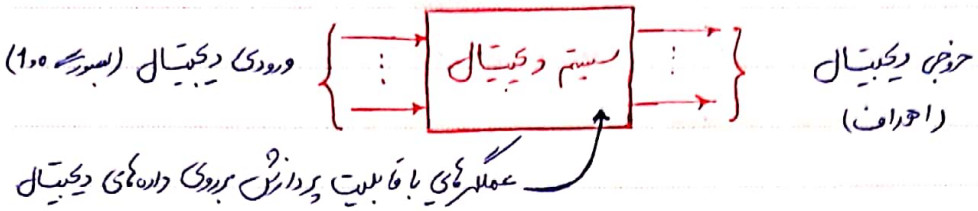


فهرست مطالب :

- سیستم‌های عددی (سیستم باینری Binary)
- دروازه‌های منطقی
- ساده سازی توابع
- طراحی سیستم‌های مدارهای ترکیبی ساده
- طراحی سیستم‌های مدارهای ترکیبی پیچیده‌تر
- طراحی سیستم‌های مدارهای ترتیبی سنکرون
- طراحی سیستم‌های مدارهای ترتیبی آسنکرون
- ساختار داخلی دروازه‌های منطقی

Subject:
Year. 97 Month. 10 Date. 14 ( )

« سیستم های عددی »

سیستم دهدهی (اعشاری) (Decimal):

مبنا (پایه): $r=10$ ، ارقام: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

$(385)_{10} = 3\times10^2 + 8\times10^1 + 5\times10^0$

مانند → ارزش‌ها → $10^2\ 10^1\ 10^0$

سیستم دودویی (باینری) (Binary):

مبنا (پایه): $r=2$ ، ارقام: 0, 1

$(100101)_2 = 1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 = (37)_{10}$

مانند → $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

سیستم مبنای هشت (octal):

مبنا (پایه): $r=8$ ، ارقام: 0, 1, 2, 3, 4, 5, 6, 7

$(173)_8 = 1\times8^2 + 7\times8^1 + 3\times8^0 = (123)_{10}$

مانند → $8^2\ 8^1\ 8^0$

سیستم مبنای شانزده (Hexa Decimal):

مبنا (پایه): $r=16$ ، ارقام: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

10 11 12 13 14 15

$(15EF)_{16} = 1\times16^3 + 5\times16^2 + 14\times16^1 + 15\times16^0 = (5615)_{10}$

مانند → $16^3\ 16^2\ 16^1\ 16^0$

نکته:

اگر اعداد بصورت اعشاری یا کسری بود مانند سیستم دهدهی عمل کرده و ارزش های بعد از ممیز توان منفی دارند.

$(385/91)_{10} = 3\times10^2 + 8\times10^1 + 5\times10^0 + 9\times10^{-1} + 1\times10^{-2}$

مانند → $10^2\ 10^1\ 10^0\ 10^{-1}\ 10^{-2}$

$(100101.101)_2 = 1\times2^5 + 0\times2^4 + 0\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

$= (37/625)_{10}$

مانند → $2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}$

$(173.25)_8 = 1\times8^2 + 7\times8^1 + 3\times8^0 + 2\times8^{-1} + 5\times8^{-2} = (123.\cdots)_{10}$

مانند → $8^2\ 8^1\ 8^0\ 8^{-1}\ 8^{-2}$

$(15EF.10A)_{16} = 1\times16^3 + 5\times16^2 + 14\times16^1 + 15\times16^0 + 1\times16^{-1} + 0\times16^{-2} + 10\times16^{-3}$

مانند → $16^3\ 16^2\ 16^1\ 16^0\ 16^{-1}\ 16^{-2}\ 16^{-3}$

تبدیل سیستم‌های عددی به یکدیگر:

1) تبدیل مبنای r به مبنای 10

وزن (ارزش): $r^n \cdots r^1\ r^0\ r^{-1}\ r^{-2} \cdots r^m$

$$(a_n \cdots a_1 a_0 \cdot a_{-1} a_{-2} \cdots a_{-m}) = \overbrace{(a_n \times r^n + \cdots + a_1 \times r^1 + a_0 \times r^0)}^{\text{بخش صحیح } (x)} + \overbrace{(a_{-1} \times r^{-1} + a_{-2} \times r^{-2} + \cdots + a_{-m} \times r^{-m})}^{\text{بخش کسری } (y)}$$

$$= (x.y)_{10}$$

2) تبدیل از مبنای 10 به مبنای r

برای بدست آوردن رقم‌ها در مبنای r، بخش صحیح را بدون تقسیمات متوالی شرکت می‌دهیم. در هر مرحله تقسیم بر مبنای r کرده، باقیمانده حاصل از تقسیم یکی از رقم‌های مورد نظر خواهد بود و توقف عملیات زمانی خواهد بود که خارج قسمت صفر شود.

محاسبه قسمت صحیح ← روش تقسیمات متوالی:

$$\frac{X}{r} = (\underbrace{a_n \times r^{n-1} + \cdots + a_1 \times r^0}_{\text{صحیح } (X_1)} + a_0 \times r^{-1}) = X_1 + \frac{a_0}{r}$$

$$\frac{X_1}{r} = (\underbrace{a_n \times r^{n-2} + \cdots + a_2 \times r^0}_{\text{صحیح } (X_2)} + a_1 \times r^{-1}) = X_2 + \frac{a_1}{r}$$

$$X \mid r \quad \overline{a_0}$$
$$X_1 \mid r \quad \overline{a_1}$$
$$\cdots$$
$$\mid r \quad \overline{a_n}$$

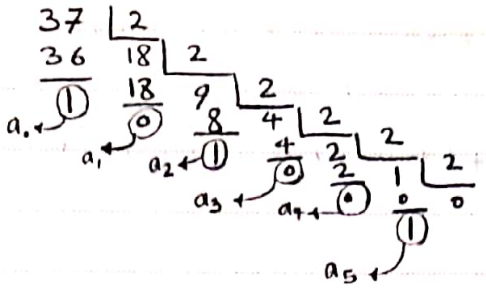
محاسبه قسمت اعشاری ← روش ضرب‌های متوالی:

$$y \times r = (\underbrace{a_{-1} \times r^0}_{\text{صحیح}} + \underbrace{a_{-2} \times r^{-1} + \cdots + a_{-m} \times r^{-m+1}}_{y_1 \text{ کسری}}) = a_{-1} + r \times y_1$$

$$y_1 \times r = (\underbrace{a_{-2} \times r^0}_{\text{صحیح}} + \underbrace{a_{-3} \times r^{-1} + \cdots + a_{-m} \times r^{-m+2}}_{y_2 \text{ کسری}}) = a_{-2} + r \times y_2$$

در روش ضرب‌های متوالی، بخش کسری را در هر مرحله در پایه r ضرب نموده، بخش صحیح حاصلضرب، رقم مورد نظر را به ما می‌دهد. انجام عملیات زمانی تمام می‌شود که قسمت کسری صفر شود و همچنین براساس میزان دقت مورد نظر بعد از نقطه ممیز است.

مثال 1 : عدد $(37.625)_{10}$ را در مبنای 2 به دست آورید.



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

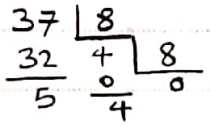
$0.625 \times 2 = 1.25 \Rightarrow a_{-1} = 1$

$0.25 \times 2 = 0.50 \Rightarrow a_{-2} = 0$ $\Rightarrow (0.625)_{10} = (0.101)_2$

$0.5 \times 2 = 1.0 \Rightarrow a_{-3} = 1$

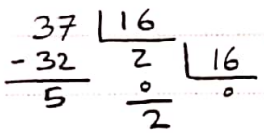
$\Rightarrow (37.625)_{10} = (100101.101)_2$

ب) عدد $(37.625)_{10}$ را در مبنای 8 و 16 محاسبه کنید.



$(37)_{10} = (45)_8$, $0.625 \times 8 = 5.0$

$\Rightarrow (37.625)_{10} = (45.5)_8$



$(37)_{10} = (25)_{16}$, $0.625 \times 16 = 10.0$

$\Rightarrow (37.625)_{10} = (25.A)_{16}$

مثال 2 : عدد $(101011.111010)_2$ را در مبنای 10 به دست آورید.

$(101011.111010)_2$
$2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}\ 2^{-3}\ 2^{-4}\ 2^{-5}\ 2^{-6}$

$= (1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0)$
$+ (1\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 1\times2^{-5} + 0\times2^{-6})$

$\Rightarrow (101011.111010)_2 = (43.\cdots)_{10}$

ب) معادل عدد در مبنای 8 را بیابید.

$(1\times2^5 + 0\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0) + (1\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 0\times2^{-4} + 1\times2^{-5} + 0\times2^{-6})$

$= \left[(1\times2^2 + 0\times2^1 + 1\times2^0)\times2^3 + (0\times2^2 + 1\times2^1 + 1\times2^0)\right] + \left[(1\times2^2 + 1\times2^1 + 1\times2^0)\times2^{-3} + (0\times2^2 + 1\times2^1 + 0\times2^0)\times(2^{-3})^2\right]$

$= \left[5\times8^1 + 3\times8^0\right] + \left[7\times8^{-1} + 2\times8^{-2}\right]$

$\Rightarrow (101011.111010)_2 = (53.72)_8$

Subject:
Year. 97 Month. 10 Date. 16 ( )

درروش دیگر برای تبدیل از مبنای 2 به مبنای 8، برای هر سه رقم در مبنای 2 معادل باینری یک رقمی آن را قرار می‌دهیم.
بطوری که ←

$$(x\ y\ z)$$
$$\uparrow 2^2\ \uparrow 2^1\ \uparrow 2^0$$

$$(\overbrace{101}\overbrace{011}.\overbrace{111}\overbrace{010})_2 = (53.72)_8$$

ج) عدد $(612.37)_8$ را به مبنای 2 تبدیل کنید.

برای تبدیل از مبنای 8 به مبنای 2، برای هر رقم در مبنای 8 معادل باینری سه رقمی آن را قرار می‌دهیم.

$$(6\ 1\ 2\ .\ 3\ 7)_8 = (110001010.011111)_2$$

110 ↓ 001 ; 010 ↓ 011 ; 111

د) عدد $(101011.111010)_2$ را در مبنای 16 بیابید.

$$(1\times2^5+0\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0)+(1\times2^{-1}+1\times2^{-2}+1\times2^{-3}+0\times2^{-4}+1\times2^{-5}+0\times2^{-6})$$
$$=\left[(0\times2^3+0\times2^2+1\times2^1+0\times2^0)\times16+(1\times2^3+0\times2^2+1\times2^1+1\times2^0)\times1\right]$$
$$+\left[(1\times2^3+1\times2^2+1\times2^1+0\times2^0)\times2^{-4}+(1\times2^3+0\times2^2+0\times2^1+0\times2^0)\times(2^{-4})^2\right]$$

$$=\left[2\times16^1+11\times16^0\right]+\left[14\times16^{-1}+8\times16^{-2}\right]$$
$$\Rightarrow (101011.111010)_2 = (2B.E8)_{16}$$

درروش دیگر برای تبدیل از مبنای 2 به مبنای 16، برای هر چهار رقم در مبنای 2 معادل باینری یک رقمی آن قرار می‌دهیم و از نقطه میبنیم به اندازه چهاربیت، چهاربیت به سمت چپ و راست می‌رویم. بطوری که:

$$(w\ x\ y\ z)$$
$$\uparrow 2^3\ \uparrow 2^2\ \uparrow 2^1\ \uparrow 2^0$$

$$(\overbrace{0010}\overbrace{1011}.\overbrace{1110}\overbrace{1000})_2 = (2B.E8)_{16}$$

هـ) عدد $(110001010.011111)_2$ را در مبنای 16 بیابید.

$$\overbrace{(0001}^{1}\overbrace{1000}^{8}\overbrace{1010}^{10}.\overbrace{0111}^{7}\overbrace{1100)}^{12}_2 = (18A.7C)_{16}$$

Subject:
Year. 97 Month. 10 Date. 16 ( )

و عدد $(9D1.3F)_{16}$ در مبنای 2 بیابید.

$= (100111010001.00111111)_2$

$(9\ D\ 1\ .\ 3\ F)_{16}$

1001 ← 9 ، D → 1101 ، 1 → 0001 ، 3 → 0011 ، F → 1111

اعمال محاسباتی بر روی سیستم های باینری :

روند کلی عملیات محاسباتی مشابه روند سیستم های دهدهی می باشد.

جمع محاسباتی ← برای مثال در مبنای 2، چون تنها رقم 0 و 1 داریم یکی از چهار حالت زیر رخ می دهد.

$0+0=0$ ، $0+1=1$ ، $1+0=1$ ، $1+1=10$

① ① ← رقم نقلی

مانند →
$$\begin{array}{r} 1011010 \\ +\ 1011100 \\ \hline 10110110 \end{array}$$

رقم نقلی نهایی ← (carry Bit)

(* رقم نقلی نهایی رقمی است که اگر در جمع 7 بیت داشته باشیم و جواب 8 بیت شد، رقم اول از سمت چپ نقلی نهایی است)

تفریق ← برای مثال در مبنای 2، چون تنها رقم 0 و 1 داریم یکی از چهار حالت زیر رخ می دهد.

$0-0=0$ ، ① $0-1=1$ ، $1-0=1$ ، $1-1=0$

رقم قرضی ← ①

رقم قرضی ← ①

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

مانند →

ضرب ← همانند جمع های متوالی است. برای مثال در مبنای 2 چون رقم 0 و 1 داریم یکی از چهار حالت زیر رخ می دهد.

$0\times0=0$ ، $0\times1=0$ ، $1\times0=0$ ، $1\times1=1$

مانند →
$$\begin{array}{r} 1001 \\ \times\ 101 \\ \hline 1001 \\ 0000 \\ 1001 \\ \hline 101101 \end{array}$$

تقسیم ← همانند تفریق های متوالی است.

مانند →
$$\begin{array}{r|l} 1001 & 10 \\ -\ 10 & 1 \\ \hline 0111 & \end{array}$$

مکمل (متمم) یک عدد در مبنای r : (complement)

مکمل r
مکمل r-1

مکمل r ← اگر N یک عدد در مبنای r با تعداد n رقم صحیح و m رقم کسری باشد.

$(N)_r$ عدد r مکمل = r's comp. $= \begin{cases} r^n - N & N \neq 0 \\ 0 & N = 0 \end{cases}$

مثال 3 : مکمل r اعداد $(19{,}625)_{10}$ و $(1101.1010)_2$ را بیابید.

$(19{,}625)_{10}$ عدد 10 مکمل $= 10^2 - (19{,}625)_{10} = (80.375)_{10}$

$(1101.1010)_2$ عدد 2 مکمل $= 2^9 - (1101.1010)_2 = (0010.0110)_2$

در روش دیگر برای نوشتن مکمل 2 عدد $(N)_2$ می‌توانیم بصورت زیر عمل کنیم.
از کم ارزش ترین رقم N (از سمت راست) تا به اولین رقم 1 که می‌رسیم همان ارقام را نوشته پس از آن به بعد ارقام 1 به 0 و ارقام 0 به 1 تبدیل می‌شود.

مکمل r-1 ← اگر N یک عدد در مبنای r با تعداد n رقم صحیح و m رقم کسری باشد.

مکمل (r-1) عدد $(N)_r$ $= (r-1)$'s comp. $= r^n - r^{-m} - (N)_r$

مثال 4 : مکمل r-1 اعداد $(19{,}625)_{10}$ و $(1101.1010)_2$ را بیابید.

$(19{,}625)_{10}$ عدد 9 مکمل $= 10^2 - 10^{-3} - (19.625)_{10} = (80.374)_{10}$

$(1101.1010)_2$ عدد 1 مکمل $= 2^4 - 2^{-4} - (1101.1010)_2 = (0010.0101)_2$

در روش دیگر برای نوشتن مکمل 1 عدد (N_2) می‌توانیم بصورت زیر عمل کنیم.
در رقم های عدد N ، رقم های 1 به 0 و رقم های 0 به 1 تبدیل می‌شود.

انجام عمل تفریق توسط بکارگیری مکمل 2 و عمل جمع:

محاسبه با بکارگیری مکمل 2 ← (M-N=?)

1/ مکمل 2 عدد N را حساب می‌کنیم.

2/ جمع M با مکمل 2 عدد N.

3/ اگر حاصل جمع دارای بیت نقلی نهایی شد، با حذف نمودن بیت نقلی، حاصل تفریق مثبت خواهد آمد.
اگر حاصل جمع دارای بیت نقلی نهایی نشد، مکمل 2 حاصل جمع با علامت منفی برابر حاصل تفریق خواهد بود.

1101001 → مکمل 2 عدد N → 1101001
- 1011101 → مانند → + 0100011
X0001100 → 0001100

1001101 → مکمل 2 عدد N → 1001101
- 1100010 → مانند → + 0011110
1101011 → منفی مکمل 2 → -0010101

محاسبه با بکارگیری مکمل 1 ← (M-N=?)

1/ مکمل 1 عدد N را محاسبه می‌کنیم.

2/ جمع M با مکمل 1 عدد N.

3/ اگر حاصل جمع دارای بیت نقلی نهایی شد، با حذف نمودن بیت نقلی، حاصل تفریق برابر حاصل جمع مرحله قبل بعلاوه 1 می‌باشد.
اگر حاصل جمع دارای بیت نقلی نهایی نشد، مکمل 1 حاصل جمع با علامت منفی برابر حاصل تفریق خواهد بود.

1101001 → مکمل 1 عدد N → 1101001
- 1011101 → مانند → + 0100010
X0001011
+ 1
0001100

1001101 → مکمل 1 عدد N → 1001101
- 1100010 → مانند → + 0011101
1101010 → منفی مکمل 1 → -0010101

Subject:
Year. 97 Month. 11 Date. 23 ( )

نمایش اعداد با علامت:

سیستم هفت بیتی:

- 0 برای عدد مثبت
- 1 برای عدد منفی

1. روش مقدار-علامت ← مقدار عدد مورد نظر همراه با بیت علامت
2. روش مکمل یک-علامت ←
   - عدد مثبت ← عدد به همراه بیت علامت (0)
   - عدد منفی ← مکمل عدد به همراه بیت علامت
3. روش مکمل دو-علامت ←
   - عدد مثبت ← عدد به همراه بیت علامت (0)
   - عدد منفی ← مکمل دو عدد به همراه بیت علامت

(روش 3 پرکاربردتر است)

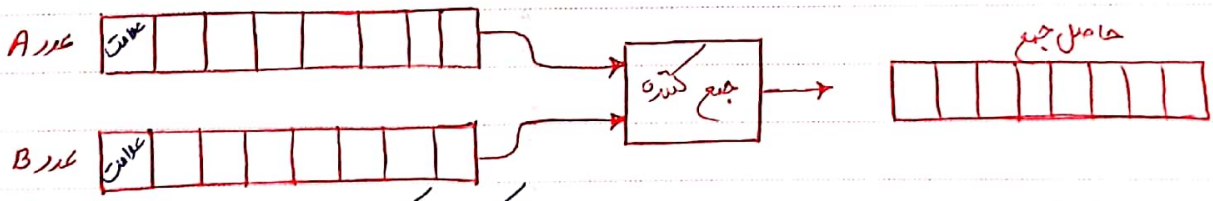
مثال 5: عدد +9 و -9 به صورت سیستم هفت بیتی با سه روش بالا نشان دهید.

روش مقدار-علامت:
- +9 ← 0,0001001
- -9 ← 1,0001001
- ← بیت علامت

روش مکمل یک-علامت:
- +9 ← 0,0001001
- -9 ← 1,1110110
- ← بیت علامت

روش مکمل دو-علامت:
- +9 ← 0,0001001
- -9 ← 1,1110111
- ← بیت علامت

انجام عمل محاسباتی در سیستم با روش مکمل دو:



مثال 6: عمل محاسباتی C بر روی دو عدد ±9 و ±14 انجام دهید. (فقط 7 بیت جمع کننده احتیاج داریم)

$$\begin{array}{r} +9 \\ +14 \\ \hline +23 \end{array} \rightarrow \begin{array}{r} 0\ 0001001 \\ 0\ 0001110 \\ \hline 0\ 0010111 \end{array}$$
← بیت علامت

$$\begin{array}{r} -9 \\ +14 \\ \hline +5 \end{array} \rightarrow \begin{array}{r} 1\ 1110111 \\ 0\ 0001110 \\ \hline (1)\ 00000101 \end{array}$$
(1) ← بیت نقلی، بیت علامت

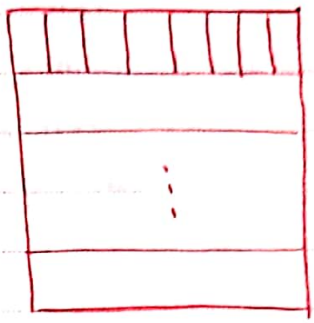
$$\begin{array}{r} +9 \\ -14 \\ \hline -5 \end{array} \rightarrow \begin{array}{r} 0\ 0001001 \\ 1111\ 0010 \\ \hline 11111011 \end{array}$$
← بیت علامت، مکمل دو عدد

$$\begin{array}{r} -9 \\ -14 \\ \hline -23 \end{array} \rightarrow \begin{array}{r} 1\ 1110\ 111 \\ 1111\ 0010 \\ \hline (1)11\ 101001 \end{array}$$
(1) ← بیت نقلی، بیت علامت، مکمل دو، عدد

حافظه (Memory) و ثبات (Register):

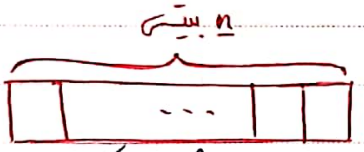
هردو واحد فوق، واحدهایی برای نگهداری اطلاعات باینری (ترکیبات 0 و 1) در سیستم‌های دیجیتال می‌باشند.
حافظه برای واحدی که نگهداری بلندمدت اطلاعات را به عهده دارد.
و ثبات برای واحدی که نگهداری کوتاه‌مدت اطلاعات را به عهده دارد.
واضح است که در سیستم‌های دیجیتال برای نگهداری هر بیت 0 یا 1 یک خانه لازم می‌باشد. بنابراین مثلاً در یک سیستم هشت بیتی، ثبات‌ها بصورت 8 خانه در کنار یکدیگر بیان می‌شوند.

word ← word ← در یک سیستم هشت بیتی، شامل 8 بیت داده‌های باینری 0 و 1 می‌باشد.



کدهای باینری:

هر ترکیب از 0 و 1 که بصورت منحصر بفرد بیانگر یک کاراکتر خاص باشد را یک کد باینری می‌نامند.



⟵ 2^n ترکیب باینری منحصر بفرد

واضح است که اگر بخواهیم m کاراکتر خاص را کدگذاری کنیم به کدهای n بیتی نیازمی‌باشیم به نحوی که $m \leq 2^n$
برای سیستم‌های دیجیتال می‌توان کدهای مختلفی را تعریف نمود، ولی باید جدول کد برای کارکردن با سیستم معلوم باشد.

جدول کد:

کاراکتر	کلمه کد
A	ـ ـ ـ
B	ـ ـ ـ ـ
⋮	
*	ـ ـ
(	ـ ـ
⋮	
0	ـ ـ
1	ـ ـ ـ
⋮	

کدهای عددی:

فرض می‌کنیم که سیستم دیجیتال مورد نظر ما با اعداد سروکار دارد. بنابراین لازم است برای وارد نمودن این اعداد به داخل آن از کدهایی برای نمایش رقم‌های دهگانه استفاده نماییم.

$m=10 \Rightarrow m=10 < 2^4 \Rightarrow n=4$

1/ کد BCD (Binary coded Decimal) ← یک کد چهار بیتی که ارزش معادل باینری ارقام مورد نظر در بیت می آید. (معمولاً از این کد استفاده می کنیم)

2/ کد Ex-3 (بعلاوه 3) ← Ex-3 = BCD + 0011

3/ کد 8 4 -2 -1 ← وزن هر بیت بصورت 1-، 2-، 4، 8 است.

4/ کد 2 4 2 1 ← وزن هر بیت بصورت 1، 2، 4، 2 است.

کد دسیمال مورد نظر (رقم ها)	BCD	Ex-3	8 4 -2 -1	2 4 2 1
0	0000	0011	0000	0000
1	0001	0100	0111	0001
2	0010	0101	0110	0010
3	0011	0110	0101	0011
4	0100	0111	0100	0100
5	0101	1000	1011	0101
6	0110	1001	1010	
7	0111	1010	1001	
8	1000	1011	1000	
9	1001	1100	1111	

مثال 1: عدد 695 را بصورت سیستم BCD و Ex-3 نشان دهید.

$$695 = (0110\ 1001\ 0101)_{BCD} \quad , \quad 695 = (1001\ 1100\ 1000)_{Ex-3}$$

کدگری:

ترکیبات باینری منحصر بفردی که بصورت متوالی بیان می شوند و دارای این ویژگی خاص می باشند که هر کلمه ی کد با کلمه ی کد قبلی و بعدی خود فقط در یک بیت متفاوت است.

کدگذاری دوتایی: 00
01
11
10

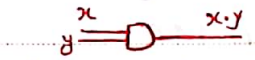
منطق باینری (Binary Login):

منظور از منطق باینری بیان یک روش ریاضی برای توصیف و پردازش داده‌های دوتایی می‌باشد.
در منطق باینری ۳ نوع اپراتور تانیکی عملیات برای انجام پردازش‌های مورد نظر تعریف گردد که از جمله این عملیات سه عملگر پایه‌ای زیر می‌باشند:

AND	"و" منطقی
OR	"یا" منطقی
NOT	"نقی" منطقی

با در نظر گرفتن متغیرهای باینری x و y، برای هرکدام از عملگرهای روبرو، جدول عملکردی برای توصیف عملگرها بیان می‌گردد.

عملگر AND ← نماد ریاضی آن بصورت $x.y$ است.

نماد مداری آن بصورت سرویرویت ← 

اگر حداقل یک از متغیرهای وارد شونده صفر باشد، خروجی صفر می‌شود و زمانی 1 می‌گردد که تمامی ورودی‌ها (متغیرهای وارد شونده)، 1 باشد.

جدول عملگر →

x	y	x.y (حاصل AND)
0	0	0
0	1	0
1	0	0
1	1	1

عملگر OR ← نماد ریاضی آن بصورت $x+y$ است.

نماد مداری آن بصورت روبرویت ← 

اگر حداقل یک از متغیرهای وارد شونده به آن 1 باشد، حاصل آن 1 می‌شود و وقتی صفر می‌شود که تمامی متغیرها صفر باشند.

جدول عملگر →

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

عملگر NOT ← نماد ریاضی آن بصورت $\bar{x}=x'$ است.

نماد مداری آن بصورت روبرویت ← 

جدول عملگر →

x	$\bar{x}=x'$
0	1
1	0

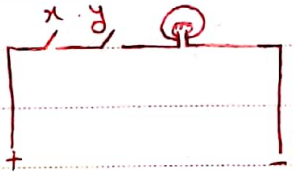
نکته:
۳ نوع بذکر است دو عملگر معرفی شده • (AND) و + (OR)، دو عملگر منطقی می‌باشند و با عملیات ریاضی ضرب (•) و جمع (+) متفاوت می‌باشند.

مدارات کلیدی برای توصیف عملگرهای باینری:

کلید { باز: 0 ، بسته: 1 }

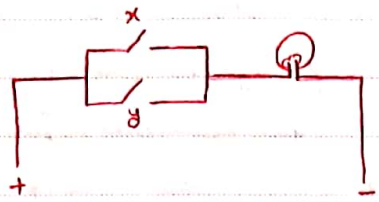
لامپ { خاموش: 0 ، روشن: 1 }

توصیف AND ←

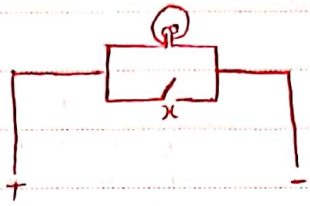


Subject:
Year. 97 Month. 11 Date. 28 ( )

توصیف OR

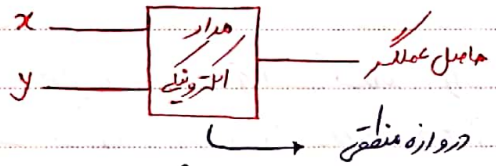


توصیف NOT



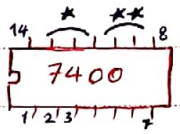
دروازه های منطقی (Logic Gates):

منظور از یک دروازه منطقی، مداری است که یک عملگر را درعمل اجرا می کند.

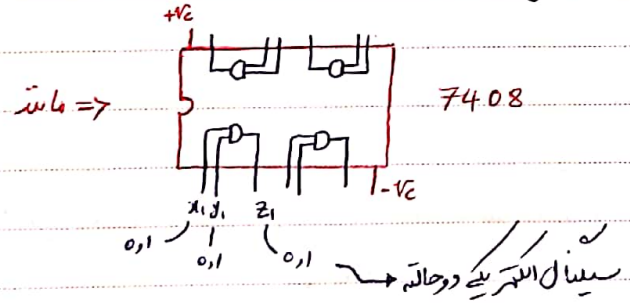


دروازه ها معمولاً در بسته های چندتایی بصورت یک تراشه (مدار مجتمع) (IC) طراحی و تولید می شوند.

Integrated circuit

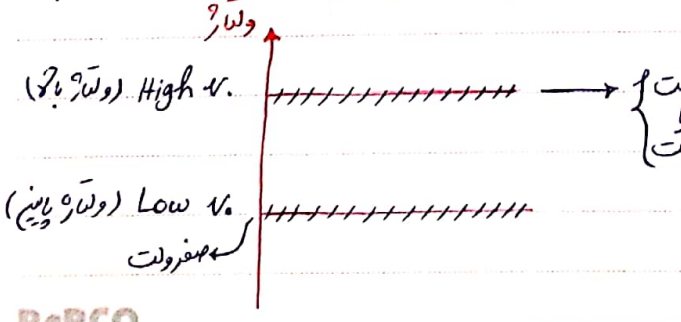


* ← چند رقم اول بیانگر تکنولوژی ساخت دروازه است. مانند TTL ، MOS ، CMOS
* ← چند رقم بعدی بیانگر عملگرها یا مدارهای عمل کننده داخلی است.



سیگنال های باینری:

واضح است که برای کارعملی با یک دروازه منطقی و وارد نمودن داده های باینری باید از یکسری سیگنال های الکتریکی مناسب استفاده نماییم که با توجه به ماهیت باینری مورد نظر، این سیگنال ها نیز دوحالته باشند. که برای این منظور از دوسطح ولتاژ متفاوت استفاده می شود.



PAPCO

13

درحالی که در مورد سیگنال ها چیزی بیان نشده بود، Low V. (صفرولت) را معادل صفر منطقی و High V. را معادل 1 منطقی درنظر می‌گیریم.

اختصاص سطوح ولتاژ به مقادیر منطق ←

منطق مثبت: High V. → "1" , Low V. → "0"

منطق منفی: High V. → "0" , Low V. → "1"

دروازه AND براساس سطوح ولتاژ در دو منطق:

منطق مثبت

x	y	x.y
L	L	L
L	H	L
H	L	L
H	H	H

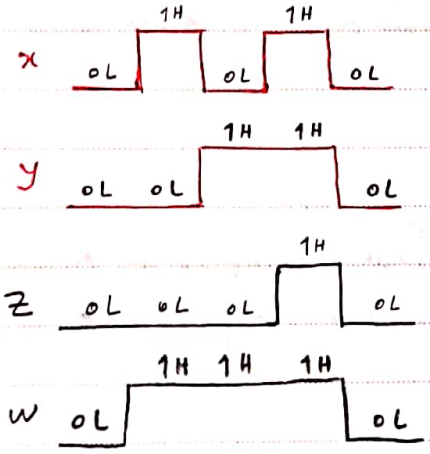
منطق منفی

x	y	x.y
L	L	H
L	H	H
H	L	H
H	H	L

نکته:

تغذیه همان سطوح ولتاژی است که باید به ترانزیستور داده شود.

مثال 8: سیگنال خروجی دروازه AND و OR را برای دو سیگنال x و y بدست آورید.



سیگنال خروجی دروازه AND ←

سیگنال خروجی دروازه OR ←

تقسیم بندی ترانزیستورها براساس پیچیدگی ساختاری آنها:

scale ←

small ← SSI

Medium ← MSI

Large ← LSI

VLSI

جبر بول (Boolean Algebra):

یک سیستم جبر ریاضی که از آن برای کار کردن در سیستم‌های باینری استفاده می‌نماییم.
سیستم جبر بول یک سیستم ریاضی، تعریف شده بر مجموعه عناصر B و دو عملگر با علایم $+$ و $\cdot$ می‌باشد که اصول زیر بر روی آن حکم فرماست:

- اصل اول (B_1): بسته بودن
  - $\forall x,y \in B \longrightarrow x+y \in B$ ← 1 نسبت به $+$
  - $\forall x,y \in B \longrightarrow x \cdot y \in B$ ← 2 نسبت به $\cdot$
- اصل دوم (B_2): جابجایی
  - $\forall x,y \in B \longrightarrow x+y = y+x$ ← 1 نسبت به $+$
  - $\forall x,y \in B \longrightarrow x \cdot y = y \cdot x$ ← 2 نسبت به $\cdot$
- اصل سوم (B_3): عضو خنثی
  - $\exists e \in B, \forall x \in B : e+x = x+e = x$ ← 1 برای عملگر $+$
  - $\exists f \in B, \forall x \in B : f \cdot x = x \cdot f = x$ ← 2 برای عملگر $\cdot$
- اصل چهارم (B_4): عضو معکوس (x' عضو معکوس x)
  - $\forall x \in B, \exists x' \in B : x+x' = x'+x = f$ ← 1 برای عملگر $+$
  - $\forall x \in B, \exists x' \in B : x \cdot x' = x' \cdot x = e$ ← 2 برای عملگر $\cdot$
- اصل پنجم (B_5): پخشی
  - $\forall x,y,z \in B \longrightarrow x+(y \cdot z) = (x+y) \cdot (x+z)$ ← 1 پخش $+$ روی $\cdot$
  - $\forall x,y,z \in B \longrightarrow x \cdot (y+z) = (x \cdot y)+(x \cdot z)$ ← 2 پخش $\cdot$ روی $+$

اصل ششم (B_6): اصل وجودی ← مجموعه B شامل حداقل 2 عضو متفاوت باشد.

جبر بول باینری:

با در نظر گرفتن: مجموعه عناصر $B=\{0,1\}$ و عملگرهای $+$ (OR) و $\cdot$ (AND) و اصول اشاره شده برای جبر بول، جبر بول باینری تعریف می‌شود.

- اصل اول (B_1): بسته بودن
  - $\forall x \in B \longrightarrow x+x \in B$ ← 1
  - $\forall x \in B \longrightarrow x \cdot x \in B$ ← 2
- اصل دوم (B_2): جابجایی
  - $\forall x,y \in B \longrightarrow x+y = y+x$ ← 1
  - $\forall x,y \in B \longrightarrow x \cdot y = y \cdot x$ ← 2
- اصل سوم (B_3): عضو خنثی
  - $\forall x \in B, (e=0) \longrightarrow x+0 = 0+x = x$ ← 1
  - $\forall x \in B, (f=1) \longrightarrow x \cdot 1 = 1 \cdot x = x$ ← 2
- اصل چهارم (B_4): عضو معکوس
  - $\forall x \in B, \exists x' = \bar{x} \in B \longrightarrow x+x' = x'+x = 1$ ← 1
  - $\forall x \in B, \exists x' = \bar{x} \in B \longrightarrow x \cdot x' = x' \cdot x = 0$ ← 2
- اصل پنجم (B_5): پخشی (توزیعی)
  - $\forall x,y,z \in B \longrightarrow x \cdot (y+z) = x \cdot y + x \cdot z$ ← 1
  - $\forall x,y,z \in B \longrightarrow x+(y \cdot z) = (x+y) \cdot (x+z)$ ← 2

اصل ششم (B_6): اصل وجودی ← وجود دو عضو متفاوت

بررسی و اثبات قضایا براساس جدول درستی:

یک روش برای اثبات قضایا، روش جدول درستی می باشد. که در آن باید درستی رابطه در دو طرف تساوی برای تمام مقادیر معتبر برای متغیرها بررسی گردد.

اثبات ← $x\cdot(y+z) = x\cdot y + x\cdot z$ ← مانند

x	y	z	$x\cdot(y+z)$	$x\cdot y + x\cdot z$
0	0	0		
0	0	1		
0	1	0		
⋮	⋮	⋮		
1	1	1		

← ستون جدول ها را کامل کرده و مشاهده می شود که عبارت های دو ستون با یکدیگر برابرند. (روش طولانی است و کاربردی ندارد)

قاعده دوگان (Duality):

در صورتی که در یک عبارت بایزری عملگرهای $+ \leftarrow \cdot$ و $\cdot \leftarrow +$ و تعویض عناصر $1 \leftarrow 0$ و $0 \leftarrow 1$ ایجاد شود، عبارت بایزری بدست آمده معتبر خواهد بود، این دو عبارت را دوگان یکدیگر می نامند.

برخی قضایای جبر بول بایزری:

$$x+x=x \leftarrow 1 \qquad x\cdot x=x \leftarrow 2 \quad \Big\} T_1$$

اثبات 2 ← $x\cdot x \overset{\text{خنثی}}{=} x\cdot x+0 \overset{\text{مکمل}}{=} x\cdot x+x'\cdot x \overset{\text{پخشی}}{=} x\cdot(x+x') \overset{\text{مکمل}}{=} x\cdot 1 \overset{\text{خنثی}}{=} x$

اثبات 1 ← $x+x \overset{\text{خنثی}}{=} (x+x).1 \overset{\text{مکمل}}{=} (x+x)\cdot(x'+x) \overset{\text{پخشی}}{=} x+(x\cdot x') \overset{\text{مکمل}}{=} x+0 \overset{\text{خنثی}}{=} x$

اثبات 1 را می توان از دوگان اثبات 2 بدست آورد.

$$x+1=1 \leftarrow 1 \qquad x\cdot 0=0 \leftarrow 2 \quad \Big\} T_2$$

اثبات 2 ← $x\cdot 0 \overset{\text{خنثی}}{=} x\cdot 0+0 \overset{\text{مکمل}}{=} x\cdot 0+x'\cdot x \overset{\text{پخشی}}{=} x\cdot(0+x') \overset{\text{خنثی}}{=} x\cdot x' \overset{\text{مکمل}}{=} 0$

$$(\bar{x}) = (x')' = x \leftarrow T_3$$

اثبات ← $y = x' \xrightarrow[\text{برای } y]{\text{برقراری مکمل}} \begin{cases} y+y'=1 \\ y\cdot y'=0 \end{cases} \xrightarrow{y=x'} \begin{cases} x'+(x')'=1 \\ x'\cdot(x')'=0 \end{cases} \Rightarrow (x')' = x$

$$x+(y+z) = (x+y)+z \leftarrow 1 \qquad x\cdot(y\cdot z) = (x\cdot y)\cdot z \leftarrow 2 \quad \Big\} T_4 \text{ (شرکت پذیری)}$$

اثبات ←

x	y	z	$y+z$	$x+(y+z)$	$x+y$	$(x+y)+z$
0	0	0				
⋮	⋮	⋮				
1	1	1				

← با تکمیل ستون های جدول درستی رابطه مشخص می شود. قسمت 2 نیز بصورت دوگان قسمت 1 اثبات می شود.

$(x+y)' = x' \cdot y'$ 1
$(x \cdot y)' = x' + y'$ 2 } T_5 (دمورگان)

اثبات 1 → براساس اصل متمم : $\begin{cases} (x+y)+(x'\cdot y') \stackrel{?}{=} 1 \\ (x+y)\cdot(x'\cdot y') \stackrel{?}{=} 0 \end{cases}$

$(x+y)+(x'\cdot y') \overset{\text{شرکت پذیری}}{=} x+[y+(x'\cdot y')] \overset{\text{بخشی}}{=} x+[\underbrace{(y+x')\cdot(y+y')}_{\text{مکمل 1}}]$

$x+(y+x') \overset{\text{مکمل و حشو}}{=} (x+x')+y \overset{\text{شرکت پذیری}}{=} 1+y = y$

اثبات 2 → براساس اصل متمم : $\begin{cases} (x\cdot y)+(x'+y') \stackrel{?}{=} 1 \\ (x\cdot y)\cdot(x'+y') = 0 \end{cases}$

$(x\cdot y)\cdot(x'+y') \overset{\text{شرکت پذیری}}{=} x\cdot[y\cdot(x'+y')] \overset{\text{بخشی}}{=} x\cdot[\underbrace{(y\cdot x')+(y\cdot y')}_{\text{مکمل 0}}]$

$x\cdot(y\cdot x') \overset{\text{مکمل و حشو}}{=} (x\cdot x')\cdot y \overset{\text{شرکت پذیری}}{=} 0\cdot y = 0$

$x+(x\cdot y) = x$ 1
$x\cdot(x+y) = x$ 2 } T_6 (قانون جذب)

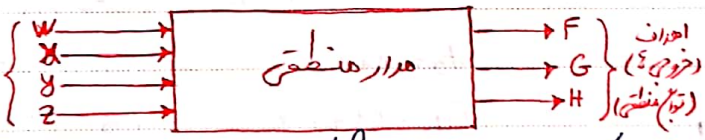
اثبات 2 → $x\cdot(x+y) \overset{\text{حشو}}{=} (x+0)\cdot(x+y) \overset{\text{بخشی}}{=} x+(0\cdot y) = x+0 = x$

اثبات 1 → $x+(x\cdot y) \overset{\text{حشو}}{=} (x\cdot 1)+(x\cdot y) \overset{\text{بخشی}}{=} x\cdot(1+y) = x\cdot 1 = x$

اولویت در محاسبه ارزش یک عبارت جبری :

پرانتز 1 ← NOT 2 ← AND 3 ← OR 4

توابع منطقی (بایتری) (بولی) :



متصور از یک تابع بایتری (منطقی) ، یک عبارت جبری می باشد که براساس چند متغیر بایتری (مستقل) که با عملگرهای بایتری به یکدیگر متصل شده اند . این تابع بایتری می تواند بعنوان یک تابع هدف (خروجی) در مدارهای منطقی باشد.

عامه → $F_{(w,x,y,z)} = wx'y + w'y'z$

مدت آوردن جدول درستی (عملکرد) یا عبارت دیگر مدت آوردن ارزش توابع :

واضح است که چون متغیرهای مستقل ، بایتری بوده و فقط مقادیر 0 و 1 را می گیرند و با عملگرهای بایتری ترکیب شده اند ، حاصل ارزش این عبارت های بایتری 0 یا 1 خواهد بود . بنابراین هر تابع بایتری برای هر ترکیب بایتری 0 و 1 از متغیرهای آن تابع ، یکی از ارزش های 0 یا 1 را خواهد گرفت .

Subject :
Year. 97 Month. 11/12 Date. 30/05

مثال 2 : ارزش توابع زیر را بدست آورید.

$F_1 = xyz'$ $F_2 = x+y'z$ $F_3 = x'y'z + x'yz + xy'$ $F_4 = xy' + x'z$

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

دو تابع معادل (برابر) :

دو تابع f معادل می گویند درصورتی که ارزش آنها برای هرکدام از ترکیبات باینری متغیرهای ورودی توابع بطور متناظر برابر باشد. مانند F_3 و F_4 مثال بالا

نکته :

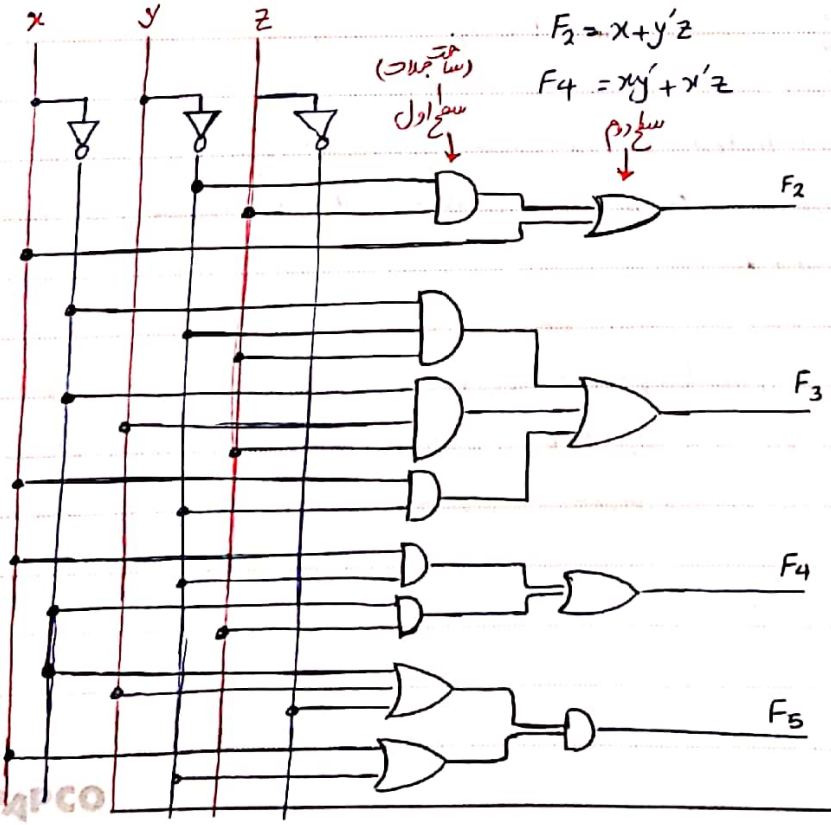
می توان در مدارها از توابع معادل بجای یکدیگر استفاده نمود. ⇐ ساده سازی در توابع باینری

پیاده سازی توابع باینری با دروازه های منطقی :

طبق مثال بالا توابع F_2، F_3 و F_4 پیاده سازی می کنیم.

$F_2 = x+y'z$ $F_3 = x'y'z + x'yz + xy'$

$F_4 = xy' + x'z$ $F_5 = (x'+y+z').(x+y')$



پیاده‌سازی توابع در دو مرحله (دو سطح) انجام شده است. به این نوع پیاده سازی، پیاده سازی دوسطحی گوییم.
در سطح اول جملات توسط متغیرها و مکمل آنها ساخته می‌شوند و در سطح دوم جملات برای ساخت تابع ترکیب می‌شوند.

ساده سازی توابع:

یکی از مهمترین مراحل قبل از پیاده سازی توابع بایزی (یکی از مراحل مهم طراحی مدارهای منطقی) ساده سازی توابع بایزی می باشد. منظور از ساده سازی، بدست آوردن ساده ترین فرم نمایش یک تابع (یعنی فرمی با کمترین دروازه های متغیر) می باشد.

فعلاً در این فصل، ساده سازی را با بکارگیری اصول و قضایای جبر بول بایزی انجام می دهیم.

$x+x'=1$, $x+x+x=x$, $F+1=1$, $x+xy=x$

$x\cdot x'=0$, $x\cdot x\cdot x=x$, $F\cdot 0=0$ ← عالمت

مثال 10: توابع زیر را ساده کنید.

الف) $F_3=x'y'z+x'yz+xy'$

$F_3=x'y'z+x'yz+xy' \overset{\text{پخشی}}{=\!=} x'z(\overbrace{y'+y}^{1})+xy' \overset{\text{مکمل}}{=\!=} x'z+xy'$

ب) $G_1=x+x'y$

$G_1=x+x'y \overset{\text{پخشی}}{=\!=} \overbrace{(x+x')}^{1}(x+y) \overset{\text{مکمل}}{=\!=} x+y$

ج) $G_2=x\cdot(x'+y)$

روش اول: $G_2=x\cdot(x'+y) \overset{\text{پخشی}}{=\!=} \overbrace{xx'}^{0}+xy=xy$

روش دوم: G_2 دوگان G_1 است پس ساده شده آنها نیز دوگان از $G_2 \Rightarrow G_2=xy$

د) $G_3=xy+x'z+yz$

$G_3=xy+x'z+yz=xy+x'z+yz\cdot 1=xy+x'z+yz\cdot(x+x')$

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

ه) $G_4=(x+y)\cdot(x'+z)\cdot(y+z)$

روش اول: $G_4=(x+y)\cdot(x'+z)\cdot(y+z)=(x+y)\cdot(x'+z)\cdot(y+z+0)$

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

$=\dots$

روش دوم: G_4 دوگان G_3 است $\Rightarrow G_4=(x+y)\cdot(x'+z)$

مکمل تابع:

مکمل تابع F یا $\bar{F}=F'$ نمایش می‌دهیم که از نقطه ارزش مکمل ارزش‌های تابع F خواهد داشت.
(یعنی هرجا F برابر 0 باشد ← $F=0$ و هرجا F برابر 1 باشد ← $F=1$). برای بدست آوردن عبارت جبری مکمل یک تابع می‌توان از دو روند زیر استفاده کرد:

1) اعمال دو مرحله‌ای قانون دمورگان
2) ابتدا دوگان تابع را بدست می‌آوریم و سپس کلیه متغیرها را مکمل می‌کنیم.

مثال 11 : مکمل توابع زیر را بدست آورید . (اعمال دومرحله ای قانون دمورگان)

الف) $G_1 = x + x'y$

$$G_1' = \overline{[x + x'y]} = x' \cdot (x'y)' = x' \cdot (x + y')$$

ب) $G_2 = xy + x'z + yz$

$$G_2' = \overline{[xy + x'z + yz]} = (xy)' \cdot (x'z)' \cdot (yz)' = (x' + y') \cdot (x + z') \cdot (y' + z')$$

ج) قسمت الف و ب را از روش دوم انجام دهید .

$G_1 = x + x'y \longrightarrow$ دوگان $G_1 = x \cdot (x' + y) \longrightarrow G_1' = x' \cdot (x + y')$

$G_2 = xy + x'z + yz \longrightarrow$ دوگان $G_2 = (x + y) \cdot (x' + z) \cdot (y + z) \longrightarrow G_2' = (x' + y') \cdot (x + z') \cdot (y' + z')$

نکته :

$$(P + Q)' = P' \cdot Q' \quad , \quad (P \cdot Q)' = P' + Q'$$

نکته :

ساده شده مکمل یک تابع برابر مکمل ساده شده تابع می باشد .

انواع نمایش توابع بایناری :

نمایش هایی که استاندارد هستند

1) نمایش sop (Sum of product) ← نمایشی که جملات با AND ساخته می شود و با OR ترکیب می شود .
(جملات ضربی ← ترکیب جمعی)

مانند ← $G_1 = xy + x'z + yz$

2) نمایش pos ← نمایشی که جملات با OR ساخته می شود و با AND ترکیب می شوند .
(جملات جمعی ← ترکیب ضربی)

مانند ← $G_2 = (x + y) \cdot (x' + z) \cdot (y + z)$

نمایش های کانونیک ← نمایشی از تابع که در تمام جملات آن ، تمام متغیرهای تابع (به فرم مکمل یا عادی) حضور داشته باشند .

1) مجموع minterms (فرم sop) ← جملاتی که با AND ساخته می شوند و در نمایش آن ها تمام متغیرها (فرم عادی یا مکمل) حضور داشته باشند .

minterms حاصل از دو متغیر ← $x \cdot y$ ، $x \cdot y'$ ، $x' \cdot y$ و $x' \cdot y'$ ← 4 متغیر

minterms حاصل از سه متغیر ← $x \cdot y \cdot z$ ، $x \cdot y \cdot z'$ ، $x \cdot y' \cdot z$ ، $x \cdot y' \cdot z'$ ، $x' \cdot y \cdot z$ ، $x' \cdot y \cdot z'$ ، $x' \cdot y' \cdot z$ و $x' \cdot y' \cdot z'$

minterms حاصل از n متغیر ← 2^n متغیر

مانند ← $G_3 = x'y'z + x'yz + xy'z$

2) حاصل ضرب maxterms (فرم pos) ← جملاتی که با OR ساخته می شوند و در نمایش آن ها تمام متغیرها (فرم عادی یا مکمل) حضور داشته باشند .

maxterms حاصل از دو متغیر ← $x + y$ ، $x + y'$ ، $x' + y$ و $x' + y'$ ← 4 متغیر

maxterms حاصل از سه متغیر ← 8 متغیر و maxterms حاصل از n متغیر ← 2^n متغیر

مانند ← $G_4 = (x + y' + z) \cdot (x' + y' + z') \cdot (x + y' + z)$

: Maxterms , minterms

اختصاص ترکیبات باینری به : Maxt. , mint.

با در نظر گرفتن حالت سه متغیری x, y, z :

mint. ← جملات ضربی از متغیرها

AND

Mi	Maxt.	x y z	mint.	mi
M_0	$x+y+z$	0 0 0	$x'y'z'$	m_0
M_1	$x+y+z'$	0 0 1	$x'y'z$	m_1
M_2	$x+y'+z$	0 1 0	$x'yz'$	m_2
M_3	$x+y'+z'$	0 1 1	$x'yz$	m_3
M_4	$x'+y+z$	1 0 0	$xy'z'$	m_4
M_5	$x'+y+z'$	1 0 1	$xy'z$	m_5
M_6	$x'+y'+z$	1 1 0	xyz'	m_6
M_7	$x'+y'+z'$	1 1 1	xyz	m_7

ویژگی AND ← با توجه به 1 شدن خروجی AND در سطری که فقط تمام ورودی های آن 1 باشد.

نحوه اختصاص ← mint. برای یک ترکیب باینری اختصاص داده می شود که عبارت mint. برای آن ترکیب را برگرداند

مثلا: چهار متغیری w, x, y, z : $w x' y' z' \leftarrow m_{10}$ (1010)

چهار متغیری w, x, y, z : $w' x y' z \leftarrow m_5$ (0101)

Maxt. ← جمله جمعی از متغیرها

OR

ویژگی OR ← فقط زمانی صفر است که تمام ورودی های آن صفر باشد و در غیر اینصورت 1 خواهد بود.

اختصاص Maxt ← عبارت Maxt. برای یک ترکیب باینری در نظر می گیریم که Max. برای آن ترکیب باینری صفر شود.

مثلا: چهار متغیری w, x, y, z : $M_{11} \leftarrow w+x+y'+z'$ (1011)

چهار متغیری w, x, y, z : $w+x+y'+z \leftarrow M_2$

نمایش کانونیک توابع باینری :

مجموع mint.

حاصلضرب Maxt.

نمایش کانونیک براساس جدول درستی تابع :

مثال 12 = نمایش کانونیک f_1 داده شده در جدول درستی ۵ بیابید.

$f_{1(x,y,z)}$ برابر است با مجموع mint. هایی متناظر با سطرهایی که f_1 در آن ها برابر 1 باشد.

x y z	mint.	$f_{1(x,y,z)}$
0 0 0	$x'y'z'$	0
0 0 1	$x'y'z$	1
0 1 0	$x'yz'$	1
0 1 1	$x'yz$	0
1 0 0	$xy'z'$	1
1 0 1	$xy'z$	0
1 1 0	xyz'	1
1 1 1	xyz	0

$f_{1(x,y,z)}$ = mint. مجموع $= x'y'z + x'yz' + xy'z' + xyz'$

001

010

100

110

$= m_1 + m_2 + m_4 + m_6 = \sum(1,2,4,6)$

نکته:

با در نظر گرفتن اندیس‌های 1 تابع، mint. متناظر با آن ترکیب باینری ای است که اندیس تابع برای آن 1 شده است و بصورت OR با بقیه جملات می‌نویسیم.

مثال 13: عبارت‌های کانونیک f_2 داده شده در جدول درستی را بیابید:

x y z	mint.	f_2
0 0 0	$x'y'z'$	1
0 0 1	$x'y'z$	1
0 1 0	$x'yz'$	0
0 1 1	$x'yz$	0
1 0 0	$xy'z'$	0
1 0 1	$xy'z$	1
1 1 0	xyz'	0
1 1 1	xyz	0

$$f_{2(x,y,z)} = x'y'z' + x'y'z + xy'z$$
$$= m_0 + m_1 + m_5$$
$$= \Sigma(0,1,5)$$

مثال 14: عبارت‌های Maxt. برای f_2, f_1 را بیابید.

Maxt.	x y z	f_1	f_2
$x+y+z$	0 0 0	0	1
$x+y+z'$	0 0 1	1	1
$x+y'+z$	0 1 0	1	0
$x+y'+z'$	0 1 1	0	0
$x'+y+z$	1 0 0	1	0
$x'+y+z'$	1 0 1	0	1
$x'+y'+z$	1 1 0	1	0
$x'+y'+z'$	1 1 1	0	0

$$f_1(x,y,z) = (x+y+z)\cdot(x+y'+z')\cdot(x'+y+z')\cdot(x'+y'+z)$$

0 0 0 ← $(x+y+z)$

0 1 1 ← $(x+y'+z')$

1 0 1 ← $(x'+y+z')$

1 1 1 ← $(x'+y'+z)$

$$= M_0 \cdot M_3 \cdot M_5 \cdot M_7$$
$$= \pi(0,3,5,7)$$

$$f_2(x,y,z) = (x+y'+z)\cdot(x+y'+z')\cdot(x'+y+z)\cdot(x'+y'+z)\cdot(x'+y'+z')$$
$$= M_2 \cdot M_3 \cdot M_4 \cdot M_6 \cdot M_7$$
$$= \pi(2,3,4,6,7)$$

نکته:

با در نظر گرفتن اندیس‌های 0 تابع، Maxt. متناظر با آن ترکیب باینری ای است که اندیس تابع برای آن 0 است و بصورت AND با بقیه جملات می‌نویسیم.

نکته:

می‌توان از روی یک نمایش، نمایش دیگری بدست آورد بصورتی که شماره‌هایی که جزو یک نمایش نیست، در نمایش دیگر حتماً وجود دارد.

بدست آوردن نمایش کانونیک اروی عبارت جبری (نمایش) استاندارد توابع:

روش اول ← رسم جدول درستی ← نوشتن فرم کانونیک اروی جدول درستی.

روش دوم ← بطور مستقیم عبارت جبری استاندارد به کانونیک تبدیل نماییم.

مثال 15: نمایش کانونیک توابع بزیر بیابید.

الف) $g_{1(x,y,z)} = x'yz + xy + y'z = x'yz + xy\cdot 1 + 1\cdot y'z$

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

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

$= m_3 + m_7 + m_6 + m_5 + m_1 = \Sigma(1,3,5,6,7)$

$\Rightarrow g_{1(x,y,z)} = M_0\cdot M_2\cdot M_4 = \pi(0,2,4) = (x+y+z)\cdot(x+y'+z)\cdot(x'+y+z)$

ب) $g_{2(x,y,z)} = xy + z' = xy\cdot 1 + z'\cdot 1\cdot 1 = xy\cdot(z+z') + z'\cdot(x+x')\cdot(y+y')$

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

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

$= m_7 + m_6 + m_2 + m_4 + m_0 = \Sigma(0,2,4,6,7)$

$\Rightarrow g_{2(x,y,z)} = M_1\cdot M_3\cdot M_5 = \pi(1,3,5) = (x+y+z')\cdot(x+y'+z')\cdot(x'+y+z')$

ج) $g_{3(x,y,z)} = (x+y+z')\cdot(x+z')\cdot(y+z')$

$= (x+y+z')\cdot(x+z'+0)\cdot(0+y+z')$

$= (x+y+z')\cdot(x+z'+yy')\cdot(xx'+y+z')$

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

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

$= M_3\cdot M_1\cdot M_5 = \pi(1,3,5)$

$\Rightarrow g_{3(x,y,z)} = m_0 + m_2 + m_4 + m_6 + m_7 = \Sigma(0,2,4,6,7)$

د) $H_{(x,y,z)} = 1 = \Sigma(0,1,2,3,4,5,6,7)$

ه) $F_{(x,y,z)} = 0 = \pi(0,1,2,3,4,5,6,7)$

عملگرهای منطقی :

با دو متغیر باینری x و y ، می‌توان ۱۶ تابع متفاوت را تعریف نمود .

x	y	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

f_1: $(x+y)'$ — f_2: $x'y$ — f_3: x' — f_4: xy' — f_5: y' — f_6: $x \oplus y$ — f_7: $(x \cdot y)'$ — f_8: $x \cdot y$ — f_9: $x \odot y$ — f_{10}: y — f_{11}: $x'+y$ — f_{12}: x — f_{13}: $x+y'$ — f_{14}: $x+y$

الف) دو تابع ثابت ← $f_{15} = 1$ ، $f_0 = 0$

ب) چهار تابع فقط تابعی از یک متغیر خواهد بود ← $f_5 = y'$ ، $f_3 = x'$ ، $f_{10} = y$ ، $f_{12} = x$

ج) مابقی توابع (۱۰ تابع دیگر) به هر دو متغیر وابسته خواهند بود ←

AND ⟶ $f_8 = x \cdot y$

OR ⟶ $f_{14} = x + y$

NAND ⟶ $f_7 = (x \cdot y)' = (x \uparrow y)$

NOR ⟶ $f_1 = (x+y)' = (x \downarrow y)$

(یا انحصاری) XOR ⟶ $f_6 = x'y + xy' = x \oplus y$ (Exclusive-OR) یا (EX-OR)

(برابری) X NOR ⟶ $f_9 = x'y' + xy = x \odot y$ (Exclusive-NOR) یا (EX-NOR)

چهار تابع زیر دارای ویژگی جابجایی متغیرها نمی باشند . آنها بعنوان یک عملگر نمی توان تعریف نمود .

$f_2 = x'y = x/y$ $\qquad$ $f_{11} = x'+y = x \subset y$

$f_4 = xy' = y/x$ $\qquad$ $f_{13} = x+y' = x \supset y$

دروازه های منطقی :

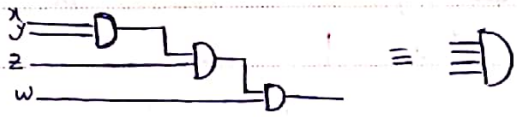
برای آنکه یک تابع باینری را بصورت سخت افزاری برنامه یک دروازه منطقی پیاده سازی نمایم . واضح است که این تابع باید ویژگی عملگر منطقی بودن را دارا باشد . یعنی یکی از شرایط مهم برای ساخت دروازه منطقی برقراری خاصیت جابجایی می باشد . برخی ویژگی های دیگری که دارای اهمیت می باشد (در ساخت دروازه منطقی) عبارتند از :

برآورد خاصیت شرکت پذیری ← ساخت دروازه های با بیش از یک ورودی

هزینه اقتصادی ←

NOT →
AND →
OR →
NAND →
NOR →
XOR →
XNOR →

نکته:
همچنین اگر تعداد متغیرها بیشتر از دو باشد و خاصیت شرکت پذیری برقرار باشد داریم:



تعریف کامل برای ارزش گذاری عملگرهای XOR و XNOR:
← عملگر XOR یک عملگر فرد می‌باشد بدین مفهوم که خروجی آن موقعی ۱ می‌شود که تعداد ۱های ورودی‌های آن فرد باشد.
← عملگر XNOR یک عملگر زوج است بدین مفهوم که خروجی آن موقعی ۱ می‌شود که تعداد ۰های ورودی‌های آن زوج باشد.

x y z	$x \oplus y \oplus z$	$x \odot y \odot z$
0 0 0	0	0
0 0 1	1	1
0 1 0	1	1
0 1 1	0	0
1 0 0	1	1
1 0 1	0	0
1 1 0	0	0
1 1 1	1	1

$x \oplus y = (x \odot y)'$
$x \odot y = (x \oplus y)'$
$x \oplus y \oplus z = x \odot y \odot z$
$(x \oplus y \oplus z)' = x \odot y \oplus z = x \oplus y \odot z$

Subject:
Year. 97 Month. 12 Date. 12/14

ساده سازی توابع بایتری :

روش اول : استفاده از جدول کارنو ← جدول کارنو جدولی است که شامل یکبار خانه برای هرکدام از ترکیبات بایتری است.

جدول کارنو برای دو متغیر (x,y) ←

x	y	mint.
0	0	$x'y'$
0	1	$x'y$
1	0	xy'
1	1	xy

x \ y	y' (0)	y (1)
x' (0)	00 $x'y'$ 0	01 $x'y$ 1
x (1)	10 xy' 2	11 xy 3

جدول کارنو برای سه متغیر (x,y,z) ←

x	y	z	mint.
0	0	0	$m_0(x'y'z')$
0	0	1	$m_1(x'y'z)$
0	1	0	$m_2(x'yz')$
0	1	1	$m_3(x'yz)$
1	0	0	$m_4(xy'z')$
1	0	1	$m_5(xy'z)$
1	1	0	$m_6(xyz')$
1	1	1	$m_7(xyz)$

x \ yz	y'z' (00)	y'z (01)	yz (11)	yz' (10)
x' (0)	000 m_0 0	001 m_1 1	011 m_3 3	010 m_2 2
x (1)	100 m_4 4	101 m_5 5	111 m_7 7	110 m_6 6

نگاشت تابع در جدول کارنو ←

$f_1(x,y) = x'y$ (مفهوم اشتراک : AND)

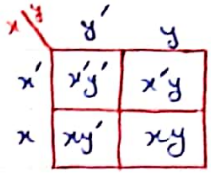
x \ y	y'	y
x'	0	1
x	0	0

$f_2(x,y) = x+y'$ (مفهوم اجتماع : OR)

x \ y	y'	y
x'	0 1	1 0
x	2 1	3 1

x	y	f_1	f_2
0	0	0	1
0	1	1	0
1	0	0	1
1	1	0	1

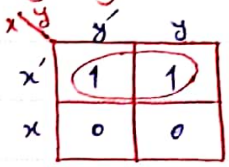
ساده سازی تابع دو متغیری :



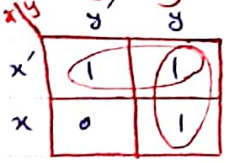
$x'y + xy = \overbrace{(x'+x)}^{1}y = y$

هر دو خانه مجاور هم در جدول کارنو در فرم بایُل یک از متغیرها متفاوت می باشند (یکبار دارو یکبار بدون پریم) . لذا می توان با ترکیب این دو خانه آن متغیر را حذف نمود . (براساس OR شدن دو فرم ملفل هم)

$f_3(x,y) = x'y' + x'y = x'(y+y') = x'$



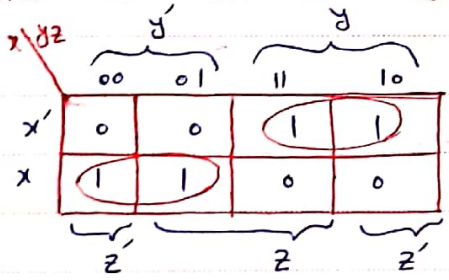
$f_4(x,y) = x'y' + x'y + xy = x' + y$



هر خانه جدول کارنو در راستای ساده سازی می توان بیش از یک بار نیز در ترکیبات شرکت داد .

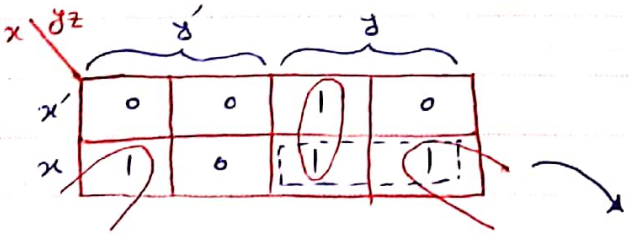
توابع سه متغیری ←

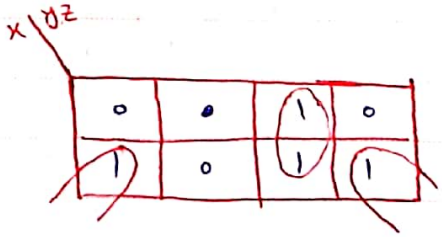
$g_1(x,y,z) = x'yz + xyz' + xy'z' + xy'z = \Sigma(3,2,4,5) = x'y + xy'$



$g_2(x,y,z) = x'yz + xyz' + xyz + xyz' = yz + \cancel{xy} + xz'$

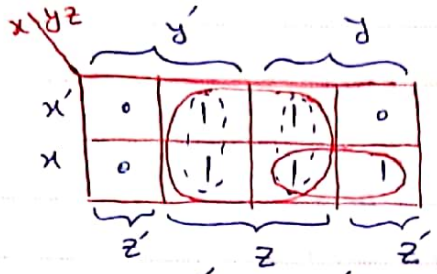
↳ باحذف این قسمت g_2 تغییری نمی کند .





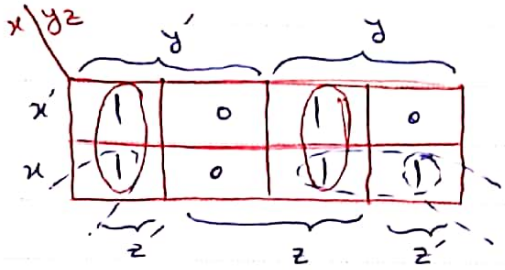
Subject:
Year. 97 Month. 12 Date. 14 ( )

$g_{3(x,y,z)} = y'z + xy'z + yz + xyz' = y'z + yz + xy = z + xy$

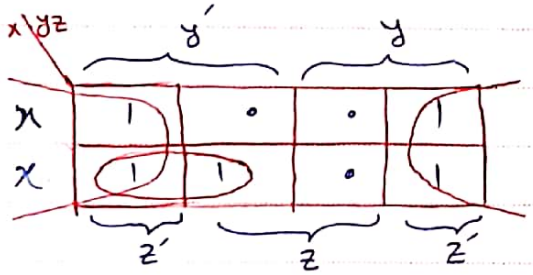


اگر خانه نگاورهم داشته باشیم با ترکیب این خانه دو متغیر حذف می شود و یک عبارت تک متغیری بدست می آید

$g_{4(x,y,z)} = \Sigma(0,3,4,6,7) = yz + y'z' + \begin{cases} xz' \\ xy \end{cases}$

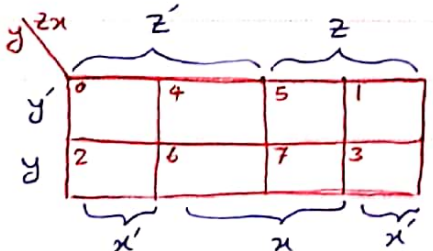


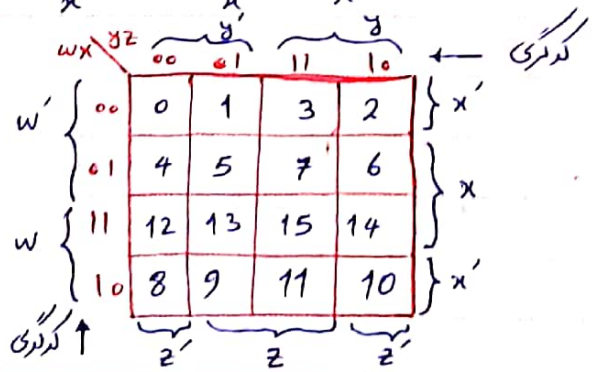
$g_{5(x,y,z)} = \Sigma(0,2,4,5,6) = z' + xy'$



$g_{3(x,y,z)} = y'z + xy'z + yz + xyz' = z + xy$

(ردیف و ستون جدول ۵ عوض می کنیم)





توابع چهار متغیری ←

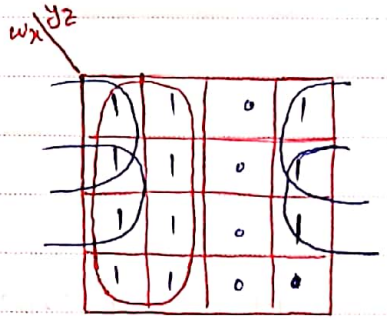
16 mint. ⇐ w, x, y, z

⇐ 16 ترکیب باینری ⇐ جدول کارنو 16 خانه ای

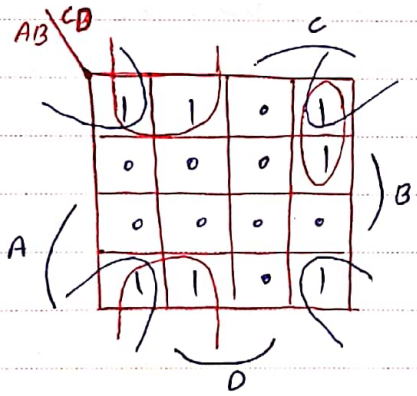
جدول کارنو چهار متغیری ← هر خانه یک جمله چهار حرفی می باشد.

اگر دو خانه مجاور هم داشته باشیم، با ترکیب این دو خانه یک متغیر حذف می شود، یک جمله سه حرفی حاصل می شود.
اگر چهار خانه مجاور هم داشته باشیم، با ترکیب این چهار خانه دو متغیر حذف می شود. یک جمله دو حرفی حاصل می شود.
اگر هشت خانه مجاور هم داشته باشیم، با ترکیب این هشت خانه، سه متغیر حذف می شود. یک جمله یک حرفی حاصل می شود.

$$h_1(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14) = y' + w'z' + xz'$$



$$h_2(A,B,C,D) = A'B'C' + B'CD' + A'BCD' + AB'C' = B'C' + A'CD' + B'D'$$



حالت پنج متغیری ←

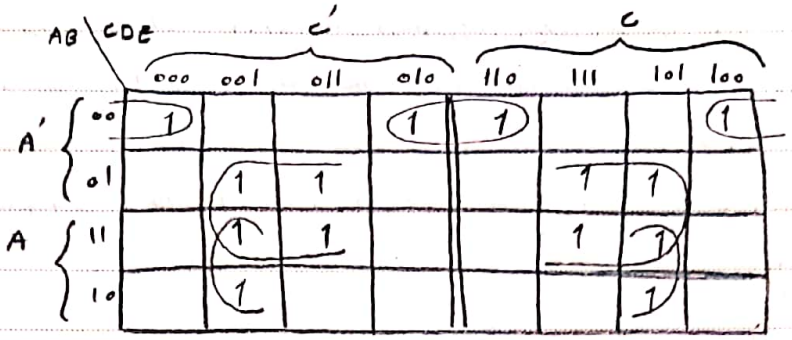
متغیرهای A, B, C, D, E ← $2^5 = 32$ ترکیب باینری ← (mint.) جدول کارنو 32 خانه ای

AB \ CDE	C'				C				
	000	001	011	010	110	111	101	100	
A' 00	0	1	3	2	6	7	5	4	B'
A' 01	8	9	11	10	14	15	13	12	B
A 11	24	25	27	26	30	32	29	28	B
A 10	16	17	19	18	22	23	21	20	B'
	D'		D		D'				
	E'	E	E'	E	E'				

خانه هایی که با تکون جدول به روی یکدیگر قرار می گیرند مجاور هستند. مانند ← خانه های 29 و 25

مثال 16 : تابع زیر را ساده کنید .

$$F_{(A,B,C,D,E)} = \Sigma(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$$



$$\Rightarrow F_{(A,B,C,D,E)} = BE + A'B'E' + AD'E$$

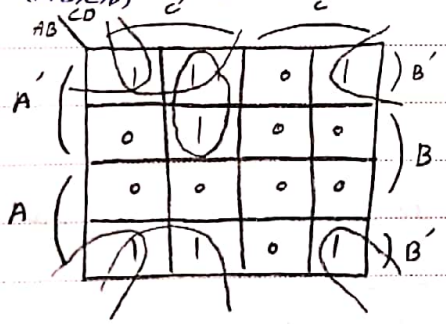
ساده سازی به فرم pos و ساده سازی ناکامل توابع :

← برای ساده سازی فرم ناکامل تابع ، بصورت sop باید صفرهای جدول کارنو تابع را در نظر گرفت و فرم ساده شده را بدست آوردیم .

← برای بدست آوردن فرم ساده شده تابع بصورت pos ، می توان با اعمال ناکامل به ساده شده ناکامل تابع ، فرم مورد نظر را بدست آورد .

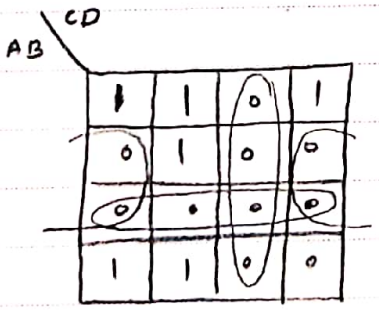
مثال 17 : برای تابع زیر ساده سازی فرم ناکامل تابع و ساده سازی به فرم pos را انجام دهید .

$$F_{(A,B,C,D)} = \Sigma(0,1,2,5,8,9,10)$$



$$\Rightarrow F_{(A,B,C,D)} = B'D' + B'C' + A'C'D$$

ساده سازی فرم ناکامل :



$$F'_{(A,B,C,D)} = BD' + CD + AB$$

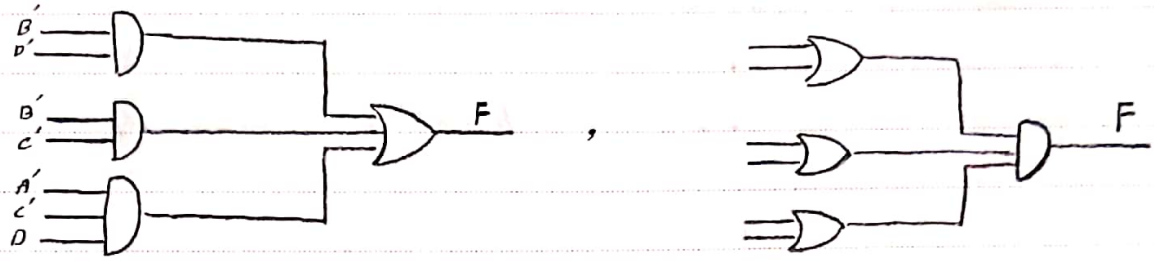
فرم ساده شده بصورت pos :

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

نمایش ساده شده POS برای مکمل تابع :

$$(\text{SOP ساده شده } F)' = F' = [B'D' + B'C' + A'C'D]' = (B+D)(B+C)(A+C+D')$$

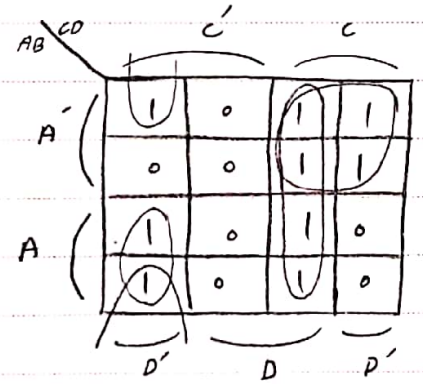
تمام نمایش های ساده شده می‌توان بصورت یک مدار دوسطحی (AND-OR) یا (OR-AND) پیاده سازی نمود.



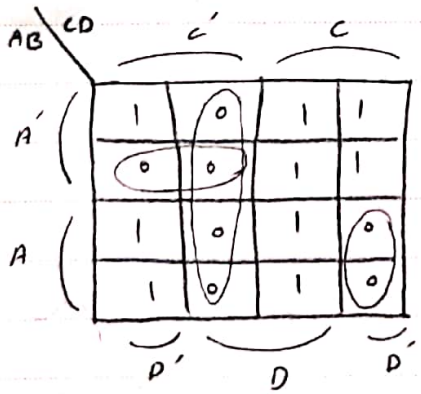
ساده سازی توابع داده شده به فرم استاندارد دوم (POS) یا حاصلضرب Maxt. :
با داکتن شماره Maxt. ، درخانه های متناظر با آن را صفر قرار داده و بقیه خانه ها را می‌گذارند.

مثال 18 : تابع زیر را ساده کنید.

$$G_{(A,B,C,D)} = \pi(1,4,5,9,10,13,14)$$



ترکیب ۱ → $G = CD + A'C + B'C'D' + AC'D'$



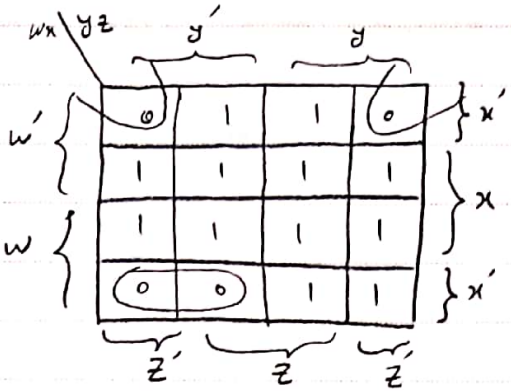
ترکیب ۲ → $G' = C'D + A'BC' + ACD'$

مثال 19 : تابع زیر را ساده کنید.

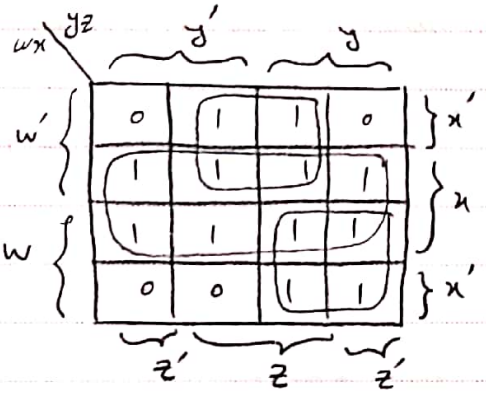
$$H_{(w,x,y,z)} = (w+x+y+z)(w'+x+y)(w'+x+y+z)(w'+x+y+z')(w+x+y'+z)$$

$$\Rightarrow H'_{(w,x,y,z)} = w'x'y'z' + wx'y' + wx'y'z' + wx'yz + w'x'yz'$$

قرار دادن صفرها منتظر با این جملات در جدول کارنو



$$H' = x'z' + wx' \longrightarrow H = (x+z)(w'+x)$$

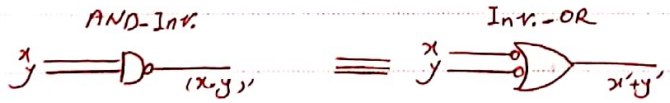


$$H = x + wy + w'z \longrightarrow H' = x' \cdot (w'+y') \cdot (w+z')$$

پیاده سازی (نمایش) توابع دوسطحی با دروازه های NAND ، NOR :

ابتدا فرم های معادل زیر را برای دروازه های NAND ، NOR در نظر می گیریم.

NAND :

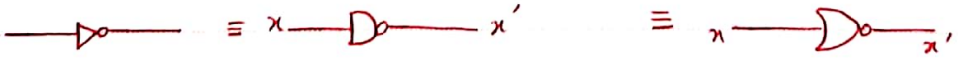


$$(x \cdot y)' \overset{\text{دمورگان}}{=\!=} x'+y'$$

NOR :

$$(x+y)' \overset{\text{دمورگان}}{=\!=} x' \cdot y'$$

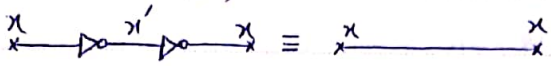
NOT :



نکته:

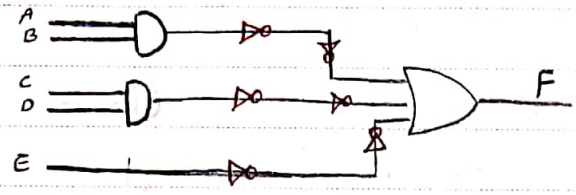
مکمل مکمل هر متغیر = متغیر

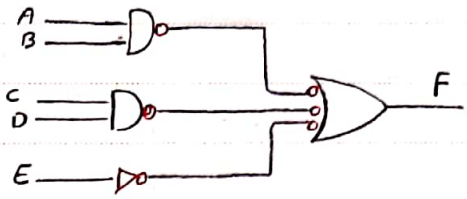
$(x')' = x$

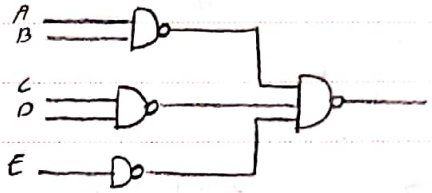


مثال 20: تابع زیر را با دروازه NAND نمایش دهید.

$F_{(A,B,C,D,E)} = AB + CD + E$

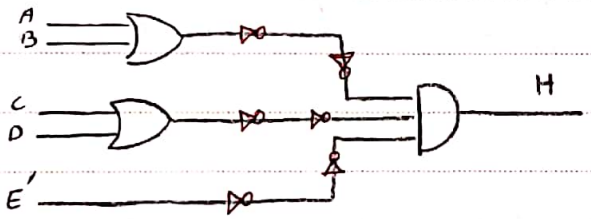


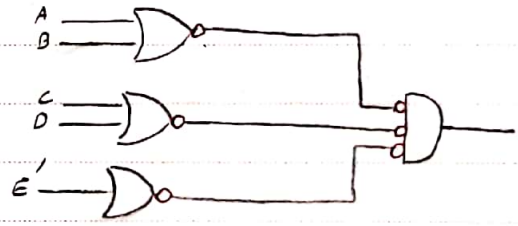


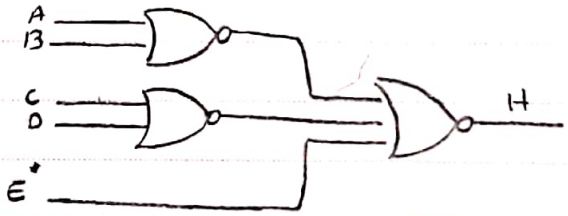


مثال 21: تابع زیر را با دروازه NOR نمایش دهید.

$H_{(A,B,C,D)} = (A+B).(C+D).E'$







Subject :
Year. 97 Month. 12 Date. 21 ( )

نمایش (پیاده سازی) های متفاوت توابع دوسطحی :

با چهار نوع دروازه AND ، OR ، NOR و NAND ، کلاً 16 نمایش دوسطحی میتوان تشکیل داد.
از این 16 نمایش دوسطحی ، 2 نمایش ، نمایش تبدیل شونده به نمایش یک سطحی می باشند. (Degenerate)

AND - AND , OR - OR

واضح است که این هر دو نوع ترکیب ، برای پیاده سازی توابع دوسطحی قابل استفاده نیست و فقط برای پیاده سازی یک جمله مورد استفاده می باشد.

14 ترکیب نمایش دوسطحی باقیمانده ، قابل تبدیل به نمایش یک سطحی نیستند (Non-Degenerate)
بنابراین می توان از آنها برای پیاده سازی توابع دوسطحی استفاده نمود.

دوتایی

دسته AND-OR : AND - OR ، NAND - NAND ، OR - AND ، NOR - NOR : دسته OR-AND

دسته OR-AND-Invr : NOR - OR ، OR - NAND ، NAND - AND ، AND - NOR : دسته AND-OR-Invr

اهمیت نام هایی که در نظر گرفته برای هر دسته (خانواده) در نحوه فرم ساده شده لازم برای انجام پیاده سازی در هر دسته (خانواده) می باشد.

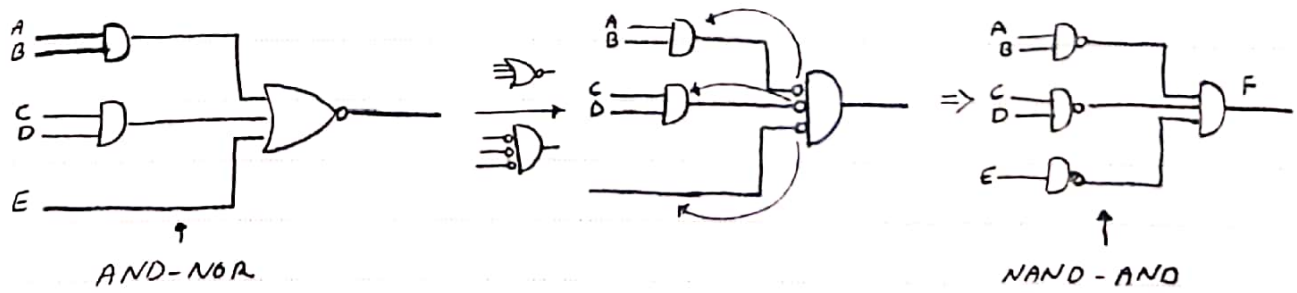
دسته AND-OR-Invr :

برای پیاده سازی توابع با فرم این دسته ، باید فرم ساده شده مکمل این تابع بصورت مجموع حاصل ضرب ها بدست آوریم و سپس مطابق رابطه زیر نمایش تابع را بدست آوریم:

$$F = \left[\underbrace{(\text{ضرب}) + (\text{ضرب}) + (\text{ضرب})}_{F'} \right]'$$

مثال 22 : فرض کنید $F' = AB + CD + E$ باشد ، F را بیابید.
(فرم ساده شده مکمل تابع F)

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



دسته OR-AND-Inv. :

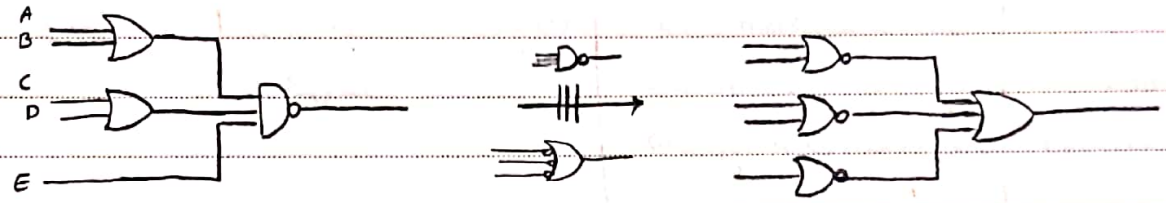
برای پیاده سازی توابع با فرم کلی این دسته تابع باید بصورت :

$$F = [(\text{جمع}) \cdot (\text{جمع}) \cdot (\text{جمع})]' \qquad F'$$

یعنی باید فرم ساده شده مکمل تابع بصورت حاصل ضرب جمع ها داشته باشیم و سپس $F=(F')'$ را پیاده سازی کنیم.

مثال 23: تابع $F' = (A+B)\cdot(C+D)\cdot E$ را پیاده سازی کنید.

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



حالات (شرایط) Don't-Care : (بی اهمیت - نامعتبر - نامطلوب - بی استفاده)

اشاره شده هر تابع باینری به ازای هر ترکیب باینری از متغیرهای آن یکی از دو ارزش 0 یا 1 را اختیار می کند.
ولی در برخی مسائل برخی از ترکیبات ورودی اصلا با حضور بعنوان ورودی یک تابع را ندارند. (مجاز به حضور نیست، امکان رخ دادن آن موجود نیست) به چنین حالات ورودی، حالات D.C. تابع می گویند.

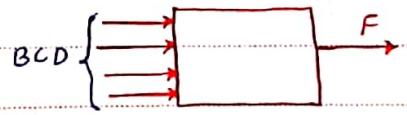
واضح است که تحت این شرایط ارزش تابع برای این حالات D.C. اهمیت نخواهد داشت و به همین دلیل در موقع ارزش گذاری تابع معادل ارزش این حالات را با X نمایش می دهیم.

براساس توضیح فوق علت اهمیت نداشتن ارزش تابع برای حالات D.C. در ساده سازی توابع در صورتی که به ساده سازی بیشتر تابع کمک نماید می توان یکی معادل X یکی از دو ارزش 0 یا 1 بصورت دلخواه قرار داد.
بعنوان مثال، اگر تابعی از کدهای BCD باشد یعنی W,X,Y,Z نشان دهنده یک کد BCD باشد.

$F(W,X,Y,Z)$

W	X	Y	Z	F
0	0	0	0	
⋮				
1	0	0	1	

حالات مجاز BCD



$F(W,X,Y,Z) = \Sigma \dots$

$d(W,X,Y,Z) = \Sigma(10,11,12,13,14,15)$

				F
1	0	1	0	X
		⋮		X
				⋮
1	1	1	1	X

D.C.

Subject:
Year. 97 Month. 12 Date. 21 ( )

مثال 24: برای تابع $F = \Sigma(0,2,5,6,7,8)$ ساده کنید !!!

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

(y, z, w)

=> بدون استفاده از D.C: $F = x'y'z' + w'xz + w'yz$

با استفاده از D.C: $F = xz' + xz + \begin{cases} yz' \\ xy \end{cases}$

روش جدول بندی:

mint. (براساس تعداد 1 های آن)		ترکیب min هر دسته با دسته بعدی در صورت دارا بودن دو شرط: ① شماره پایین بزرگتر ② تفاوت mint ها توان 2 — min	تفاوت	دو شرط قبلی به همراه یکسان بودن بخش تفاوت ها
(0001)	1 ✓	(1,9)	(8)	(8,9,10,11) (1,2)
(0100)	4 ✓	(4,6)	(2)	(8,10,9,11) (2,1)
(1000)	8 ✓	(8,9)	(1) ✓	
		(8,10)	(2) ✓	
(0110)	6 ✓			
(1001)	9 ✓	(6,7)	(1)	
(1010)	10 ✓	(9,11)	(2) ✓	
		(10,11)	(1) ✓	
(0111)	7 ✓			
(1011)	11 ✓	(7,15)	(8)	
(1111)	15 ✓	(11,15)	(4)	

انتخاب جملات نامزد برای حضور در فرم ساده سازی تابع:
یعنی تعیین جملاتی که تاکنون در هیچ ترکیبی مشترک داده نشده اند. (جملاتی که ✓ ندارند)

(1,9) (8) : X 0 0 1 → $x'y'z$

(4,6) (2) : 0 1 X 0 → $w'xz'$

(6,7) (1) : 0 1 1 X → $w'xy$

(7,15) (8) : X 1 1 1 → xyz

(11,15) (4) : 1 X 1 1 → wyz

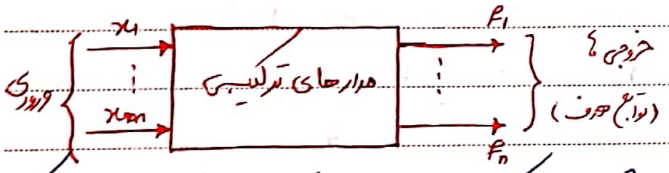
(8,9,10,11) (1,2) : 1 0 X X → wx'

Subject:
Year. 97 Month. 12 Date. 21/01/18

جمله min / نام جمله	1	4	6	7	8	9	10	11	15
✓ $x'y'z$	X					X			
✓ $w'xz'$		X	X						
○ $w'xy$			X	X					
✓ xyz				X					X
○ wyz								X	X
✓ wx'					X	X	X	X	
	✓	✓	✓	? ✓	✓	✓	✓	✓	? ✓

$$F = xy'z + w'xz' + wx' + xyz$$

طراحی مدارهای ترکیبی (ساده):



* یک مدار مجموعه‌ای از عملگرهایی است که پردازش مورد نظر و مطلوب را بروی یک سری ورودی‌های مدار برای رسیدن به یک سری توابع هدف مطلوب (خروجی‌ها) انجام می‌دهد.
* در صورتی که هر خروجی در هر لحظه فقط به ورودی‌های در همان لحظه وابسته باشد، مدار مورد نظر مدار ترکیبی می‌نامند.

$$f_i(x_1, \dots, x_m) = \dots$$

* براساس توصیف فوق واضح است که هر مدار ترکیبی را می‌توان توسط یک سری توابع بایزی معرفی نمود.

روند طراحی مدارهای ترکیبی:

- شناخت دقیق مسئله
- تعیین و نام‌گذاری ورودی‌ها و خروجی‌های مورد نیاز مدار
- تعیین جدول درستی بیان کننده رابطه بین ورودی‌ها و خروجی‌ها
- ساده سازی توابع بایزی خروجی (توابع هدف) (با در نظر گرفتن امکانات موجود مدار)
- پیاده سازی مدار (یکی از مدارهایی که باید در نظر گرفته شود، کم کردن هزینه مدار در طراحی‌هاست)

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

مراحل بیت نیم جمع کننده (Half-Adder) : (HA)

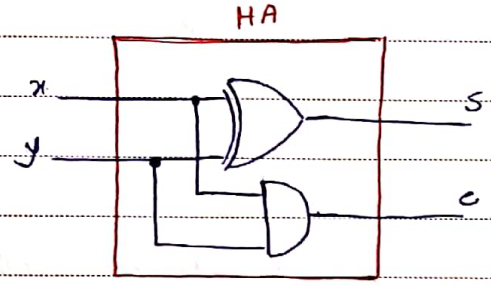
مداری که عمل جمع دو بیت باینری را انجام دهد.

ورودی ۲ عدد: x, y — خروجی ۲ عدد: s حاصل جمع، c بیت نقلی

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\begin{cases} c = xy \\ s = x'y + xy' \end{cases} \qquad \begin{cases} c = xy \\ s = (xy + x'y')' = (c + x'y')' \end{cases}$$

$$\begin{cases} c = xy \\ s = x \oplus y \end{cases}$$



مراحل جمع کننده کامل (Full-Adder) : (FA)

مراحل مداری که عمل جمع سه بیت باینری را انجام دهد: ($x + y + z$ جمع)

ورودی ۳ عدد: x, y, z — خروجی ۲ عدد: S_{FA}, C

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

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

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

$$\Rightarrow s = (x \oplus y) \oplus z$$

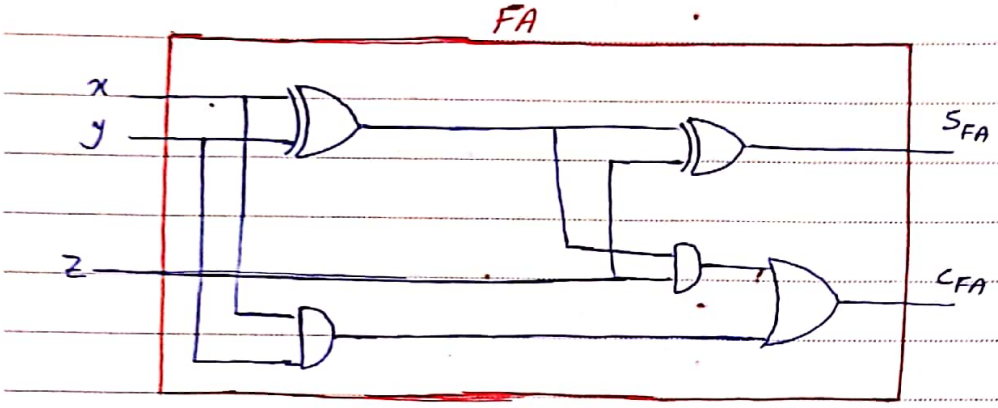
x \ yz	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$c = xy + xz + yz = x'yz + xy'z + xy$$

$$\Rightarrow c = xy + (x \oplus y)z$$

Subject:
Year. 98 Month. 01 Date. 18 ( )

$$(x+y+z) = [\underbrace{(x+y)}_{WA}+z]$$



طراحی یک نیم تفریق کننده (HD):

مداری که عمل تفریق دو بیت باینری را انجام دهد.

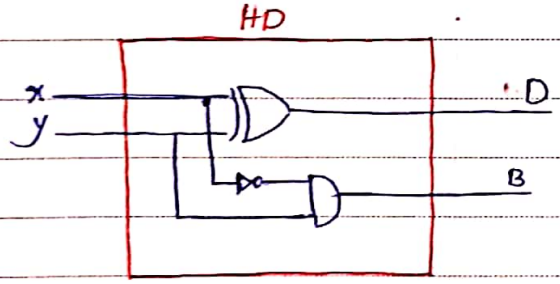
ورودی ها: 2 عدد $\begin{matrix} x \\ y \end{matrix}\}$ 2 عدد

خروجی ها: 2 عدد $\begin{matrix} D: \text{حاصل تفریق} \\ B: \text{بیت قرضی} \end{matrix}\}$

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

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

$B = x'y$



طراحی تفریق کننده تمام (FD):

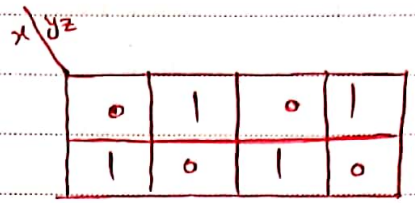
طراحی مداری که عمل تفریق سه بیت باینری را انجام دهد.

$$x-y-z = [(x-y)-z] = x-(y+z)$$

ورودی ها: 3 عدد ورودی $\begin{matrix} x \\ y \\ z \end{matrix}\}$

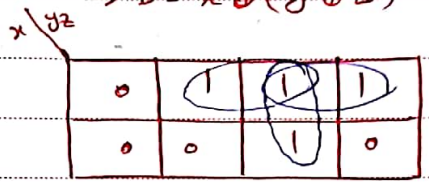
خروجی ها: 2 عدد خروجی $\begin{matrix} D \\ B \end{matrix}\}$

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



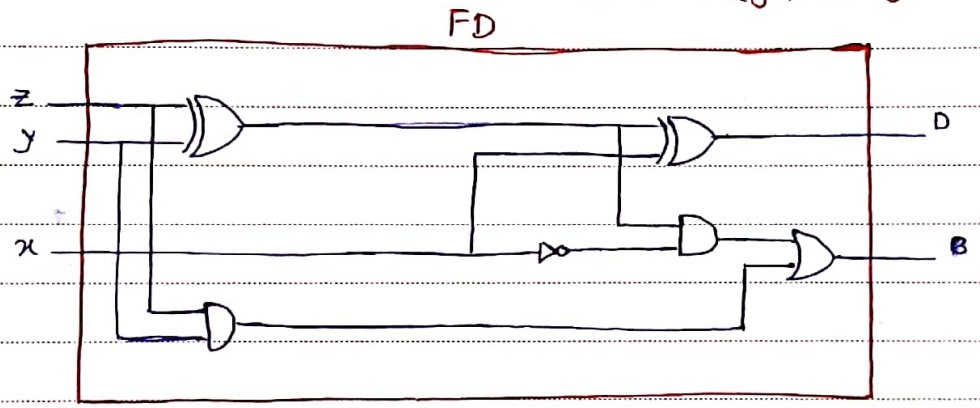
$D = x'y'z + x'yz' + xy'z' + xyz$

$\Rightarrow D = x \oplus (y \oplus z)$

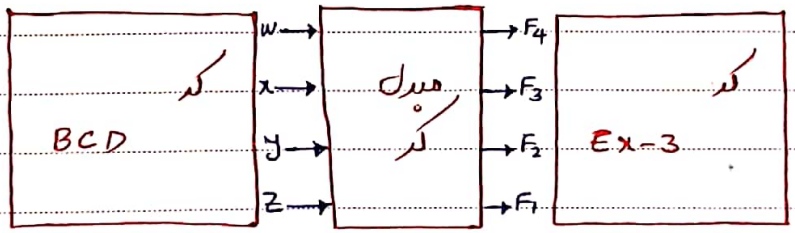


$B = x'y + x'z + yz = x'y'z + x'yz' + yz$

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



طراحی یک مبدل کد:



ورودی‌ها (کد BCD): 4 ورودی
- w
- x
- y
- z

خروجی‌ها (کد Ex-3): 4 خروجی
- F_1
- F_2
- F_3
- F_4

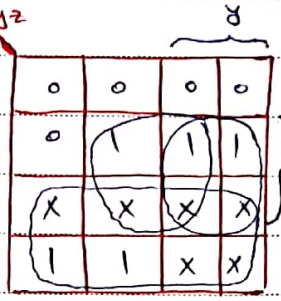
Subject:
Year. 98 Month. 01 Date. 18/1/20

BCD کد + 0011 = Ex-3

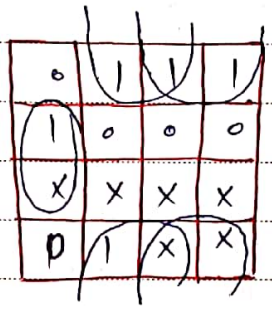
w	x	y	z	F_4	F_3	F_2	F_1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	X	X	X	X
⋮							
1	1	1	1				

کدهای معتبر

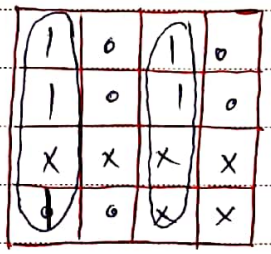
حالات D.-Co



$F_4 = w + xz + xy$



$F_3 = x'z + x'y + xy'z'$



$F_2 = yz + y'z' = y \odot z$

$F_1 = z'$

روش آنالیز:

منظور از آنالیز یک مدار ترکیبی داده شده، بدست آوردن روابط توابع خروجی (حروف) براساس متغیرهای ورودی و (درصورت امکان) تشخیص عملکرد مدار می‌باشد.

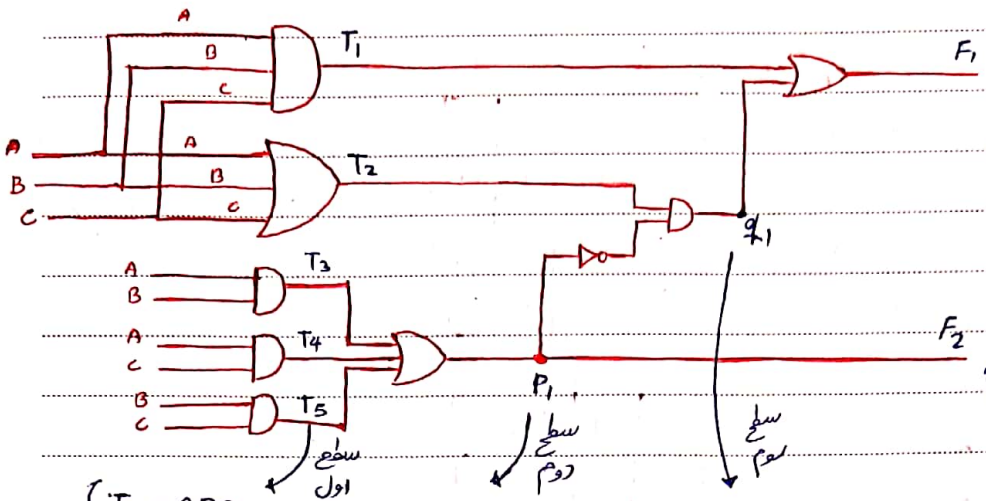
← تشخیص خروجی‌ها و ورودی‌ها و نام‌گذاری آنها

← سطح به سطح توابع خروجی دروازه‌ها را براساس متغیرهای ورودی بدست آورده، تا در نهایت در سطح آخر رابطه توابع خروجی (حروف) براساس متغیرهای ورودی حاصل شود.

← از روی روابط بدست آمده می‌توان جدول عملکرد مدار را رسم کرد و (امکان) تشخیص عملکرد مدار نیز وجود دارد.

PAPCO

41



$$\begin{cases} T_1 = ABC \\ T_2 = A+B+C \\ T_3 = AB \\ T_4 = AC \\ T_5 = BC \end{cases}$$

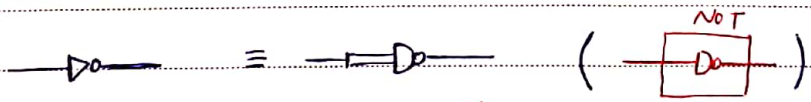
$$\begin{cases} P_1 = T_3 + T_4 + T_5 \\ \quad = AB + AC + BC \end{cases}$$

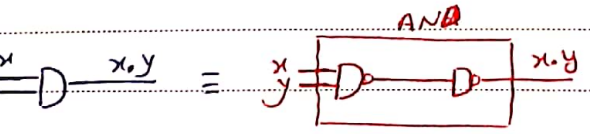
$$\begin{cases} q_1 = T_2 P_1' \\ \quad = (A+B+C)(AC+AB+BC)' \end{cases}$$

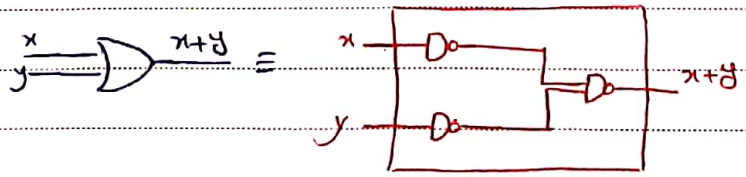
$$\begin{cases} F_1 = q_1 + T_1 = A'BC' + AB'C' + A'B'C + ABC \\ \quad = (A'B + AB')C' + (A'B' + AB)C \\ \quad = \underbrace{(A \oplus B)}_{y} C' + \underbrace{(A \odot B)}_{y'} C \\ \quad = \overbrace{(A \oplus B)}^{y} \oplus C = S_{FA} \\ F_2 = P_1 = AB + AC + BC = C_{FA} \end{cases}$$

مدارهای NAND چند سطحی:

روند تبدیل یک مدار چند سطحی طراحی شده براساس دروازه‌های NOT، OR، AND به یک مدار چند سطحی براساس دروازه NAND به صورت قرار دادن معادل‌هایی بدست آمده برای هرکدام از سه نوع دروازه NOT، OR و AND به جای دروازه NAND در مدار طراحی شده می‌باشد. (واضح است که بعد از قرارگیری دروازه‌های NAND می‌توان ساده سازی جبری براساس قاعده حذف دو NAND تک ورودی در یک خط انجام داد.)

NOT: 

AND: 

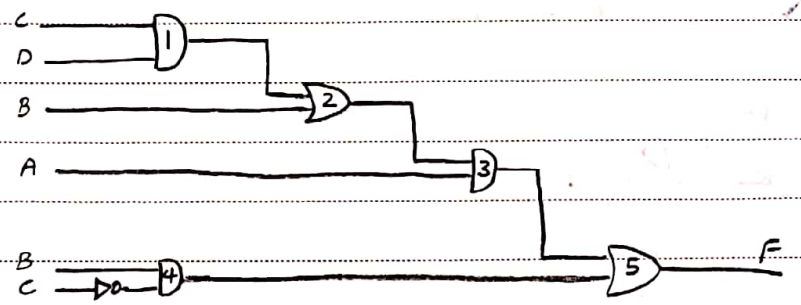
OR: 

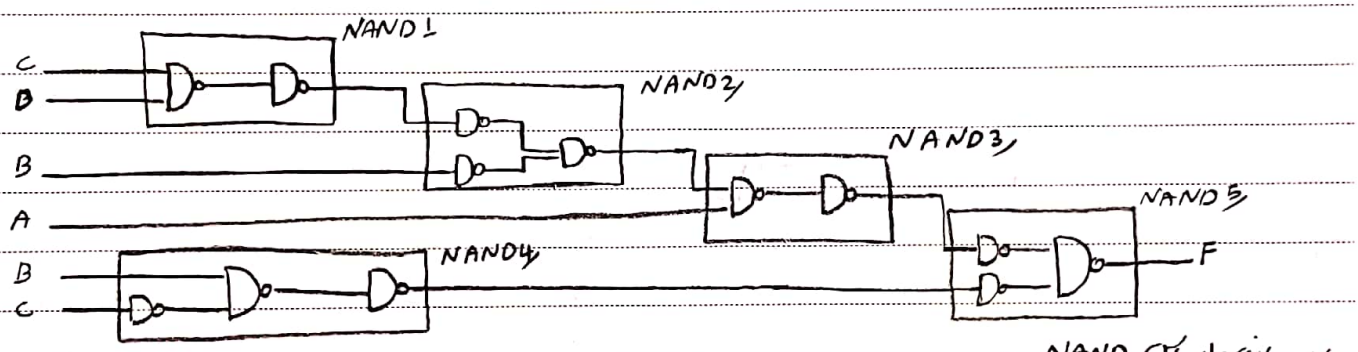
Subject:
Year. 98 Month. 01 Date. 20 ( )

قاعده عکس:

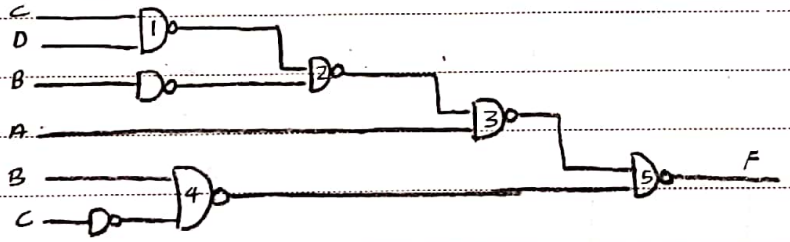
از سطح آخر شبکه در میان یک مدل AND-Inv. دروازه‌ی NAND مدل Inv.-OR آن را با دروازه‌ی دیگر و در نهایت دایره‌ی روی یک خط را حذف می‌کنیم.

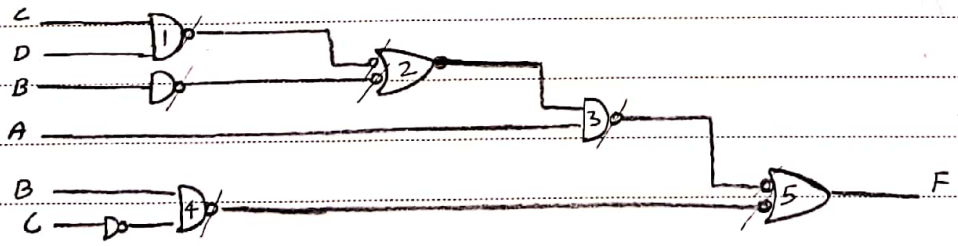
مثال 25: $F = A(B+CD)+BC'$ را به صورت NAND پیاده سازی کنید.

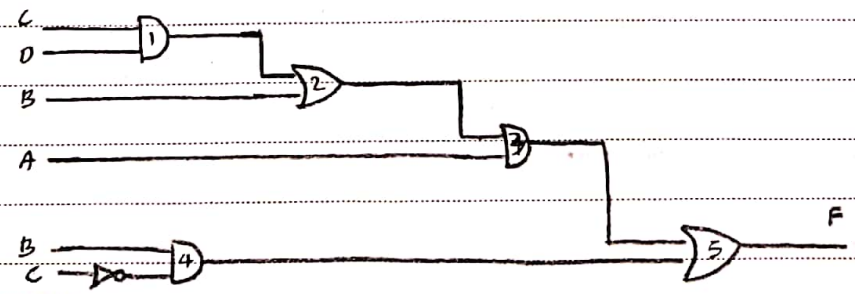




پیاده سازی با استفاده از NAND:







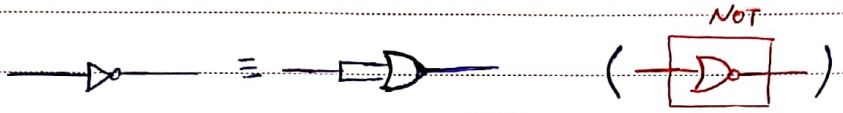
PAPCO

42 43

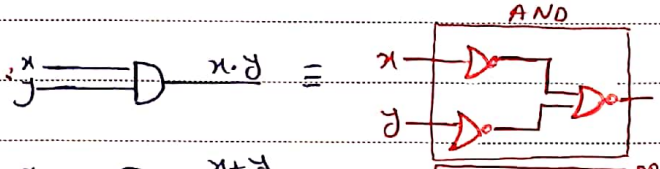
مدارهای NOR چندسطحی:

روند تبدیل یک مدار چندسطحی طراحی شده براساس دروازه‌های NOT، OR، AND به یک مدار چندسطحی براساس دروازه NOR بصورت قرار دادن معادل‌های بدست آمده برای هرکدام از سه نوع دروازه AND، OR، NOT برحسب دروازه NOR در مدار طراحی شده می‌باشد. (واضح است که بعد از قرارگیری دروازه‌های NOR می‌توان ساده‌سازی‌هایی براساس قاعده حذف در NOR تک ورودی در یک خط انجام داد.

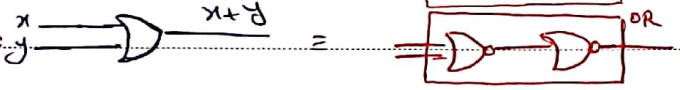
NOT:



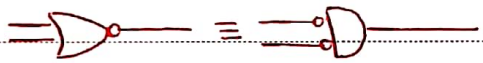
AND:



OR:



قاعده عکس:

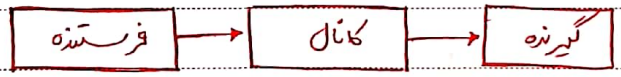


از سطح آخر یک درمیان یک مدار NOR-Inv. دروازه‌های NOR مدل Inv.-AND آنها قرار می‌دهیم و در نهایت دایره‌های روی یک خط را حذف می‌کنیم.

طراحی مدارهای چک کننده و تولید کننده بیت بررسی خطا (parity bit):

مقابله ←

ارسال داده‌های باینری



قانون: فرد کردن تعداد 1ها

101 1

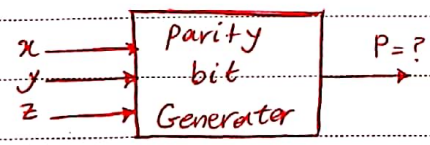
111 0

ارسال

1001

خراب شدن خط

مولد بیت پاریتی:

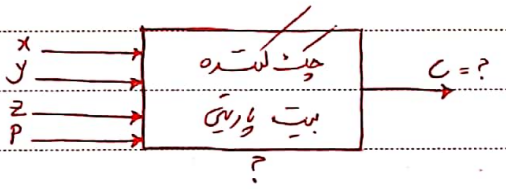


با درنظر گرفتن قانون فرد کردن تعداد 1ها در حل بسته چهاربیتی xyzp

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

x\yz				
	1	0	1	0
	0	1	0	1

$P = \Sigma(0,3,5,6) = [x \oplus y \oplus z]' = x \oplus y$



$C = \begin{cases} 0 \rightarrow \text{(خطا ندارد (تعداد 1های دریافتی فرد باشد} \\ 1 \rightarrow \text{خطا دارد (تعداد 1های دریافتی زوج باشد)} \end{cases}$

ماست ←

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

xy\zp				
	1	0	1	0
	0	1	0	1
	1	0	1	0
	0	1	0	1

$C = [x \oplus y \oplus z \oplus p]' = x \odot y \odot z \odot p$

طراحی مدارهای ترکیبی پیچیده تر (M.S.I):

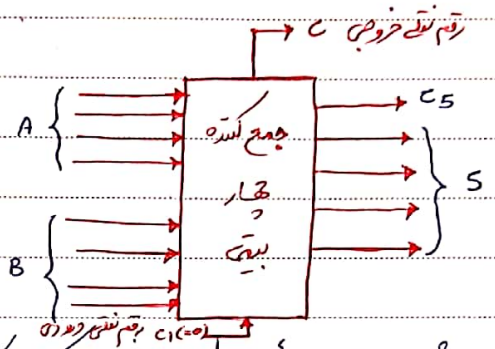
طراحی جمع کننده باینری چهار بیتی ←

$A: A_4\ A_3\ A_2\ A_1$

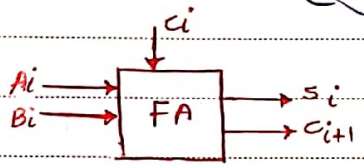
$+\ B: B_4\ B_3\ B_2\ B_1$

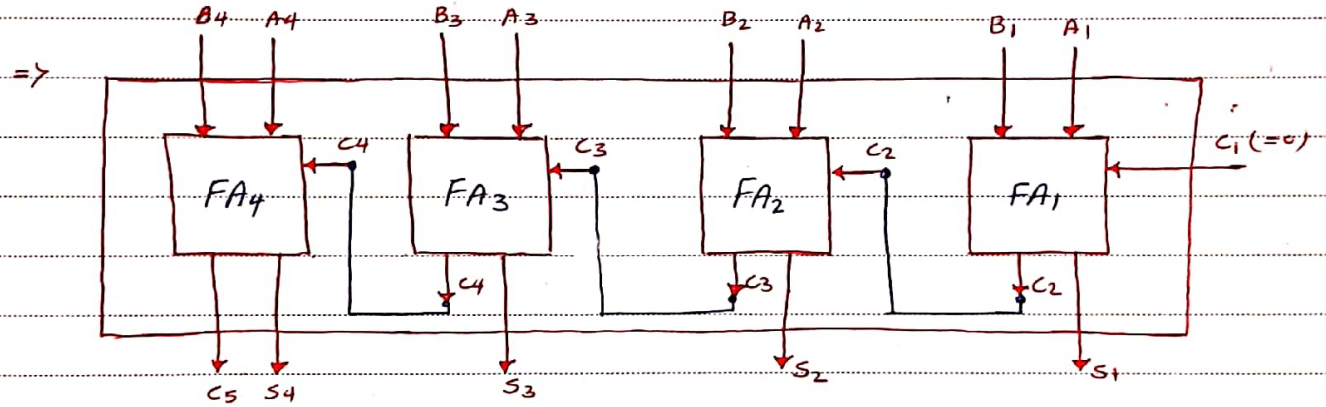
$C_5\ S_4\ S_3\ S_2\ S_1$

$F_4\ F_3\ F_2\ F_1$



با توجه به مدار FA طراحی شده و با توجه به شکل مقابل، واضح است که برای انجام جمع دو عدد چهار بیتی می توان از چهار عدد FA استفاده نمود (طبق جمع باقی):

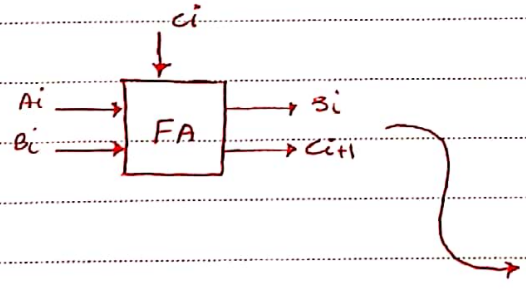


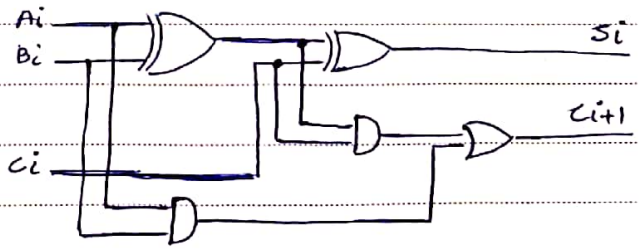


معماری باقی دارای مشکل انتشار رقم نقلی می باشد، یعنی هر FA باید منتظر تولید رقم نقلی در FA مرحله قبل باشد. که برای برطرف نمودن این مشکل از مدار پیش بینی کننده رقم نقلی استفاده شود.

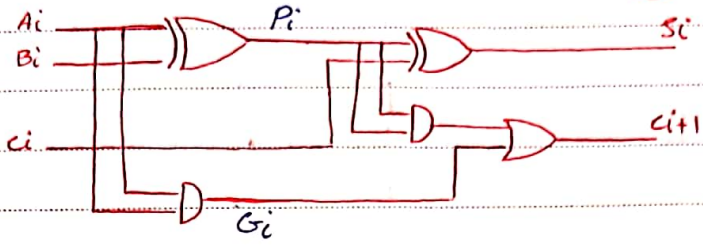
(هر FA یک مدار با سه سطح تأخیر است (3T))

نکته:





مدار پیش بینی کننده رقم نقلی:



برای یک FA داریم:

$P_i = A_i \oplus B_i$

$G_i = A_i \cdot B_i$, $i = 1, 2, 3, 4$

$S_i = P_i \oplus C_i$

$C_{i+1} = P_i C_i + G_i$

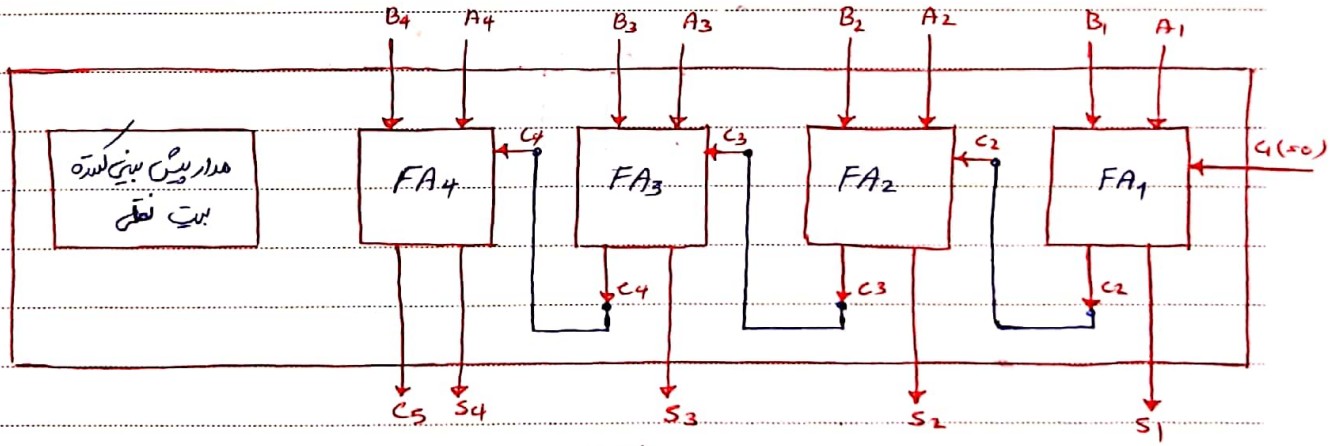
هر چهار عدد FA، متغیرهای ورودی خود را بطور همزمان دریافت سطح تأخیر تولید می کنند.

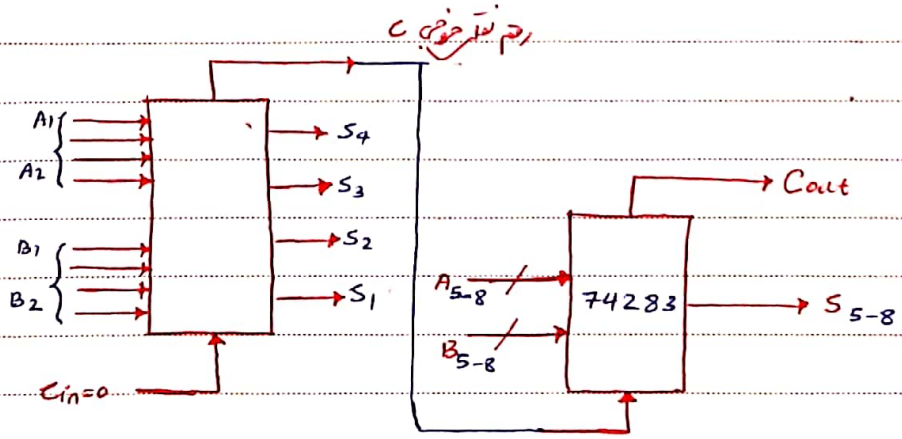
$C_2 = P_1 C_1 + G_1$ ← در دو سطح بعد از سطح اولیه

$C_3 = P_2 C_2 + G_2 = P_2 P_1 C_1 + P_2 G_1 + G_2$ ← در دو سطح بعد از سطح اولیه (یعنی همزمان با C_2)

$C_4 = P_3 C_3 + G_3 = P_3 P_2 P_1 C_1 + P_3 P_2 G_1 + P_3 G_2 + G_3$ ← در دو سطح بعد از سطح اولیه (همزمان با C_2 و C_3)

$C_5 = P_4 C_4 + G_4 = P_4 P_3 P_2 P_1 C_1 + P_4 P_3 P_2 G_1 + P_4 P_3 G_2 + P_4 G_3 + G_4$ ← در دو سطح بعد از سطح اولیه (همزمان با C_2، C_3 و C_4)





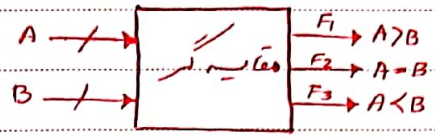
Subject:
Year. 98 Month. 01 Date. 27 ( )
/02 /03

مدار مقایسه کننده دو عدد چهاربیتی :

مداری که دو عدد چهاربیتی A و B را بگیرد و در خروجی‌های آن، نتیجه مقایسه شده دو عدد را نشان می‌دهد.

$A : A_4\ A_3\ A_2\ A_1$

$B : B_4\ B_3\ B_2\ B_1$



برای بررسی تساوی دو بیت از تابع X-NOR می‌توان استفاده کرد:

$$X_i = A_i \odot B_i$$

$$X_i = \begin{cases} 1 & , A_i = B_i \\ 0 & , A_i \neq B_i \end{cases} \quad , i = 1,2,3,4$$

طراحی خروجی F_2 ←

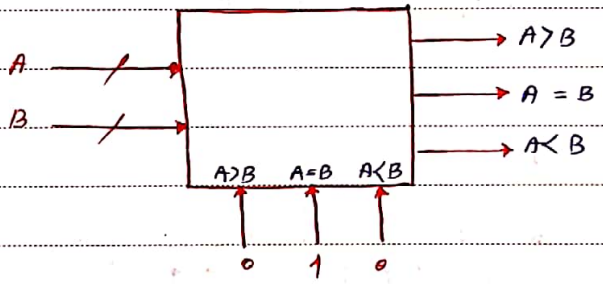
$$F_2 = \begin{cases} 1 & , A = B \\ 0 & , A \neq B \end{cases} \rightarrow F_2 = X_4 X_3 X_2 X_1$$

طراحی خروجی F_1 ←

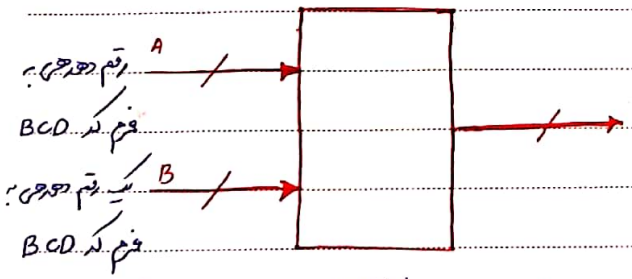
$$F_1 = \begin{cases} 1 & , A > B \\ 0 & , A \not> B \end{cases} \rightarrow F_1 = X_4 B_4' + X_4 A_3 B_3' + X_4 X_3 A_2 B_2' + X_4 X_3 X_2 A_1 B_1'$$

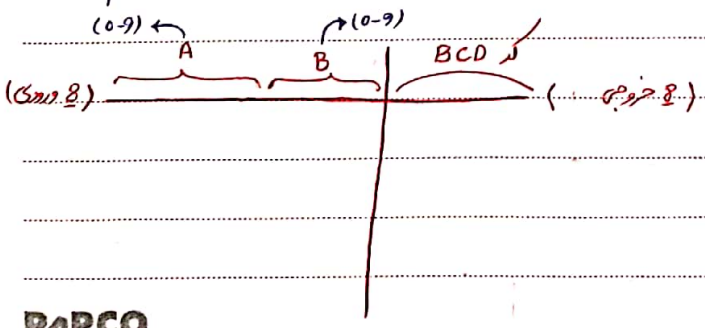
طراحی خروجی F_3 ←

$$F_3 = \begin{cases} 1 & , A < B \\ 0 & , A \not< B \end{cases} \rightarrow F_3 = A_4' B_4 + X_4 A_3' B_3 + X_4 X_3 A_2' B_2 + X_4 X_3 X_2 A_1' B_1$$



جمع کننده BCD (دو رقمی) :

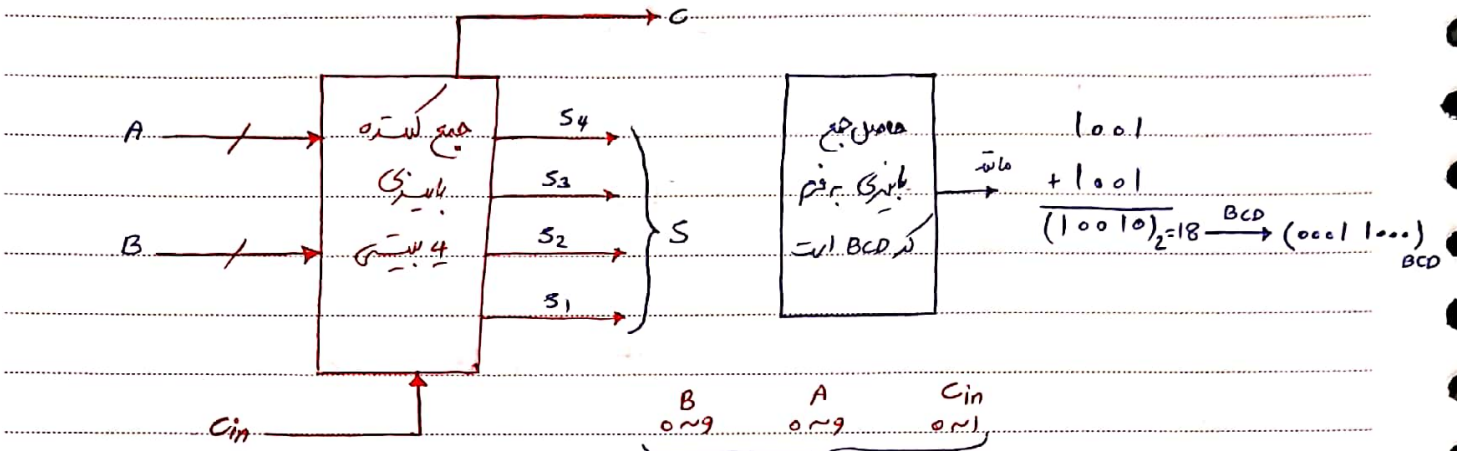




P4PCO

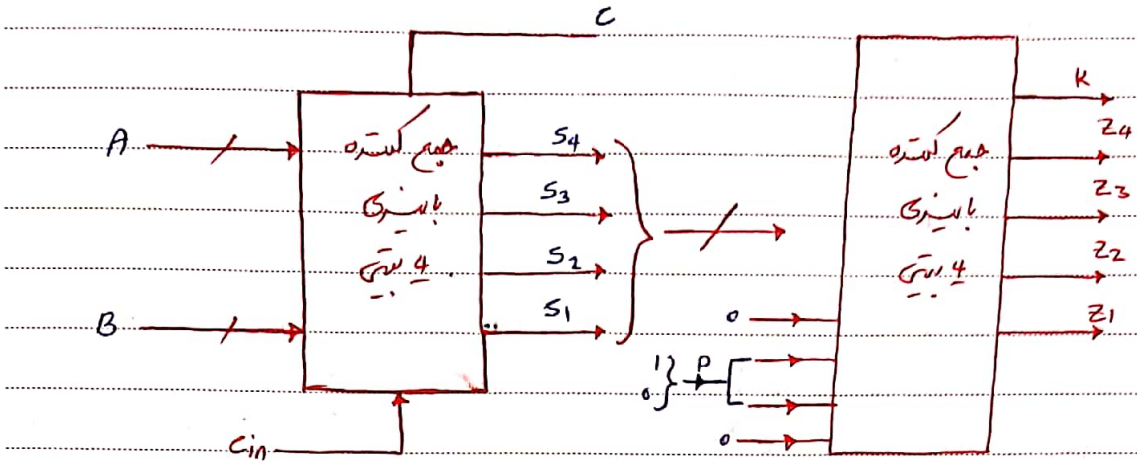
Subject:
Year. 98 Month. 02 Date. 03 ( )

واضح است که برای جمع نمودن دو عدد چهار بیتی A و B می‌توان از یک جمع کننده باینری استفاده کرد. خروجی این جمع کننده حاصل جمع A+B بصورت یک کد باینری می‌شود. و هدف بدست آوردن حاصل جمع بصورت یک کد BCD می‌باشد. بنابراین لازم است تا حاصل جمع باینری دو رقم BCD به کد BCD تبدیل شود.



49 X

X



با توجه به خروجی جمع کننده اول $C\ S_4\ S_3\ S_2\ S_1$ می‌باشد. برای درست آوردن حاصل جمع مورد نظر BCD لازم است از یک جمع کننده چهار بیتی دیگر استفاده نمود که بصورت زیر عملیات P بر روی حاصل جمع CS انجام می‌دهد:

اگر CS در دسته اول باشد یعنی (0 تا 9) باشد ⟵

$P \rightarrow 0 : CS + 0000 = KZ$

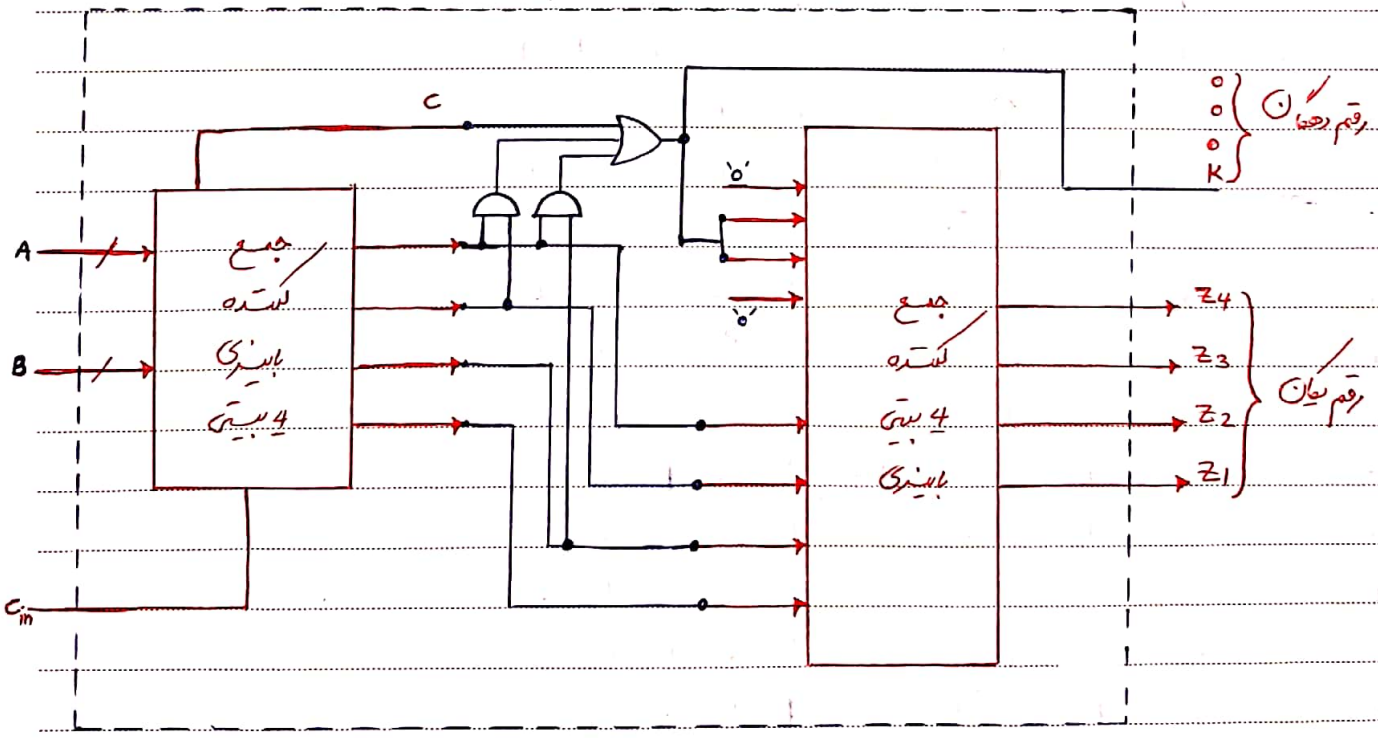
اگر CS در دسته دوم باشد یعنی (10 تا 19) باشد ⟵

$P \rightarrow 1 : CS + 0110 = KZ$

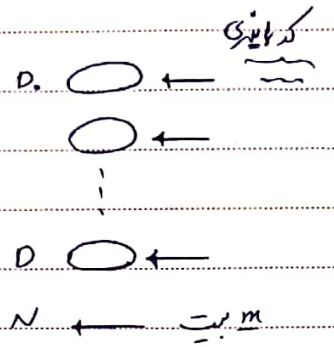
} P

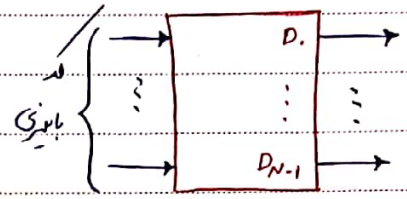
درست آوردن عبارت جبری برای تابع مشخص کننده P:

$$P = C + S_4S_3 + S_4S_2$$



دیکودر (Decoder) یا رمزگشا:

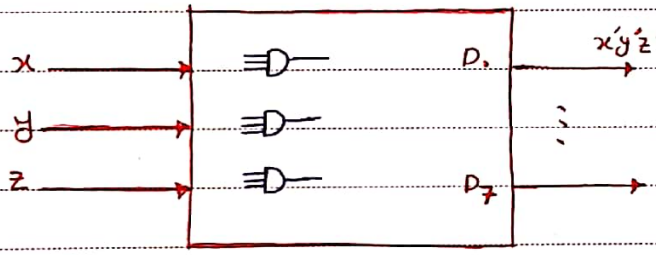




دیکودر مداری است که با ورود n کد متناظر با N پارامتر کدگذاری شده در خروجی، یا پاسخ مورد نظر (پارامتری که در آن وارد شده) را فعال می کند:

$$2^m \geq N$$

دیکودر 3x8 (Dec. 3x8):



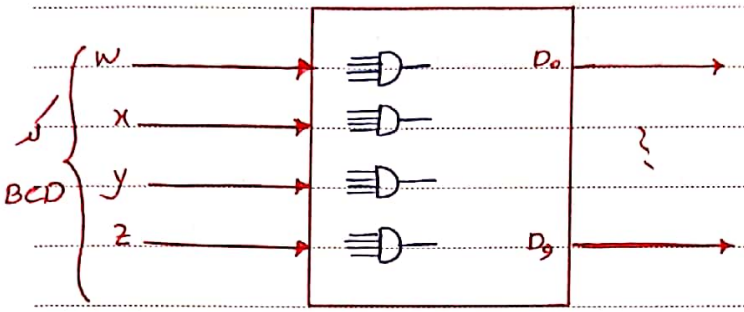
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

$D_0 = x'y'z'$ (m_0)
$D_1 = x'y'z$ (m_1)
$D_2 = x'yz'$ (m_2)
$D_3 = x'yz$ (m_3)
$D_4 = xy'z'$ (m_4)
$D_5 = xy'z$ (m_5)
$D_6 = xyz'$ (m_6)
$D_7 = xyz$ (m_7)

* درجدول اصل تمایزات و همچنین قطر اصلی می تواند ۰ و بقیه درایه ها ۱ باشد. *

نکته:
خروجی های یک Dec. $m \times 2^m$ می‌تواند بعنوان min های عبارتهای ورودی در نظر گرفته شود.

دیکودر BCD به ۱۰ رقم دهدهی:



Dec. 4×10

w	x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1
DC													

wx \ yz

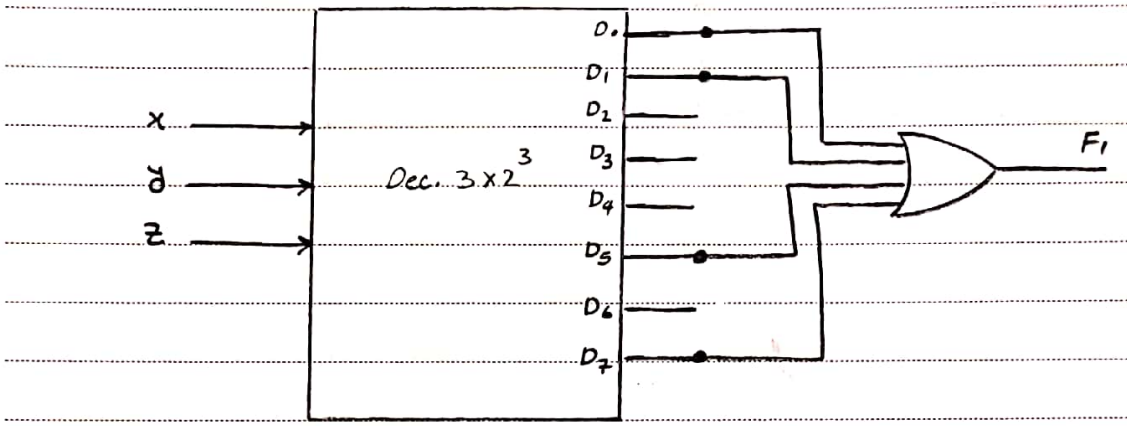
D_0	D_1	D_3	D_2
D_4	D_5	D_7	D_6
X	X	X	X
D_8	D_9	X	X

پیاده سازی توابع باینری توسط Dec. مناسب:

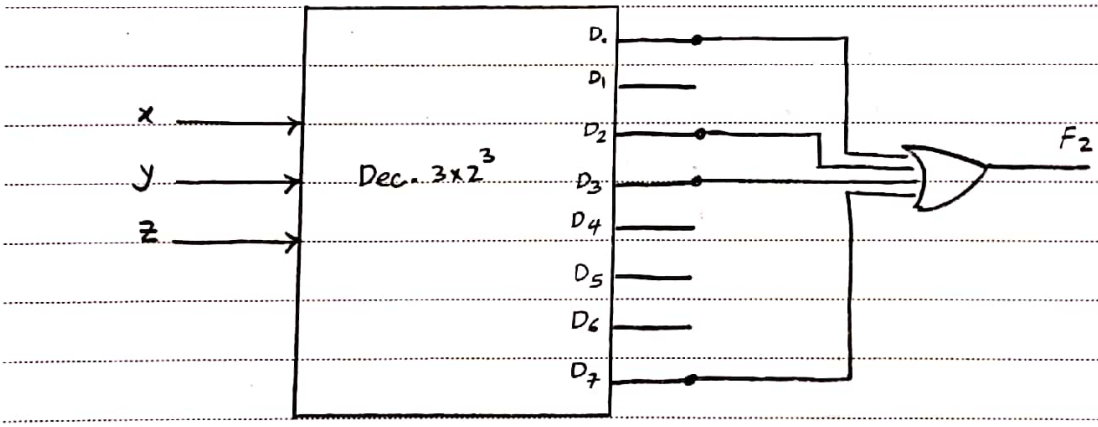
با توجه به آنکه خروجی های Dec. می توان به صورت min های متناظر ورودی ها در نظر گرفت لذا از این مفهوم Dec. می توان برای پیاده سازی توابع استفاده نمود یعنی با یک Dec. $m \times 2^m$ می توان کلیه توابع m متغیری را با به کارگیری دروازه های مناسب پیاده سازی کرد.

مثال 26: توابع زیر را پیاده سازی کنید.

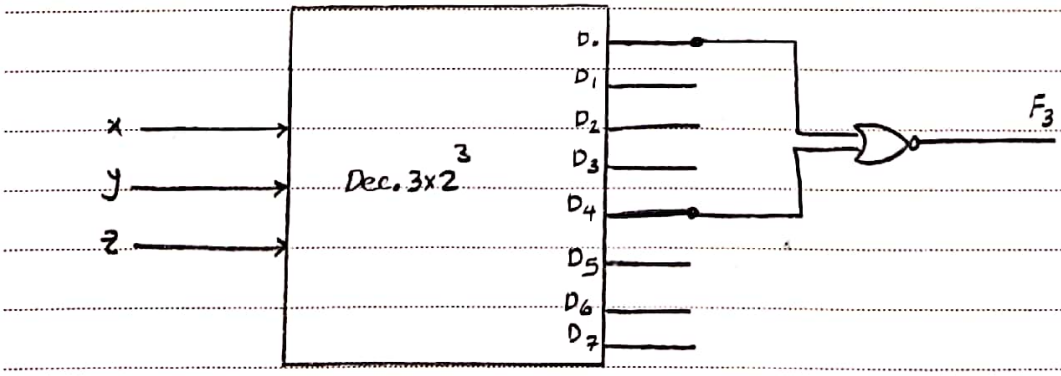
الف) $F_1(x,y,z) = \sum(0,1,5,7) = m_0 + m_1 + m_5 + m_7$



ب) $F_2(x,y,z) = x'y + x'y'z' + yz = \sum(0,2,3,7)$

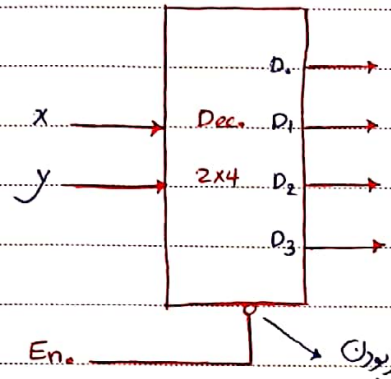


ج) $F_3(x,y,z) = \sum(1,2,3,5,6,7) \rightarrow F_3'(x,y,z) = \sum(0,4)$



دیکودر (Decoder) با خط (ـس) فعال کننده (Enable) :

یک خط (ـس) فعال کننده، یک ورودی به عضو مورد نظر می باشد که با فعال شدن آن، آن عضو عمل اصلی خود را می تواند انجام دهد و در صورت غیر فعال بودن آن خط، عضو مورد نظر تبدیل به غیر فعال می گردد. (این خط (ـس) ورودی غیر از خطوط تغذیه مدار کمی باشد.)



این سر در می تواند فعال یک یا فعال صفر باشد:

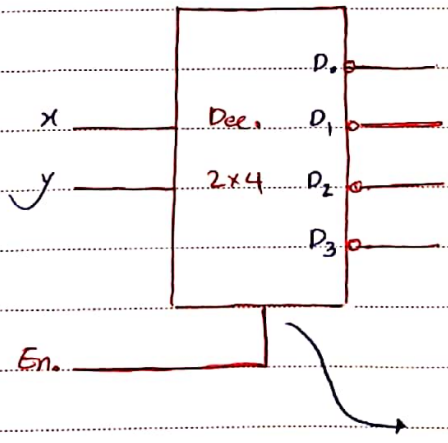
En: { 1 → عضو فعال ; 0 → عضو غیر فعال }

در صورتی که فعال ۰ باشد:

$\overline{En}$: { 1 → عضو غیر فعال ; 0 → عضو فعال }

جدول عملکرد یک Dec.2x4 فعال صفر دارای خط En. فعال ۱ :

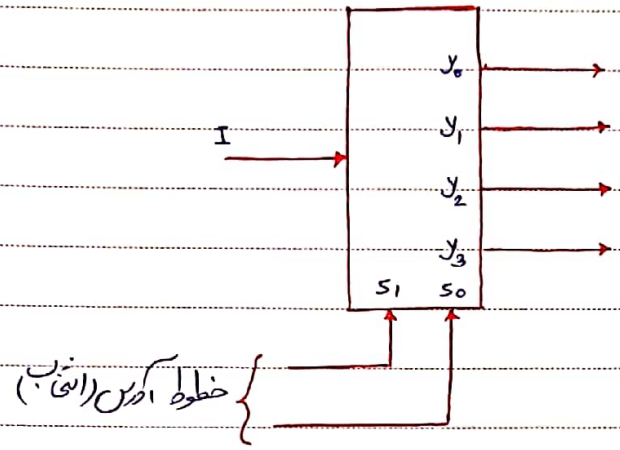
	En.	x	y	D_0	D_1	D_2	D_3
غیر فعال →	0	X	X	1	1	1	1
فعال →	1	0	0	0	1	1	1
	1	0	1	1	0	1	1
	1	1	0	1	1	0	1
	1	1	1	1	1	1	0



Subject:
Year. 98 Month. 02 Date. 08 ( )

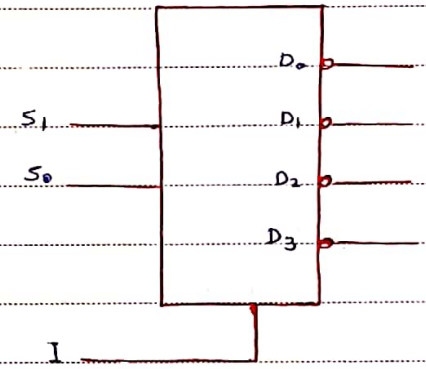
دی مالتی پلکسر (Demultiplexer) یا پخش کننده :

عنصری است که داده‌ی باینری از یک خط ورودی می‌گیرد و بر اساس آدرس مشخص شده بر روی خطوط آدرس آن (خطوط انتخاب) ، ورودی را به یکی از چند خط خروجی خود منتقل می‌کند.



خطوط آدرس به نحوی ، کدگذاری خطوط خروجی را بعهده دارند . (یعنی هر خروجی ، یک آدرس منحصر بفرد باید داشته باشد .) لذا برای یک دی مالتی پلکسر با m خط خروجی ، n خط آدرس لازم است به نحوی که :

$$2^n \geq m$$



I	S_1	S_0	D_0	D_1	D_2	D_3
0	x	x	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	1	1	1	0

با تغییر نقش ورودی‌های یک Dec. می‌توان آن را تبدیل به یک Demux. نمود :

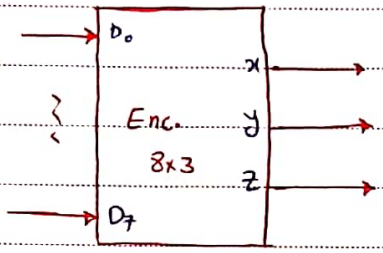
خط En در Dec. ← خط ورودی داده I در Demux

خط ورودی در Dec. ← خطوط آدرس در Demux

Encoder (انکودر) یا کدکننده:

Encoder عمل عکس یک Decoder را انجام می‌دهد یعنی با فعال کردن یکی از چند ورودی Enc.، کد مورد نظر برای آن ورودی بر روی خطوط خروجی Enc. ظاهر می‌شود.

$Enc.\ m \times n\ ,\ m \leq 2^n$



جدول عملکرد یک Enc. 8x3:

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

$$x = D_4 + D_5 + D_6 + D_7$$
$$y = D_2 + D_3 + D_6 + D_7$$
$$z = D_1 + D_3 + D_6 + D_7$$

نکته:
در یک Enc. عادی باید فقط یکی از ورودی‌ها فعال شده باشد و بقیه غیرفعال باشند تا کد ورودی در خروجی مشاهده شود.

نکته:
در Enc. اولویت دار، براساس یک اولویت در نظر گرفته شده، مدارهای Enc. اینگونه می‌پذیرد که در این نوع Enc. می‌توانند ورودی‌های مختلف با هم نیز فعال گردند ولی تنها ورودی دارای اولویت ظاهر می‌گردد.

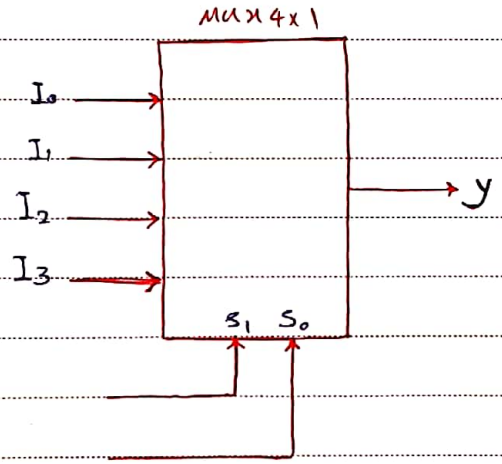
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	0	1	1
1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

(اولویت با ورودی بزرگتر)

مالتی پلکسر (Multiplexer) یا انتخاب کننده:

مالتی پلکسر عضوی است که داده‌های وارد شونده از چند خط ورودی با تعداد محدودی از خطوط خروجی، بطور خاص یک خط خروجی براساس آدرس اعمال شده به خطوط آدرس منتقل می‌کند.

یک mux mx1 دارای m خط ورودی و 1 خط خروجی و n آدرس می‌باشد. $\leftarrow 2^n \leq m$



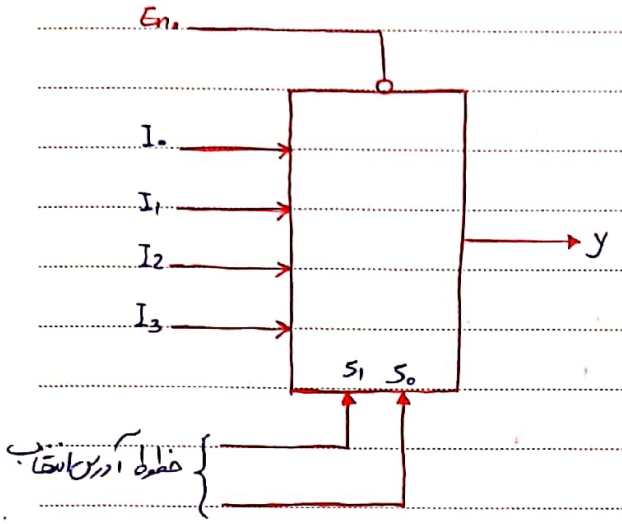
واضح است که هر خط ورودی باید یک آدرس مخصوص بخود داشته باشد.

جدول عملکرد MUX 4x1 =

آدرس S_1	آدرس S_0	خروجی y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$y = \underbrace{S_1' S_0'}_{1} I_0 + S_1' S_0 I_1 + S_1 S_0' I_2 + S_1 S_0 I_3$$

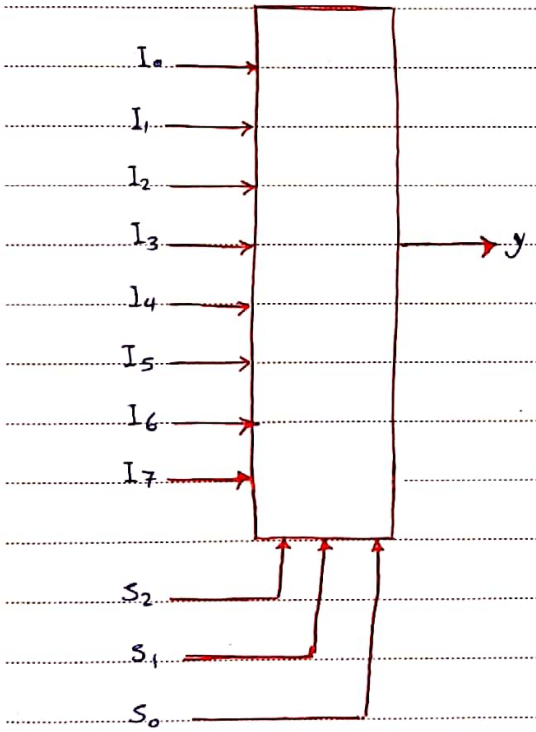
نکته:
یک MUX می‌تواند دارای خط En. (فعال ساز) باشد و این خط می‌تواند فعال صفر یا 1 باشد.



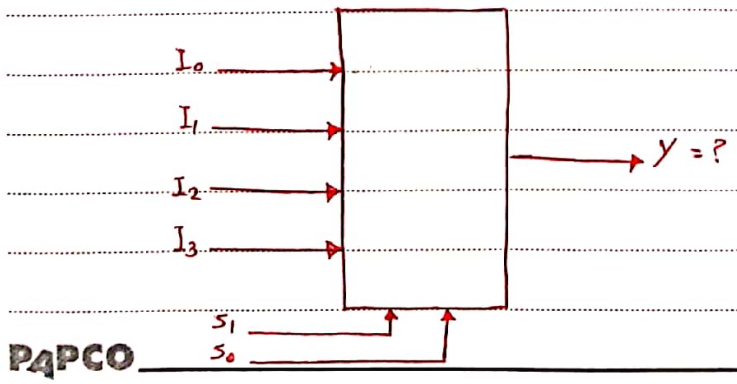
خط En. فعال صفر

آدرس			خروجی
$\overline{En.}$	S_1	S_0	y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	X	X	0

استفاده از MUX برای پیاده سازی توابع باینری:



تابع F زیر را توسط یک MUX مناسب پیاده سازی نمایید:



Subject:
Year. 98 Month. 02 Date. 08/15

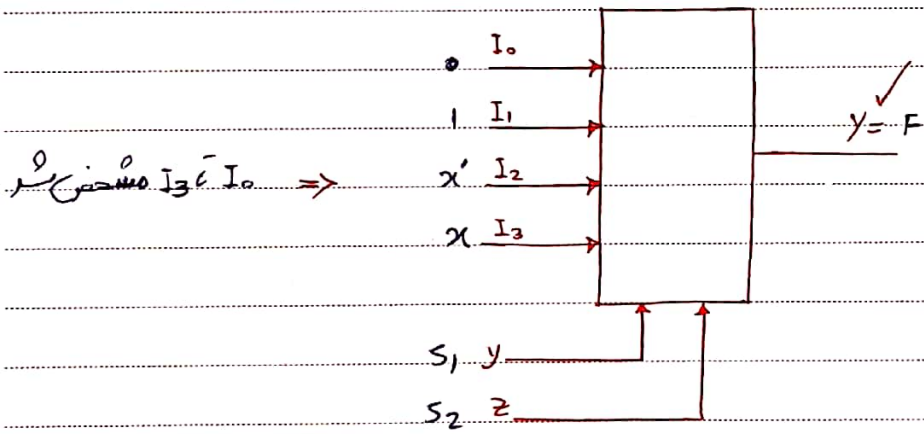
$I_0 \sim I_3$ x	S_1 y	S_0 z	Y F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$S_1 S_0 = 00 \xrightarrow[F=0]{Y=I_0} I_0 = 0 \Rightarrow Y = F$ $(yz = 00)$

$S_1 S_0 = 01 \xrightarrow[F=1]{Y=I_1} I_1 = 1 \Rightarrow Y = F$ $(yz = 01)$

$S_1 S_0 = 10 \xrightarrow[F=x']{Y=I_2} I_2 = x' \Rightarrow Y = F$ $(yz = 10)$

$S_1 S_0 = 11 \xrightarrow[F=x]{Y=I_3} I_3 = x \Rightarrow Y = F$ $(yz = 11)$



I_0 تا I_3 مشخص شد $\Rightarrow$

پیاده سازی توابع باینری با مالتی پلکسر:

← ارتباط با mint های ایجاد شده توسط خطوط آدرس

← برحسب mint های متغیرهای ورودی

مثال ـ تابع $F(x,y,z) = \Sigma(1,3,5,6)$ را با مالتی پلکسر پیاده سازی نمایید.

← مالتی پلکسر مناسب برای پیاده سازی یک تابع n متغیری، یک $MUX\ 2^{n-1} \times 1$ می باشد.

در این مثال : $n=3 \Rightarrow MUX\ 2^2 \times 1$

Subject:
Year. 98 Month. 02 Date. 15 ( )

← بعد از تعیین اندازه MUX، متغیرهای متصل شونده به خطوط آدرس S مشخص می نماییم.
در حالت کلی نحوه تخصیص دلخواه می باشد ولی مناسب تر است تا کم ارزش ترین متغیر به کم ارزش ترین خط آدرس داده شود.

در این مثال : $S_0 = z$, $S_1 = y$

← رسم جدولی که بیانگر ارتباط mint ها با آدرس خطوط داده MUX می باشد.

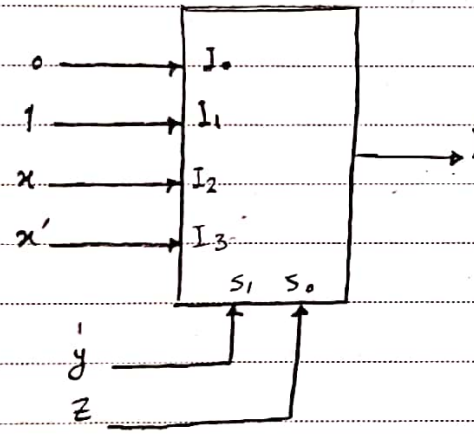
در این مثال : متغیر باقی مانده ← x

$yz \rightarrow$ / $x \downarrow$	I_0 $y'z'$ (00)	I_1 $y'z$ (01)	I_2 yz' (10)	I_3 yz (11)	← $S_1 S_0$
x'	0	(1)	2	(3)	
x	4	(5)	(6)	7	
	0	1	x	x'	← فقط به اطلاعات x' نیاز داریم.

فقط به اطلاعات x نیاز داریم ← $x+x'$

به اطلاعات x و x' نیاز داریم ← 1

به اطلاعات x، x' نیازی نداریم ← 0 ← $(x+x')'$

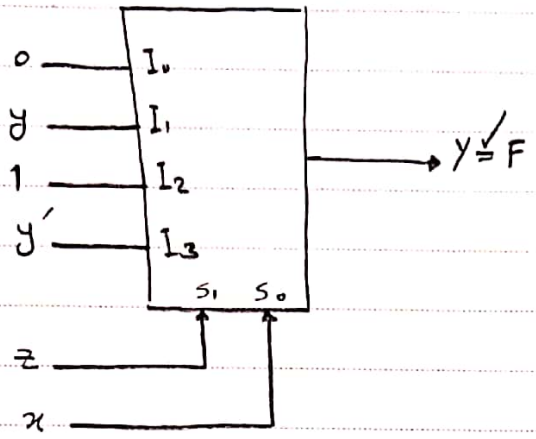


← انتخاب mint ها باید در عبارت تابع حضور داشته باشد.

← تعیین داده ها به گونه ای که براساس متغیرهای باقی مانده باید به خطوط MUX وارد شوند.

مثال — : تابع $F(x,y,z) = \sum(1,3,5,6)$ با درآکثرینتی به MUX 4×1 با خطوط آدرس $S_0 = x$ و $S_1 = z$ پیاده سازی کنید.

$y \downarrow$ / $zx \rightarrow$	I_0 $z'x'$	I_1 $z'x$	I_2 zx'	I_3 zx	← $S_1 S_0$
y'	0	4	(1)	(5)	
y	2	(6)	(3)	7	
	0	y	1	y'	

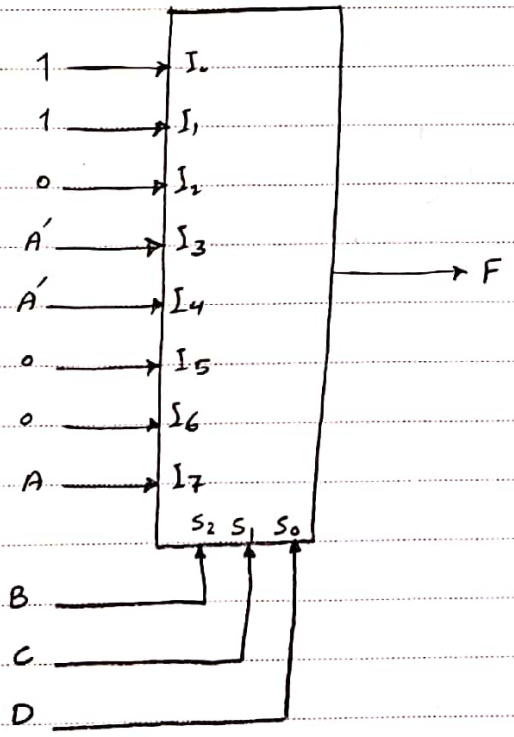


مثال: تابع $F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$ را با مالتی پلکسر پیاده سازی کنید.

$n=4 \Rightarrow$ MUX 8×1

$S_0 = D$, $S_1 = C$, $S_2 = B$

BCD / A	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7 ← $S_2 S_1 S_0$
A'	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15
	1	1	0	A'	A'	0	0	A

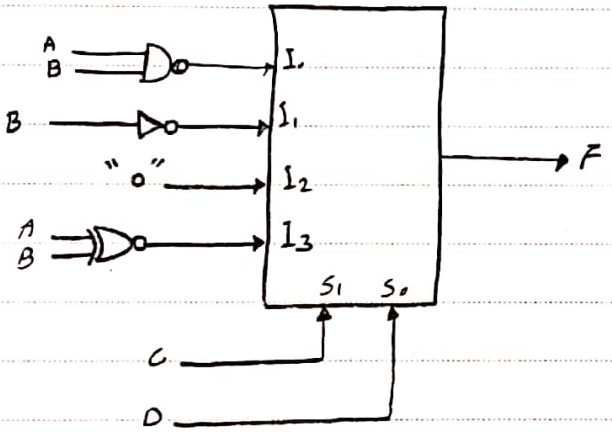


مثال – : تابع $F(A,B,C,D)=\Sigma(0,1,3,4,8,9,15)$ را با MUX 4x1 پیاده سازی نمایید.

$S_1 = C$ ، $S_0 = D$

AB \ CD	I_0 $C'D'$	I_1 $C'D$	I_2 CD'	I_3 CD ← S_1S_0
$A'B'$	(0)	(1)	2	(3)
$A'B$	(4)	5	6	7
AB'	(8)	(9)	10	11
AB	12	13	14	(15)
	$A'+B'$	B'	0	$A \odot B$ → $A'B'+AB$

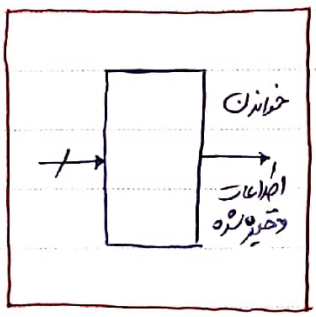
$(A'B'+A'B+AB') = (AB)'$ ← $A'+B'$ ، $A'B'+AB'$ ← B'



حافظه فقط خواندنی (ROM)(Read Only Memory)

برای نگهداری اطلاعات در مدت طولانی از حافظه (Memory) و برای نگهداری اطلاعات در مدت کوتاه از حافظه موقت استفاده می شود.

حافظه ← واحد نگهداری اطلاعات دیجیتالی برای بلند مدت (Memory)

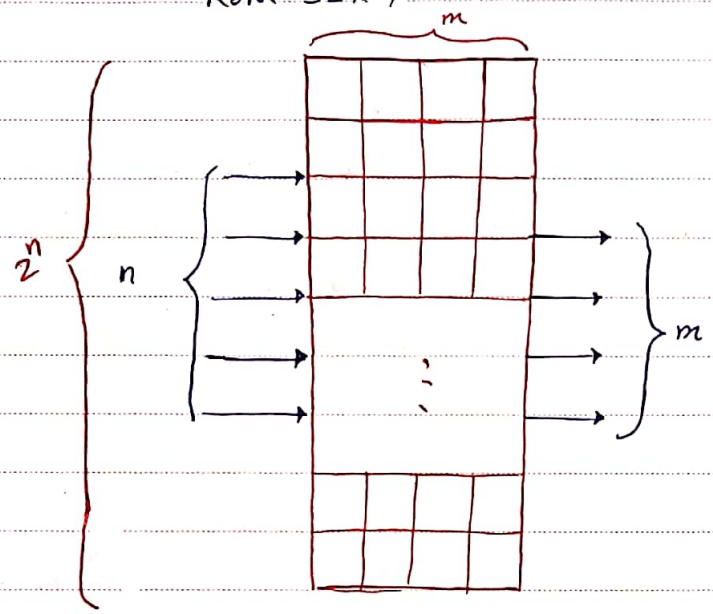


word : 4 8 16 32 64 128

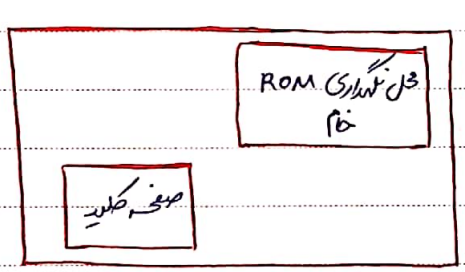
← در حافظه (بطور کلی) هر سطر یا تعریف محل ذخیره یک word سیستم می باشد.
← هر حافظه براساس تعداد کلمات ذخیره شده در داخل آن و تعداد بیت های هر کلمه بیان می شود.

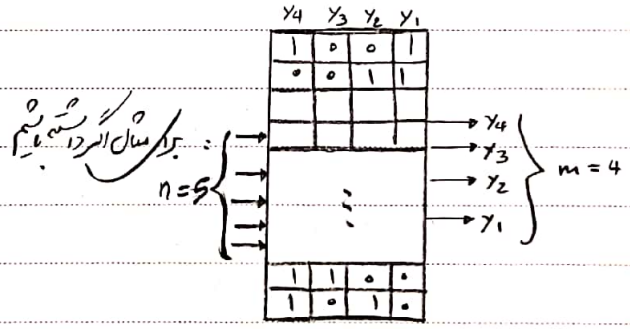
مانند → ROM $2^n \times m$
تعداد کلمات ← → تعداد بیت هر کلمه

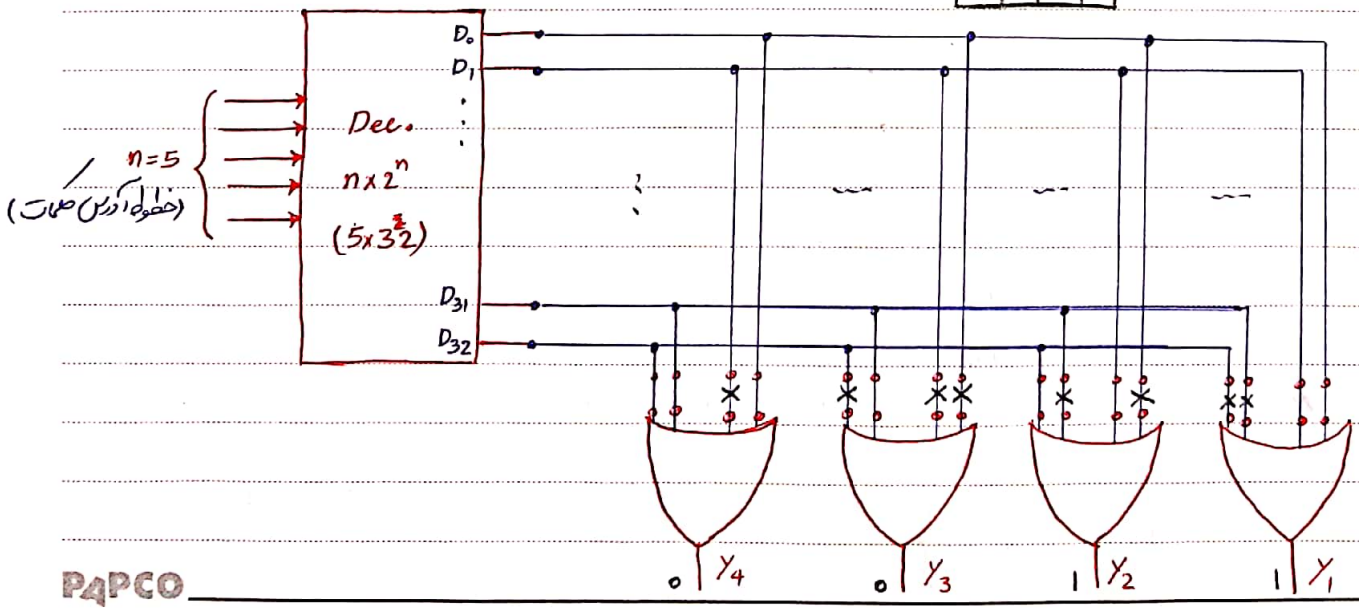
ROM 32×4



← واضح است که هر word باید یک آدرس منحصر بفرد داشته باشد که بعنوان خطوط ورودی ROM عمل کند.





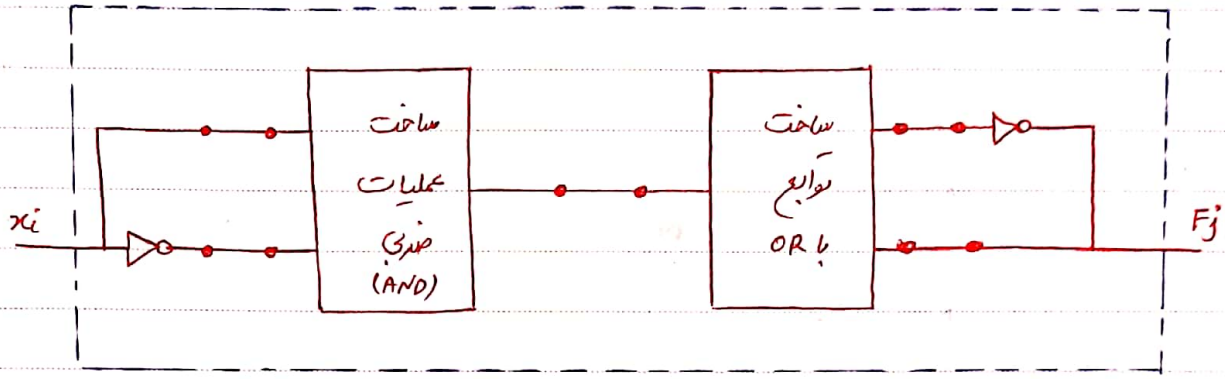


Subject :
Year. 98 Month. 02 Date. 15 ( )

← از ROM می‌توان برای پیاده سازی توابع بایندری نیز استفاده نمود بدین نحو که هرخروجی ROM نمایانگر یک تابع وابسته به خطوط آدرس ROM می‌توان در نظر گرفت :

F_5

PLA (Programmable Logic Array) :



« آرا قابل برنامه ریزی »

← ساده سازی توابع و مکمل آنها

* ← انتخاب جملات ضربی (ساخت با AND) لازم (حداقل جملات لازم) برای پیاده سازی کلیه توابع

← رسم جدول برنامه ریزی PLA (نحوه اتصالات PLA) مشخص می‌نماید.

پیاده سازی F_1 و F_2 با یک PLA :

$$F_1(A,B,C) = \Sigma(3,5,6,7) \quad , \quad F_2(A,B,C) = \Sigma(0,2,4,7)$$

F_1 ساده شده :
$$\begin{cases} F_1 = AC + AB + BC \\ F_2 = B\acute{C}\acute{} + A\acute{C}\acute{} + A\acute{B}\acute{} \end{cases}$$

F_2 ساده شده :
$$\begin{cases} F_2 = B\acute{C}\acute{} + A\acute{C}\acute{} + ABC \\ F_2\acute{} = B\acute{C} + A\acute{C} + ABC\acute{} \end{cases}$$

جملات ضربی ⟵ $B\acute{C}\acute{}$, $A\acute{C}\acute{}$, $A\acute{B}\acute{}$, ABC

جدول برنامه ریزی :

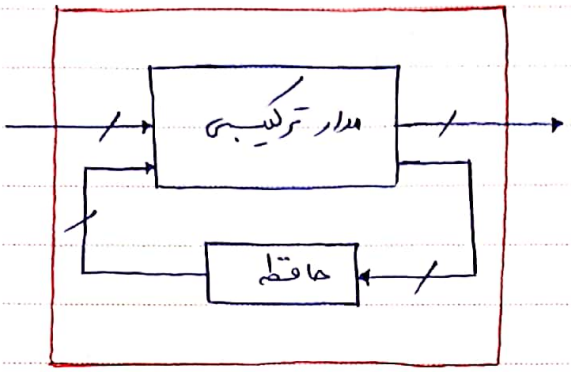
جملات ضربی	A	B	C	F_1	F_2
$B\acute{C}\acute{}$	-	0	0	1	1
$A\acute{C}\acute{}$	0	-	0	1	1
$A\acute{B}\acute{}$	0	0	-	1	0
ABC	1	1	1	0	1
				C	T

1 : داشتن جمله
0 : نداشتن جمله

C : فرم مکمل
T : فرم عادی

قطع اتصال عادی : 0
قطع اتصال با NOT : 1
قطع هر دو اتصال ورودی : -

مدارهای ترتیبی :



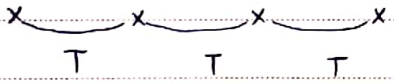
« مدار ترتیبی »

مدارهای ترتیبی ، مدارهایی می باشند که خروجی (های) مدار علاوه بر وابستگی به ورودی(های) مدار ، به حالت مدار نیز وابسته خواهد بود . حالت مدار پارامترهایی از مدار می باشد که در داخل یک واحد حافظه نگهداری می شود و بعنوان ورودی در مدار عمل می کند . این حالت بعد از هر مرحله تولید خروجی ، تولید می گردد . (حالت ، بعنوان یک فیدبک در مدارهای ترتیبی عمل می کند .)

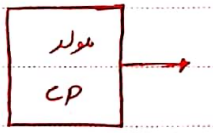
← کاملاً واضح است به دلیل تأثیر حالت مدار ، ترتیب در مدارهای اشاره شده اهمیت خواهد داشت .
این ترتیب در مدارهای ترتیبی توسط یک یا چند متغیر زمانی اعمال می شود . مدارهای ترتیبی از نظر این پارامتر زمانی به دو دسته تقسیم می شوند :

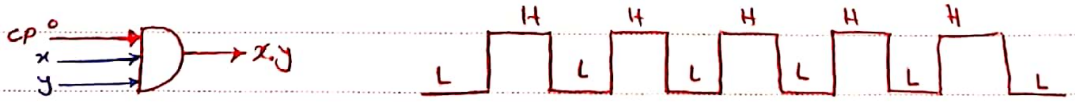
۱) مدارهای ترتیبی سنکرون (هم زمان) ← در این مدارها تأثیر ورودی و حالت مدار و تولید خروجی تحت کنترل یک سیگنال زمانی بوده و اعمال تغییرات در فاصله های زمانی مشخص رخ می دهد .

۲) مدارهای ترتیبی آسنکرون (غیر هم زمان) ← در مدارهای آسنکرون لزوماً کلیه بخش های آن تحت کنترل زمانی سیگنال زمانی نمی باشند و اعمال تغییرات در این بخش ها با اعمال تغییر در ورودی و حالت اتفاق می افتد .



← در مدارهای ترتیبی یک سیگنال زمانی (سیگنالی است که کنترل زمانی مدار را بعهده دارد) لازم می باشد . معمولاً این سیگنال زمانی توسط واحدی بنام واحد پالس ساعت تولید می شود . این مولد پالس ساعت یک سیگنال دو حالته با فرکانس های مشخص و دوره های T و $\frac{1}{f}$ مشخص تولید می کند .



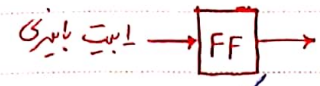


(هرگاه ۰ باشد در حالت (L) و هرگاه ۱ باشد در حالت (H) است .)

(FF) Flip-Flop :

همانطور که اشاره شد، در مدارهای ترتیبی، یک بلوک حافظه برای نگهداری حالت مداری لازم می‌باشد که این بخش توسط مدارهای باناگ (FF) ساخته می‌شود.

← یک FF عنصری است که عمل ثبت، نگهداری و انتقال یک بیت باینری را می‌تواند عهده دار گردد.



← براساس پیچیدگی مدار و تعداد حالات مدار، تعداد بیت‌هایی که باید در واحد حافظه مدار ترتیبی نگهداری شود تعیین شده و بر همین اساس همان تعداد FF در ساختار حافظه قرار می‌گیرد.

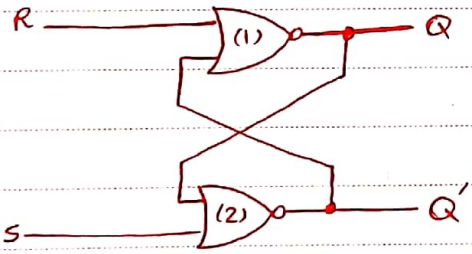
← FF ها معمولاً دارای دو خروجی که مکمل یکدیگر می‌باشند، هستند و برای دادن اطلاعات به آنها یک یا دو خط ورودی دارند. CP نیز بعنوان یکی از ورودی‌های آنها خواهد بود.

چهار نوع FF را معرفی می‌کنیم که عموماً آنها دارای یکی از دو نوع مدار پایه نوری می‌باشند.

۱- مدار پایه براساس دروازه NOR ←

مشاهده می‌شود بعلت وجود فیدبک در این ساختار پایه، خود مدار پایه، یک مدار ترتیبی خواهد بود.
(این مدار از نوع آسنکرون می‌باشد.)

می دانیم: $x, y \rightarrow (x+y)' = x'y'$



R	S	Q	Q'
1	0	0	1
0	0	0	1
0	1	1	0
0	0	1	0
1	1	0	0

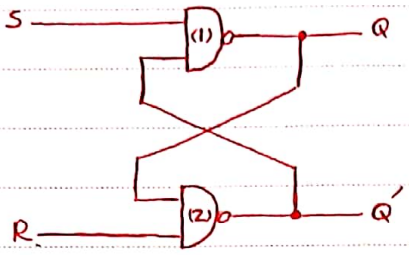
حالت — حالت — ورودی نامطلوب

زیرا Q و Q' مکمل یکدیگر نیستند

مدارهای عملگرد مدار پایه:
- $(R=1, S=0)$ ← شرایط reset ⇒ $Q=0, Q'=1$
- $(R=0, S=1)$ ← شرایط set ⇒ $Q=1, Q'=0$
- $(R=0, S=0)$ ← شرایط ابقاء ⇒ مقدار خروجی قبل (حالت) مدار
- $(R=1, S=1)$ ← شرایط غیرمجاز ⇒ -

۲ مدار پایه براساس دروازه NAND ←

می‌دانیم : $(x.y)' = (x'+y')$



R	S	Q	Q'
1	0	1	0
1	1	1	0
0	1	0	1
1	1	0	1
0	0	1	1

حالت

حالت

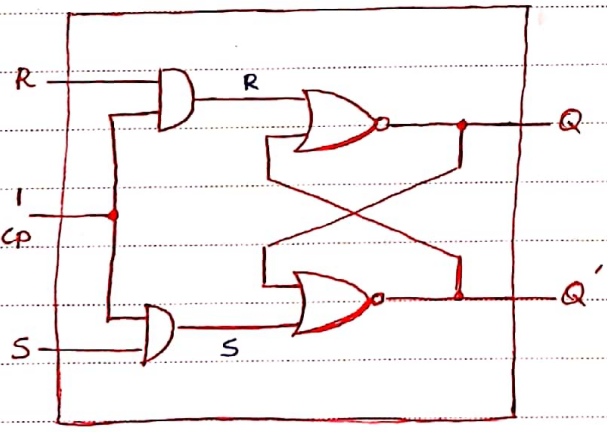
ورودی نامطلوب

زیرا Q و Q' مکمل یکدیگر نیستند.

مدهای عملکرد مدار پایه:
- $(R=1, S=0)$ → شرایط Set ⇒ $Q=1, Q'=0$
- $(R=1, S=1)$ → شرایط ابقاء ⇒ حفظ خروجی قبل (حالت) مدار
- $(R=0, S=1)$ → شرایط Reset ⇒ $Q=0, Q'=1$
- $(R=0, S=0)$ → شرایط غیرمجاز ⇒ —

RS-FF با پالس ساعت:

مدار RS-FF یک مدار ترتیبی سنکرون می‌باشد و عملکرد آن در صورت فعال بودن پالس ساعت ($CP=1$) دقیقاً مشابه مدار پایه براساس NOR می‌باشد.



با توجه به ($CP=1$) مدهای عملکرد RS-FF بصورت زیر است:

- $R=1, S=0$ → Reset
- $R=0, S=1$ → Set
- $R=0, S=0$ → ابقاء
- $R=1, S=1$ → غیرمجاز

Subject:
Year. 98 Month. 02 Date. 17 ( )

با فرض فعال بودن مدار RS-FF (ترگیر شدن توسط cp یعنی cp در حالتی باشد که مدار فعال باشد)
(در مدار رسم شده یعنی cp=1).

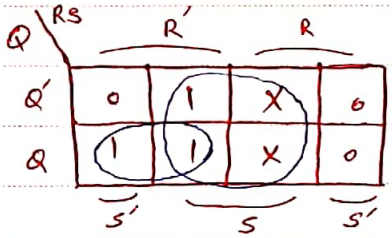
جدول عملکرد RS-FF بصورت زیر می‌باشد:

حالت فعلی	ورودی‌ها		cp=1 خروجی
Q(t)	R	S	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
→ 0	1	1	X
1	0	0	1
1	0	1	1
1	1	0	0
→ 1	1	1	X

معادله مشخصه RS-FF:

معادله مشخصه عبارت جبری است که خروجی را براساس ورودی‌ها حالت مدار بیان می‌کند.

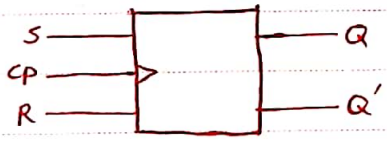
$$Q(t+1) = F_{(Q(t),R,S)} = ?$$



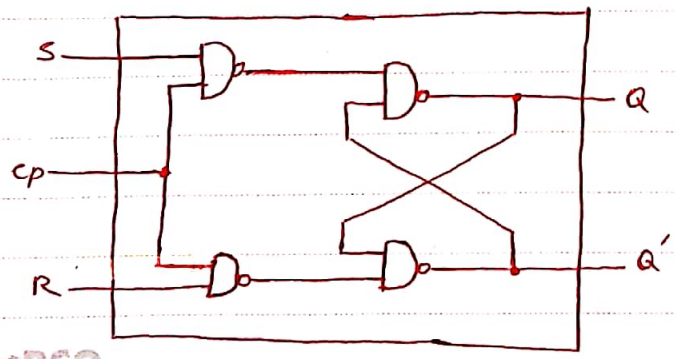
$$Q(t+1) = S + R'Q(t)$$

$$RS = 0$$

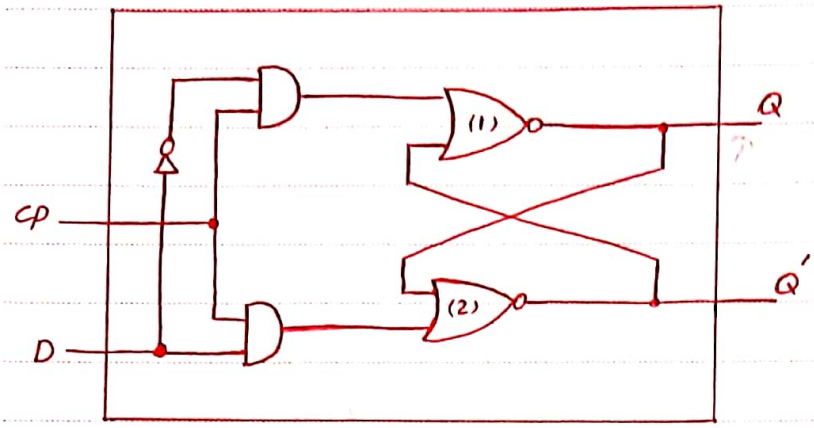
نماد مداری RS-FF بشکل زیر است:



RS-FF با پالس ساعت cp براساس NAND می‌توان رسم کرد.



D-FF بایاس ساعت:



تفاوت جدول عملکرد و مدارهای آن با RS-FF بایاس ساعت بصورت زیر است.

حالت فعلی	ورودی‌ها		خروجی
Q(t)	R	S	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	X
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	X

$\begin{cases} R=1 \\ S=0 \end{cases}$ → Reset

$\begin{cases} R=0 \\ S=1 \end{cases}$ → set

$\begin{cases} R=0 \\ S=0 \end{cases}$ → القاء

$\begin{cases} R=1 \\ S=1 \end{cases}$ → غیرمجاز

جدول عملکرد D-FF بصورت زیر است:

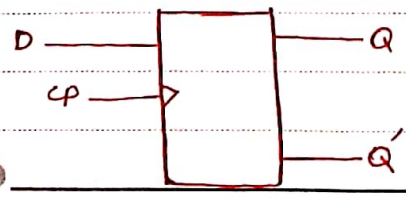
Q(t)	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

$\begin{cases} D=0 \rightarrow \text{Reset} \\ D=1 \rightarrow \text{set} \end{cases}$

معادله مشخصه D-FF:

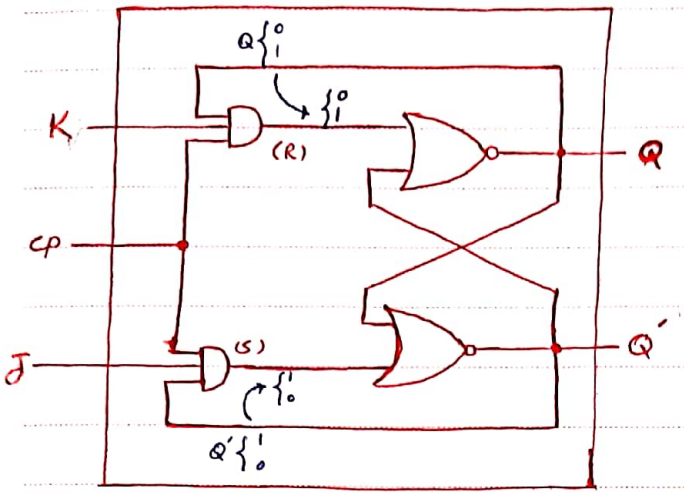
$$Q(t+1) = D$$

نماد مداری D-FF بشکل زیر است:



Subject:
Year. 98 Month. 02 Date. 22( )

JK-FF :



در RS-FF مشکل همزمان ! شدن ورودی‌ها بعنوان یک حالت نامطلوب وجود دارد که با انجام فیدبک از خروجی به ورودی‌های AND در مدار JK-FF این مشکل برطرف می‌گردد.

← در مدار JK-FF ، cp = 0 ، FF غیرفعال می‌کند و FF موقتی فعال است که cp = 1 باشد.

مدهای عملکرد آن بصورت زیر است :

ابقاء → J = 0, K = 0

Reset → J = 0, K = 1

Set → J = 1, K = 0

مکمل‌گیری → J = 1, K = 1

جدول عملکرد آن بصورت زیر است :

cp = 1

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

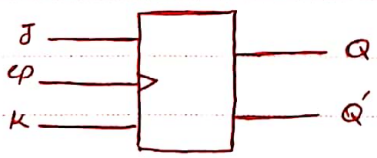
ابقاء → Q(t+1) = Q(t)

Reset → Q(t+1) = 0

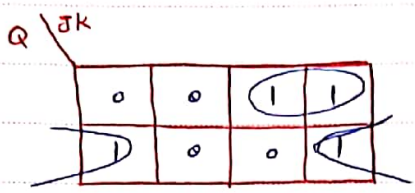
Set → Q(t+1) = 1

مکمل‌گیری → Q(t+1) = Q'(t)

نمادهای JK-FF بصورت زیر است :

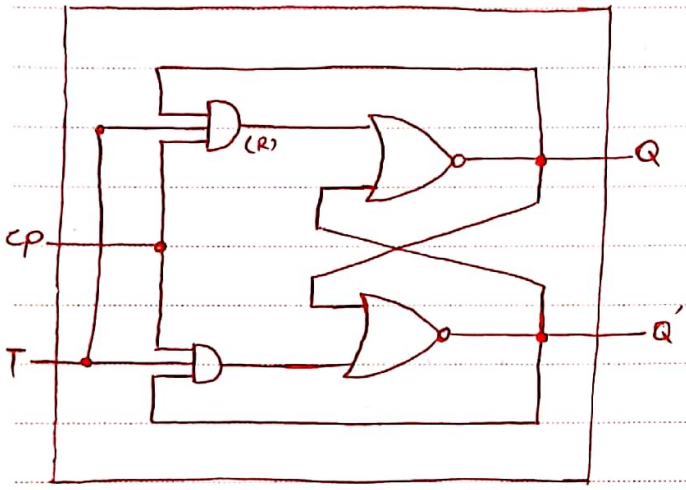


معادله مشخصه JK-FF :



$\Rightarrow Q(t+1) = \bar{Q}J + Q\bar{K}$

T-FF :



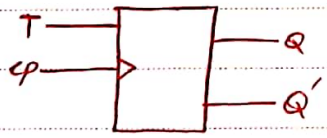
مدهای عملکرد آن بصورت زیر است :

$\begin{cases} T=0 \longrightarrow \text{ابقا} \\ T=1 \longrightarrow \text{مکمل} \end{cases}$

جدول عملکرد آن بصورت زیر است :

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

نماد مداری T-FF بصورت زیر است :



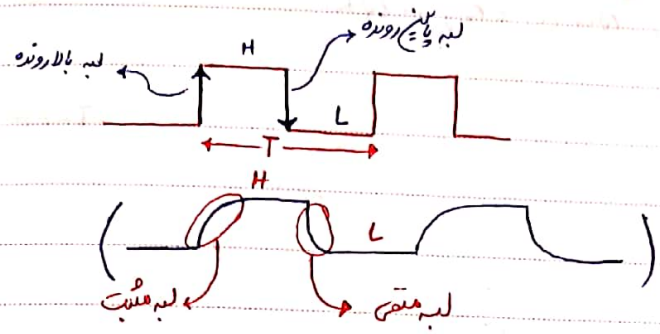
معادله مشخصه T-FF :

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

ترگیر کردن FF ها:

منظور از ترگیر کردن یک FF، آماده فعالیت عادی نمودن FF می باشد که توسط پالس ساعت مناسب اعمال می‌شود.

← cp



هر cp دارای دو سطح (level) و لبه می باشد:

سطح: 1) H (بالا) (مثبت) ← "1"
2) L (پایین) (منفی) ← "0"

هر cp دارای دو لبه (Edge) می باشد:

لبه: 1) بالارونده (لبه مثبت)
2) پایین رونده (لبه منفی)

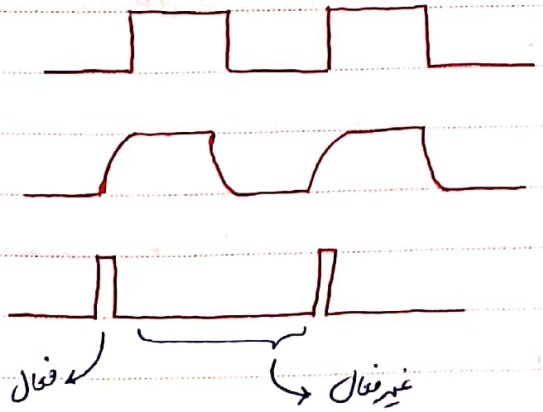
← ترگیر شدن FF در سطح در برخی موارد ممکن است مشکلاتی به همراه داشته باشد. (با توجه به کوچک بودن فاصله زمانی انتقال داده در FF نسبت به دوره فعال بودن FF در سطح)

مثلاً ← در مدار مکمل گیری در JK-FF با 1 ماندن ورودی‌ها در یک سطح، عمل مکمل گیری چندین بار اتفاق می‌افتد.

یا با تغییر چند باره داده‌های ورودی در یک سطح برای عملیات set یا Reset مشکل ایجاد خواهد شد.

← یکی از مهمترین کارها برای مقابله با مشکل فوق، ترگیر نمودن FF در لبه‌ها یا ی سطوح می باشد.

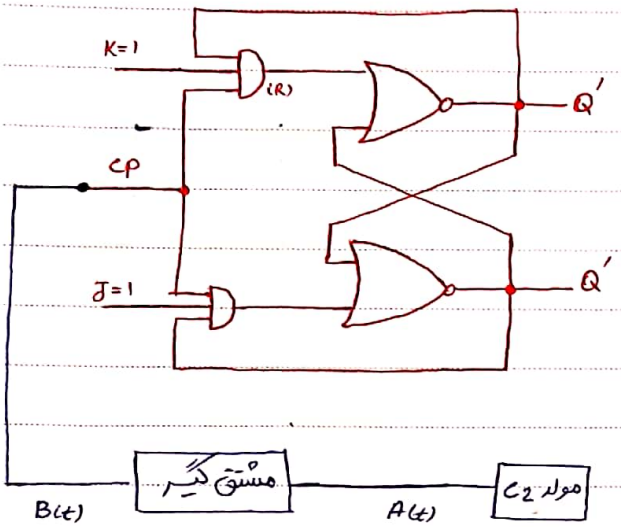
یعنی FF فقط برای یک زمان کوتاه (مدت لبه) فعال می‌شود و تا پالس بعدی غیرفعال خواهد ماند. به این ترتیب در هر دوره پالس ساعت فقط یک عمل مورد نظر انجام خواهد شد.

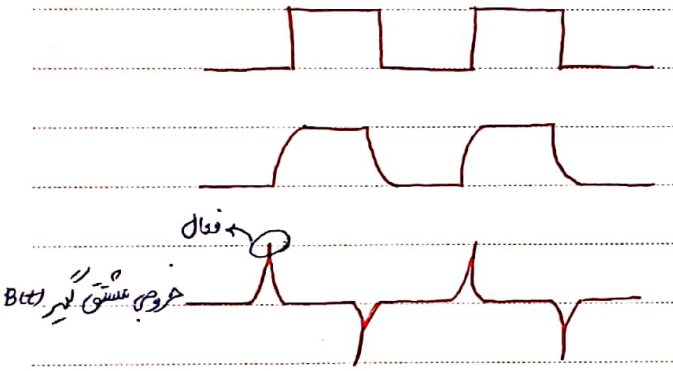


← برای ایجاد FF ترگیر شونده در لبه:

← تغییر در ساختار (اضافه نمودن مدارهای پایه‌ای برروی ورودی‌های مدار موجود FF ترگیر شونده در سطح)
(در کتاب مطالعه شود ☺)

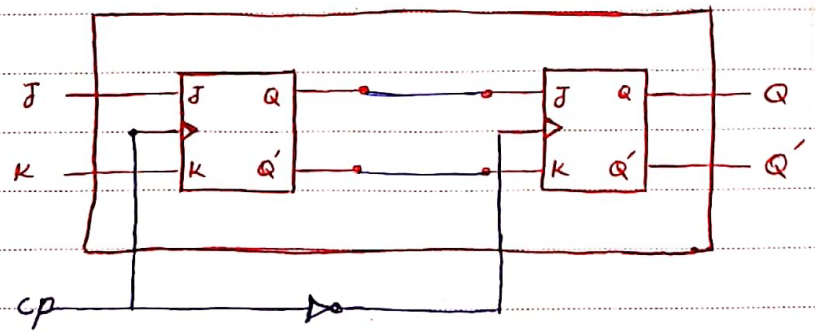
← قبل از اعمال cp به FF ترگیر شونده در سطح، سیگنال cp را از یک مشتق گیر عبور دهیم و خروجی مشتق گیر به cp، FF وصل شود.





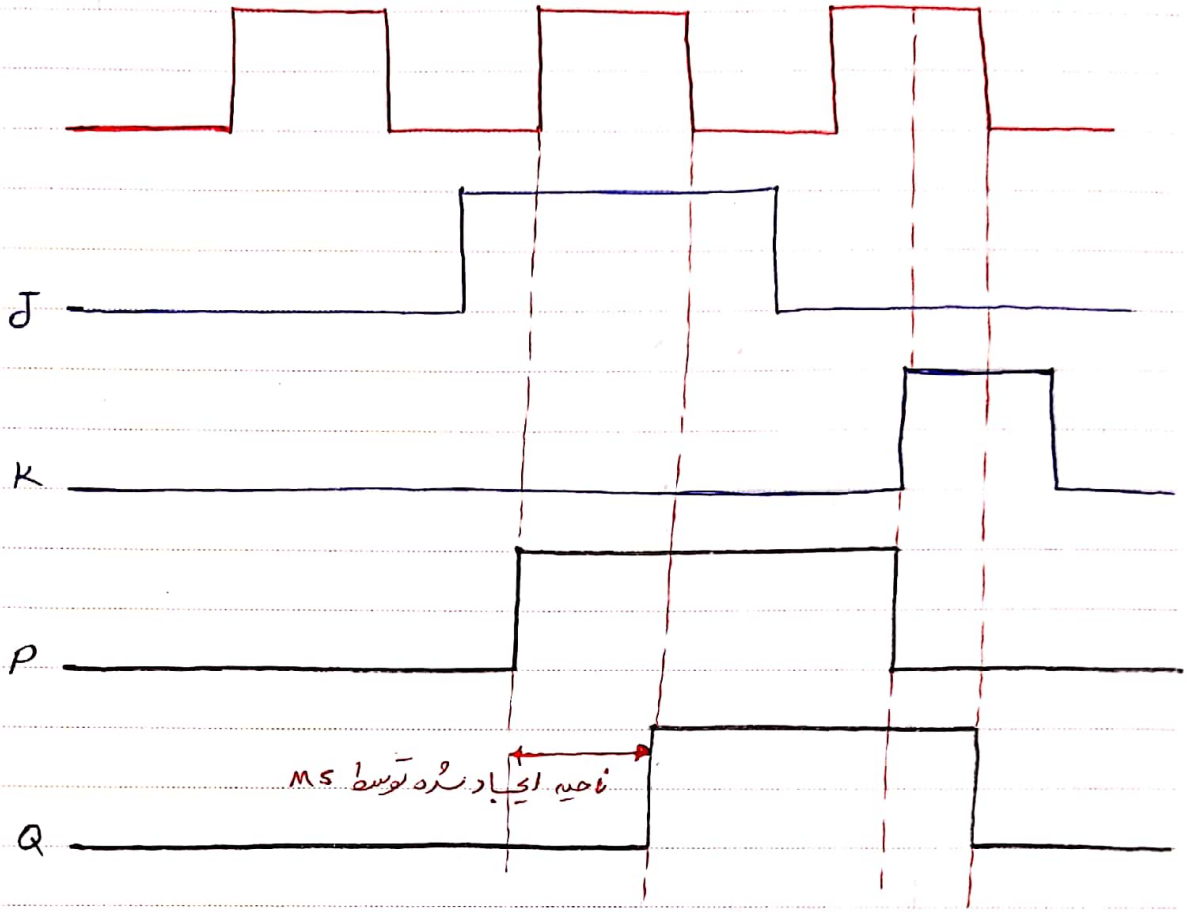
Master - Slave FF:

MS-FF ← یک ساختار متشکل از دو عدد FF که بصورت سری به یکدیگر وصل شده اند می باشد و تفاوتی که در این دو FF وجود دارد نحوه ترگیر شدن آنهاست.



مثال عملکرد MS-FF :

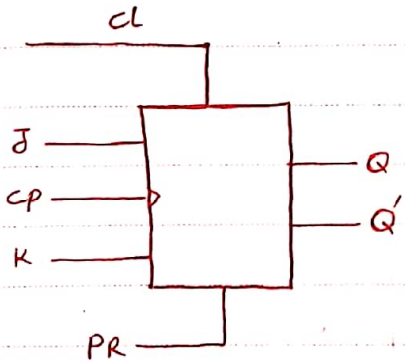
با فرض اینکه JK-MS-FF شامل دو عدد JK - FF فعال شونده در سطح مثبت باشد،



ورودی های آسنکرون (ناهمزمان) در FF ها:

ورودی های اصلی FF ورودی سنکرون می باشند که همراه با فعال cp، تأثیرشان در حوزه اعمال می شود و برخلاف این ورودی ها، FF معمولاً شامل ورودی های دیگری می باشند که بدون توجه به مقدار cp و به محض فعال شدن این ورودی ها، تأثیرشان در خروجی ظاهر می شود. به این ورودی ها، ورودی های آسنکرون می گویند که به دو نوع است:

- clean ← برای صفر کردن آسنکرون خروجی
- preset ← برای یک کردن آسنکرون خروجی

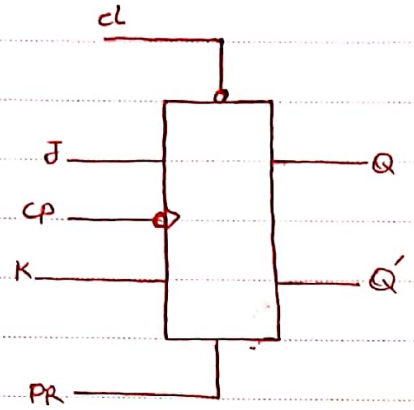


واضح است که یکی از موارد کاربرد این ورودی، انتقال FF به یک حالت اولیه دلخواه می باشد.

← اگر FF در ورودی آسنکرون داشته باشد یکی از آنها حتماً در اولویت است.

← این ورودی ها می توانند فعال ۰ یا ۱ باشند.

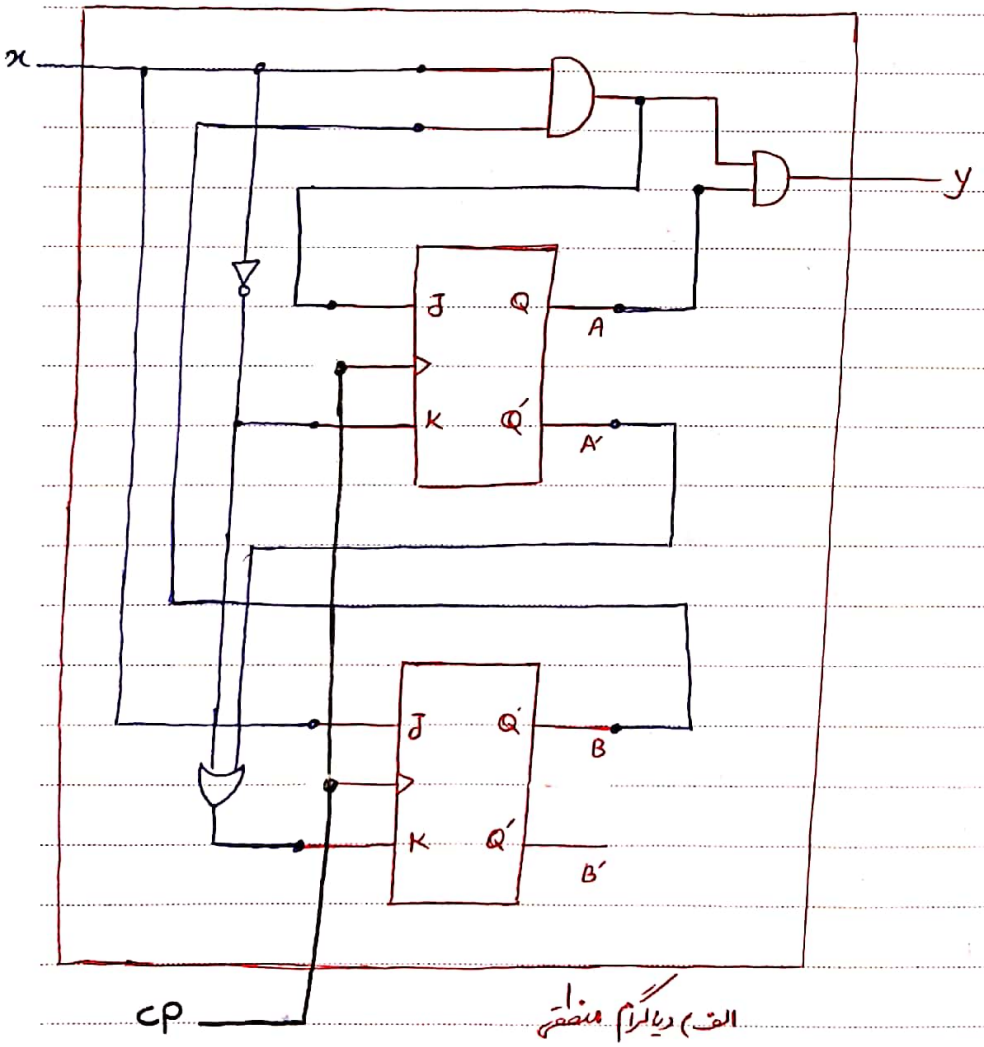
یک نمونه ورودی آسنکرون در FF و بصورت زیر است :



اولویت ↑		لبه منفی ↑			
$\overline{CL}$	PR	CP	J	K	عملکرد FF (خروجی)
0	X	X	X	X	clear to "0"
1	1	X	X	X	preset to "1"
1	0	⊓↓	0	0	ابهام
1	0	↑⊓↓	0	1	Reset
1	0	↓	1	0	set
1	0	↓	1	1	مکمل

تحلیل مدار های ترتیبی :

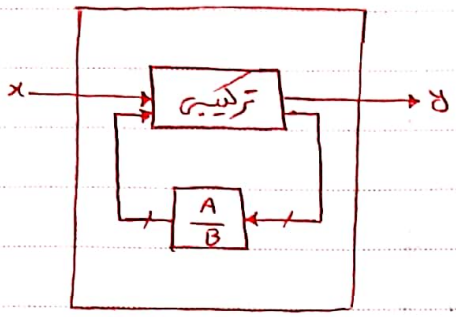
مدار شامل دو JK-FF با نام های $\underline{A}$ و $\underline{B}$ و یک ورودی x و یک خروجی y در نظر بگیرید.



الف) دیاگرام منطقی

Subject:
Year. 98 Month. 02 Date. 24 ( )

منظور از تحلیل یک مدار بررسی عملکرد مدار می‌باشد، یعنی اگر دیاگرام منطقی مدار داده شده باشد.
هدف از تحلیل، بدست آوردن روابط مابین ورودی‌ها، خروجی‌ها و حالات مدار
(حالات فعلی و بعدی) می‌باشد.



← حالت مدار براساس محتوای FF‌های مدار تعیین می‌گردد.
در مثال مورد بحث چون دو Jk-FF، A و B داریم تعداد حالت‌های مدار، 4 حالت خواهد بود.

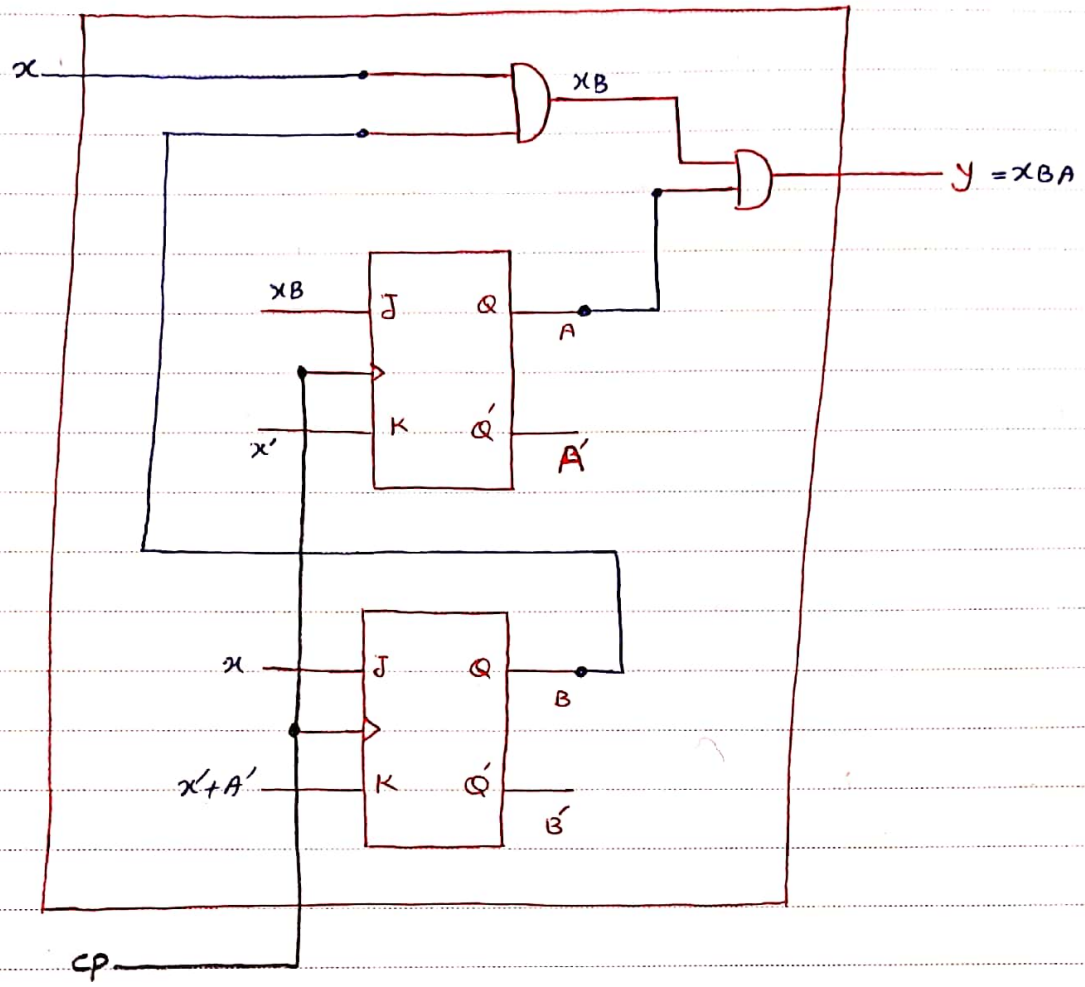
→ AB: {00, 01, 10, 11}

ب) جدول حالت:
خروجی از بخش ترکیبی
حالت بعدی FF‌ها بسته به نوع FF و اطلاعات ورودی به خطوط ورودی FF‌ها می‌باشد.
همچنین از قبل می‌دانیم:

عملکرد	K	J
ابقاء	0	0
Reset	1	0
Set	0	1
مکمل	1	1

ورودی	حالت فعلی		حالت بعدی		خروجی
x	A(t)	B(t)	A(t+1)	B(t+1)	y
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	1	0
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	1	1

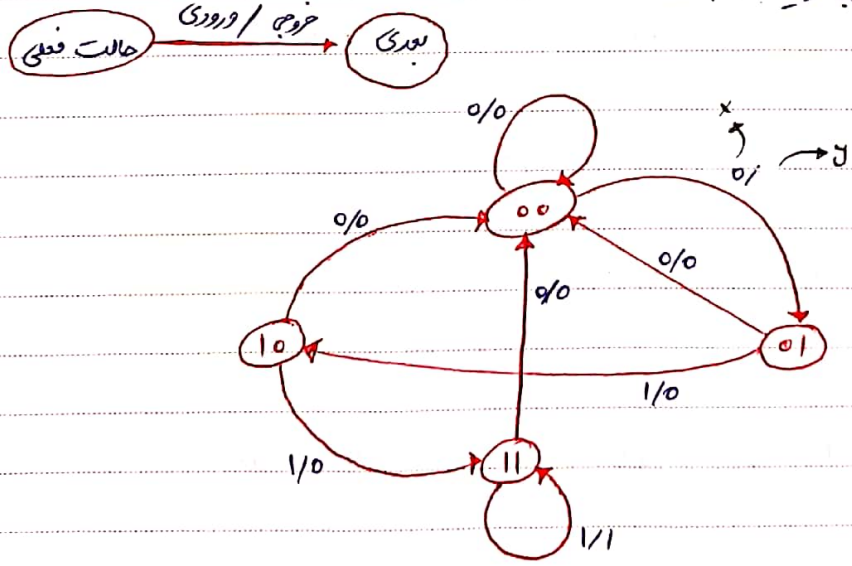
* دیاگرام منطقی ساده‌تر در صفحه بعد *



ج) دیاگرام حالت:

← برای هر حالت مدار، نقطه ای در صفحه در نظر می گیریم.

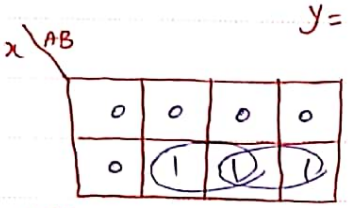
← از هر حالت برای هر ارزش ورودی، یک خط جهت دار خارج می کنیم که نشان دهنده حال حالت و ارزش خروجی تولیدی می باشد.



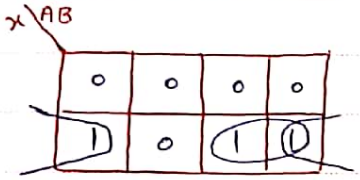
نکته:
اگر مداری n عدد FF، l عدد ورودی و k عدد خروجی داشته باشد.
l ← تعداد فلش‌های وارد شونده به هر حالت Q می‌توان تعیین کرد
2^l ← فلش از هر حالت خارج می‌شود
2^n ← حالت وجود دارد.

ب) معادلات حالت:
معادلاتی که خروجی، وضعیت بعدی FF، Q FF برحسب ورودی، وضعیت فعلی FF بیان می‌کنند.

$y = xAB$



$\longrightarrow A(t+1) = xA + xB$



$\longrightarrow B(t+1) = xA + xB'$

ج) توابع ورودی:
بیان عبارت جبری برای خروجی ورودی‌های FF برحسب ورودی مدار و وضعیت فعلی می‌باشد.

$y = xAB$

$$A\text{-FF} \begin{cases} J_A = xB \\ K_A = x' \end{cases} \quad \text{و} \quad B\text{-FF} \begin{cases} J_B = x \\ K_B = x' + A' \end{cases}$$

← ارتباط دهنده معادلات حالت و توابع ورودی، معادله مشخصه FF می‌باشد.

JK-FF ⟶ معادله مشخصه $Q(t+1) = JQ' + K'Q$

$A\text{-JK-FF} \longrightarrow A(t+1) = J_A A' + K_A' A \overset{\text{توابع ورودی}}{=\!=} (xB)A' + (x')'A = x(B+A) = xA + xB$

$\Rightarrow A(t+1) = xA + xB$

$B\text{-JK-FF} \longrightarrow B(t+1) = J_B B' + K_B' B = xB' + (x' + A')B = x(B' + A) = xA + xB'$

$\Rightarrow B(t+1) = xA + xB'$

عکس حالت قبل ←

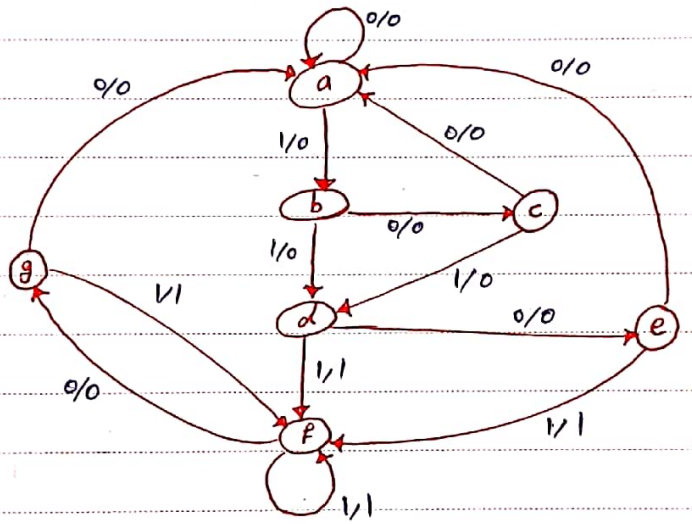
$$\begin{cases} A(t+1) = xA + xB = xA + xB(A+A') = (x+xB)A + xBA' \\ A(t+1) = J_A A' + K_{A'} A \end{cases}$$

$$\begin{cases} J_A = xB \\ K_{A'} = x + xB = x \end{cases} \longrightarrow \begin{cases} J_A = xB \\ K_A = x' \end{cases}$$

بررسی کاهش حالات:

← در برخی مسائل ممکن است حالات معادل وجود داشته باشد. دو حالت p معادل گویند اگر برای کلیه ورودی های یکسان، حالت بعدی یکسان داشته باشند.

در یک مسئله در صورت وجود دو حالت معادل می‌توان از یکی به جای دیگری استفاده نمود ← یک حالت کاهش می‌یابد



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

با تقریب جدول حالت با g مشاهده می‌شود $e \equiv g$ و $d \equiv f$ است در نتیجه حالت معادل با حذف دو سطر آخر جدول با g و تبدیل g به e و f به d بدست می‌آید. (صفحه بعد)

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

احتصاص مقادیر باینری به حالات بیان شده بصورت پارامتری :

احتصاص مقادیر باینری براساس تعداد FF های مورد نظر مدار برای حالات بطور دلخواه انجام می پذیرد.

جدول تحریک FF ها :

جدول تحریک RS-FF ←

تغییر حالت $Q(t)$	تغییر حالت $Q(t+1)$	ورودی لازم R	ورودی لازم S
0	0	x	0
0	1	0	1
1	0	1	0
1	1	1	x

می دانیم : ⟿

R	S	عملکرد
1	0	set
1	1	ابقاء
0	1	set
0	0	نامطلوب

جدول تحریک JK-FF ←

$Q(t)$	$Q(t+1)$	K	J	
0	0	x	0	ابقاء / Reset
0	1	x	1	set مکمل
1	0	1	x	Reset مکمل
1	1	0	x	ابقاء / set

می دانیم : ⟿

K	J	عملکرد
0	0	ابقاء
0	1	set
1	0	Reset
1	1	مکمل

جدول تحریک D-FF ←

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

می دانیم : ⟿

D	عملکرد
0	Reset
1	set

جدول تحریک T-FF

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

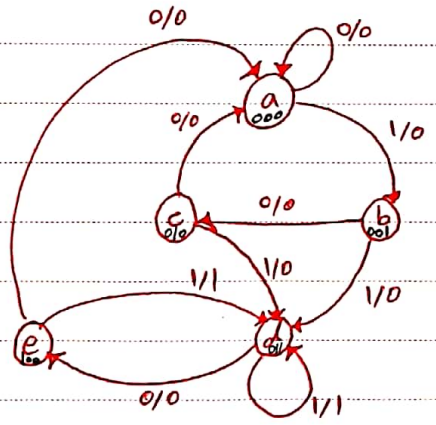
عملکرد	T
ابقاء	0
مکمل	1

می‌دانیم :

روند طراحی مدارهای ترتیبی :

1. شناخت دقیق مسئله
2. تعیین ورودی‌ها و خروجی‌ها و حالات (و تعداد FF‌ها) مدار و نام گذاری
3. تعیین نوع FF‌های مدار (درصورتی عدم تعیین توسط مسئله)
4. جدول عملکرد مدار (جدولی که رابطه ورودی‌ها و حالات فعلی با خروجی‌ها و حالات بعدی Q نهایی می‌دهد.)
5. بررسی مسئله کاهش حالات مدار (وجود حالات معادل)
6. اضافه نمودن جدول تحریک به جدول عملکرد براساس نوع FF‌ها و تعیین حالات
7. ساده سازی توابع خروجی اصلی و توابع ورودی FF‌ها. (با در نظر گرفتن امکانات مسئله)
8. پیاده سازی مدار

مثال : مطلوبست طراحی مداری که دارای دیاگرام حالت روبرو باشد :



- مدار یک ورودی x و یک خروجی y دارد
- مدار دارای 5 حالت است $\Rightarrow$ 3 عدد FF لازم داریم.
- از سه عدد JK-FF با نام‌های A و B و C استفاده می‌کنیم.
- اختصاص مقادیر باینری به حالات مدار : (با توجه به 3 عدد FF موجود در مدار، هر حالت با سه بیت مشخص می‌شود)
- بررسی کاهش حالات : حالت معادل نداریم

$$\begin{cases} a \longrightarrow 000 \\ b \longrightarrow 001 \\ c \longrightarrow 010 \\ d \longrightarrow 011 \\ e \longrightarrow 100 \end{cases}$$

ورودی	حالت فعلی				حالت بعدی			خروجی	(A)		(B)		(C)	
x	A	B	C		$A_{(t+1)}$	$B_{(t+1)}$	$C_{(t+1)}$	y	K_A	J_A	K_B	J_B	K_C	J_C
0	0	0	0	(a)	0	0	0 (a)	0	X	0	X	0	X	0
1	0	0	0		0	0	1 (b)	0	X	0	X	0	X	0
0	0	0	1	(b)	0	1	0 (c)	0	X	0	X	1	1	X
1	0	0	1		0	1	1 (d)	0	X	0	X	1	0	X
0	0	1	0	(c)	0	0	0 (a)	0	X	0	1	X	X	0
1	0	1	0		0	1	1 (b)	0	X	0	0	X	X	1
0	0	1	1	(d)	1	0	0 (c)	0	X	1	1	X	1	X
1	0	1	1		0	1	1 (d)	1	X	0	0	X	0	X
0	1	0	0	(e)	0	0	0 (a)	0	1	X	X	0	X	0
1	1	0	0		0	1	1 (b)	1	1	X	X	1	X	1

مقایسه دوستون A و $A_{(t+1)}$ برای محاسبه K_A و J_A

مقایسه دوستون B و $B_{(t+1)}$ برای محاسبه K_B و J_B

مقایسه دوستون C و $C_{(t+1)}$ برای محاسبه K_C و J_C

← رسم جدول تحریک (جدول بالا)

← ساده سازی توابع ورودی FF و توابع خروجی مدار (مقادیر فلش دار در جدول بالا)
نکته مهم در این حالت پیدا کردن مقادیر حالت Dontcare است.

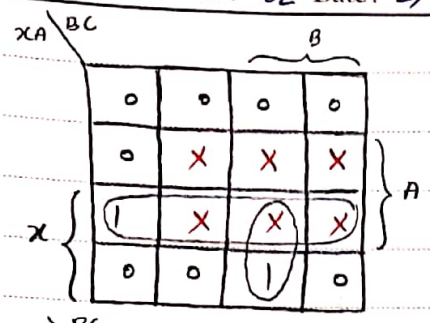
تعیین ترکیبات D-C. ⇐

حالاتی موجود نیست

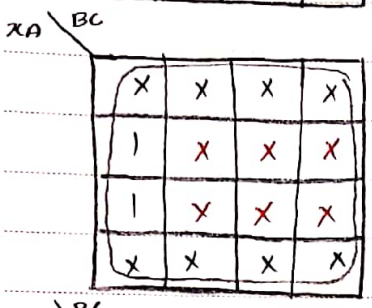
	x	A	B	C	
D-C. حالات	0	1	0	1	5
	1	1	0	1	13
	0	1	1	0	6
	1	1	1	0	14
	0	1	1	1	7
	1	1	1	1	15

* حالات D-C. در جدول کارنو با رنگ قرمز مشخص شده است *

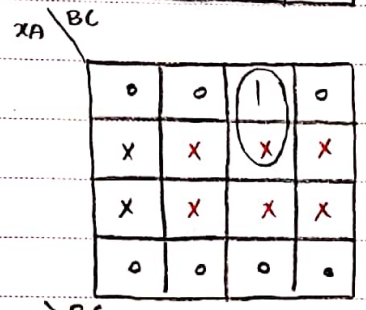
Subject:
Year. 98 Month. 02 Date. 29 ( )



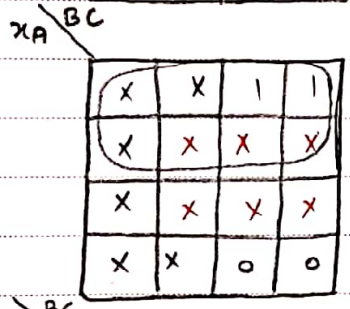
$\Rightarrow y = xA + xBC$



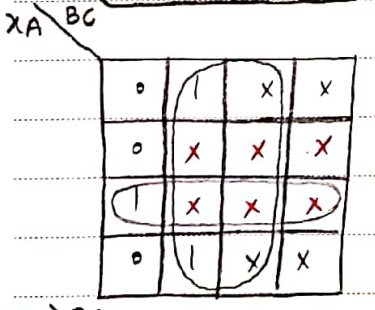
$\Rightarrow K_A = 1$



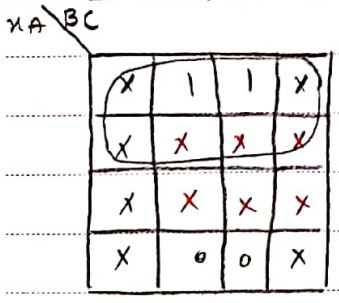
$\Rightarrow J_A = x'BC$



$\Rightarrow K_B = x'$



$\Rightarrow J_B = C + xA$

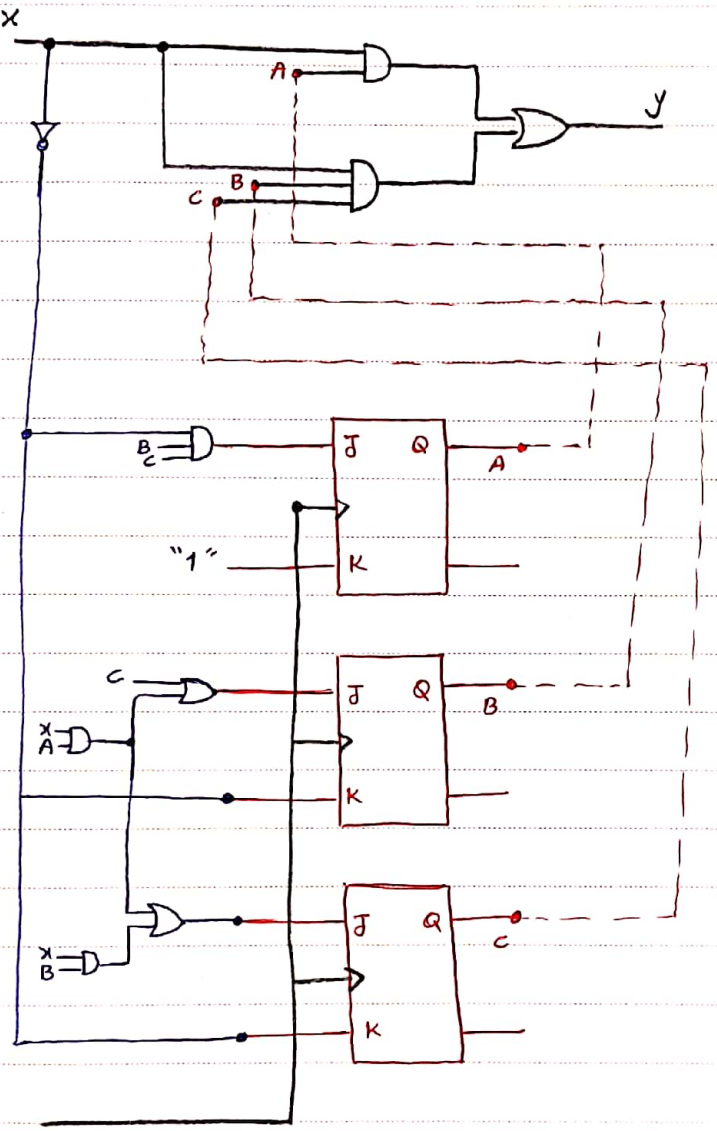


$\Rightarrow K_C = x'$

xA \ BC	00	01	11	10
0	X	X	X	0
0	X	X	X	X
1	X	X	X	X
0	X	X	X	1

$\Rightarrow J_C = xA + xB$

پیاده سازی مدار



شمارنده (Counters) :

منظور از یک شمارنده، یک مدار ترتیبی است که ما بین تعدادی حالات مشخص بطور چرخشی تغییر حالت می دهد و هر انتقال حالت با اعمال پالس تحریک کننده (پالس ساعت) اتفاق می افتد.

(در شمارنده ها، خروجی ها همان خروجی FF ها ی مدار می باشند.)

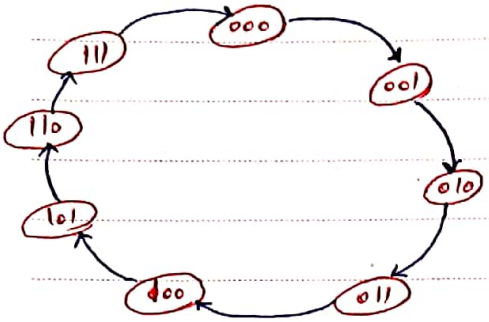
دنباله شمارشی یک شمارنده :

دنباله ای توسط معادل دهدهی حالات شمارنده بیان می شود.

Subject:
Year. 98 Month. 02 Date. 29 ( )

طراحی یک شمارنده باینری 3 بیتی :

برای مثال شمارنده ای که دارای دیاگرام حالت بصورت بورو باشد :



نکته ← دنباله شمارشی یک شمارنده : دنباله ای توسط معادل دهدهی حالات شمارنده بیان می شود.

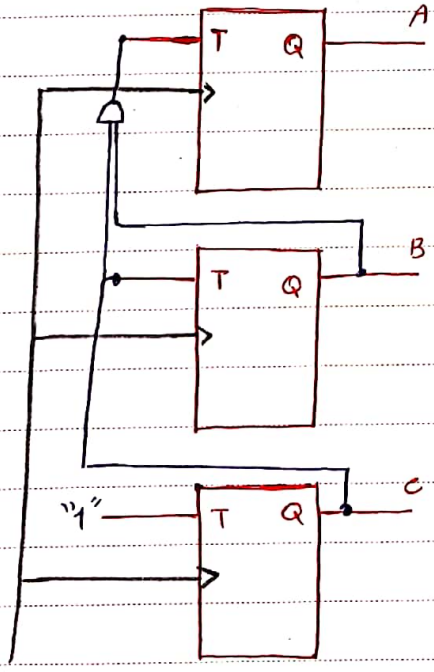
دنباله شمارشی ← 0 → 1 → 2 → 3 → 4 → 5 → 6 → 7

سه عدد T-FF با نام های A و B و C در نظر می گیریم :

A	B	C	T_A	T_B	T_C
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	0	0	1
1	1	1	1	1	1

$T_A = BC$ ، $T_B = C$ ، $T_C = 1$

(با استفاده از جدول کارنو یا دهنی)



Subject:
Year. 98 Month. 02 Date. 29 ( )

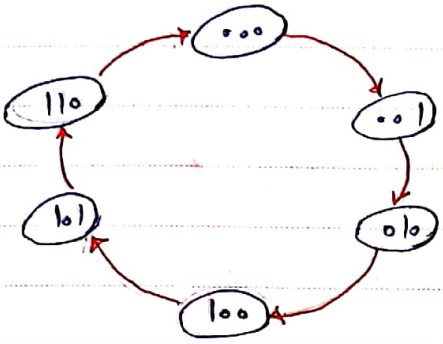
طراحی شمارنده‌ای که دارای دنباله شمارشی زیر باشد:

0 → 1 → 2 → 4 → 5 → 6 → 0

← مدار به 3 عدد FF نیاز دارد. (از نوع JK و با نام‌های A، B و C)

و می‌دانیم:

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



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

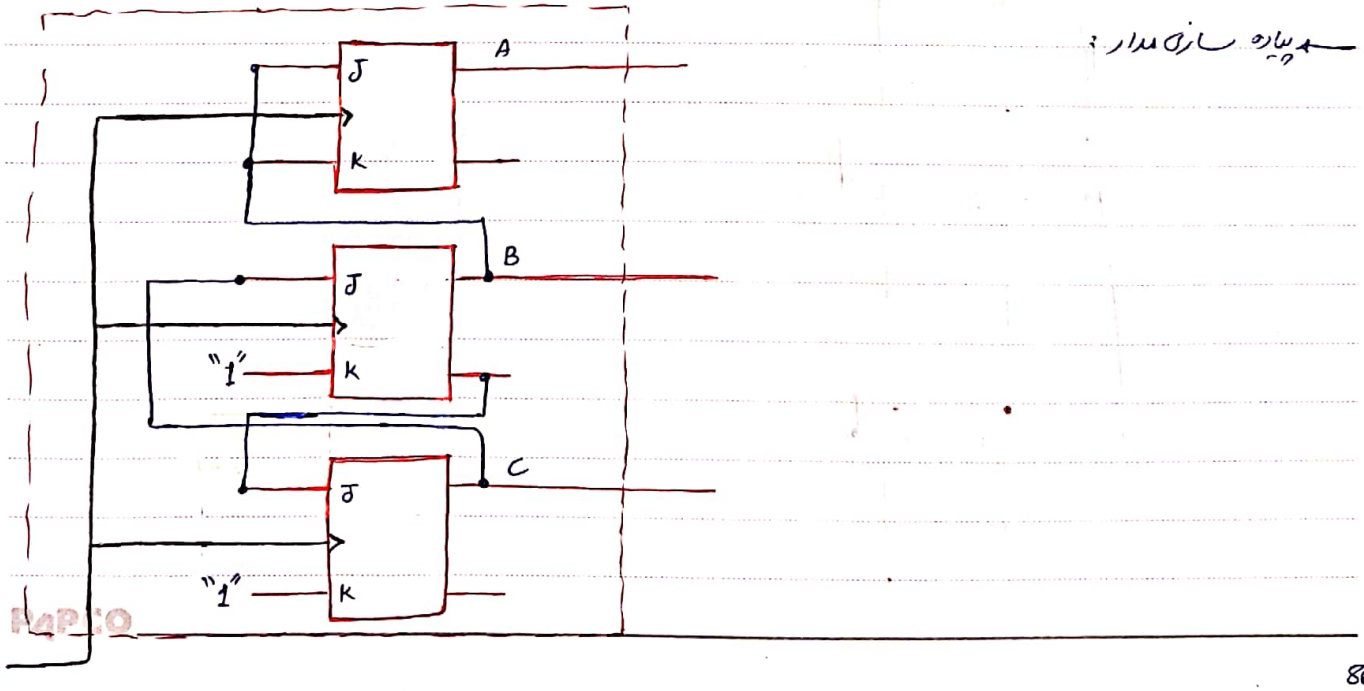
حالات D.C. { 011 , 110 }

$J_A = B$, $K_A = B$

$J_B = C$, $K_B = 1$

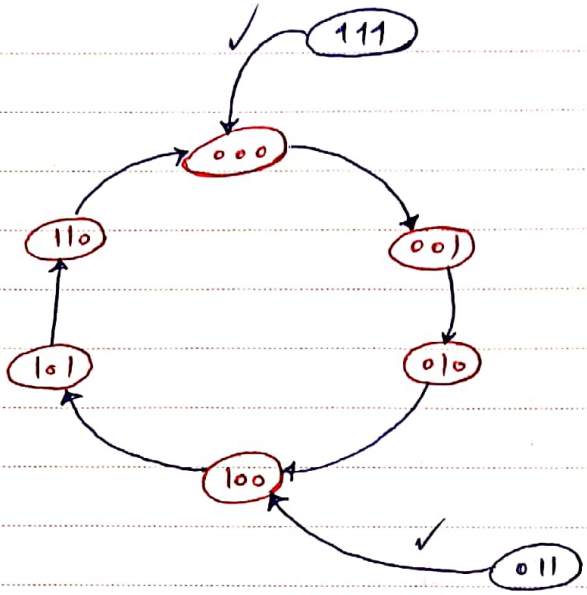
$J_C = B'$, $K_C = 1$

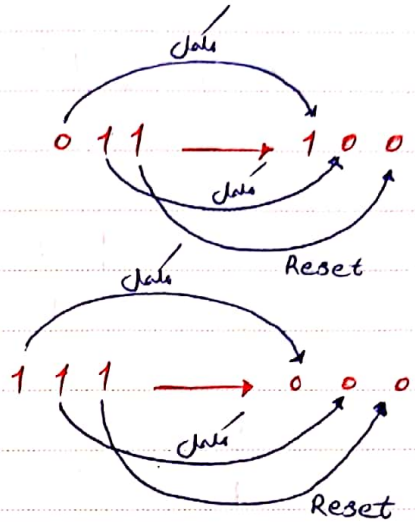
← پیاده سازی مدار:



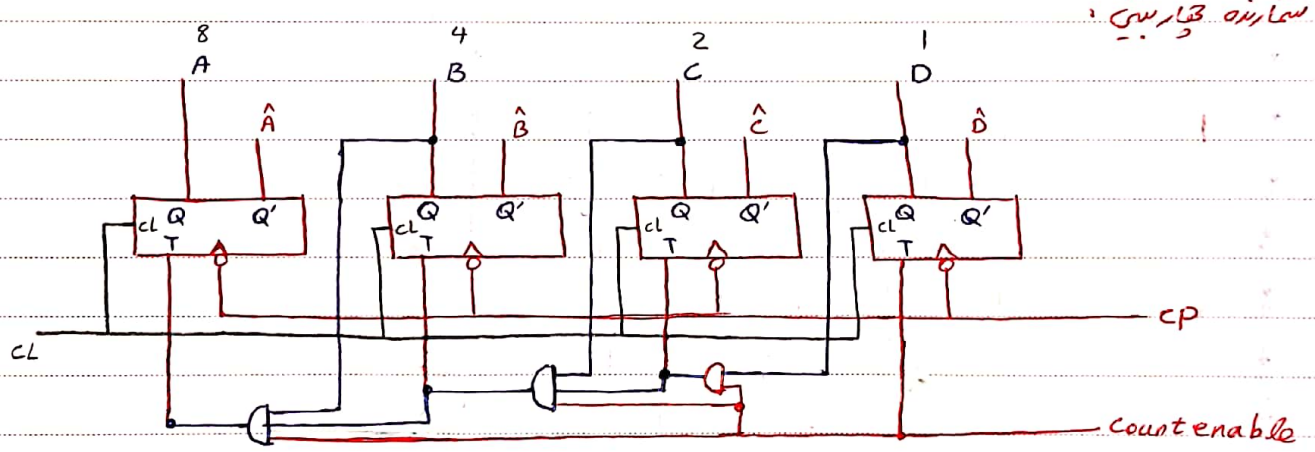
خودآغاز بودن شمارنده:

شمارنده ای که اگر وارد یکی از حالات نامطلوب گردد، بعد از یک (یا چند) پالس ساعت وارد حالات مطلوب شده و جریان عادی شمارنده برقرار شود.





شمارنده چهاربیتی:



- در شمارنده سنکرون CP بطور یکسان به ورودی CP تمام FF ها وارد می‌شود.
- در شمارنده ها، ترگر شدن FF توسط CP در لبه ها انجام می‌گیرد.
- دنباله شمارش آن بصورت نیز است:

- در شمارنده، FF ها "مجموعه" دارای ورودی های آسنکرون clear (cl) و/یا preset برای برون شمارنده به یک حالت اولیه دلخواه می باشند.

← برخی شمارنده ها دارای ورودی های کنترلی برای تحت کنترل گرفتن شروع شمارش شمارنده می باشند که تحت عنوان count enable که با فعال شدن این خط کنترلی یک شمارنده آماده کار، شروع به شمارش می نماید

ترریگر شده

count enable { 1 → شروع شمارش
0 → توقف شمارش

	A	B	C	D	Â	B̂	Ĉ	D̂	
0	0	0	0	0	1	1	1	1	15
1	0	0	0	1	1	1	1	0	14
2	0	0	1	0	1	1	0	1	13
3	0	0	1	1	1	1	0	0	12
4	0	1	0	0	1	0	1	1	11
5	0	1	0	1	1	0	1	0	10
6	0	1	1	0	1	0	0	1	9
7	0	1	1	1	1	0	0	0	8
8	1	0	0	0	0	1	1	1	7
9	1	0	0	1	0	1	1	0	6
10	1	0	1	0	0	1	0	1	5
11	1	0	1	1	0	1	0	0	4
12	1	1	0	0	0	0	1	1	3
13	1	1	0	1	0	0	1	0	2
14	1	1	1	0	0	0	0	1	1
15	1	1	1	1	0	0	0	0	0

← شمارنده رسم شده در سمت چپ جدول فوق افزایشی می باشد.
← شمارنده کاهشی بصورت زیر می باشد.

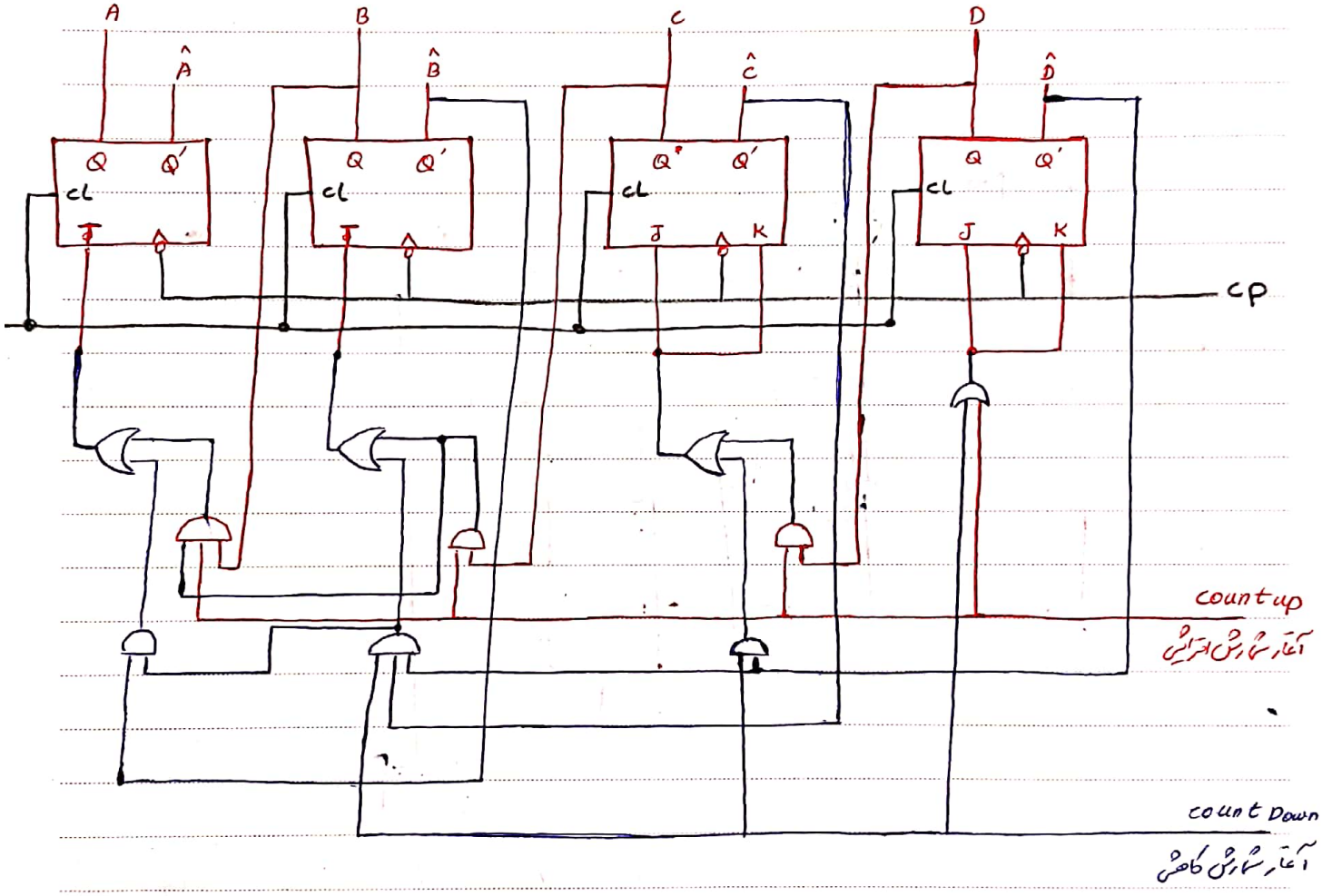
15 → 14 → ... → 2 → 1

(طبق ستون سمت راست جدول)

شمارنده با دو خط up, Down } enable :

می خواهیم شمارنده بصورت زیر عمل کند:

up	Down	عملکرد
0	1	شمارش افزایشی
1	0	شمارش کاهش
0	0	توقف شمارش

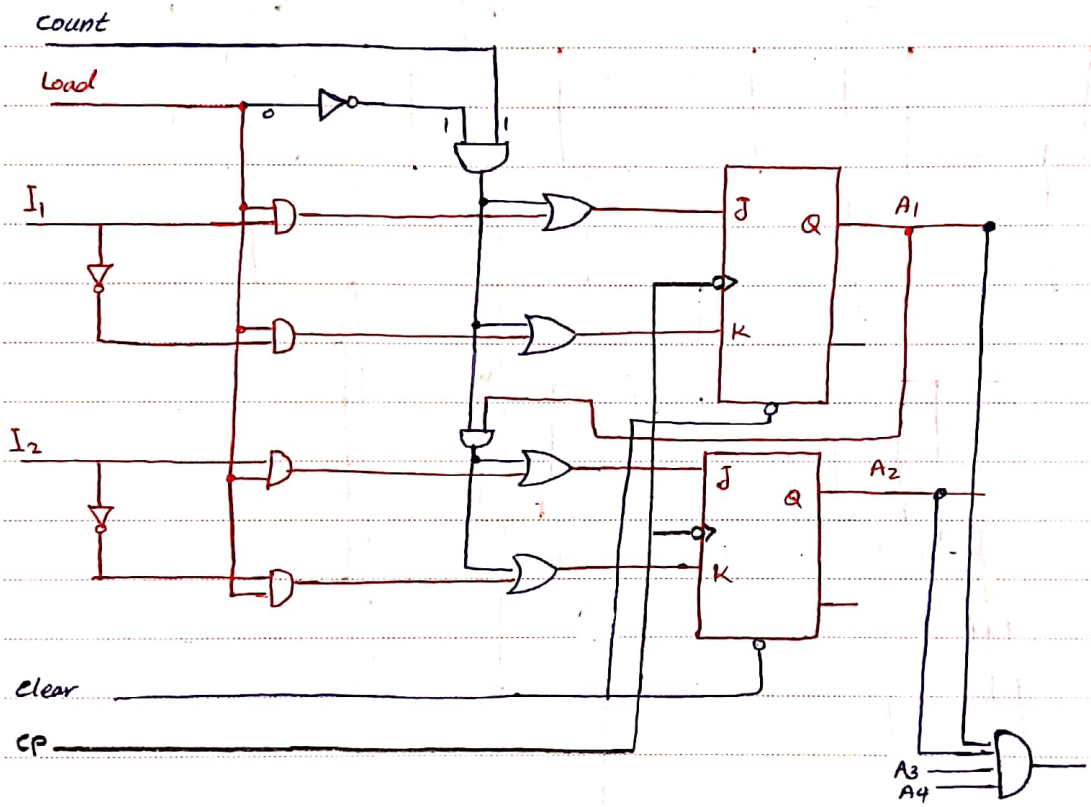


شمارنده دودویی (باینری) با امکان بارشدن (load) موازی:

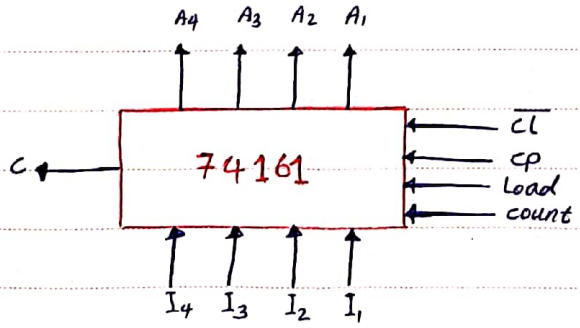
می خواهیم به شمارنده امکان بارشدن موازی e اضافه نماییم، بدین مفهوم که با مثال خط کنترل load در شمارنده اطلاعات باینری که بروی خطوط ورودی موازی شمارنده وجود دارد به داخل شمارنده بارگذاری شود.

خط load یک خط سنکرون شمارنده می باشد.

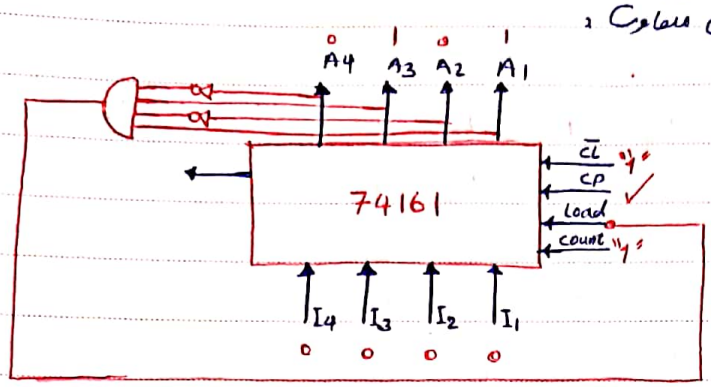
$\overline{cl}$	Load	cp	count	عملکرد
0	x	x	x	clear to "0"
1	1	↓	x	parallel loading : $I_i \longrightarrow A_i$
1	0	↓	1	شمارش
1	0	↓	0	توقف

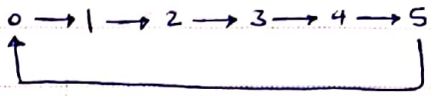


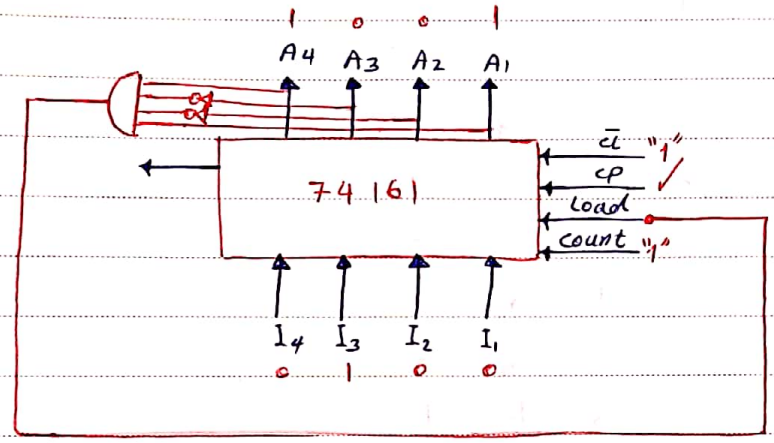
طرح (طراحی) شمارنده با N - mod و از یک مقدار اولیه دلخواه :
↳ منظور یک شماره N تایی

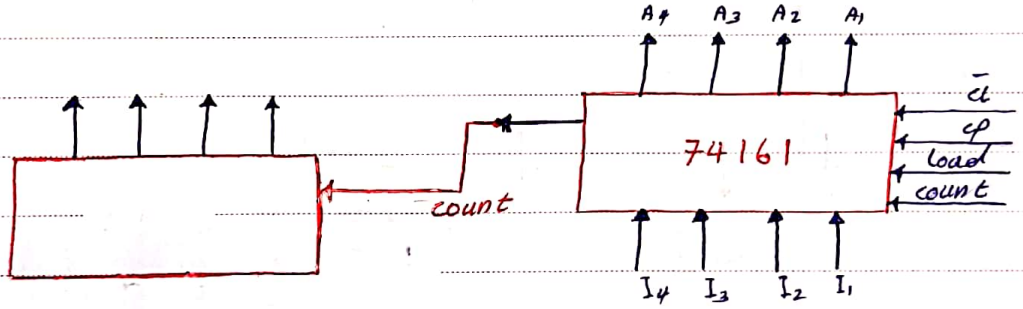


چند نمونه طراحی شمارنده Mod-6 یا شمارنده با دنباله‌های شمارشی متفاوت :



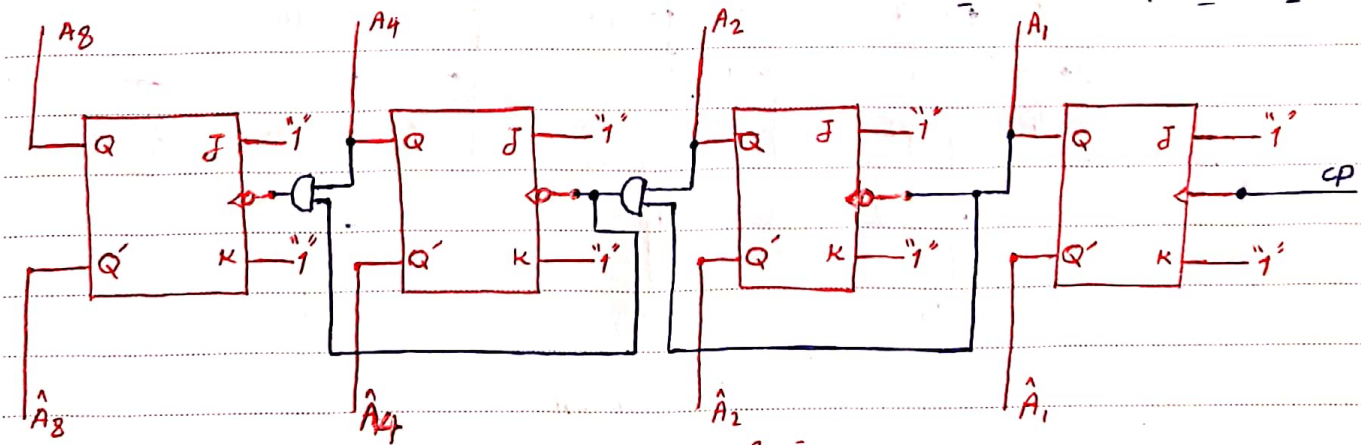






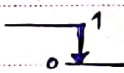
شمارنده های آسنکرون ، شمارنده های موج گونه (Ripple counter)

در این شمارنده لزوماً تمام FF ها، خط CP آنها از یک سمت مدار وصل نیست و سیگنال کلاک برای تریگر شدن هر یک از سیگنال دیگر موجود در مدار می گیرند.

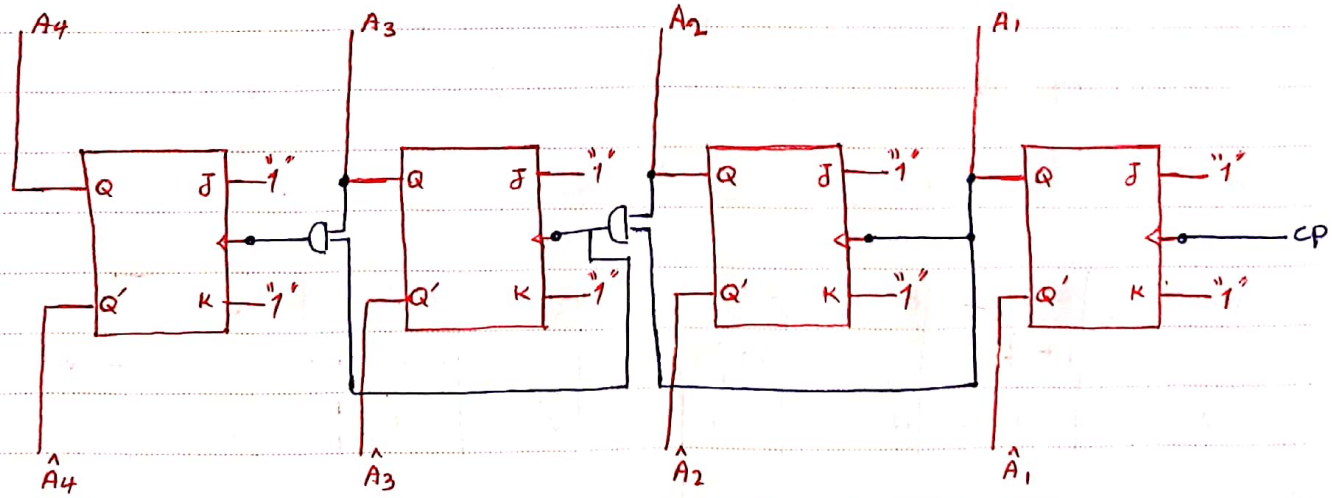


« ادامه دارد »

	A_8	A_4	A_2	A_1	$\hat{A}$	$\hat{B}$	$\hat{C}$	$\hat{D}$
0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	0
2	0	0	1	0	1	1	0	1
3	0	0	1	1	1	1	0	0
4	0	1	0	0	1	0	1	1
5	0	1	0	1	1	0	1	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	0	1	1	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	0	1
11	1	0	1	1	0	1	0	0
12	1	1	0	0	0	0	1	1
13	1	1	0	1	0	0	1	0
14	1	1	1	0	0	0	0	1
15	1	1	1	1	0	0	0	0



FF در لبه‌ی منفی پالس ترتیگر می‌شود

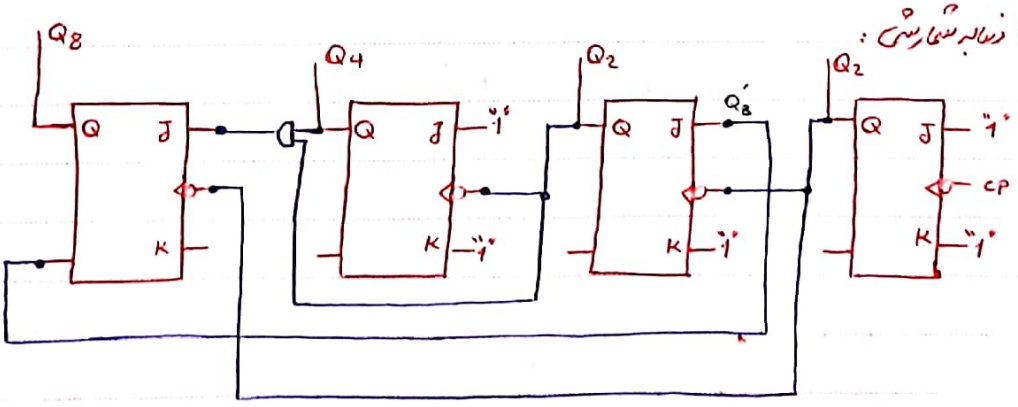


"ناهمگن"

	A_8	A_4	A_2	A_1	$\hat{A}$	$\hat{B}$	$\hat{C}$	$\hat{D}$
0	0	0	0	0	1	1	1	1
1	0	0	0	1	1	1	1	0
2	0	0	1	0	1	1	0	1
3	0	0	1	1	1	1	0	0
4	0	1	0	0	1	0	1	1
5	0	1	0	1	1	0	1	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	0	1	1	1
9	1	0	0	1	0	1	1	0
10	1	0	1	0	0	1	0	1
11	1	0	1	1	0	1	0	0
12	1	1	0	0	0	0	1	1
13	1	1	0	1	0	0	1	0
14	1	1	1	0	0	0	0	1
15	1	1	1	1	0	0	0	0

مکمل

مکمل



نحوه به هم وصل کردن دنباله شمارشی :

استفاده از روش سنتی آنالیز که در آن لازم است تا عملکرد هرکدام از FF ها مشخص نماییم.

یعنی : FF چه موقع تریگر می‌شود.

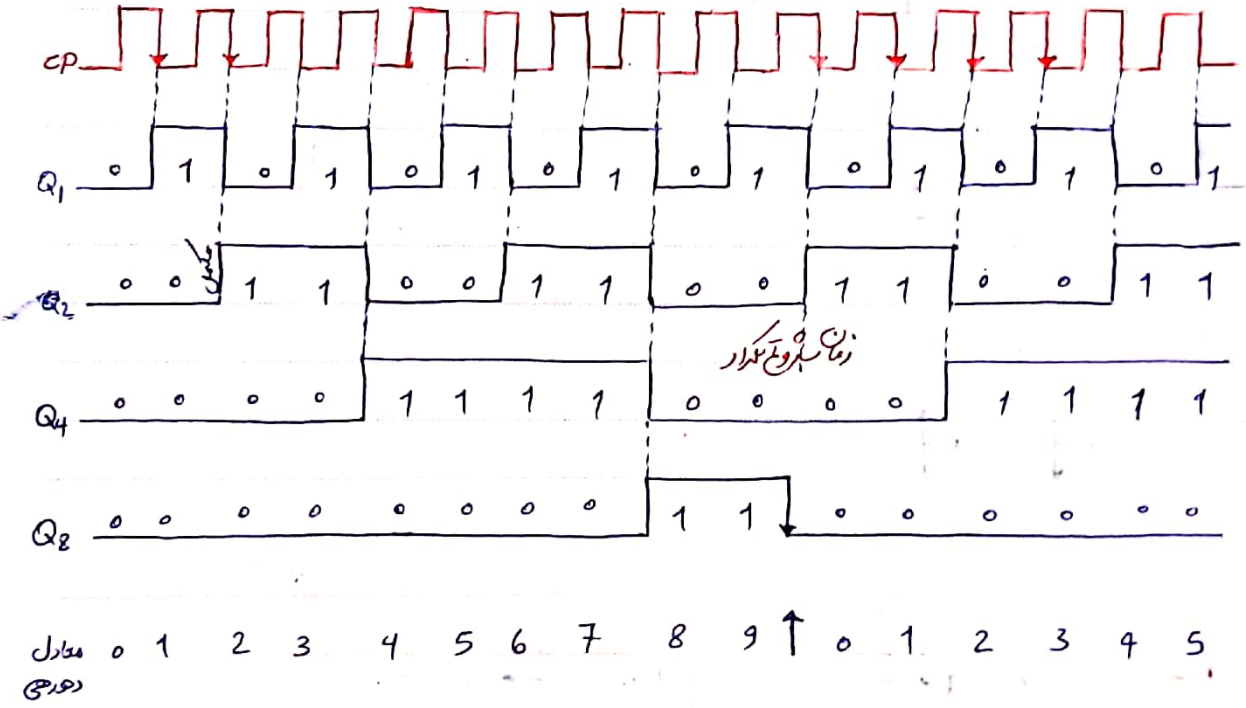
و وقتی تریگر شد، چه کاری انجام می‌دهد.

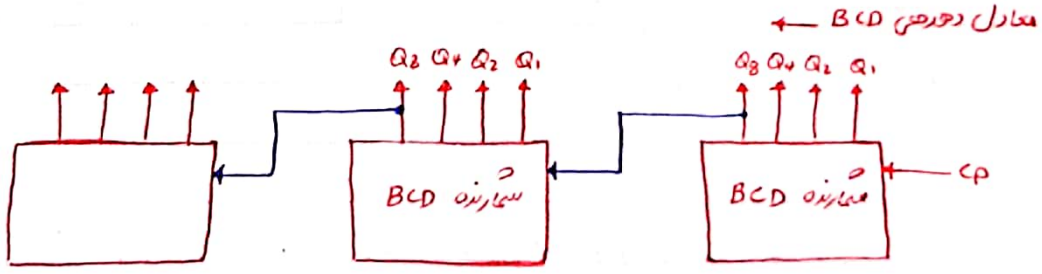
عملکرد Q_1 ← در لبه پایینی رونده پالس ساعت تریگر می‌شود ← مکمل گیری

عملکرد Q_2 ← وقتی تریگر می‌شود که Q_1 از 1 ← 0 منتقل شود. ←
- اگر $Q_8 = 0$ ← Q_2 مکمل شود.
- اگر $Q_8 = 1$ ← Q_2 می‌شود Reset.

عملکرد Q_4 ← وقتی تریگر می‌شود که Q_2 از 1 ← 0 منتقل شود. ← مکمل شدن

عملکرد Q_8 ← وقتی تریگر می‌شود که Q_1 از 1 ← 0 منتقل شود. ←
- اگر $Q_2Q_4 = 0$ ← Q_8 می‌شود Reset.
- اگر $Q_2Q_4 = 1$ ← Q_8 مکمل می‌شود.

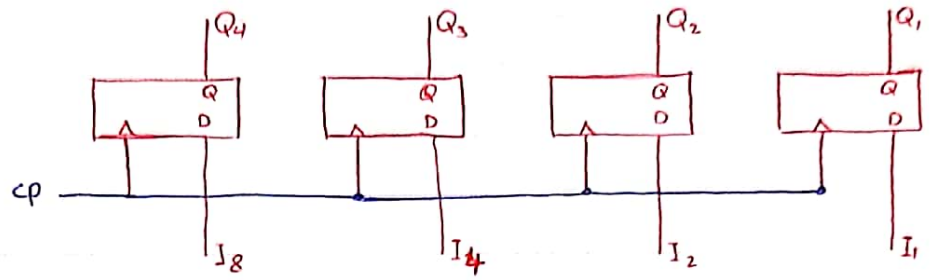


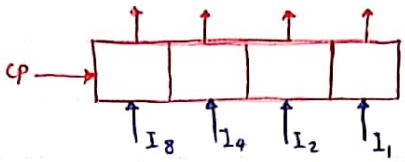


ثبات ها (Register) :

ثبات ها حافظه های نگهدارنده موقت داده ها باینری می باشند. ثبات ها که بصورت بلوک های نگهدارنده چندبیتی (تعداد بیت word سیستم مورد نظر) می باشند، برای نگهداری هر بیت به یک FF نیاز دارد.

ثبات چهار بیتی با D-FF :



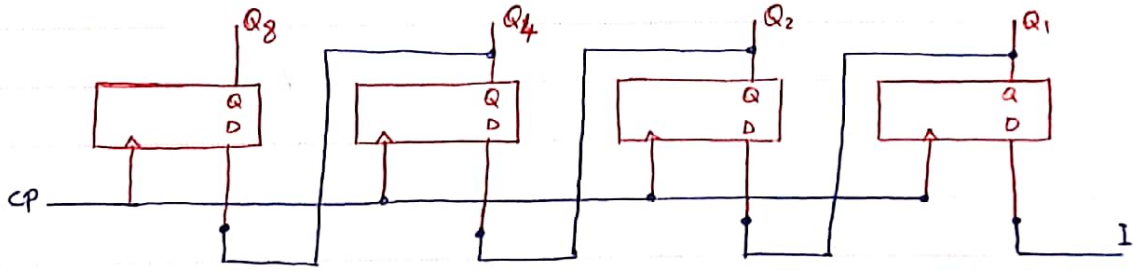


← ثباتی که با استفاده از FF های فعال شونده وسطح پالس ساعت ساخته می شوند latch می گویند.

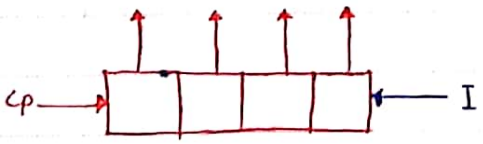
ثبات ها از نظر نحوه انتقال داده به داخل آنها :

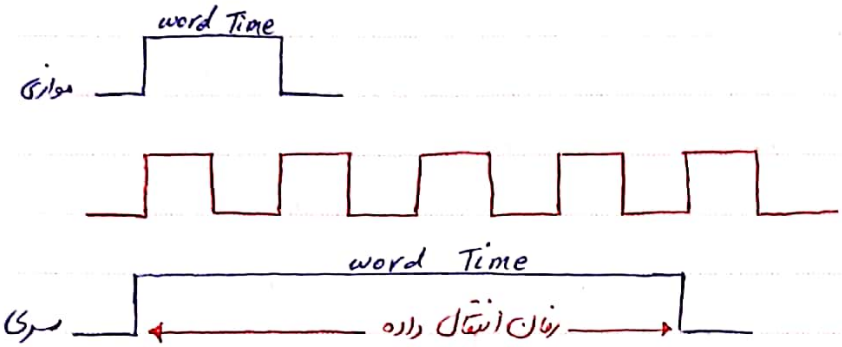
ثبات های بار دار شونده موازی ← در یک ترتیگر تمام FF ها براساس داده های موجود بروی خطوط داده ورودی بارگذاری می شوند.

ثبات های بار دار شونده سری ← در بین ثبات ها یک خط ورودی داده وجود دارد و داده ها طی چند پالس ساعت (ابعاد ثبات) با شیفت محتوای FF ها بطور متوالی، داده مورد نظر به داخل ثبات بارگذاری می شود.
(به این ثبات ها، ثبات های انتقالی گفته می شود.)



← یک تفاوت بین ثبات موازی و سری ← در موازی، داده ها در یک پالس به ثبات منتقل می شود ولی در ثبات سری داده ها طی چند پالس، به ثبات منتقل می شود.





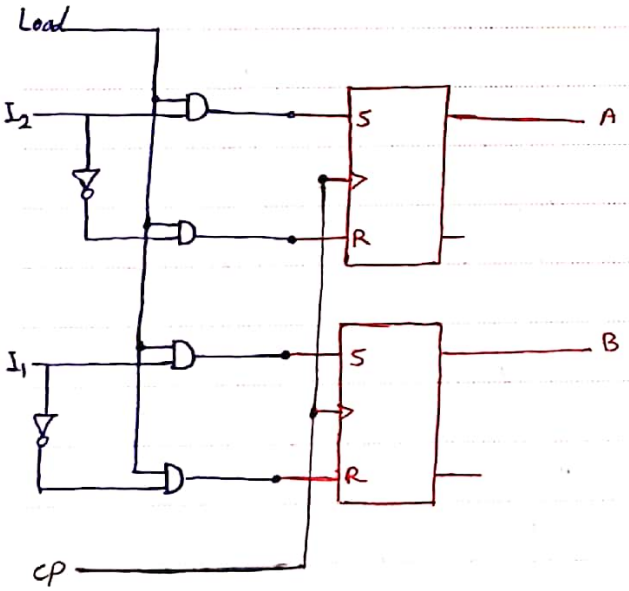
$I \longrightarrow Q_1$ 1

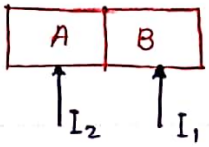
$Q_1 \longrightarrow Q_2$ 2

$Q_2 \longrightarrow Q$ 3

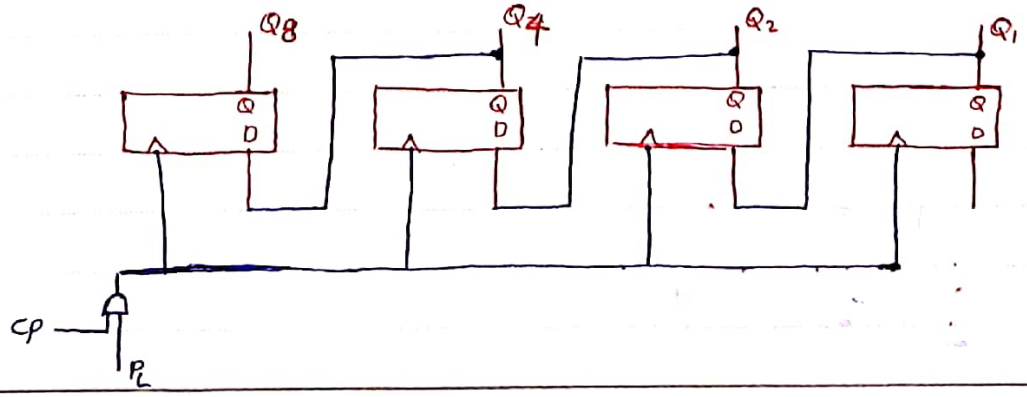
$Q \longrightarrow Q$ 4

ثبات موازی دوبیتی با RS-FF :

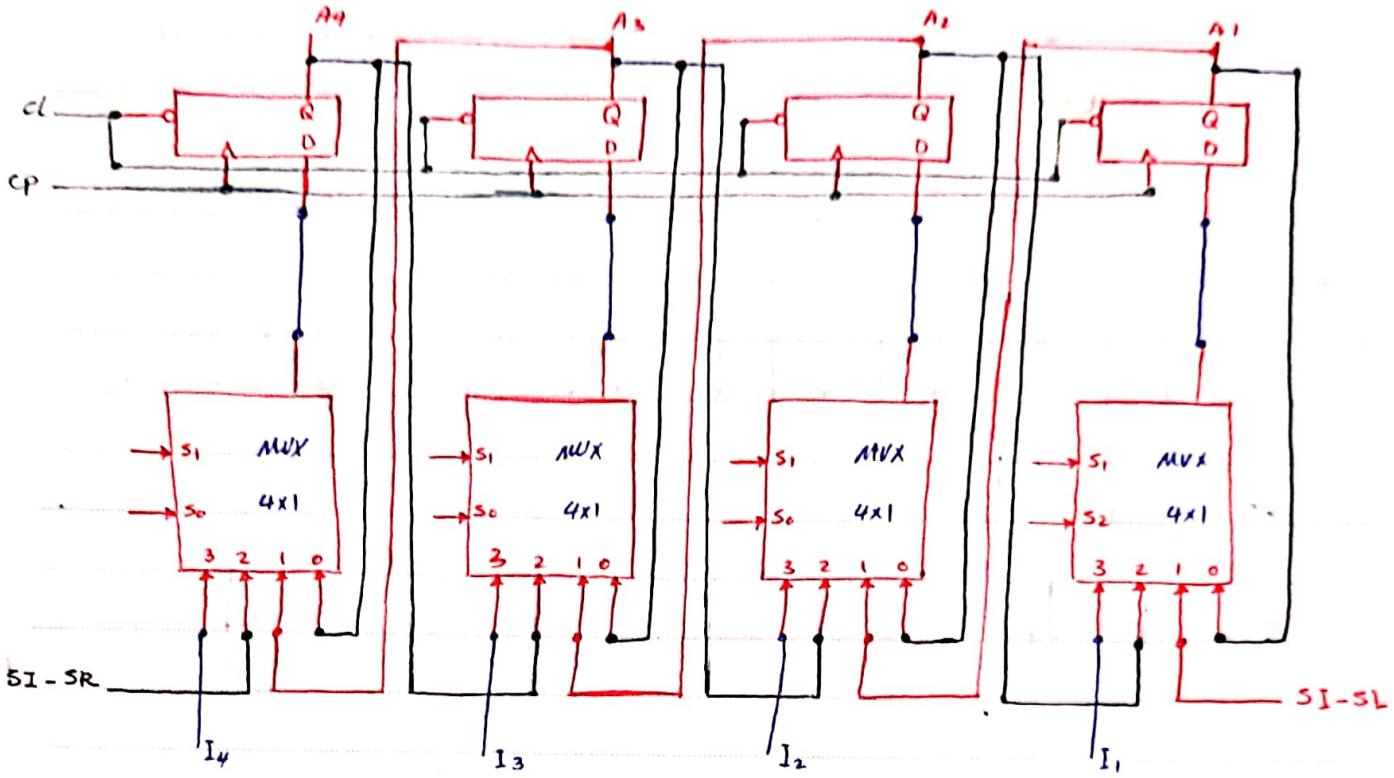




خط load ← یک خط کنترلی برای کنترل نمودن زمان انتقال داده ها به داخل ثبات به غیر از cp می باشد.

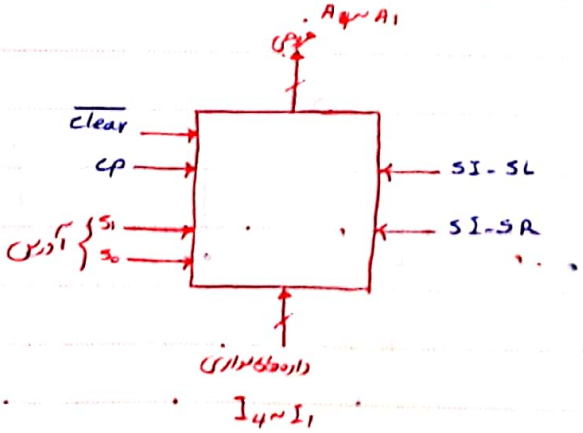


ثبات انتقالی دوجهته با امکان بارشدن موازی :



ثبات می خواهیم داشته باشیم که براساس آدرس داده شده بصورت جدول زیر عمل کند :

	S_1	S_0	عملکرد
✓	0	0	نگهداری داده های موجود (ابقاء)
✓	0	1	شیفت سری چپ - SI (Shift Left)
✓	1	0	شیفت سری راست - Serial Input (Shift Right)
✓	1	1	بارشدن موازی ثبات - Parellel Input

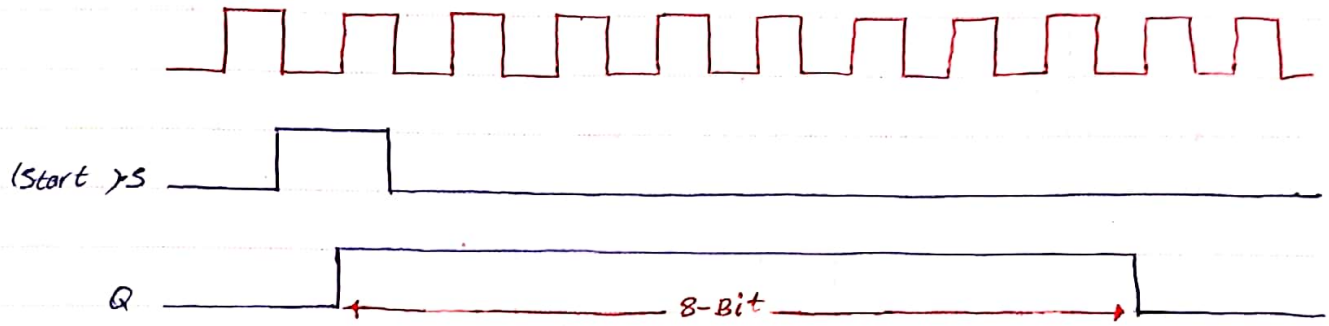


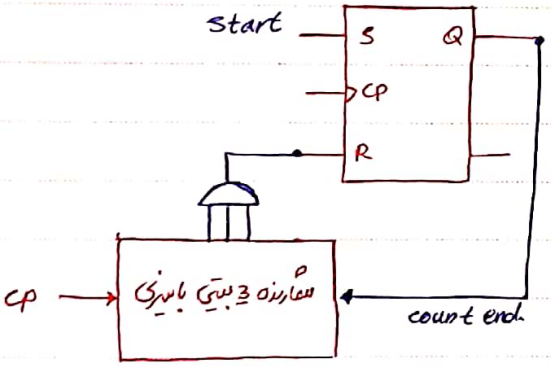
تولید دنباله های زمانی (Timing Sequences) :

مدار هایی که دنباله های زمانی مشابه با مدارهای موازی و سری تولید می کنند و سیگنال های تولید شده بعنوان سیگنال کنترل انتقال می توانند مورد استفاده قرار گیرند.

← در واقع به دنبال تولید سیگنال های Word-Time برای مدارهای سری و موازی می باشیم.

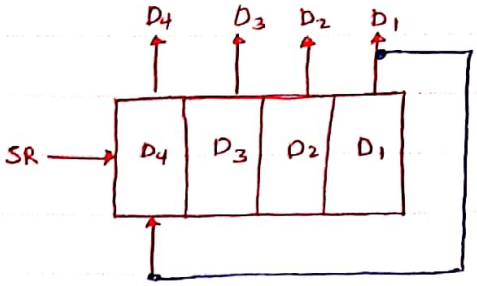
مدار مولد سیگنال Word-Time برای یک word هشت بیتی :

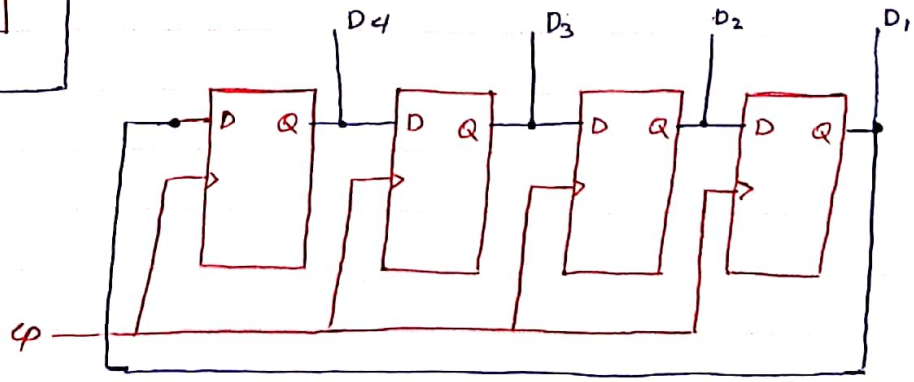


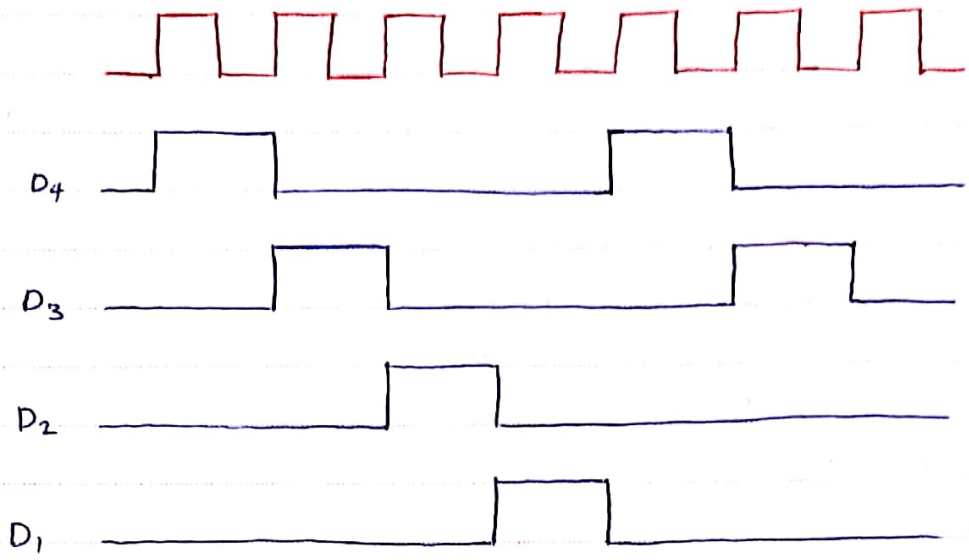


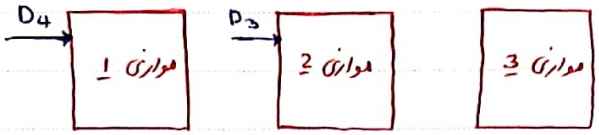
شمارنده حلقوی (Ring counter) :

چند FF بصورت سری طبق شکل زیر استفاده می کنند :

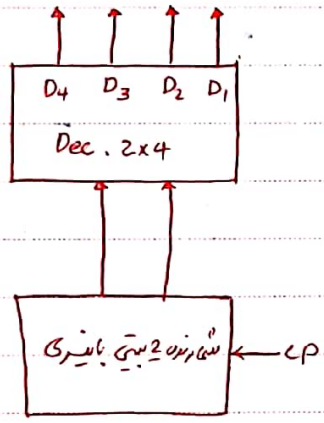




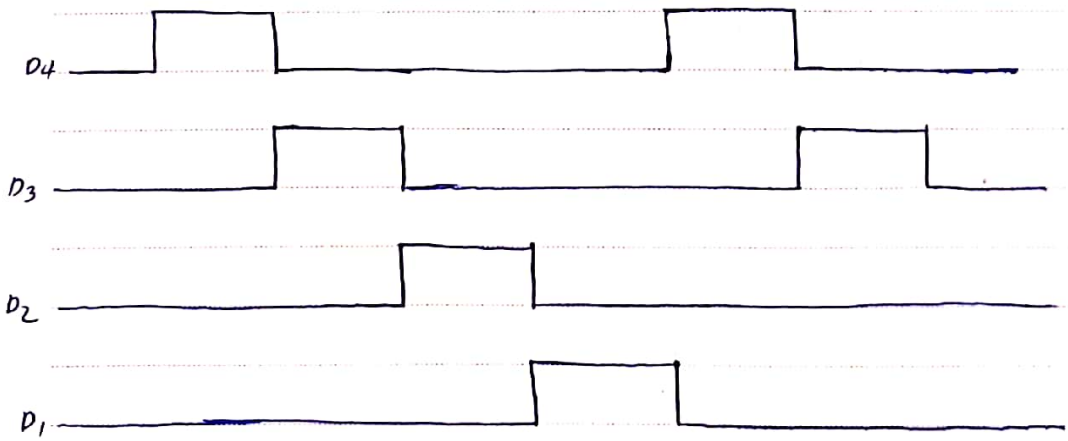




طراحی بر اساس دیکودر:



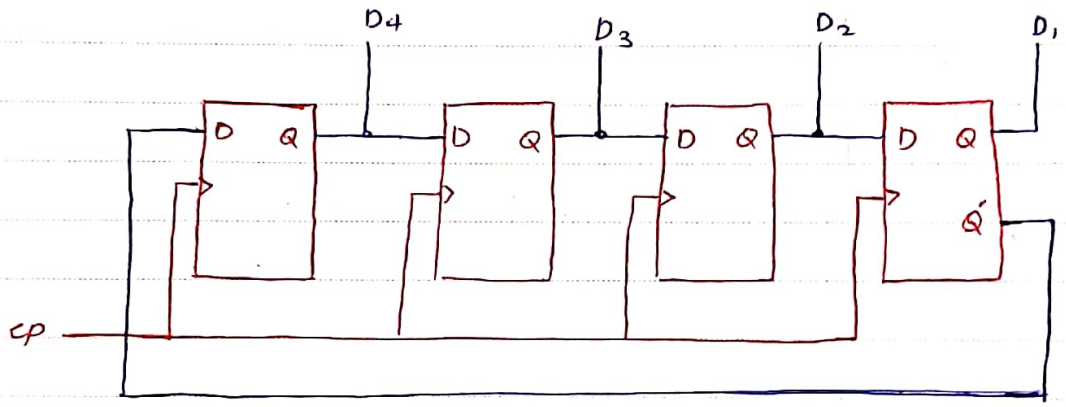
y_1	y_0	
0	0	$D_1 = 1$
0	1	$D_2 = 1$
1	0	$D_3 = 1$
1	1	$D_4 = 1$

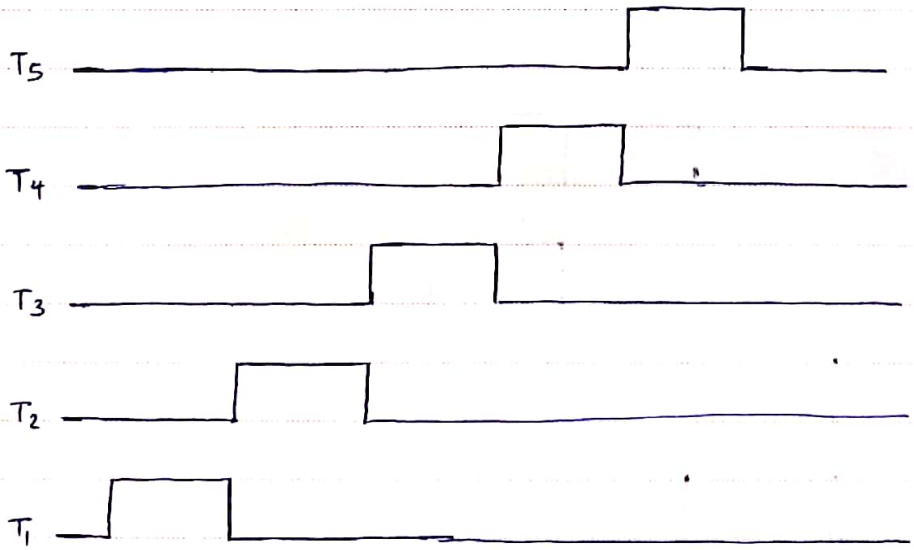


شمارنده جانسون:

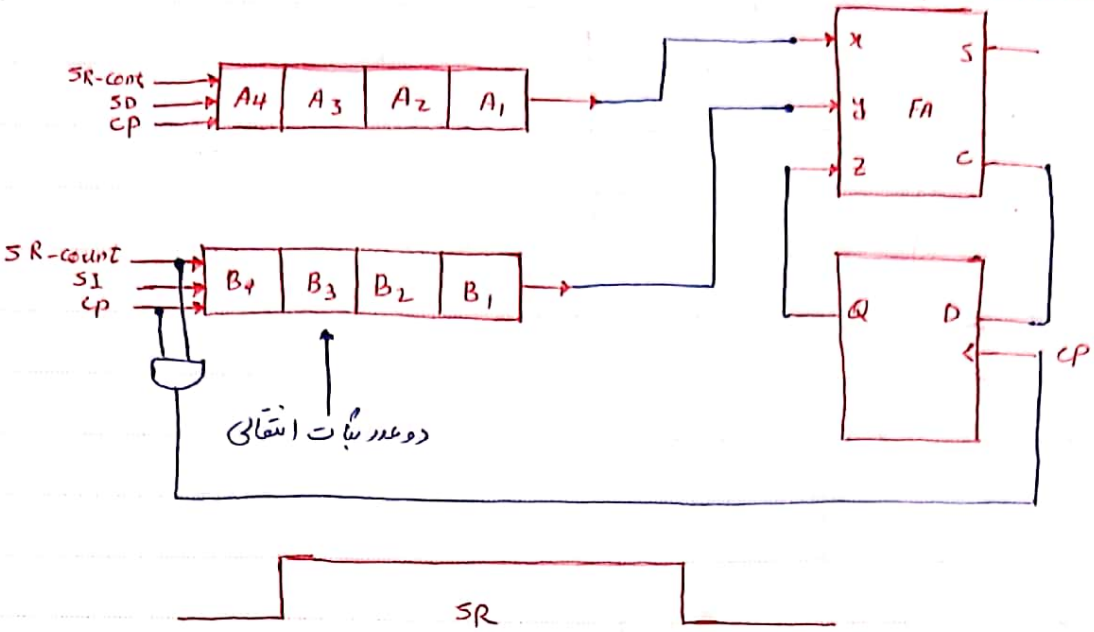
در این شمارنده از چند FF بطور متوازی استفاده شده و خروجی $\bar{Q}$ کم ارزش ترین، به ورودی پرارزش ترین FF وصل شده است (ساختار انتقال سری با ساعت مثبت) در حالت اول تمام FF ها را clear می کنیم.

شماره پالس	D_4	D_3	D_2	D_1	توابع ورودی مقابل
1	0	0	0	0	$T_1 = \bar{D_4}\,\bar{D_1}$
2	1	0	0	0	$T_2 = D_4\,\bar{D_3}$
3	1	1	0	0	$T_3 = D_3\,\bar{D_2}$
4	1	1	1	0	$T_4 = D_2\,\bar{D_1}$
5	1	1	1	1	$T_5 = D_4\,D_1$
6	0	1	1	1	$T_6 = \bar{D_4}\,D_3$
7	0	0	1	1	$T_7 = \bar{D_3}\,D_2$
8	0	0	0	1	$T_8 = \bar{T_2}\,T_1$

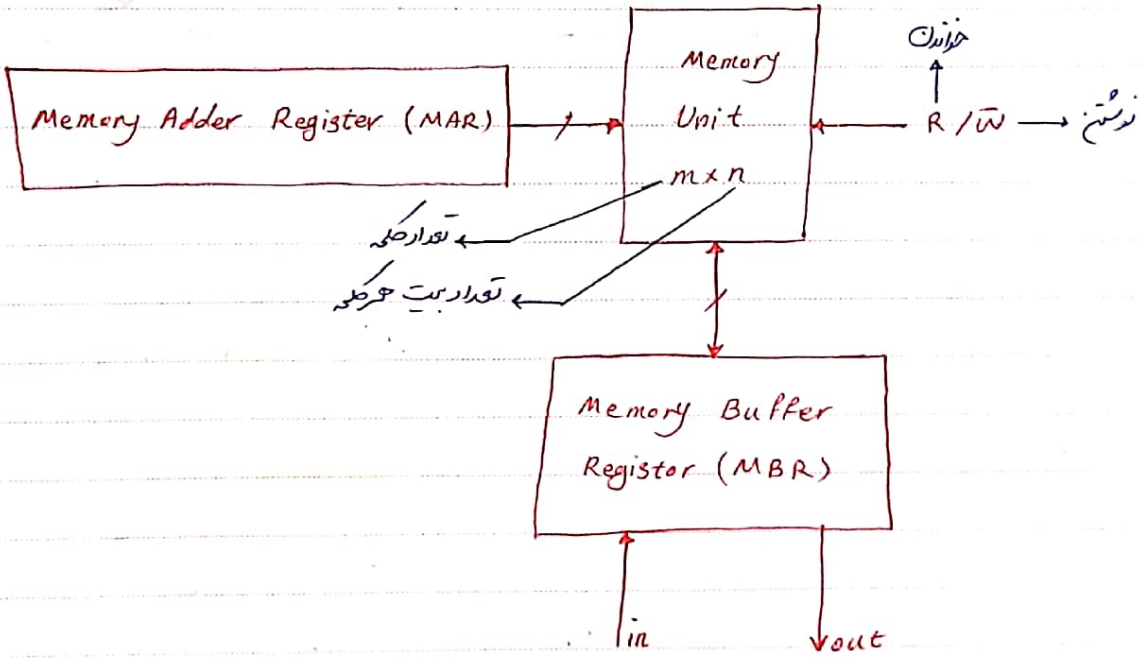




جمع کننده ۴ بیتی سریع (نوع FA) :

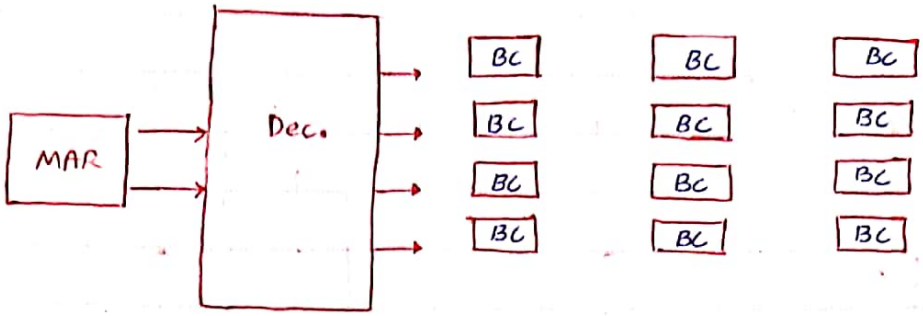


حافظه دسترسی تصادفی (جزو انتخابی) : Random Access Memory (RAM)



فرض { m = 4, n = 3

Subject:
Year. 98 Month. 03 Date. 07 ( )



BC ← سلول های باینری که عمل ذخیره سازی ۱ بیت باینری را دارند و بلوک های تشکیل دهنده واحد اصلی حافظه می باشند.