مدار یک
۸:۰۰-۹:۴۵
Subject . ۹۸/۷/۹

بسمه تعالی

Year. Month. Day.

مدار منطقی

* سیستم های اعداد

* جبر بولی — میانترم *

مدار منطقی → مدار ترکیبی / مدارهای ترتیبی — پایانترم

* ساده سازی با جدول کارنو

www.shafieian_education.ir

میانترم : ۸ نمره
پایانترم : ۱۰ نمره
تکلیف : ۲ نمره

مدار منطقی ← موریس مانو < دکتر دیانی / دکتر حسن سیدرضی
مدار منطقی نلسون

نیازمندی
↓
رفتار ← توصیف سیستم
↓
طراحی مدار ← مدار منطقی (گیت و ...) | مدار منطقی
↓
پیاده سازی مدار با عناصر الکترونیک ← (ترانزیستور، دیود، ...) | الکترونیک دیجیتال
↓
فیزیک الکترونیک

Subject . Year. Month. Day.

فصل اول . سیستم های اعداد

مبنای ۱۰ $\rightarrow (5672)_{10} = 5\times10^3 + 6\times10^2 + 7\times10^1 + 2\times10^0$

یک عدد در مبنای ۱۰ $\longrightarrow$ ۰-۹

یک عدد در مبنای ۲ $\longrightarrow$ ۰ - (۲-۱)

$$*\ (a_5\, a_4\, a_3\, a_2\, a_1\, a_0 . a_{-1}\, a_{-2}\, a_{-3})_{10} = a_5\times10^5 + a_4\times10^4 + \dots + a_0\times10^0 + a_{-1}\times10^{-1} + a_{-2}\times10^{-2} + a_{-3}\times10^{-3}$$

یک عدد در مبنای 10 مثل 5672 مقداری معادل 5 هزارتایی 6 صدتایی به علاوه 7 ده تایی و 2 یکی را نشان می‌دهد.

درمبنای ۱۰ بالاترین رقم 9 میباشد یعنی درمبنای 10 ارقام صفرتا نه وجود دارد.

بنابراین نمایش کلی یک عدد درمبنای را داریم: *

که درآن a_j هایی از ارقام (۰-۹) میباشند واندیس j ارزش مکان آن رقم را نشان می‌دهد.

$0.7 = \frac{7}{10} = 7\times10^{-1}$

$0.75 = \frac{75}{100} = \frac{7}{10} + \frac{5}{100} = 7\times10^{-1} + 5\times10^{-2}$

$(23.7)_{10} = 2\times10^1 + 3\times10^0 + 7\times10^{-1}$

1 0 -1

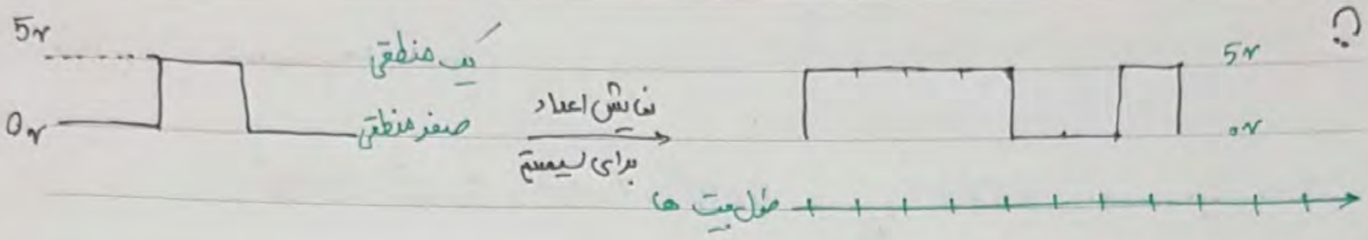
درسیستم اعداد باینری فقط دو مقدار صفر و یک به کار میرود.

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

$(11110.01)_2 = 1\times2^4 + 1\times2^3 + 1\times2^2 + 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}$

$= (30.25)_{10}$

Ganjineh



به طور کلی یک عدد در مبنای r به صورت حاصل ضرب توان های r در ضرائب مربوطه به صورت زیر نوشته میشود.

$$(a_n a_{n-1} a_{n-2} \cdots a_1 a_0 . a_{-1} a_{-2} a_{-3})_r = (a_n \times r^n + a_{n-1} \times r^{n-1} + \cdots + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} + a_{-3} r^{-3})_{10}$$

مثال:

$$(4121.2)_5 = 4\times5^3 + 1\times5^2 + 2\times5^1 + 1\times5^0 + 2\times5^{-1} = (636.4)_{10}$$

3 2 1 0 -1

درصورتی که سیستم اعداد دارای مبنای بزرگتر از 10 باشد برای رقم های بالاتر از 10 از حروف انگلیسی استفاده می شود. به عنوان مثال برای اعداد در مبنای 16 برای نمایش ارقام 10 _ 11 _ ... _ 15 به ترتیب از حروف A تا F استفاده می شود.

$10 \rightarrow A \quad 11 \rightarrow B \quad 12 \rightarrow C \quad 13 \rightarrow D \quad 14 \rightarrow E \quad 15 \rightarrow F$

$$\overset{3}{A}\,\overset{2}{6}\,\overset{1}{5}\,\overset{0}{E}$$

$$(A65E)_{16} = 10\times16^3 + 6\times16^2 + 5\times16^1 + 14\times10^0 = (40960 + 1536 + 80 + 14)_{10}$$

H

$$(630.5)_8 = (6\times8^2 + 3\times8^1 + 0\times8^0 + 5\times8^{-1})_{10}$$

تبدیل از مبنای غیر ۱۰ به ۱۰ قبلا بیان شد.

تبدیل از مبنای ۱۰ به غیر ۱۰ به صورت روبرو است.

$(37)_{10} = (100101)_2$

$(154)_{10} = (232)_8$

برای تبدیل یک عدد اعشاری در مبنای ۱۰ به مبنای غیر ۱۰ به جای تقسیم از ضرب به صورت زیر استفاده می‌کنیم.

$(0.6875)_{10} = (0.1011)_2$

$0.6875 \times 2 = 1.375 = \underbrace{1}_{a_{-1}} + 0.375$

$0.375 \times 2 = 0.75 = \underbrace{0}_{a_{-2}} + 0.75$

$0.75 \times 2 = 1.5 = \underbrace{1}_{a_{-3}} + 0.5$

$0.5 \times 2 = 1 = \underbrace{1}_{a_{-4}} + 0.0$

تبدیل قسمت کسری عدد در مبنای ۱۰ به مبنای ۸ شبیه روش بالا می‌باشد با این تفاوت که به جای ضرب قسمت کسری در 2 آن را در ۸ ضرب می‌کنیم.

$(0.513)_{10} = (0.4064)_8$

$0.513 \times 8 = 4.104 = \underbrace{4}_{a_{-1}} + 0.104$ $\quad$ $0.104 \times 8 = 0.832 = \underbrace{0}_{a_{-2}} + 0.832$

$0.832 \times 8 = 6.565 = \underbrace{6}_{a_{-3}} + 0.565$ $\quad$ $0.565 \times 8 = 4.52 = \underbrace{4}_{a_{-4}} + 0.52$

...

برای تبدیل عدد در مبنای ۱۰ به ۸ قسمت صحیح و کسری با بید قسمت های صحیح و کسری را به طور جدا تبدیل و سپس دو جواب را با هم ترکیب نمود.

$(151.513)_{10} = (227.4064)_8$

$$\begin{array}{r|l} 151 & 8 \\ \underline{144} & 18 \;|\; 8 \\ 7 & \underline{16} \;\; 2 \\ & 2 \end{array}$$

تبدیل مبنای هشت و شانزده: هر رقم در مبنای هشت معادل سه بیت عدد باینری می باشد و هر رقم در مبنای 16 معادل 4 بیت عدد باینری می باشد.

تبدیل از مبنای 2 به مبنای 8 با گروه کردن عدد باینری به دسته های 3 تایی از ممیز به چپ و راست حاصل میشود و به جای هر سه بیت عدد باینری یک رقم معادل در مبنای 8 قرار میگیرد.

$(\underbrace{010}_{2}\underbrace{110}_{6}.\underbrace{111}_{7}\underbrace{000}_{0})_2 = (26.70)_8$

$(\underbrace{0001}_{1}\underbrace{0110}_{6}.\underbrace{1110}_{E})_2 = (16.E)_{16}$

تبدیل عدد از مبنای 8 , 16 به مبنای 2 با روش عکس عمل بالا انجام میشود و هر رقم در مبنای 8 به 3 رقم باینری معادل آن و در مبنای 16 به چهار رقم باینری معادل آن تبدیل می شود.

$(7\,3.2)_8 = (111\,011.010)_2$ ($7 \to 111$, $3 \to 011$, $2 \to 010$)

$(307.D) = (0011\,0000\,0111.1101)_2$ ($3 \to 0011$, $0 \to 0000$, $7 \to 0111$, $D \to 1101$)

عملیات +، −، ×، ÷ دو عدد باینری :

جمع :

$$\begin{array}{r} \scriptstyle 1\,1\,1\quad 1 \\ 1011101 \\ +\ 0111101 \\ \hline 10011010 \end{array}$$

تفریق :

$(10) \cong 2$

$$\begin{array}{r} 1011\cancel{1}\cancel{0} \\ -\ 100101 \\ \hline 001001 \end{array} \qquad \begin{array}{r} 0\,\cancel{2}\,\cancel{2}\,2 \\ \cancel{1}\,\cancel{0}\,\cancel{0}\,\cancel{0} \\ -\ 111 \\ \hline 0001 \end{array}$$

ضرب :

$$\begin{array}{r} 1101 \\ \times\ 101 \\ \hline 1101 \\ +\ 00000 \\ +\ 110100 \\ \hline 1000001 \end{array}$$

سار۲ / ۹۸٫۷٫۱۰ / ۸:۰۰-۹:۴۵

مکمل های اعداد : اگر عدد N در پایه r شامل n رقم باشد مکمل $r-1$ عدد طبق تعریف برابر است با

$(r^n - 1) - N =$ مکمل $r-1$

در سیستم اعداد دهدهی

$10^1 = 10 \rightarrow 10^1 - 1 = 9$

$10^2 = 100 \rightarrow 10^2 - 1 = 99$

$10^3 = 1000 \rightarrow 10^3 - 1 = 999$

$\vdots$

$10^n = \underbrace{1000\ldots0}_{n\text{ رقم}} \rightarrow 10^n - 1 = \underbrace{99\ldots9}_{n\text{ تا}}$

برای بدست آوردن مکمل ۹ عدد N بایستی تک تک ارقام N را از ۹ کم کنیم.

$$\underset{n \text{ رقم}}{N} \longrightarrow (r^n-1)-N = (\underbrace{10^n-1}_{n \text{ تا } 9})-N$$

$$N=379 \xrightarrow{\text{مکمل } 9} 999-379=620$$

دراعداد باینری $r=2$ و $r-1=1$ است پس مکمل یک عدد N برابر است با : $(2^n-1)-N$

$$r=2 \longrightarrow r^0=2^0=1 \longrightarrow r^0-1=1-1=0$$

$$r^1=2^1=10 \longrightarrow r^1-1=1$$

$$r^2=2^2=100 \longrightarrow r^2-1=11$$

$$r^3=2^3=1000 \longrightarrow r^3-1=111$$

$$\vdots$$

$$r^n=2^n=1\underbrace{000\ldots0}_{n \text{ رقم}} \longrightarrow r^n-1=\underbrace{111\ldots1}_{n \text{ رقم یک}}$$

$$\left.\begin{array}{l} r=2 \\ \text{مکمل } r-1 \end{array}\right\} \longrightarrow (\underbrace{r^n-1}_{n \text{ تا یک}})-N \longrightarrow \text{کم کردن تک تک ارقام از یک}$$

$N \leftarrow n$ رقم

برای محاسبه مکمل یک عدد باینری N با تبدیل صفرها به یک و یک ها به صفر بدست می آید.

در نتیجه مکمل $r-1$ اعداد در مبنای 8 و 16 با تفریق هر رقم آنها به ترتیب از 7 و F حاصل میشود.

مکمل r اعداد: طبق تعریف مکمل r عدد n رقمی در پایه r برابر است با:

$r^n - N$

$(r^n - 1) - N + 1$

بنابراین مکمل r عدد برابر است با مکمل r-1 بعلاوه یک؛ بنابراین مکمل 10 هر عدد دهدهی برابر با مکمل 9 آن بعلاوه عدد یک میباشد، همچنین مکمل 2 هر عدد باینری برابر با مکمل یک آن بعلاوه عدد یک میباشد.

از آنجا که نمایش عدد 10^n به صورت یک 1 و n 0 هست راست آن میباشد تا مکمل 10 هر عدد N که برابر با $10^n - N$ است را میتوان بصورت زیر بدست آورد.

از سمت راست کم ارزش ترین رقم های صفر بدون تغییر میماند، اولین رقم سمت راست که صفر نیست از 10 تفریق میگردد و بقیه ارقام از 9 تفریق میشوند.

$N = 17800 \longrightarrow$ مکمل 10 = ?

(1) مکمل 9 عدد N ←

$$\begin{array}{r} 99999 \\ -\ 17800 \\ \hline 82199 \\ +\ \ \ \ \ \ 1 \\ \hline 82200 \end{array}$$

(2) مکمل 10 = 82200 $\xrightarrow{\text{چون}}$ $10^5 - N \Rightarrow$

$$\begin{array}{r} 100000 \\ 17800 \\ \hline 82200 \end{array}$$

به طور مشابه مکمل 2 اعداد باینری نیز بصورت زیر بدست میاید.

از سمت راست کم ارزش ترین رقم های صفر بدون تغییر می مانند، اولین رقم یک از سمت راست بدون تغییر مانده یا به عبارتی از 2 کم میشود، در بقیه رقم ها صفرها به یک و یک ها به صفر تبدیل میشوند.

مکمل 1 عدد N ← 01011

$+\ \ \ \ 1$

مکمل 2 عدد N ← 01100

$N = 10100 \longrightarrow$ مکمل 2 = ?

$N = 10100 \rightarrow$ مکمل 2 = 01100

$r^n - N = 2^5 - N = 100000 - N \rightsquigarrow$

$$\begin{array}{r} 100000 \\ -\ 10100 \\ \hline 01100 \end{array}$$

Subject . Year. Month. Day.

نکته اول: اگر عدد N دارای ممیز باشد به طور موقت ممیز را حذف کرده و مکمل r یا r-1 آن را بدست می‌آوریم سپس ممیز را در همان مکان نسبی اولیه قرار میدهیم.

نکته دوم: مکمل مکمل عدد برابر خود عدد است.

$$N \xrightarrow{\text{مکمل } r} r^n - N \xrightarrow{\text{مکمل } r} r^n - (r^n - N) = N$$

تفریق به روش مکمل ها:

استفاده از مکمل ها نسبت به روش قرضی در هنگام عمل تفریق برای ماشین راحت تر است.
تفریق دو عدد n رقمی $M - N$ را که بی علامت باشد در پایه r میتوان به روش زیر انجام داد.

1. به مفروق منه M مکمل r مفروق N اضافه میشود $M + (r^n - N) = M - N + r^n$

2. اگر $M \geq N$ باشد مجموع حاصل یک بیت نقلی نهایی r^n تولید خواهد کرد که از آن صرف نظر میشود.
در نتیجه تنها $M - N$ باقی میماند

3. اگر $N > M$ باشد مجموع ما بیت نقلی نهایی تولید نمیکند و برابر $r^n - (N - M)$ می‌شود. یعنی مکمل r عدد $N - M$ می‌باشد پس برای بدست آوردن جواب به صورت معمولی میتوان مکمل r مجموع را بدست آورد و یک علامت منفی جلوی آن قرار داد.

با به کار بردن مکمل 10 تفریق های زیر را انجام دهید.

$$\begin{array}{r} 8252 \\ -\ 3260 \\ \hline \end{array} \Rightarrow \begin{array}{r} 8252 \\ +\ 6740 \\ \hline \cancel{1}4992 \end{array}$$

$2532 - 0420 = ?$

$$\begin{array}{r} 2532 \\ +\ 9580 \\ \hline \cancel{1}2112 \end{array}$$

$4550 - 7532 = ?$

$$\begin{array}{r} 4550 \\ +2468 \\ \hline 7018 \end{array}$$

مکمل 10 عدد $N-M$ → حاصل $= -2982$

$9532.6 - 0722.8 = ?$ →

$$\begin{array}{r} 9532.6 \\ +\ 9277.2 \\ \hline 1\,8809.8 \end{array}$$

با بکار بردن مکمل 2 تفریق های باینری زیر را انجام دهید.

$X = 11001$ $\quad X - Y = ?$ →

$Y = 10011$ $\quad Y - X = ?$

$$\begin{array}{r} 11001 \\ +01101 \\ \hline 1\,00110 \end{array} \qquad \begin{array}{r} 10011 \\ +00111 \\ \hline 11010 \end{array}$$

مکمل 2 عدد $Y-X$

↓ مکمل 2

-00110

تفریق به روش r-1 :

$$\begin{array}{r} 8252 \\ -3260 \\ \hline ? \end{array}$$

تفریق به روش مکمل r-1 →

$$\begin{array}{r} 8252 \\ +6739 \\ \hline 1\,4991 \\ +\quad 1 \\ \hline 4992 \end{array}$$

6739 ← مکمل 9 عدد N

$$M - N = M + \left((r^n - 1) - N\right) = M - N + (r^n - 1) = (M - N) + r^n - 1$$

$$\begin{array}{r} 4550 \\ -7532 \\ \hline ? \end{array} \longrightarrow \begin{array}{r} 4550 \\ +2467 \\ \hline 7017 \end{array}$$

7017 → مکمل 9 عدد $N-M$

↓ مکمل 9

-2982

اعداد باینری علامت دار: در ریاضیات معمولی اعداد منفی را با علامت منها و اعداد مثبت را با علامت بعلاوه نشان داده میشوند اما در کامپیوتر هر چیز را باید بوسیله رقم های باینری نشان داد.

اعداد باینری را به دو صورت با علامت یا بدون علامت میتوان نشان داد. در اعداد با علامت معمول این است که سمت چپ ترین بیت به عنوان بیت علامت شناخته میشود که اگر صفر باشد معرف عدد مثبت و اگر یک باشد معرف عدد منفی است ولی در اعداد بدون علامت بیت سمت چپ پرارزش ترین بیت در آن عدد میباشد.

MSB: most significant Bit

LSB: least significant Bit

به عنوان مثال مجموعه اعداد 8 بیتی در سیستم اعداد بدون علامت به صورت زیر تعریف میشود:

0= 00000000 ⟶ 11111111 = 255

MSB MSB

اما مجموعه اعداد 8 بیتی در سیستم مقدار علامت به صورت زیر تعریف میشود:

بیت علامت بیت علامت

11111111 ⟶ 01111111

-127 +127

مدار منطقی ۳

00000000

11111111 $\xrightarrow{\text{بی علامت}}$ 255
msb $\xrightarrow{\text{با علامت}}$ -127

01111111 $\xrightarrow{\text{علامت}}$ +127

برای نمایش اعداد منفی سیستم دیگری به نام مکمل - علامت وجود دارد که در آن برای نمایش منفی یک عدد از مکمل یک یا مکمل دو آن عدد استفاده می‌شود. چون اعداد مثبت همیشه با صفر درست چپشان شروع می‌شوند لذا مکمل آنها همواره با یک از سمت چپشان آغاز می‌گردد که بیانگر عدد منفی است.

(8 بیت)

9 در سیستم اعداد بی علامت → 00001001

9 در سیستم اعداد علامت دار → 00001001

-9 در سیستم اعداد علامت دار →
- مقدار علامت → 10001001
- مکمل یک → 11110110
- مکمل دو → 11110111

جمع حسابی در اعداد با علامت:

جمع دو عدد باینری در سیستم مقدار - علامت مانند جمع معمولی حسابی است به این طریق که اگر علامت دو عدد یکسان باشد ما دو مقدار آنها را با هم جمع می‌کنیم و علامت مشترک را در نظر می‌گیریم. در صورتی که علامت دو عدد مختلف باشد مقدار کوچکتر را از مقدار بزرگتر تفریق می‌کنیم و علامت بزرگتر را برای نتیجه اختیار می‌کنیم.

$$\begin{array}{r} -13 \\ +\ -9 \\ \hline ? \end{array} \longrightarrow \begin{array}{r} 10001101 \\ +\ 10001001 \\ \hline 100010110 \end{array} \qquad \begin{array}{r} 13 \\ +\ -9 \\ \hline ? \end{array} \longrightarrow \begin{array}{r} 00001101 \\ -\ 10001001 \\ \hline 00000100 \end{array}$$

1

اصلش اینه که علامت رو تو عمل جمع در نظر نمیگیریم و بعد بیت علامت اعداد بیت علامت رو میذاریم.

علامت عدد بزرگتر را در نظر میگیریم نه حاصل تفریق آنها را!

Subject . ______ Year. ____ Month. ____ Day. ____

برای جمع اعداد در سیستم مکمل ـ علامت نیازی به مقایسه یا تفریق نیست بلکه فقط عمل جمع کافیست. برای جمع دو عدد باینری با علامت که در آن اعداد منفی با مکمل دو نشان داده شده‌اند باید دو عدد شامل بیت علامتشان را با هم جمع نمود و بیت نقلی خارج شده از بیت علامت را صرف نظر کرد.

```
 -6   --+6-->  00000110  --مکمل 2-->     11111010
+13   --+13->  00001101  ------->     + 00001101
----                                  ----------
  ?                                   ~~1~~00000111
```

```
                                           111
 -6   --+6-->  00000110  --مکمل 2-->     11111010
-13   --+13->  00001101  --مکمل 2-->   + 11110011
----                                  ----------
  ?                                   ~~1~~11101101   : مکمل 2 حاصل
                          مکمل 2 محاسبه می‌شود
                                        00010011 = (19)_10
```

وقتی مکمل 2 بدست میاد علامت رو میبینیم و وقتی دوباره مکمل 2 میگیریم ازش مقدار حاصل بدست میاد و لزومی نداره که حتی علامت درست نشون داده شد باشه چون ما داریم برعکس میکنیم همشو

تفریق حسابی: برای تفریق حسابی در سیستم مکمل 2، مکمل 2 مفروق (شامل بیت علامت) را به مفروق‌منه (شامل بیت علامت) اضافه و بیت نقلی خارج شده از بیت علامت را صرف نظر می‌کنیم.

```
-19 در سیستم مکمل 2  -->  11101101          -6            11111010
                                         - -19    -->  + 00010011
                                         -----           ----------
                                            ?           ~~1~~00001101
  1
  10
  101
+ 111
+ 111
+ 111
-----
11010
```

Subject . Year. Month. Day.

تقسیم اعداد باینری: در تقسیم اعداد باینری اعشاری
ها مثل معادل دهدهی عمل میکنیم ولی در تقسیم اعداد اعشاری
علاوه بر اینکه عدد باینری ها شد اعداد اعشاری تقسیم شود
در نهایت به تعداد عددی که به سمت چپ شیفت میدهیم تا باعث
از بین رفتن اعشار در مقسوم علیه شویم به همان تعداد
باقی مانده عدد را به سمت راست شیفت میدهیم ولی خارج قسمت
همان خواهد بود.

```
1110111 | 1001
1001      1101   خارج قسمت
-----
01011
 1001
-----
 001011
   1001
   ----
   0010   باقیمانده
```

```
11111111 | 101
101        110011
----
0101
 101
-----
000111
   101
   ----
   0101
    101
    ---
    000
```

110111 0111.101 / 111.01

```
1101110111 0.1 | 11101
-011101          011110100   خارج قسمت
-------
 110100
 -11101
 ------
  101111
 - 11101
 -------
  100101
 - 11101
 -------
  0 1000 11
  - 11101
  ---------
   001 10 0.1
       ↓
   0011.001  ← باقیمانده
```

Subject . Year. Month. Day.

```
11111.1010 | 1001.11

1111110.10 | 100111
- 100111   | 110
---------
  0110000
-  100111
---------
  0010011
```

کدهای باینری :

با توجه به اینکه اعدادی که بین قسمت های مختلف کامپیوتر رد و بدل میشوند مجموعه‌ای از صفر و یک هستند و دستگاه ورودی مجبور است اعداد گرفته شده دهدهی را به مجموعه‌ای از صفر و یک تبدیل نماید به برای کامپیوتر و مدارهای دیجیتال قابل فهم باشد لذا از کدهای باینری استفاده می‌شود.

۰ → 0000
۱ → 0001
۲ → 0010
۳ → 0011
۴ → 0100
⋮
۸ → 1000
۹ → 1001

کدهای دهدهی : به منظور کد کردن رقم های دهدهی (۰ تا ۹) حداقل به چهار بیت نیاز می‌باشد. انواع مختلف این کدها عبارتند از :

1. BCD : Binary coded decimal : (8 4 2 1)

برای مثال $1001 = (1\times8) + (0\times4) + (0\times2) + (1\times1) = 9$

2. BCD+3

برای مثال $0 = 0011$, $1 = 0100$, $2 = 0101$, ...

3. (8 4 -2 -1)

$0 = 0000$, $\underbrace{1 = 0111}_{(0\times8)+(1\times4)+(1\times-2)+(1\times-1)=1}$, $2 = 0110$, ... , $9 = 1111$

4. (2 4 2 1)

$0 = 0000$, $1 = 0001$, ... , $9 = 1111$

5. (5 0 4 3 2 1 0)

در این روش همواره دو بیت باید یک باشند.

$0 = 0100001$, $1 = 0100010$, ... , $9 = 1010000$

* از روش سوم به بعد اعداد باینری حاصل خودمکمل اند یعنی با جابجا کردن صفرها با یک ها به مکمل عدد در مبنای اصلیش دست خواهد آمد.

مثال　　کد

395 →	Binary	110001011
	BCD	001110010101
	BCD+3	011011001000

سیستم دو - پنجی (quintaly) (5 0 4 3 2 1 0) چون از پنج تا صفر و دو تا یک تشکیل شده است قابلیت تشخیص خطا را دارد. به عبارت دیگر اگر به اشتباه صفر به یک و یا یک به صفر تبدیل شود آن را تشخیص می‌دهد.

کدهای آشکارسازی خطا: هدف این کدها این است که تغییر بیت یا خطای احتمالی ایجاد شده در اطلاعات باینری را به هنگام انتقال تشخیص دهد. یکی از معمول‌ترین راه‌ها برای این منظور استفاده از بیت توازن می‌باشد.

بیت توازن بیتی است اضافی که جزئی از پیام بوده و طوری به پیام اضافه می‌شود که تعداد کل یک‌های انتقالی فرد یا زوج شود.

کدهای تشخیص خطا:
- توازن ساده (Simple Parity)
- کد همینگ (Haming code)

کد توازن ساده:
- بیت توازن فرد (odd Parity)
- بیت توازن زوج (Even Parity)

بیت توازن فرد: بیتی است اضافی که جزئی از پیام بوده و سبب می‌شود که تعداد کل یک‌ها در پیام فرد باشد.
بیت توازن زوج: بیتی است اضافی که جزئی از پیام بوده و سبب می‌شود که تعداد کل یک‌ها در پیام زوج باشد.

0001 → [Even Parity] → [10001] ارسال → [check]
- 10001 → ✓
- 11001 → ×
- 10111 → ×✓

سیستم قابلیت تشخیص مکان خطا را ندارد بنابراین با شمردن تعداد یک‌های کد ارسالی آن را تایید می‌کند.

0001 → [odd Parity] → [00001] ارسال → [check]
- 01001 ×
- 01101 ×✓
- 00001 ✓

Subject . Year. Month. Day

کد همینگ: یک کد تشخیص و تصحیح خطا می‌باشد. در این کد k بیت توازن به n بیت اطلاعات داده اضافه می‌شود. آن مکان‌هایی که توان‌هایی از دو باشند برای توازن رزرو شده‌اند

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P_1	P_2	_	P_4	_	_	_	P_8	_	_	_	_	_	_	_	P_{16}

$$\underbrace{11011001}_{\text{اطلاعات باینری}} \rightsquigarrow \underset{1}{P_1}\,\underset{2}{P_2}\,\underset{3}{1}\,\underset{4}{P_4}\,\underset{5}{1}\,\underset{6}{0}\,\underset{7}{1}\,\underset{8}{P_8}\,\underset{9}{1}\,\underset{10}{0}\,\underset{11}{0}\,\underset{12}{1} = 0$$

x	y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

$P_1 =$ XOR of bits $(3, 5, 7, 9, 11) =$ XOR$(1, 1, 1, 1, 0) = 0$

$P_2 =$ XOR of bits $(3, 6, 7, 10, 11) =$ XOR$(1, 0, 1, 0, 0) = 0$

$P_4 =$ XOR of bits $(5, 6, 7, 12) =$ XOR$(1, 0, 1, 1) = 1$

$P_8 =$ XOR of bits $(9, 10, 11, 12) =$ XOR$(1, 0, 0, 1) = 0$

$$\left.\begin{array}{l} C_1 = \text{XOR of bits}(1, 3, 5, 7, 9, 11) \\ C_2 = \text{XOR of bits}(2, 3, 6, 7, 10, 11) \\ C_4 = \text{XOR of bits}(4, 5, 6, 7, 12) \\ C_8 = \text{XOR of bits}(8, 9, 10, 11, 12) \end{array}\right\} C = C_8 C_4 C_2 C_1$$

C:
- Zero $\longrightarrow C = 0000 \longrightarrow$ یعنی خطا نداریم
- Non Zero $\longrightarrow$ معادل دهدهی عدد بدست آمده محل خطا را نشان میدهد.

کد گری (Gray code): خاصیت کد گری این است که در هنگام انتقال اطلاعات تقاطیب تغییرش دو عدد رخ میدهد.

روش تبدیل اعداد دهدهی به کد گری:

1 معادل باینری عدد را مینویسیم

2 صفر را به ابتدای عدد باینری (سمت چپ عدد) اضافه میکنیم.

3 بیت n ام کد گری نتیجه XOR کردن بیت n ام و n+1 ام از عدد باینری است.

$(8)_{10} = (\quad)_{Gray}$

هرکدام با بیت بعدی خودش XOR میشود.

$(8)_{10} = (1000)_2 \longrightarrow (01000) \longrightarrow (1100)_{Gray}$

	Binary	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
...	...	...
13	1101	1011
14	1110	1001
15	1111	1000

در این فصل ابتدا قوانین جبر بول به عنوان پایه اصلی طراحی مدارهای سوئیچی بررسی میگردد و سپس روش طراحی مدارهای دیجیتال با استفاده از قطعات ساده مورد بررسی قرار خواهد گرفت. جبر بول (جبر سوئیچی) مدارهای سوئیچی اجزای اصلی ساختمان دستگاه های دیجیتالی و کامپیوترها هستند.

سوئیچ ها دارای دو وضعیت مشخص و کاملا قابل تفکیک می باشند و جبر بول ریاضیات مربوط به این سیستم ها دو وضعیتی هستند.

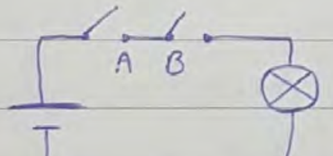
یک متغیر A در سیستم دو وضعیتی دارای یکی از دو حالت 0 (low) یا 1 (high) است.

جدول تغییرات و نمایش سوئیچی متغیر A به صورت زیر است.

A	
0	A=0 (low)
1	A=1 (high)

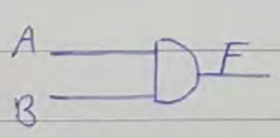
توابع مختلف را می توان با استفاده از متغیرهای A و B و C و ... ساخت که عملیات مشخصی را انجام می دهند. در اینجا ابتدا توابع مشخص و مدار معادل آنها معرفی می شود که با استفاده از آنها میتوان توابع پیچیده تر را ساخت.

تابع منطقی AND (گیت AND)



مدار معادل سوئیچی

A	B	F
۰	۰	۰
۰	۱	۰
۱	۰	۰
۱	۱	۱

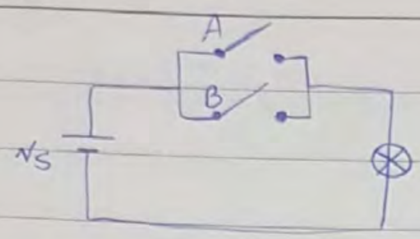


نمایش سمبولیک

$F = A \cdot B$

جدول درستی (Truth Table)

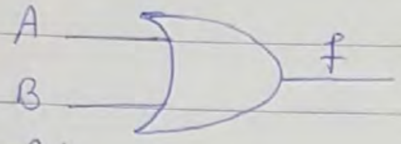
تابع منطقی OR (گیت OR)



A	B	f
0	0	0
0	1	1
1	0	1
1	1	1

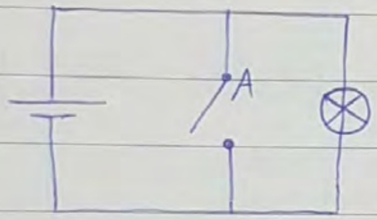
جدول درستی

$$f = A + B$$



نمایش سمبولیک

تابع منطقی Not (گیت Not)



A	f
0	1
1	0

جدول درستی

$$f = \bar{A} = \mathring{A}$$

نمایش سمبولیک

قوانین جبر بول اصول زیر را میتوان از مدارهای سوئیچی بالا استنباط کرد.

$$\left.\begin{array}{l} 0 \cdot 0 = 0 \\ 0 \cdot 1 = 0 \\ 1 \cdot 0 = 0 \\ 1 \cdot 1 = 1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} A \cdot 0 = 0 \\ A \cdot 1 = A \end{array}\right.$$

$$\left.\begin{array}{l} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 0 = 1 \\ 1 + 1 = 1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} A + 0 = A \\ A + 1 = 1 \end{array}\right.$$

$$\left\{\begin{array}{l} 0' = 1 \\ 1' = 0 \end{array}\right.$$

$$\left\{\begin{array}{l} \text{if } A \neq 1 \Rightarrow A = 0 \\ \text{if } A \neq 0 \Rightarrow A = 1 \end{array}\right.$$

قضایای جبر بول: قضایای زیر روابط بین متغیرها در جبر بول را نشان می‌دهند که درستی آنها را می‌توان با استفاده از مدار سوئیچی و اصول بالا تحقیق کرد.

1 قانون جابجایی:

$A.B = B.A \quad , \quad A+B = B+A$

2 قانون جمع‌پذیری:

$A.(B.C) = (A.B).C$

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

3 قانون توزیع‌پذیری:

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

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

4 قانون صفر و یک:

$A.1 = A \qquad A+1 = 1$

$A.0 = 0 \qquad A+0 = A$

5 قانون مکمل:

$A.\bar{A} = 0 \qquad A.A = A$

$A+\bar{A} = 1 \qquad A+A = A \qquad \bar{\bar{A}} = A$

6 قانون جذب:

$A+AB = A \longrightarrow A(\underbrace{1+B}_{1}) = A$

$A.(A+B) = A$

7 قانون دمورگان:

$\overline{A+B} = \bar{A}.\bar{B} \xrightarrow{\text{تعمیم}} \overline{A+B+C+\dots} = \bar{A}+\bar{B}+\bar{C}+\dots$

$\overline{A.B} = \bar{A}+\bar{B} \xrightarrow{\text{تعمیم}} \overline{A.B+C.\dots} = \bar{A}+\bar{B}+\bar{C}+\dots$

استفاده از جدول درستی برای اثبات قضایا و تساوی های جبر بول: با استفاده از جدول درستی می‌توان درستی قضایا و تساوی‌ها را تحقیق نمود.

A	B	$A\cdot B$	$A+A\cdot B$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1
=			=

$$A+A\cdot B=A$$

مثال) درستی تساوی های زیر را با استفاده از جدول درستی تحقیق کنید.

A	B	$A\cdot B$	$\overline{A\cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0
			=

A	B	$\bar{A}$	$\bar{B}$	$\bar{A}+\bar{B}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0
				=

$$\overline{A\cdot B}=\bar{A}+\bar{B}$$

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

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

$$A\bar{B} + AC + BC = A\bar{B} + BC$$

A	B	C	$\bar{B}$	$A\bar{B}$	AC	BC	$A\bar{B}+AC+BC$	$A\bar{B}+BC$
0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	0	0	0	1	1	1
1	0	0	1	1	0	0	1	1
1	0	1	1	1	1	0	1	1
1	1	0	0	0	0	0	0	0
1	1	1	0	0	1	1	1	1
							=	=

اصل دوگان: هر عبارت جبری نتیجه شده از جبر بول با تعویض عملگرها و عناصر خنثی باز هم معتبر می باشد. در جبر بول دوارزشی اگر دوگان یک عبارت جبری مورد نیاز باشد کافی است فقط عملگرهای OR و AND تعویض و یک ها به صفر و صفرها به یک تبدیل شوند مثلاً:

$$x + yZ = x$$

$$x \cdot (y+Z) = x$$

$$x + 1 = 1 \xrightarrow{\text{دوگان}} x \cdot 0 = 0$$

Subject . ______ Year. ______ Month. ______ Day. ______

ساده کردن توابع منطقی با استفاده از قضایای جبر بول و فاکتورگیری:

مثلاً:

$$F_1 = AC + ABC = AC\overbrace{(1+B)}^{1} = AC$$

$$F_2 = A + \bar{A}B = \underbrace{(A+\bar{A})}_{1} \cdot (A+B) = 1 \cdot (A+B) = A+B$$

$$A + BC = (A+B) \cdot (A+C)$$

$$F_3 = A + \bar{A}C + \bar{C}D = \underbrace{\underbrace{(A+\bar{A})}_{1} \cdot (A+C)}_{A+C} + \bar{C}D$$

$$= A + C + \bar{C}D = A + \underbrace{(C+\bar{C})}_{1} \cdot (C+D) = A + C + D$$

توابع منطقی زیر را با استفاده از قضایای جبر بول و فاکتورگیری ساده کنید.

$$F_1 = \overbrace{AB + \bar{A}B\bar{C}}^{B(A+\bar{A}\bar{C})} + \bar{A}BC = B(\overbrace{(A+\bar{A})}^{1}\cdot(A+\bar{C})) + \bar{A}BC = B(A+\bar{C}) + \bar{A}BC$$

$A + BC = (A+B)\cdot(A+C)$

$$= B(A + \bar{C} + \bar{A}C) = B(\bar{C} + \underbrace{\underbrace{(A+\bar{A})}_{1}\cdot(A+C)}_{A+C})$$

$x + yzw = (x+y)(x+z)(x+w)$

$$= B(\bar{C} + A + C) = AB$$

$$F_2 = (AB + \bar{A}\bar{B})\cdot(\bar{A}+B)\cdot A\bar{B} = \underbrace{(\underbrace{ABA\bar{B}}_{0} + \underbrace{\bar{A}\bar{B}A\bar{B}}_{0})}_{0}\cdot(\bar{A}+B) = 0$$

$$F_3 = A + B + \bar{A}\bar{B} + \bar{A}B(A+\bar{B}) = A + B + \bar{A}\bar{B} + (\overbrace{\bar{A}BA + \bar{A}B\bar{B}}^{0}) = A + B + \bar{A}\bar{B}$$

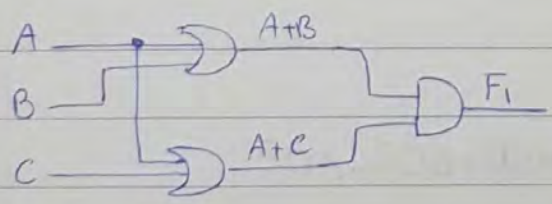
$$= (A+B) + (\overline{A+B}) = 1$$

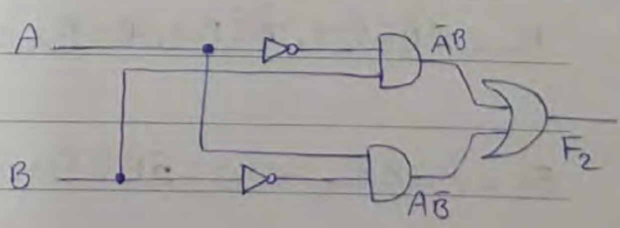
$\overline{xy} = \bar{x} + \bar{y}$ $\quad (\overline{x+y}) = \bar{x}\cdot\bar{y}$

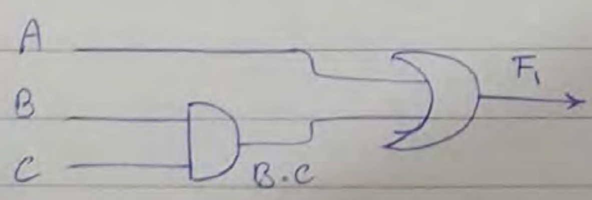
ساختن توابع منطقی با استفاده از گیت های and ، or ، not :

برای ساختن توابع منطقی با استفاده از گیت ها ابتدا باید تابع را تا حد امکان ساده نموده و سپس آن را ساخت.

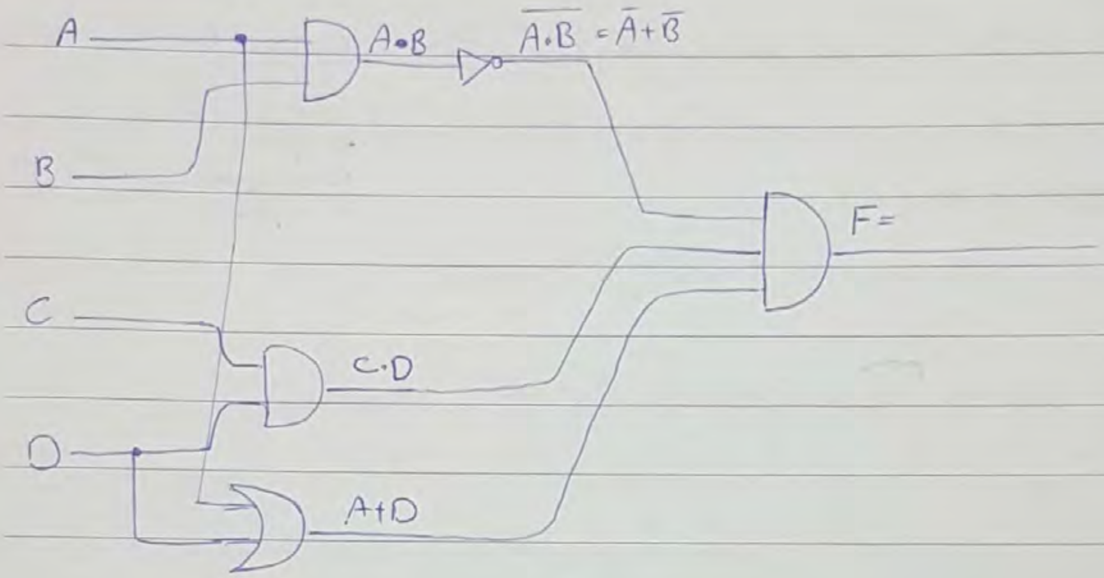
$F_1 = (A+B)(A+C) = A + B\cdot C$ $\qquad F_2 = \bar{A}B + A\bar{B}$







مثال) تابع F را با توجه به دیاگرام منطقی زیر بدست آورید.



$$F=(\bar{A}+\bar{B})\cdot(A+D)\cdot CD=\Big[\underbrace{A(\bar{A}+\bar{B})}_{\underbrace{A\bar{A}+A\bar{B}}_{A\bar{B}}}+D(\bar{A}+\bar{B})\Big]\cdot CD$$

$$=\Big[A\bar{B}+D(\bar{A}+\bar{B})\Big]\cdot CD=A\bar{B}CD+\overbrace{CD(\bar{A}+\bar{B})}^{\bar{A}CD+\bar{B}CD}=\bar{B}CD(A+1)+\bar{A}CD$$

$$=A\bar{B}CD+\bar{A}CD=CD(A\bar{B}+\bar{A})=CD(\underbrace{(A+\bar{A})}_{1}\cdot(\bar{A}+\bar{B}))=(\bar{A}+\bar{B})CD$$

توابع زیر را ساده نموده و با استفاده از گیت‌های منطقی پیاده‌سازی کنید.

$$F_1=\bar{A}\bar{B}\bar{C}\bar{D}+A\bar{B}\bar{C}\bar{D}+ABCD+\bar{A}BCD$$

$$F_2=\bar{A}B\bar{C}D+AB\bar{C}D+A\bar{B}\bar{C}D+ABCD=ABD+BD\bar{C}+AD\bar{C}$$

Subject . Year. Month. Day.

فرم کانونیک (فرم متعارف)

الف) نمایش توابع منطقی مینترم ها (minterm) و خواص آن.

دو متغیر A و B همراه با مکمل های آنها را میتوان در چهار حالت ($2^2=4$) به صورت دو به دو باهم and نمود که این چهار صورت عبارتند از:

$$A,B\begin{cases}\bar{A}\bar{B}\\ \bar{A}B\\ A\bar{B}\\ AB\end{cases}$$

هر یک از این حالات را یک مینترم (minterm) می نامند. تابعی که به صورت مجموع تعدادی از مینترم ها نوشته شده باشد می گویند تابع بر حسب مجموع مینترم ها یا مجموع حاصلضرب ها (SOP) (sum of Products) نوشته شده است. اگر هر بیت عدد باینری صفر باشد متغیر نظیر آن با پریم و در صورت یک بودن بدون پریم نمایش داده می شود.

هر مینترم نیز با سمبل m_j در جدول درستی نشان داده می شود که j معادل دهدهی حالت مربوطه است.

A	B	m
0	0	$m_0=\bar{A}\bar{B}$
0	1	$m_1=\bar{A}B$
1	0	$m_2=A\bar{B}$
1	1	$m_3=AB$

تابع بول را میتوان با استفاده از جدول درستی و با در نظر گرفتن مینترم هایی که تابع برای آنها یک است و اجرای عملگر OR روی آنها به صورت جبری بیان نمود.

	A	B	C	F_1	
m_0	0	0	0	0	
m_1	0	0	1	0	
m_2	0	1	0	0	
	0	1	1	1	$m_3=\bar{A}BC$
	1	0	0	1	$m_4=A\bar{B}\bar{C}$
	1	0	1	0	
	1	1	0	1	$m_6=AB\bar{C}$
m_7	1	1	1	0	

$$F_1(A,B,C)=\bar{A}BC+A\bar{B}\bar{C}+AB\bar{C}$$

$$F_1(A,B,C)=m_3+m_4+m_6$$

$$F_1(A,B,C)=\sum m(3,4,6)$$

Subject .

جدول درستی تابع زیر را رسم کنید

$$F_2 = \overset{4}{\underset{100}{A\bar{B}\bar{C}}} + \overset{6}{\underset{110}{AB\bar{C}}} + \overset{2}{\underset{010}{\bar{A}B\bar{C}}} + \overset{7}{\underset{111}{ABC}} + \overset{6}{\underset{110}{AB\bar{C}}}$$

A	B	C	F_2
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

اگر تابعی به صورت مینترم‌ها نباشد با استفاده از روش زیر می‌توان آن را به صورت مجموعه مینترم‌ها درآورد. ابتدا جمله‌ای که به صورت مجموع مینترم نیست تعیین کرده سپس آن را در جمله $(\bar{x}+x)$ ضرب می‌کنیم که در اینجا x متغیری است که در آن جمله وجود ندارد.

تابع زیر را به صورت مجموع مینترم‌ها درآورید.

$F_1 = A + \bar{B}C \qquad F_1(A,B,C)$

$A = A(B+\bar{B}) = AB + A\bar{B} \longrightarrow (AB+A\bar{B})(C+\bar{C}) = AB(C+\bar{C}) + A\bar{B}(C+\bar{C}) = ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C}$

$\bar{B}C = \bar{B}C(A+\bar{A}) = A\bar{B}C + \bar{A}\bar{B}C$

$\Rightarrow F_1 = \underset{\substack{m_7 \\ 111}}{ABC} + \underset{\substack{m_6 \\ 110}}{AB\bar{C}} + \cancel{A\bar{B}C} + \underset{\substack{m_1 \\ 001}}{\bar{A}\bar{B}C} + \underset{\substack{m_5 \\ 101}}{A\bar{B}C} + \underset{\substack{m_4 \\ 100}}{A\bar{B}\bar{C}} = m_1 + m_4 + m_5 + m_6 + m_7$

$= \sum(1,4,5,6,7)$

راه دیگر برای بدست آوردن مینترم‌های یک تابع استفاده از جدول درستی است.

مثال) تابع F_1 در مثال قبل را با استفاده از جدول درستی به صورت مجموع مینترم‌ها بنویسید.

A	B	C	$\bar{B}$	$\bar{B}C$	F_1: $\bar{B}C+A$	
0	0	0	1	0	0	
0	0	1	1	1	1	$\rightarrow m_1$
0	1	0	0	0	0	
0	1	1	0	0	0	
1	0	0	1	0	1	$\rightarrow m_4$
1	0	1	1	0	1	$\rightarrow m_5$
1	1	0	0	0	1	$\rightarrow m_6$
1	1	1	0	0	1	$\rightarrow m_7$

$F_1 = m_1 + m_4 + m_5 + m_6 + m_7$
$= \sum(1,4,5,6,7)$

Subject . Year. Month. Day.

ب) نمایش توابع منطقی برحسب ماکسترم ها (maxterm) :

دو متغیر A و B همراه با مکمل های آنها را میتوان به چهار صورت ($4 = 2^2$) دو به دو با هم OR نمود. این چهار صورت عبارتند از:

$A+B \rightarrow M_0$

$\bar{A}+B \rightarrow M_1$

$A+\bar{B} \rightarrow M_2$

$\bar{A}+\bar{B} \rightarrow M_3$

هریک از این جملات را یک ماکسترم مینامند
تابعی را که به صورت حاصلضرب تعدادی ماکسترم نوشته شده باشد میگویند که تابع برحسب حاصلضرب ماکسترم نوشته شده است.

POS = Product of sum

در مورد ماکسترم ها نیز هر جمله به یکی از حالات دو متغیر A و B نسبت داده میشود و جدول درستی تابعی که برحسب جملات ماکسترم نوشته شده باشد را میتوان مستقیما رسم نمود و یا از روی جدول درستی فرمول تابع را برحسب حاصل ضرب ماکسترم ها نوشت

روش مستقیم حاصلضرب ماکسترم ها از جدول درستی بدین صورت است که برای هر ترکیبی از متغیر ها که تابع به ازای آنها در جدول درستی صفر است یک ماکسترم از ترکیب متغیرها تشکیل و سپس تمام آنها را And می نماییم.

قابل ذکر است در هنگام نوشتن ماکسترم اگر هر بیت عدد باینری صفر باشد متغیر نظیر آن بدون پریم و در صورت یک بودن با پریم نمایش داده میشود.

مثال) جدول درستی تابع F را رسم کنید.

$F(ABC) = \prod(1,3,5) = M_1 M_3 M_5$

$M_1 \rightarrow 001 \rightarrow A+B+\bar{C}$

$M_3 \rightarrow 011 \rightarrow A+\bar{B}+\bar{C}$

$M_5 \rightarrow 101 \rightarrow \bar{A}+B+\bar{C}$

A	B	C	F_1
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

حاصل ضرب ماکسترم ها:

تابع بول n متغیری را میتوان به صورت حاصل ضرب ماکسترم ها نیز نمایش داد. برای این کار ابتدا باید تابع را به صورت جمله های OR شده درآوریم لذا از اصل توزیع پذیری: $x+yz=(x+y)\cdot(x+z)$ استفاده میکنیم. سپس اگر متغیری مانند x در یک جمله OR وجود نداشت آن جمله را با عبارت $x\bar{x}$ OR مینمائیم.

مثال) تابع F را به صورت حاصلضرب ماکسترم ها درآورید.

$$F = xy+\bar{x}z = (xy+\bar{x})\cdot(xy+z) = \underbrace{(x+\bar{x})}_{1}\cdot(\bar{x}+y)\cdot(x+z)\cdot(y+z)$$

حذف ← 1

حواست به ترتیب متغیرها باشه

$A\cdot A = A$ *

$$\Rightarrow (\overbrace{\bar{x}}^{A}+\overset{B}{y}+\overset{C}{z\bar{z}})\cdot(x+z+y\bar{y})\cdot(y+z+x\bar{x})$$

$$=\cancel{(\bar{x}+y+z)}(\bar{x}+y+\bar{z})\cdot(x+z+y)(x+z+\bar{y})\cdot\cancel{(y+z+x)}\cdot(y+z+\bar{x})$$

$$F=(\bar{x}+y+z)\cdot(\bar{x}+y+\bar{z})\cdot(x+y+z)\cdot(x+\bar{y}+\bar{z}) = \Pi(0,2,4,5)=M_0M_2M_4M_5$$

$(\bar{x}+y+z) \rightarrow 100 \rightarrow M_4$

$(\bar{x}+y+\bar{z}) \rightarrow 101 \rightarrow M_5$

$(x+y+z) \rightarrow 000 \rightarrow M_0$

$(x+\bar{y}+\bar{z}) \rightarrow 010 \rightarrow M_2$

تبدیل فرم های کانونیک به یکدیگر

روش کلی برای تبدیل یک فرم کانونیک به فرم دیگر تعویض علامت های Σ و Π و نوشتن آن شماره های که در فرم اصلی تابع وجود ندارد میباشد. چون تابع اصلی برابر مجموع مینترم هایی است که تابع به ازای آنها برابر یک میباشد پس مکمل آن برابر مجموع مینترم هایی است که تابع به ازای آنها مساوی صفر است.

$$F = F(x,y) = \overset{\Pi}{\cancel{\Sigma}}(0,3) = \overset{\bar{x}\bar{y}}{m_0} + \overset{xy}{m_3}$$

$$\bar{F} = \Sigma(1,2) = m_1 + m_2 = x\bar{y} + \bar{x}y$$

$$\bar{\bar{F}} = \overline{(x\bar{y} + \bar{x}y)} = (\bar{x}+y)\cdot(x+\bar{y}) = M_1 \cdot M_2 = \Pi(1,2)$$

$$(\bar{x}+y) \rightarrow 10 \rightarrow M_2 \qquad (x+\bar{y}) \rightarrow 01 \rightarrow M_1$$

$$F(x,y,z) = \Sigma(1,3,6,7) \longrightarrow F(x,y,z) = \Pi(0,2,4,5)$$

در حالت کلی میتوان گفت:

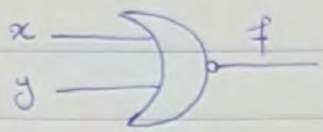
$$\boxed{\bar{m}_j = M_j}$$

قانون کلی: مکمل مینترم‌های ژ برابر ماکسترم‌های ژ است. ($M_j = \bar{m}_j$)

گیت‌های منطقی:

۱) گیت NOR

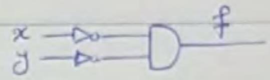
x	y	f
0	0	1
0	1	0
1	0	0
1	1	0



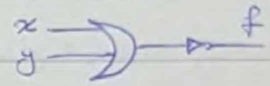
$f = (\overline{x+y}) = \bar{x} \cdot \bar{y}$

یا

OR - Not

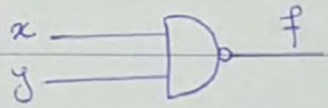


Not - And



۲) گیت NAND:

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

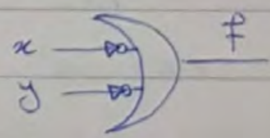


$f(\overline{x \cdot y}) = \bar{x} + \bar{y}$

یا

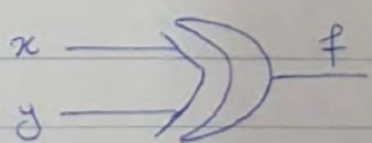
And - Not

Not - OR



۳) گیت XOR

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

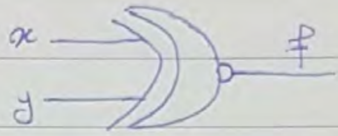


$f = x\bar{y} + \bar{x}y$

$f = x + y$

XNOR (4

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



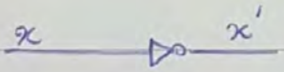
$$f = x \odot y = xy + x'y'$$

نکته: $[x \oplus y]' = x \odot y$

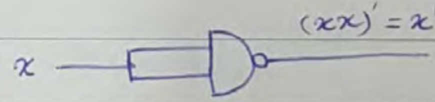
$$x \oplus y = xy' + x'y \longrightarrow (xy' + x'y)' = (x'+y).(x+y')$$

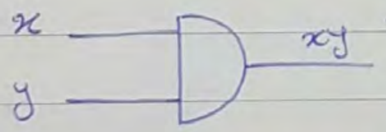
$$= (x'+y)x + (x'+y)y' = \underbrace{xx'}_{0} + xy + x'y' + \underbrace{yy'}_{0} = xy + x'y' = x \odot y$$

تبدیل گیت ها به هم :

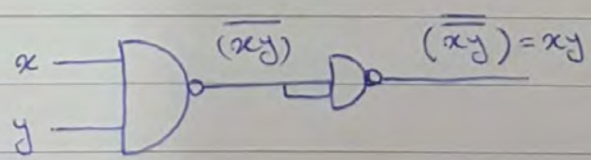


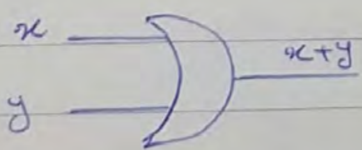
$(xx)' = x'$

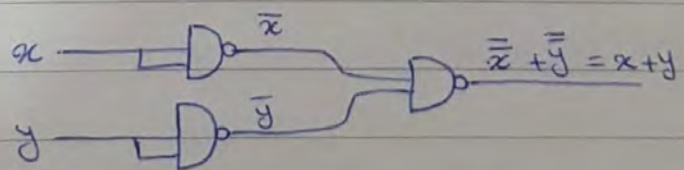
$\longrightarrow$ 



$\overline{\overline{(xy)}} = xy$







فصل سوم:

نقشه کارنو و ساده کردن توابع با استفاده از آن:

یکی از روش های ساده کردن توابع منطقی استفاده از نقشه هاست که یکی از معمول ترین آنها نقشه کارنو میباشد. با استفاده از این روش توابع منطقی تا حد امکان ساده می شوند.

نقشه کارنو یک تابع n متغیری دارای 2^n خانه می باشد که هر خانه جایگاه یک مینترم است.

نحوه قرار گرفتن یک مینترم به صورتی است که هر خانه نسبت به خانه های اطراف تنها در یکی از متغیرها تغییر وضعیت داده است. در واقع نقشه دیاگرام تصویری از تمام حالاتی است که یک تابع بول به صورت استاندارد میتواند داشته باشد.

نقشه کارنو دو متغیره:

$n=2 \longrightarrow 2^2 = 4$ خانه یا مینترم

$x \backslash y$	0	1
0	$x'y'$ m_0	$x'y$ m_1
1	xy' m_2	xy m_3

در هنگام کار با نقشه کارنو مینترم هایی که تابع مورد نظر شامل آنها می باشد را در نقشه با عدد 1 مشخص می کنیم.

مثال) نقشه کارنو عبارت زیر را نمایش دهید.

$F_1 = xy$

$x \backslash y$	0	1
0		
1		1

$F_2 = x+y = x(y+y') + y(x+x') = xy + xy' + xy + x'y$

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

$x \backslash y$	0	1
0		1
1	1	1

~~مثال) نقشه کارنو عبارت زیر را بکشید.~~

نقشه کارنو سه متغیره: شامل 8 مینترم میباشد ($n=3 \longrightarrow 2^3 = 8$) که مینترم ها باید به ترتیب کد گری باشند. که در این صورت از هر مربع به مربع دیگر فقط یک بیت از صفر به یک یا از یک به صفر تغییر می کند. برای رسم نقشه کارنوی یک تابع کافیست مینترم هایی را که تابع شامل آنها می باشد با عدد 1 در نقشه مشخص کنیم.

$x \backslash yz$	00	01	11	10
0	$x'y'z'$ m_0	$x'y'z$ m_1	$x'yz$ m_3	$x'yz'$ m_2
1	$xy'z'$ m_4	$xy'z$ m_5	xyz m_7	xyz' m_6

$F(x,y,z)$

مثال) نقشه کارنو عبارت زیر را بکشید.

$$F(x,y,z) = m_3 + m_5 = \underset{011}{x'yz} + \underset{101}{xy'z}$$

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

$$F_2 = m_5 + m_7 = xy'z + xyz$$

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

مثال) تابع بول زیر را با استفاده از نقشه کارنو رسم و ساده نمایید.

$$F(x,y,z) = \sum(0,1,6,7) = x'y'z' + x'y'z + xyz' + xyz = \underset{000}{m_0} + \underset{001}{m_1} + \underset{110}{m_6} + \underset{111}{m_7}$$

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

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

نکته: گاهی اوقات دو مربع هم‌جوار هستند اما به هم چسبیده نیستند.

$F(x,y,z) = \underset{000}{m_0} + \underset{010}{m_2} = x'y'z' + x'yz' = x'z'$

x \ yz	00	01	11	10
0	1			1
1				

$F(x,y,z) = \sum(3,4,6,7) = x'yz + xy'z' + xyz' + xyz = yz + xz'$

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

xy *

در نقشه کارنو ساده کردن را تا جایی ادامه میدهیم که یکی باقی مانده باشد که هم‌جوار داشته باشد.

یعنی در مثال بالا نیازی نیست هم‌جواری * را تشکیل دهیم

ولی در کل اگر یک 1 باقی بماند که قابل ساده شدن نباشد عبارت را عینا می‌نویسیم.

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

$x'y'z$

$xz' + xy + x'y'z$

ترکیب چهار مربع هم‌جوار:

$F(x,y,z) = \sum(0,2,4,6) = x'y'z' + x'yz' + xy'z' + xyz'$

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

$\Rightarrow$

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

$\rightarrow F = z'$

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

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

$\rightarrow F = x'$

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

پس در حالت سه متغیره می‌توان گفت:

الف) یک مربع نظیر یک مینترم است که با یک جمله سه متغیری نمایش داده می‌شود.

ب) دو مربع همجوار با یک جمله دو متغیری نمایش داده می‌شود

پ) چهار مربع همجوار با یک جمله تک متغیری نمایش داده می‌شود

ت) تمام مربع های یک نقشه (در این حالت هشت مربع) تولید تابعی میکنند که همواره یک است.

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

مثال)

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

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

$\rightarrow F = xy' + z'$

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

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

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

$BC(A+A') = ABC + A'BC$

$A \backslash BC$	00	01	11	10
0		1	1	1
1		1	1	

$\rightarrow F = A'B + C$

نکات مهم در انتخاب مربع های همجوار و ساده سازی آنها:

۱. مربع هایی که در یک سطر یا یک ستون هستند هم‌جوارند

۲. مربع هایی که در ابتدا و انتهای یک سطر یا ستون قرار دارند هم‌جوارند.

۳. ترتیب خانه ها باید به گونه‌ای باشد که مربع های هم‌جوار انتخاب شده دارای تعداد 2^n باشند.

به طور کلی در گروه های 2^n خانه‌ای از نقشه کارنو n متغیر باید حذف شود.

ع) گروه ها (همسایگی ها) را تا حد امکان بزرگ میگیریم تا به عبارت ساده تری برسیم و گروه بندی را تا جایی ادامه میدهیم که هر مینترم حداقل یک بار مورد استفاده قرار گیرد.

نقشه کارنو چهار متغیره:

$n=4 \rightsquigarrow 2^n = 2^4 = 16$ خانه

$F(x,y,z,w)$

$xy \backslash zw$	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	$x'yz'w$ m_5	m_7	m_6
11	m_{12}	m_{13}	$xyzw$ m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

در حالت چهار متغیره:

الف) یک مربع یک مینترم را نشان میدهد که یک جمله چهار متغیره است.

ب) دو مربع همجوار یک جمله سه متغیره را نشان میدهد.

ج) چهار مربع همجوار یک جمله دو متغیره را نشان میدهد.

د) هشت مربع همجوار یک جمله یک متغیره را نشان میدهد.

هـ) شانزده مربع همجوار نمایش این است که تابع همواره یک است.

مدار ۹۸/۸/۲۹ چهارشنبه ۸ - ۹:۴۵

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

$wx \backslash yz$	۰۰	۰۱	۱۱	۱۰
۰۰	۱	۱		۱
۰۱	۱	۱		۱
۱۱	۱	۱		۱
۱۰	۱	۱		

$F(w,x,y,z) = y' + w'z' + xz'$

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

AB \ CD	00	01	11	10
00	1	1		1
01				1
11				
10	1	1		1

$A'B'C' : A'B'C'(D+D') = \overset{0001}{A'B'C'D} + \overset{0000}{A'B'C'D'}$

$B'CD' : (A+A')B'CD' = \overset{1010}{AB'CD'} + \overset{0010}{A'B'CD'}$

$AB'C' : AB'C'(D+D') = \underset{1001}{AB'C'D} + \underset{1000}{AB'C'D'}$

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

$F(A,B,C,D) = \Sigma(1,4,5,6,12,14,15)$

1 : 0001
4 : 0100
5 : 0101
6 : 0110
12 : 1100
14 : 1110
15 : 1111

AB \ CD	00	01	11	10
00		1		
01	1	1		1
11	1		1	1
10				

$F = BD' + ABC + A'C'D$

نقشه کارنو پنج متغیره:

$n=5 \longrightarrow$ تعداد خانه ها $= 2^5 = 32$

نقشه کارنو پنج متغیره شماره ۱:

$F(A,B,C,D,E)$

11001 → ?
ABCDE

A = 0

BC \ DE				
	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 = 1

BC \ DE				
	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}

Subject . Year. Month. Day.

نقشه کارنو پنج متغیره شماره دو :

AB \ CDE	000	001	011	010	110	111	101	100
00	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
01	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
11	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
10	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}

* در نقشه کارنو پنج متغیره شماره یک مربع هایی که در مکان های یکسان قرار دارند هم جوارند مثلاً m_4 و m_{20} هم جوارند یا m_{15} و m_{31} هم جوارند.

* در نقشه کارنو پنج متغیره شماره دو اگر دو بخش نقشه را قرینه یکدیگر فرض کنیم در این صورت هر مربع با تصویر خود هم جوار است مانند m_9 و m_{13} یا m_{11} و m_{15}.

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

$A=0$

BC \ DE	00	01	11	10
00	1			1
01	1			1
11		1		
10		1		

$A=1$

BC \ DE	00	01	11	10
00				
01		1	1	
11		1	1	
10		1		

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

Ganjineh

$F(A,B,C,D,E) = \Sigma(0,2,3,4,5,6,7,11,15,16,18,19,23,27,31)$

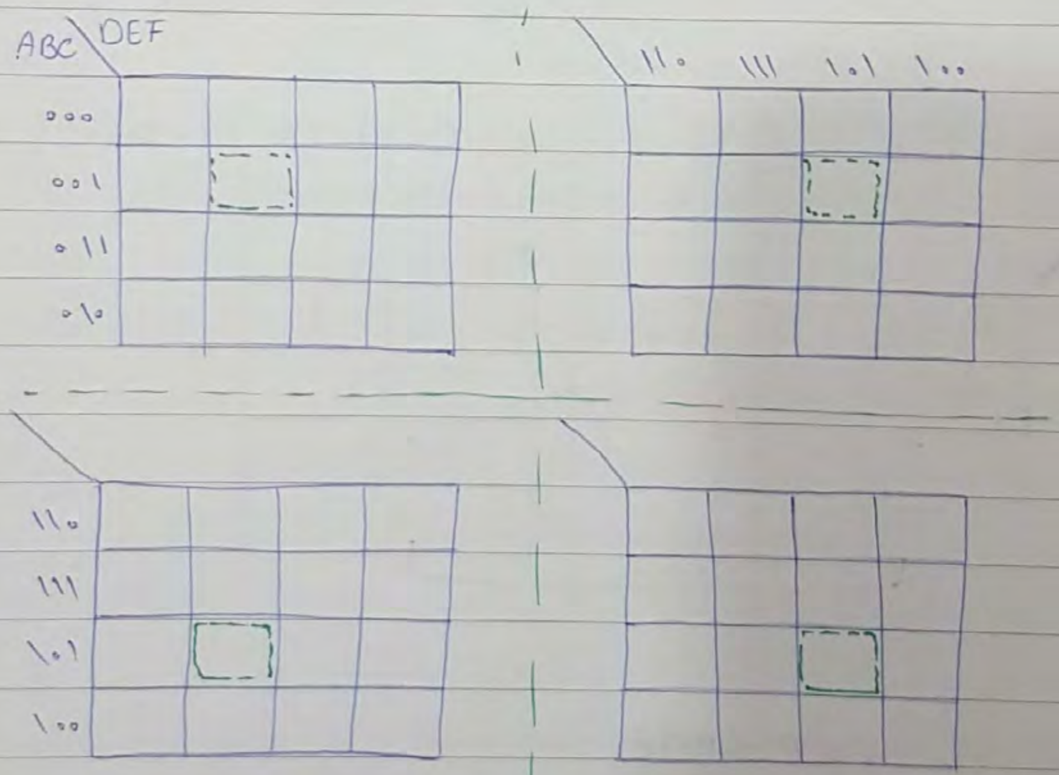
AB \ CDE	000	001	011	010	110	111	101	100
00	1		1	1	1	1	1	1
01			1			1		
11			1			1		
10	1		1	1		1		

$F = A'B'C + B'C'E' + DE$

نقشه شش متغیره :

$n=6 \longrightarrow 2^6 = 64$ خانه

$F(A,B,C,D,E,F)$



* در نقشه کارنوی شش متغیره میتوان چهار بخش مختلف نقشه را به صورت قرینه های افقی و عمودی فرض کرد
به عنوان مثال مربع های m_9، m_{13}، m_{45} و m_{41} هم جوارند.

ساده نمودن تابع برحسب حاصلضرب مجموع ها:

در مثال های قبل توابع بول ساده شده از نقشه کارنو به صورت مجموع حاصلضرب ها بودند ولی با کمی تغییر می توان تابع را به صورت حاصلضرب مجموع ها درآورد.

نمایش تابع F برحسب مینترم ها ⟵ ۱ های داخل نقشه

نمایش مکمل تابع F (F') ⟵ مربع های فاقد ۱

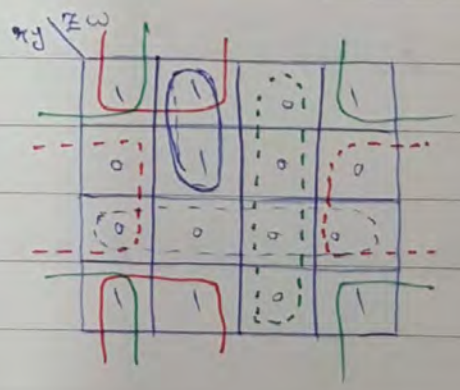
بنابراین چنانچه درون مربع های خالی صفر بگذاریم و مربع های هم جوار آنها را ترکیب کنیم فرم ساده شده F' بدست می آید.

از آنجا که مکمل مکمل تابع، خود تابع میباشد ($(F')' = F$) با استفاده از تئوری دمورگان تابع برحسب حاصلضرب مجموع ها ساده می شود.

چهار ۹۸،۹،۵ شنبه ۸:۰۰ - ۹:۴۵

مثال: تابع زیر را برحسب مجموع حاصلضرب ها و حاصلضرب مجموع ها ساده کنید.

$F(x,y,z,w) = \Sigma(0,1,2,5,8,9,10)$



$F(x,y,z,w) = y'w' + yz' + x'z'w$

$F'(x,y,z,w) = xy + zw + yw'$

$(F')' = F$

$(F')' = (xy + zw + yw')' = (x'+y') \cdot (z'+w') \cdot (y'+w)$

نتیجه: چنانچه نقشه کارنو یک تابع داده شده باشد با ترکیب یک ها تابع برحسب مجموع حاصلضرب ها بدست می آید
و با ترکیب صفرها تابع برحسب حاصلضرب مجموع ها حاصل میشود.

در نقشه کارنو مثال قبل صفرهای همسایه ها را پیدا میکنیم $\underrightarrow{POS} \Rightarrow F = (x'+y').(z'+w').(y'+w)$
به ازای ۱ ها (پریم) و به ازای صفرها (خود) متغیر را مینویسیم و OR میکنیم و مجموعاً AND میکنیم.

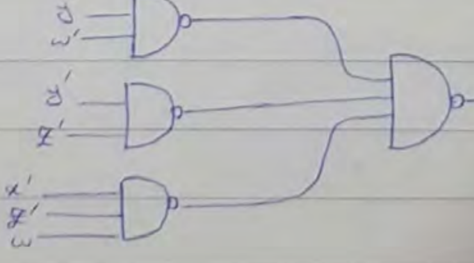
پیاده سازی تابع با گیت های NAND و NOR :

به علت سادگی ساخت گیت های NAND و NOR نسبت به بقیه گیت ها در مدارات دیجیتال از این دو گیت بیشتر استفاده می شود.

قاعده پیاده سازی تابع با گیت NAND (پیاده سازی دو طبقه):

۱. تابع باید ساده شود و به صورت مجموع حاصلضرب ها تبدیل گردد.
۲. برای هر جمله حاصلضرب که حداقل شامل دو متغیر باشد یک گیت NAND کشیده می شود و به ورودی های گیت NAND، متغیرهای جمله وارد میشوند. این گروه گیت های طبقه اول را تشکیل می دهند.
۳. در طبقه دوم یک گیت NAND کشیده می شود که ورودی های آن از خروجی گیتهای طبقه اول می آید.
٤. یک جمله با یک متغیر در طبقه اول نیاز به یک معکوس کننده دارد.

برای مثال صفحه قبل داریم: $F(x,y,z,w) = y'w' + y'z' + x'z'w$

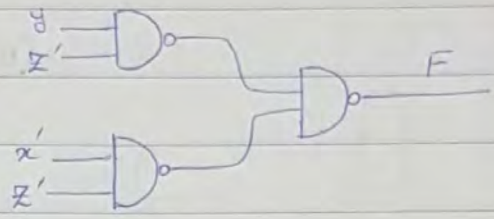


$$F = ((y'w')' . (y'z')' . (x'z'w)')'$$
$$= y'w' + y'z' + x'z'w$$

مثال) تابع زیر را با گیت NAND پیاده سازی کنید.

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

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



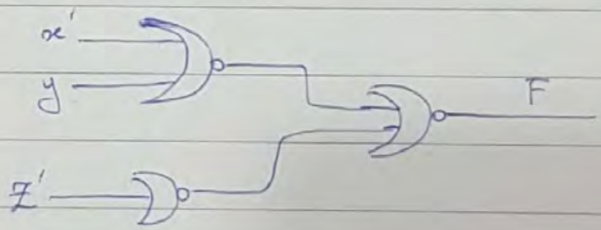
$F = yz' + x'z'$

پیاده سازی تابع با گیت NOR :

قاعده پیاده سازی تابع با گیت NOR نظیر گیت NAND میباشد با این تفاوت که تابع باید به صورت حاصلضرب مجموع ها ساده گردد و به جای گیت های NAND از گیت های NOR استفاده شود.

به عنوان مثال اگر بخواهیم مثال قبل را با گیت NOR پیاده سازی کنیم داریم:

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

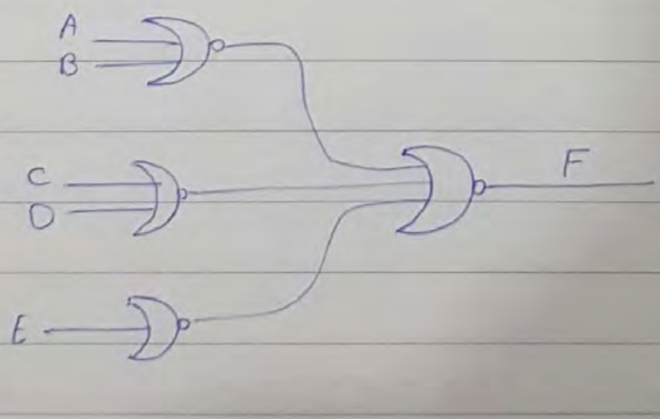


$F' = Z + xy'$

$(F')' = Z' \cdot (x' + y)$

$F = Z' \cdot (x' + y)$

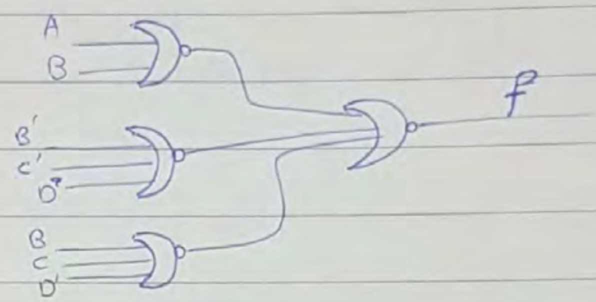
$F = (A+B) \cdot (C+D) \cdot E$



مثال) تابع زیر را به روش POS ساده کنید. $f(A,B,C,D) = \Pi(0,1,2,3,6,9,14)$: POS

AB \ CD	00	01	11	10
00	0	0	0	0
01				0
11				0
10		0		

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



$= \Sigma(4,5,7,8,10,11,12,13,15)$: SOP

سوال) به روش POS ساده کنید.

$$F(A,B,C,D,E) = \Sigma(1,3,5,7,8,9,10,12,13,17,18,19,21,22,23,25,26,27,28,29)$$

پاسخ: $f(A,B,C,D,E) = (B'+C'+D')(A+B'+D'+E')(B+D+E)(A+B+D'+E)(A'+C+D+E)$

Don't care:

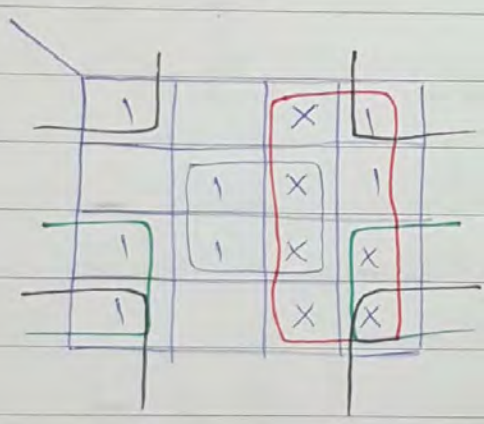
شرایط بی اهمیت :

چنانچه مقدار تابع به ازای بعضی از خانه های نقشه کارنو مشخص نباشند آن خانه ها را برحسب نیاز صفر یا یک در نظر گرفته و تابع را ساده می نماییم.

برای مشخص کردن این مینترم های بی اهمیت به جای قرار دادن صفر یا یک مقدار (x) را در نقشه کارنو قرار میدهیم لذا قرار دادن علامت ضربدر داخل مربع های نقشه نشان دهنده مقدار بی اهمیت مینترم برای تابع است.

مثال) تابع زیر را ساده کنید درصورتی که تابع به ازای $F(w,x,y,z)=\Sigma(0,2,5,6,8,12,13)$ برابر یک باشد و به ازای مینترم های d بی اهمیت باشد که:

$d(w,x,y,z)=\Sigma(3,7,10,11,14,15)$



$F(w,x,y,z)=y+xz+wz'+x'z'$

Queen mc clavski (QM)

روش جدول بندی: کوین مک کلاسکی

این روش برای زمانی که تعداد متغیر ها بیش از 5 یا 6 می باشد استفاده می شود. روش QM یک روش الگوریتمی است و به راحتی قابل پیاده سازی توسط ماشین است.

این روش برای هر تعداد متغیر جواب میدهد اما نقشه کارنو حداکثر 6 متغیر را ساده می کند.

همچنین می توان همزمان چندین تابع را با روش QM ساده کرد. البته روش QM بسیار زمان بر است.

مراحل این روش:

1. دسته بندی مینترم ها برحسب تعداد یک ها (ستون الف)

2. هر دو مینترمی که فقط در یک متغیر با هم اختلاف دارند می توانند با هم ترکیب شوند و در این ترکیب متغیری که اختلاف مینترمی

آنها ست حذف می‌شود. (ستون ب)

* کنار هر مینترم ترکیب شده (✓) گذاشته میشود و به جای هر متغیر حذف شده یک (—) گذاشته می‌شود.

۳- عملیات مقایسه نظیر حالات قبل برای ستون (ب) هم انجام میشود. و در این صورت مقایسه تنها برای مینترم های امکان دارد که خط تیره های در مکان یکسان دارند.

مدار ۹۸/۹/۴ ۸:۰۰..۹:۴۵ چهارشنبه

تابع بول زیر را با بکار بردن روش جدول بندی ساده نمایید.

$$f = \Sigma(0, 1, 2, 8, 10, 11, 14, 15)$$

0000, 0001, 0010, 1000, 1010, 1011, 1110, 1111

ستون (الف)

معادل دهدهی	w	x	y	z	
0 →	0	0	0	0	✓
1 →	0	0	0	1	✓
2 →	0	0	1	0	✓
8 →	1	0	0	0	✓
10 →	1	0	1	0	✓
11 →	1	0	1	1	✓
14 →	1	1	1	0	✓
15 →	1	1	1	1	✓

ستون (ب)

	w	x	y	z	
(0,1) →	0	0	0	—	✗ → PI_3
(0,2) →	0	0	—	0	✓
(0,8) →	—	0	0	0	✓
(2,10) →	—	0	1	0	✓
(8,10) →	1	0	—	0	✓
(10,11) →	1	0	1	—	✓
(10,14) →	1	—	1	0	✓
(11,15) →	1	—	1	1	✓
(14,15) →	1	1	1	—	✓

ستون (پ)

	w	x	y	z	
(0,2,8,10) →	—	0	—	0	✗ → PI_1
(0,8,2,10) →	—	0	—	0	✗
(10,11,14,15) →	1	—	1	—	✗ → PI_2
(10,14,11,15) →	1	—	1	—	✗

✗ . تکراری نمی‌خواهیم پس این دوتا رو در نظر نمی‌گیریم

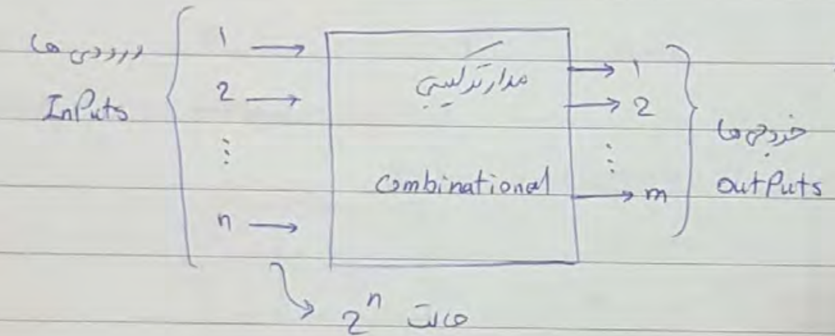
$$f = w'x'y' + x'z' + wy$$

انتهای مبحث میانترم

فصل چهارم :

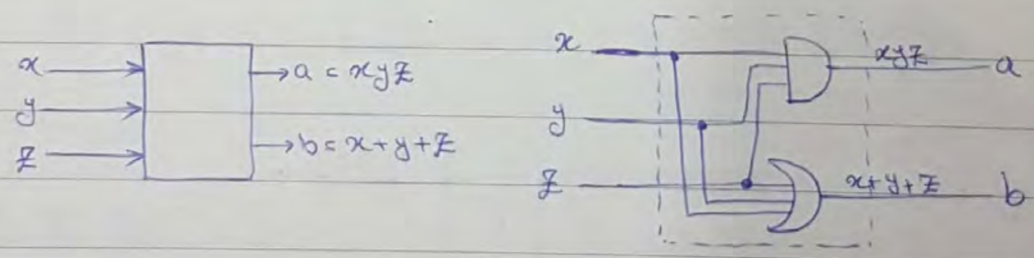
مدارهای منطقی ترکیبی : مدار منطقی ترکیبی مداری است که دارای تعدادی ورودی و خروجی میباشد به طوری که خروجی ها فقط به ورودی های فعلی وابسته هستند یا به عبارت دیگر خروجی آنها در هر لحظه به وسیله ترکیب حاضر متغیرهای ورودی در همان لحظه مشخص میشود. این مدارها دارای حافظه نیستند یعنی خروجی به ورودی های گذشته وابسته نیست .

هدف از این فصل تحلیل و طراحی مدار های ترکیبی است .



~~روش طراحی مدارهای ترکیبی :~~

مثال)



روش طراحی مدارهای ترکیبی :

۱ مشخص کردن صورت مسئله

۲ تعیین تعداد متغیرهای ورودی و توابع خروجی

۳ تخصیص حروف به ورودی ها و خروجی ها

۴ تشکیل جدول درستی که رابطه بین ورودی ها و خروجی ها را نشان میدهد (با ترکیب n متغیر ورودی 2^n ترکیب (صفر) وجود دارد.

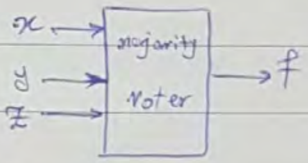
۵ بدست آوردن تابع بول ساده شده برای هر خروجی توسط نگاشت کارنو، جبر بول، نقشه ها، یا روش جدول بندی .

۴ رسم مدار منطقی با توجه به تابع ساده شده .

نکته : در حالاتی که بعضی ترکیب های ورودی هرگز بوجود نمی آیند این ترکیب ها شرایط بی اهمیت نامیده می شوند .

مثال) مداری طراحی کنید که براساس اکثریت آراء سه نفر رای را صادر کند؟

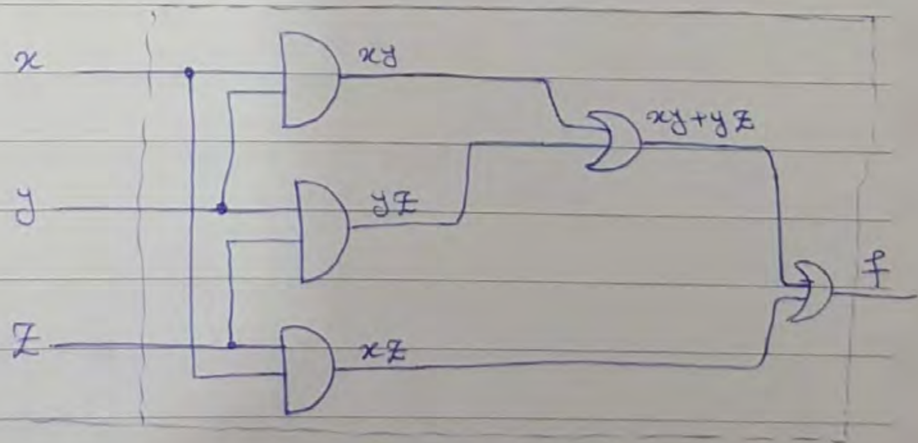
x	y	z	f	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\leftarrow m_3$
1	0	0	0	
1	0	1	1	$\leftarrow m_5$
1	1	0	1	$\leftarrow m_6$
1	1	1	1	$\leftarrow m_7$



$$f = \Sigma(3,5,6,7)$$
$$= x'yz + xy'z + xyz' + xyz$$

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

$$\rightarrow f = yz + xy + xz$$



majority voter

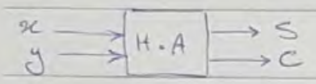
مدار ۱۳، ۹، ۹۸   ۸:۰۰-۹:۴۵ چهارشنبه

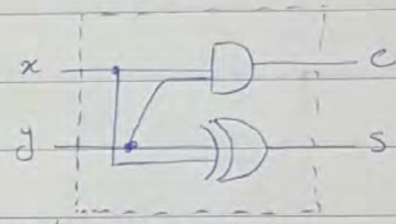
(۱) جمع کننده = Adder

→ نیم جمع کننده = half Adder = h.A → جمع دو رقم باینری را انجام میدهد

→ تمام جمع کننده = Full Adder = F.A → جمع سه رقم باینری را انجام میدهد

مثال) ورودی‌ها = x, y ، خروجی‌ها = S, C



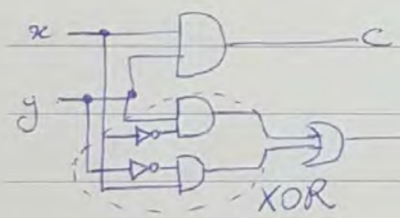


x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

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

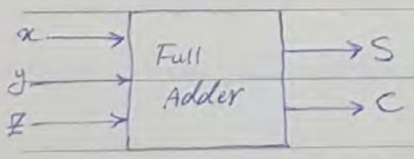
$\rightarrow C = xy$

C:



اگر متوجه نمیشدیم S برابر با $x \oplus y$ هست باید به این شکل ساده سازی میکردیم.

مثال) ورودی‌ها = x, y, Z ، خروجی‌ها = C, S



C: *

$x \backslash yZ$	00	01	11	10
0			1	
1		1	1	1

S:

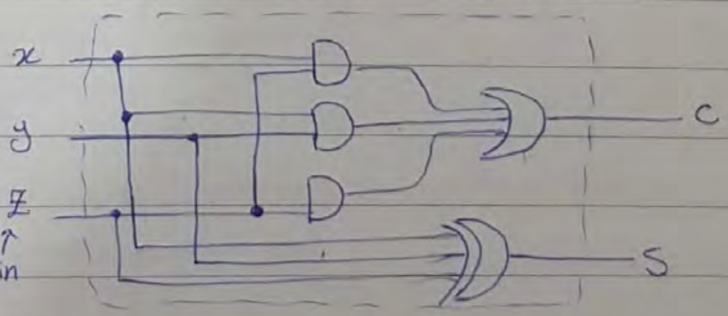
$x \backslash yZ$	00	01	11	10
0		1		1
1	1		1	

ساده نمیشود!

$$= \underbrace{Z}_{A}\underbrace{(x\oplus y)'}_{B'} + \underbrace{Z'}_{A'}\underbrace{(x\oplus y)}_{B} = x \oplus y \oplus Z$$

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

Z ↑ cin



$\rightarrow C = x'yZ + xy'Z + xyZ' + xyZ \xrightarrow{*} = xZ + xy + yZ$

$\rightarrow S = x'y'Z + x'yZ' + xy'Z' + xyZ = Z(\underbrace{xy + x'y'}_{XNOR}) + Z'(\underbrace{x'y + xy'}_{XOR}) = Z(\underbrace{x \odot y}_{(x\oplus y)'}) + Z'(x \oplus y)$

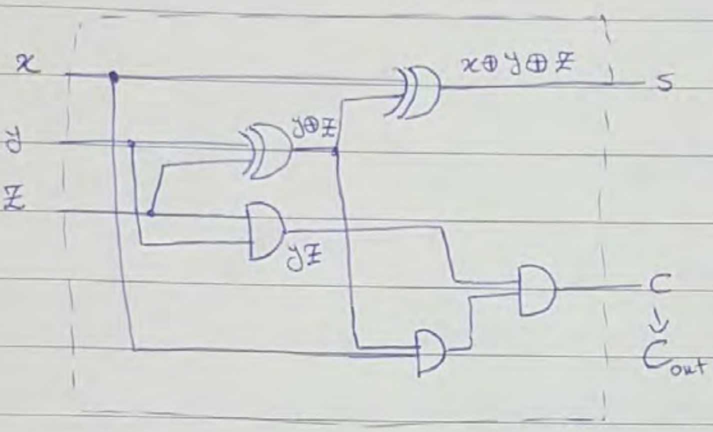
Subject . Year. Month. Day.

اگه بخوایم ساده‌ترش کنیم :

$$C = \underline{x'yz} + xy'z + xyz' + \underline{xyz}$$

$$= yz(x+x') + \underbrace{\underbrace{xy'z + xyz'}_{x(y'z+yz')}}_{y\oplus z} = yz + x(y\oplus z)$$

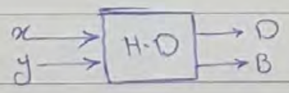
ترکیبات قبلی چون گیت سه ورودی نداریم
در نهایت از هفت گیت استفاده می‌شود
بنابراین بهتر است که تا جای ممکن
تابع نهایی را ساده کنیم.



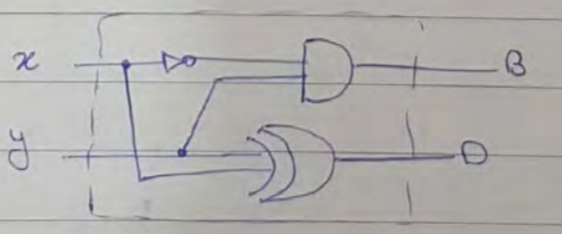
② تفریق کننده‌ها — نیم تفریق کننده

(Half Differentiator)

الف) نیم تفریق کننده: مداریست ترکیبی که دو بیت را از هم کم کرده و تفاضل را در خروجی به ما میدهد. این مدار خروجی دیگری نیز دارد که مشخص کننده عمل قرض کردن (Borrow) از ستون بعدی عدد میباشد.

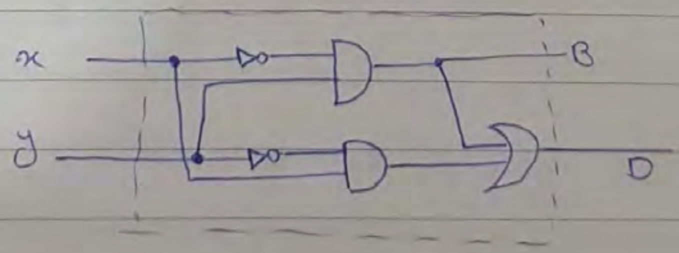


x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



$B = x'y$

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

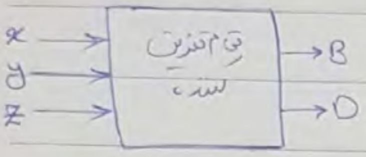


$x - y = ?$

Subject . Year. Month. Day.

ب) تمام تفریق کننده: مداری ترکیبیست که عمل تفریق را بین دو بیت با توجه به یک واحد قرض گرفته شده در طبقه قبلی انجام میدهد.

(Full Differentiator)



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

B:

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

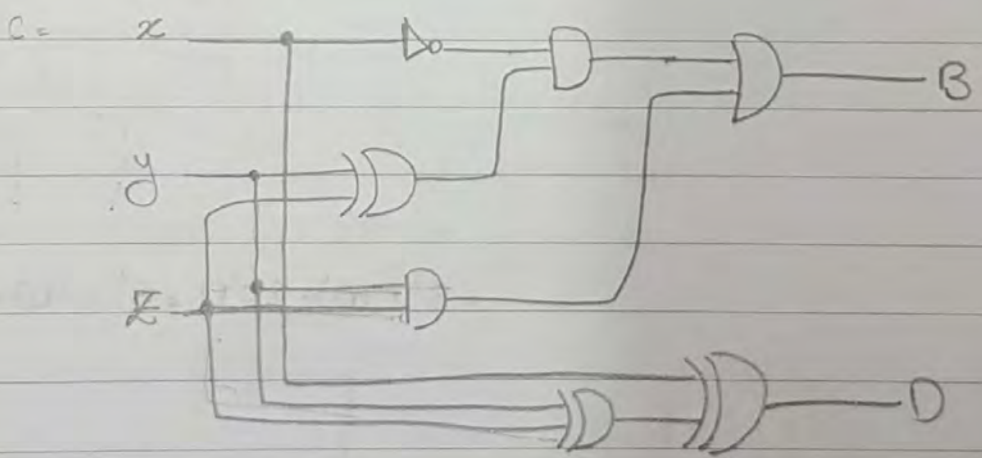
D:

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

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

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

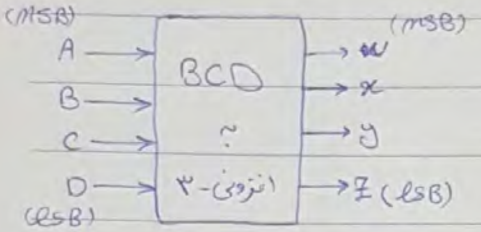
$$B = x'(y'z + yz) + yz(x + x') = x'(y \oplus z) + yz$$



$$D = x'y'z + x'yz' + xy'z' + xyz = x'\underbrace{(y'z + yz')}_{y \oplus z} + x\underbrace{(y'z' + yz)}_{y \odot z}$$

$$= \underbrace{x'}_{A}\underbrace{(y \oplus z)}_{B} + \underbrace{x}_{A'}\underbrace{(y \oplus z)'}_{B'} = x \oplus y \oplus z$$

تبدیل کدها: میخواهیم مداری طراحی کنیم که عدد BCD (0-9) را به عنوان ورودی بپذیرد و کد افزونی-۳ معادل آن را در خروجی نمایش دهد.



ورودی ها: اعداد BCD از ۰ تا ۹

خروجی: افزونی سه اعداد وارد شده

$w = f(A,B,C,D) = ?$

$x = f(A,B,C,D) = ?$

⋮

	A	B	C	D	w	x	y	z
0→	0	0	0	0	0	0	1	1
1→	0	0	0	1	0	1	0	0
2→	0	0	1	0	0	1	0	1
3→	0	0	1	1	0	1	1	0
4→	0	1	0	0	0	1	1	1
5→	0	1	0	1	1	0	0	0
6→	0	1	1	0	1	0	0	1
7→	0	1	1	1	1	0	1	0
8→	1	0	0	0	1	0	1	1
9→	1	0	0	1	1	1	0	0
10→	1	0	1	0	x	x	x	x
11→	1	0	1	1	x	x	x	x
12→	1	1	0	0			⋮	
13→	1	1	0	1			⋮	
14→	1	1	1	0				
15→	1	1	1	1	x	x	x	x

حالات بی اهمیت

Don't care

Z: AB \ CD

AB \ CD	00	01	11	10
00	1			1
01	1			1
11	X	X	X	X
10	1			X

$Z = D'$

x: AB \ CD

AB \ CD	00	01	11	10
00		1	1	1
01	1			
11	X	X	X	X
10		1	X	X

$x = B'C + B'D + BC'D'$

$= B'(C+D) + BC'D = B'(C+D) + B(C+D)'$

$B \oplus C \oplus D$

y: AB \ CD

AB \ CD	00	01	11	10
00	1		1	
01	1		1	
11	X	X	X	X
10	1		X	X

$y = CD + C'D'$

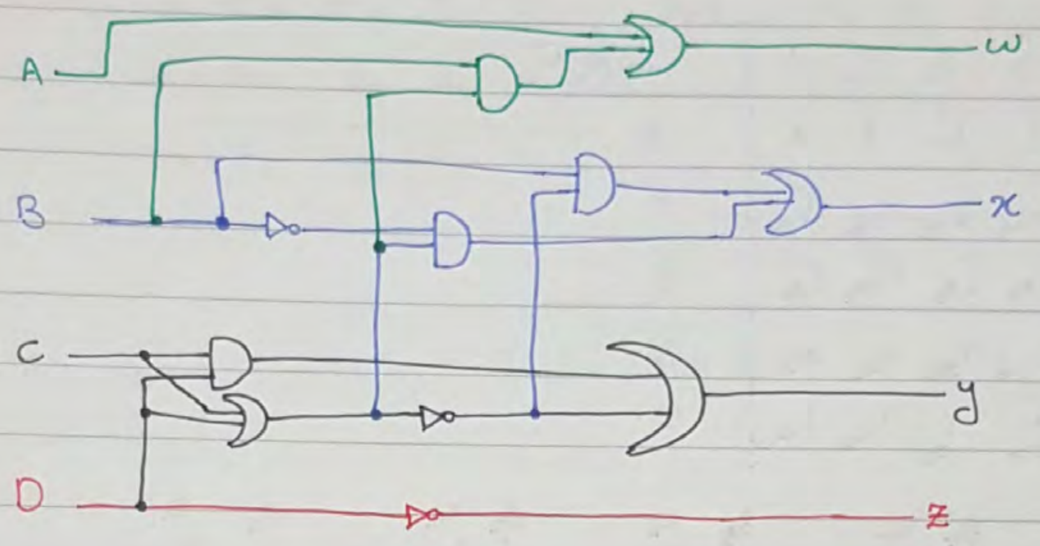
$= CD + (C+D)'$

w: AB \ CD

AB \ CD	00	01	11	10
00				
01		1	1	1
11	X	X	X	X
10	1	1	X	X

$w = A + BC + BD$

$= A + B(C+D)$



مدار، ۹۸/۹/۱۹ ۸:۰۰-۹:۴۵ سه‌شنبه

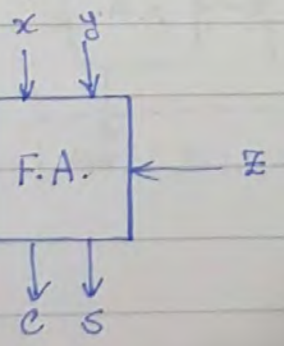
$$\begin{array}{r} {}^{1}\;\;\;\; \\ 1001 \\ 1101 \\ \hline 10110 \end{array}$$

جمع کننده باینری:
- جمع کننده سری
- جمع کننده موازی ✓

مدار تمام جمع کننده که قبلا معرفی شد مجموع دو بیت و بیت نقلی طبقه قبلی را به ما میدهد. برای جمع دو عدد n بیتی به روش موازی نیاز به n جمع کننده میباشد. شایان ذکر است جمع دو عدد n بیتی a و b به دو صورت میتواند انجام شود که:

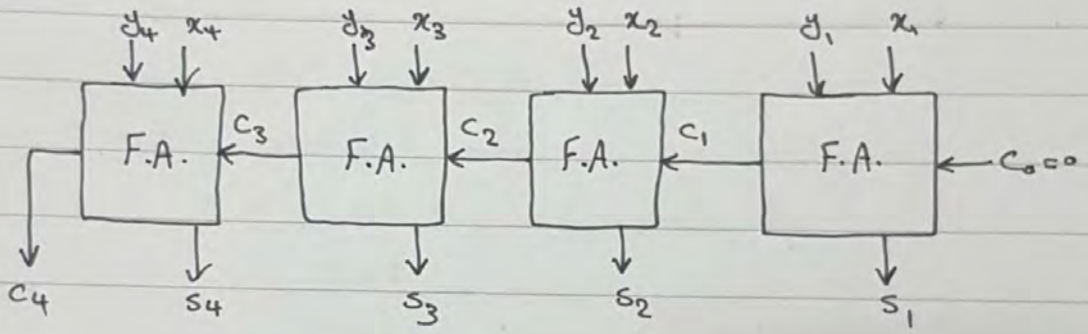
۱. در روش جمع سری فقط یک تمام جمع کننده به کار میرود.

۲. در روش موازی n مدار تمام جمع کننده به کار برده میشود و تمام n بیت های a و b را به طور همزمان به این جمع کننده ها وارد میکند.



$n=4$

	4	3	2	1	قام جمع کننده
	1	1	1	0	Z
$A \to$	0_{x_4}	1_{x_3}	1_{x_2}	1_{x_1}	x
$B \to$	0_{y_4}	1_{y_3}	0_{y_2}	1_{y_1}	y
	1_{s_4}	1_{s_3}	0_{s_2}	0_{s_1}	S
	0_{c_4}	1_{c_3}	1_{c_2}	1_{c_1}	C



☆ جمع کننده ـ تفریق کر باینری:

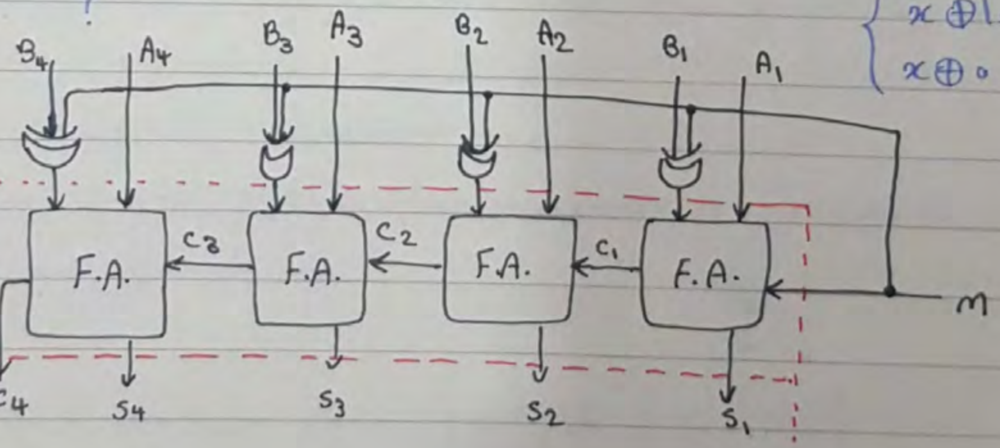
$A_4 \; A_3 \; A_2 \; A_1$

$\pm \; B_4 \; B_3 \; B_2 \; B_1$

?

$\{1 \oplus 0 = 1,\ 1 \oplus 1 = 0,\ 0 \oplus 1 = 1,\ 0 \oplus 0 = 0\}$

$$\begin{cases} x \oplus 1 = \bar{x} \\ x \oplus 0 = x \end{cases}$$



$M=0 \longrightarrow$ جمع کننده $\to A+B$

$A=1 \longrightarrow$ تفریق کننده $\to A-B$

Scanned with CamScanner

$$
\begin{array}{rcccc}
M=0 \rightarrow & A_4 & A_3 & A_2 & A_1 \\
+ & B_4 & B_3 & B_2 & B_1 \\
 & & & & +0 \\
\hline
C_4 & S_4 & S_3 & S_2 & S_1
\end{array}
\qquad
\begin{array}{rcccc}
M=1 \rightarrow & A_4 & A_3 & A_2 & A_1 \\
+ & B'_4 & B'_3 & B'_2 & B'_1 \\
+ & & & & 1 \\
\hline
C_4 & S_4 & S_3 & S_2 & S_1
\end{array}
$$

$A + ($ مکمل B $2) = A - B$ ⇐ مکمل $\underline{2}$

انتشار رقم نقلی :

در یک جمع کننده طولانی ترین زمان انتشار زمان تاخیری است که یک رقم نقلی برای انتشار در طول جمع کننده به آن نیاز دارد .

برای روشن شدن مطلب خروجی S_4 را در جمع کننده در نظر میگیریم .

ورودی A_4 و B_4 مستقیما به این مدار وارد میشوند ولی تا زمانی که رقم نقلی C_3 به مقدار پایه دار خود نرسد مقدار خروجی نقلی C_4 به وضعیت نهایی و پایه دار خود نمیرسد.

این مسئله به طور مشابه برای C_2 و C_3 نیز وجود دارد .

برای محاسبه انتشار رقم نقلی برحسب تعداد طبقات گیت ها باید به سراغ ساختمان مدار تمام جمع کننده برویم . زمان لازم جهت انتشار رقم نقلی در یک تمام جمع کننده برابر است با تاخیر دو گیت .

در حالتی که دو عدد چهار بیتی با هم جمع میشوند خروجی نقلی C_5 پس از تاخیر $2\times4=8$ طبقه گیت به مقدار نهایی خود میرسد .

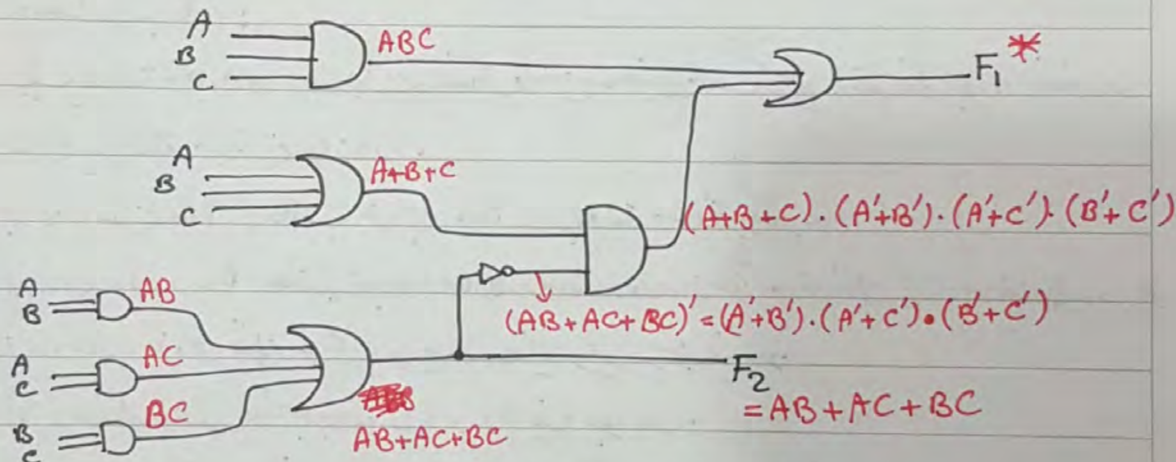
در نتیجه برای جمع کننده موازی n بیتی زمان انتشار رقم نقلی برای عبور از طبقات برابر زمان $2n$ طبقه گیت میباشد .

نکته: کل زمان انتشار در جمع کننده موازی چهار بیتی برابر زمان انتشار در یک نیم جمع کننده به علاوه زمان انتشار هشت طبقه گیت خواهد بود.

تحلیل مدارهای ترکیبی :

روش تحلیل مدارهای ترکیبی عکس روش طراحی آن میباشد به این معنی که با مدار منطقی آغاز و با توابع منطقی یا توزیع و طرز کار آن پایان می یابد. برای این کار از ابتدای مدار منطقی شروع کرده (از سمتی که ورودی ها وارد میشوند.) و خروجی ها را بدست می آوریم و به سمت انتهای مدار میرویم.

مثال) خروجی های F_1 و F_2 را در مدار زیر بیابید.

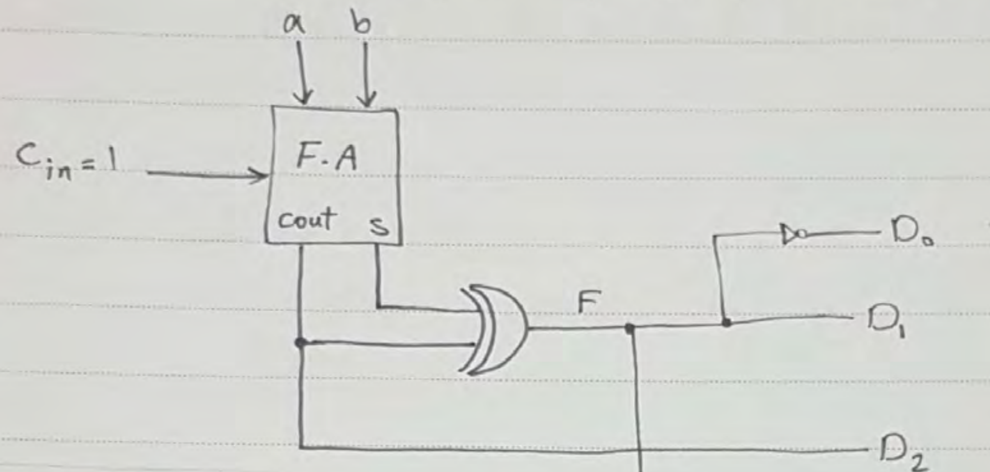


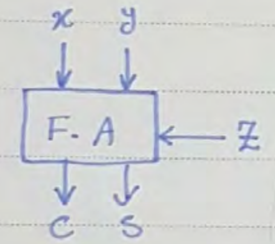
$$* \quad F_1 = ABC + \underbrace{\left[(A+B+C)\cdot(A'+B')\cdot(A'+C')\cdot(B'+C') \right]}_{A'BC'+A'B'C+AB'C'}$$

$$\rightarrow \begin{cases} F_1 = ABC + A'BC' + A'B'C + AB'C' \\ F_2 = AB + AC + BC \end{cases}$$

Subject:
Date

در مدار شکل زیر خروجی های d_0 ، d_1 ، d_2 را برحسب a و b بنویسید.





$S = x \oplus y \oplus z$

$C = x + y$

$S = a \oplus b \oplus C_{in} = a \oplus \underbrace{b \oplus 1}_{b'} = a \oplus b' = ab + a'b'$

$C_{out} = a + b$

$D_0 = F' = (S \oplus C_{out})' = S \odot C_{out} = (ab + a'b') \odot (a+b) = \dots$

$D_1 = F = (S \oplus C_{out}) = (ab + a'b') \oplus (a+b) = \dots$

$D_2 = C_{out} = a + b$

$D_3 = F + \bar{C}_{in} = F \oplus 0 = F = S \oplus C_{out} = (ab + a'b') \oplus (a+b) = \dots$

$\boxed{D_1 = D_3}$

Subject:
Date

مقایسه‌گر دو مقدار :

مقایسه‌گر یک مدار ترکیبی است که دو عدد A و B را مقایسه کرده و مقدارشان را نسبت به هم تعیین می‌کند.

$\{A=B \,,\, A>B \,,\, A<B\}$

$A \to$ n بیتی $\Rightarrow 2^n$

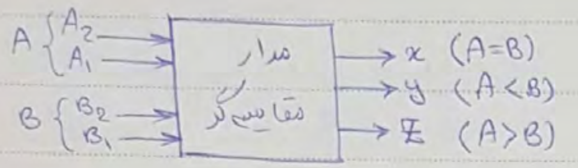
$B \to$ n بیتی $\Rightarrow 2^n$

$2^n \times 2^n = 2^{2n}$

if $n=2 \longrightarrow \underbrace{A, B}_{\text{2 بیتی}} \to \begin{cases} A = A_2A_1 \\ B = B_2B_1 \end{cases}$

A_2	A_1	B_2	B_1	x	y	z
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

Subject:
Date



$$\begin{cases} \text{if } A=B \Rightarrow X=1 \ , \ y=Z=0 \\ \text{if } A<B \Rightarrow y=1 \ , \ x=Z=0 \\ \text{if } A>B \rightarrow Z=1 \ , \ x=y=0 \end{cases}$$

$$X(A_2, A_1, B_2, B_1) = \sum m(0, 5, 10, 15)$$

$B_2B_1 \backslash A_2A_1$	00	01	11	10
00	1			
01		1		
11			1	
10				1

$$X = A_2'A_1'B_2'B_1' + A_2'A_1B_2'B_1 + A_2A_1B_2B_1 + A_2A_1'B_2B_1'$$

طبق جدول درستی $\rightarrow$
$$\begin{cases} Y = \sum m(1, 2, 3, 6, 7, 11) \\ Z = \sum m(4, 8, 9, 12, 13, 14) \end{cases}$$

استفاده از روش الگوریتمی:

$A = A_4 A_3 A_2 A_1$

$B = B_4 B_3 B_2 B_1$

$x_i = A_i B_i + A_i' B_i' \qquad i = 1, 2, 3, 4$

درصورتی $x_i = 1$ است که بیت های هم مکان با هم برابر باشند.

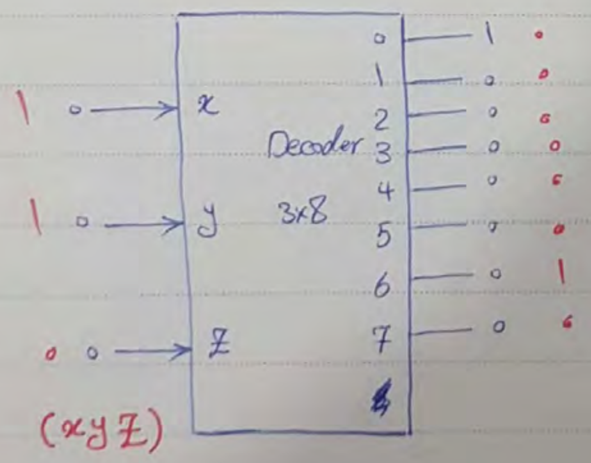
$$(A>B) = A_4 B_4' + x_4 A_3 B_3' + x_4 x_3 A_2 B_2' + x_4 x_3 x_2 A_1 B_1'$$

$$(A<B) = A_4' B_4 + x_4 A_3' B_3 + x_4 x_3 A_2' B_2 + x_4 x_3 x_2 A_1' B_1$$

$$(A=B) = x_4 x_3 x_2 x_1$$

Decoders and Encoders: دیکدرها و انکدرها:

دیکدر یک مدار ترکیبی است که اطلاعات باینری n خط ورودی را به حداکثر 2^n خط خروجی یا مینترم تبدیل می کند. به عنوان مثال دیکدر 3 به 8 سه ورودی را به هشت خروجی دیکد میکند.
هر خروجی نمایش یکی از مینترم های متغیرهای ورودی است.
یکی از کاربرد های این مدار تبدیل کد باینری به کد مبنای هشت است.



برخی از مدارهای دیکدر را در قالب مدارهای مجتمع (IC) عرضه میکنند.

به عنوان مثال IC یا تراشه 7447 و 7448 تراشه‌های دیکدر هستند که سه ورودی و هفت خروجی دارند. این تراشه‌ها در حقیقت دیکدر BCD به معادل 7 segment هستند.

یا تراشه 74138 یک دیکدر BCD به معادل دهدهی است.

پیاده سازی مدارهای ترکیبی با دیکدر:

هر تابع بول به فرم کانونیک را میتوان با مجموع مینترم ها نمایش داد.

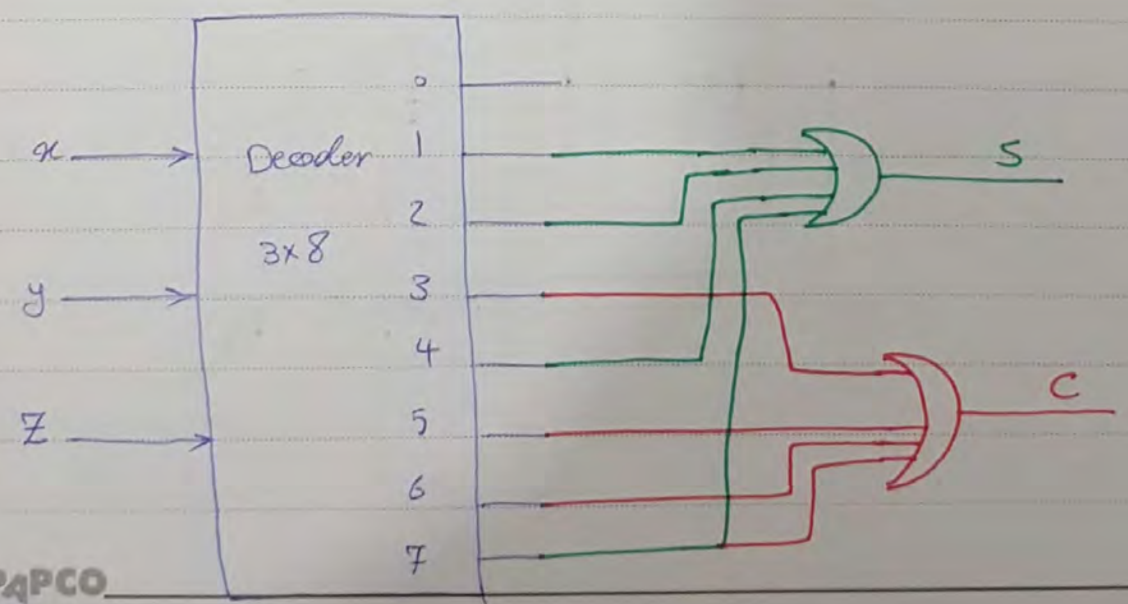
دیکدر برای تولید مینترم ها به کار میرود و به وسیله یک گیت OR میتوان مجموع مینترم ها را تشکیل داد.

بنابراین هر مدار ترکیبی با n ورودی و m خروجی با یک دیکدر n به 2^n و یک گیت OR با m ورودی قابل پیاده سازی است.

مثال) مدار تمام جمع کننده را با یک دیکدر و دو گیت OR پیاده سازی نمایید.

$$S = x'y'z + x'yz' + xy'z' + xyz = \sum m(1,2,4,7)$$

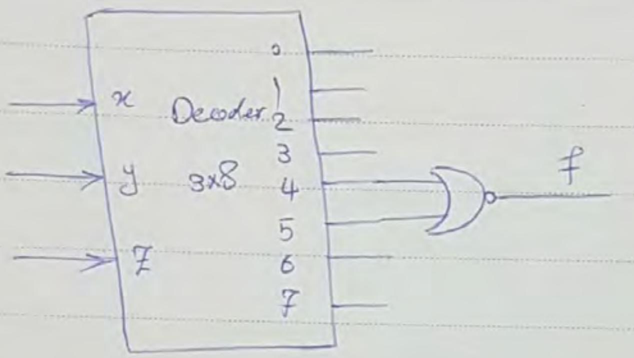
$$C = x'yz + xy'z + xyz' + xyz = \sum m(3,5,6,7)$$



در صورتی که تابع دارای تعداد زیادی مینترم باشد گیت OR دارای تعداد زیادی ورودی میشود، در این حالت به جای تابع f مکمل آن یعنی تابع f' را پیاده‌سازی میکنیم و یک گیت Not در خروجی آن قرار میدهیم.

به عبارت دیگر آن مینترم هایی را که در تابع f وجود ندارند با هم NOR میکنیم.

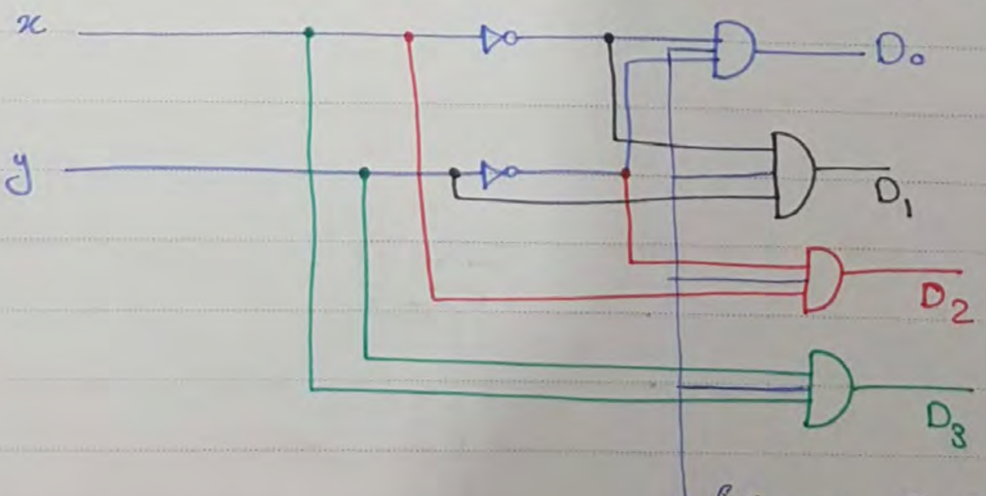
* معمولا اگر تعداد مینترم ها از $\frac{2^n}{2}$ بیشتر بود با استفاده از f' تابع f را پیاده سازی میکنیم.



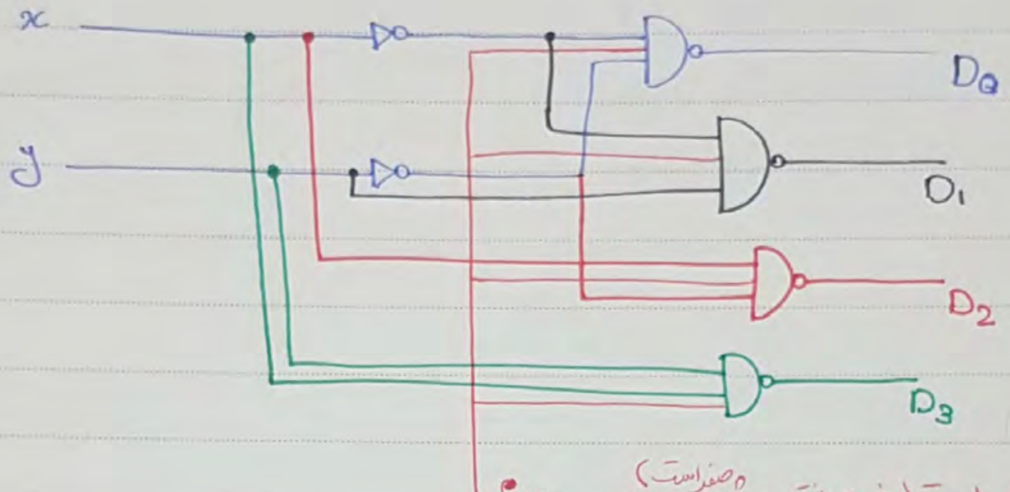
$f(x,y,z) = \sum(0,1,2,3,6,7)$

در بعضی حالات که مدار ترکیبی دارای تعداد زیادی خروجی است و هر خروجی یا مکمل آن با تعداد کمی از مینترم ها بیان شود روش پیاده سازی با استفاده از دیکدر بهترین حالت است.

دیکدر با فعال ساز:

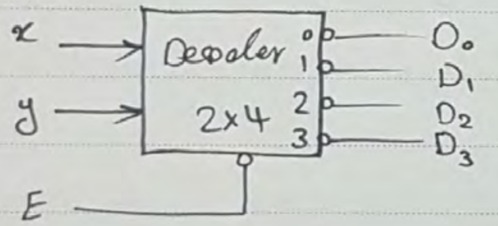


E { E=0 → تمام خروجی ها صفر هستند
E=1 → دیکدر فعال است



$E=0 \rightarrow$ دیکدر فعال است (خروجی انتخاب شده صفر است)

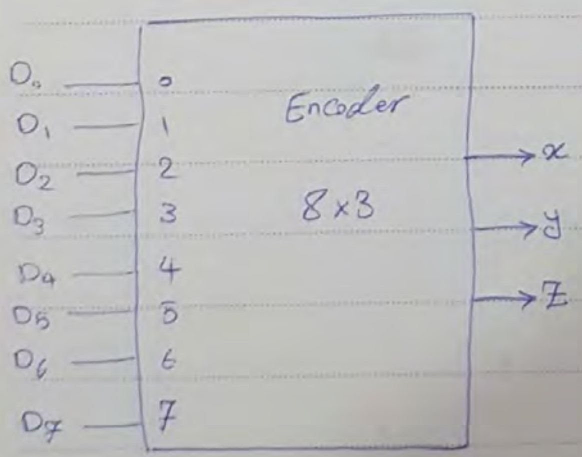
$E=1 \rightarrow$ تمام خروجی ها یک هستند



Encoders:

انکدرها:

انکدر یک مدار دیجیتال است که عکس عمل دیکدر را انجام میدهد یعنی دارای 2^n خط ورودی و n خط خروجی است که خط های خروجی کد باینری شماره متغیر ورودی را تولید میکند.



D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	x	y	z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
			⋮						⋮	
1	0	0	0	0	0	0	0	1	1	1

Subject:
Date

$$
\begin{cases}
x = D_4 + D_5 + D_6 + D_7 \\
y = D_2 + D_3 + D_6 + D_7 \\
z = D_1 + D_3 + D_5 + D_7
\end{cases}
$$

در انکدر شکل بالا :

1 اگر دو ورودی در یک زمان فعال شوند خروجی ترکیبی تولید میشود که صحیح نیست . مثلاً اگر D_3 و D_4 همزمان یک شوند خروجی بصورت ۱۱۱ میشود که معرف D_7 است .

2 هنگامی که همه ورودی ها صفر باشند، همه خروجی ها صفر هستند. از طرفی هنگامی که D_0 نیز برابر صفر باشد باز هم تمام خروجی ها برابر صفر میشود .

مدار ۳، ۱۰، ۹۸   ۸:۰۰ - ۹:۴۵

انکدر اولویت دار :

یک مدار ترکیبی است که به ورودی ها اولویت می دهد . این انکدر به نحوی کار میکند که وقتی بیش از یک ورودی همزمان (یک) هستند ورودی دارای اولویت بیشتر پیک می کند .

ورودی ها				خروجی ها		
D_0	D_1	D_2	D_3	x	y	V ← بیت اعتبار
۰	۰	۰	۰	X	X	۰
۱	۰	۰	۰	۰	۰	۱
X	۱	۰	۰	۰	۱	۱
X	X	۱	۰	۱	۰	۱
X	X	X	۱	۱	۱	۱

Subject:
Date

$x = f(D_0, D_1, D_2, D_3) = ?$

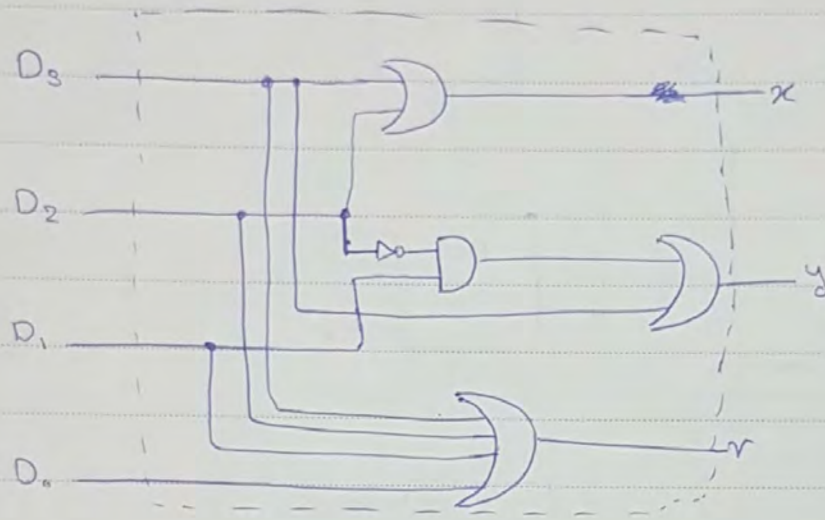
$y = f(D_0, D_1, D_2, D_3) = ? \xrightarrow[\text{با نقشه کارنو}]{\text{ساده‌سازی}}$

$v = f(D_0, D_1, D_2, D_3) = ?$

$x = D_2 + D_3$

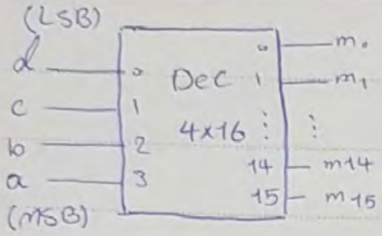
$y = D_3 + D_1 D_2'$

$v = D_0 + D_1 + D_2 + D_3$

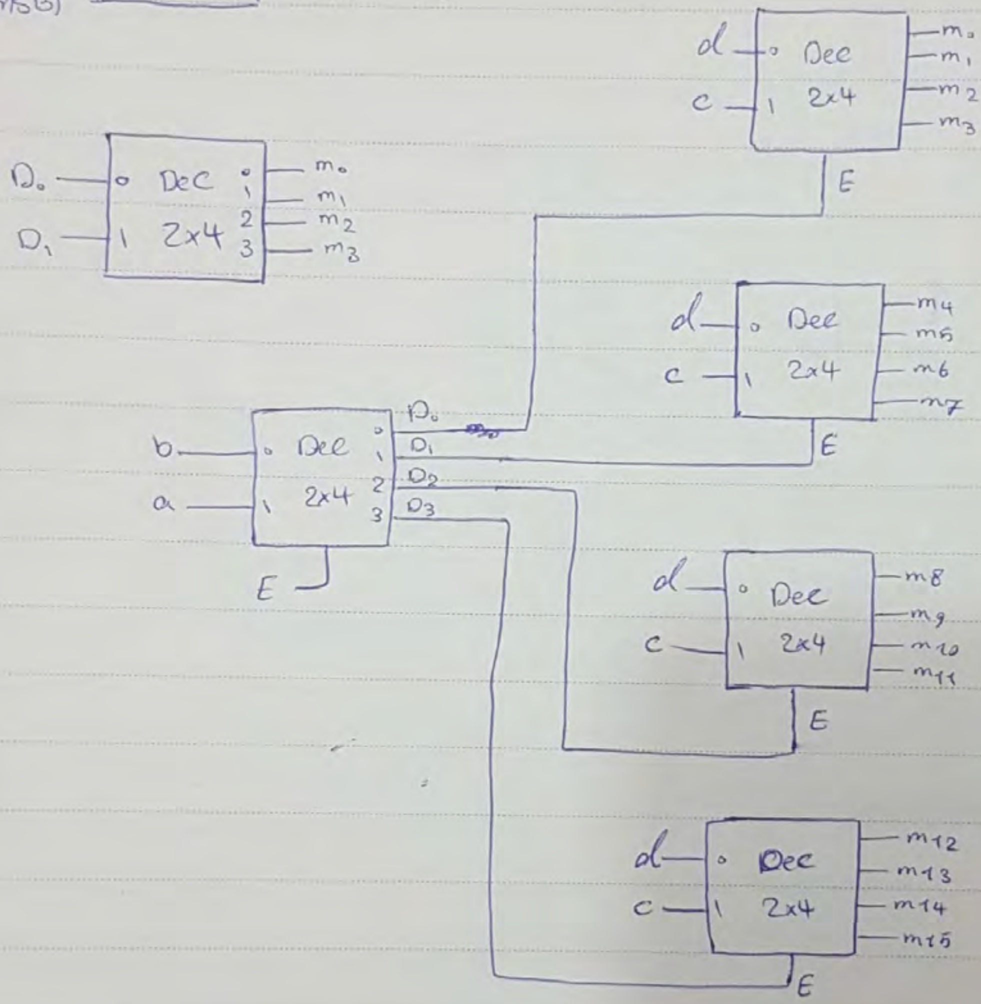


ساده‌سازی با نقشه کارنو به عهده خودتان.

Subject:
Date



با استفاده از دیکدر ۲x۴ یک دیکدر ۴x۱۶ بسازید.



ab = 00 ⟶ D0 = 1 ⟶ cd = 00, 01, 10, 11

0000 ⟶ m0
0001 ⟶ m1
0010 ⟶ m2
0011 ⟶ m3

ab = 01 ⟶ D1 = 1 ⟶ دیکدر دوم ⇒ cd = 00 … 11

abcd = 0100 ⟶ m4
0101 ⟶ m5
0110 ⟶ m6
0111 ⟶ m7

ab = 10 ⟶ D2 = 1 ⟶ دیکدر سوم ⇒ cd = 00 … 11

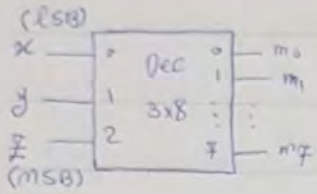
abcd = 1000 ⟶ m8
1001 ⟶ m9
1010 ⟶ m10
1011 ⟶ m11

ab = 11 ⟶ D3 = 1 ⟶ دیکدر چهارم ⇒ cD = 00 … 11

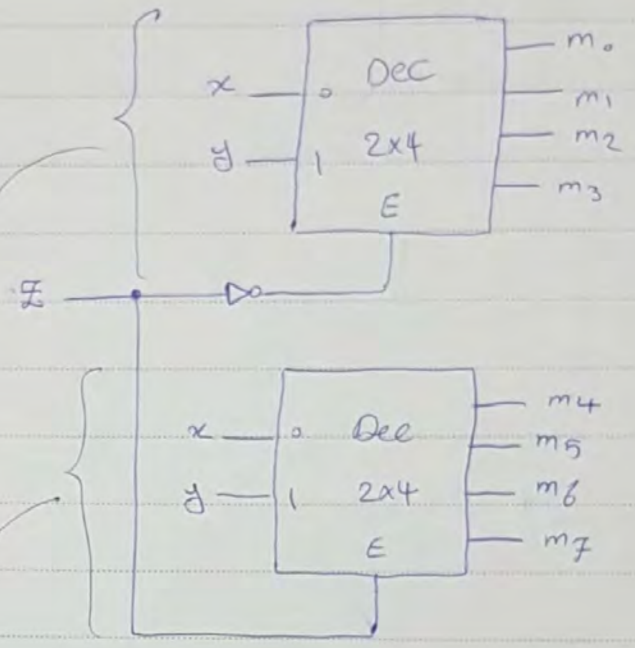
abcd = 1100 ⟶ m12
1101 ⟶ m13
1110 ⟶ m14
1111 ⟶ m15

Subject:
Date

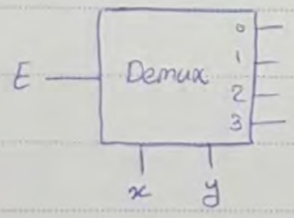
یک دیکدر 3x8 را با استفاده از دو دیکدر 2x4 بسازید.



z	y	x	
0	0	0	0-3
0	0	1	
0	1	0	
0	1	1	
1	0	0	4-7
1	0	1	
1	1	0	
1	1	1	



دی مالتی پلکسر
(Demultiplexer)
(Demux)



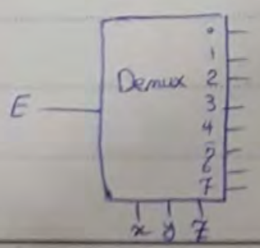
$E=0 \rightarrow$ همه خروجی ها صفر هستند

$E=1 \xrightarrow{xy=10}$ خروجی 2 برابر یک میشود

فعال ساز در اینجا به عنوان ورودی است، x و y در شکل بالا به عنوان خطوط انتخاب (selection line) هستند که به آنها خطوط آدرس نیز گفته میشود.

دیکدر با فعال ساز میتواند به عنوان دی مالتی پلکسر کار کند.

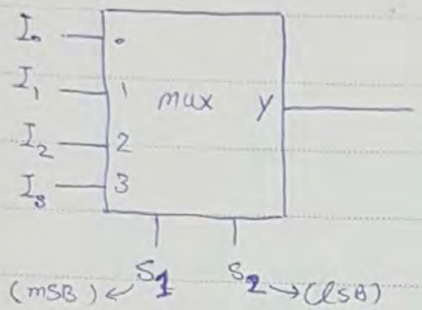
دی مالتی پلکسر مداریست که اطلاعات را از یک خط دریافت می کند و آن را به یکی از 2^n خط خروجی انتقال میدهد. انتخاب یک خط خروجی خاص به وسیله مقدار بیت n خط انتخاب (selection line) مشخص میگردد.



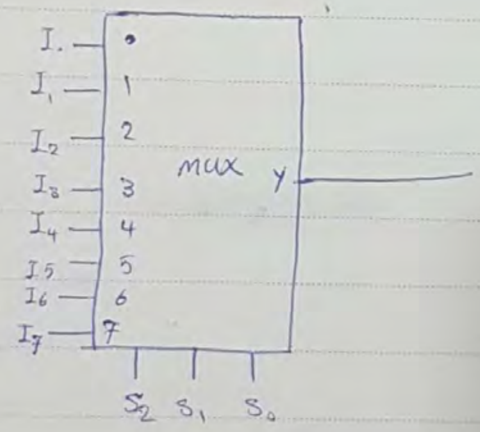
نکته

نکته: مدارهای دیکدر روی مالتی پلکسر را می توان به گونه ای به هم متصل نمود که یک دیکدر بزرگتر حاصل شود.

عملکرد مالتی پلکسر انتقال تعداد زیادی واحدهای اطلاعات بر روی تعداد کمتری از کانال یا خطوط میباشد.
مولتی پلکسر دیجیتال یک مدار ترکیبی است که اطلاعات باینری را از بین یکی از خط های ورودی انتخاب می کند و به تنها خروجی خود هدایت می کند که این انتخاب با استفاده از خطوط انتخاب انجام می شود.



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



$S_2 S_1 S_0 = 011 \rightarrow Y = I_3$

$S_2 S_1 S_0 = 101 \rightarrow y = I_5$

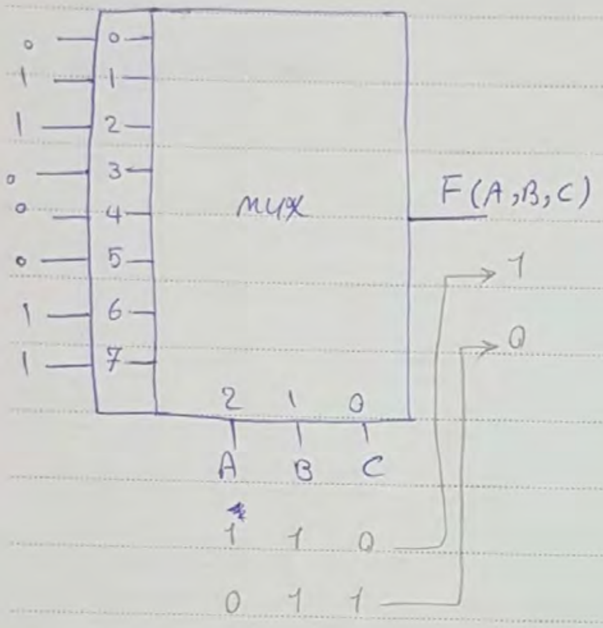
پیاده سازی تابع بول با مالتی پلکسر:
روش اول: در این روش مینترم هایی را که در تابع وجود دارند با یک کردن خطوط ورودی نظیر در مالتی پلکسر انتخاب میکنیم لذا در این روش برای پیاده کردن تابع بول n متغیره یک مالتی پلکسر با 2^n ورودی نیاز میباشد.
روش دوم: در این روش اگر تابع بول $n+1$ متغیر داشته باشد n متغیر را به خط های انتخاب کننده مالتی پلکسر متصل می کنیم و یک ورودی باقیمانده را به عنوان ورودی مالتی پلکسر استفاده می کنیم. در این روش اگر متغیر A به عنوان ورودی انتخاب شده باشد ورودی های مالتی پلکسر مقادیر A، A'، 0 یا 1 خواهند بود.

Subject:
Date

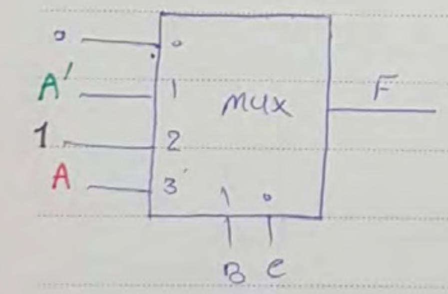
مثال) تابع زیر را با دو روش بیان شده پیاده سازی کنید.

$F(A,B,C) = \Sigma(1,2,6,7)$

(روش اول) $n=3 \longrightarrow$ mux ورودی های $= 2^n = 8$



(روش دوم) $n+1=3 \longrightarrow n=2$



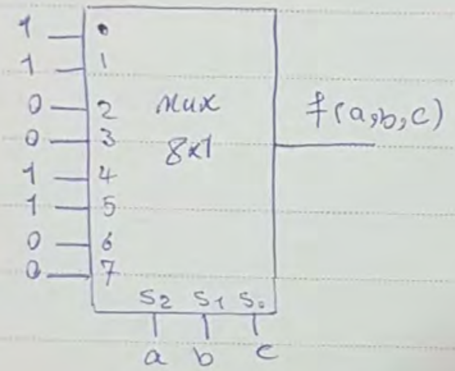
A	B	C	F(A,B,C)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

به عنوان مثال در مورد عدد صفر هر دو متغیر B و C مقدار صفر دارند حال در جدول این مورد را پیدا میکنیم. دو حالت وجود دارد که B و C هر دو صفر هستند. مشاهده میشود که در این حالت یکی $A=0$ است که برای آن $F=0$ است و همچنین دیگری $A=1$ و برای آن $F=0$ است پس نتیجه میگیریم F مستقل از A میباشد بنابراین مقدار $A=0$ است. (مقدار ورودی مالتی پلکسر برابر صفر است).

تابع زیر را با mux 8x1 ، mux 4x1 و mux 2x1 پیاده سازی کنید.

$$f(a,b,c) = \sum(0,1,4,5)$$

mux 8x1:



mux 4x1:

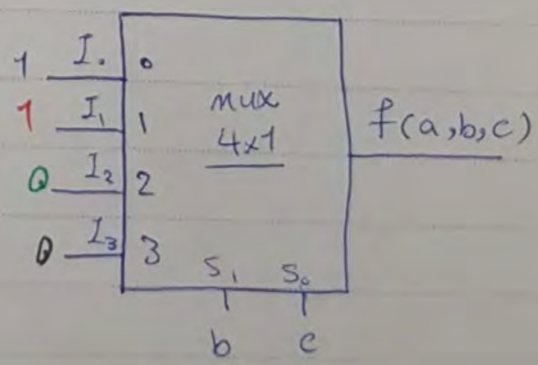
a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

$bc = 00 \rightarrow a = 0 \rightarrow f = 1$
$\quad\quad\quad\;\, \searrow a = 1 \rightarrow f = 1$

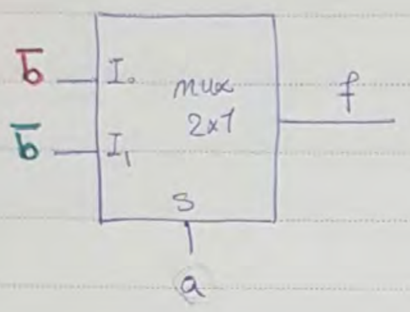
$bc = 01 \rightarrow a = 0 \rightarrow f = 1$
$\quad\quad\quad\;\, \searrow a = 1 \rightarrow f = 1$

$bc = 10 \rightarrow a = 0 \rightarrow f = 0$
$\quad\quad\quad\;\, \searrow a = 1 \rightarrow f = 0$

$bc = 11 \rightarrow a = 0 \rightarrow f = 0$
$\quad\quad\quad\;\, \searrow a = 1 \rightarrow f = 0$



mux 2x1:

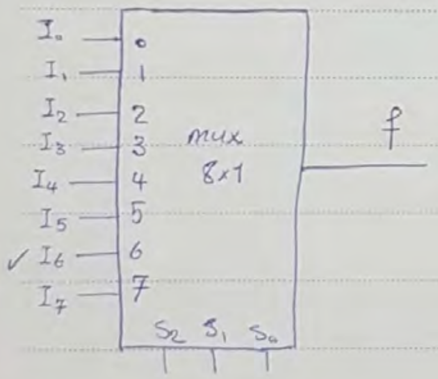


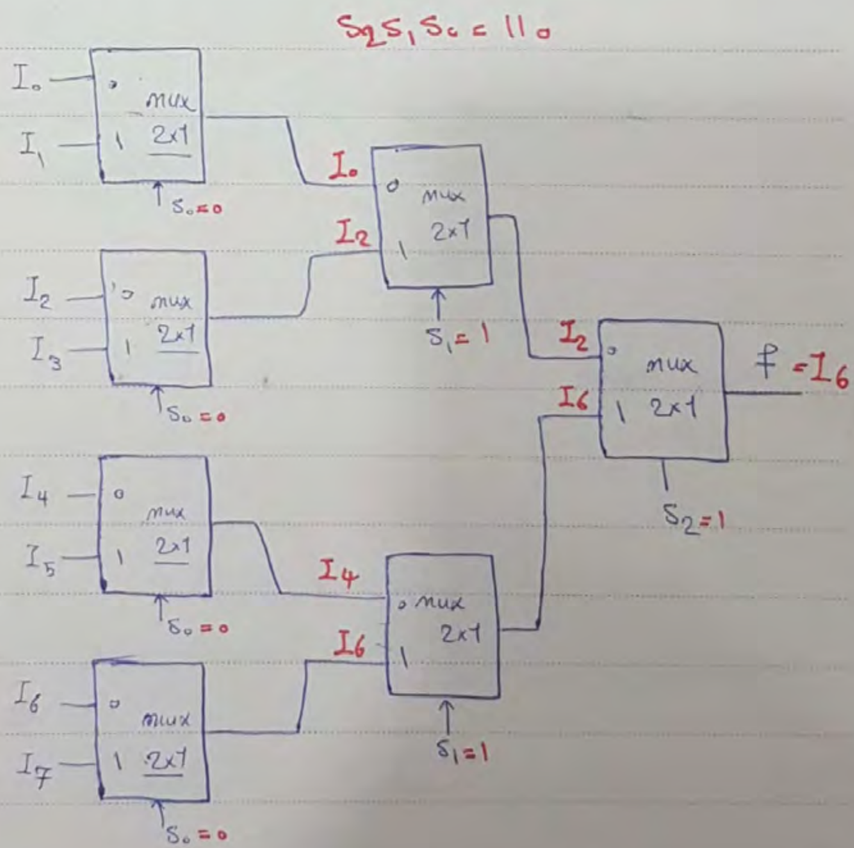
a	b	c	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

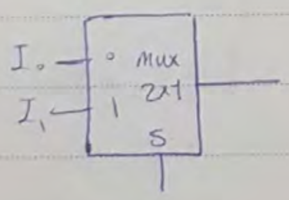
$I_0 = \bar b$

$I_1 = \bar b$

بااستفاده از mux2x1 یک ماکس 8x1 بسازید.





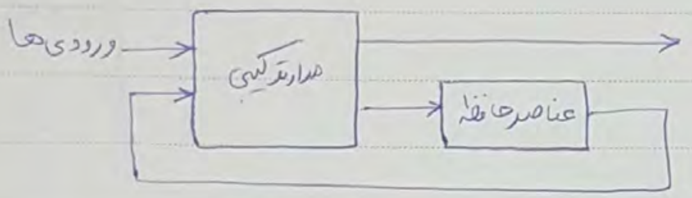


Subject:
Date

فصل پنجم: مدارهای ترتیبی سنکرون:

مدارهای ترتیبی مدارهای منطقی هستند که در آن ترتیب اعمال ورودی ها مهم می باشد به عبارت دیگر خروجی یک مدار ترتیبی به ورودی در همان لحظه و لحظات قبل بستگی دارد.

همان طور که در بلوک دیاگرام شکل زیر مشاهده میکنید خروجی مدار ترتیبی نه فقط تابع ورودی ها ست بلکه تابع حالت فعلی عناصر حافظه نیز می باشد.

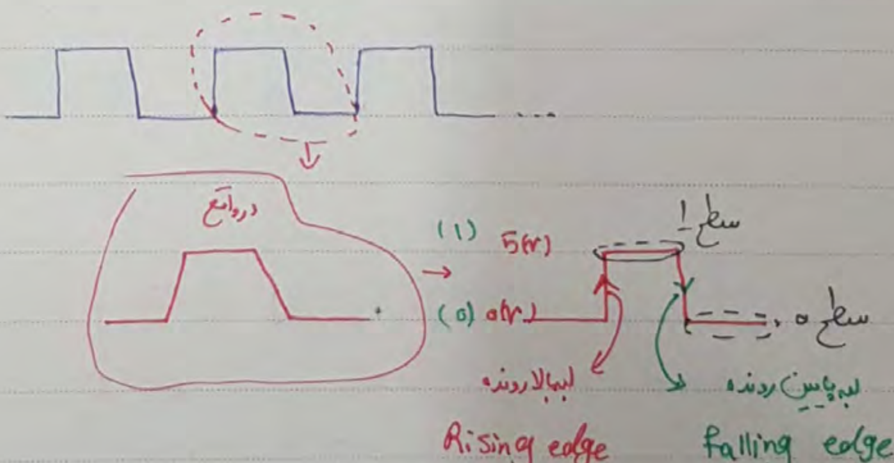


(بلوک دیاگرام یک مدار ترتیبی)

مدارهای ترتیبی:

- سنکرون (همزمان): مدارهایی که در آنها عملیات توسط یک پالس ساعت کنترل می شود. (Synchronous)
- آسنکرون (غیرهمزمان): مدارهایی که در آنها شروع عملیات به عهده ورودی ها است. (Asynchronous)

پالس ساعت
clock Pulse
CP
clk



دو عنصر حافظه در مدارهای سوئیچینگ (switching)، لچ‌ها (latch) و فلیپ فلاپ‌ها (flip-flop) هستند. تفاوت این دو در وجود یا عدم وجود سیگنال کنترلی پالس ساعت است.

فلیپ فلاپ‌ها: فلیپ فلاپ یک عنصر حافظه است که یک حالت باینری را به طور دائم در خود نگه میدارد مگر اینکه به وسیله سیگنال ورودی به حالت دیگری برود. اختلاف اساسی بین فلیپ فلاپ‌های مختلف در تعداد ورودی‌ها و همچنین نحوه تاثیر این ورودی‌ها بر حالت فلیپ فلاپ‌ها میباشد.

منظور از حالت فلیپ فلاپ خروجی آن است.

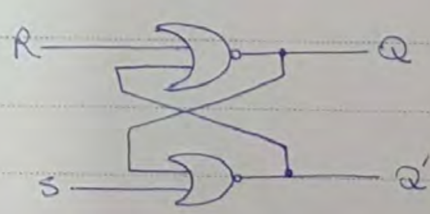
انواع فلیپ فلاپ‌ها:

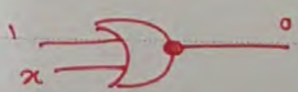
1- فلیپ فلاپ پایه (مدار اساسی فلیپ فلاپ): فلیپ فلاپ‌های زیر فلیپ فلاپ پایه میباشند که انواع پیچیده فلیپ فلاپ‌ها براساس آنها ساخته میشوند.

از آنجا که در این مدارها یک اتصال ضربدری از خروجی یک گیت به ورودی گیت دیگر یک مسیر فیدبک (feed back) از خروجی به ورودی ایجاد میکند لذا این مدار جزء مدارهای ترتیبی میباشد.

جدول صحت‌ها

	S	R	Q	Q'	
	1	0	1	0	R: Reset
(بعداز S=1 و R=0)	0	0	1	0	S: set
	0	1	0	1	
(بعداز S=0 و R=1)	0	0	0	1	Q = خروجی یا حالت فلیپ فلاپ
تعریف نشده	1	1	0	0	Q' = مکمل خروجی فلیپ فلاپ





حالتی را در نظر میگیریم که S = 1 و R = 0 باشد در نتیجه گیت Q میشه NOR بین صفر و هر چیزی که میشه 1 حالا یک وارد گیت دوم میشه و Q' میشه

صفر یک به هر چیزی میشه صفر حالا Q و Q' میان تو گیت‌ها قرار میگیرن. حالا اگه S و R رو بدیم صفر به ما خروجی NOR میده Q = 1

در فلیپ فلاپ نقاط حالت صفر صفر هست که قبلش چی بوده و خروجی حالت به هم مربوط نیستن.

Subject:
Date

مدار 5 دی

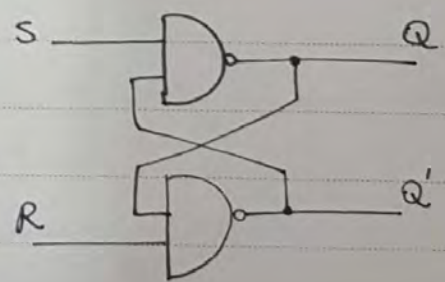
جبرانی 1 ۸:۰۰ - ۹:۴۵

S	R	Q	Q'	
۱	۰	۱	۰	
۰	۰	۱	۰	(بعد از R=۰, S=۱)
۰	۱	۰	۱	
۰	۰	۰	۱	(بعد از R=۱, S=۰)
۱	۱	۰	۰	(تعریف نشده)

در حالت اول که S مساوی یک است Q' حتماً صفر است. حال NOR بالا این ورودی صفر را دارد و R هم ورودیش هم صفر است پس Q مساوی یک میشود.

در حالت دوم که R=۰ باقی مانده است Q همان یک میباشد. حال این یک ورودی NOR پایینی میشود بنابراین دو ورودی ۱ و ۰ داریم پس Q' نیز برابر صفر است.

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(بعد از R=0 , S=1)
0	1	1	0	
1	1	1	0	(بعد از R=1 , S=0)
0	0	1	1	→ تعریف نشده



یک گیت NAND اگر حداقل یک ورودی صفر داشته باشد خروجی آن یک است.

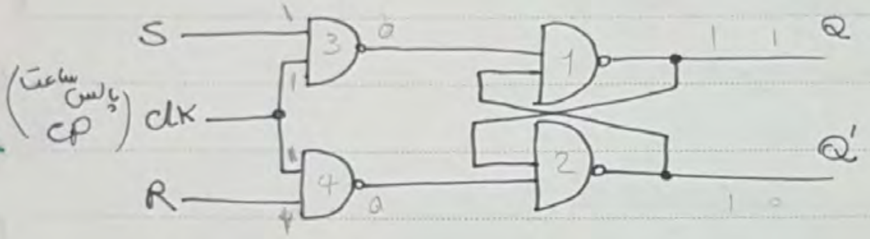
PAPCO

Subject:
Date:

در فلیپ فلاپ پایه با گیت NOR اگر S و R هر دو یک باشند هر دو خروجی Q و Q' مساوی صفر میشوند که این حالت تعریف نشده است و نباید استفاده شود.

حالا اگر هر دو ورودی به صفر برگردند خروجی فلیپ فلاپ ها نا مشخص است و بستگی به این دارد که کدام ورودی لحظه‌ای بیشتر در یک بماند.

فلیپ فلاپ R-S :



if $CP=0 \longrightarrow$ خروجی گیت های 3 و 4 برابر یک میشوند.

if $CP=1 \longrightarrow \begin{cases} S=1,\ R=0 \longrightarrow Q=1,\ Q'=0 \\ S=0,\ R=1 \longrightarrow Q=0,\ Q'=1 \\ S=1,\ R=1 \longrightarrow Q=Q'=1 \end{cases}$ چی میشود؟

$S=0,\ R=0 \longrightarrow$ تعریف نشده

در این حالت که S و R هر دو یک هستند و در زمانی که CP به صفر برمیگردد به شرط آنکه R و S هر دو یک باشند خروجی غیر قابل پیش بینی میباشد و وابسته به تاخیر گیت هاست. لذا این مدار کمتر مورد استفاده قرار میگیرد و برای درک ساختار فلیپ فلاپ هاست چون

Subject:
Date

$Q(t)$	S	R	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	نامعین
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	نامعین

(جدول مشخصه)

$Q = Q(t) =$ حالت f.f قبل از به کار بردن CP

$Q^* = Q(t+1)$ حالت بعدی f.f بعد از CP

S.R. f.f در مدارهای آسنکرون استفاده می‌شود و هر تغییری در ورودی فوراً خروجی را تغییر می‌دهد و به همین دلیل برای مدارهای سنکرون مناسب نیست.

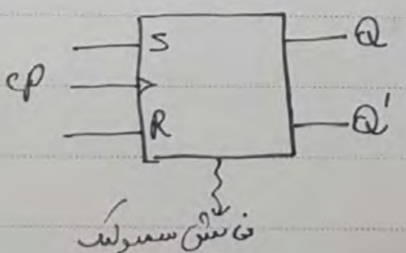
جدول بالا به این صورت تعبیر می‌شود که با دانستن ورودی‌های S و R و اعمال یک پالس ساعت در ورودی CP فلیپ‌فلاپ از حالت فعلی Q به حالت بعدی $Q(t+1)$ می‌رود.

$$Q(t+1) = f(Q(t), S, R)$$

Q \ SR	00	01	11	10
0			X	1
1	1		X	1

$Q(t+1) = S + QR'$ ← معادله مشخصه

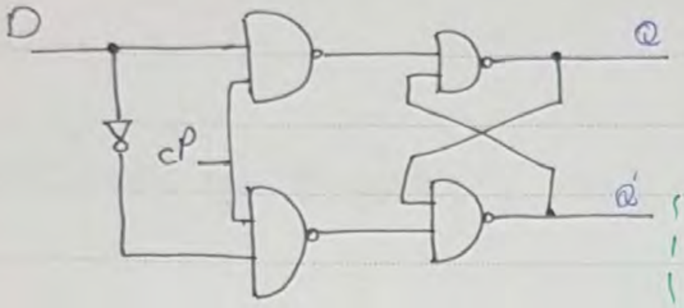
$SR = 0$ نامعین



نمایش سمبولیک

فلیپ فلاپ D : یک راه برای حذف حالت غیرقابل پیش بینی فلیپ فلاپ R-S این است که هرگز S و R با هم یک نشوند.

برای این کار از فلیپ فلاپ D استفاده می‌شود.



CP	D	Q	Q(t+1)
↑ یا 1	0	0	0
↑ یا 1	0	1	0
↑ یا 1	1	0	1
↑ یا 1	1	1	1

$CP=0 \longrightarrow$ خروجی گیت های 3 و 4 برابر صفر می‌شود

در نتیجه خروجی تغییر نمی‌کند.

$CP=1 \ , \ D=1 \longrightarrow Q=1$

$CP=1 \ , \ D=0 \longrightarrow Q=0$

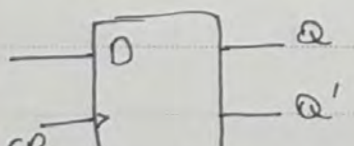
فرض اولیه رسم جدول با $CP=1$ بوده است بنابراین ستون CP را نمی‌کشیم.

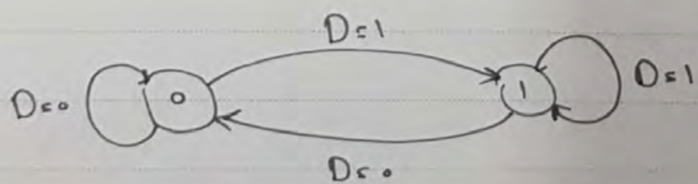
Q \ D	0	1
0		1
1		1

$\boxed{Q(t+1)=D}$

بنابراین در فلیپ فلاپ D زمانی که CP برابر یک باشد خروجی مساوی اطلاعات ورودی است و هنگامی که $CP=0$ باشد اطلاعات با قبلی که در خروجی Q فلیپ فلاپ است نگه داری میشود تا اینکه دوباره CP یک شود.

به فلیپ فلاپ D گاهی لچ D نیز گفته می‌شود.

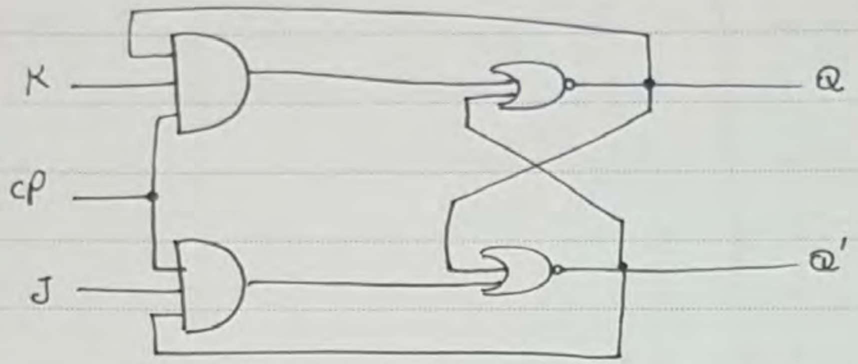




فلیپ فلاپ JK : فلیپ فلاپ JK نوع کامل‌تر شده‌ی فلیپ فلاپ RS میباشد. به این معنا که حالت غیرقابل پیش بینی فلیپ فلاپ RS در فلیپ فلاپ JK موجود و قابل تعریف است.

$S \rightarrow J$ (Jump)

$R \rightarrow K$ (Kill)



$J=1, K=0 \xrightarrow{CP=1} Q=1$

$J=0, K=1 \xrightarrow{CP=1} Q=0$

$J=1, K=1 \xrightarrow{CP=1}$ خروجی با پالس ساعت مکمل می‌شود

$J=0, K=0 \xrightarrow{CP=1}$ خروجی تغییر نمی‌کند

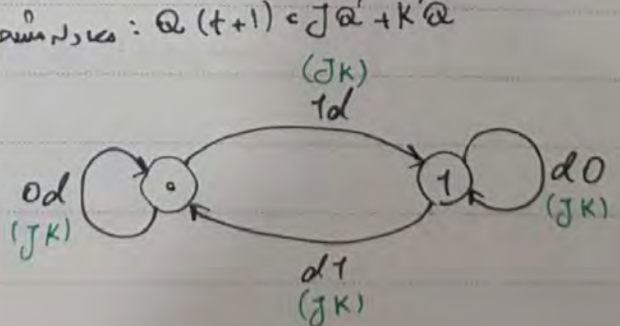
درحالی که J و K هردو یک باشند و $CP=1$ باشد به علت وجود feedback در ورودی گیت های AND خروجی دوباره مکمل میشود و این عمل مرتب تکرار می‌شود یعنی Q و Q' به طور مکرر صفر و یک میشوند تا زمانی که $CP=0$ شود.

برای جلوگیری از این عمل باید زمان تداوم پالس ساعت کوتاه‌تر از زمان تاخیر فلیپ فلاپ باشد.

این محدودیت در پالس ساعت را میتوان با استفاده از فلیپ فلاپ master-slave (MS) (تابع - متبوع) و یا راه اندازی لبه‌ای پالس برطرف نمود.

Q \ JK	00	01	11	10
0			1	1
1	1			1

معادله مشخصه: $Q(t+1) = JQ' + K'Q$



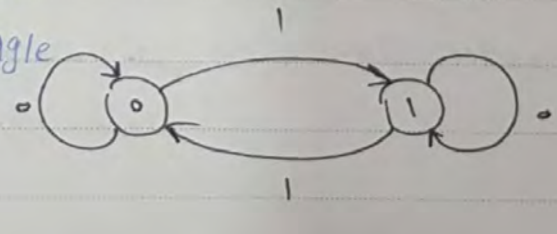
$Q(t)$	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

فلیپ‌فلاپ T : فلیپ‌فلاپ T نوع تک ورودی فلیپ فلاپ JK میباشد. بدین ترتیب که ورودی J و K در فلیپ فلاپ JK به هم متصل و آن را ورودی T مینامیم.

$Q(t)$	T	$Q(t+1)$
0	0	0 → cte
0	1	1 → Toggle
1	0	1 → cte
1	1	0 → Toggle

$T = 1 \xrightarrow{CP=1}$ حالت f.f عوض میشود

$T = 0 \xrightarrow{CP=1}$ خروجی تغییر نمی‌کند



$Q \backslash T$	0	1
0		1
1	1	

معادله مشخصه: $Q(t+1) = TQ' + T'Q = T \oplus Q$

جداول مشخصه فلیپ فلاپ :

JK f.f

J	K	Q(t+1)
0	0	Q(t) → cte
0	1	0 → Reset
1	0	1 → set
1	1	Q′(t) → Toggle

R.S f.f

S	R	Q(t+1)
0	0	Q(t) → cte
0	1	0 → Reset
1	0	1 → set
1	1	? → غیرقابل پیش بینی

D. f.f

D	Q(t+1)
0	0 → cte
1	1 → cte

T . f.f

T	Q(t+1)
0	Q(t) → cte
1	Q′(t) → Toggle

تحلیل مدارهای ترتیبی سنکرون :

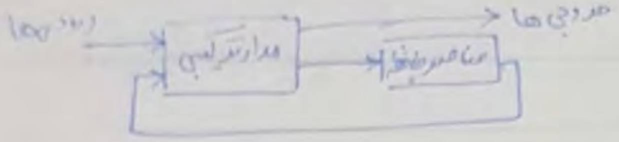
~~۱_ با استفاده از خروجی های فلیپ فلاپ ها روابط خروجی مدار را بدست می آوریم~~

۱_ استخراج تابع

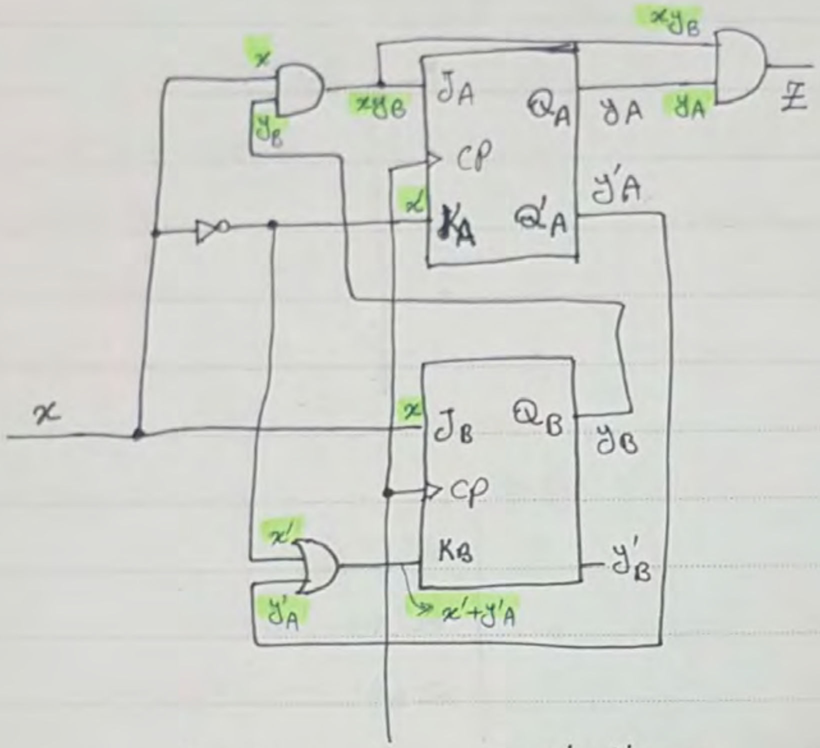
۱_ با استفاده از خروجی های فلیپ فلاپ ها روابط خروجی مدار را بدست می آوریم ، برای این منظور مقدار تابع bool ورودی هر فلیپ فلاپ و نیز خروجی ها را برحسب حالت فعلی فلیپ فلاپ ها و ورودی ها بدست می آوریم .

۲_ جدول حالتی رسم می کنیم که در آن حالت فعلی فلیپ فلاپ ها ، ورودی مدار ، ورودی فلیپ فلاپ ها ، حالت بعدی فلیپ فلاپ ها و خروجی مدار وجود داشته باشد .

۳_ رسم دیاگرام حالت



مثال) مدار زیر را تحلیل کنید.



$J_A = xy_B$

$K_A = x'$

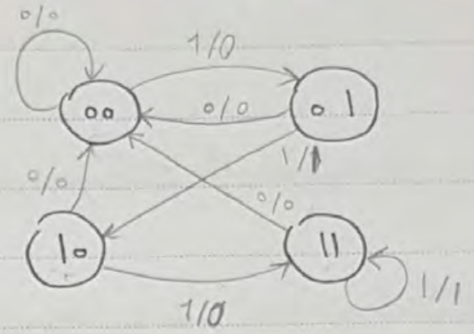
$J_B = x$

$K_B = x' + y'_A$

$Z = xy_Ay_B$

ورودی ها		ورودی مستقل	ورودی فلیپ فلاپ ها				خروجی ها (حالت بعدی)		خروجی های مستقل
$y_A(t)$	$y_B(t)$	x	J_A	K_A	J_B	K_B	$y_A(t+1)$	$y_B(t+1)$	$Z(t+1)$
0	0	0	0	1	0	1	0	0	0
0	0	1	0	0	1	1	0	1	0
0	1	0	0	1	0	1	0	0	0
0	1	1	1	0	1	1	1	0	0
1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	1	0	1	1	1
1	1	0	0	1	0	1	0	0	0
1	1	1	1	0	1	0	1	1	1

رسم مدار با فرض x/Z :



جداول تحریک فلیپ‌فلاپ‌ها:

جدول مشخصه برای تحلیل مدار ترتیبی مفید می‌باشد. برای طراحی مدار ترتیبی معمولاً تغییر از یک حالت به حالت دیگر را داریم و میخواهیم بدانیم چه ورودی باید برای فلیپ‌فلاپ قرار دهیم تا این تغییر انجام شود.

جدول تحریک ورودی‌های لازم برای تغییر حالت فلیپ‌فلاپ را مشخص می‌کند.

RS f.f

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

JK f.f

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

D f.f

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

T f.f

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

Subject:
Date

طراحی مدارهای ترتیبی :

طراحی یک مدار ترتیبی با پالس ساعت از یک مجموعه مشخصات شروع میشود و به توابع منطقی و دیاگرام منطقی خاتمه می یابد.

اگر در مدارهای ترکیبی به جدول درستی نیاز داشتیم، مدارهای ترتیبی نیاز به بدست آوردن جدول حالت دارند.

بنابراین برای طراحی مدار ترتیبی مراحل زیر را طی میکنیم :

1. از روی صورت مسئله جدول حالت را رسم می کنیم یا معادل آن دیاگرام حالت را رسم می کنیم.

2. با استفاده از روش های کاهش حالات جدول حالات را ساده می کنیم. (نیازی برای انجام توسط ما نیست 😊)

3. عملیات تخصیص حالت را انجام داده و اگر مرحله دو را انجام دادیم جدول حالت را دوباره تشکیل میدهیم.
(عملاً این مرحله هم انجام نمیشه. ?!)

4. پس از رسم جدول حالت باید تعداد فلیپ فلاپ های مورد نیاز و نوع آنها مشخص گردد که این کار با استفاده از تعداد حالات مورد نیاز مشخص میگردد.

5. جدول تحریک را تشکیل داده و ورودی های فلیپ فلاپ ها را برای هر تغییر حالت مشخص میکنیم.

6. پس از کامل شدن ستون مربوط به هرکدام از ورودی های فلیپ فلاپ ها با روش نقشه کارنو یا روش های ساده سازی دیگر توابع خروجی مدار و توابع ورودی فلیپ فلاپ ها مشخص میگردد.

7. رسم شماتیک مدار

Subject:
Date

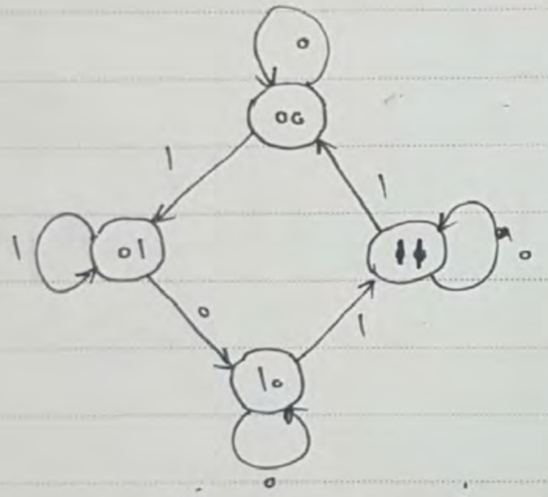
کاربرد فلیپ فلاپ ها:

فلیپ فلاپ D برای کاربرد های انتقال اطلاعات مانند شیفت دهنده ها استفاده می شود.

فلیپ فلاپ T در کاربرد هایی که نیاز به مکمل کردن دارند استفاده می شوند مانند شمارنده ها

فلیپ فلاپ های JK و RS دارای کاربرد های عمومی هستند.

مثال) مدار ترتیبی با پالس ساعت طراحی کنید که با کدام حالت آن مطابق شکل زیر باشد.
(از فلیپ فلاپ JK استفاده شود)



x: ورودی مستقل

AB : حالت فلیپ فلاپ ها

حالت فعلی		ورودی ها	حالت بعدی		ورودی های فلیپ فلاپ ها			
$A(t)$	$B(t)$	x	$A(t+1)$	$B(t+1)$	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	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
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

PAPCO

Scanned with CamScanner

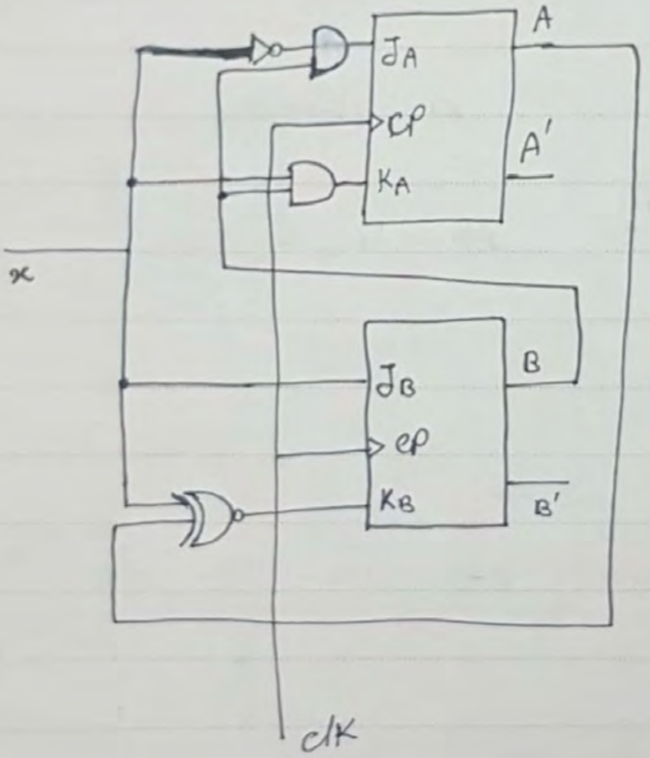
$J_A = f(A(t), B(t), x) = ?$

$x \backslash AB$	00	01	11	10
0		1	X	X
1			X	X

به همین ترتیب داریم:

$$\begin{cases} K_A = xB \\ J_B = x \\ K_B = A \odot x \end{cases}$$

$J_A = x'B$



Subject:
Date

مثال) با فلیپ‌فلاپ JK شمارنده‌ای طراحی کنید که به صورت زیر عمل کند.

0 → 3 → 7 → 6 → 4 → 2 → 1 → 5 → 0

000 → 011 → 111 → 110 → 100 → 010 → 001 → 101 → 000

$A_3(t)$	$A_2(t)$	$A_1(t)$	$A_3(t+1)$	$A_2(t+1)$	$A_1(t+1)$	J_{A_3}	K_{A_3}	J_{A_2}	K_{A_2}	J_{A_1}	K_{A_1}
حالت فعلی			حالت بعدی								
0	0	0	0	1	1	0	X	1	X	1	X
0	1	1	1	1	1	1	X	X	0	X	0
1	1	1	1	1	0	X	0	X	0	1	X
1	1	0	1	0	0	X	0	1	X	0	X
1	0	0	0	1	0	X	1	X	1	0	X
0	1	0	0	0	1	0	X	1	X	X	1
0	0	1	1	0	1	1	X	0	X	X	0
1	0	1	0	0	0	X	1	0	X	1	X

$J_{A_3} = f(A_3(t), A_2(t), A_1(t)) = ?$

$A_3 \backslash A_2A_1$	00	01	11	10
0		X	X	
1	1	1	X	X

$J_{A_3} = A_3$

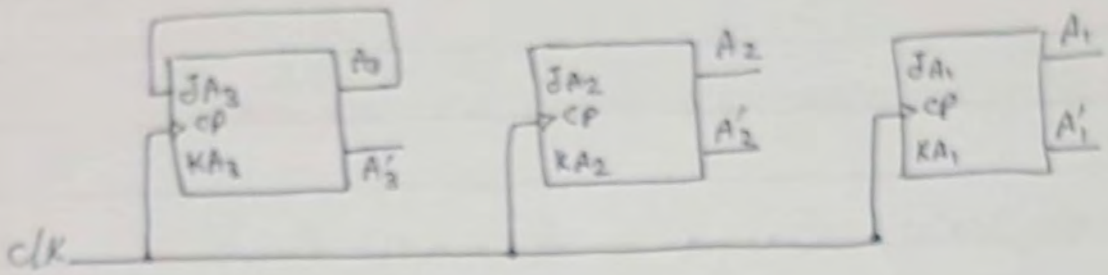
$K_{A_3} = ?$

$J_{A_2} = ?$

$K_{A_2} = ?$

$J_{A_1} = ?$

$K_{A_1} = ?$



تكمل مهمة ..

Subject:
Date

در مثال قبل اگر با T flip-flop پیاده‌سازی می‌کردیم:

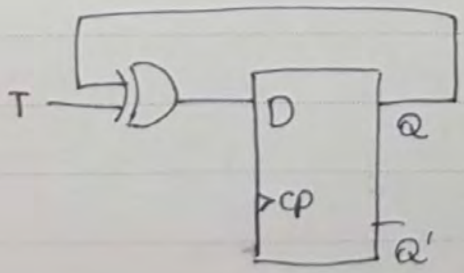
حالات فعلی								
$A_3(t)$	$A_2(t)$	$A_1(t)$	$A_3(t+1)$	$A_2(t+1)$	$A_1(t+1)$	T_{A_3}	T_{A_2}	T_{A_1}
0	0	0	0	1	1	0	1	1
0	1	1	1	1	1	1	0	0
1	1	1	1	1	0	0	0	1
1	1	0	1	0	0	0	1	0
1	0	0	0	1	0	1	1	0
0	1	0	0	0	1	0	1	1
0	0	1	1	0	1	1	0	0
1	0	1	0	0	0	1	0	1

مثال) با استفاده از فلیپ فلاپ D فلیپ فلاپ T بسازید.

$$D\text{-f.f}: \; Q(t+1) = D$$

$$T\text{-f.f}: \; Q(t+1) = T \oplus Q(t)$$

$$D = T \oplus Q(t)$$

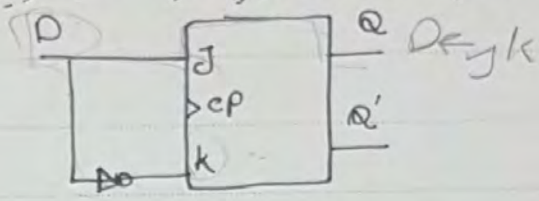


Subject:
Date

A مثال) با استفاده از فلیپ فلاپ T فلیپ فلاپ D بسازید.

B مثال) با استفاده از فلیپ فلاپ D فلیپ فلاپ JK بسازید.

با استفاده از فلیپ فلاپ JK فلیپ فلاپ D بسازید.



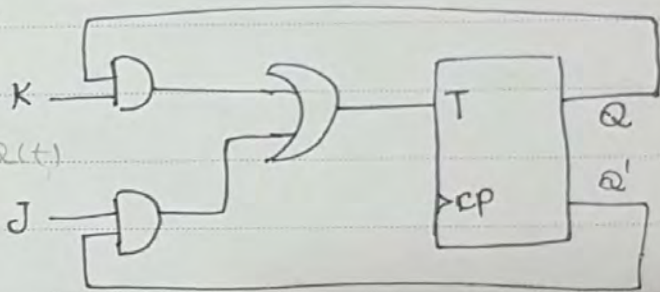
با استفاده از فلیپ فلاپ T فلیپ فلاپ JK بسازید.

$T\text{-}F.F: Q(t+1) = T \oplus Q(t) = TQ' + T'Q$

$J.K\text{-}FF: Q(t+1) = JQ' + K'Q$

$T \oplus Q(t) = JQ' + K'Q \rightarrow T = (JQ' + K'Q) \oplus Q(t)$

$\rightarrow T = KQ + JQ'$



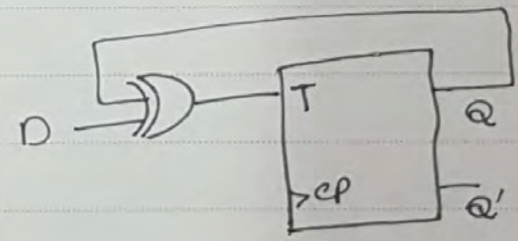
$D \leftarrow T$

ج مثال A)

$D\text{-}F.F: Q(t+1) = D$

$T\text{-}F.F: Q(t+1) = T \oplus Q(t)$

$D = T \oplus Q(t) \longrightarrow T = Q(t) \oplus D$



$JK \leftarrow D$

ج مثال B)

$D\text{-}F.F: Q(t+1) = D$

$J.K\text{-}F.F: Q(t+1) = JQ' + K'Q$

$JQ' + K'Q = D$