتی که ورودی کنترل 0 شود، خروجی غیر فعال شده و گیت بدون توجه به مقدار ورودی‌های اصلی به حالت امپدانس بالا می‌رود.

اصلی‌ترین ویژگی و مزیت گیت‌های سه حالته این است که تعداد زیادی از خروجی‌های سه حالته می‌توانند به‌هم متصل شوند، بدون آن‌که تأثیری بر روی بار شدن داشته باشند.

کاربرد گیت‌های سه حالته : یکی از کاربردهای گیت‌های سه حالته، در طراحی گذرگاه داده (Data Bus) در سیستم‌های دیجیتال است. به‌عنوان کاربرد دیگری از این نوع گیت‌ها می‌توان ساخت مالتی پلکسر را نام برد که نحوه‌ی طراحی مالتی پلکسرهای 2 به 1 و 4 به 1 با استفاده از گیت‌های سه حالته در شکل مشاهده می‌شود.



(الف) مالتی پلکسر ۲ به ۱



(ب) مالتی پلکسر ۴ به ۱

فصل پنجم

مدارهای منطقی ترتیبی همزمان

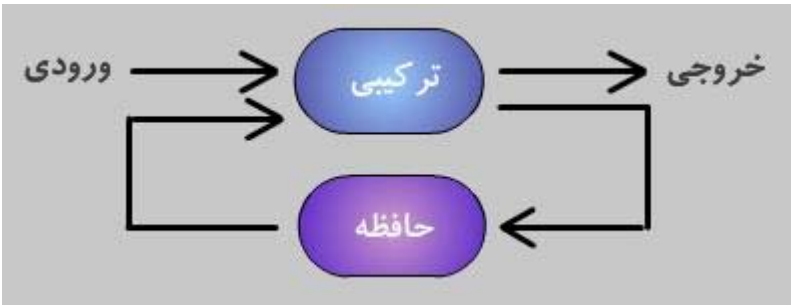
عناصر حافظه در مدارات ترتیبی (لچ‌ها و فلیپ فلاپ‌ها)

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- شناخت مدارهای منطقی ترتیبی
- شناخت عناصر حافظه در مدارات ترتیبی شامل لچ‌ها و فلیپ فلاپ‌ها و نحوه طراحی آن‌ها
- انواع فلیپ فلاپ‌ها از نظر نحوه تحریک سیگنال ساعت
- فلیپ فلاپ‌های Master-Slave
- ورودی‌های غیر همزمان فلیپ فلاپ‌ها

انواع مدارهای منطقی : مدارهای منطقی در سیستم های دیجیتال می‌توانند از نوع ترکیبی یا ترتیبی باشند. مدارهای ترکیبی در فصل قبل به تفصیل مورد بررسی قرار گرفتند. در این فصل به بررسی مدارهای ترتیبی، فلیپ فلاپ‌ها و نحوه تحلیل و طراحی آن‌ها می‌پردازیم.

مدارهای منطقی ترتیبی : یک مدار ترتیبی، متشکل از مداری ترکیبی است که عناصر حافظه برای ایجاد یک مسیر پسخورد به آن وصل شده‌اند. عناصر حافظه قطعاتی هستند که می‌توانند اطلاعات دودویی را ذخیره کنند. بنابراین مدارهای ترتیبی علاوه بر گیت های منطقی، از عناصر حافظه نیز استفاده می‌کنند. خروجی‌های این مدارها در هر لحظه از زمان، تابعی از ورودی‌ها و حالت عناصر حافظه در آن لحظه است. لذا عملکرد مدار باید به‌وسیله حالات داخلی و ترتیب زمانی ورودی‌ها مشخص گردد.



مدارهای ترتیبی غیر همزمان : دسته‌ای دیگر از مدارهای ترتیبی، مدارهای ترتیبی غیر همزمان هستند. رفتار یک مدار ترتیبی غیر همزمان به ترتیب تغییر سیگنال‌های ورودی که می‌توانند در هر لحظه از زمان روی آن تاثیر کنند، وابسته می‌باشد. عناصر حافظه‌ای که در این نوع مدارات به کار می‌روند، نوعی وسایل تاخیر زمانی هستند که از گیت‌های منطقی تشکیل می‌شوند. لذا این نوع مدارات را می‌توان مدارهایی ترکیبی با پسخورد دانست. به دلیل وجود پسخورد در بین گیت‌های منطقی، هر مدار ترتیبی غیر همزمان در هر لحظه از زمان ممکن است ناپایدار شود. مدارهای ترتیبی غیر همزمان در این فصل مورد بررسی قرار نخواهند گرفت.

مدارهای ترتیبی همزمان : سیگنال‌هایی را مورد استفاده قرار می‌دهند که فقط در لحظات گسسته‌ای از زمان روی عناصر حافظه اثر می‌گذارند.همزمانی در این مدارات با وسیله‌ای به نام مولد پالس ساعت انجام می‌شود.

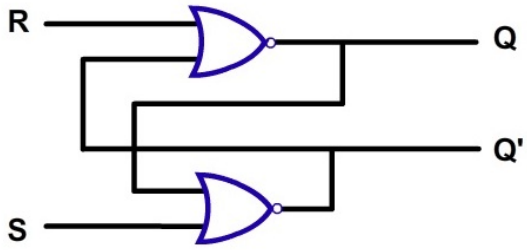
فلیپ فلاپ‌ها : عناصر ذخیره سازی دودویی در مدارهای ترتیبی ساعت‌دار را فلیپ فلاپ گویند. هر فلیپ فلاپ یک وسیله ذخیره سازی دودویی بوده و تا زمانی که تغذیه به مدارش اعمال شود، قادر است یک بیت اطلاعات را در خود ذخیره کند.

لچ‌ها : ساده‌ترین نوع فلیپ فلاپ‌ها هستندکه با سطوح سیگنال پالس ساعت عمل می‌کنند. لچ‌ها مدارهای مبنایی هستند که همه فلیپ فلاپ‌ها با آن‌ها ساخته می‌شوند. تفاوت میان لچ‌های مختلف در نحوه تغییر حالت و قابلیت های آن‌ها می‌باشد. اولین نوع لچ که مورد بررسی قرار می‌دهیم، لچ SR است که با گیت‌های NAND یا NOR ساخته می‌شود.

SR Latch با NOR : مداری با دو گیت NOR است که به طور متقاطع به هم وصل شده‌اند. این مدار دو ورودی دارد که با S به معنی نشاندن (Set) و R به معنی بازنشانی (Reset) نام‌گذاری شده‌اند.

در این لچ، وقتی Q=1 و Q'=0 باشند، گوییم لچ در حالت نشانده (Set) است. اگر Q=0 و Q'=1 باشند، گوییم لچ در حالت بازنشانی (Reset) است. خروجی‌های Q و Q' متمم یکدیگرند. با این وجود وقتی هر دو ورودی همزمان 1 شوند، حالت نامعین 0 برای هر دو خروجی رخ خواهد داد. نمودار مداری و جدول مشخصه (درستی) این نوع لچ در زیر مشاهده می‌شوند.

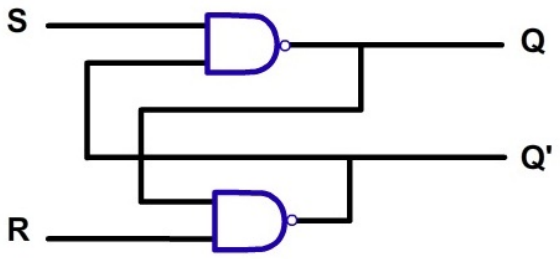
S	R	Q	Q'	
1	0	1	0	SET
0	1	0	1	RESET
0	0	0	1	با فرض Q=0
0	0	1	0	با فرض Q=1
1	1	0	0	حالت غیر مجاز



SR Latch با NAND : مداری با دو گیت NAND است که به طور متقاطع به هم وصل شده‌اند. این مدار دو ورودی دارد که با S به معنی نشاندن (Set) و R به معنی بازنشانی (Reset) نام‌گذاری شده‌اند.

در این لچ، وقتی Q=1 و Q'=0 باشند، گوییم لچ در حالت نشانده (Set) است. اگر Q=0 و Q'=1 باشند، گوییم لچ در حالت بازنشانی (Reset) است. خروجی‌های Q و Q' متمم یکدیگرند؛ با این وجود وقتی هر دو ورودی همزمان 0 شوند، حالت نامعین 1 برای هر دو خروجی رخ خواهد داد. نمودار مداری و جدول مشخصه (درستی) این نوع لچ در زیر مشاهده می‌شوند.

S	R	Q	Q'	
1	0	0	1	RESET
0	1	1	0	SET
1	1	0	1	با فرض Q=0
1	1	1	0	با فرض Q=1
0	0	1	1	حالت غیر مجاز



نکته ۱ : ورودی‌های مدار NOR-Latch از نوع Active High و ورودی‌های مدار NAND-Latch از نـوع Active Low می‌باشند.

نکته ۲ : برای این‌که خروجی لچ ثابت بماند و با هر تغییر ورودی، خروجی آن نیز تغییر نکند، لازم است یک ورودی کنترلی اضافه شود تا تغییرات ورودی در صورت فعال بودن آن اعمال گردد. این ورودی کنترلی را کلاک یا سیگنال ساعت می‌نامیم و مداری که ورودی‌های آن با پالس‌های ساعت همگام می‌شوند، فلیپ فلاپ نامیده می‌شود.

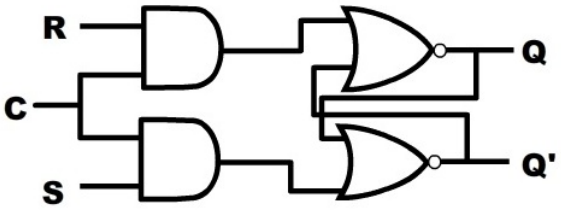
انواع فلیپ فلاپ :

1. SR-F.F. (Set Reset Flip Flop)
2. D-F.F. (Delay Flip Flop)
3. JK-F.F. (Jump Kill Flip Flop)
4. T-F.F. (Toggle Flip Flop)

فلیپ فلاپ SR (SR-Flip Flop) : مشابه لچ‌های SR، این فلیپ فلاپ را می‌توان با گیت‌های NAND یا NOR ساخت. به‌عنوان مثال همان گونه که در شکل نیز مشاهده می‌شود، برای ساخت این فلیپ فلاپ با گیت‌های NOR، ورودی‌های S و R ابتدا با یک ورودی کنترل C، AND شده و سپس وارد گیت‌های NOR طبقه خروجی می‌شوند. در این فلیپ فلاپ هرگاه ورودی کنترل برابر 0 باشد، حالت Latch پیش آمده و با تغییر ورودی‌ها، حالت لچ تغییر نخواهد کرد. با فعال شدن ورودی C، تغییرات S و R به مدار اعمال خواهند شد.

جدول مشخصه و نمودار مداری فلیپ فلاپ SR در زیر مشاهده می‌شوند. در جدول زیر، حالت فعلی یعنی خروجی فعلی به‌عنوان ستون ورودی ترسیم شده و حالت بعدی یعنی خروجی جدید به‌عنوان ستون خروجی ترسیم شده است.

C	S	R	Q(t+1)
0	x	x	Q(t)
1	0	0	Q(t)
1	0	1	0
1	1	0	1
1	1	1	x



از روی جدول و با استفاده از روش‌های ساده سازی می‌توان معادله مشخصه فلیپ فلاپ SR را بـه‌دسـت‌آورد کـه برابر خواهد شد با :

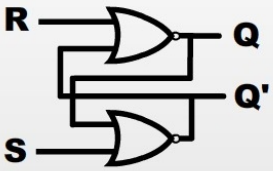
$$Q(t+1) = S + R'Q$$

تعریف ۱ : جدول مشخصه یک فلیپ فلاپ، حالت بعدی فلیپ فلاپ را به‌صورت تابعی از ورودی‌ها و حالـت فعلـی آن بیان می‌کند.

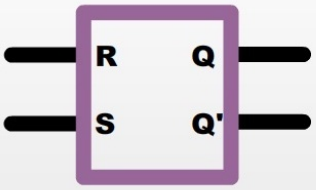
تعریف ۲ : معادله مشخصه یک فلیپ فلاپ، شکل دیگری از جدول مشخصه آن بوده و حالت بعدی فلیپ فلاپ را به‌صورت تابعی از ورودی‌ها و حالت فعلی آن بیان می‌کند.

نمودار مداری، جدول مشخصه و دیاگرام بلوکی لچ‌ها و فلیپ فلاپ‌های SR به‌صورت خلاصه در زیر مشاهده مـی-شوند :

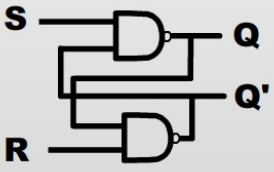
NOR-LATCH:



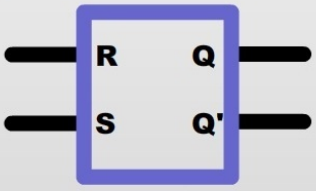
S	R	$Q_{(t+1)}$
0	1	RESET
1	0	SET
0	0	LATCH
1	1	X



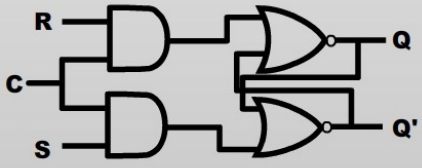
NAND-LATCH:



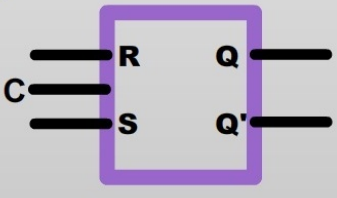
S	R	$Q_{(t+1)}$
0	1	SET
1	0	RESET
1	1	LATCH
0	0	X



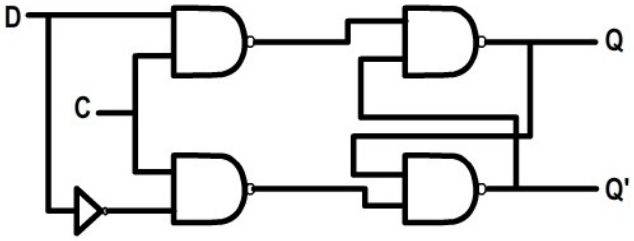
SR-F.F(SET&RESET)

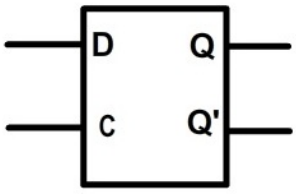


S	R	$Q_{(t+1)}$
0	0	Q_t
0	1	RESET
1	0	SET
1	1	X



فلیپ فلاپ D (D-F.F.) : این نوع فلیپ فلاپ به Delay Flip Flop معروف بوده و تنها دارای یک ورودی D می‌باشد. در صورت فعال بودن ورودی کنترل، هر مقداری در ورودی D وجود داشته باشد، در خروجی Q نیز ظاهر خواهد شد. نمودار مداری و دیاگرام بلوکی D-F.F. در شکل زیر مشاهده می‌شود.





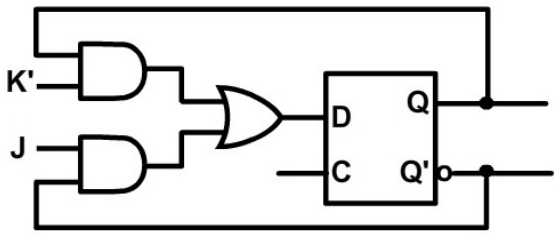
جدول مشخصه و معادله مشخصه D-F.F. نیز به صورت زیرند :

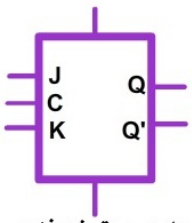
C	Q	D	$Q_{(t+1)}$
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

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

فلیپ فلاپ JK (JK-F.F.) : این فلیپ فلاپ دارای دو ورودی J و K است و کامل‌ترین نوع فلیپ فلاپ‌ها می‌باشد. زیرا می‌توان خروجی این فلیپ فلاپ را به 1 نشاند، به 0 بازنشانی کرد، متمم حالت قبلی نمود و یا بدون تغییر نگه‌داشت. فلیپ فلاپ JK را همان‌گونه که در شکل نیز مشاهده می‌شود، می‌توان با استفاده از D-F.F. و یا SR-Latch و گیت‌های AND طراحی نمود.

نمودار مداری و بلوک دیاگرام JK-F.F در شکل زیر مشاهده می شوند.





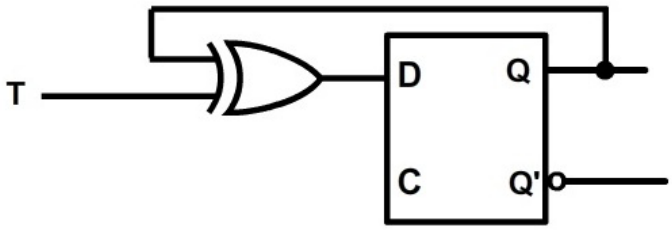
جدول مشخصه و معادله مشخصه JK-F.F به صورت زیرند :

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

J	K	$Q_{(t+1)}$
0	0	Q
0	1	0
1	0	1
1	1	Q'

$$Q_{(t+1)} = JQ' + K'Q$$

فلیپ فلاپ T (T-F.F.) : این نوع فلیپ فلاپ به فلیپ فلاپ متمم ساز معروف بوده و به راحتی می‌توان آن را با اتصال دو ورودی یک JK-F.F. به یکدیگر ساخت. ورودی این فلیپ فلاپ T نام دارد. اگر این ورودی برابر 0 باشد، حالت خروجی فلیپ فلاپ تغییر نخواهد کرد و اگر هم برابر 1 باشد، خروجی فلیپ فلاپ متمم خروجی فعلی آن خواهد شد. نمودار مداری T-F.F. به‌صورت زیر است :



جدول مشخصه و معادله مشخصه T-F.F. به صورت زیرند :

Q	T	$Q_{(t+1)}$
0	0	0
0	1	1
1	0	1
1	1	0

$$Q_{(t+1)} = T \oplus Q$$

انواع فلیپ فلاپ‌ها از نظر نحوه تحریک سیگنال ساعت :

تحریک شونده با سطح :

تحریک شونده با سطح مثبت (Positive Level Triggered) : وقتی سیگنال ساعت 1 می‌شود، ورودی‌ها فعال می‌شوند.

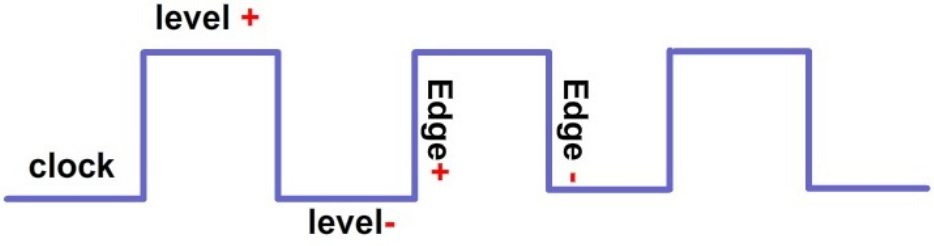
تحریک شونده با سطح منفی (Negative Level Triggered) : وقتی سیگنال ساعت 0 می‌شود، ورودی‌ها فعال می‌شوند.

تحریک شونده با لبه :

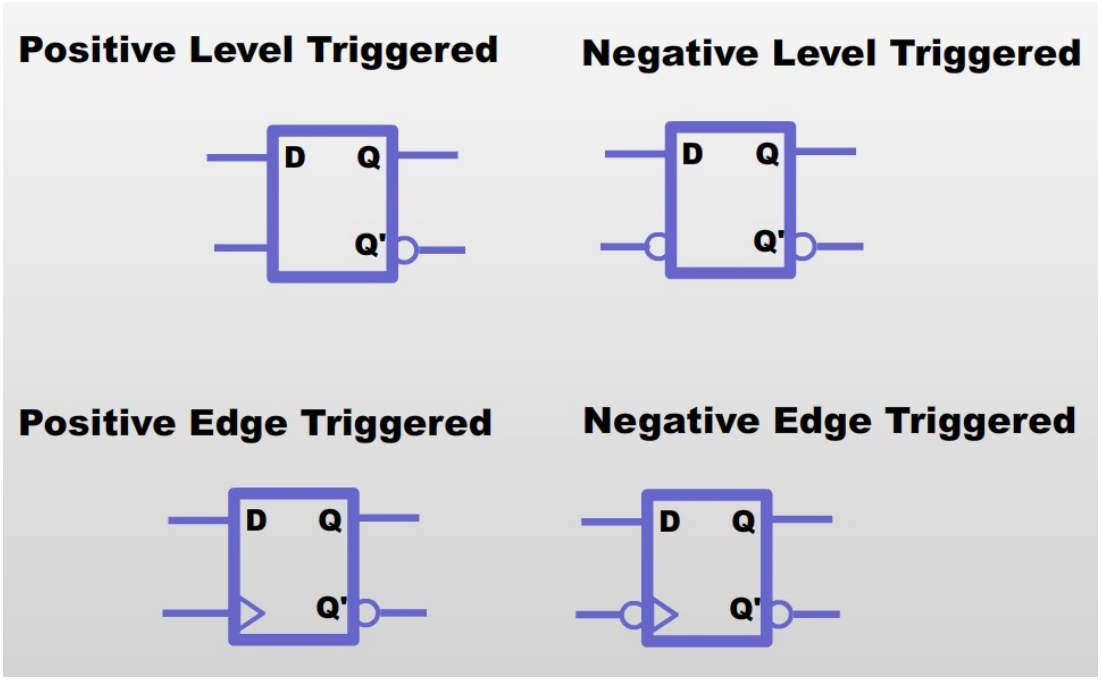
تحریک شونده با لبه مثبت (Positive Edge Triggered) : زمانی که سیگنال ساعت از 0 به 1 تغییر می‌کند، ورودی‌ها فعال می‌شوند.

تحریک شونده با لبه منفی (Negative Edge Triggered) : زمانی که سیگنال ساعت از 1 به 0 تغییر می‌کند، ورودی‌ها فعال می‌شوند.

انواع تحریک های ممکن برای فلیپ فلاپ ها بر روی سیگنال شکل زیر مشخص شده اند.

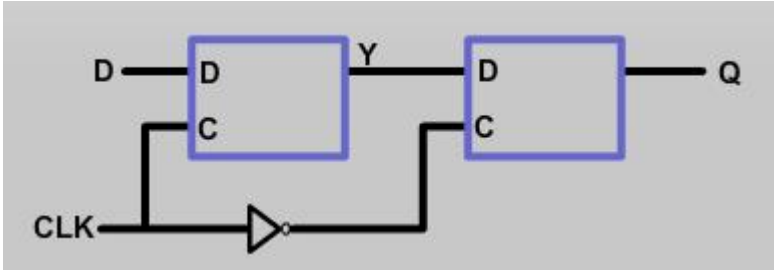


دیاگرام بلوکی انواع فلیپ فلاپ از نظر نحوه تاثیر سیگنال ساعت برای فلیپ فلاپ D در شکل زیر مشاهده می‌شوند. برای سایر فلیپ فلاپ‌ها نیز از سمبل‌های مشابه جهت مشخص نمودن نوع تحریک فلیپ فلاپ استفاده می‌شود.

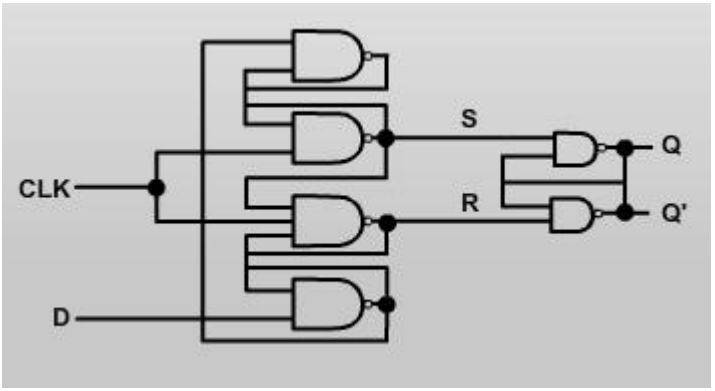


نکته : معمولاً در عمل از فلیپ فلاپ‌های تحریک شونده با لبه استفاده می شود، زیرا حساسیت این نوع فلیپ فلاپ‌ها در مقابل نویز کم‌تر بوده و احتمال بروز خطا پائین‌تر است. در مداراتی که از این نوع فلیپ فلاپ‌ها استفاده می‌شود، مشکل ناپایداری وجود نداشته و تمامی حالات مدار قابل پیش بینی خواهند بود.

فلیپ فلاپ حاکم - تابع (Master-Slave Flip Flop) : یکی از روش‌های ساخت فلیپ فلاپ‌های تحریک شونده با لبه، استفاده از پیکربندی Master-Slave مطابق شکل زیر می‌باشد. همان گونه که در شکل نیز مشاهده می‌شود، برای ساخت یک D-F.F. تحریک شونده با لبه از دو D-F.F. معمولی استفاده شده و سیگنال ساعت مستقیماً وارد یک فلیپ فلاپ شده و متمم آن وارد فلیپ فلاپ دیگر می‌شود. این نحوه اتصال سیگنال ساعت به فلیپ فلاپ‌ها، باعث می‌شود که تنها در لبه‌های پالس‌های ساعت، ورودی‌ها بتوانند حالت فلیپ فلاپ را تغییر دهند.



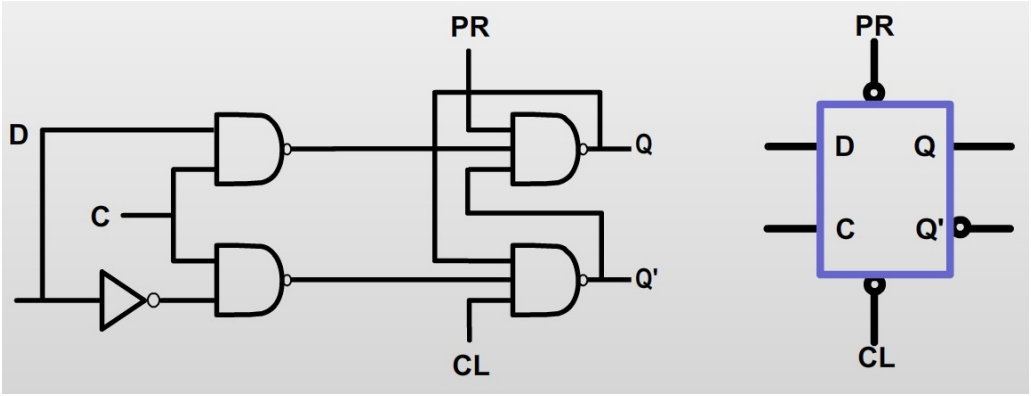
فلیپ فلاپ D تحریک شونده با لبه : روش دیگر ساخت یک D-F.F. تحریک شونده با لبه، اتصال ۳ عدد D-Latch به یکدیگر، مطابق شکل زیر می‌باشد. در این شکل یک D-F.F. تحریک شونده با لبه مثبت طراحی شده است.



در این مدار ورودی‌ها فقط با لبه بالا رونده یا مثبت سیگنال ساعت فعال می‌شوند و در زمانی که سیگنال ساعت یک یا صفر باشد فعالیتی وجود نداشته و حالت فلیپ فلاپ تغییر نخواهد کرد.
اشکال عمده طراحی فوق این است که در همه حالت‌ها، خروجی‌های Q و 'Q متمم یکدیگر نیستند.
جدول مشخصه D-F.F. تحریک شونده با لبه مثبت در زیر مشاهده می‌شود :

Clk	Q	D	Q(t+1)
^	0	0	0
^	0	1	1
^	1	0	0
^	1	1	1

ورودی‌های غیر همزمان فلیپ فلاپ‌ها : اکثر فلیپ فلاپ‌ها دارای ورودی‌های غیر همزمان برای وارد شدن به یک حالت خاص (Set یا Reset) می‌باشند. این ورودی‌ها برای بردن همه فلیپ فلاپ‌های سیستم به یک حالت آغازین معلوم (Set یا Reset)، قبل از اعمال پالس ساعت به کار می‌روند. این ورودی‌ها را ورودی‌های آسنکرون یا غیر همزمان می‌نامند، زیرا هماهنگ با پالس‌های ساعت عمل نکرده و هرگاه فعال شوند، حالت خروجی فلیپ فلاپ را تغییر خواهند داد.
اکثر فلیپ فلاپ‌ها معمولاً دو ورودی غیر همزمان Clear و Preset دارند. ورودی Clear مستقل از ورودی‌های فلیپ فلاپ و وضعیت پالس‌های ساعت، خروجی فلیپ فلاپ را 0 می‌کند. ورودی Preset نیز مستقل از ورودی‌های فلیپ فلاپ و وضعیت پالس‌های ساعت، خروجی فلیپ فلاپ را 1 خواهد کرد.
شکل زیر دیاگرام بلوکی و نمودار مداری یک D-F.F. با ورودی‌های غیر همزمان Clear و Preset را نشان می‌دهد.



در شکل فوق، اگر Clear = 0 شود، آنگاه Q = 0 و Q' = 1 خواهند شد. همچنین اگر Preset = 0 شود، آنگاه Q = 1 و Q' = 0 خواهند شد. توجه شود که ورودی‌های غیر همزمان معمولاً Active Low می‌باشند.
جدول مشخصه یک D-F.F. با ورودی‌های غیر همزمان در زیر مشاهده می‌شود :

CL	PR	Clk	D	Q(t+1)
0	1	X	X	0
1	0	X	X	1
1	1	0	X	Q
1	1	1	0	0
1	1	1	1	1

در این نوع فلیپ فلاپ‌ها باید توجه شود که هر دو ورودی Clear و Preset همزمان فعال نشوند، زیرا حالت مدار غیر قابل پیش بینی شده و باعث ناپایداری و بروز خطا در مدار خواهد شد.

تحلیل و طراحی مدارهای منطقی ترتیبی

هدف از این بخش آشنایی با مفاهیم زیر می‌باشد.

- نحوه تحلیل و آنالیز مدارهای منطقی ترتیبی
- مراحل طراحی مدارهای منطقی ترتیبی
- مفهوم کاهش حالت در مدارهای منطقی ترتیبی
- طراحی مدارهای ترتیبی شامل شمارنده‌ها، مدارهای ردیاب الگو و ...

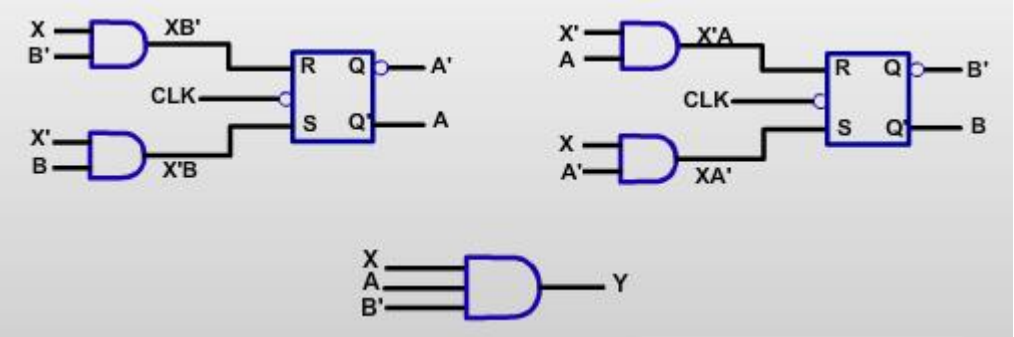
مدارهای ترتیبی ساعت دار : رفتار یک مدار ترتیبی ساعت‌دار با ورودی‌ها، خروجی‌ها و حالت فلیپ فلاپ‌ها مشخص می‌گردد. خروجی‌ها و حالت بعدی، هر دو تابعی از ورودی‌ها و حالت فعلی مدار هستند. رشته‌های زمانی ورودی‌ها، خروجی‌ها و حالات فلیپ فلاپ‌ها را می‌توان در جدولی با نام جدول حالت متشکل از چهار بخش حالت فعلی، ورودی، حالت بعدی و خروجی نمایش داد.

تحلیل مدارهای ترتیبی ساعت دار : تحلیل یک مدار ترتیبی به معنی تهیه جدول یا نموداری از رشته زمانی ورودی‌ها، خروجی‌ها و حالات فلیپ فلاپ‌ها است که با نمودار مداری و یا معادلات ورودی فلیپ فلاپ‌ها و خروجی‌های ترکیبی مدار آغاز شده و به رسم جدول حالت، نمودار حالت و یا تشریح عملکرد مدار منتهی می‌شود.

مراحل تحلیل و آنالیز یک مدار ترتیبی :

۱. تخصیص سمبل‌های حرفی به فلیپ فلاپ‌ها، و سپس مشخص نمودن متغیرهای حالت، ورودی‌ها و خروجی‌های مدار.
۲. به دست آوردن معادله حالت فلیپ فلاپ‌های مدار : معادله حالت فلیپ فلاپ را می‌توان با جایگذاری معادله ورودی (های) آن در معادله مشخصه‌اش به دست آورد.
۳. رسم جدول حالت مدار : برای به‌دست‌آوردن جدول حالت یک مدار ترتیبی، به دو روش می‌توان عمل نمود. یک روش این است که برای هر ترکیب از ورودی‌ها و حالات فعلی، با استفاده از معادله حالت هر فلیپ فلاپ، حالت بعدی آن را به دست آورد. روش دیگر، استفاده از جدول مشخصه هر فلیپ فلاپ در به دست آوردن حالت بعدی فلیپ فلاپ است.
۴. رسم نمودار حالت مدار : اطلاعات موجود در جدول حالت را می‌توان به‌صورت گرافیکی با نمودار حالت نمایش داد. در این نمودار، هر حالت با یک دایره نشان داده شده و تغییر بین حالت‌ها با خطوط جهت داری که دو دایره را به هم وصل می‌کنند، نمایش داده می‌شود. در روی خطوط، ورودی و خروجی را نوشته و با "/" از هم جدا می‌کنیم. توجه داریم که نمودار حالت، شکل ترسیمی جدول حالت است و برای تحلیل عملکرد مدار مناسب‌تر می‌باشد.
۵. تحلیل عملکرد مدار در صورت لزوم از روی جدول حالت و یا نمودار حالت.

مثال ۱ : جدول حالت و نمودار حالت مدار ترتیبی شکل زیر را رسم نمایید.



حل : این مدار دارای یک ورودی با نام X، یک خروجی ترکیبی با نام Y و دو فلیپ فلاپ SR است که حالت‌های آن‌ها را با A و B نمایش می‌دهیم. لذا جدول حالت از ۴ بخش حالت فعلی شامل A و B، ورودی شامل X، حالت بعدی شامل A و B و خروجی شامل Y تشکیل خواهد شد.

با توجه به شکل‌ها، معادلات ورودی فلیپ فلاپ‌ها و خروجی Y به شرح زیر به دست می‌آیند :

$R_A = B'X$ $S_A = BX'$ $R_B = AX'$ $S_B = A'X$ $Y = AB'X$

با جایگذاری معادلات فوق در معادله مشخصه SR-F.F.، معادلات حالت فلیپ فلاپ‌های A و B به‌صورت زیر به دست خواهند آمد :

$$Q_{(t+1)} = S + R'Q$$

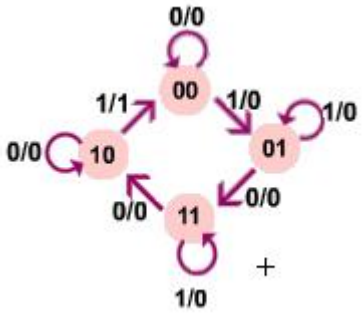
$$A_{(t+1)} = X'B + (XB')'A = X'B + X'A + AB$$
$$B_{(t+1)} = XA' + (X'A)'B = XA' + XB + A'B$$

حال با استفاده از معادلات حالت به دست آمده در بخش قبل و همچنین معادله Y، می‌توان جدول حالت مدار را رسم نمود. جدول حالت را به دو صورت زیر می‌توان رسم نمود. توجه شود که در جدول اول، ستون‌های حالت بعدی و خروجی مدار به ازای مقادیر مختلف ورودی، تقسیم بندی شده‌اند.

در انتها با استفاده از اطلاعات جدول حالت، نمودار حالت مدار را به‌صورت زیر رسم می‌کنیم :

حالت فعلی	ورودی	حالت بعدی	خروجی
AB	X	AB	Y
00	0	00	0
00	1	01	0
01	0	11	0
01	1	01	0
10	0	10	0
10	1	00	1
11	0	10	0
11	1	11	0

حالت فعلی	حالت بعدی		خروجی	
	X=0	X=1	X=0	X=1
AB	AB	AB	Y	Y
00	00	01	0	0
01	11	01	0	0
10	10	00	0	1
11	10	11	0	0



طراحی مدارهای ترتیبی : طراحی یک مدار ترتیبی ساعت دار با بیان مشخصات مسئله و مجموعه‌ای از ویژگی-های مدار مورد نظر آغاز شده و با رسم نمودار منطقی یا مجموعه‌ای از توابع بیان گر آن نمودار پایان می‌یابد. اولین قدم در طراحی مدارهای ترتیبی، تهیه جدول حالت یا نمایش معادلی از آن مانند نمودار حالت است.

مراحل طراحی یک مدار ترتیبی :

۱. به دست آوردن نمودار حالت مدار با توجه به مشخصات مسئله و عملکرد مورد نظر مدار
۲. در صورت لزوم، کاهش تعداد حالات
۳. تخصیص مقادیر دودویی به حالت‌ها
۴. رسم جدول حالت کد شده دودویی
۵. انتخاب نوع فلیپ فلاپ
۶. پرکردن بخش ورودی فلیپ‌فلاپ‌ها در جدول حالت با استفاده از جدول تحریک فلیپ فلاپ
۷. به دست آوردن معادلات ساده شده ورودی فلیپ فلاپ‌ها و معادلات خروجی (در صورت وجود)
۸. ترسیم نمودار منطقی مدار

تعریف ۱ : در زمان طراحی مدارهای ترتیبی، حالت فعلی و حالت بعدی مورد انتظار هر فلیپ فلاپ را با توجه به مشخصات مسئله مورد نظر می‌دانیم. می‌خواهیم بدانیم که چه ورودی‌هایی باید اعمال شوند تا گذر مورد نظر از حالت فعلی به حالت بعدی انجام شود. برای این منظور از جدول تحریک فلیپ فلاپ استفاده می شود که در آن برای هر گذر حالت فلیپ فلاپ، ورودی‌های لازم مشخص می‌باشند.

جدول تحریک همه فلیپ فلاپ ها در جدول زیر نشان داده شده است:

SR.FF

Q	Q(t+1)	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK.FF

Q	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

T.FF

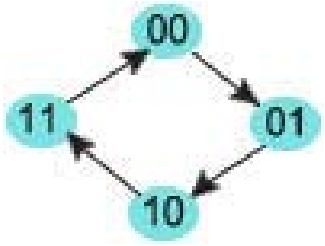
Q	Q(t+1)	T
0	0	0
0	1	1
1	0	1
1	1	0

D.FF

Q	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

مثال ۱ : با استفاده از فلیپ فلاپ SR، یک شمارنده دودویی ۲ بیتی طراحی نمایید که دائماً اعداد 0، 1، 2 و 3 را بشمارد.

حل : با توجه به مشخصات مدار مورد نظر، نمودار حالت مدار به شکل زیر خواهد بود :

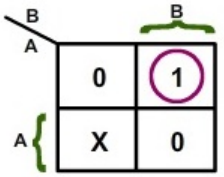


برای نمایش ۴ حالت می‌توان از ۲ عدد فلیپ فلاپ استفاده نمود که آن ها را A و B نام گذاری می‌کنیم. حال با توجه به‌صورت مسئله که طراحی با فلیپ فلاپ SR مورد نظر است، همچنین با توجه به جدول تحریک این نوع فلیپ فلاپ، جدول حالت مدار به شکل زیر به دست خواهد آمد :

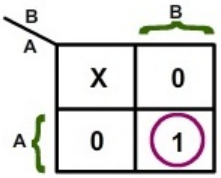
حالت فعلی		حالت بعدی		ورودی فلیپ فلاپ ها			
A	B	A(t+1)	B(t+1)	S_A	R_A	S_B	R_B
0	0	0	1	0	X	1	0
0	1	1	0	1	0	0	1
1	0	1	1	X	0	1	0
1	1	0	0	0	1	0	1

توضیح اضافه : جهت یادگیری نحوه پرکردن ستون ورودی فلیپ فلاپ‌ها با استفاده از جدول تحریک، این روال را برای سطر اول جدول فوق شرح می‌دهیم :
در سطر اول، حالت فعلی A برابر 0 و حالت بعدی آن برابر 0 است. بنابراین $S_A = 0$ و $R_A = X$ باعث این گذر حالت خواهند شد. همچنین در این سطر، حالت فعلی B برابر 0 و حالت بعدی آن برابر 1 است. بنابراین $S_B = 1$ و $R_B = 0$ باعث این گذر حالت خواهند شد.

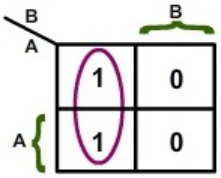
حال با استفاده از جداول کارنو، معادلات ساده شده ورودی‌های فلیپ فلاپ‌ها را به دست می‌آوریم :



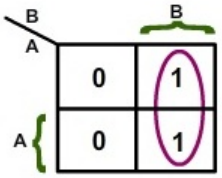
$S_A = A'B$



$R_A = AB$

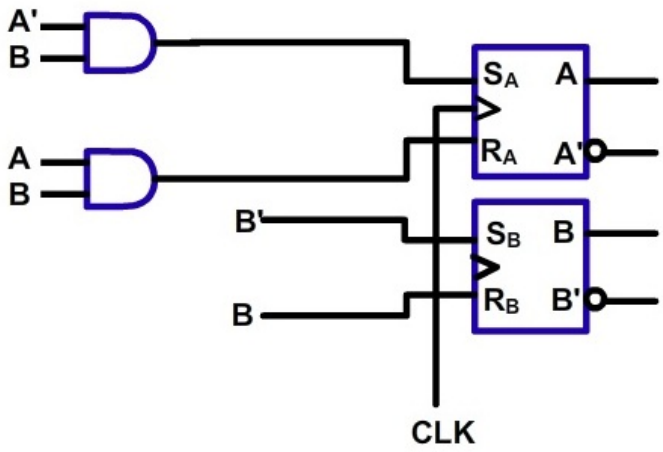


$S_B = B'$



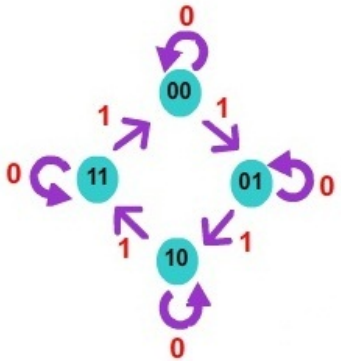
$R_B = B$

در مرحله آخر با توجه به معادلات ورودی فلیپ فلاپ‌ها، نمودار مداری را رسم می‌کنیم.



مثال ۲ : با استفاده از فلیپ فلاپ JK، یک شمارنده دودویی ۲ بیتی با یک ورودی X طراحی نمایید به نحوی که اگر ورودی X صفر بود، شمارشی صورت نگیرد ومدار در همان حالت باقی بماند و زمانی که X برابر یک بود، شمارش به‌صورت صعودی انجام شود.

حل : با توجه به مشخصات مدار مورد نظر، نمودار حالت مدار به شکل زیر خواهد بود :

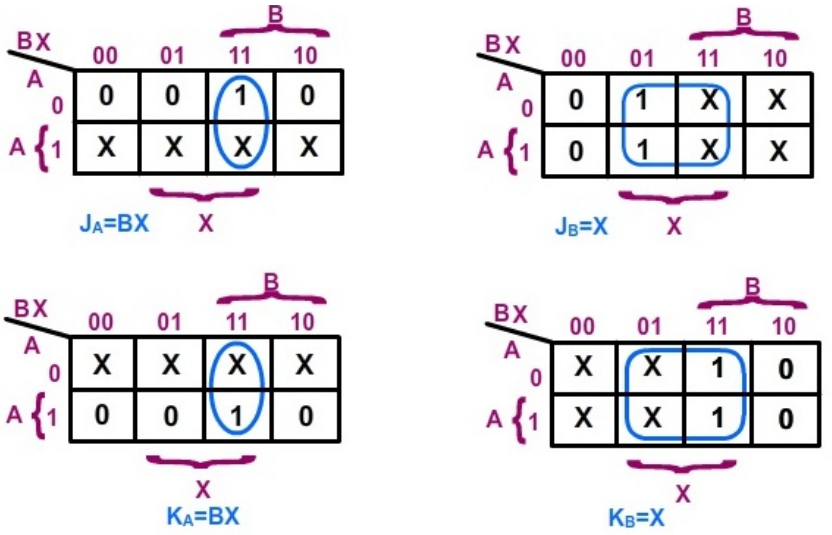


با توجه به صورت مسئله که طراحی با فلیپ فلاپ JK مورد نظر است، همچنین با توجه به جدول تحریک این نوع فلیپ فلاپ، جدول حالت مدار به شکل زیر به دست خواهد آمد :

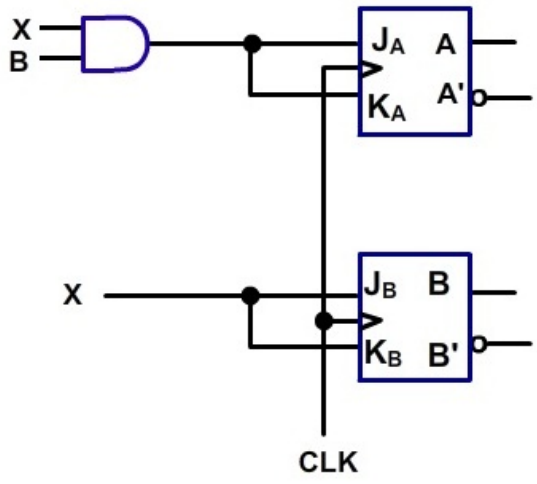
حالت فعلی		ورودی	حالت بعدی		ورودی فلیپ فلاپ ها			
A	B	X	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	0	1	0	X	X	0
0	1	1	1	0	1	X	X	1
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

توضیح اضافه : جهت یادگیری نحوه پرکردن ستون ورودی فلیپ فلاپ‌ها با استفاده از جدول تحریک، این روال را برای سطرهای اول و دوم جدول فوق شرح می‌دهیم :
در سطر اول، حالت فعلی A برابر 0 و حالت بعدی آن برابر 0 است، بنابراین $J_A = 0$ و $K_A = X$ باعث این گذر حالت خواهند شد. همچنین در سطر دوم، حالت فعلی B برابر 0 و حالت بعدی آن برابر 1 است، بنابراین $J_B = 1$ و $K_B = X$ باعث این گذر حالت خواهند شد.

حال با استفاده از جداول کارنو، معادلات ساده شده ورودی‌های فلیپ فلاپ‌ها را به دست می‌آوریم :

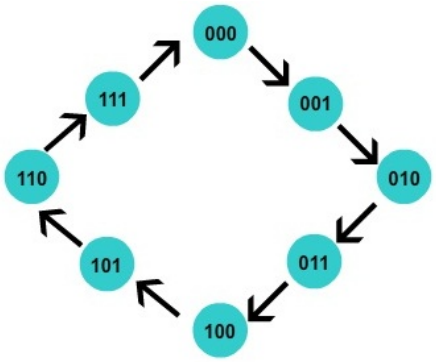


در مرحله آخر با توجه به معادلات ورودی فلیپ فلاپ‌ها، نمودار مداری را رسم می‌کنیم.



مثال ۳ : با استفاده از فلیپ فلاپ T، یک شمارنده دودویی ۳ بیتی طراحی نمایید که دائماً اعداد 0، 1، تا 7 را بشمارد.

حل : با توجه به مشخصات مدار مورد نظر، نمودار حالت مدار به شکل زیر خواهد بود :



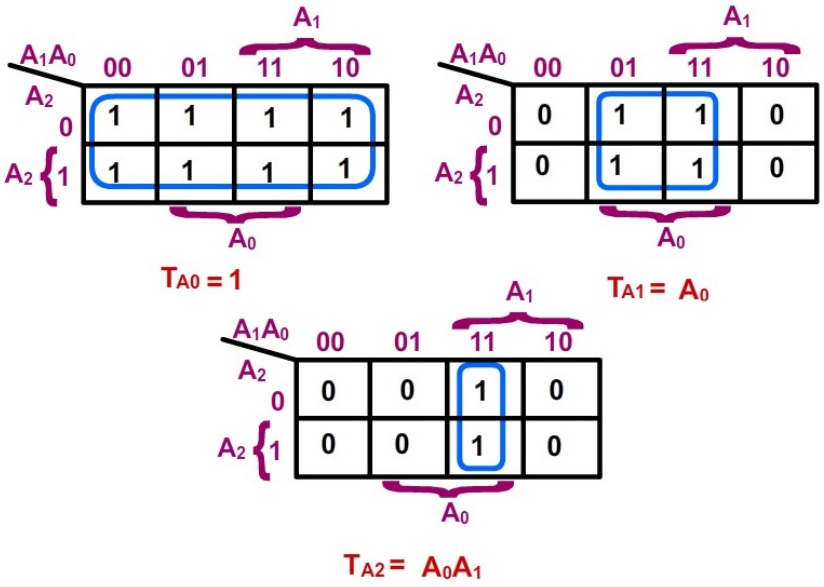
با توجه به‌صورت مسئله که طراحی با فلیپ فلاپ T مورد نظر است، همچنین با توجه به جدول تحریک این نوع فلیپ فلاپ، جدول حالت مدار به شکل زیر به دست خواهد آمد :

حالت فعلی			حالت بعدی			ورودی فلیپ فلاپ ها		
A2	A1	A0	A2	A1	A0	T_{A2}	T_{A1}	T_{A0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

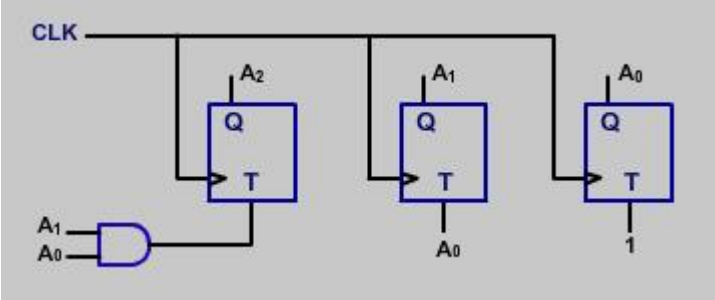
توضیح اضافه : جهت یادگیری نحوه پرکردن ستون ورودی فلیپ فلاپ‌ها با استفاده از جدول تحریک، این روال را برای سطر اول جدول فوق شرح می‌دهیم :

در سطر اول، حالت فعلی A_2 و A_1 برابر 0 و حالت بعدی آن ها برابر با 0 هستند. پس $T_{A2} = 0$ و $T_{A1} = 0$ باعث این گذر حالات خواهند شد. همچنین در این سطر، حالت فعلی A_0 برابر 0 و حالت بعدی آن برابر 1 است، بنابراین $T_{A0} = 1$ باعث این گذر حالت خواهد شد.

حال با استفاده از جداول کارنو، معادلات ساده شده ورودی‌های فلیپ فلاپ‌ها را به دست می‌آوریم :



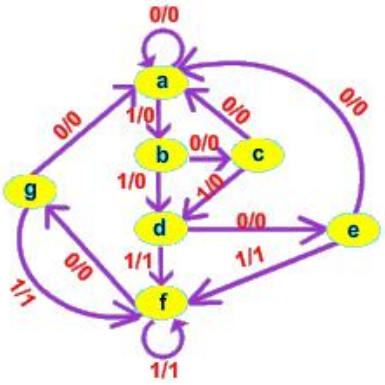
در مرحله آخر با توجه به معادلات ورودی فلیپ فلاپ‌ها، نمودار مداری را رسم می‌کنیم.



کاهش حالت در مدارهای ترتیبی : کاهش تعداد فلیپ فلاپ‌ها در یک مدار ترتیبی را کاهش حالت می‌نامند. البته لازم به ذکر است که کاهش حالت گاهی اوقات ممکن است به کاهش تعداد فلیپ فلاپ‌ها منجر گردد. الگوریتم‌های کاهش حالت باید به نحوی باشند که رابطه ورودی- خروجی مدار را حفظ نمایند. در این حالت دو مدار را معادل می‌نامیم اگر به هر دو مدار ورودی یکسانی اعمال شود و خروجی‌های مشابهی برای تمام رشته‌های ورودی تولید گردد.

حالات معادل : دو حالت را معادل گوییم اگر به هر دو حالت، ورودی‌های یکسانی اعمال شود و خروجی‌های مشابهی برای تمام ورودی‌ها تولید کنند و مدار را به حالت‌های معادل ببرند.

مثال : الگوریتم کاهش حالت را بر روی نمودار حالت زیر اعمال نموده و سپس، مدار کاهش یافته را با فلیپ فلاپ D طراحی نمایید.



جدول حالت اولیه :

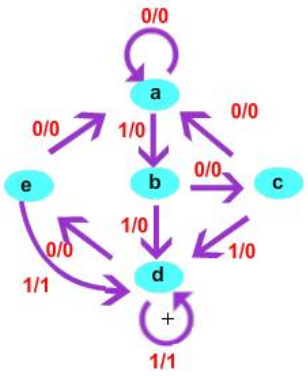
حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g = e g	a	f	0	1

جدول حالت کاهش یافته :

حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f = d f	e	f	0	1

حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

نمودار حالت کاهش یافته :



تخصیص مقادیر دودویی به حالت‌ها : حداقل تعداد بیت‌های لازم برای نمایش ۵ حالت، ۳ بیت است. لذا از بین ۸ ترکیب بیتی مختلف، می‌توان ۵ ترکیب را به دلخواه به حالت‌های a تا f اختصاص داد که سه ترکیب بی استفاده نیز خواهیم داشت. در این مثال، تخصیص مقادیر را به‌صورت زیر انجام داده‌ایم :

a= 001	b= 010	c=011	d=100	e= 101

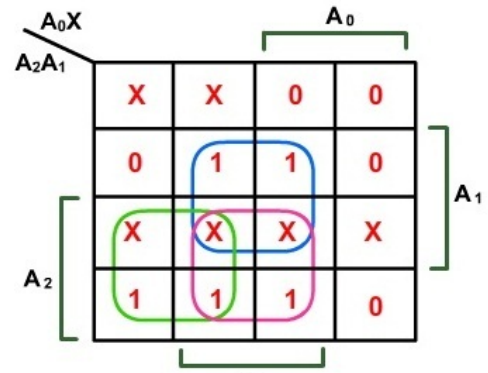
جدول حالت کد شده دودویی :

حالت فعلی	حالت بعدی		خروجی	
	x = 0	x = 1	x = 0	x = 1
a 001	001	010	0	0
b 010	011	100	0	0
c 011	001	100	0	0
d 100	101	100	0	1
e 101	001	100	0	1

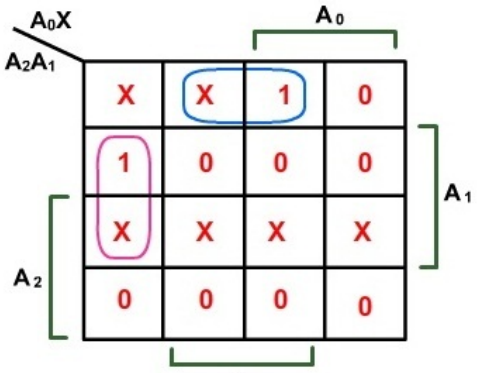
حالت فعلی			ورودی	حالت بعدی			خروجی
A_2	A_1	A_0	X	A_2	A_1	A_0	y
0	0	1	0	0	0	1	0
0	0	1	1	0	1	0	0
0	1	0	0	0	1	1	0
0	1	0	1	1	0	0	0
0	1	1	0	0	0	1	0
0	1	1	1	1	0	0	0
1	0	0	0	1	0	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	1	0
1	0	1	1	1	0	0	1

برای طراحی این مدار از فلیپ فلاپ‌های D استفاده خواهد شد. بنابراین نیازی به رسم ستون ورودی فلیپ فلاپ-ها و استفاده از جدول تحریک فلیپ فلاپ D نیست، زیرا حالت بعدی و ورودی فلیپ فلاپ D با هم برابرند.

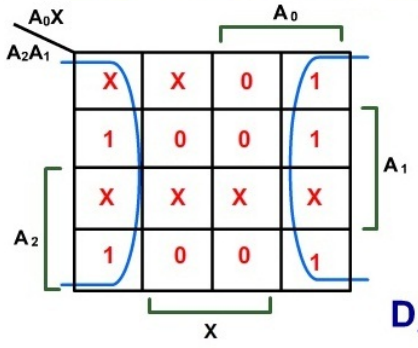
با استفاده از جداول کارنو، معادلات ساده شده ورودی‌های فلیپ فلاپ‌ها و معادله خروجی مدار را به دست می-آوریم؛ توجه داریم که در این مثال، به سه فلیپ فلاپ نوع D نیاز است که با نام‌های A0، A1 و A2 نام‌گذاری شده‌اند.



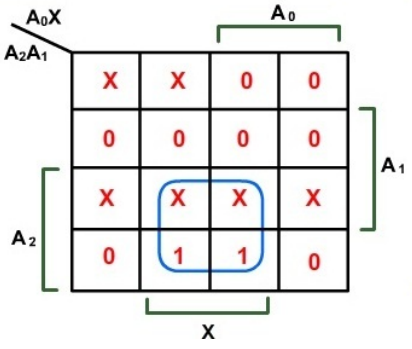
$$D_{A2} = A_1X + A_2A_0' + A_2X$$



$$D_{A1} = A_2'A_1'X + A_1A_0'X'$$



$$D_{A0} = X'$$



$$Y = A_2.X$$

انواع مدارهای منطقی ترتیبی :

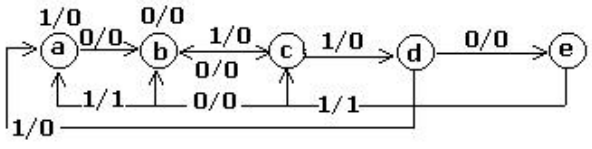
ماشین مدل Moore : در این مدارات، خروجی فقط تابعی از متغیرهای حالت می‌باشد.

ماشین مدل Mealy : در این مدارات، خروجی تابعی از متغیرهای حالت و ورودی‌ها می‌باشد.

مثال : مداری طراحی کنید که وقتی در ورودی آن، ترکیب بیتی زیر دریافت شد، خروجی آن به مدت یک کلاک یک بشود، در غیر این صورت خروجی صفر باشد. مدار را با هر دو مدل Moore و Mealy طراحی نمایید.

01101

مدل Mealy :



قبلی	بعدی		خروجی	
a	b	a	0	0
b	b	c	0	0
c	b	d	0	0
d	e	a	0	0
e	d	a	0	1

a = 00001101101000111100

حالت = abbbbcdeaabcbbbcdaa

مدل Moore :

	x = 0	x = 1	
a	b	a	0
b	b	c	0
c	b	d	0
d	e	a	0
e	b	e	0
f	b	a	1