«مدار منطقی»

A_{10} decimal $= \{0,1,2,\dots,9\}$ < مبنا ۱۰

B_{2} Binary $= \{0,1\}$ < مبنا ۲

C_{8} octal $= \{0,1,2,\dots,7\}$ < مبنا ۸

D_{16} Hexadecimal $= \{0,1,\dots,9,\overbrace{A,B,C,D,E,F}^{\text{هنگام نمایش}}\}$ < مبنا ۱۶

$\underbrace{10\ \ 11\ \ 12\ \ 13\ \ 14\ \ 15}_{\text{هنگام محاسبه}}$

فرم خلاصه اعداد $a_n a_{n-1} a_{n-2}, \dots a_2 a_1 a_0 / a_{-1} a_{-2} a_{-3} \dots$

$$= a_n r^n + a_{n-1} r^{n-1} + a_{n-2} r^{n-2} \dots + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} + \dots$$

فرم گسترده نویسی اعداد

a : ضریب؛ عنصری از مبنا

r : مبنا

n : ستون مربوط به عدد با شروع سمت چپ ممیز اعشار با شماره صفر و افزایش به سمت چپ و کاهش سمت راست.

در هر مبنایی به تعداد مبنا عضو داریم که بزرگترین عضو مبنا از خود مبنا کوچکتر است و

Date :

با ترکیب اعضای مبنا می‌توان حالات مبنا ایجاد نمود. برای محاسبه مقدار عدد در مبنای ۱۰ از فرمول فوق استفاده می‌شود. در این فرمول a ضریب عضوی از مبنا، r مبنا و n ستون می‌باشد که در سمت چپ ممیز اعشار با مقدار صفر شروع شده افزایش می‌یابد و در سمت راست کاهش می‌یابد.

مثال: ٤٠٠٢٥١,٠٢٠١

گسترده $\rightarrow (4\times10^{5}) + (0\times10^{4}) + (0\times10^{3}) + (2\times10^{2}) + (5\times10^{1}) + (1\times10^{0})$

$+ (0\times10^{-1}) + (2\times10^{-2}) + (0\times10^{-3}) + (1\times10^{-4})$

$400000 + 0 + 0 + 200 + 50 + 1 + 0 + \frac{2}{10} + 0 + \frac{1}{100}$

تبدیل مبنا

$(93)_{10} \rightarrow (\quad)_2$

$(\quad)_8$

$(\quad)_{16}$

10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1
			9	3

$93 \div 2 = 46$, باقیمانده 1

$46 \div 2 = 23$, باقیمانده 0

$23 \div 2 = 11$, باقیمانده 1

$11 \div 2 = 5$, باقیمانده 1

$5 \div 2 = 2$, باقیمانده 1

$2 \div 2 = 1$, باقیمانده 0

$1 \div 2 = 0$, باقیمانده 1

* خارج قسمت صفر

$۲^{۷}$	$۲^{۶}$	$۲^{۵}$	$۲^{۴}$	$۲^{۳}$	$۲^{۲}$	$۲^{۱}$	$۲^{۰}$	مبنای ۲
۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱	
	۱	۰	۱	۱	۱	۰	۱	

$۸^{۲}$	$۸^{۱}$	$۸^{۰}$	مبنای ۸
۶۴	۸	۱	
۱	۳	۵	

$۱۶^{۲}$	$۱۶^{۱}$	$۱۶^{۰}$	مبنای ۱۶
۲۵۶	۱۶	۱	
	۵	D	
		۱۳	

برای تبدیل عدد از مبنای ۱۰ به دیگر مبناها کافیست که عدد در مبنای ۱۰ را به طور متوالی بر مبنای جدید تقسیم نمایید. باقی مانده را نگه داشته و خارج قسمت را مجدداً تقسیم می‌کنیم تا خارج قسمت صفر شود. باقی مانده ها را از ابتدا تا انتها می‌نویسیم.

روش دوم این است که عدد مبنای ۱۰ را به توان هایی از مبنای جدید یا ضرایب مناسب تقسیم کنیم. برای تبدیل از دیگر مبناها به مبنای ۱۰ از روش قبل استفاده می‌شود.

جمع مبناها :

$$\begin{array}{r} 11 \\ ۳۹٤\ (10) \\ +\ ۳۸۹\ (10) \\ \hline ۷۸۳ \end{array} \qquad \begin{array}{r} 11 \\ ۲۳۶\ (8) \\ +\ ٤۵۵\ (8) \\ \hline ۷۱۳\ (8) \end{array} \qquad \begin{array}{r} 11\ 111 \\ 1010011 \\ +\ 1011111 \\ \hline 10110110 \end{array} \qquad \begin{array}{r} 111 \\ E59C\ (H) \\ +\ 9BCD\ (H) \\ \hline 18169\ H \end{array}$$

در جمع مبناها : هنگامی که حاصل جمع ستونی از مبنا بزرگتر یا مساوی گردد به اندازه مبنا از آن کم شده و یک واحد به ستون بالاتر اضافه می‌شود.

تفریق مبناها :

$$\begin{array}{r} ۳۹٤\ (10) \\ -\ ۳۸۹\ (10) \\ \hline 00۵ \end{array} \qquad \begin{array}{r} ۲۳۶\ (8) \\ -\ ٤۵۵\ (8) \\ \hline 0۵٤ \end{array} \qquad \begin{array}{r} 1010011\ (2) \\ -\ 1011111\ (2) \\ \hline 0010100 \end{array} \qquad \begin{array}{r} E59C\ (H) \\ -\ 9BCD\ (H) \\ \hline 49CF \end{array}$$

هنگام تفریق مبناها در صورتی که عدد دوم از عدد اول بزرگتر باشد نیاز به قرض می‌باشد و این قرض از ستون بالاتر دریافت می‌شود. یک واحد از ستون بالاتر کم شده و به اندازه ی مبنا به ستون کم ارزش اضافه می‌شود.

ضرب مبناها :

$$\begin{array}{r} 101101 \\ \times\ 10110\ (2) \\ \hline 01101010 \\ 01011010 \\ 1011010000 \\ \hline 1111011110 \end{array} \qquad \begin{array}{r} ۵11 \\ -\ ۳۳۰ \\ \hline \\ +\ ۵1۲ \\ \hline ۹۹۰ \end{array} \qquad \begin{array}{r} ٤۵ \\ \times\ ۲۷\ (8) \\ \hline ٤۰۳ \\ 11\ ۲۰ \\ \hline 1۵۲۳ \end{array}$$

جمع عدد زیر صرفا

٦٤	۸	1
	٤	۳
	٤	۵
	٤	۲
	1	1

در حاصل ضرب مبنای ۲ با توجه به اینکه عدد دوم دارای ۰ و ۱ می باشد به ازای صفرها جمله ضرب صفر می گردد و به ازای یک ها عدد اول نوشته می شود. برای جملات بعدی باید به تعداد ارقامی که در جلوی عدد دوم قرار دارد ابتدا صفر نوشته شود و سپس حاصلضرب اضافه شود.

مکمل ها

تعریف:

- مکمل مبنای کاهش یافته: مکمل ۹ (در مبنای ۱۰)، مکمل ۱ (در مبنای ۲)
- مکمل مبنا: مکمل ۱۰، مکمل ۲

$(r^k - 1) - N$ (k: تعداد ارقام)

$r^k - N$ (N: عدد، r: مبنا)

مکمل مبنای کاهش یافته (مکمل ۹) اعداد زیر را حساب کنید.

۴۰۵	۳۰۷۱	۴۹۵۲۱	۶۷۹۹۵۴۳۲۷
$(10^3-1)-405$	$(10^4-1)-3071$	۹۹۹۹۹	کافیست تک تک ارقام را از ۹ کم کنیم.
۹۹۹	۹۹۹۹	۴۹۵۲۱ -	
۴۰۵ -	۳۰۷۱ -	۵۰۴۷۸	
۵۹۴	۶۹۲۸		۳۲۰۰۴۵۶۷۲

برای محاسبه مکمل مبنای کاهش یافته کافیست تک تک ارقام عدد را از مبنای کاهش یافته کم کنید.

(برای محاسبه مکمل ۹ تک تک ارقام را از ۹ کم می‌نماییم و برای محاسبه مکمل ۱ تک تک ارقام از ۱ کم می‌شود.)

برای محاسبه مکمل مبنا به دو روش می‌توان اقدام نمود:

۱) از طریق فرمول ← بدین شکل در صورت وجود تعدادی صفر در سمت راست عدد صفرها را نوشته اولین رقم را از ۱۰ و بقیه ارقام را از ۹ کم می‌کنیم.

۲) روش دوم ← کافیست مکمل کاهش یافته عدد را محاسبه نموده سپس با یک جمع کنیم.

مکمل ۲ اعداد زیر را محاسبه کنید

۱۰۱۱	۱۰۱۰۱	۱۱۰ ۱۰۱۱۰۰۰	۱۰ ۱۱۱۱ ۰۰۰۰ ۱۱۰۱
$۲^۴$ − ۱۰۱۱	$۲^۵$ − ۱۰۱۰۱	مکمل ۲	
۱۰۰۰۰	۲۰۰۰۰۰	۰۰۱۰۱۰۱۰۰۰	۰۱۰۰۰۰۱۱۱۱۰۰۱۱
− ۱۰۱۱	− ۱۰۱۰۱		
۰۱۰۱	۰۱۰۱۱		

روش دوم ← مکمل ۱ +

۰۱۰۰	۰۱۰۱۰	۰۰۱۰۱۰۰۱۱۱
+ ۱	+ ۱	+ ۱
۰۱۰۱	۱۰۱۰۱۱	۰۰۱۰۱۰۱۰۰۰

Date :

برای محاسبه مکمل ۲ درصورت وجود تعدادی صفر در سمت راست عدد صفرها را نوشته

اولین یک را نیز می‌نویسیم بقیه بیت ها را مکمل می‌کنیم.
NOT می‌کنیم

مکمل ۱

مکمل ۱ اعداد زیر را محاسبه کنید.

1011	10101	1101011	1011110000 11.1
$(2^4-1)-1011$	$(2^5-1)-10101$	1111111	0100001111 00.0
1111	11111	− 1101011	
− 1011	− 10101	0010100	
100	01010		

جمع بری

$\begin{array}{r}1\\-1\\\hline 0\end{array}$ $\begin{array}{r}1\\-0\\\hline 1\end{array}$

برای محاسبه مکمل یک کافیست تک تک بیت ها را Not (تفضیل) کنیم یعنی اگر صفر است

به یک و اگر یک باشد به صفر تبدیل کنیم.

مکمل مبنا :

مکمل ۱۰ اعداد زیر را حساب کنید.

٤٠٥	٣٠٧١	٤٩٥٢١	٦٧٩٩٥٤٣٢٢٧
$١٠^٣ - ٤٠٥$	$١٠^٤ - ٣٠٧١$	١٠٠٠٠٠	٣٢٠٠٤٥٦٧٧٣
١٠٠٠	١٠٠٠٠	− ٤٩٥٢١	
− ٤٠٥	− ٣٠٧١	٥٠٤٧٩	
٥٩٥	٦٩٢٩		

مکمل ۱۰ = مکمل ۹ + ۱ ← روش ۲

٥٩٤	٦٩٢٨
+ ١	+ ١
٥٩٥	٦٩٢٩

اعداد علامت دار ۵

اعداد مثبت و منفی

$\underset{\text{علامت}}{\underline{1}}$ 0011001

۱ روش مقدار علامت

۲ روش مکمل ۱

۳ روش مکمل ۲ (روش مورد استفاده)

$-25 = \underset{\text{علامت}}{\underline{1}}\ 11..110$

$-26 =$

$\underline{1}\ 11..111$

$+25 = \underset{\text{علامت}}{\underline{0}}\ \underset{\text{مقدار}}{\underline{011001}}$

بیت علامت = با ارزش ترین بیت = سمت چپ ترین بیت

Sign out

عدد مثبت ← ۰

عدد منفی ← ۱

$10101000 \xrightarrow{\text{بیت علامت ۱ است}}$ عدد منفی است ← مکمل ۲ ← $-(01011000) = -88$

$01000001 \xrightarrow{\text{بیت علامت صفر است}}$ عدد مثبت ← $+65$

$01011101 \xrightarrow{\text{بیت علامت صفر است}}$ $+93$

$10101111 \xrightarrow{\text{عدد منفی}}$ مکمل ۲ ← $-(01010001) = -81$

روش مکمل ۲ $10111110 \quad -(01000010) \rightarrow -128 + 32 + 16 + 8 + 4 + 2 = -66$

اعداد علامت دار به صورت عدد مثبت و منفی وجود دارد و با ارزش ترین بیت به عنوان

بیت علامت تعریف می‌شود و در صورتی که این بیت صفر باشد عدد مثبت

و اگر یک باشد منفی در نظر گرفته می‌شود.

برای اعداد منفی به منظور محاسبه مقدار آنها ابتدا باید از عدد مکمل ۲ گرفته شده

و سپس مقدار آنها محاسبه شود. زیرا برای ذخیره اعداد منفی در کامپیوتر مکمل ۲

عدد مثبت را به عنوان عدد منفی نگهداری می‌کند.

عدد منفی

$10101101 \xrightarrow{\text{مکمل ۲}} -(01010011) = -۸۳$

۱۲۸ ۶۴ ۳۲ ۱۶ ۸ ۴ ۲ ۱

$10101101 \longrightarrow -۱۲۸ + (۳۲ + ۸ + ۴ + ۱) = -۱۲۸ + ۴۵ = -۸۳$

۶۴ ۳۲ ۱۶ ۸ ۴ ۲ ۱

$01010110 = +۶۴ + ۱۶ + ۴ + ۲ = +۸۶$

+

ASCII Code — کد اسکی

American standard Code for information interchange

کد استاندارد آمریکایی برای تبادل اطلاعات

A-Z : ۲۶

a-z : ۲۶

0-9 : ۱۰ → ۱۰۲ کد لازم است

@ / ؟ } ۴۰

تعداد ستون‌ها ← n

حالت $۲ = ۲^۱$

bit: 0 , 1

تعداد حالات = ۲ ← مبنا

کد اسکی توسعه یافته

Date :

کد اسکی حروف

	٦٤	٣٢	١٦	٨	۴	٢	١
A = ٦٥	0 1	0	0	0	0	0	1
Z = ٩٠	0 1	0	1	1	0	1	0
a = ٩٧	0 1	1	0	0	0	0	1
z = ١٢٢	0 1	1	1	1	0	1	0
0 = ٤٨	0 0	1	1	0	0	0	0
9 = ٥٧	0 0	1	1	1	0	0	1

٨ bit = 1 byte = 1 character حرف

١٦ bit = ٢ byte = 1 world کلمه

٣٢ bit = ٤ byte = double world کلمه مضاعف

$٢^{١٠}$ byte = ١٠٢٤ byte = 1 KB

$٢^{١٠} \times ٢^{١٠}$ byte = $٢^{٢٠}$ byte = 1 MB

$٢^{٣٠}$ byte = 1 GB

$٢^{٤٠}$ byte = 1 TB

Date :

مثال:

$۳۲\,GB = ۲^{۵} \times ۲^{۳۰}\ byte = ۲^{۳۵}\ byte$

$۲^{۳۳}\ byte = ۲^{۳۰} \times ۲^{۳}\ byte = ۸\,GB$

۱۲۸	۶۴	۳۲	۱۶	۸	۴	۲	۱

مثال اعداد ۸ بیتی

سازه اعداد

اعداد بدون علامت

۰۰۰۰۰۰۰۰ = ۰

۱۱۱۱۱۱۱۱ = ۲۵۵

تعداد اعداد تولید شده = ۲۵۶ = $۲^{۸}$ (۰ ⟵ ۲۵۵)

اعداد علامت دار

اعداد مثبت

۱۲۸ عدد { ۰ ۰۰۰۰۰۰۰ = +۰ ، ۰ ۱۱۱۱۱۱۱ = +۱۲۷

اعداد منفی

۱۲۸ عدد منفی { ۱ ۰۰۰۰۰۰۰ = −۱۲۸ ، ۱۱۱۱۱۱۱۱ = −۱

*

* = که جمعاً ۲۵۶ عدد نیمی در مثبت و نیمی در منفی

علامتدار (−۱۲۸ ⟵ +۱۲۷)

بدون علامت (۰ ⟵ ۲۵۵)

جمع و تفریق اعداد علامت دار

$+A + B$

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

$-A + B$

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

مکمل ۲ B

Date :

۱۲۸ ٦٤ ۳۲ ۱٦ ۸ ٤ ۲ ۱

A = ۵۵
- +۵۵ = 0 0 1 1 0 1 1 1
- −۵۵ = 1 1 0 0 1 0 0 1

B = ٤۵
- +٤۵ = 0 0 1 0 1 1 0 1
- −٤۵ = 1 1 0 1 0 0 1 1

+٤۵ + ۵۵ = +۱۰۰	+٤۵ − ۵۵ = −۱۰	−٤۵ + ۵۵ = +۱۰	−٤۵ − ۵۵ = −۱۰۰
00101101	00101101	11010011	11010011
+ 00110111	+ 11001001	+ 00110111	+ 11001001
$\boxed{01100100}$	$\boxed{11110110}$	1 $\boxed{00001010}$	1 $\boxed{10011100}$
→ +۱۰۰	→ −۱۲۸+۱۱۸=−۱۰	→ +۱۰	→ −۱۲۸ +۲۸ = −۱۰۰

overFlow — جمع و تفریق اعداد سرریز

+۷۰ = 0 1 0 0 0 1 1 0

−۷۰ = 1 0 1 1 1 0 1 0

+۸۰ = 0 1 0 1 0 0 0 0

−۸۰ = 1 0 1 1 0 0 0 0

+۸۰ +۷۰ = +۱۵۰ سرریز داریم	−۸۰ +۷۰ = −۱۰ سرریز نداریم	−۷۰ +۸۰ = +۱۰ سرریز نداریم	−۷۰ −۸۰ = −۱۵۰ سرریز داریم
01000110	01000110	10111010	10111010
+ 01010000	+ 10110000	+ 01010000	+ 10110000
10010110	11110110	100001010	101101010
$C_y = 1$	$C_y = 0$	$C_y = 1$	$C_y = 0$
$C_v = 0$	$C_v = 0$	$C_v = 1$	$C_v = 1$
سرریز داریم جواب غلط	جواب درست	جواب درست	جواب غلط

به منظور تشخیص سرریز در مبنای ۲ بدین صورت عمل می‌کنیم که در صورتیکه نقلی ورودی به طبقه آخر و نقلی خروجی از طبقه آخر برابر نباشد بیانگر این است که سرریز رخ داده است.

C_7	C_6	of سرریز
0	0	0
1	0	1
1	1	0
0	1	1

رفع مشکل سرریز :

+70	0000000001000110
+80	0000000001010000
+150	0000000010010110

سرریز داریم

علامت

۱۲۸ ، ۱۶ ، ۴ ۲

درست ⟹ +150

کد باینری				کد آیکن							
8	4	2	1	2	4	2	1	8	-4	-2	-1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	1	1	1
0	0	1	0	0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	1	0	1	0	1
0	1	0	0	0	1	0	0	0	1	0	0
0	1	0	1	1	0	1	1	1	0	1	1
0	1	1	0	1	1	0	0	1	0	1	0
0	1	1	1	1	1	0	1	1	0	0	1
1	0	0	0	1	1	1	0	1	0	0	0
1	0	0	1	1	1	1	1	1	1	1	1

کدهای دودویی

(سه افزا) کد افزونی ۳ → binary +3

Date :

0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0

کدگری (کد انعکاسی):

گری دوبیتی

۲	۱
x	y
0	0
0	1
1	1
1	0

آینه

٤	۲	۱
x	y	z
0	0	0
0	0	1
0	1	1
0	1	0
1	1	0
1	1	1
1	0	1
1	0	0

۸	٤	۲	۱
x	y	z	w
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0
0	1	1	1
0	1	0	1
0	1	0	0
1	1	0	0
1	1	0	1
1	1	1	1
1	1	1	0
1	0	1	0
1	0	1	1
1	0	0	1
1	0	0	0

* کد گری کدی می‌باشد که از هر شماره به شماره دیگر تنها یک تغییر دارد و تمام
کدها نسبت به کد شماره قبل یا بعد از خود تنها یک تغییر دارند.

کد BCD (Binary Coded Decimal)

اعداد ۰ تا ۹ که در مبنای ۱۰ هستند اگر بصورت Binary بنویسیم به آنها کد BCD می‌گویند

	۸	۴	۲	۱
۰	۰	۰	۰	۰
۱	۰	۰	۰	۱
۲	۰	۰	۱	۰
۳	۰	۰	۱	۱
۴	۰	۱	۰	۰
۵	۰	۱	۰	۱
۶	۰	۱	۱	۰
۷	۰	۱	۱	۱
۸	۱	۰	۰	۰
۹	۱	۰	۰	۱

۱۰ $\xrightarrow{\text{باینری}}$ ۱۰۱۰
۱۰ $\xrightarrow{BCD}$ ۰۰۰۱۰۰۰۰

۱۳۷ $\xrightarrow{\text{در باینری}}$ ۱۰۰۰۱۰۰۱
۱۳۷ $\xrightarrow{BCD}$ ۰۰۰۱۰۰۱۱۰۱۱۱

از صفر تا ۹ باینری و BCD مشترک است.

کد BCD یک کد ۴ بیتی که تنها اعداد ۰ تا ۹ را پوشش می‌دهد و این کد برای
نمایش اعدادی که هر رقم باید بطور جداگانه در خروجی نمایش داده شوند مفید است.

کد توازن : parity code

توازن فرد parity odd

توازن زوج parity even

فرد

مبدأ	0100 0001	1
مقصد	0100 0011	0

خطا

≠ خطا رخ داده است.

زوج

مبدأ	0100 0001	0
مقصد	0100 0011	1

≠ خطا رخ داده است.

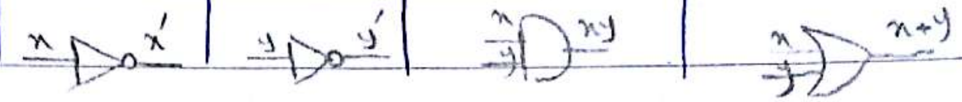
کد توازن بیت اضافه است که به همراه داده ارسال می‌شود تا در صورت بروز خطا، خطا را آشکار کند این بیت می‌تواند برای توازن زوج یا فرد استفاده شود.

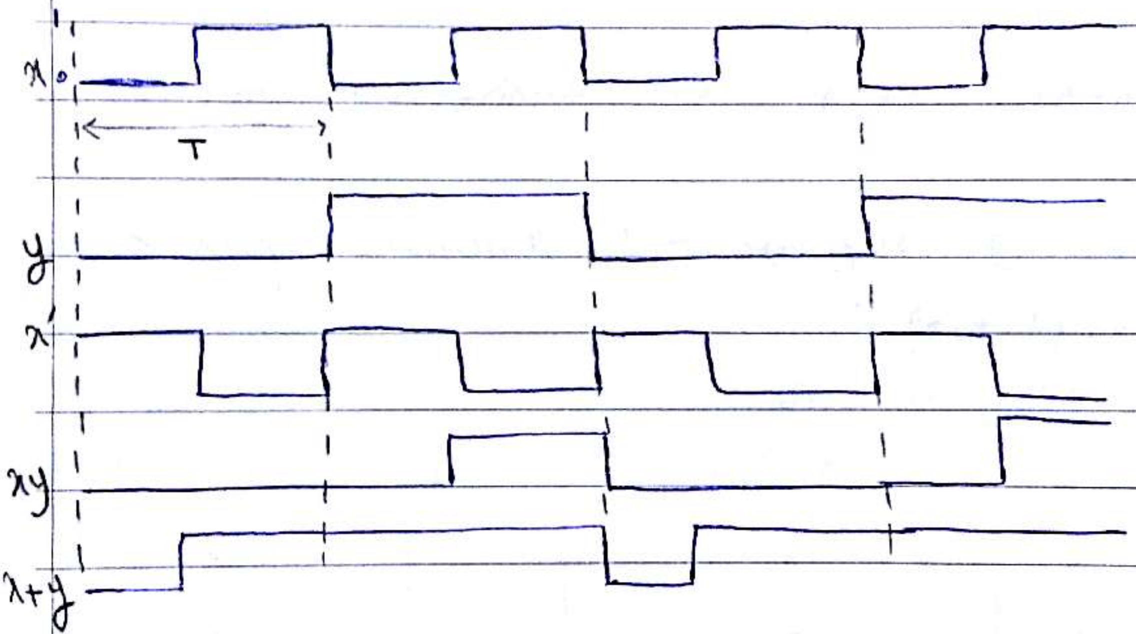
مبدأ و مقصد از یک توازن استفاده می‌نمایند. در صورتی که بیش از یک بیت خطا رخ دهد قابل تشخیص نیست.

منطق دودویی : Binary logic

مجموعه باینری $B = \{0 , 1\}$

ورودی		NOT		And	Or
x	y	$x' = \bar{x}$	y'	$x.y$ xy	$x+y$
0	0	1	1	0	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	1	1





$$\underset{\text{دوره تناوب}}{T}\ (\text{second}) = \frac{1}{\underset{\text{فرکانس}}{f}}\ \text{Hertz}$$

قضایا و اصول جبر بول :

بازگشت $(x')' = x$

عنصر خنثی $x \overset{or}{+} 0 = 0 + x = x$ $\qquad x.1 = 1.x = x$

$x+1 = 1+x = 1$ $\qquad x.0 = 0.x = 0$

$x+x = x$ $\qquad x.x = x$

عنصر مکمل $x+x' = x'+x = 1$ $\qquad x.x' = x'.x = 0$

خاصیت جابجایی $x+y = y+x$ $\qquad xy = yx$

شرکت پذیری $x+(y+z) = (x+y)+z$ $\qquad x.(y.z) = (xy).z$

قانون دمورگان $(x+y)' = x'.y'$ $\qquad (x.y)' = x'+y'$

توزیع پذیری $x+yz = (x+y)(x+z)$ $\qquad x.(y+z) = xy+xz$

قانون جذب $x+xy = x$ $\qquad x.(x+y) = x$

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

2) $x.(x+y) = (x+0).(x+y) = x+y.0 = x+0 = x$

روش دوم (با استفاده از جدول صحت درستی) : ② ①

$x+yz \overset{?}{=} (x+y)(x+z)$

x	y	z	yz	① $x+yz$	$x+y$	$x+z$	② $(x+y)(x+z)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Date :

دو روش برای اثبات تئوری ها وجود دارد در روش اول از سمت چپ رابطه شروع نموده و با کمک اصول جبر بول به رابطه سمت راست می رسیم. در روش دوم از جدول حالات استفاده نموده و دو طرف مساوی را در جدول پیاده سازی می کنیم باید مقادیر خروجی تطبیق به تطبیق با هم برابر باشند.

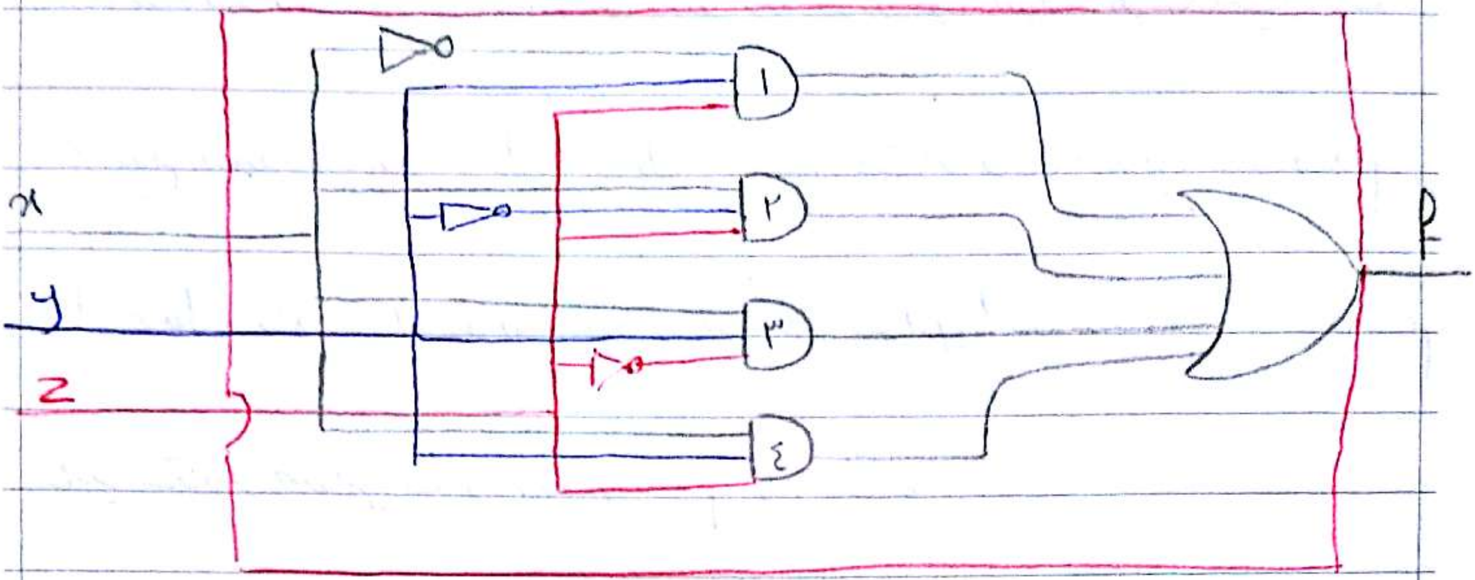
توابع بول (دو ارزشی) ۳

مثال: یک مسابقه وزنه برداری با ۳ داور، داوری می شود که هر کدام از داورها می توانند وزنه را تأیید یا رد نمایند. وزنه زمانی پذیرفته می شود که اکثریت داوران آنرا تأیید کنند. مدار آنرا طراحی و پیاده سازی کنید.

x	y	z	F = majority = voter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F = \underset{①}{x'yz} + \underset{②}{xy'z} + \underset{③}{xyz'} + \underset{④}{xyz}$$

Gate گیت



مثال: مداری طراحی کنید که اعداد زوج سه بیتی و بزرگتر از سه را مشخص کند

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

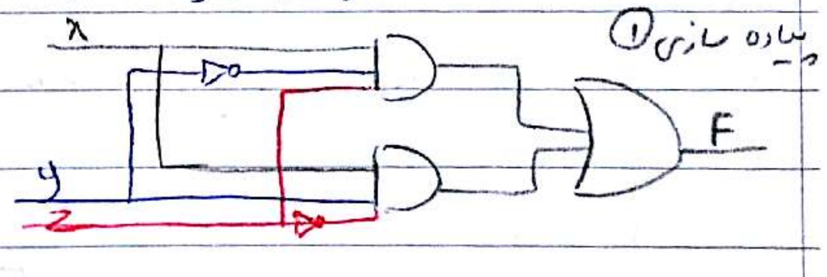
تعداد ورودی ها n

تعداد تابع $= 2^{2^n}$

۳ ورودی $\longrightarrow 2^{2^3} = 2^8 = 256$

$F = xy'z' + xyz'$

پیاده سازی ①



پیاده سازی ②

$F = xz'\underbrace{(y+y')}_{1} = xz'$

$F' = (xy'z' + xyz')' = (xy'z')' \cdot (xyz')' =$

$(x'+y+z) \cdot (x'+y'+z)$

@ comp ۹۴۱

@ class - status

Date : . .

$(A+B)' = A' \cdot B'$

$x = B+C$

$(A+B+C)' = (A+x)' = A' \cdot x' = A' \cdot B' \cdot C'$

مکمل کردن تابع
$$\begin{cases} x \longleftrightarrow x' \\ \cdot \longleftrightarrow + \end{cases}$$

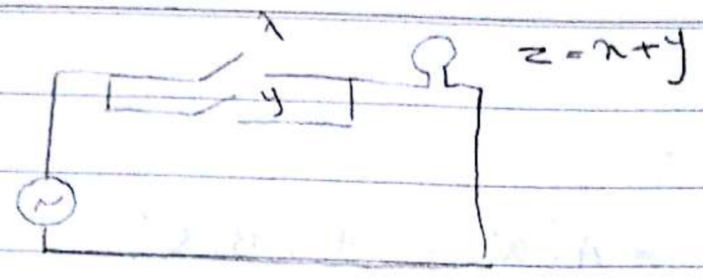
• برای محاسبه‌ی مکمل تابع کافیست که تک تک عملوندها و عملگرها را مکمل کنیم عملگر And به or و or به And تبدیل می‌شود.

_ توابع زیر را ساده کنید.

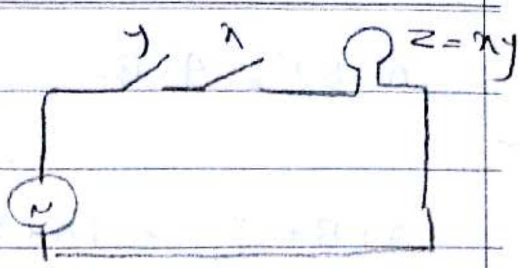
$x(x'+y) = xx' + xy = xy$

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

$(x+y)(x'+y') = xx' + xy' + x'y + yy' = xy' + x'y$



Or



And

فرم های متعارف و استاندارد :

ورودی ها

مینترم ها ماکسترم ها

فرم های متعارف =>

4 A	2 B	1 C	جمله	علامت	علامت	
0	0	0	$A'B'C'$	m_0	$A+B+C$	M_0
0	0	1	$A'B'C$	m_1	$A+B+C'$	M_1
0	1	0	$A'BC'$	m_2	$A+B'+C$	M_2
0	1	1	$A'BC$	m_3	$A+B'+C'$	M_3
1	0	0	$AB'C'$	m_4	$A'+B+C$	M_4
1	0	1	$AB'C$	m_5	$A'+B+C'$	M_5
1	1	0	ABC'	m_6	$A'+B'+C$	M_6
1	1	1	ABC	m_7	$A'+B'+C'$	M_7

جملات تابع می‌تواند به فرم And کردن ورودی بیان شود که به این جمله مینترم گفته می‌شود و مینترم متغیری که ورودی آن صفر باشد پریم دار ظاهر می‌شود.

شماره جمله از جمع ارزش متغیرهایی که در ورودی مقدار یک دارند بدست می‌آید.

مکمل مینترم ، ماکسترم گفته می‌شود در ماکسترم متغیرهای ورودی باهم OR می‌شوند

و متغیر پریم دار ظاهر می شود که در ورودی 1 باشد. شماره جملات از جمع ارزش متغیرهای پریم دار بدست می آید.

تبدیل فرم ها به یکدیگر:

مثال: تابع مربوط به آشکارساز اعداد اول سه بیتی را بنویسید (کوچکتر از ۸)

ورودی ها

A (۴)	B (۲)	C (۱)	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

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

$$F = m_2 + m_3 + m_5 + m_7$$ ← حاصل جمع مینترم ها

$$\boxed{f = \Sigma(2,3,5,7)}$$

$$\boxed{f' = \Sigma(0,1,4,6)} = A'B'C' + A'B'C + AB'C' + ABC'$$

$$F \xrightarrow{NOT} F'$$

$$F' = (A'BC' + A'BC + AB'C + ABC)'$$

$$F' = (A+B'+C)\cdot(A+B'+C')\cdot(A'+B+C')\cdot(A'+B'+C')$$

$$\boxed{F' = M_2 \cdot M_3 \cdot M_5 \cdot M_7 = \Pi(2,3,5,7)}$$ ← حاصلضرب ماکسترم

$$\boxed{F = \Pi(0,1,4,6)}$$

Date :

* برای تبدیل تابع فرم جمع مینترم ها به ضرب ماکسترم ها باید شماره هایی که

در تابع ذکر نشده است را به عنوان شماره مورد استفاده در ضرب ماکسترم ها

بکار برد.

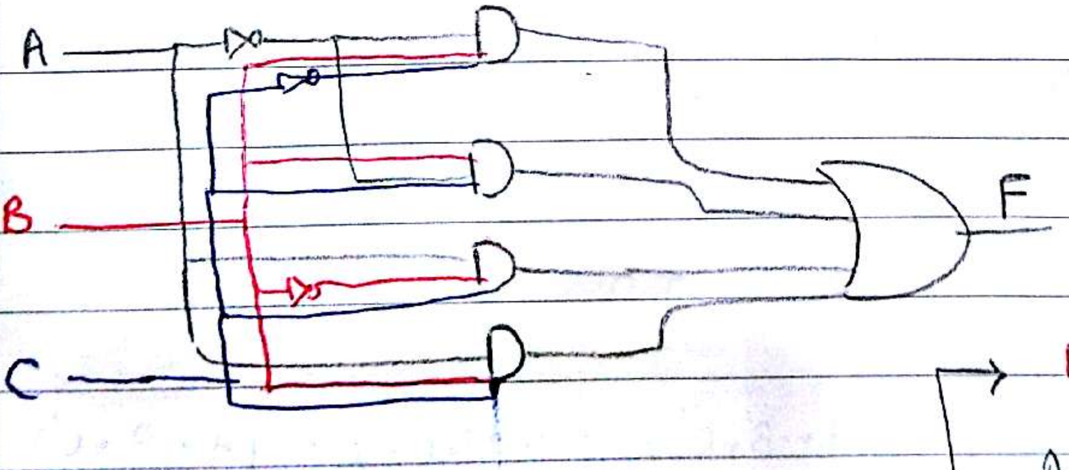
فرم های استاندارد ⇐

در فرم استاندارد تابع بعضی از متغیرهای ورودی در بعضی از حالات حضور

ندارد و این متغیر حذف می‌شود.

فرم های تابع
- فرم متعارف (کانونیکال)
  - حاصل جمع مینترم ها invert-And-or
  - حاصل ضرب ماکسترم ها invert-or-And
- ساده شود ↓
- فرم استاندارد
  - حاصل جمع حاصل ضربها
  - حاصلضرب حاصل جمع ها

مثال: $F = A'BC' + A'BC + AB'C + ABC$



$F = A'B(C'+C) + AC(B'+B) \rightarrow F = A'B + AC$

AZAD

عملگرهای منطقی و گیت های منطقی:

تعداد ورودی های تابع = n $\Longrightarrow$ 2^n حالت

2^{2^2} تابع

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}
0	0	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	1	1	1	1
		$f=0$	Nor $(x+y)'$	$x'y$	Not $F=x'$	xy'	Not y'	$x'y+y'x$ آشکارساز نابرابری Eor = Xor	$(xy)'$ NanD	xy And	آشکارساز برابری XNor $xy+x'y'$	buffer	$(xy')'$

F_{12}	F_{13}	F_{14}	F_{15}
0	1	0	1
0	0	1	1
1	1	1	1
1	1	1	1
buffer $x \rhd x$	$(x'y)'$	or $x+y$	$f=1$

ساده سازی توابع با استفاده از جدول (نقشه) کارنو

$$xy + x'y = y\underbrace{(x+x')}_{1} = y$$

متغیری که تغییر می‌کند حذف می‌شود

هنگامیکه تنها یک تغییر داشته باشیم می‌توان فاکتور گرفت.

$$xy' + x'y$$

۱- نقشه (جدول) دو متغیره

A	B	
0	0	$A'B'$
0	1	$A'B$
1	0	AB'
1	1	AB

	0	1
0	00	01
1	10	11

A \ B	B'	B
A'	$A'B'$	$A'B$
A	AB'	AB

m_0	m_1
m_2	m_3

$F_1 = \Sigma(0,1)$

	B'	B
A'	1	1
A	0	0

A'

$F_2 = \Sigma(0,2)$

		B
	1	0
A	1	0

B'

$F_3 = \Sigma(0,3)$

1	0
0	1

$A'B' + AB$

$F_4 = \Sigma(1,2)$

0	1
1	0

$A'B + AB'$

$F_5 = \Sigma(1,3)$

0	1
0	1

B

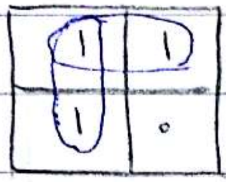
$F_6 = \Sigma(2,3)$

0	0
1	1

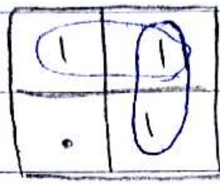
A

Date : . .

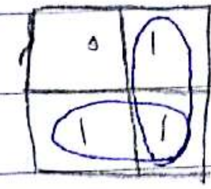
$F_7 = \Sigma(0,1,2)$ $F_8 = \Sigma(0,1,3)$ $F_9 = \Sigma(1,2,3)$



$B' + A'$



$B + A'$



$B + A$

دو جمله از تابع هنگامی می‌تواند ساده شود که تنها یک متغیر تغییر داشته باشد. متغیری که تغییر دارد از بین می‌رود. جدول دو متغیره دارای ٤ خانه می‌باشد که هر خانه به یک جمله از مینترم نسبت داده می‌شود. هر تا زدن افقی که سبب شود ۲ خانه بر روی هم منطبق گردد به این دو خانه، خانه‌های همجوار گفته می‌شود و این خانه‌ها تنها در یک متغیر تغییر دارند و متغیری که تغییر می‌کند از بین می‌رود. در پیوند ۲ تایی یک متغیر، در پیوند ٤ تایی دو متغیر، در پیوند ۸ تایی سه متغیر و در پیوند ١٦ تایی چهار متغیر از بین می‌رود. امکان تشکیل پیوند مورب وجود ندارد.

پیوندهای ٦ تایی و ١٠ تایی نیز نداریم.

۲_ جدول سه متغیره

A	B	C	
0	0	0	$A'B'C'$
0	0	1	$A'B'C$
0	1	0	$A'BC'$
0	1	1	$A'BC$
1	0	0	$AB'C'$
1	0	1	$AB'C$
1	1	0	ABC'
1	1	1	ABC

	00	01	11	10
0	000	001	011	010
1	100	101	111	110

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

A \ BC	B'		B	
A'	$A'B'C'$	$A'B'C$	$A'BC$	$A'BC'$
A	$AB'C'$	$AB'C$	ABC	ABC'
	C'	C	C	C'

$F_1 = \Sigma(1,2,4,5)$

		B	
0	1	0	1
1	1	0	0

(row 2 = A; columns 2–3 = C)

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

$F_2 = \Sigma(0,3,6,7)$

1	0	1	0
0	0	1	1

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

$F_3 = \Sigma(0,1,2,5,6)$

1	1	0	1
0	1	0	1

$BC' + B'C + A'B'$

$F_4 = \Sigma(0,2,4,6,7)$

1	0	0	1
1	0	1	1

$AB + C'$

$F_5 = \Sigma(0,1,4,5,7)$

1	1	0	0
1	1	1	0

$B' + AC$

$F_6 = \Sigma(0,2,5,7)$

1	0	0	1
0	1	1	0

$A'C' + AC$

$F_7 = \Sigma(1,3,6,7)$

غیرضروری ؟

0	1	1	0
0	0	1	1

$AB + A'C$

$F_8 = \Sigma(0,2,3,4,5,6,7)$

1	0	0	1
1	1	1	1

$C' + A$

$F_9 = \Sigma(0,2,4,6)$

1	0	0	1
1	0	0	1

C'

$F_{10} = \Sigma(0,1,2,3,7)$

1	1	1	1
0	0	1	0

$A' + BC$

$F_{11} = \Sigma(0,1,3,6,7)$

1	1	1	0
0	0	1	1

$AB + A'B'$

؟

$F_{12} = \Sigma(1,3,4,5,6,7)$

0	1	1	0
1	1	1	1

$A + C$

۳- جدول ۴ متغیره

	00	01	11	10
00	0000	0001	0011	0010
01	0100	0101	0111	0110
11	1100	1101	1111	1110
10	1000	1001	1011	1010

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

C

$A'B'C'D'$	$A'B'C'D$	$A'B'CD$	$A'B'CD'$
$A'BC'D'$	$A'BC'D$	$A'BCD$	$A'BCD'$
$ABC'D'$	$ABC'D$	$ABCD$	$ABCD'$
$AB'C'D'$	$AB'C'D$	$AB'CD$	$AB'CD'$

B

A

D

Date : . .

$F_1 = \Sigma$ (۰, ۱, ۲, ۴, ۵, ۸, ۱۰)

1	1	0	1
1	1	0	0
0	0	0	0
1	0	0	1

$B'D' + A'C'$

$F_2 = \Sigma$ (۱, ۳, ۴, ۵, ۶, ۷, ۹, ۱۱)

0	1	1	0
1	1	1	1
0	0	0	0
0	1	1	0

$A'B + B'D$

$F_3 = \Sigma$ (۰, ۲, ۴, ۶, ۸, ۱۰, ۱۴, ۱۵)

1	0	0	1
1	0	0	1
0	0	1	1
1	0	0	1

$B'D' + A'D' + ABC$

$F_4 = \Sigma$ (۱, ۳, ۵, ۷, ۹, ۱۱, ۱۲, ۱۳)

0	1	1	0
0	1	1	0
1	1	0	0
0	1	1	0

$B'D + A'D + ABC'$

$F_5 = \Sigma$ (۰, ۲, ۳, ۴, ۶, ۸, ۱۱, ۱۳)

1	0	1	1
1	0	0	1
0	1	0	0
1	0	1	0

$A'D' + B'CD + B'C'D' + ABC'D$

$F_6 = \Sigma$ (۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۱۰, ۱۲, ۱۴)

0	1	1	1
0	1	1	1
1	0	0	1
1	0	0	1

$AD' + A'D + CD'$

$F_7 = \Sigma(0,3,6,9,12,15)$ $\qquad$ $F_8 = \Sigma(0,2,5,7,10,12,15)$

1	0	1	0
0	0	0	1
1	0	1	0
0	1	0	0

ساده نمیشود ۶ جمله

1	0	0	1
0	1	1	0
1	0	1	0
0	0	0	1

$A'B'D' + B'CD' + BCD + A'BD + ABC'D'$

هنگام ساده سازی تا زمانی که پیوند های بزرگتری در دسترس می باشد به سراغ پیوند کوچکتر نمی رویم و در پیوند های برابر از خانه ای شروع می کنیم که شانس تشکیل پیوند آن کمتر باشد.

ساده سازی بفرم ضرب حاصل جمع ها

$F_1 = \Sigma(0,1,2,4,5,8,10)$

$\Pi(3,6,7,9,11,12,13,14,15)$

$F_2 = \Pi(0,2,8,10,12,13,14,15)$

1	1	0	1
1	1	0	0
0	0	0	0
1	0	0	1

A, B, C, D

$F' = BC + AB + AD + CD$

0	1	1	0
1	1	1	1
0	0	0	0
0	1	1	0

$F' = B'D' + AB + AD'$

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

$F_3 = \Sigma(0,2,4,6,8,10,14,15)$

$F_3 = \Pi(1,3,5,7,9,11,12,13)$

1	0	0	1
1	0	0	1
0	0	1	1
1	0	0	1

$f' = B'D + A'D + ABC'$

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

$F_4 = \Sigma(1,3,5,7,9,11,12,13)$

$F_4 = \Pi(0,2,4,6,8,10,14,15)$

0	1	1	0
0	1	1	0
1	1	0	0
0	1	1	0

$f' = B'D' + A'D' + ABC$

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

در پیاده سازی تابع بفرم ضرب حاصل جمع ها برای ساده سازی آن می توان ابتدا f' را محاسبه نموده سپس با مکمل کردن آن f را محاسبه کنیم. برای محاسبه f' صفرهای جدول را انتخاب نموده با پیوند های مناسب تابع را ساده می نماییم. سپس پس از کامل نمودن پیوند ها تابع را مجدداً مکمل می نماییم تا مقدار f حاصل شود. تابع حاصل شده بفرم ضرب حاصل جمع ها می باشد.

٤- جدول پنج متغیره

A′ (D, E, B, C):

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

A (D, E, B, C):

m_{16}	m_{17}	m_{19}	m_{18}
m_{20}	m_{21}	m_{23}	m_{22}
m_{28}	m_{29}	m_{31}	m_{30}
m_{24}	m_{25}	m_{27}	m_{26}

در جدول پنج متغیره ۳۲ خانه موجود است و از ۲ جدول چهار متغیره استفاده

می‌کنند پیوندها مانند قبل است به جز اینکه تکرار در صورت تطبیق خانه‌های

تطبیق به تطبیق از دو جهت بر روی هم دیگر آنها را نیز می‌توان ساده نمود.

$$F(A,B,C,D,E) = \Sigma(0,2,5,7,8,10,12,13,14,15,16,18,19,21,23,24,26,27,29,31)$$

A′ (D, E, B, C):

1	0	0	1
0	1	1	0
1	1	1	1
1	0	0	1

A (D, E, B, C):

1	0	1	1
0	1	1	0
0	1	1	0
1	0	1	1

$$F = C'E' + CE + ADE + A'BC$$

مدارات منطقی
۱ـ مدارات ترکیبی
۲ـ مدارات ترتیبی

در مدارات ترکیبی خروجی در هر لحظه تنها به ورودی همان لحظه بستگی دارد.

در مدارات ترتیبی خروجی در هر لحظه به ورودی همان لحظه و لحظات قبل وابسته می باشد و خود به دو دسته تقسیم می شود.

در مدارات ترتیبی سنکرون (همزمان) مدار با پالس ساعت فعال می شود. در مدارات ترتیبی آسنکرون نیازی به پالس ساعت برای فعال سازی نمی باشد در خروجی بلافاصله پس از دریافت ورودی تولید می شود هر دو گروه از مدار دارای عنصر حافظه می باشند.

طراحی مدارات ترکیبی:
مراحل زیر در طراحی مدارات ترکیبی به کار گرفته می شوند:

۱ـ تشکیل جدول حالات یا جدول درستی (در جدول درستی در سمت چپ متغیرهای ورودی و در سمت راست متغیرهای خروجی قرار می گیرند و شامل تمام

حالات متغیرها در ورودی می باشد.)

۲- نسبت دادن متغیرهای خروجی به هرکدام از ورودیها و خروجی ها.

۳- استفاده از جدول کارنو برای استخراج توابع خروجی با استفاده از ستونهای سمت راست جدول.

۴- پیاده سازی توابع استخراج شده با استفاده از گیت های منطقی.

(Half Adder)

مثال : یک نیم جمع کننده طراحی و پیاده سازی کنید. نیم جمع کننده مداریست که ۲ بیت را باهم جمع میکند.

۱۳ +
۱
۱۴

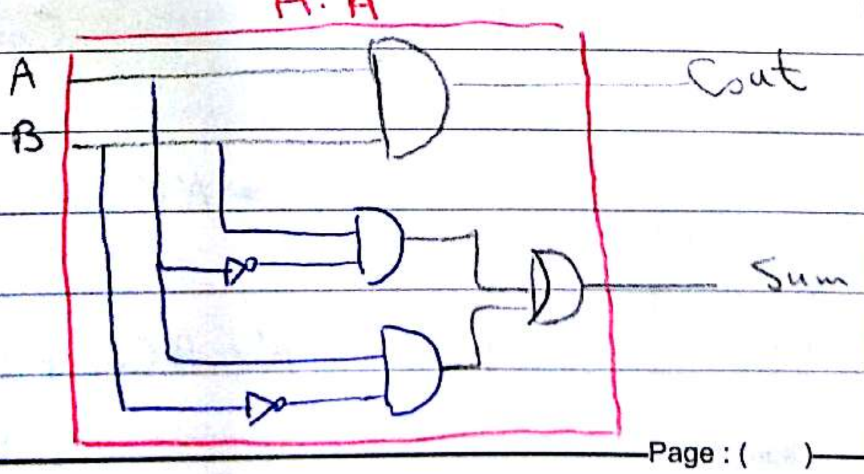
۱ ۱ ۰ ۱
۱
۰ sum
۱ carry
(انتقال/ نقلی)

۱۰۱۰
۱
۱ sum
۰ carry

A	B	Cout	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

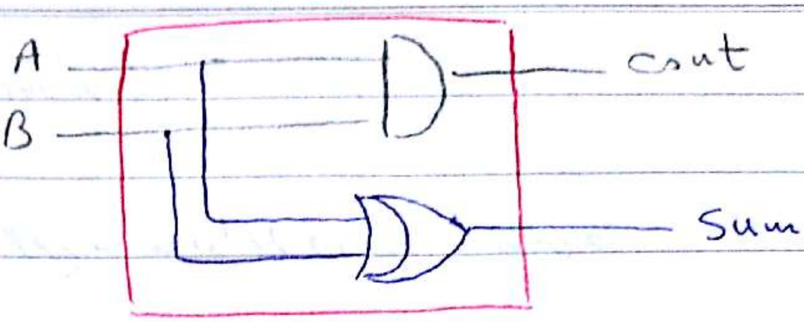
$Cout = A \cdot B$

$Sum = A'B + AB'$



$Cout = AB$

$Sum = A \oplus B$



مثال: یک مدار تمام جمع کننده طراحی کنید. مدار تمام جمع کننده مداری است که ۳ بیت را باهم جمع می کند.

$$\begin{array}{r} ۱۴ \\ +\ ۱۳ \\ \hline ۲۷ \end{array}$$

		1	0	0	0	Cin
		1	1	1	0	A
+		1	1	0	1	B
1	1	0	1	1		sum
		1	1	0	0	Cout

ورودی ها خروجی ها

A	B	C	Cout	Sum
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

$Cout = \Sigma(۳, ۵, ۶, ۷)$

$Sum = \Sigma(۱, ۲, ۴, ۷)$

B

	0	0	1	0
A	0	1	1	1

Cin

$Cout = BC_{in} + AC_{in} + AB$

B

	0	1	0	1
A	1	0	1	0

Cin

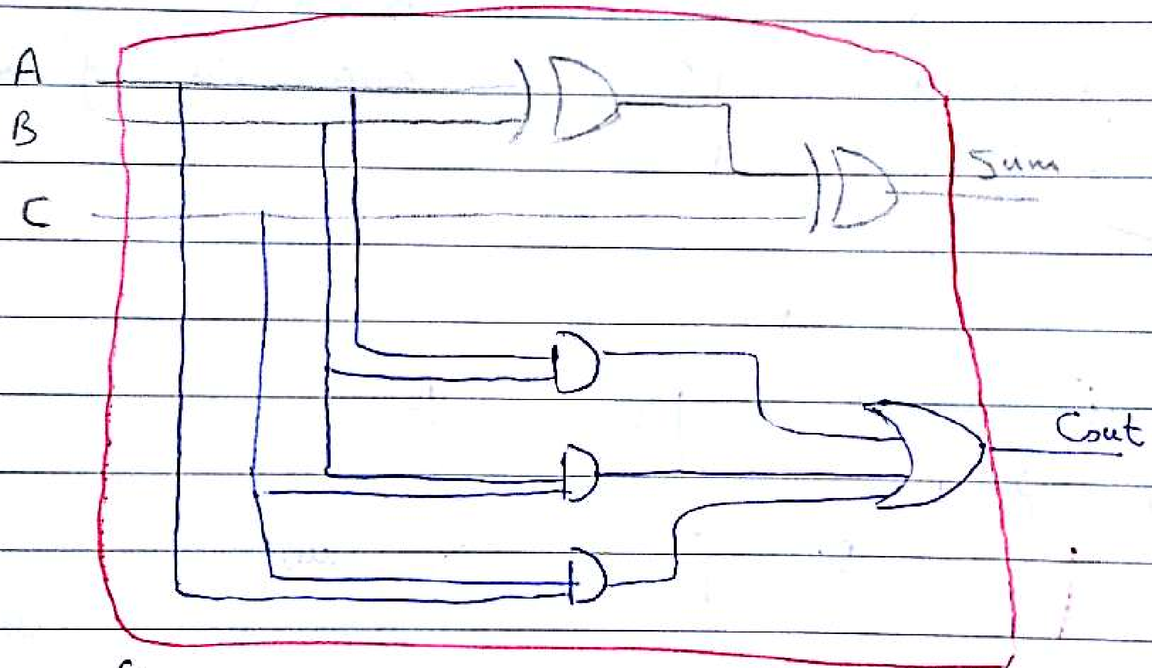
$Sum = A'B'C_{in} + A'BC'_{in} + AB'C'_{in} + ABC_{in}$

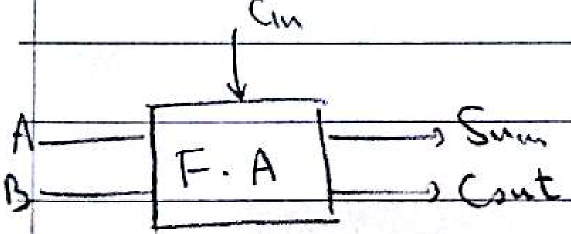
$= A'(B'C_{in} + BC'_{in}) + A(B'C'_{in} + BC_{in})$

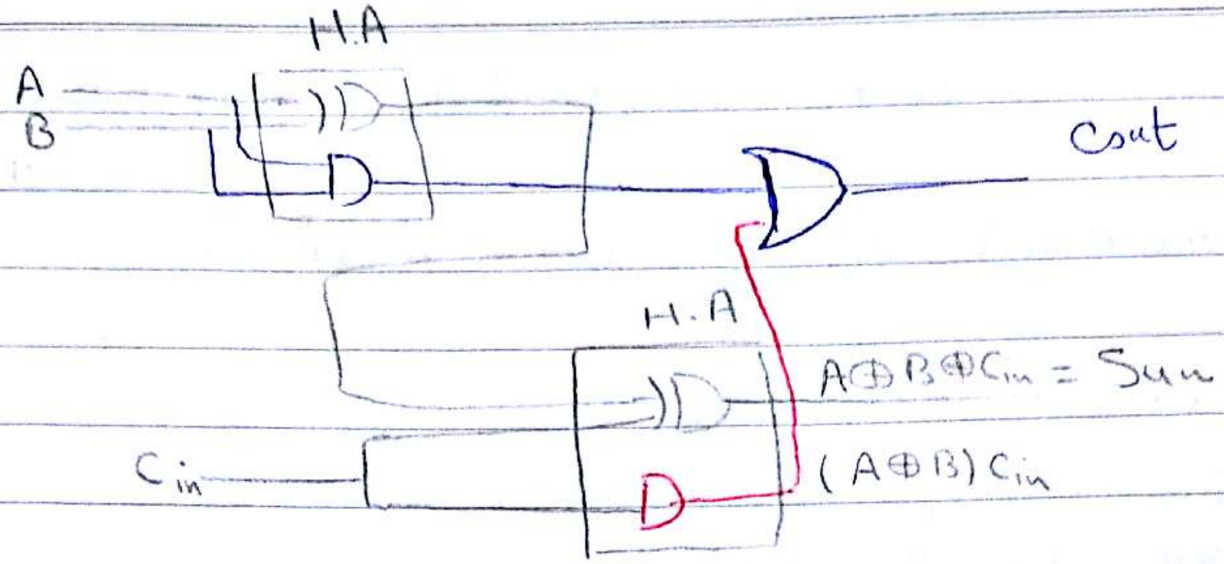
$$Sum = A'(B \oplus C_{in}) + A(B \odot C_{in}) \rightarrow Sum = A'(B \oplus C_{in}) + A(B \oplus C_{in})' \rightarrow Sum = A \oplus B \oplus C_{in}$$

$$\boxed{\begin{aligned} xor &= B'C + BC' = B \oplus C \\ xNor &= B'C' + BC = B \odot C \\ (B \odot C)' &= B \oplus C \end{aligned}}$$

Full Adder







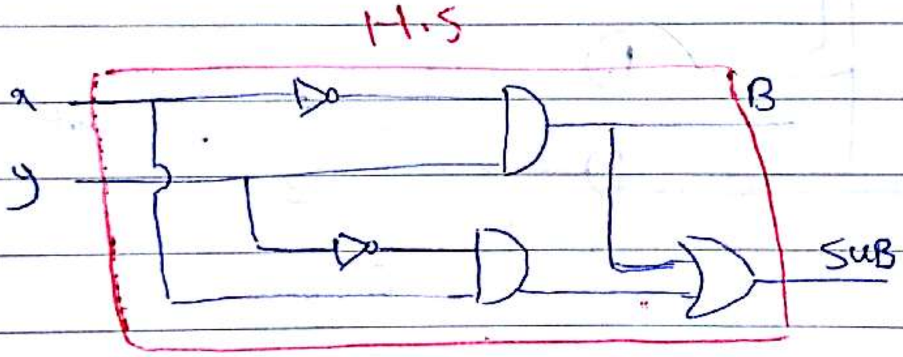
طراحی مدار نیم تفریق کننده (H.S)

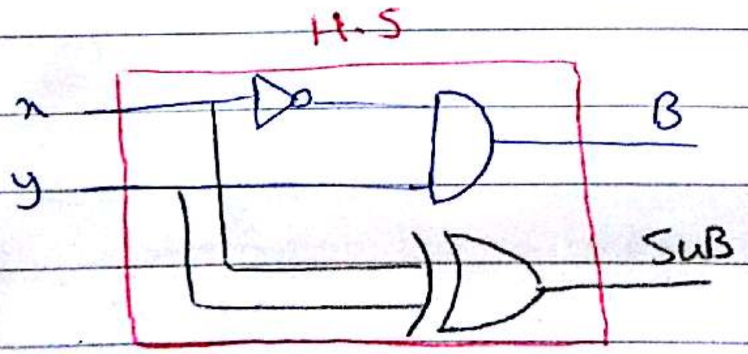
Half Subtracter

x	y	Borrow	SUB
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$B = x'y$

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





نیم تفریق گر مداریست که دو بیت را از هم کم می نماید.

طراحی مدار تمام تفریق گر (F.S)

Full Subtract

تمام تفریق گر مداریست که ۳ بیت را از هم کم می نماید دو بیت از طبقه فعلی و

یک بیت مربوط به قرض احتمالی به طبقات پایین تر.

x	y	B_{in}	B_{out}	SuB
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

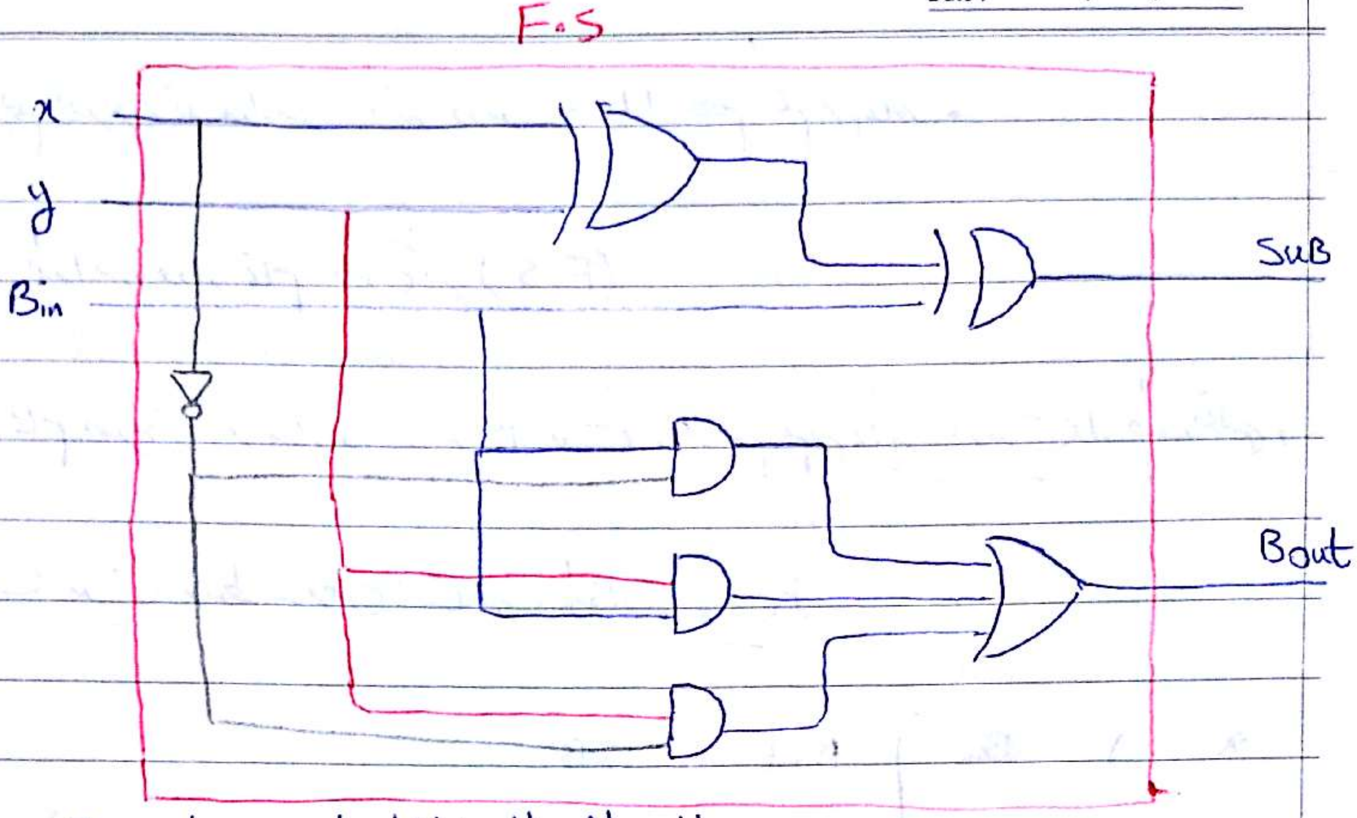
$$B_{out} = \Sigma(1,2,3,7) = yB_{in} + x'B_{in} + x'y$$

			y	
	0	1	1	1
x	0	0	1	0
		B_{in}		

$$SuB = \Sigma(1,2,4,7) = x'y'B_{in} + x'yB'_{in} + xy'B'_{in} + xyB_{in} =$$

$$x'(y \oplus B_{in}) + x(y \odot B_{in}) = x \oplus y \oplus B_{in}$$

			y	
	0	1	0	1
x	1	0	1	0
		B_{in}		



طراحی مدار ضرب کننده ۲ بیتی

$n \times n = 2n$

a_1	a_0	b_1	b_0	H_3	H_2	H_1	H_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

۰	۱	۳	۲
٤	۵	۷	۶
۱۲	۱۳	۱۵	۱٤
۸	۹	۱۱	۱۰

$\boxed{H_۳ = a_1 a_0 b_1 b_0}$

b_1

0	0	0	0
0	0	0	0
0	0	0	1
0	0	1	1

a_0, a_1, b_0

$\boxed{H_۲ = a_1 b_1 b_0' + a_1 a_0' b_1}$

b_1

0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0

a_0, a_1, b_0

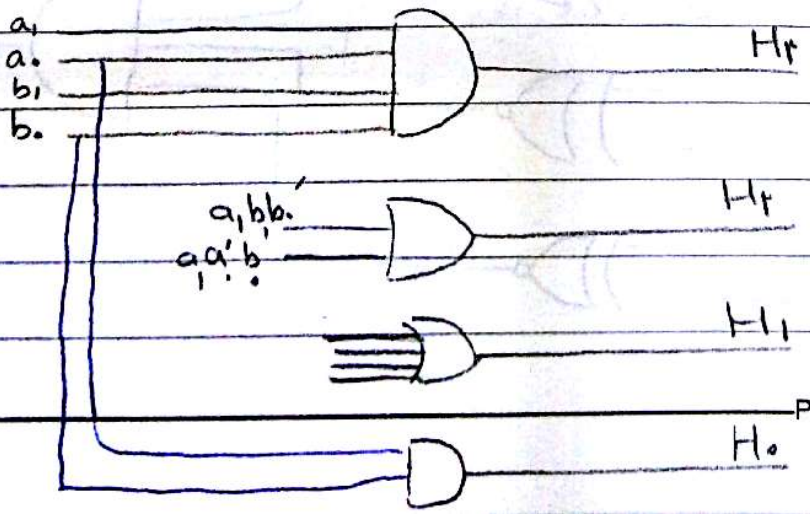
$$\boxed{H_1 = a_1 a_0' b_0 + a_1 b_1' b_0 + a_0 b_1 b_0' + a_1' a_0 b_1}$$

b_1

0	0	0	0
0	1	1	0
0	1	1	0
0	0	0	0

a_0, a_1, b_0

$\boxed{H_0 = b_0 a_0}$



مقایسه گر Comparator

طراحی مقایسه گر ۴ بیتی

٨ ٤ ٢ ١

$A = A_3 A_2 A_1 A_0$

$B = B_3 B_2 B_1 B_0$

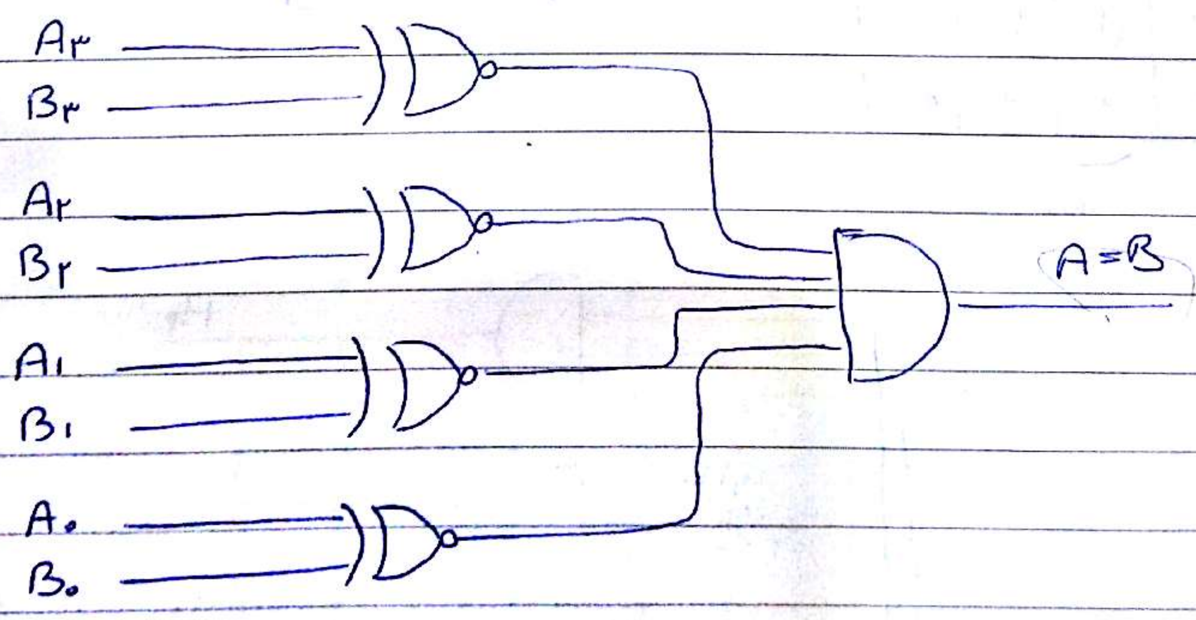
$$A > B \iff A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

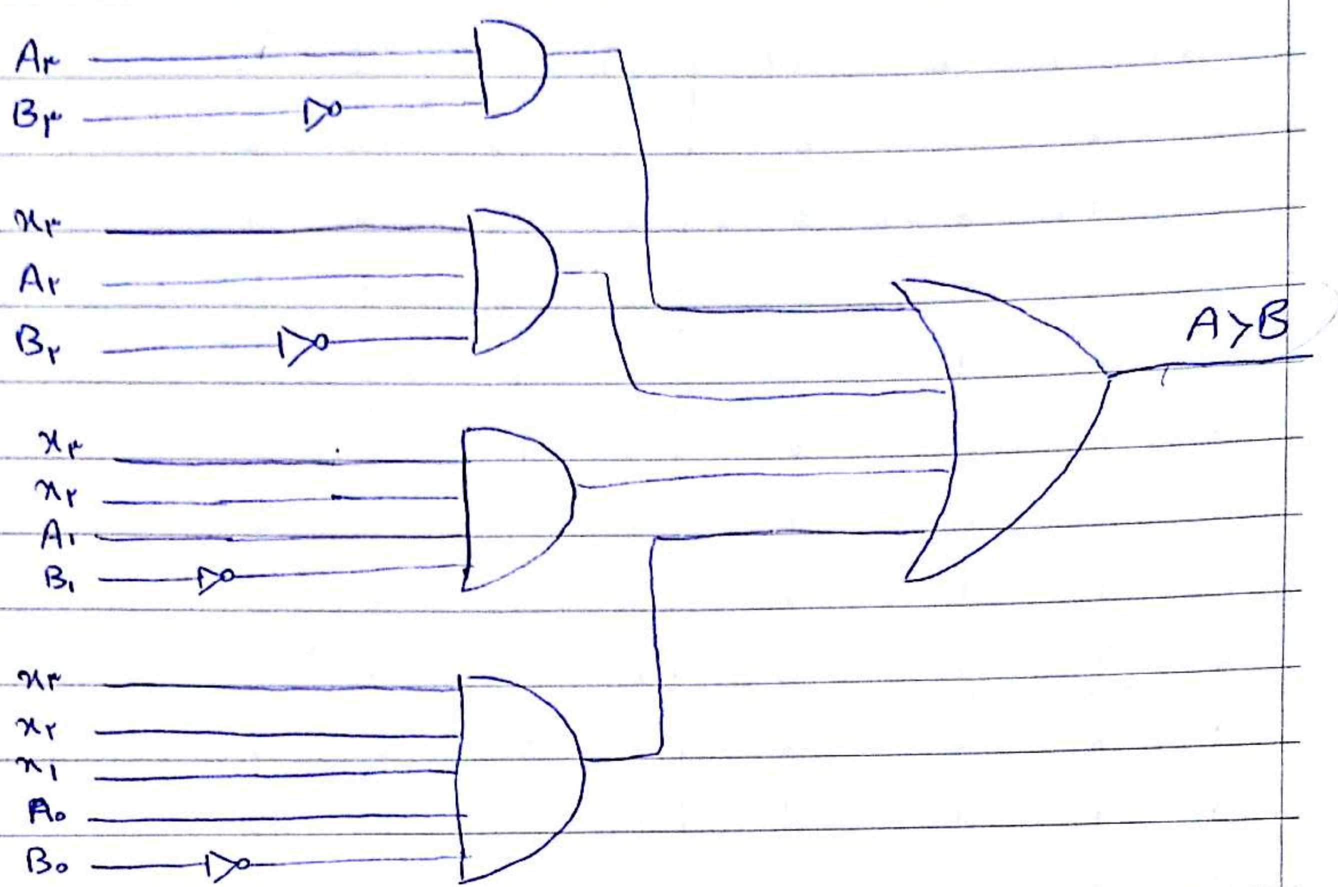
A = ۱۰۰۱	
B = ۰۱۱۱	
A = ۰۱۰۱	A = ۱۱۰۱
B = ۰۰۱۱	B = ۱۰۱۱

$$A = B \iff x_3 x_2 x_1 x_0$$

$$A < B \iff A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0$$

$x_i = A_i B_i + A_i' B_i' = A_i \odot B_i$ آشکارساز برابری





مبدل کد ۵

مبدل کد یک مدار ترکیبی است که یک کد را به عنوان ورودی دریافت نموده و

کد دیگری را در خروجی تحویل می دهد.

مثال: یک مبدل کد ۴ بیتی که کد باینری را به گری تبدیل نماید طراحی و پیاده سازی کنید.

جواب صفحه بعد

Date : . .

ورودی باینری — خروجی گری

x	y	z	w	A	B	C	D	
0	0	0	0	0	0	0	0	
0	0	0	1	0	0	0	1	آینه
0	0	1	0	0	0	1	1	
0	0	1	1	0	0	1	0	آینه
0	1	0	0	0	1	1	0	
0	1	0	1	0	1	1	1	
0	1	1	0	0	1	0	1	
0	1	1	1	0	1	0	0	آینه
1	0	0	0	1	1	0	0	
1	0	0	1	1	1	0	1	
1	0	1	0	1	1	1	1	
1	0	1	1	1	1	1	0	
1	1	0	0	1	0	1	0	
1	1	0	1	1	0	1	1	
1	1	1	0	1	0	0	1	
1	1	1	1	1	0	0	0	

$\boxed{A = x}$

			z	z	
	0	0	0	0	
	1	1	1	1	y
x	0	0	0	0	y
x	1	1	1	1	
		w	w		

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

Date : . .

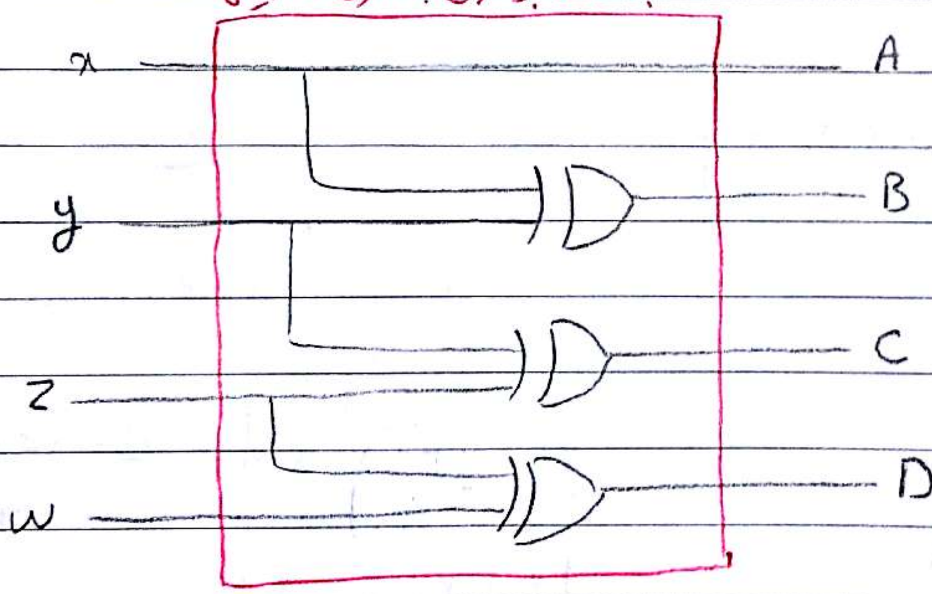
0	0	1	1
1	1	0	0
1	1	0	0
0	0	1	1

$$C = yz' + y'z = y \oplus z$$

0	1	0	1
0	1	0	1
0	1	0	1
0	1	0	1

$$D = z'w + zw'$$

مبدل کد باینری به گری ۴ بیتی



مثال: یک مبدل کد طراحی کنید که کد ۱ ۲ ۴ ۲ به کد ۱- ۲- ۴ ۸ تبدیل کند.

	۲	۴	۲	۱	۸	۴	-۲	-۱
۰	۰	۰	۰	۰	۰	۰	۰	۰
۱	۰	۰	۰	۱	۰	۱	۱	۱
۲	۰	۰	۱	۰	۰	۱	۱	۰
۳	۰	۰	۱	۱	۰	۱	۰	۱
۴	۰	۱	۰	۰	۰	۱	۰	۰
۱۱ ۵	۱	۰	۱	۱	۱	۰	۱	۱
۶	۱	۱	۰	۰	۱	۰	۱	۰
۷	۱	۱	۰	۱	۱	۰	۰	۱
۸	۱	۱	۱	۰	۱	۰	۰	۰
۱۵ ۹	۱	۱	۱	۱	۱	۱	۱	۱

$d = \Sigma(۵,۶,۷,۸,۹,۱۰)$

Don't Care حالت‌های

$\boxed{A = x}$

0	1	1	1
1	X	X	X
0	0	1	0
X	X	0	X

$B = x'z + x'w + yzw + x'y'z'$

0	1	0	1
0	X	X	X
1	0	1	0
X	X	1	X

0	1	1	0
0	X	X	X
0	1	1	0
X	X	1	X

$D = w$

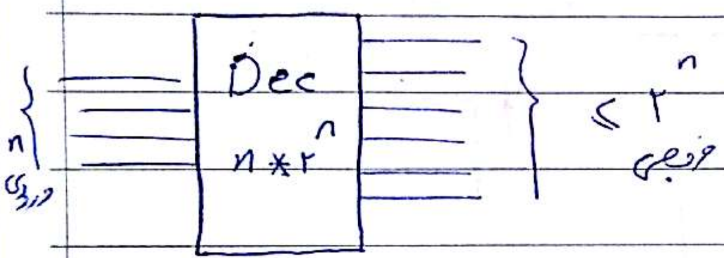
$C = x'z'w + x'zw' + xz'w' + xzw$

$C = x'(z \oplus w) + x(z \odot w)$

$= x \oplus z \oplus w$

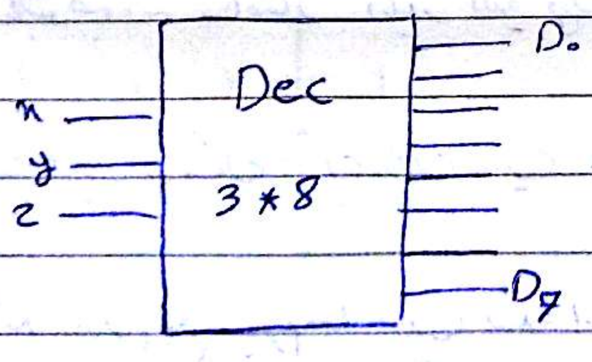
در بعضی از کدها بعضی از حالات ورودی هرگز اتفاق نمی افتد به این حالات، حالات بی اهمیت گفته می‌شود و در مدار هنگام جایگذاری در جدول شماره آن با x ، or مشخص می‌شود در ساده سازی اگر کمک کند پیوند قوی تری تشکیل شود به آن مقدار یک داده و استفاده می‌شود در غیر اینصورت از آن صرف نظر می‌شود و صفر می‌شود.

کدگشا (Decoder)



دیکدر یا کدگشا یک مدار ترکیبی با n ورودی و حداکثر 2^n خروجی است که براساس مقدار ورودی آن در هر لحظه یکی از خروجی ها فعال و بقیه خروجی ها غیر فعال می باشند.

طراحی دیکدر 3*8



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

Dec 3*8

عبارت مینترم

$D_0 = x'y'z'$

$D_1 = x'y'z$

$D_2 = x'yz'$

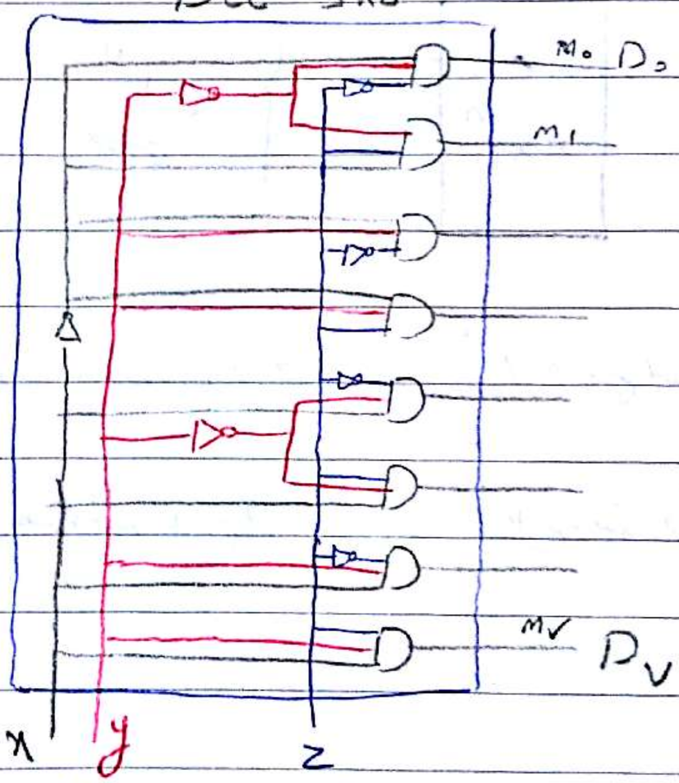
$D_3 = x'yz$

$D_4 = xy'z'$

$D_5 = xy'z$

$D_6 = xyz'$

$D_7 = xyz$

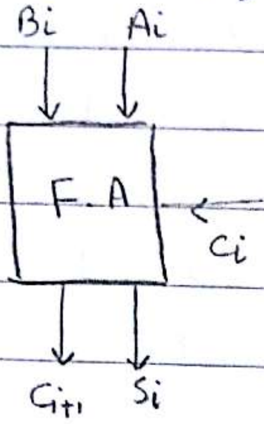


با توجه به ساختار دیکدر دیده می‌شود که خروجی های دیکدر عبارت مینترم می‌باشد.

بنابراین به شکل پیش فرض بیان نمود که از دیکدر برای پیاده سازی توابعی

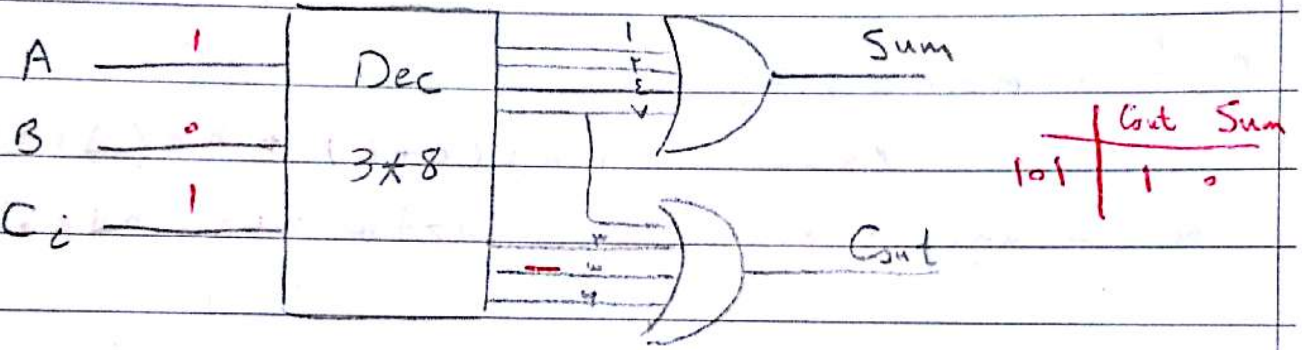
که به فرم جمع مینترم ها بیان شده اند استفاده می‌شود.

مثال: با استفاده از دیکدر مناسب یک تمام جمع کننده طراحی و پیاده‌سازی کنید.



$$Sum(A,B,C_i) = \Sigma(1,3,4,7) = m_1 + m_2 + m_4 + m_7 = x'y'z + x'yz' + xy'z' + xyz$$

$$C_{out}(A,B,C_i) = \Sigma(3,5,6,7)$$



مثال: یک مدار اکثریت ۳ بیتی طراحی و با استفاده از دیکدر مناسب پیاده‌سازی کنید.

x	y	z	majority
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\boxed{m = \Sigma(3,5,6,7)}$$

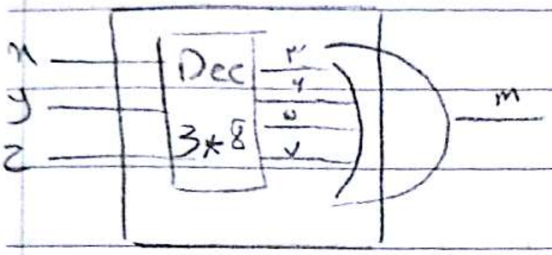
$$m' = \Sigma(0,1,2,4)$$

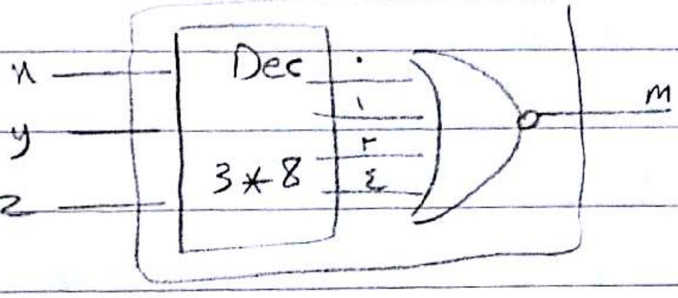
$$\boxed{m = \Pi(0,1,2,4)}$$

			y	
	0	0	1	0
x	0	1	1	1
		z		

$$m = yz + xz + xy$$

مثال ۱: $m=\Sigma(۳,۵,۶,۷)$ مثال ۲: $m=\Pi(۰,۱,۲,۴)$

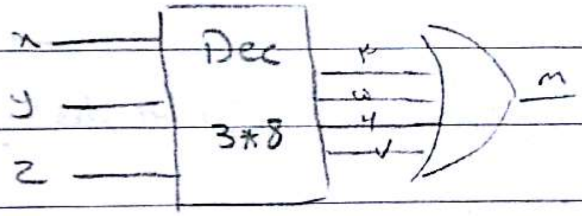


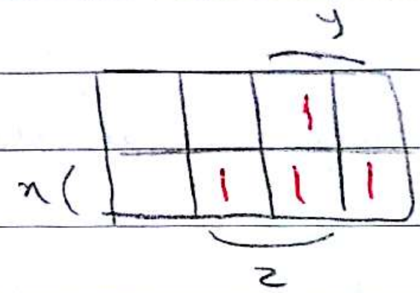


مثال ۳: $m = yz + xy + xz$

$(x+x')yz + xy(z+z') + xz(y+y')$

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





ساخت دیکدرهای بزرگتر

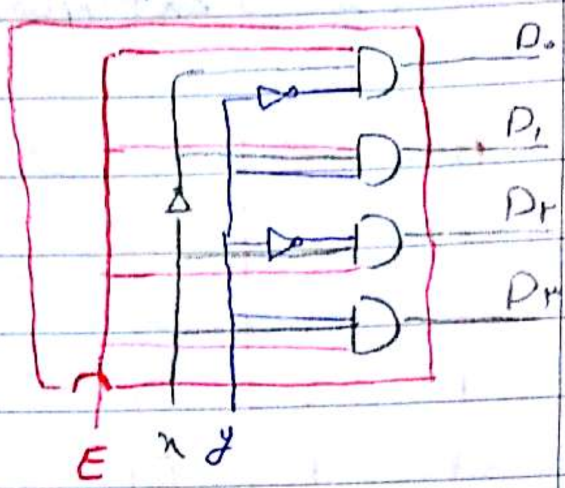
مثال یک: ساخت دیکدر ۳*۸ با استفاده از دیکدرهای ۲*۴ با خط فعالساز

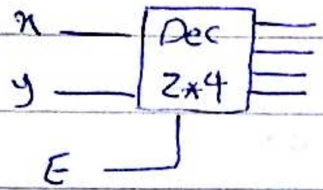
دیکدر با خط فعالساز:

طراحی دیکدر ۲*۴ با خط فعالساز

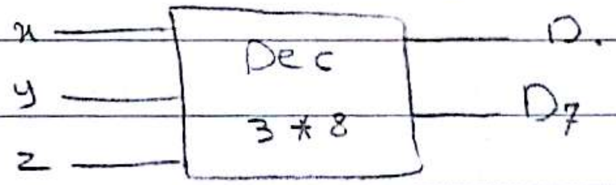
در دیکدر با خط فعالساز در صورتی که خط فعالساز صفر باشد همه‌ی خروجی‌ها صفر می‌باشد و در صورت یک شدن این خط براساس مقدار ورودی یکی از خروجی‌ها ۱ و بقیه صفر می‌باشند.

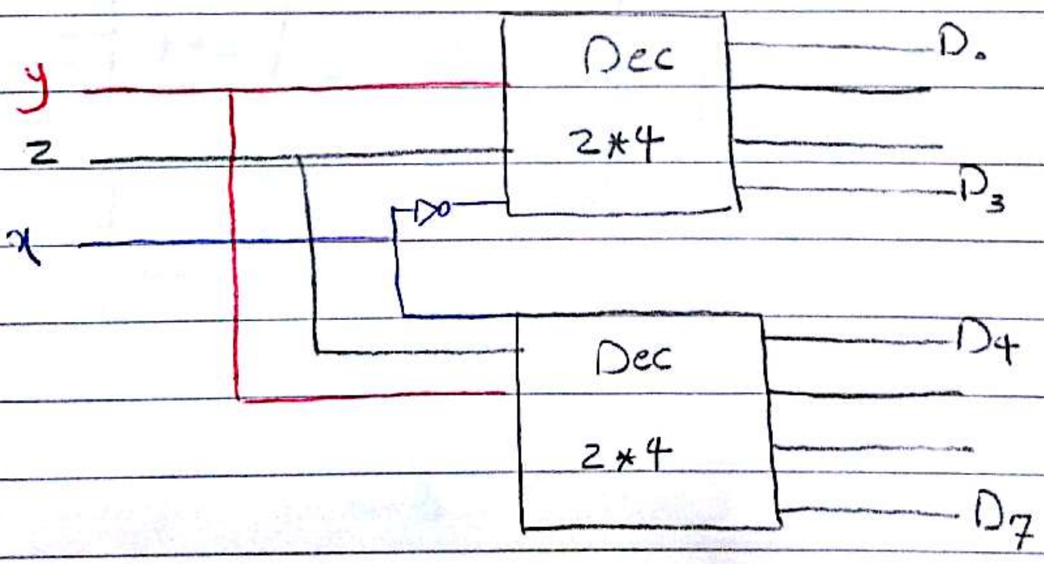
E	x	y	D_0	D_1	D_2	D_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1





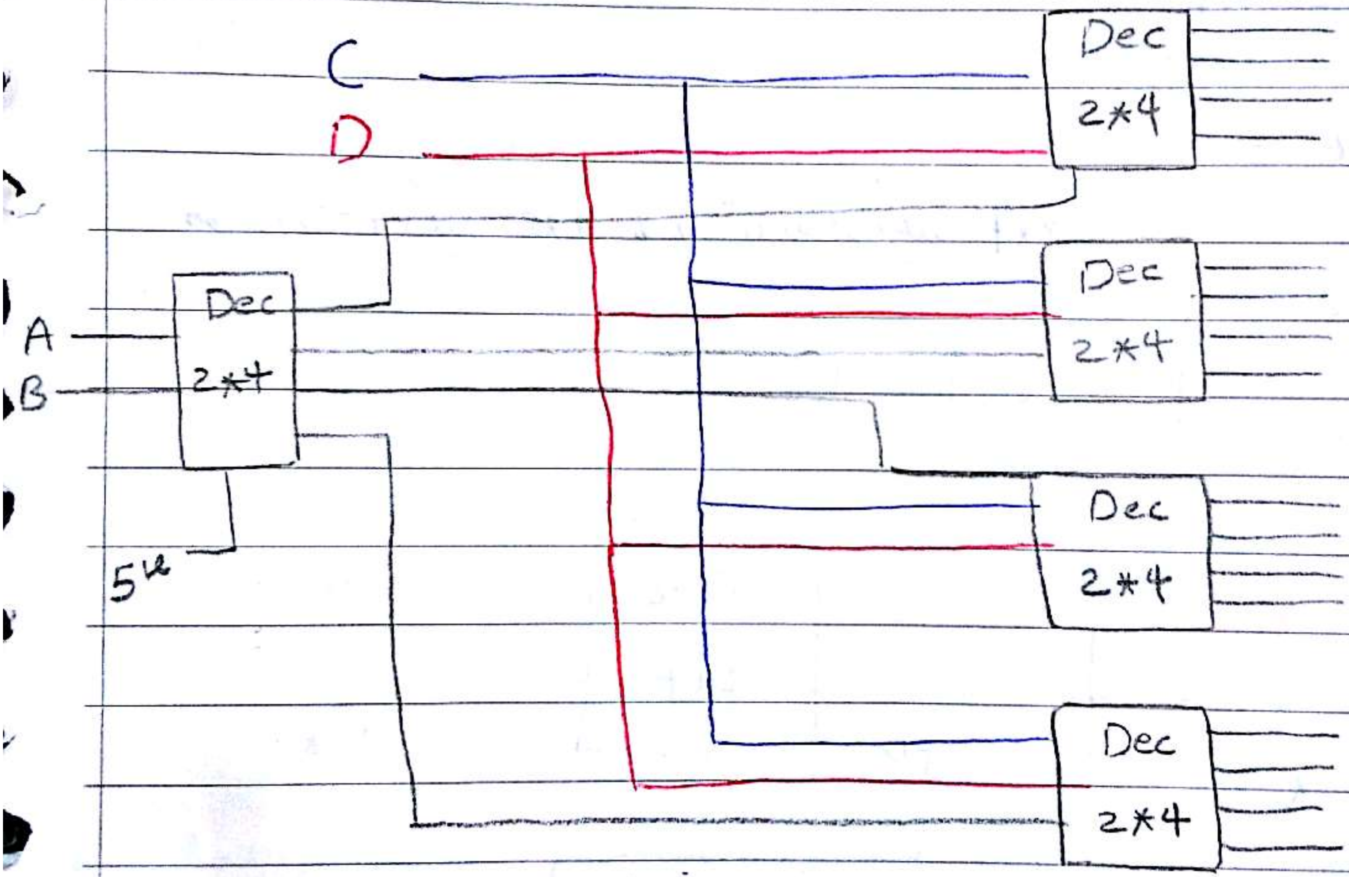
ساخت دیکدر 3*8 با استفاده از دیکدر 2*4



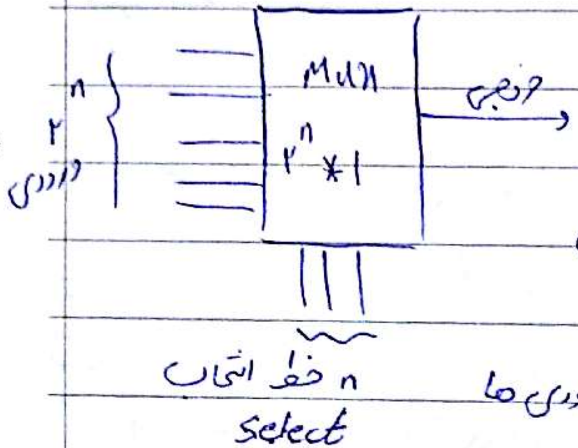


- یک دیکدر 4*16 با استفاده از دیکدرهای 2*4 بسازید.

A	B	C	D
0	0	D_0	
0	1	D_1	
1	0	D_2	
1	1	D_3	



مالتی پلکسر Multiplexer

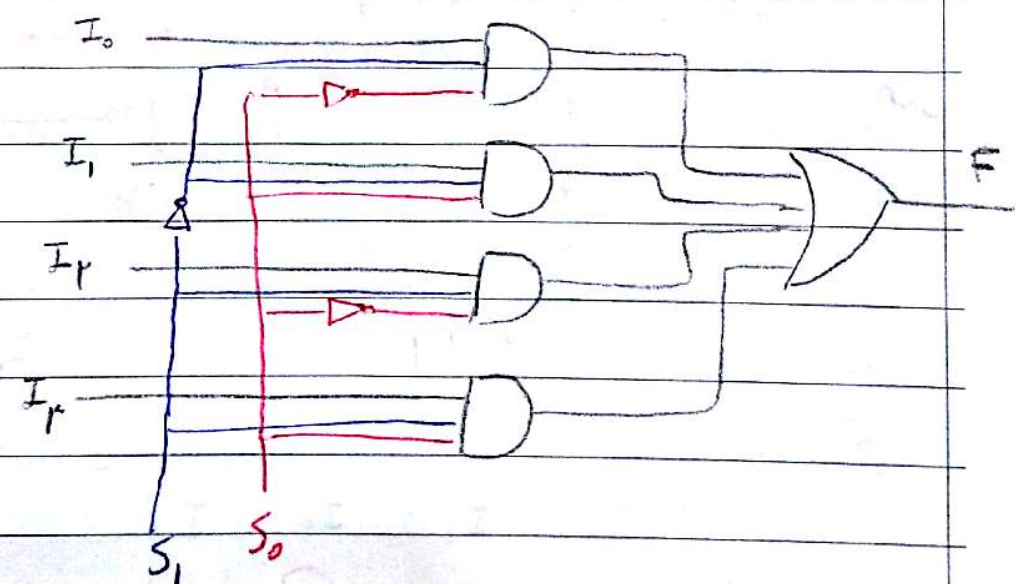


مالتی پلکسر یک مدار ترکیبی با 2^n ورودی و یک خروجی می باشد که n تعداد خطوط انتخاب می باشد براساس مقدار خط انتخاب یکی از ورودی ها به خروجی منتقل می شود و از بقیه صرف نظر می گردد.

مثال: یک مالتی پلکسر $4*1$ را طراحی و پیاده سازی کنید.

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

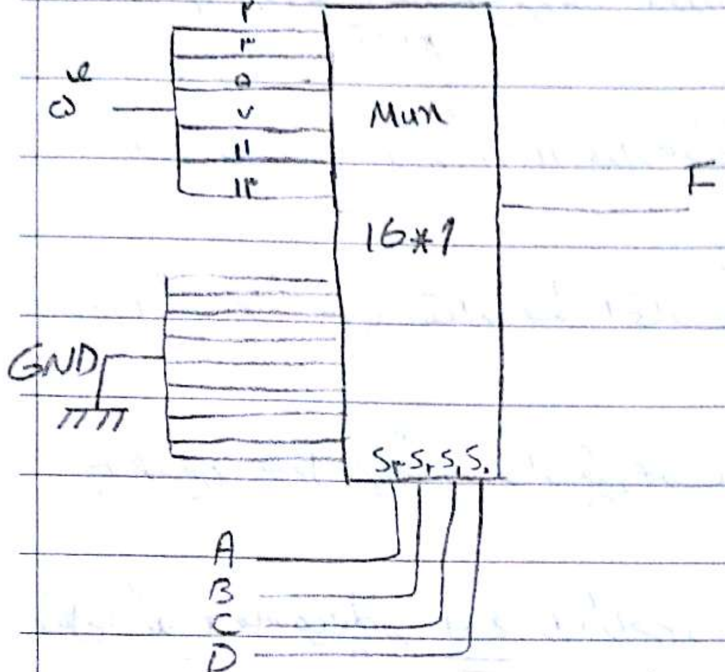
$$F = I_0 S_1' S_0' + I_1 S_1' S_0 + I_2 S_1 S_0' + I_3 S_1 S_0$$



طراحی مدارات با مالتی پلکسر

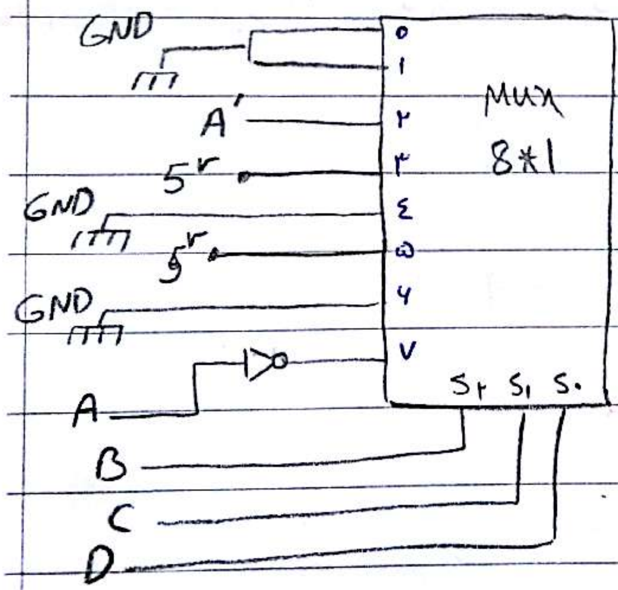
مثال: یک آشکارساز عدد اول ۴ بیتی با استفاده از مالتی پلکسر مناسب پیاده سازی کنید.

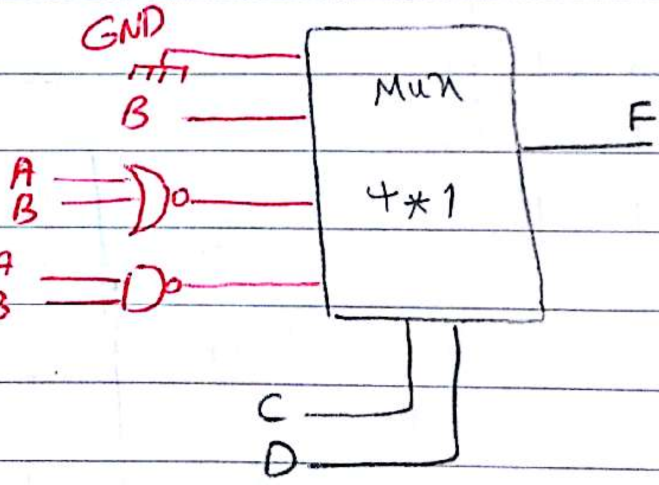
$F(A,B,C,D) = \Sigma(2,3,5,7,11,13)$ حداکثر $2^4 = 16$



جدول پیاده سازی

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A'	۰	۱	۲	۳	۴	۵	۶	۷
A	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
	۰	۰	A'	$A+A' = 1$	۰	۱	۰	A'





	I_0	I_1	I_2	I_3
A'B'	۰	۱	۲	۳
A'B	۴	۵	۶	۷
AB'	۸	۹	۱۰	۱۱
AB	۱۲	۱۳	۱۴	۱۵
	۰			

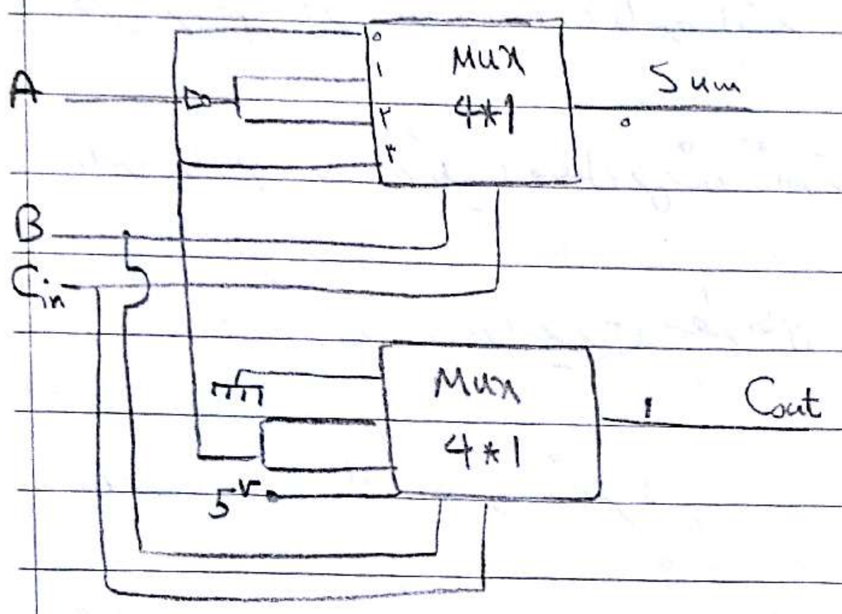
$I_1 = A'B + AB = B$

$I_2 = A'B' = (A+B)'$

$I_3 = A'B' + A'B + AB' = A' + B' = (AB)'$

مثال: یک تمام جمع کننده با استفاده از مالتی پلکسرهای ۱*۴ طراحی و پیاده سازی کنید

(A, B, C_{in}) $\quad Sum = \Sigma(1,2,4,7) \quad C_{out} = \Sigma(3,5,6,7)$



	I_0	I_1	I_2	I_3
A'	0	1	2	3
A	4	5	6	7
	A	A'	A'	A

	I_0	I_1	I_2	I_3
A'	0	1	2	3
A	4	5	6	7
	0	A	A	1

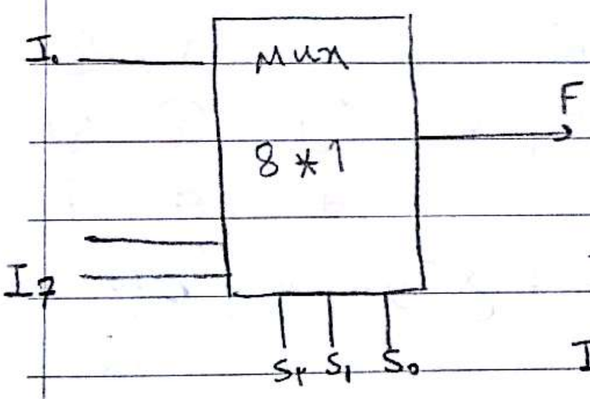
برای پیاده سازی تابع با مالتی پلکسر ترجیحاً از مالتی پلکسری استفاده می کنیم که تعداد خطوط انتخاب آن از تعداد ورودی های تابع حداقل یکی کمتر باشد سپس ورودی های تابع را از کم ارزش به پرارزش به ورودی های متناظر مالتی پلکسر متصل می کنیم و خطوط باقیمانده را به جدول زوجها می بریم سطرهای جدول با حالات متفاوت ورودی تابع که استفاده نشده پر می شود ستون های جدول با ورودی های مالتی پلکسر پر می شود

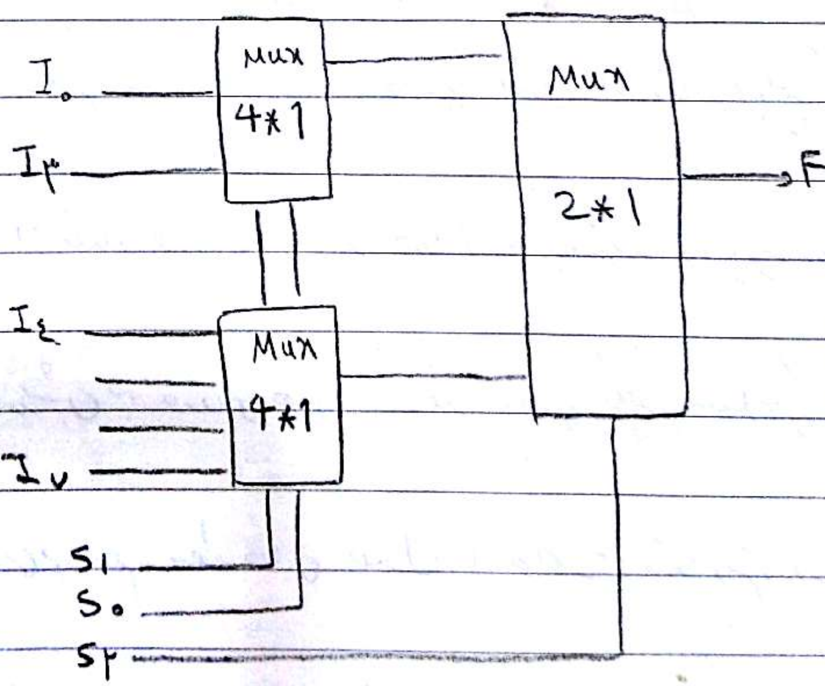
Date :

قلاقی آنها با شماره مینترم کامل می شود سپس تابع دلخواه را با دایره مشخص می کنیم اگر ستونی بدون دایره باشد به صفر منتقل می شود اگر کل ستون دایره دار می شود به یک منتقل می شود اگر تنها یک دایره باشد به سطر روبه روی آن منتقل می گردد در غیر اینصورت سطرهای دایره دار باهم OR می کنند.

ساخت مالتی پلکسر های بزرگتر :

مثال: یک مالتی پلکسر 8*1 با استفاده از مالتی پلکسرهای 4*1 یا 2*1 بسازید.





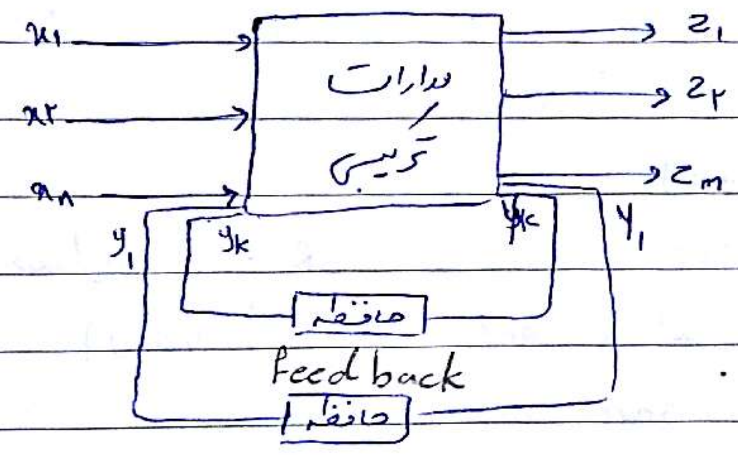
مدارات ترتیبی :

مدار ترتیبی از اضافه کردن حافظه به مدارات ترکیبی بدست می آید. مدارات ترتیبی مدارات حافظه داری باشند عنصر حافظه فلیپ فلاپ می باشد.

در مدارات ترتیبی سنکرون مدار با پالس ساعت فعال می شود در حالیکه

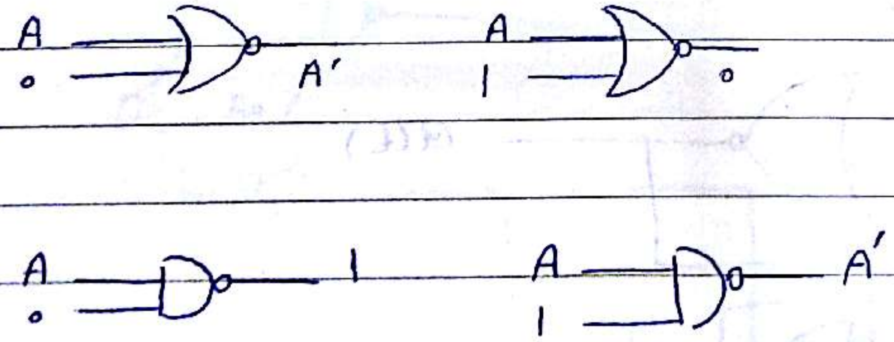
فعالسازی

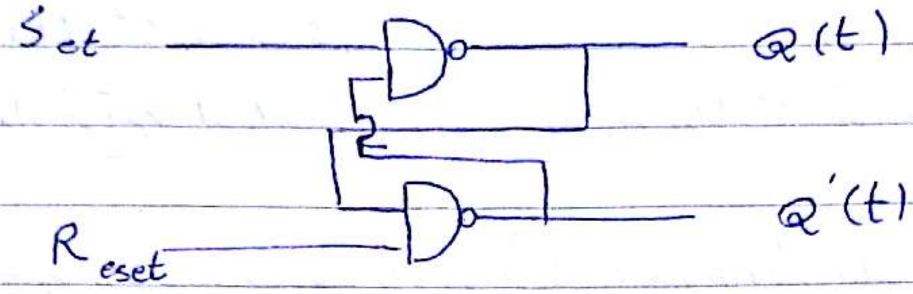
در مدارات ترتیبی آسنکرون برای ۲ مدار نیاز به ساعت نیست و بلافاصله پس از دریافت ورودی خروجی تولید می شود.



لچ SR

نگهدارنده (latch)

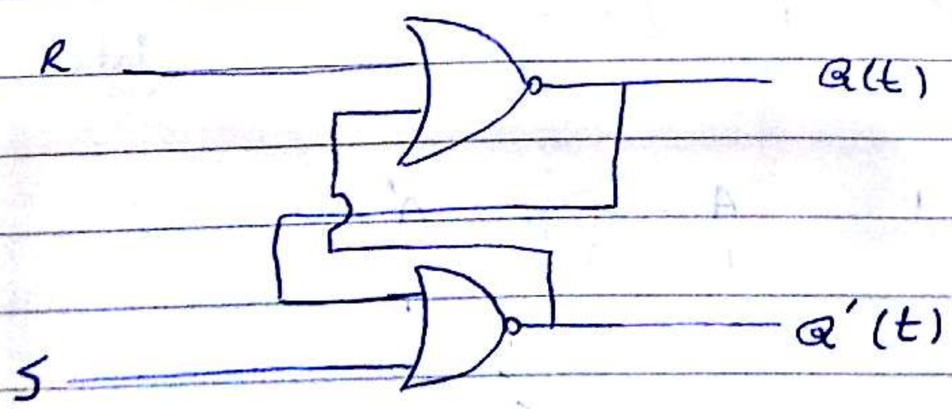




S	R	Q(t)	Q'(t)	
0	1	1	0	
1	1	1	0	
1	0	0	1	
1	1	0	1	
0	0	1	1	نامعین

جدول مشخصه لچ SR (NanD)

S	R	Q(t+1) لحظه بعد	
0	1	1	set
1	0	0	reset یا clear
1	1	Q(t)	save (ذخیره)
0	0	?	نامعین



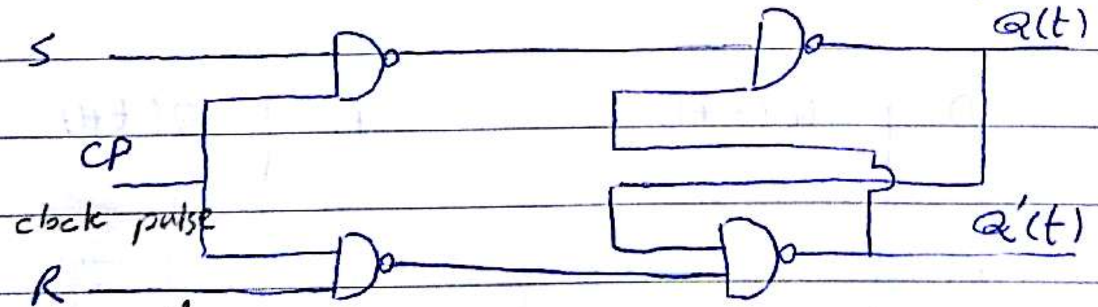
	S	R	Q(t)	Q'(t)	
	0	1	0	1	
بعد از SR=01	0	0	0	1	
	1	0	1	0	
بعد از SR=10	0	0	1	0	
	1	1	0	0	نامعین

لحظه بعد

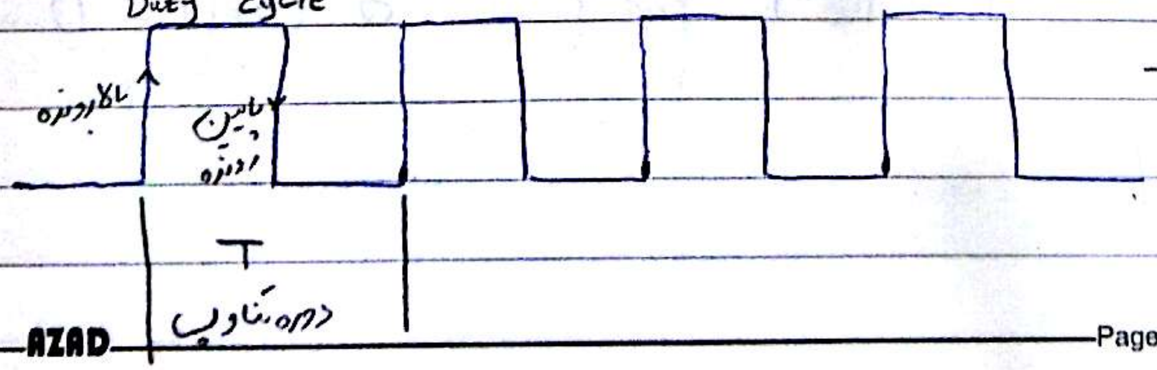
S	R	Q(t+1)	
0	0	Q(t)	save
0	1	0	reset
1	0	1	set
1	1	?	نامعین

جدول مشخصه ی SR
(Nor)

فلیپ - فلاپ SR
Flip - Flop



مثال
$D_C = 50\%$
Duty cycle



$T = \frac{1}{f}$ فرکانس

 یا 

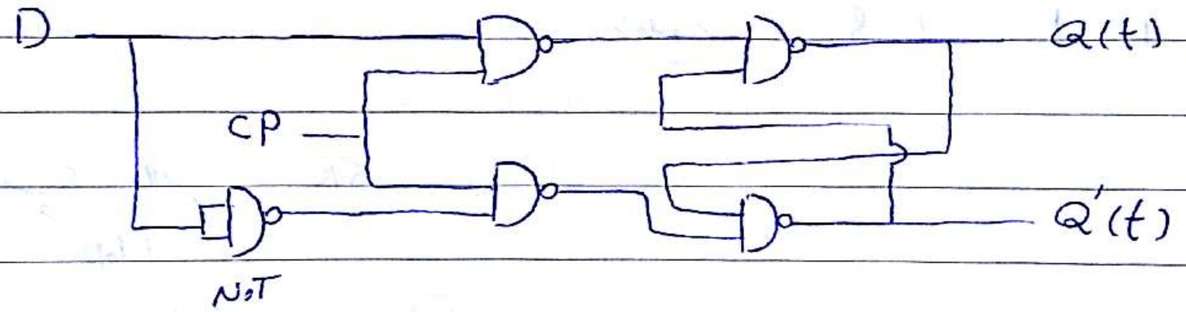
جدول مشخصه فلیپ-فلاپ SR

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

S	R	Q(t+1)	
0	0	Q(t)	save
0	1	0	reset
1	0	1	set
1	1	?	نامعین

فلیپ-فلاپ D

Data-F.F

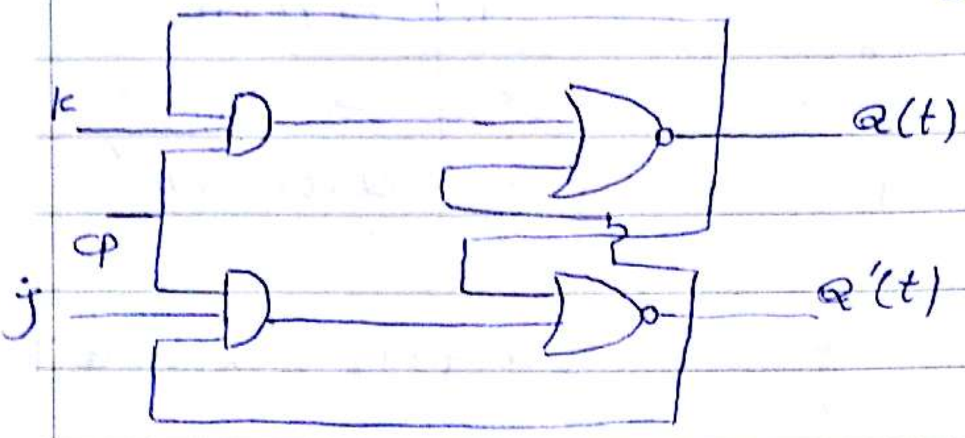


Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

D	Q(t+1)
0	0
1	1

$Q(t+1) = D$

فلیپ - فلاپ jk

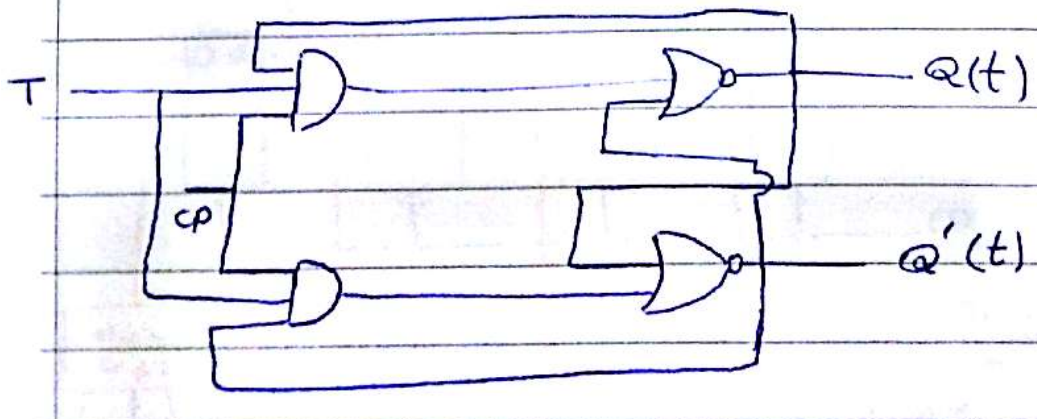


Q(t)	j	k	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

جدول مشخصه jk فلیپ فلاپ

j	k	Q(t+1)
0	0	Q(t) save
0	1	0 reset
1	0	1 set
1	1	Q'(t) مکمل complement

فلیپ فلاپ T



Date : . .

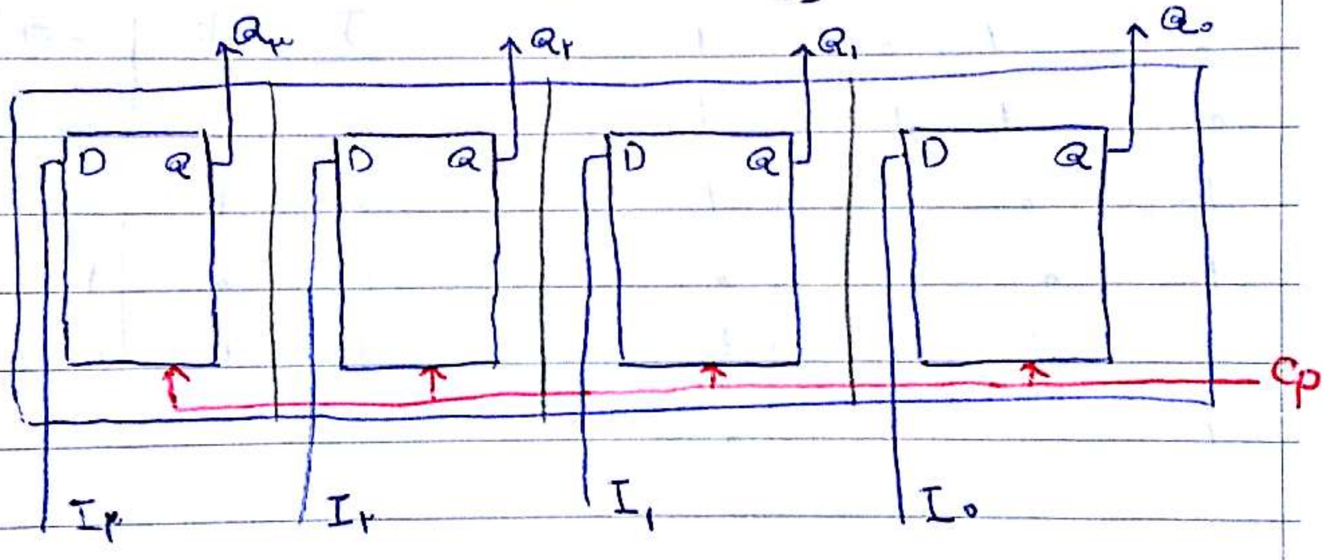
Q(t)	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

T	Q(t+1)
0	Q(t) save
1	Q'(t) مکمل

$$Q(t+1) = Q(t) \oplus T$$

ثبات Register

ساخت ثبات ٤ بیتی D_F.F

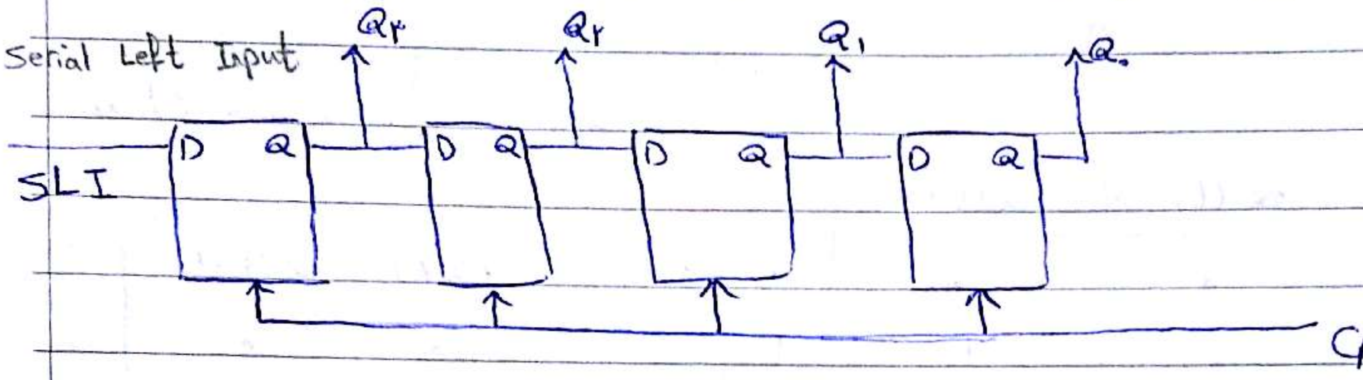


AZAD

شیفت رجیستر (Shift Register)

* به شیفتی که امکان شیفت داده به سمت چپ یا راست داشته باشد شیفت رجیستر گفته می‌شود.

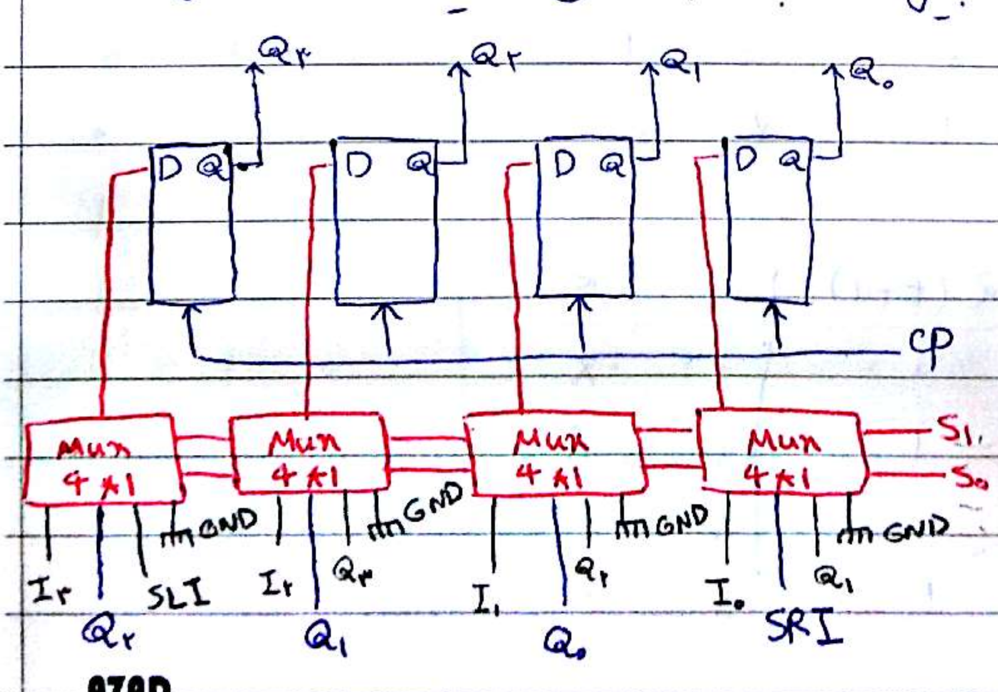
شیفت رجیستر با امکان شیفت به راست



۰۱۱۰۱۱۰۱۱۱۰۱۱۰

0 0 0 0
0 0 0 1
1 0 0 0
1 1 0 0
0 1 1 0

شیفت رجیستر ۴ بیتی دو جهته با امکان دریافت داده موازی



S_1	S_0	Option
0	0	صفر کردن بیت‌ها
0	1	شیفت به راست
1	0	شیفت به چپ
1	1	ورودی موازی

جدول مشخصه فلیپ - فلاپ ها :

T	Q(t+1)
0	Q(t)
1	Q'(t)

D	Q(t+1)
0	0
1	1

جدول تحریک فلیپ فلاپ ها :

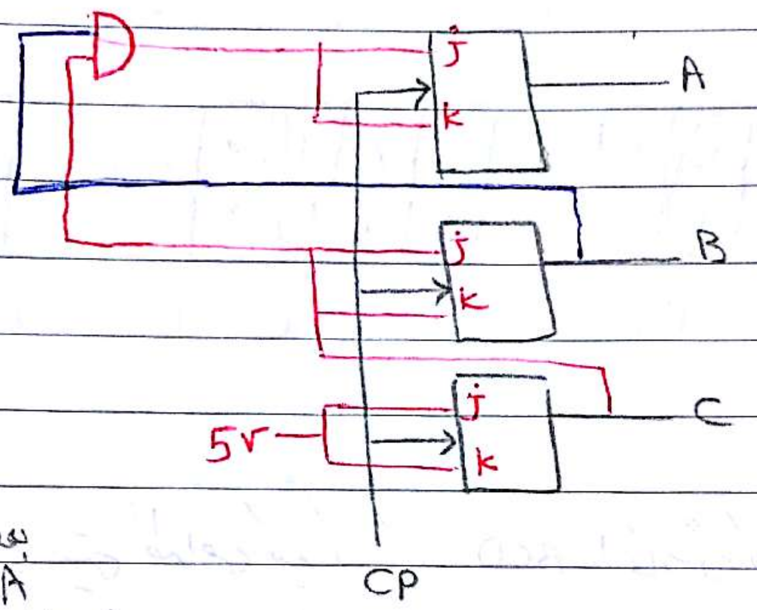
Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

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

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

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

مثال: یک شمارنده ۳ بیتی طراحی کنید که اعداد باینری ۳ بیتی را شمارش کند از فلیپ فلاپ های J K استفاده کنید.



قبلی بعدی

$$J_A = \underset{Q(t)}{A} \times \underset{Q(t+1)}{A}$$

حالات فعلی			حالات بعدی								
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

0	0	1	0
X	X	X	X

$J_A = BC$

0	1	X	X
0	1	X	X

$J_B = C$

1	X	X	1
1	X	X	1

$J_C = 1$

X	X	X	X
0	0	1	0

$K_A = BC$

X	X	1	0
X	X	1	0

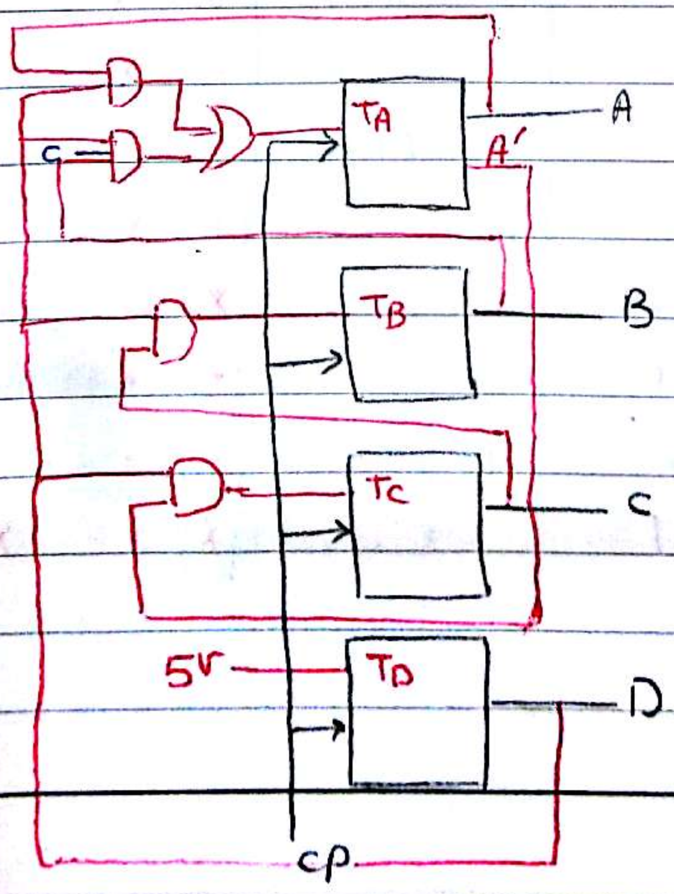
$K_B = C$

X	1	1	X
X	1	X	X

$K_C = 1$

مثال: یک شمارنده ۴ بیتی طراحی کنید که کد BCD را شمارش کند. از فلیپ فلاپ های نوع T استفاده کنید.

T	$Q(t+1)$
0	$Q(t)$
1	$Q'(t)$



Date :

حالت فعلی				حالت بعدی				ورودی فلیپ فلاپ ها			
A	B	C	D	A	B	C	D	T_A	T_B	T_C	T_D
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

0	0	0	0
0	0	1	0
X	X	X	X
0	1	X	X

$T_A = AD + BCD$

0	0	1	0
0	0	1	0
X	X	X	X
0	0	X	X

$T_B = CD$

0	1	1	0
0	1	1	0
X	X	X	X
0	0	X	X

$T_D = 1$

$T_C = A'D$