مدارهای منطقی

Year: ----- Month: ----- Day: ----- Subject: -----

فصل اول: سیستم های دودویی

آنالوگ و دیجیتال

سیستم آنالوگ سیستمی پیوسته است و نامتناهی. در حقیقت اگر در این سیستم دو عدد در نظر بگیریم بین این دو عدد یا دو صفحه چندین عدد و صفحه نیز وجود دارد. برای مثال در یک ساعت آنالوگ چرخ دنده ها پیوسته در حال حرکت اند که بین هر دو ثانیه از آن چند ثانیه دیگر را نیز نشان داده می شود.

سیستم دیجیتال سیستمی گسسته است که نمی تواند نامتناهی شود (برای تولید یک سیستم منطقی باید متناهی شود). در حقیقت اگر در این سیستم دو عدد در نظر بگیریم دیگر بین آن دو عدد دیگری وجود ندارد. مانند اعداد طبیعی و ....

برای مثال در یک ساعت دیجیتال باید نهایتا یک دقت داشته باشد. مثلا باید ثانیه، میلی ثانیه، میکرو ثانیه و .... باشد و نمی توانیم بین دو ثانیه ثانیه ای دیگر پیدا کنیم. نکته ای دیگر که وجود دارد این سیستم باید حتما دودویی شود. پس:

نکته: برای طراحی یک سیستم، باید سیستمی گسسته و متناهی انتخاب شود که برای

Shahab

استفاده از آن باید سیستم دیجیتال شود. اما سیستم دیجیتال نمی‌شود مگر اینکه دودویی بشود. باز هم به زبان ساده‌تر داریم:

سیستم ← گسسته و متناهی ← دودویی

۱. * | دودویی | اصلی

برای تعریف آن یک مثال می‌زنیم. مثلاً می‌خواهیم یک سیستم منطقی برای باز و بسته شدن پنجره طراحی کنیم.

برای ورودی آن یا باز می‌شود (1) یا بسته می‌شود (0)

برای خروجی آن یا باز و هیچ عملی انجام نمی‌دهد (0,0) یا باز و عمل می‌کند (0,1) یا باز و عمل می‌کند (1,0) */

حال برای آخرین مثال آنالوگ و دیجیتال می‌توان عکس را مثال زد

یک عکس آنالوگ (در حقیقت به طور حقیقی عکس آنالوگ را چشم خود در نظر می‌گیریم) که در این یک طیف رنگی داریم که بین دو رنگ آن باز هم رنگ است و در عکس دیجیتال که از پیکسل تشکیل شده است بین دو پیکسل از رنگ پیکسل رنگی تکمیلی وجود ندارد.

تبدیل مبنا

در این بخش تبدیل مبنا را به چند دسته تقسیم می کنیم:

1. تبدیل مبنا از 10 به 2
2. تبدیل مبنا از 2 به 10
3. تبدیل مبنای 16 و 8

---

1. تبدیل مبنا از 10 به 2:

برای تبدیل اعداد از مبنای 10 به 2 کافیست آن عدد را به صورت متوالی بر 2 تقسیم کرده و باقی مانده ها را به صورت برعکس از آخر به اول بنویسیم. ضمنا این حرکت را تا مادامی ادامه می دهیم که خارج قسمت صفر شود. مثال:

$$
\begin{array}{llllll}
10 & \underline{|\,2} & & & & \\
\underline{10} & 5 & \underline{|\,2} & & & \\
0 & \underline{4} & 2 & \underline{|\,2} & & \\
 & 1 & \underline{2} & 1 & \underline{|\,2} & \\
 & & 0 & \underline{0} & 0 & \\
 & & & 1 & &
\end{array}
\qquad (10)_{10} = (1010)_2
$$

حال برای تبدیل یک عدد اعشاری از مبنای 10 به 2 کافیست قسمت صحیح

Year: ----- Month: ----- Day: ----- Subject: ---------

آن عدد را به روش قبل به صورت دودویی بنویسیم. سپس قسمت اعشاری آن را در عدد ۲ ضرب کنیم

بعد از آن قسمت صحیح عدد بدست آمده را یادداشت نموده و قسمت اعشاری آن را مجددا در عدد ۲ ضرب می کنیم و با ادامه این روند چند حالت زیر رخ می دهد:

1. بعد از یادداشت نمودن عدد صحیح آن قسمت اعشار صفر می شود: در این حالت دیگر تمام شده و عدد بدست آمده دودویی متناظر عدد می باشد

2. بعد از یادداشت نمودن عدد صحیح آن به عدد اعشاری اولیه می رسیم: در این حالت نیازی به ضرب دوباره نبوده و عدد به دست آمده در مبنای 2 به صورت متناوب تکرار می شود و کافیست بالای یک علامت بار (abc) بگذاریم

3. بعد از یادداشت نمودن عدد صحیح آن هیچ وقت تمام نمی شود: در این حالت تقریبا به عدد اولیه مورد نظر می رسیم و کافیست به دلیل آنکه سیستم دیجیتال بسته و متناهی است تا حدودی مشخص مثلاً 5 بیت یا ... جلو رفته و هرچه بیشتر رویم جلوتر به عدد مورد نظر نزدیک تر می شویم

Shahab

مثال: عدد 10.42 را به صورت دودویی بنویسید.

$$10,42 \Rightarrow \begin{cases} (10)_{10} = (1010)_2 \\ (0,42)_{10} = (\ ?\ )_2 \end{cases}$$

$$\left.\begin{array}{lll} 0,42 \times 2 = 0,84 & 0 \\ 0,84 \times 2 = 1,68 & 1 \\ 0,68 \times 2 = 1,36 & 1 \\ 0,36 \times 2 = 0,72 & 0 \end{array}\right\} \Rightarrow (10,42)_{10} = (1010,01101)_2$$

$$0,72 \times 2 = 1,44 \quad 1$$

این پایان ناپذیر بوده و به صورت تقریبی تا 5 رقم اعشار جلو رفتیم!

2. تبدیل اعداد از مبنای 2 به 10

کافیست برای اعداد صحیح از راست به چپ به ترتیب از 0 تا n شماره گذاری کرده و هر رقم را در 2 به توان 0 تا n ضرب کنیم.

برای قسمت اعشاری نیز از چپ به راست به صورت برعکس از 1- تا n- شماره گذاری کرده و هر رقم را در 2 به توان آنها ضرب می‌کنیم

مثال تبدیل:

$$\overset{3}{1}\overset{2}{0}\overset{1}{1}\overset{0}{0},\overset{-1}{0}\overset{-2}{1}\overset{-3}{1}\overset{-4}{0}\overset{-5}{1}$$

$$(1010,01101)_2 = (?)_{10}$$

$$1010,01101 \begin{cases} (1010) \Rightarrow 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 = \boxed{10} \\ (0,01101)_2 \Rightarrow 0\times 2^{-1} + 1\times 2^{-2} + 1\times 2^{-3} + 0\times 2^{-4} + 1\times 2^{-5} = 0,40625 \end{cases}$$

$$\Rightarrow (1010,01101)_2 = (10,40625)_{10}$$

3. تبدیل اعداد در مبنای 16/8

برای آن باید توضیح مختصری بدهیم.

تبدیل 8 تایی و هگزادسیمال گرچه برای مدار تعریف نشده است اما برای خلاصه‌کردن اعداد دودویی بسیار مناسب است. چند قانون ساده وجود دارد که بیان می‌کنم:

1) اعداد 0 تا 9 خودشان نوشته می‌شوند اما اعداد 10 تا 15 را به صورت حروف انگلیسی از A تا F برای مبنای 16 می‌نویسیم.

2) برای تبدیل از مبنای 2 به 16 علاوه بر قانون بالا اعداد صحیح را از

Shahab

Year: Month: Day: Subject:

از سمت راست و قسمت اعشار آنها را (از صورت وجود) از سمت چپ

4 تا 4 تا جدا کرده و هر قسمت را به مبنای 10 تبدیل می‌کنیم و در این قسمت

از قانون اول استفاده می‌کنیم.

تذکر: اگر در هنگام جدا کردن 4 تا، 4 تای آخر کم آمد (چه در اعشاری و چه در

اعداد صحیح) به اندازه که 4 تا تکمیل شود صفر می‌گذاریم (زیرا در اعداد صحیح قبل از عدد

و در اعداد اعشاری بعد از اعداد 0 ارزشی ندارد)

3. برای تبدیل عدد از مبنای 2 به 8 کافیست مانند تبدیل عدد از مبنای 2 به

16 عمل کرده و به جای آنکه 4 تا 4 تا جدا کنیم 3 تا 3 تا جدا کنیم و باز اگر برای

راحتی کار در انتها 3 تا کامل نشده 0 را به شیوه قبل جایگذاری کنیم. (البته

می‌توانید هم بگذارید ولی برای راحتی این کار مستحب است)

نکته و تذکر: برای مبنا 16، A و B، C، ...، F به این صورت

هستند که

	10	2
A	10	1010
B	11	1011
C	12	1100
D	13	1101
⋮		
F	15	1111

⟹ از این موضوع می‌توان نتیجه گرفت که

اعداد در مبنای 16 از 0 هستند تا 15

و ...... 8 از 0 ... 7

Shahab

مثال: عدد 1100111010 را به مبنای 16، 8، 10 تبدیل کنید.

تبدیل به مبنای 10: $1\times2^9+2^8+2^5+2^4+2^3+2=826$

تبدیل به مبنای 8: $001,100,111,010 = 1472$

تبدیل به مبنای 16: $0011,0011,1010 = 33A$

تذکر: برای تبدیل یک عدد (دودویی) به مبنای 8 می‌توانستیم آن را اول به 10 سپس به 8 تبدیل کنیم اما این کار سودمندی ندارد.

نکته: برای تبدیل برعکس از 8، 16 به 2 می‌توان هر عدد و حرف را به صورت دودویی نوشت مثال: (حتما بخش‌های 4 تایی و 3 تایی رعایت شود)

$(10FA)_{16} = (0001,0000,1111,1010)_2$

روش سخت‌تر — مبنای 10: $1\times16^3+0\times16^2+15\times16^1+10\times16^0$

$(2071)_8 = (010,000,111,001)_2$

روش سخت‌تر — مبنای 10: $2\times8^3+0\times8^2+7\times8+1\times8^0$

Year: Month: Day: Subject:

مکمل ها

مکمل ها به طور کلی به دو دسته r و r-1 مبنا تقسیم می شوند اگر N یک عدد n رقمی در مبنای r باشد.

لذا مکمل r-1 عدد N برابر است با M به طوری که $M = r^n - 1 - N$

مکمل r عدد N برابر است با L به طوری که $L = r^n - N$

مثال ۱: مکمل 9 عدد 325 چقدر است؟

* روش ساده تر آن است که به اندازه ی رقم N مکمل r را قرار دهیم سپس از هم کم کنیم و این برای کلیه ی موارد اجرایی به جز موارد خاص ... (در ادامه)

عدد 325 بر مبنای 10 است : روش فرمولی

داریم: $10^3 - 1 - 325 = 674$

روش ساده تر :
$$\begin{array}{r} 999 \\ -\ 325 \\ \hline 674 \end{array}$$

در مثال بالا عدد 325 مکمل 674 و بالعکس می باشد که بر اساس 10-1 یا r-1 به وجود آمده است.

مثال ۲: مکمل 10 عدد 325 چقدر است؟

* در اینجا کافیست به مکمل 9 عدد 325 یک واحد بیافزاییم یعنی 675

روش فرمولی : $10^3 - 325 = 675$

روش ساده تر :
$$\begin{array}{r} 1000 \\ -\ 325 \end{array}$$

اما از مهم ترین موارد در این قسمت مکمل 1 و مکمل 2 است به

به آن می پردازیم :

① مکمل 1 : روش 1 (استفاده از فرمول)

در مبنای ۲ داریم ، مکمل r-1 : مورد * $M = r^a - 1 - N$

مکمل 1 در واقع همان مکمل (2-1) است زیرا اعداد در مبنای 2 اند

روش دوم : در این روش کافیست عدد را بعد از تبدیل به مبنای 2 (البته دقت شود که در روش قبل نیز باید عدد به مبنای 2 تبدیل شود)

سپس از یک جهت همه صفرها را به عدد یک و همه یک ها را به عدد صفر تبدیل می کنیم.

مثال 1 :

$N = 5 \Rightarrow N = 101$

روش اول : $a = 3$, $r = 2$ $\rightarrow \underbrace{2^3}_{8} - 1 - 101 = 010$

8 ← در مبنای 2 → 1000

تفریق در مبنای 2

$$\begin{array}{r} 1000 \\ -\ \ \ \ 1 \\ \hline 111 \\ -101 \\ \hline 010 \end{array}$$

روش دوم: $N = 101$

مکمل ۱ عدد 3 در مبنای 2

تبدیل صفر به ۱ و ۱ به صفر $\rightarrow 010$

مثال 2: $N = 1001100111001$

مکمل ۱ عدد بالا در مبنای 2

روش دوم $\rightarrow M = 0110011000110$

② مکمل 2: روش اول (استفاده از فرمول)

مکمل ۲

* مرور: $L = r^n - N$

r همان مبنای 2 است.

روش دوم: در این روش کافیست مکمل ۱ عدد را یافته و با عدد ۱ جمع بزنید

روش سوم: در این روش بسیار ساده کافیست بعد از تبدیل عدد به مبنای 2 (در تمامی روش های بالا باید عدد به مبنای 2 تبدیل شود) از سمت راست عدد شروع کرده و تا اولین 1 که رسیدیم همه را یادداشت کنیم. سپس پس از اولین عدد 1 بقیه را از صفر به 1 و پس از عدد 1 هم صفر تبدیل کنیم.

مثال: $N = 10100$

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

روش اول

$$\begin{cases} N = 10100 \\ a = 5 \\ r = 2 \end{cases} \longrightarrow L = 2^5 - 10100$$

مبنای 2

$$\begin{array}{r} 100000 \\ -\ 10100 \\ \hline 001100 \end{array}$$

در 5 بیت

01100

روش دوم :

$$\begin{array}{r} N = 10100 \\ 01011 \\ +\quad 1 \\ \hline 01100 \end{array}$$

مکمل 1

مکمل 2

روش سوم : $N = 10\overset{\text{به یک رسیدیم}}{\overleftarrow{100}} \xrightarrow{\text{مکمل}} L = 01100$

اعداد مثبت و منفی

① علامت ـ مقدار

② مکمل 1

③ مکمل 2 ***

از این سه روش برای تعیین مثبت یا منفی بودن اعداد استفاده می کنیم که مهمترین روش در این قسمت روش سوم است

Shahab

۶) علامت - مقدار: در این روش عدد مثبت و منفی را بدون در نظر گرفتن علامت آن به عدد دودویی تبدیل می‌کنیم. سپس برای نشان دادن مثبت بودن در سمت چپ‌ترین خانه آن عدد 0 و برای نشان دادن منفی بودن در سمت چپ‌ترین خانه آن عدد، عدد 1 قرار می‌دهیم.

در این حالت کمترین مقدار و بیشترین تعداد بیت مورد نیاز برای نشان دادن مثبت یا منفی بودن آن از روش زیر استفاده می‌کنیم:

فضای n بیتی $2^n - 1$ عدد مختلف

- از این عدد منفی $-(2^{n-1} - 1)$
- تا این عدد مثبت $2^{n-1} - 1$

مثال: عدد 7+ را در سیستم علامت - مقدار نمایش دهید.

عدد 7: 111 $\rightarrow$

- $+7: 0111$
- $-7: 1111$

* نکته: در این روش‌ها ممکن است به حرف و عدد اول دودویی دو عدد مانند هم باشند

مثلاً عدد 7-: 1111 و از آن طرف عدد بدون علامت 15 نیز 1111 است که خود سیستم بنا به مورد استفاده آن را تشخیص می‌دهد.

Year: Month: Day: Subject:

تست — به کمک تشخیص فضای مورد نیاز : $2^n - 1 = 15 \Rightarrow 2^n = 16 \Rightarrow n = 4$

کمترین عدد : $-(2^{4-1} - 1) = -(8-1) = -7$

بیشترین عدد : $2^{4-1} - 1 = 7$

② مکمل ۱ : در این روش برای مثبت و منفی عدد باید به روش زیر عمل کنیم :

نکته : همهٔ اعداد مثبت علامت ـ مقدار ، مکمل ۱ ، مکمل 2 با هم برابر بوده و فقط کافیست عدد مثبت علامت ـ مقدار را بدست آورید.

۱. حال با توجه به نکتهٔ بالا فقط باید روی اعداد منفی تمرکز کنیم. برای اینکار ابتدا علامت (-) عدد را نداشته و عدد علامت دار را به مکمل ۱ تبدیل می کنیم.

2. عدد مکمل ۱ را در مبنای ده می بریم.

فضای n بیتی $2^n - 1$ حالت دارد : $\begin{cases} -(2^{n-1} - 1) \\ (2^{n-1} - 1) \end{cases}$

③ مکمل 2: در این روش برای منفی عدد نیز مانند مکمل ۱ عدد عمل کرده و بعد از تبدیل عدد به دودویی و سپس مکمل 2 را به دست آورده و آن را در مبنای 10 نوشته و یک منفی در پشت آن می گذاریم.

2^n عدد مجموعه — فضای n بیتی : $\begin{cases} -2^{n-1} \\ 2^{n-1}-1 \end{cases}$

مثال 1: عدد 7- را به سه روش نمایش دهید.

-7 :
- علامت مقدار : 1111 ⇒ -7
- مکمل 1 : 0000 ⇒ -0
- مکمل 2 : 0001 ⇒ -1

مثال 2: -6 :

10110
- علامت مقدار : 10110
- مکمل 1 : 01001 ⇒ -9
- مکمل 2 : 01010 ⇒ -10

مثال 3: -8 :

برای آنکه منفی 8 شود به 5 بیت نیاز است

11000
- علامت مقدار : 11000
- مکمل 1 : 00111 : -7 (4 بیت بیشتر نیاز است)
- مکمل 2 : 01000 : -8

Year: Month: Day: Subject:

تفریق :

برای تفریق دو عدد کامپلمنت به مثال زیر توجه کنید :

مثال : 2 - 7 را به دست آورید .

$7 - 2 = 7 + (-2) = 0111 + 1110 = 0101$

$2 : 010 \xrightarrow[\text{مکمل 2}]{\text{تبدیل به}} 1110$

کدهای دهدهی

	P	کد BCD	افزونی 3	84-2-1
0	0	0000	0011	0000
1	1	0001	0100	0111
2	1	0010	0101	0110
3	0	0011	0110	0101
4	1	0100	0111	0100
5	0	0101	1000	1011
6	0	0110	1001	1010
7	1	0111	1010	1001
8	1	1000	1011	1000
9	0	1001	1100	1111

کد مکمل / مکمل 1

کد مکمل / مکمل 1

* اگر عدد به عنوان Label باشد رقم به رقم عدد را باینری می‌کنیم اگر عدد این

Shapour خود عدد را باینری می‌کنیم .

Year: ..... Month: ..... Day: ..... Subject: .....

حل:

359

0011 0101 1001

نکته ۱: اگر 359 را به مبنای 2 ببریم کد کوتاه تری بدست می آید. این روش تنها برای محاسبات ریاضی به درد می خورد اما روش BCD فقط برای عملیات غیر محاسباتی کاربرد دارد.

اگر بتوان مکمل 1 یک عدد را نوشت به آن کد مکمل می نویسیم.

برای راحتی به خاطر سپردن کدها می توانیم به اول بیت نگاه کنیم: اگر 0 و 1 به طور مساوی در 5 بیت اول و دوم پخش شده بود باید 5 تای اول صفر و 5 تای دومی 1 باشد

تذکر 1: در سیستم BCD 6 تای آخر رها شده است.

تذکر 2: در افزونی 3، از 3 عدد اول و 3 عدد آخر چشم پوشی شده است (یعنی 2,1,0 ، 13، 14، 15 استفاده نشده است.)

تذکر 3: در سیستم 1-2-4-8 از 6 عدد وسط چشم پوشی شده است (یعنی اعداد 3,2,1 ، 12، 13، 14 استفاده نشده است.) ضرائب منفی نیز چشم پوشی بدلیل انتساب 6 عدد

نکته 2: چون در 4 بیت در حال کدگذاری هستیم 16 حالت مختلف از عدد 0 تا 15 داریم

Shahab

$n = 4 \Rightarrow 2^4 = 16$

Year: Month: Day: Subject:

	2421	5043210 دو پنجی
0	0000	0100001
1	0001	0100010
2	0010	0100100
3	0011	0101000
4	0100	0110000
5	1011	1000001
6	1100	1000010
7	1101	1000100
8	1110	1001000
9	1111	1010000

کد مکمل

آشکارساز خطا

مثال: به ازای هر خطا در سیگنال معمولاً به ازای 10^{-5} ممکن است این خطا رخ دهد یعنی:

$E: 10^{-5} \times 10^{-5} = 10^{-10}$

2 خطا

برای یافتن خطا تنها سیستم باید به مجموعه های 0 ها و 1 ها عوض شده است.

یافتن خطا

در اینجا سه راهکار وجود دارد که نام می‌بریم:

1) سیستم دو پنجی: در اینجا اگر یک 0 به عدد یک و یا 1 به 0 تبدیل شود (که در حالت کلی است) سیستم می‌تواند خطا را تشخیص دهد و Err دهد. لذا در اینجا تعداد بیت‌ها 7 است و ممکن است خطا را افزایش دهد ولی سیستم راحت‌تر می‌تواند در این حالت خطا را پیدا کند.

Year: مبدا Month: Day: Subject:

مقصد

0101000 → 0101000 مثال ۱: به احتمال ۹۸٪ خطای روی نداده است

خطا → 0111000 ۱۰۰٪ خطا است

→ نکته ۳: اگر می خواهد که یک عدد به عدد دیگری

تبدیل شود باید چند خطا رخ دهد که در این مثال عدد 0101000 (3) به عدد

10001000 (7) تبدیل شود.

② بیت توازن { ① توازن زوج ② توازن فرد }

در این حالت می توان یک بیت که به بیت توازن معروف است اعداد را

به آشکارساز خطا تبدیل کرد

نکته ۴: سیستم دوپنجی به این دلیل آشکارساز خطای خوبی است که کلاً دو عدد 1 دارد

و به همین دلیل هر صفری که به 1 تبدیل شود 1 ها فرد می شود حتی اگر 1 به 0 تبدیل شود

اما به دلیل بیت های بالای آن سودی ندارد برای پروژه های بزرگ. لذا در این

حالت بیت توازن سیستم های 4 بیتی BCD، 84-2-1، 2421 و افزونی 3

را به 5 بیت تقسیم شد و همچنین آشکارساز های خوبی نیز باشند.

① توازن زوج: در این حالت 1 ها را زوج می کنیم (ستون P ستون BCD

توجه به جدول صفحه قبل)

② توازن فرد: در این حالت 1 ها را فرد می‌سازیم / برخلاف ستون ۴ مشکل

BCD سه صفحه قبل)

قابلیت های توازن ها به بیت کم آنها و خطایابی راحت‌تر آنها است

و سیستم با تشخیص زوج بودن یا فرد بودن 1 ها خطا را تشخیص می دهد.

تذکر 4: باید حتما نوع سیستم ورودی از قبل مشخص شود تا عدد مبدا مشخص شود از

چه نوع سیستمی است.

ASCII

کد اسکی ASS-key

کد اسکی 8 بیت است که یک بیت آن بیت توازن است.

```
int i;
char c;
for( int i=0; i<256; i++)
{
    c=i;
    cout << "\n" << i << " = " << c;
}
```

Shahab

کدهای اسکی از روی کتاب خوانده شود فقط یک نکته:

برای بدست آوردن کدها ابتدا ستون و سپس کدهای ردیف ها نوشته می شود مثلا

A: $\underbrace{1000}_{\text{ردیف}}\underbrace{001}_{\text{ستون}}$: 65

a: 1100001 : 97

کد گری (Mirror)

1bit	2bit	3bit	4bit
0	00	000	0000
1	01	001	0001
	11	011	0010
	10	010	0011
$n=2^1$		110	0010
	$n=2^2$	111	0110
		101	0111
		100	0101
		$n=2^3$	0100
			1100
			1101
			1111
			1110
			1010
			1011
			1010
			1001
			1000
			$n=2^4$

کاربرد این کد در فصل سوم ...

Year: ---- Month: ---- Day: ---- Subject: ----

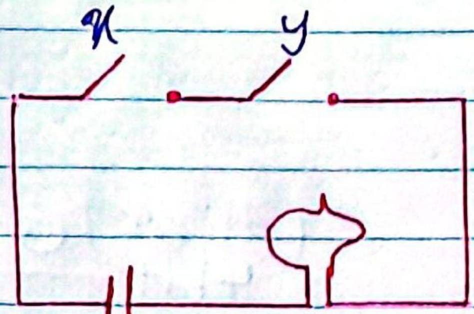
فصل دوم: منطق دودویی

سری — موازی — برعکس

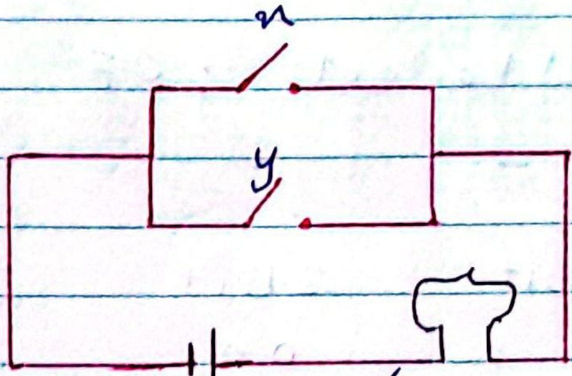
AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

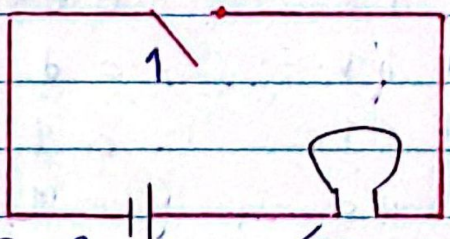
NOT	
0	1
1	0



برای آنکه لامپ روشن شود باید حتماً x و y روشن و بسته شود



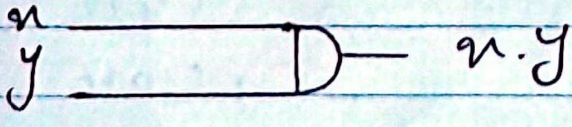
برای آنکه لامپ روشن شود کافیست یکی از x یا y بسته شود



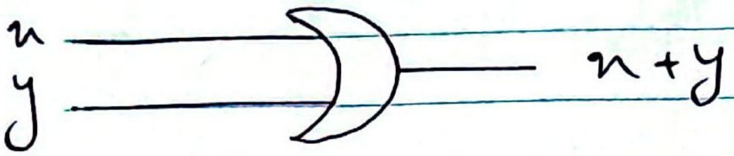
برای آنکه لامپ روشن شود باید کلید
به 0 تغییر کند

فصل دوم: گیت‌های منطقی

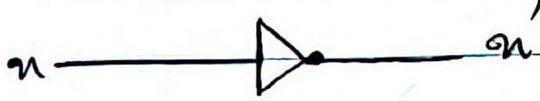
گیت and



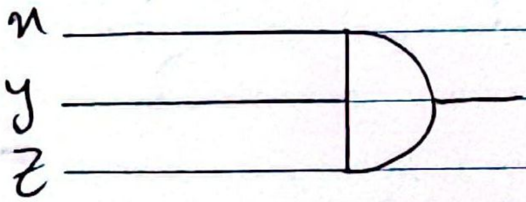
گیت OR



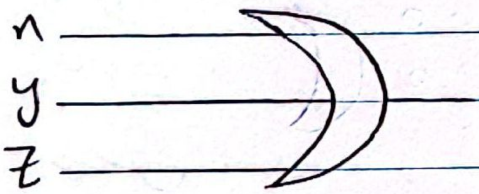
گیت NOT



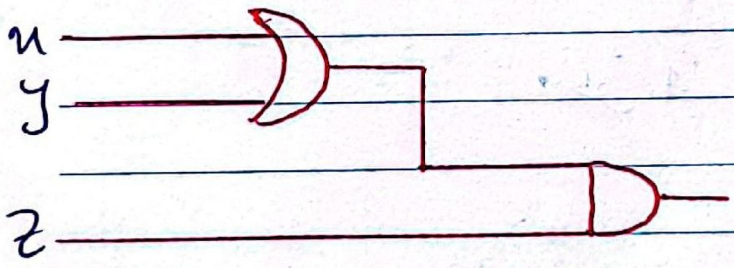
مثال 1: $x.y.z$



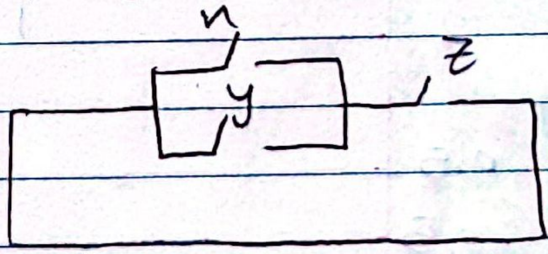
مثال 2: $x+y+z$



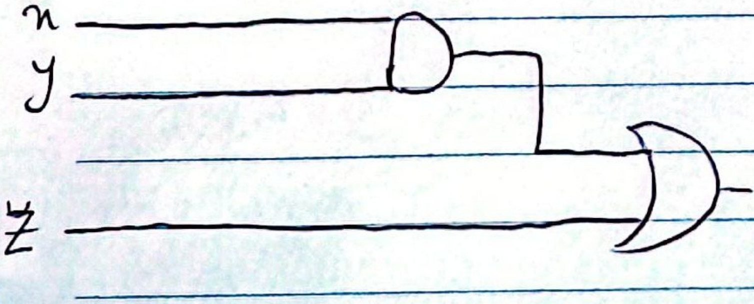
مثال 3:

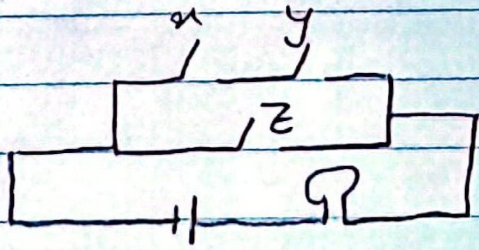


$(x+y).z$



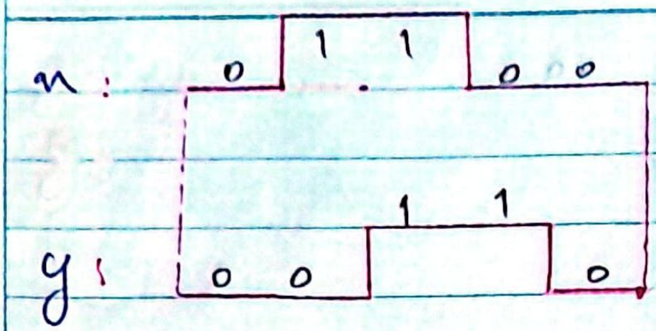
مثال 4: $(xy)+z$

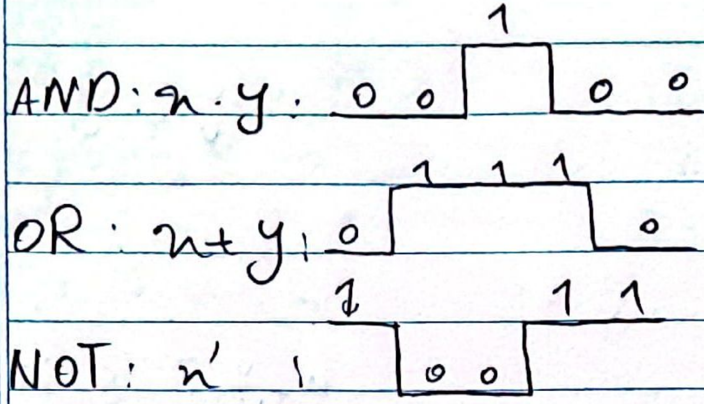




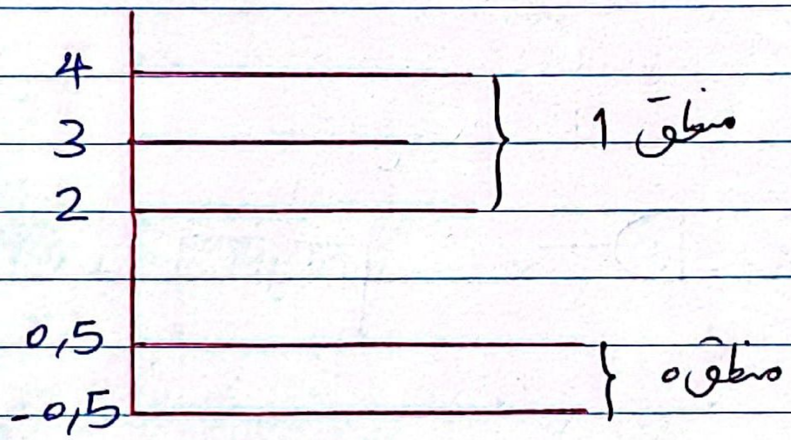
Year: Month: Day: Subject:

سیگنال





منطق



فصل دوم: جبر بول

جبر بول:

مجموعه پایه، عملگرها، اصول و تئوری ها

1. اصل بسته بودن: مجموعه A روی عملگر $\oplus$ بسته است هرگاه نتیجه عملگر $\oplus$ (هر عملی) عضو A باشد

2. اصل شرکت پذیری: $x+(y+z)=(x+y)+z$

3. جابجایی: $x+y=y+x$

4. عضو خنثی: مجموعه S نسبت به عملگر $\oplus$ دارای عضو خنثی e است اگر

$$\forall_{x\in S} \quad x+e=e+x=x$$

5. عضو وارون: به ازای هر عنصر $x\in S$ ، $y\in S$ وجود دارد که وارون x نسبت به $\oplus$ گوییم اگر

$$x+y=e$$

6. توزیع پذیری: $x*(y+z)=x*y+x*z$

Shahab

Year: ...... Month: ...... Day: ...... Subject: ......

ساختار جبری روی مجموعه B و عملگرهای Or (+) و And (.) ، اصول جبر دودویی

1. الف) عملگر + روی B بسته است

ب) عملگر . روی B بسته است

2. الف) عضو خنثی B نسبت به + برابر 0 است $x+0=0+x=x$

ب) عضو خنثی B نسبت به . برابر 1 است $x\cdot 1=1\cdot x=x$

3. الف) عملگر + روی B دارای خاصیت جابجایی است $x+y=y+x$

ب) عملگر . روی B دارای خاصیت جابجایی است $x\cdot y=y\cdot x$

4. الف) عملگر . روی + توزیع پذیر است $x\cdot(y+z)=x\cdot y+x\cdot z$

ب) عملگر + روی . توزیع پذیر است $x+(y\cdot z)=(x+y)\cdot(x+z)$

5. برای هر عنصر $x\in B$ عنصر $y=x'$ را مکمل x گوییم اگر

الف) $x+y=1$ Or

ب) $x\cdot y=0$ and

Year: Month: Day: Subject:

6. $[|B| \geq 2]$ حداقل دو عضو x, y متعلق به B هستند که $x \neq y$

$\{0, 1\}$

تفاوت های جبر ریاضی و جبر بول

1. توزیع پذیری + روی * در ریاضی نداریم اما در جبر بول توزیع پذیری

+ روی . در جبر داریم .

2. در جبر بول عنصر وارون نداریم .

3. در جبر بول اصل شرکت پذیری نداریم (البته وجود دارد ولی یک تئوری است)

4. در جبر ریاضی مکمل نداریم .

5. در جبر ریاضی روی مجموعه نامتناهی از اعداد کار می‌کند اما جبر بول در دیجیتال (دوارزشی)

روی $\{0, 1\}$ کار می‌کند

6. در جبر بول دو ارزشی عملگرهای not , or , and (مکمل) تعریف می‌شود

و اما تئوری های این قسمت عبارت اند از اما قبل از آن

Shahab

7. اصل دوگان: هر اصل در دوگانه آن درست است. اگر یک اصل دیگری اثبات شود، دوگان آن نیز درست خواهد بود. برای تبدیل به دوگان کافی است

$+ \leftarrow \cdot$ | $\cdot \leftarrow +$ | $1 \leftarrow 0$ | $0 \leftarrow 1$ تبدیل شود.

تئوری 1: الف) $x+x=x$ ب) $x \cdot x=x$

اثبات الف) $x+x=(x+x)\cdot 1=$ اصل 2-ب

$(x+x)\cdot(x+x')=$ اصل 5-الف

$x+(x\cdot x')=$ اصل 4-ب

$x+0=$ اصل 5-ب

x اصل 2-الف

به همین ترتیب طبق اصل دوگان برای ب نیز اثبات می‌شود.

تئوری 2: الف) $x+1=1$ ب) $x\cdot 0=0$

اثبات الف) $x+1=x+(x+x')=$ اصل 5-الف

$=(x+x)+x'$ تئوری شرکت‌پذیری

$=x+x'$ تئوری $x+x=x$

$=1$ اصل مکمل

Year: ----- Month: ----- Day: -----   Subject: -----------

تئوری 3: $(x')' = x$

تئوری 4: شرکت پذیری الف) $x+(y+z)=(x+y)+z$

تئوری ب) $x\cdot(y\cdot z)=(x\cdot y)\cdot z$

تئوری 5: (دمورگان) الف) $(x+y)'=x'\cdot y'$

ب) $(x\cdot y)'=x'+y'$

تئوری 6: جذب الف) $x+x\cdot y=x$

ب) $x\cdot(x+y)=x$

الف) $x+(x\cdot y) \overset{\text{عنصر خنثی}}{\equiv} (x\cdot 1)+(x\cdot y) \overset{\text{توزیع پذیری}}{\equiv} x\cdot(1+y) \overset{\text{تئوری}}{\equiv} x\cdot 1 \overset{\text{عنصر خنثی}}{\equiv} x$

اثبات شرکت پذیری:

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

$\equiv$

Shahab notebook

در حقیقت برای اثبات کردن می توان جدول ارزش های دو تابع استفاده کرد.

$$f(x,y,z) = g(x,y,z)$$

$$f(0,0,0) = g(0,0,0)$$

$$f(0,0,1) = g(0,0,1)$$

$$\vdots$$

توابع بولی

توابع مجموعه‌ای از ترتیب عملوندها و عملگرها و ضابطه‌ها و ثابت‌ها هستند که در توابع بولی به شکل زیر است:

عملگرهای بولی : not, or, and

متغیرهای بولی : x, y, z

ثابت ها : $\{0,1\}$

نکته: معمولاً ثابت‌ها نوشته نمی‌شوند زیرا یک عضو خنثی برای ضرب و جمع بوده و عملاً تاثیری ندارد.

مثال: $f_1(x,y,z) = (xy' + yz)(x+y)z'$

ضرب و جمع جابجایی

Shahab

$f_1 \equiv f_2(x,y,z) = (z'x + z'y)(xy' + yz)$

$\vdots$

تذکر: تابع را می‌توان به بی‌نهایت شکل مختلف به هم ارز نمود. لذا برای آنکه بتوان از این توابع استفاده کرد باید یک فرم مشخص برای آن در نظر گرفت.

نکته 2: دو تابع f و g را در نظر بگیرید:

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

$h(x,y) = x \cdot y$

حال مکمل از این دو تابع را بررسی می‌کنیم.

مثلاً جدول ارزش h' را رسم می‌کنیم:

x	y	h	h'
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

در اینجا از خاصیت دمورگان استفاده کردیم که قبلاً هم دیده بودیم. اکنون آن را تعمیم می‌دهیم:

$$(x_1 + x_2 + \dots + x_n)' = x_1' \cdot x_2' \cdot \dots \cdot x_n'$$

$$(x_1 \cdot x_2 \cdot \dots \cdot x_n)' = x_1' + x_2' + \dots + x_n'$$

مثال:

$f_2 = (xz' + yz')(xy' + yz)$

$$F_2' = (xz' + yz')' + (xy' + yz)'$$

$$= ((xz')' . (yz')') + ((xy')' . (yz)')$$

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

تذکر: در اینجا نیز مانند جبر ریاضی and روی or اولویت دارد.

1: $L(x,y,z) = x + y . z$

2: $L(x,y,z) = (x+y) . z$

در این دو مثال اگر $x=1$ و $z=0$ باشد در اولی L 1 خواهد بود در دومی 0 خواهد شد

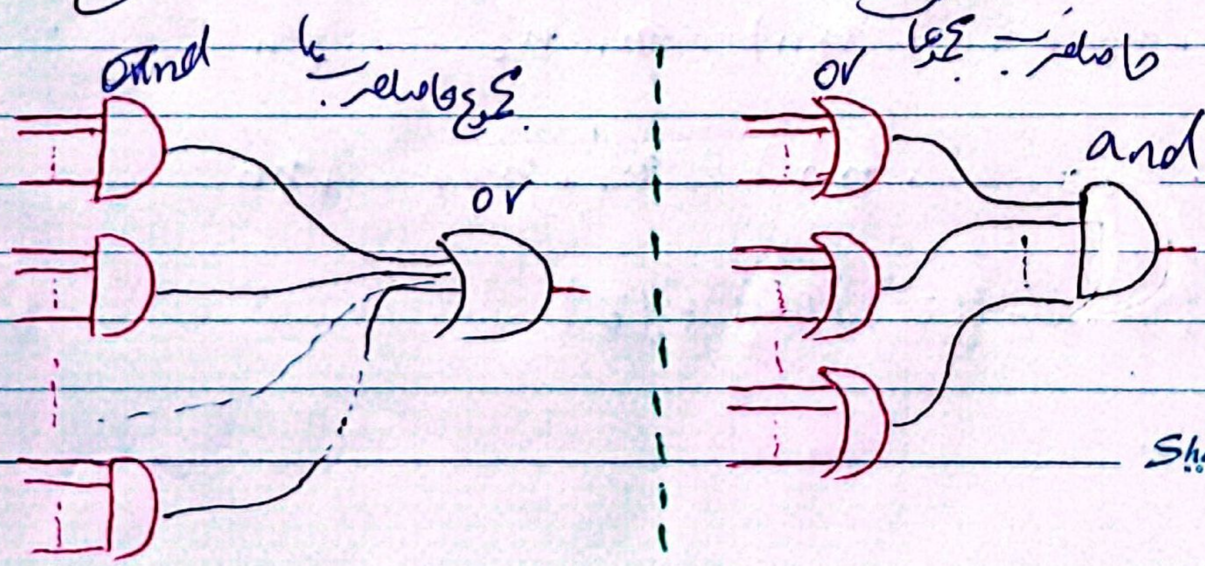
استاندارد سازی فرمول (کانونیک)

① مجموع حاصلضرب ها: $\underbrace{(\quad . \quad)}_{\text{حاصلضرب لیترال ها}} + (\quad . \quad) + (\quad . \quad)$

DNF

② حاصلضرب مجموعات: $\underbrace{(\quad + \quad)}_{\text{حاصلجمع لیترال ها}} . (\quad + \quad) . (\quad + \quad)$

CNF



نکته: تعداد پرانتزهای تو در تو هر چقدر بیشتر باشد، سطوح مدار بیشتر شده در نتیجه زمان اجرای آن افزایش می یابد.

مثال ۱: $f(x,y) = x+y$

- ۱) : $(x+y) \cdot 1$ حاصلضرب مجموعات
- ۲) : $x+y$ حاصلجمع ضرب ها

۲: $f(x,y,z) = (x+y) \cdot z$ حاصلضرب مجموعات

۳: $f(x,y,z) = \big(((x+y) \cdot z)\, x' + y(x'+z)\big)\, x'$

برای حل این باید minterm و maxterm را بشناسیم

minterms , maxterms

$\forall n \quad n = k \Rightarrow 2^k$ minterm

2^k maxterm

داریم:

نکته: همیشه رابطه روبه‌رو برقرار است

maxterms: $M_i = m'_i$

minterms: $m_i = M'_i$

ماکسترم ها را با M، مینترم ها را با m نمایش می دهیم.

Year: ...... Month: ...... Day: ...... Subject: ......

minterm ها حاصلضرب لیترال ها و maxterm ها حاصلجمع لیترال ها هستند.

مثال 3 برای درک بیشتر به جدول زیر توجه کنید.

x	y	z	minterms	maxterms	f	f'
0	0	0	$x'y'z' = m_0$	$x+y+z = M_0$	0	1
0	0	1	$x'y'z = m_1$	$x+y+z' = M_1$	0	1
0	1	0	$x'yz' = m_2$	$x+y'+z = M_2$	1	0
0	1	1	$x'yz = m_3$	$x+y'+z' = M_3$	1	0
1	0	0	$xy'z' = m_4$	$x'+y+z = M_4$	0	1
1	0	1	$xy'z = m_5$	$x'+y+z' = M_5$	0	1
1	1	0	$xyz' = m_6$	$x'+y'+z = M_6$	0	1
1	1	1	$xyz = m_7$	$x'+y'+z' = M_7$	0	1

نکته 2: در min باید آنهایی که ارزش صفر دارند را نقیض و آنهایی که 1 هستند خودشان نوشته می‌شوند و این در max ها برعکس هست.

نکته 3: minterm و maxterm ها از اندیس 0 تا 2^k-1 هستند (مثلاً در جدول بالا 3 عنصر x,y,z داریم که یعنی 2^3 تا max و 2^3 min هستند که اندیس‌هاشون هر دو از 0 تا 7 است)

تذکر: برای نوشتن حاصلجمع مینترم ها به اعداد 1 و برای نوشتن حاصلضرب maxterms به عدد 0 توجه داریم (این تذکر برای نقیض f برعکس است)

Year: ...... Month: ...... Day: ...... Subject: ......

مثال 3: $f = m_2 + m_3 = (x'yz') + (x'yz)$

$f' = (m_0 + m_1 + m_4 + m_5 + m_6 + m_7)' =$

$M_0 \cdot M_1 \cdot M_4 \cdot M_5 \cdot M_6 \cdot M_7 =$

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

$(x'y'z')$

مثال 1: $f(x,y) = x+y = x(y+y') + y(x+x') =$

$$\underline{xy} + xy' + \underline{yx} + yx' = \underset{1\ 1}{xy} + \underset{1\ 0}{xy'} + \underset{0\ 1}{x'y} = m_3 + m_2 + m_1$$

$= \Sigma(1,2,3)$

نکته 3: برای اینکه از جدول درستی نگیریم خاصیت مانند مثال 1 را اعمال کنیم.

نکته 4: برای خلاصه سازی minterm (حاصلجمع) از Σ و برای حاصلضرب maxterm

از Π استفاده می کنیم.

مثال 3 (خلاصه سازی): $f(x,y,z) = m_2 + m_3 = \Sigma(2,3)$

$f(x,y,z) = M_0 \cdot M_1 \cdot M_4 \cdots M_7 = \Pi(0,1,4,\ldots,7)$

Shahab NOTEBOOK

سایر عملیات منطقی

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

اگر دو متغیر x و y به شکل جدول روبه رو داشته باشیم،

هر کدام از قسمت های f می توانند 0 یا 1 باشند؛ به عبارت ساده تر یک عملیات دو متغیره دارای 16 = 2^4 حالت می باشد.

نتیجه: یک عملیات n متغیره ← 2^{2} تابع مختلف را دارا می باشد.

در این جا ما به دو متغیره ها می پردازیم،

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

$F_0 \equiv$ Null

$F_{15} \equiv$ Identity

$F_1 \equiv$ and

$F_{14} \equiv$ NAND

Year: .... Month: .... Day: .... Subject: ....

$F_7 \equiv$ OR $\qquad$ $F_8 \equiv$ NOR

($F_6 \equiv$ XOR) $\Rightarrow$ در ادامه $\qquad$ ($F_9 \equiv$ XNOR) $\Rightarrow$ در ادامه

$$\left.\begin{array}{l} F_{11} \equiv \text{if } y \text{ then } x \\ F_{13} \equiv \text{if } x \text{ then } y \end{array}\right\} \Rightarrow \text{Impleeation function}$$

$$\left.\begin{array}{l} F_2 \equiv x \text{ but not } y \\ F_4 \equiv y \text{ but not } x \end{array}\right\} \Rightarrow \text{Inhibition}$$

$F_3 \equiv$ transfer(x) $\qquad$ $F_{10} \equiv$ Complement(y)

$F_5 \equiv$ transfer(y) $\qquad$ $F_{12} \equiv$ Complement(x)

* * * * *

F_0: همیشه نادرست است (Null) $\qquad$ $F_{15} \equiv$ همیشه درست است (Identity)

** F_1 : $x \cdot y$ (and) $\qquad$ ** F_{14} : $x \uparrow y = (x \cdot y)' = x' + y'$ (Nand)

** F_7 : $x + y$ (or) $\qquad$ ** F_8 : $x \downarrow y = (x + y)' = x' \cdot y'$ (nor)

** F_6 : (XOr) یا مانع الجمع یعنی انحصاری است که در یک زمان نمی‌توانند باهم اتفاق بیافتند

مثلاً حسین یا تهران است یا شیراز ، مثلاً امروز هوا آفتابی است یا مهتابی است .

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

در واقع XOR زمانی در وضعیت 1 قرار می‌گیرد که دو متغیر در شرایط های متفاوت باشند

F_9

**(XNOR) ؛ $x \odot y = x'y' + xy$

F_{11}: $x \subset y$ $x + y'$ | F_2 ؛ xy' یا x/y

F_{13} ؛ $x \supset y$ $y + x'$ | F_4 ؛ $x'y$ یا y/x

F_{10} ؛ y' $F_{12} = x'$

مدارها و دروازه‌های منطقی

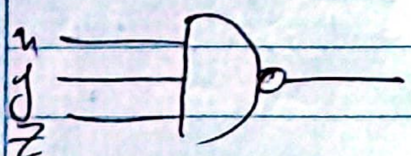
AND ؛ x, y → $x.y$ OR ؛ x, y → $x + y$

NAND ؛ x, y → $x' + y'$ NOR : x, y → $x'y'$

XOR : x, y → $yx' + y'x$ XNOR ؛ x, y → $x'y' + xy$

$x \oplus y$ $x \odot y$

نکته! وقتی تعداد ورودی‌ها را بیشتر می‌کنیم اگر خاصیت جابجایی و شرکت‌پذیری داشته باشد می‌توان برای آنها عملیات‌های چندتایی تعریف کرد مثلاً چون AND خاصیت جابجایی و شرکت‌پذیری دارد لذا برای NAND داریم هر چه را ابتدا AND و سپس Not می‌کنیم یعنی:



$(x.y.z)'$

Year: ...... Month: ...... Day: ...... Subject: ......

تذکر: NAND خاصیت شرکت‌پذیری ندارد زیرا:

$$x \uparrow (y \uparrow z) = (x \uparrow y) \uparrow z$$

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

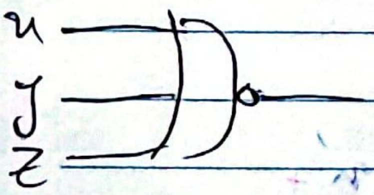
$$(x \cdot (y' + z'))' = ((x' + y') \cdot z)'$$

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

$$x' + (yz) \neq xy + z'$$

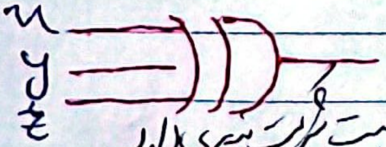
لذا به ناچار برای این عملیات از نکته قبل بهره می‌بریم.

نکته ۲: برای NOR نیز باید به شکل نکته ۱ عمل کنیم:



$(x+y+z)'$

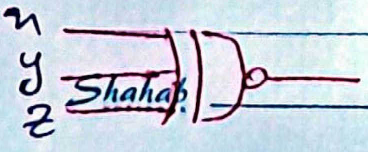
حال برای XOR



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

خاصیت شرکت‌پذیری دارد: $x \oplus y \oplus z = x \oplus (y \oplus z) = (x \oplus y) \oplus z$

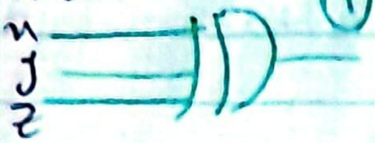
حال برای XNOR



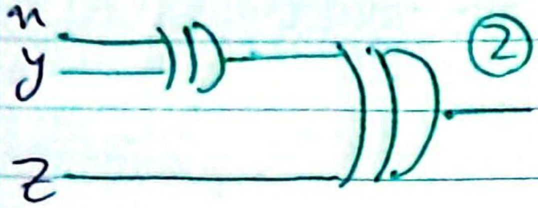
$$x \odot y \odot z = x'y'z' + xy'z' + x'yz' + xyz'$$

معادل مداری:

XOR

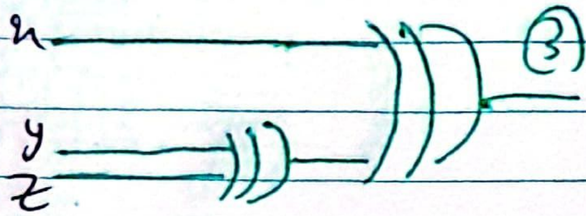
① 

$x \oplus y \oplus z$

② 

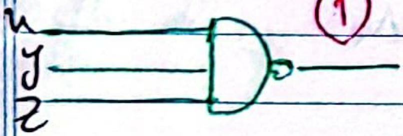
$(x \oplus y) \oplus z$

$1 = 2 = 3$

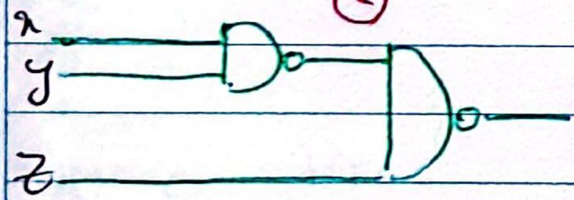
③ 

$x \oplus (y \oplus z)$

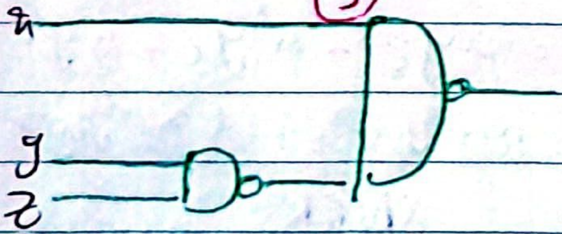
NAND

① 

$(xyz)'$

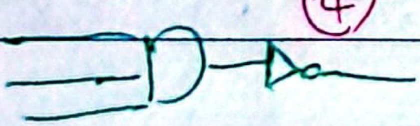
② 

$((xy)'z)'$

③ 

$(x(yz)')'$

$1 \neq 2 \neq 3$

④ 

$1 = 4'$

مدارهای مجتمع (IC) (Integrated Circuit)

IC ها معمولاً به چهار دسته تقسیم می‌شوند:

1) SSI (Small Scale Integration): حداکثر از 10 گیت منطقی تشکیل

2) MSI (Medium Scale Integration): دارای 10 تا 100 گیت منطقی

3) LSI (Large Scale Integration): 100 تا هزاران گیت منطقی

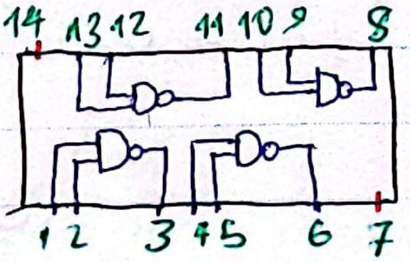
4) VLSI (Very Large Scale Integration): میلیون‌ها گیت منطقی

} IC

1) SSI: معمولاً 14 تا 16 پایه دارند (14 تا رواج است)، دارای انواع مختلفی از TTL , ECL , CMOS , ... هستند و همگی از کدهای مخصوص بهره می‌برند.

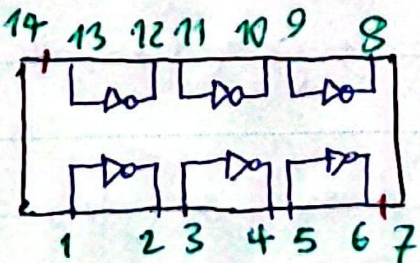
مثال:

TTL 7400
4 تا گیت NAND دارد
دو ورودی یک خروجی

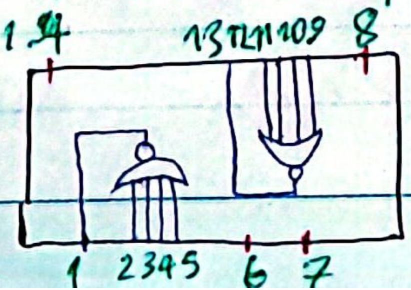


ECL 16 پایه دارد

TTL 7402
6 تا گیت NOT دارد
یک ورودی یک خروجی



CMOS 4002
دو تا گیت NOR دارد
4 ورودی 1 خروجی

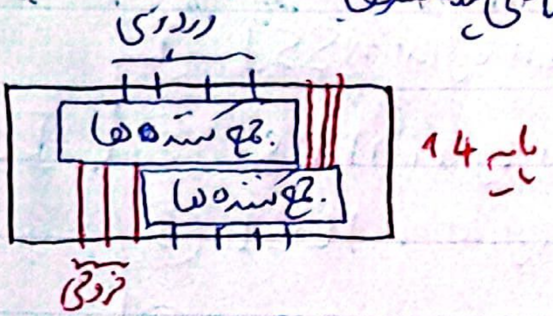


مثلاً اگر فرمی $(y+x)'$ داشته باشد آنگاه ورودی‌های اضافی را

$(x+y+0+0)'$

② MSI:

مثال: جمع کننده ها، تفریق کننده ها، دی کدر (Decoder)، انکدر (Encoder)، مدارهای ترتیبی، ماشین حساب ها



③ LSI:

مثال: CPU (Centeral Porcesser Unit)، حافظه ها (ROM, RAM)، ...

④ VLSI: آرایه حافظه، تراشه میکروکامپیوترها، ریزپردازنده ها.

سطح های گیت منطقی

برای این قسمت باید مثال سطوح را ببینیم تا چند نکته را به خاطر داشته باشید

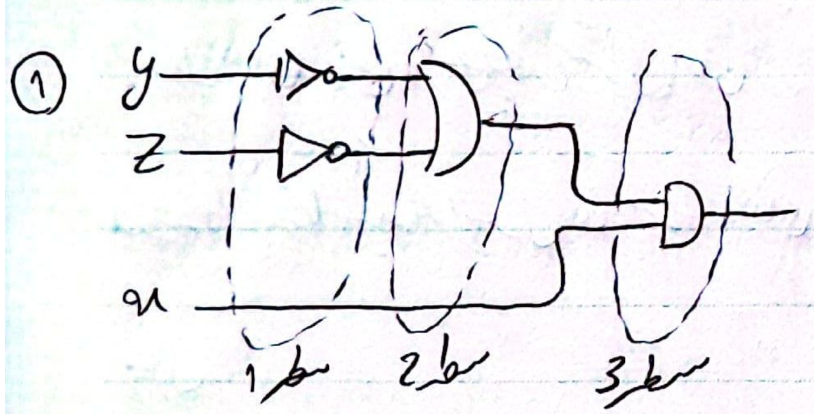
۱: اگر یک خروجی از ورودی گرفته شده باشد یک سطح حساب می گردد (در صورتی که خود کلام یک ورودی باشد)

۲: به ازای هر NOT یک سطح اضافی گردد.

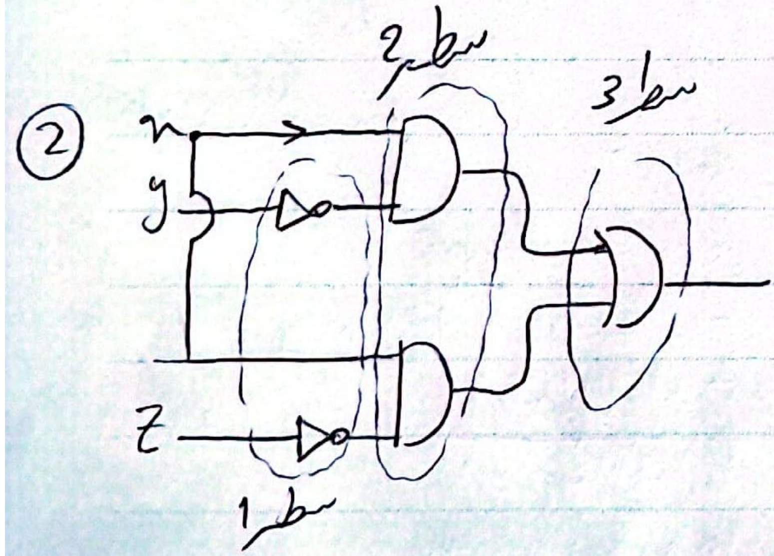
مثال: $f(x,y,z) = x(y'+z')$

① $f(x,y,z) = x(y'+z')$ دو گیت NOT ، یک گیت OR ، یک گیت AND

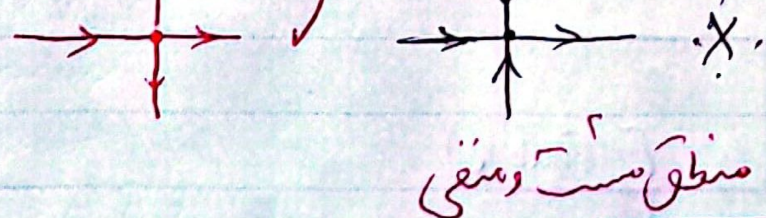
② $f(x,y,z) = xy' + xz'$ دو گیت NOT ، دو گیت AND ، یک گیت OR



نکته: ورودی به جای آنکه از دو x جدا استفاده کنیم، از انشعاب بهره ببریم



تذکر: انشعاب را طوری انتخاب می‌کنیم که اتصال کوتاه به وجود نیاید.



منطق مثبت و منفی

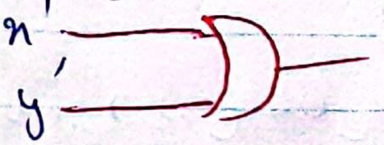
منطق مثبت : منطقی گفته می شود که از سیگنال های مثبت بهره برده باشد (منظور از سیگنال های مثبت خود گزاره است که اگر 1 باشد 1 و اگر 0 باشد 0 بر می گرداند)

منطق منفی : منطقی گفته می شود که از سیگنال های منفی بهره می برد (منظور از سیگنال های منفی، نقیض سیگنال های مثبت است)

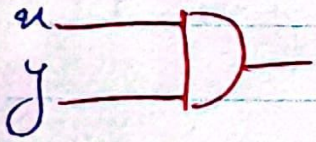
توجه داشته باشید که سیگنال ها، گزاره های ساده هستند.

مثال :

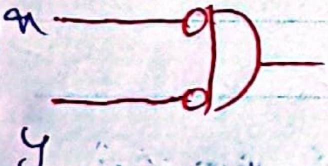
منطق منفی



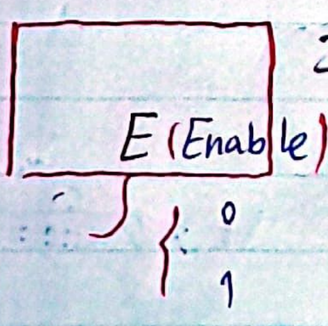
$x' + y' = (xy)' = NAND$



منطق مثبت



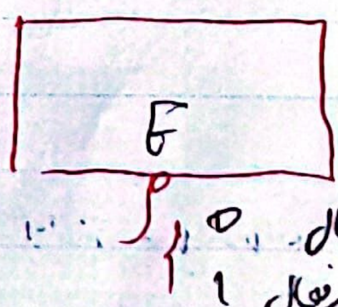
$(x' + y') = (xy)' = NAND$



منطق مثبت

0 غیرفعال

1 فعال



منطق منفی

0 فعال

1 غیرفعال

فصل سوم : ساده سازی گیت های منطقی

با توجه به درس های گذشته فهمیدیم که برای هر تابع می توان شکل های مختلفی را در نظر گرفت

اما باید بهترین تابع را با توجه به شرایط زیر انتخاب کرد :

- تعداد گیت های آن کمتر باشد
- دو فرمول با گیت یکسان دارای تعداد پایه کمتری باشد
- تعداد سطح های کمتری داشته باشد

روش های ساده سازی

1 استفاده از اصل ها و تئوری ها : مثال :

1: $f(x,y,z) = (((x(y'+z')) + y)x'$

$$(xy' + xz' + y)x' \overset{\text{توزیع}}{=} x'xy' + x'xz' + x'y \overset{\text{مکمل}}{=} x'y$$

2: $f(x,y,z) = x'y + xy' + xy \ \overset{\text{خاصیت تعددی}}{\underset{x+x=x}{=}}$

$$x'y + xy + xy' + xy \overset{\text{توزیع}}{=} y(x+x') + x(y+y') \overset{OR}{=}$$

$$y + x$$

روش ۱ همیشه پاسخگو نیست بنابراین در این قسمت به دو روش بعدی می پردازیم.

① جدول کارنو: تا 5 متغیره سازگاری کاربسته می شود. در این روش بیشترین موارد بررسی

② روش جدول بندی: برای هر n متغیری کاربرد دارد

1... : در این روش از کدگذاری بهره می بریم. ۲ تعداد متغیرها می پردازیم:

1: دو متغیره:

00	m_0	
01	m_1	
11	m_3	
10	m_2	

⇒

x \ y	y'	y
x'	m_0	m_1
x	m_2	m_3

⇒

x \ y	y'	y
x'	$x'y'$	$x'y$
x	xy'	xy

تذکر ۱: در کلیه جداول کارنو باید تنها 1 ها را دسته بندی کنیم (0 ها را نیز می توان) ولی 1 ها بهترند :)

بعد از دسته بندی

2^2 عدد 1 در جدول موجود است ⇐ باید 0 متغیره باشد

2^1 عدد 1 در جدول موجود است ⇐ باید 1 متغیره باشد

2^0 عدد 1 در جدول موجود است ⇐ باید 2 متغیره شود

⇐ برای هر خانه محسوب می شود.

تذکر ۲: هر دسته یک خانه محسوب می شود؛ مثلاً در دو متغیره 4 عدد 1 یک خانه با 4 عدد 1 محسوب می شود. همون 4 تایی خودمون :)

مثال ۱:

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

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

$\Rightarrow xy = F(x,y)$

تذکر ۲: قطری نمی توان دسته بندی کرد

مثال 2:

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

x \ y		y
	0	1
x	1	1

$F(x,y) = x + y$

مثال 3:

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

x \ y		y
	0	1
x	1	0

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

x \ y		y
	1	1
x	1	1

$\rightarrow \underbrace{x + x'}_{\text{دسته‌های دوتایی}} = \underbrace{y + y'}_{\text{دسته‌های دوتایی}} = \underbrace{x + x' + y + y'}_{\text{دسته ۴ تایی}} = 1$

2: سه متغیره:

000	m_0
001	m_1
011	m_3
010	m_2
110	m_6
111	m_7
101	m_5
100	m_4

x \ yz			y	
	m_0	m_1	m_3	m_2
x	m_4	m_5	m_7	m_6
			z	

2^3 : 8 عدد یک	صفر	متغیر
2^2 : 4 عدد یک	1	متغیر
2^1 : 2 عدد یک	2	متغیر
2^0 : 1 عدد یک	3	متغیر

مثال ۳:

$f(x,y,z) = \sum(0,1,2,4,5,7)$

$x \backslash yz$		y		
	1	1	0	1
x	1	1	1	0
		z		

$f(x,y,z) = y' + xz + x'z'$

نکته: تمامی جداول کارنو از سر و ته به هم متصل می‌شوند پس باید حواسمان به یک‌ها باشد

مثال ۴:

		B		
	1	1		1
A	1		1	1
		C		

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

3: چهار متغیره

0000	m_0
0001	m_1
0011	m_3
0010	m_2
0110	m_6
0111	m_7
0101	m_5
0100	m_4
1100	m_{12}
1101	m_{13}
1111	m_{15}
1110	m_{14}
1010	m_{10}
1001	m_9
1000	m_8

$wx \backslash yz$			y	
	m_0	m_1	m_3	m_2
x	m_4	m_5	m_7	m_6
w, x	m_{12}	m_{13}	m_{15}	m_{14}
w	m_8	m_9	m_{11}	m_{10}
		z		

تذکر: خانه‌ها به صورت مارپیچ گذاری می‌شوند (با رنگ سبز نمایش داده شده)

2^4 : 16 عدد یک 0 متغیره

2^3 : 8 عدد یک 1 متغیره

2^2 : 4 عدد یک 2 متغیره

2^1 : 2 عدد یک 3 متغیره

2^0 : 1 عدد یک 4 متغیره

نکته، در جدول کارنو ترتیب اهمیت دارد یعنی $f(w,x,y,z)$ با

$f(x,y,z,w)$ متفاوت است.

مثال 11

$wx \backslash yz$				
	1	1	1	
	1	1	1	1
	1	1		
		1		1

$f(w,x,y,z) = y'z + w'z + w'x + w'y' + xy' + wx'yz'$

مثال 12

$wx \backslash yz$				
	1	1	1	1
	1			1
	1			1
	1	1	1	1

$f(w,x,y,z) = x' + z'$

مثال 13

$AB \backslash CD$				
	1	1	1	1
	1	1	1	
	1			1

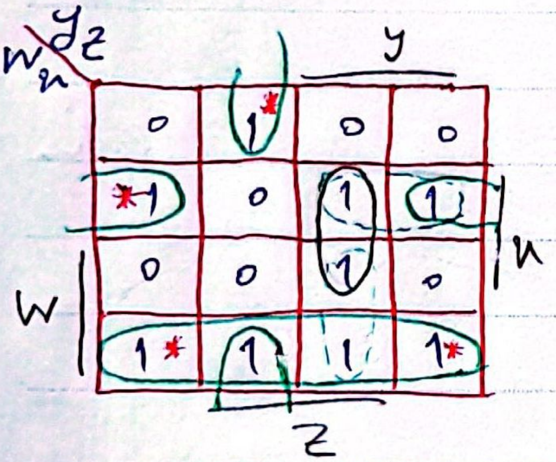
$f(A,B,C,D) = A'B' + B'D' + ABD + ABC'$

انتخاب تحیین

① تحیین: دسته‌های به جز دسته‌ی اصلی را تحیین گوییم. از بین آنها یکی انتخاب می‌شود

② تحیین اصلی: دسته‌ای که حداقل یک عنصر منحصر به فرد دارد (منظور از منحصر به فرد آن است که مثلاً اگر دسته‌ای 4 تایی انتخاب کردیم حداقل یک عنصر آن با عناصر دیگر جدول دسته‌ی 4 تایی نسازد) را در مرتبه‌ی اول انتخاب می‌کنیم.

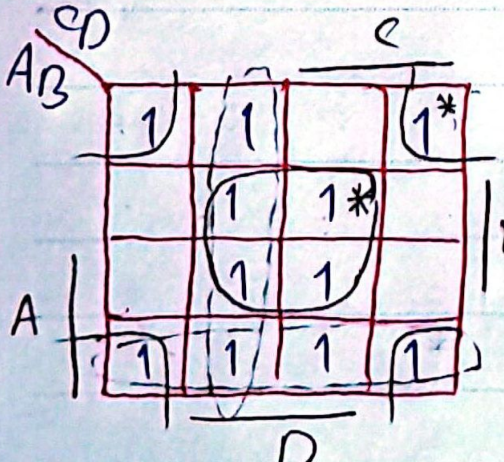
مثال ۱:



راهنما
- تحیین اصلی
- تحیین
- منحصر به فرد *
- انتخاب تحیین

$$f(w,x,y,z) = wx' + w'xz' + x'y'z + xyz$$

مثال ۲:



$$f(A,B,C,D) = BD + B'D' + AB' + C'D$$
$$= BD + B'D' + AB' + C'B'$$
$$= BD + B'D' + AD + C'B'$$
$$= BD + B'D' + AD + C'D$$

نکته: تعداد کل دسته‌های n تایی 2^{2^n}

نکته: بدترین حالت انتخاب زمانی است که یکی در میان 1 داشته باشیم در این حالت

AB \ CD

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

هر یک یک دسته : $A \oplus B \oplus C \oplus D$

برای صفرها : $A \odot B \odot C \odot D$

دسته‌بندی صفرها

برای دسته بندی صفرها دو راه داریم: ① ابتدا به همان شیوه 1 ها دسته بندی کرده

پس جواب حاصل که حاصلجمع ضرب‌ها بوده است را مکمل می‌کنیم (حاصلضرب جمع‌ها)

② در جدول n را به n' و ... z را به z' تغییر می‌دهیم

مثال 1:

AB \ CD (C, B, A, D)

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

$$f'(A,B,C,D) = BD' + A'B'CD \Rightarrow$$

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

NOTE BOOK — Year — Month — Date

مثال ۲:

AB \ CD				
	0	1	0	1
	0	1	0	0
A'	0	1	0	0
	0	1	1	0

C, B', D'

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

معرفی حالت های بی اهمیت

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	X
0	1	0	0	X
0	1	0	1	X
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	X
1	1	0	0	1
1	1	0	1	X
1	1	1	0	X
1	1	1	1	1

دسته بندی صفرها:

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

* در حالت بی اهمیت باید جاهایی که X هستند را به نحوی پر کنیم که گروه بندی بهتری داشته باشیم.

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

دسته‌بندی 1 ها

AB \ CD			C	C
	0	0	X	1
B	X	X	0	0
A B	1	X	1	X
A	1	0	X	1
		D	D	

$f: AB + AD' + CB'$

حالت بی اهمیت به صورت Σ مثال

$$f(A,B,C,D) = \Sigma(2,8,10,12,15) = d(3,4,5,11,13,14)$$

$$f(A,B,C,D) = \Pi(0,1,6,7,9) = d(3,4,5,11,13,14)$$

14، پنج متغیره

A'

		D	D	
	m_0	m_1	m_3	m_2
C	m_4	m_5	m_7	m_6
B, C	m_{12}	m_{13}	m_{15}	m_{14}
B	m_8	m_9	m_{11}	m_{10}
		E	E	

A

		D	D	
	m_{16}	m_{17}	m_{19}	m_{18}
C	m_{20}	m_{21}	m_{23}	m_{22}
B, C	m_{28}	m_{29}	m_{31}	m_{30}
B	m_{24}	m_{25}	m_{27}	m_6
		E	E	

$f=1$ 0 متغیره 32 عدد تک : 2^5

1 متغیره 16 عدد تک ؛ 2^4

2 متغیره 8 عدد تک ؛ 2^3

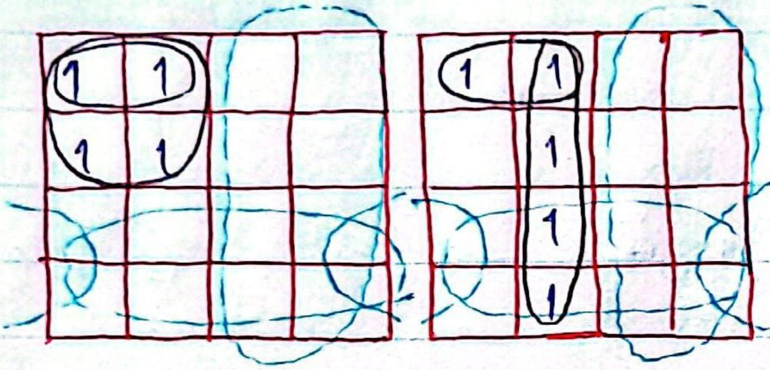
3 متغیره 4 عدد تک ؛ 2^2

4 متغیره 2 عدد تک ؛ 2^1

5 متغیره 1 عدد تک ؛ 2^0

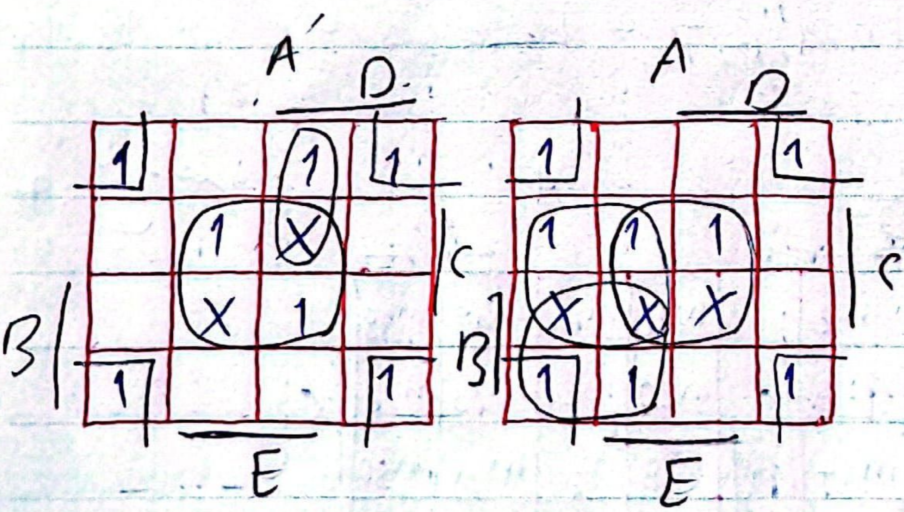
نکته : برای دسته بندی هر تعدادی که متناظر یکدیگر روی "مربع قرار دارند مجاز هم هستند.

مثال ۱:



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

مثال ۲:



$$f = CE + C'E' + ACD' + AB'D' + A'B'DE$$

2. ـــــ : در این روش براساس تعداد ۱ها طبقه‌بندی کرده و هر طبقه را با طبقه‌ی پایینی خودش مقایسه می‌کنیم درصورتی که اختلاف اعداد کناری در این روش ساده کردن تفاوتی از جمله می‌توان ساده نمود در نتیجه موارد جاخالی‌ها و بیت‌ها مقایسه می‌شود.

لذا برای درک کامل به مثال زیر توجه کنید:

مثال ۱: $f(A,B,C,D) = \sum(0,1,2,4,5,6,7,8,12,13,14,15)$

0	0000	✓
1	0001	✓
2	0010	✓
4	0100	✓
8	1000	✓
5	0101	✓
6	0110	✓
12	1100	✓
7	0111	✓
13	1101	✓
14	1110	✓
15	1111	✓

0,1	000–	✓
0,2	00–0	✓
0,4	0–00	✓
0,8	–000	✓
1,5	0–01	✓
2,6	0–10	✓
4,5	010–	✓
4,6	01–0	✓
4,12	–100	✓
8,12	1–00	✓
5,7	01–0	✓
5,13	–101	✓
6,7	011–	✓
6,14	–110	✓

12,13	110–	✓
12,14	11–0	✓
7,15	–111	✓
13,15	11–1	✓
14,15	111–	✓

0,1,4,5	0–0–	
0,2,4,6	0––0	
0,4,8,12	––00	
4,5,6,7	01––	✓
4,5,12,13	–10–	✓
4,6,12,14	–1–0	✓

5,7,13,15	_1_1
6,7,14,15	_11_
12,13,14,15	11__

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

	ABCD
4,5,6,7,12,13,14,15	_1__

مثال 2:

$f(A,B,C,D) = \Sigma(0,1,2,5,7,8,9,10,11,13,15)$

0	0000 ✓
1	0001 ✓
2	0010 ✓
8	1000 ✓
5	0101 ✓
9	1001 ✓
10	1010 ✓
7	0111 ✓
11	1011 ✓
13	1101 ✓
15	1111 ✓

0,1	000_ ✓
0,2	00_0 ✓
0,8	_000 ✓
1,5	0_01 ✓
1,9	_001 ✓
2,10	_010 ✓
8,9	10_0 ✓
5,7	01_1 ✓
5,13	_101 ✓
9,11	10_1 ✓
9,13	1_01 ✓
10,11	101_ ✓
7,15	_111 ✓
11,15	1_11 ✓
13,15	11_1 ✓

	ABCD
0,1,8,9	_00_
0,2,8,10	_0_0
1,5,9,13	__01
8,9,10,11	10__
5,7,13,15	_1_1
9,11,13,15	1__1

f	0	1	2	5	7	8	9	10	11	13	15
$B'C'$	✓	✓				✓	✓				
$B'D'$	✓		✓			✓	✓				
$C'D$		✓		✓		✓				✓	
$A'B'$						✓	✓	✓	✓		
BD				✓	✓					✓	✓
AD							✓		✓	✓	✓
All	✓	O	✓	✓	✓	✓	✓	O	O	✓	✓

نکته: اگر تک تیکی بود آنگاه قطعاً اصلی است و باید کلیه تیک ها را بزنیم و برای مربعی ها باید دنبال تکمیل All باشیم.

حل: $f(A,B,C,D) = B'D' + BD + C'D + A'B'$

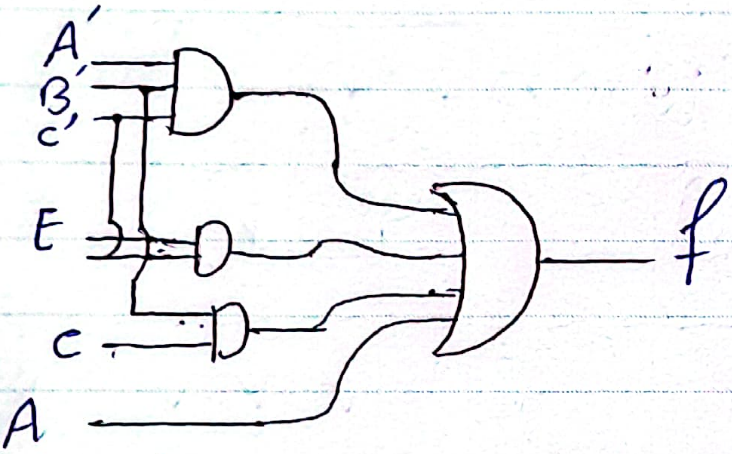
$= B'D' + BD + A'B' + B'C'$

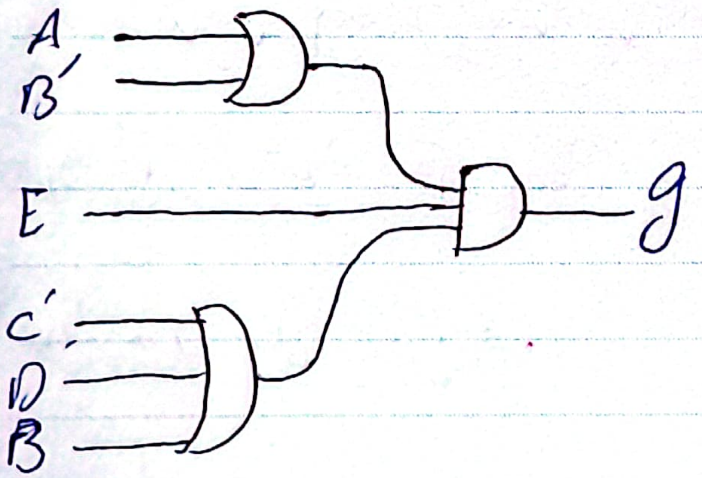
$=$

پیاده سازی NOR , NAND

$f = A + B'C' + EC' + A'B'C'$

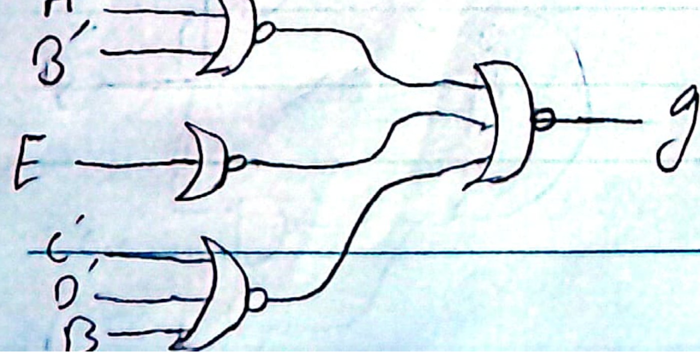
$g = (A + B') \cdot E \cdot (C' + D' + B)$



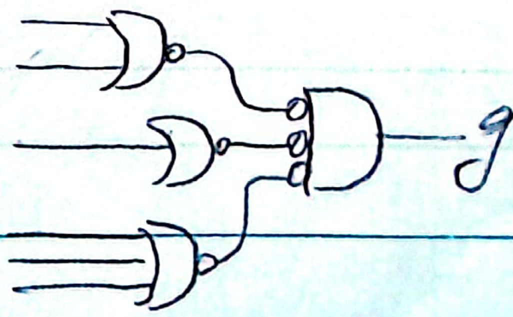


$g \Leftarrow$ NAND , NOR

روش اول



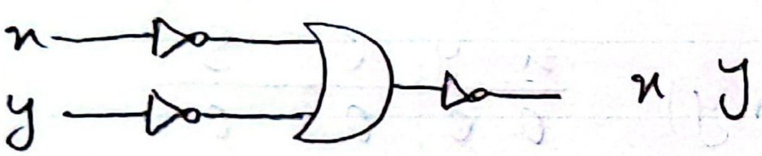
روش دوم



نکته: با موارد زیر می‌توان به تنهایی می‌توان همه توابع را بازنویسی کرد:

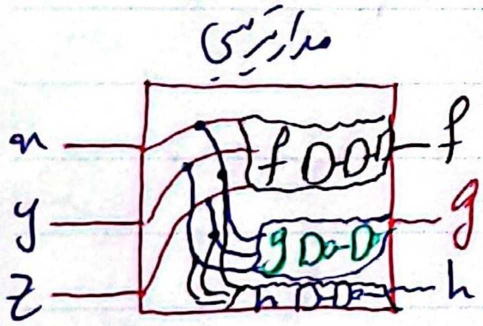
$\{AND, NOT\}$، $\{OR, NOT\}$، $\{NOR\}$، $\{NAND\}$

مثال: $(x'+y')' = x.y$



فصل 4: مدارهای ترکیبی منطقی

Boolean



$f(x,y,z) = \ldots$

$g(x,y,z) = \ldots$

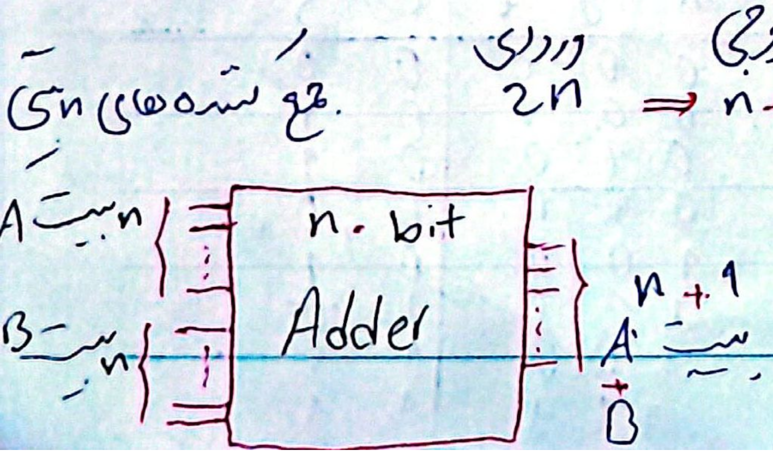
$h(x,y,z) = \ldots$

در مدارهای ترکیبی در جعبه مدار معلوم نیست که چه اتفاقی قرار است بیافتد. تنها اسم

که ترکیبی از گیت‌های مختلف است.

جمع کننده‌ها MSI

جمع کننده‌های n بیتی. ورودی $2n \Rightarrow$ خروجی $n+1$



$A = A_{n-1}A_{n-2}\cdots A_0$

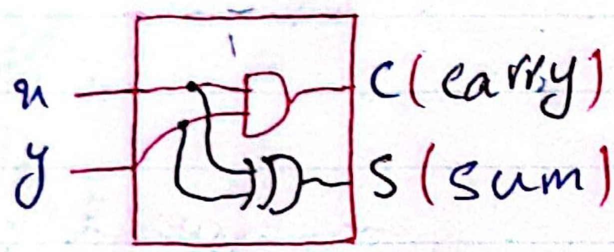
$B = B_{n-1}B_{n-2}\cdots B_0$

انواع جمع کننده ها

① Half Adders (نیم جمع کننده ها)

② Full Adders (تمام جمع کننده)

1.--



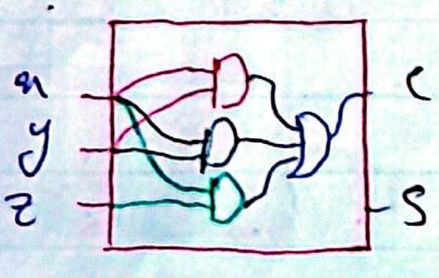
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$C = x.y = AND$

$S = x'y + y'x = XOR \; x \oplus y$

نکته: برای جمع دو به دو ما علاوه بر H.A نیاز به F.A داریم زیرا 3 عدد می شود پس قبل از مثال:

2....



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

$x \backslash yz$			y	y
	0	0	1	0
x	0	1	1	1
		z	z	

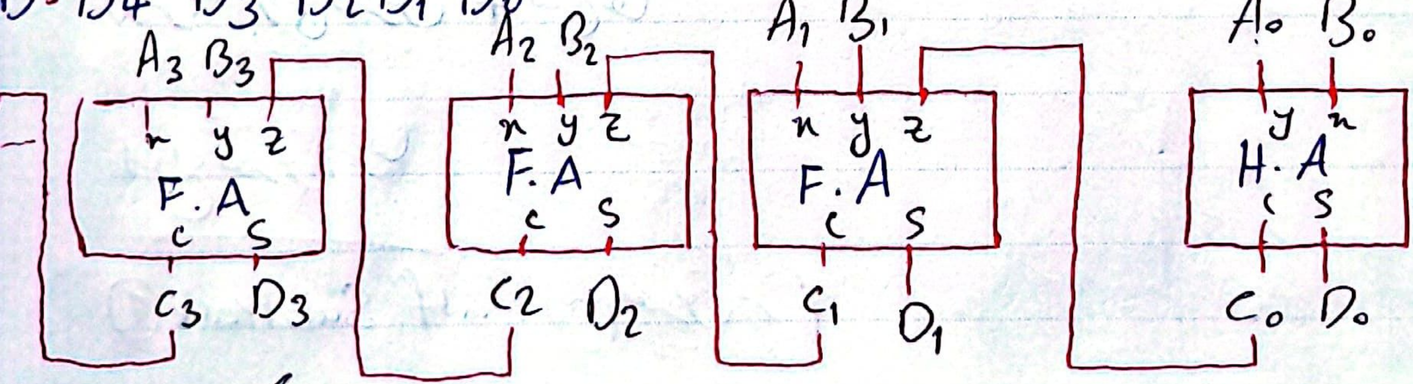
$$C = xz + yz + xy$$

0	1	0	1
1	0	1	0

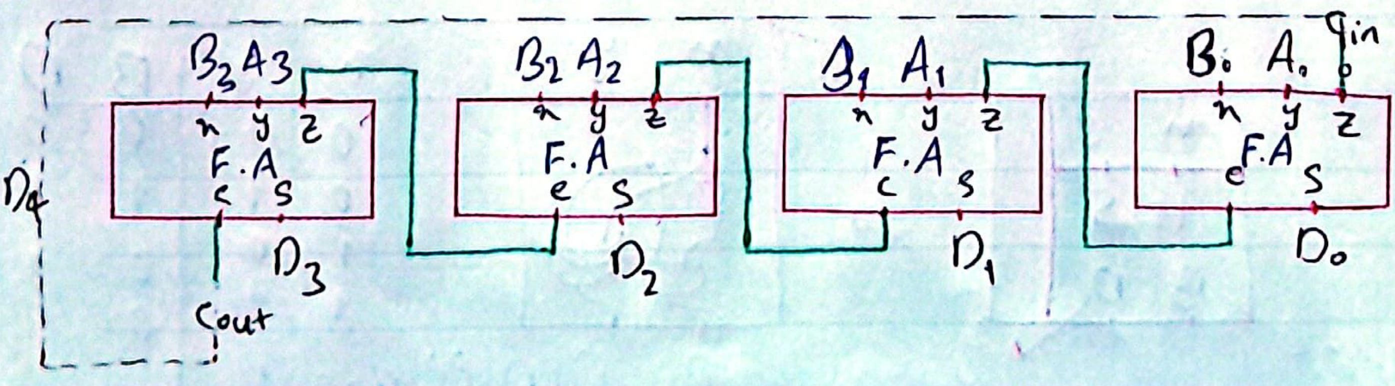
$$S = x \oplus y \oplus z$$

در حالت کلی

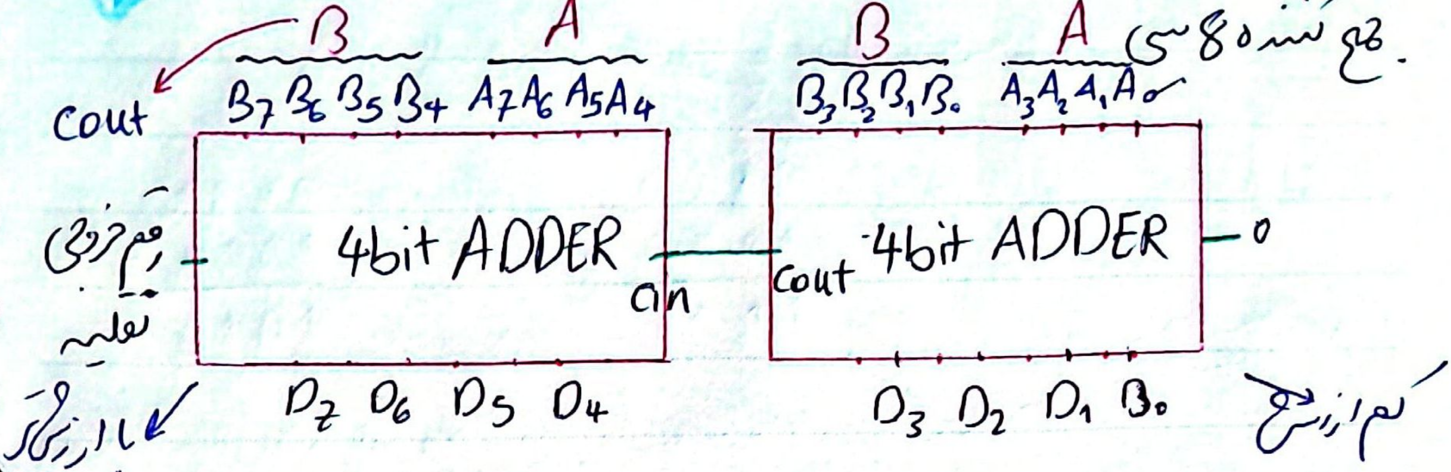
$$A = A_3 A_2 A_1 A_0$$
$$B = B_3 B_2 B_1 B_0$$
$$D = D_4\, D_3\, D_2\, D_1\, D_0$$



نکته: می توانیم به جای H.A اولیه از F.A نیز به صورت زیر استفاده کنیم:



جمع کننده 8 بیتی



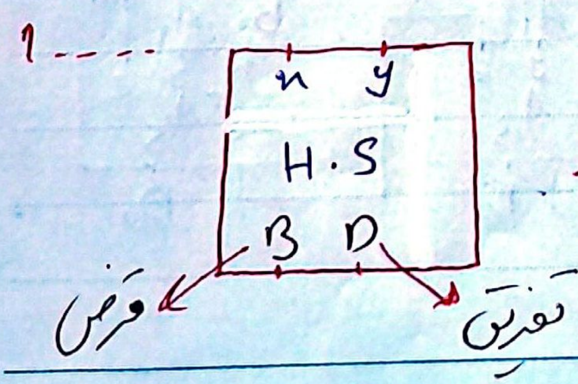
$A: \boxed{A_7 \dots A_4} \boxed{A_3 \dots A_0}$

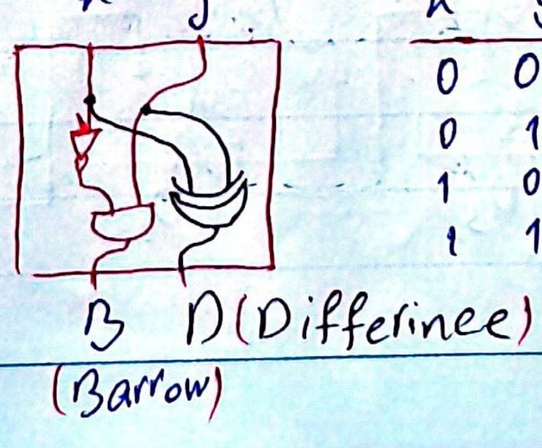
$B: \boxed{B_7 \dots B_4} \boxed{B_3 \dots B_0}$

انواع تفریق گرها

① Half Subtrack (نیم تفریق گر)

② Full Subtrack (تمام تفریق گر)



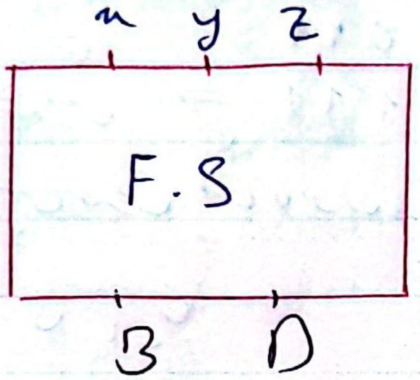


x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

$B = x'y$ $\quad y / x$

$D = x \oplus y$

2 ---- 1



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

x \ yz			y	y
	0	1	1	1
x	0	0	1	0
		z	z	

$\Rightarrow B = xy + x'y + x'z$

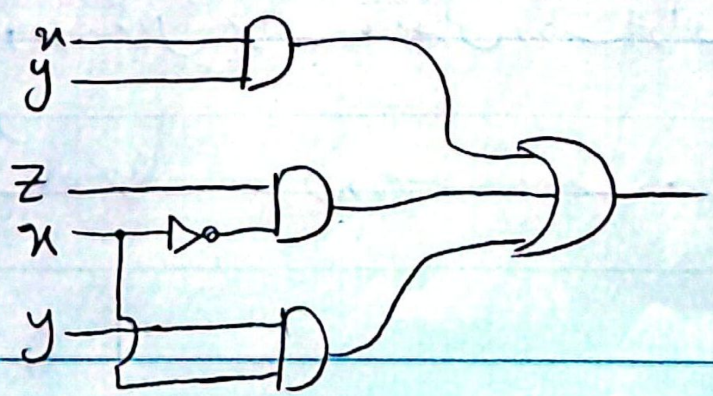
$B = z / y / x$

x \ yz			y	y
	0	1	0	1
x	1	0	1	0
		z	z	

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

$D = x \oplus y \oplus z$

B ,[illegible] 1

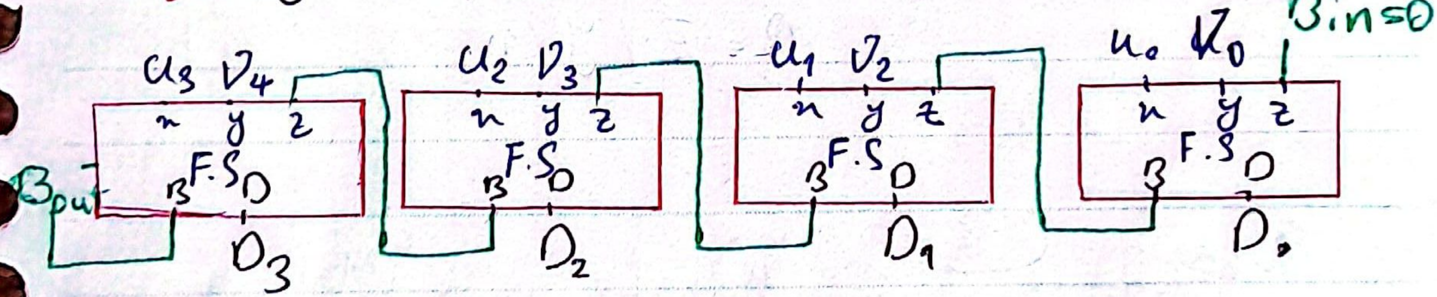


قبل از آنکه به سراغ تفریق های 3 بیتی برویم طبق مثال داریم:

$$\begin{array}{r} 0\,2\,0\,2 \\ \cancel{1}\,\cancel{0}\,\cancel{1}\,\cancel{0} \\ -\;0\,1\,0\,1 \\ \hline 0\,1\,0\,1 \end{array} \qquad \begin{array}{r} 2\;\;\; \\ 1\,2\,0\,2 \\ \cancel{0}\,\cancel{0}\,\cancel{1}\,\cancel{0} \\ -\;0\,0\,1\,1 \\ \hline 1\,1\,1\,1 \end{array}$$

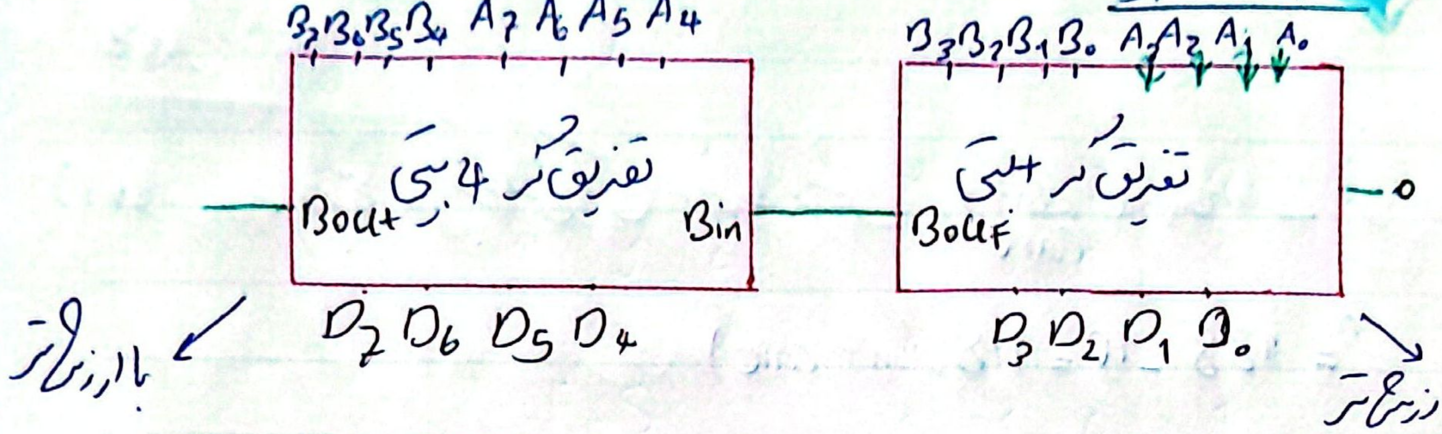
⇐ در این حالت چون همه اعداد 1 بدست آمده پس تفریق اشتباه بوده پس باید جای عدد دوم و اول را عوض کرده و حاصل را منفی کنیم پس:

$$\begin{array}{l} U = u_3\,u_2\,u_1\,u_0 \\ V = v_3\,v_2\,v_1\,v_0 \\ \hline D = D_3\,D_2\,D_1\,D_0 \end{array}$$



نکته: اگر Bout ۱ باشد آنگاه می فهمیم که نادرست بوده و با جابجا کردن V , U و قرینه کردن جواب، تفریق درست خواهد شد. اما اگر Bout 0 باشد آنگاه جواب تفریق درست است.

تفریق گر 8 بیتی



تذکر: برای تفریق 4 بیتی ها ابتدا 4 بیتی های با ارزش کمتر و سپس 4 بیتی های با ارزش بیشتر هرکدام را بررسی می کند که حاصل تفریق درست باشد (جواب Bout اگر 0 شود) لذا اگر درست نباشد باید جای A ها و B ها را عوض کنیم و سپس با استفاده از Bin را به Bout وصل می کنیم (اول و آخر)

الگوریتم طراحی مدار (برای 1_4 ورودی)

1. تعریف مسئله 2. تعیین تعداد ورودی ها و خروجی ها

3. نامگذاری ورودی ها و خروجی ها 4. محاسبه جدول درستی براساس تعریف مسئله

5. بدست آوردن ساده ترین رابطه برای هر خروجی (به کمک جدول کارنو یا جدول بندی)

6. رسم مدارهای منطقی

مبدل کد

BCD ← دودویی (برعکس آن به شرط نوشتن don't care تمام جدول است زیرا

$2^7 = 128 - 10 = 118$ don't care)

BCD ← 2421 (برعکس آن باید مدار دیگری طراحی شود)

BCD A B C D	2 4 2 1 f_4 f_3 f_2 f_1
0 0 0 0	0 0 0 0
0 0 0 1	0 0 0 1
0 0 1 0	0 0 1 0
0 0 1 1	0 0 1 1
0 1 0 0	0 1 0 0
0 1 0 1	1 0 1 1
0 1 1 0	1 1 0 0
0 1 1 1	1 1 0 1
1 0 0 0	1 1 1 0
1 0 0 1	1 1 1 1

AB \ CD			C	C
	0	0	0	0
B	0	1	1	1
A B	X	X	X	X
A	1	1	X	X
		D	D	

$f_4 = A + BD + BC$

AB \ CD			C	C
	0	0	0	0
B	1	0	1	1
A B	X	X	X	X
A	1	1	X	X
		D	D	

$f_3 = A + BC + BD'$

AB \ CD			C	C
	0	1	1	0
B	0	1	1	0
A B	X	X	X	X
A	0	1	X	X
		D	D	

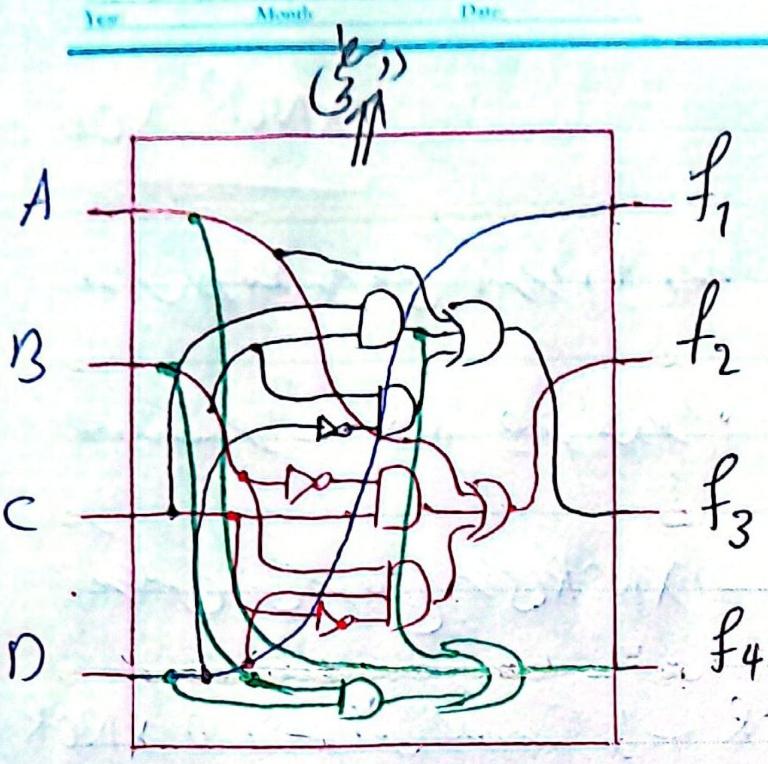
$f_1 = D$

AB \ CD			C	C
	0	0	1	1
B	0	1	0	0
A B	X	X	X	X
A	1	1	X	X
		D	D	

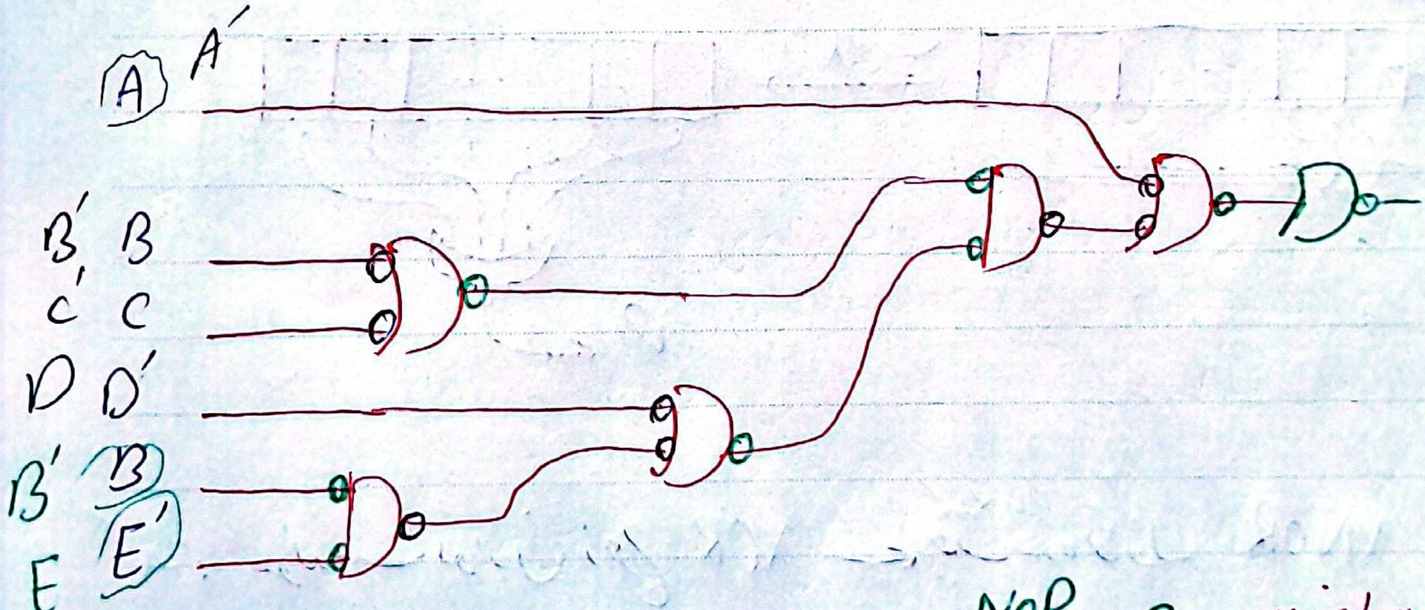
$f_2 = A + \bar{B}C + BC'D$

نکته: دقت شود که چون فقط از 0 تا 9 وجود دارد باید از 10 تا 15 X شود.

مدار منطقی:



تبدیل: NAND , NOR



NOR o

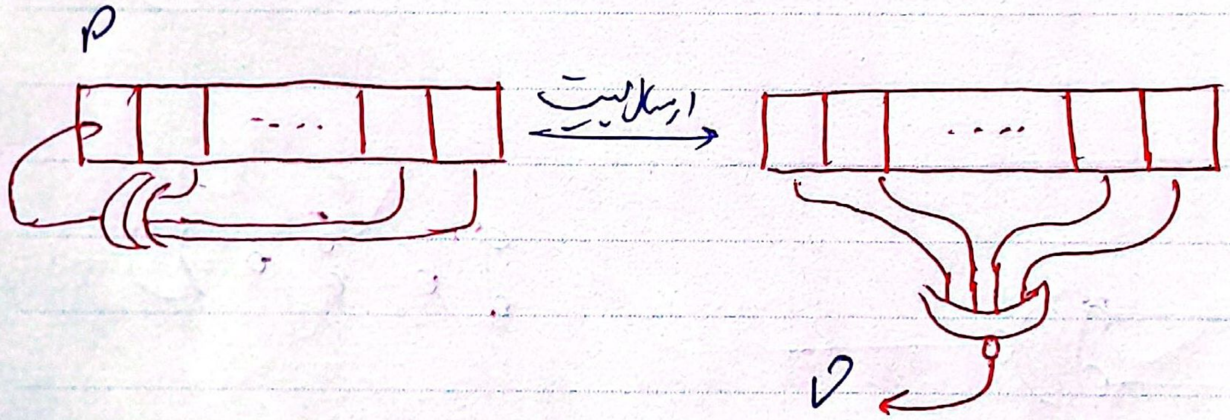
NAND o

* راهنمایی

گیت XOR , XNOR

در این مدار منطقی اگر فرد تا 1 وجود داشته باشد جواب XOR 1 در غیر این صورت جواب XOR 0 خواهد بود.

برعکس این موضوع برای XNOR است یعنی اگر تعداد 1 ها فرد باشد جواب XNOR 0 در غیر این صورت جواب XNOR 1 می‌شود. حال می‌توان از آن‌ها برای آشکارسازی خطا استفاده کرد.



نکته: مولد بیت توازن زوج XOR ← چک‌کننده توازن XNOR

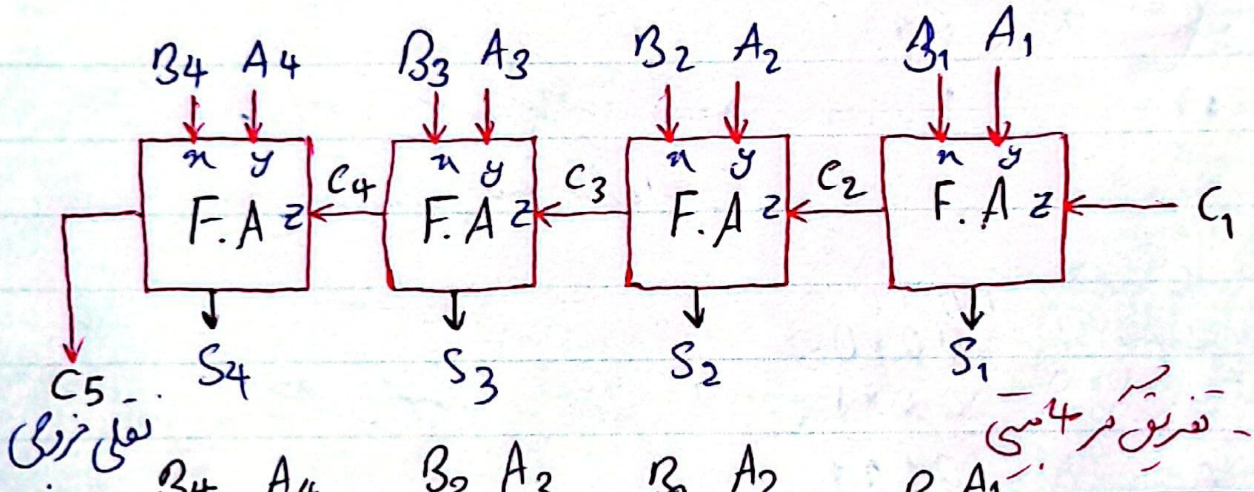
- - - - - - مرز XNOR ← $\underbrace{\text{- - - - - XOR}}_{n+1 \text{ بیت}}$

فصل 5 ، مدارهای ترکیبی :

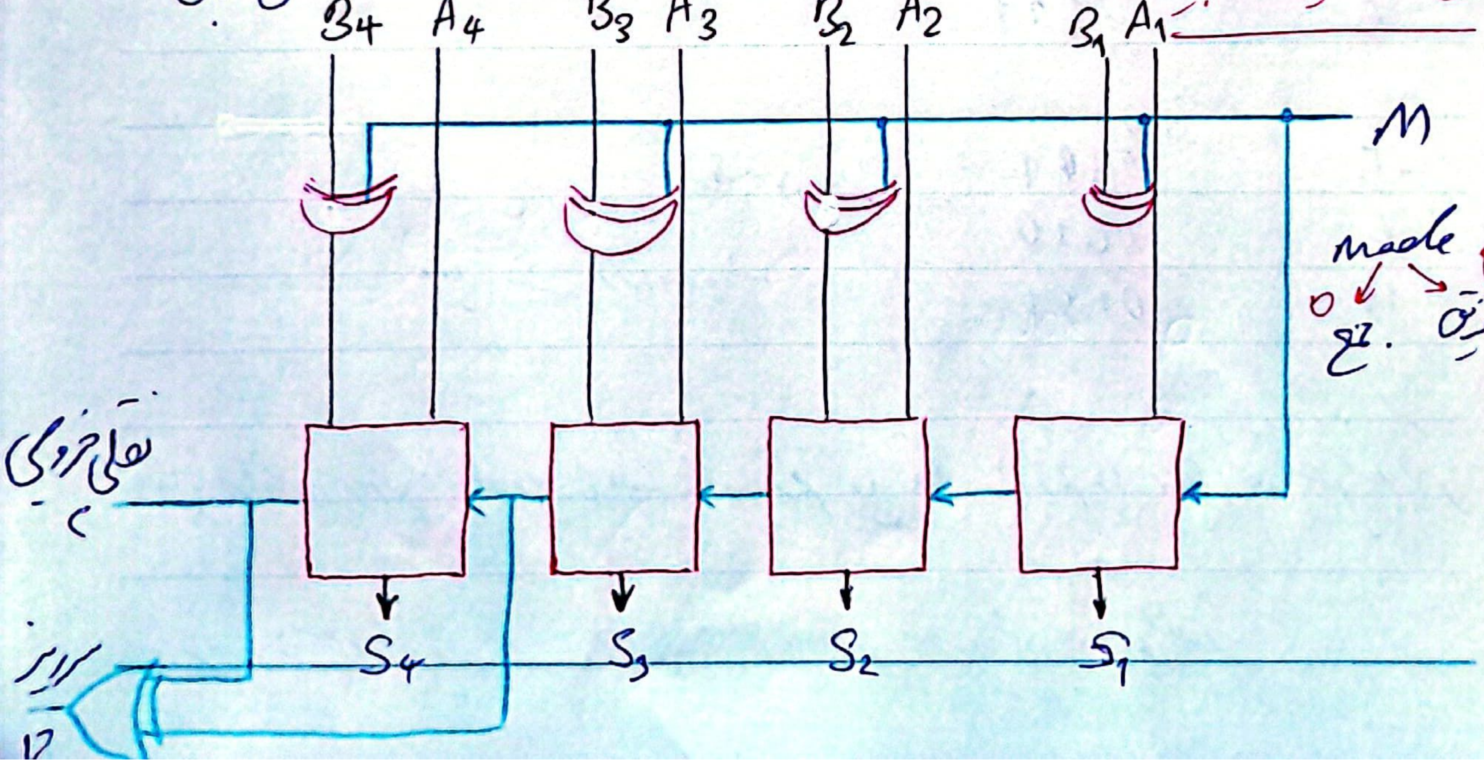
- به جدول زیر توجه کنید

	4	3	2	1		نام جمع کننده
نقلی ورودی	0	1	1	0	: C_i	z
	1	0	1	1	: A_i	x
+	0	0	1	1	: B_i	y
مجموع :	1	1	1	0		S
نقلی خروجی	0	0	1	1	: C_{i+1}	C

. جمع کننده 4 بیتی



. جمع - تفریق گر 4 بیتی



NOTE BOOK

if $M=0 \rightarrow A+B+0=A+B$ $\quad M=0$ حاصل عملیات درست

if $M=1 \rightarrow A+B'+1=A-B$ $\quad M=1$ حاصل عملیات نادرست

سرریز

سرریز زمانی اتفاق می افتد که مثلاً جمع دو عدد مثبت منفی شود و معقول نباشد

رقم نهایی ← 0100 → رقم قبلی

$$\begin{array}{r} +5 \\ + \; +6 \\ \hline 11 \end{array} \qquad \begin{array}{r} 0101 \\ + \; 0110 \\ \hline 1011 \end{array}$$

⇐ مثلاً معقول نیست چرا که جمع دو عدد مثبت منفی شده است.

رقم نقلی قبلی = 1 ، نهایی = 0 $\Rightarrow 1 \neq 0$ سرریز

0010

$$\begin{array}{r} 2 \\ + \; 3 \\ \hline 5 \end{array} \qquad \begin{array}{r} 0010 \\ 0011 \\ \hline 0101 \end{array}$$

⇐ نهایی = 0 ، قبلی = 0 $\Rightarrow 0=0$ سرریز ندارد
معقول است

نهایی = 1 ، قبلی = 0 $\Rightarrow 1 \neq 0$ سرریز

1010

$$\begin{array}{r} -5 \\ -6 \\ \hline -11 \end{array} \qquad \begin{array}{r} 1011 \\ 1010 \\ \hline 0101 \end{array}$$

⇐ معقول نیست چرا که جمع دو عدد منفی مثبت درآمده است

حال برای تشخیص سرریز کافیست که رقم نقلی قبلی را با رقم نقلی نهایی مقایسه کنیم اگر برابر بودند پس سرریز ندارد در غیر این صورت سرریز اتفاق می افتد

حال این روش هم از آن جهت ناخوشایند است که هر F.A باید منتظر باشد تا رقم نقلیه را از F.A قبلی حساب شود و به این باید تا حاصل عملیات بدست آید.

در واقع مراحل عملیات نیازمند یکدیگرند و زمان زیادی مصرف می شود.

راه حل: اگر بتوانیم رقم نقلیه در هر مرحله را مستقیماً به F.A بدهیم مشکل حل خواهد شد

داریم:

$P_i = A_i \oplus B_i \qquad S_i = P_i \oplus C_i$

$G_i = A_i B_i \qquad C_{i+1} = G_i + P_i C_i$

$C_2 = G_1 + P_1 C_1$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1) = G_2 + P_2 G_1 + P_2 P_1 C_1$

$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$

مقایسه گر دو مقدار ۴ بیتی

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

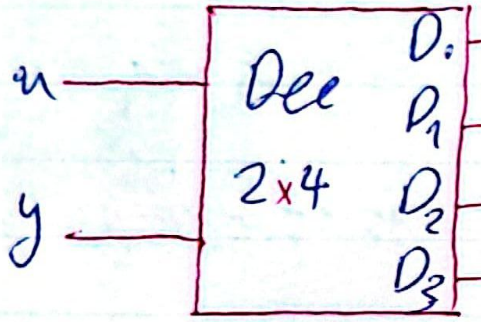
$B > A : g = A_4' B_4 + x_4 A_3' B_3 + x_4 x_3 A_2' B_2 + x_4 x_3 x_2 A_1' B_1$

$A = B : h = x_4 x_3 x_2 x_1$

نکته:

$A_3 \odot B_3 = (A_3B_3' + A_3'B_3)' \;\Big\}\; x_i = A_i \odot B_i$

decoder دیکدر



Enable

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

منطق مثبت

ورودی خروجی

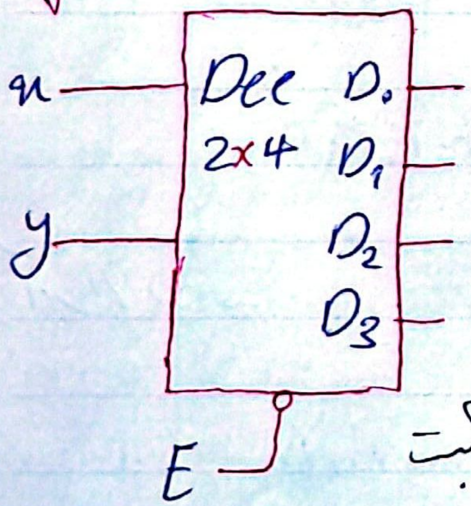
$n \times 2^n$

$x = D_2 + D_3$

$y = D_1 + D_3$

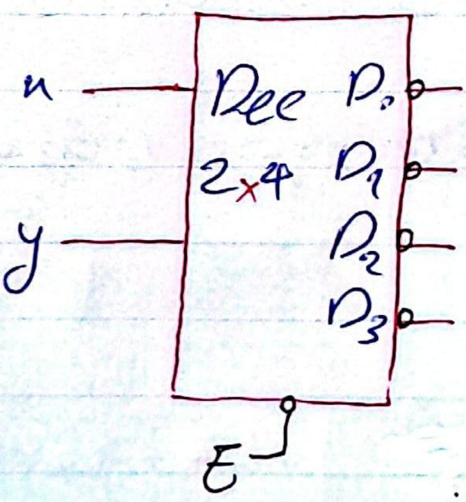
$D_0 = x'y'$ $\qquad D_2 = xy'$

$D_1 = x'y$ $\qquad D_3 = xy$



حتماً

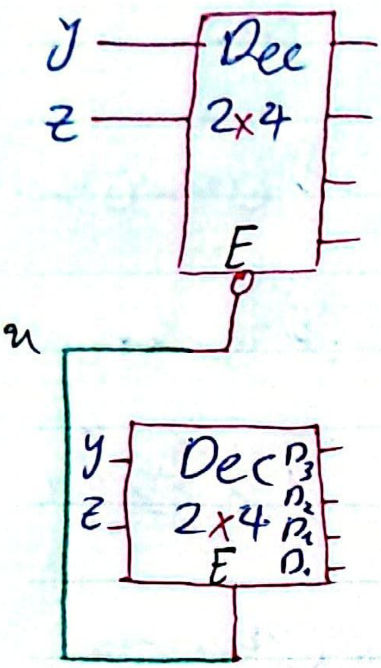
منطق مثبت



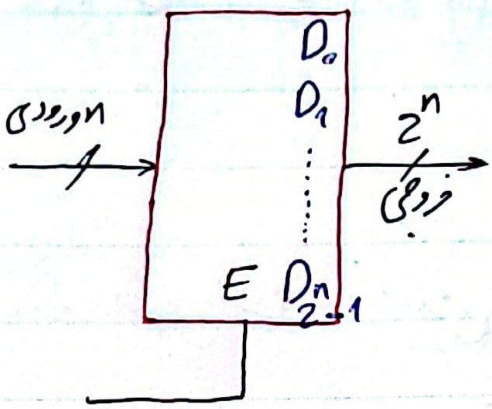
منطق منفی

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

ترکیب 2x4 Dec افزون ساز



در حالت کلی:

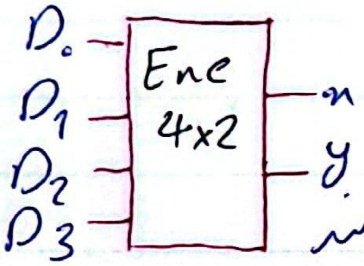


سه بیتی

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

انکدر Encoder

اگر جای ورودی و خروجی در Dec عوض شود آنگاه به Enc تبدیل می‌گردد



این مدار دو مشکل دارد:

1: اگر هیچ یکی از ورودی‌ها نباشد خروجی اشتباه تولید می‌کند

2: اگر در ورودی‌ها دو تا 1 هم باشد باز هم اشتباه می‌افتد

فقط برای یکی 1 درست کار می‌کند، لذا برای رفع مشکل معرفی انکدر با اولویت

از سمت راست ←

	D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	0	X	X	0
1	1	0	0	0	0	0	1
2	X	1	0	0	0	1	1
4	X	X	1	0	1	0	1
8	X	X	X	1	1	1	1

نکته 1: تمامی 16 حالت در اینجا وجود دارد

$1 + 2 + 4 + 8 = 16$

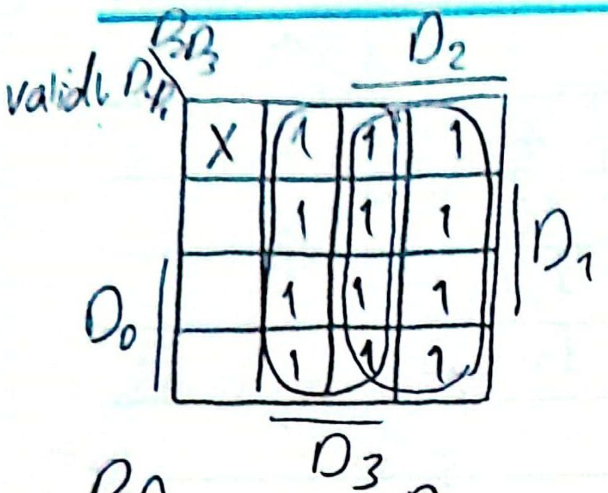
نکته 2: برای مشکل صفر نشدن تمام D به سراغ انکدر با اولویت می‌رویم.

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

بدون Valid

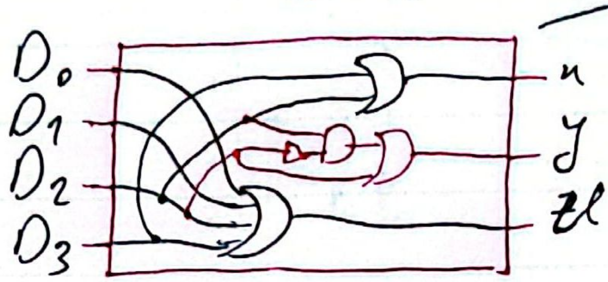
$x = D_2 + D_3$

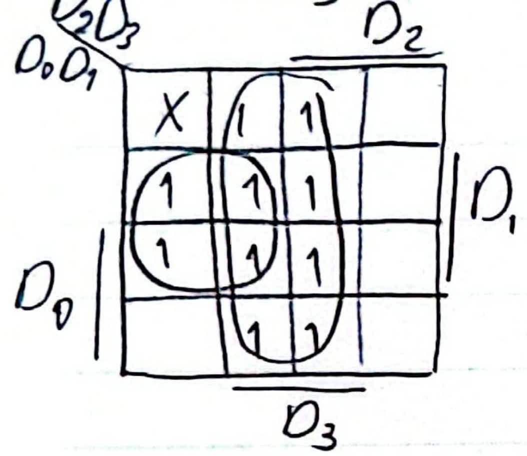
$y = D_1 + D_3$



$$\Longrightarrow\ x = D_2 + D_3$$

با الویت





$$\Longrightarrow\ y = D_2 + D_1 D_2'$$

$$V = D_0 + D_1 + D_2 + D_3$$

مالتی پلکسر multiplexer

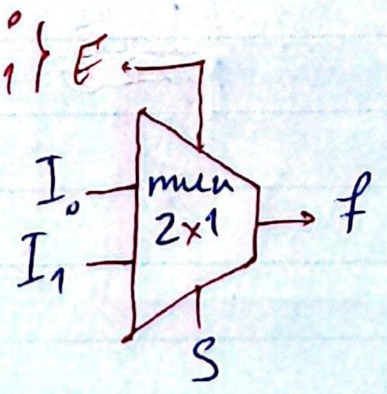
هرگاه بخواهیم از بین ورودی ها یکی را به خروجی ببریم از multiplexer استفاده می کنیم.

ورودی $2^n \times 1$ خروجی

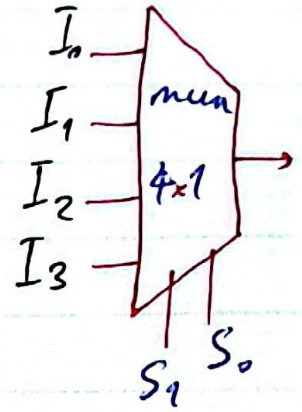
با توجه به اینکه n خط Select نیز در دسترس $2^n + n$ ورودی داریم

مثل قبل $1 \, 2 \, E$

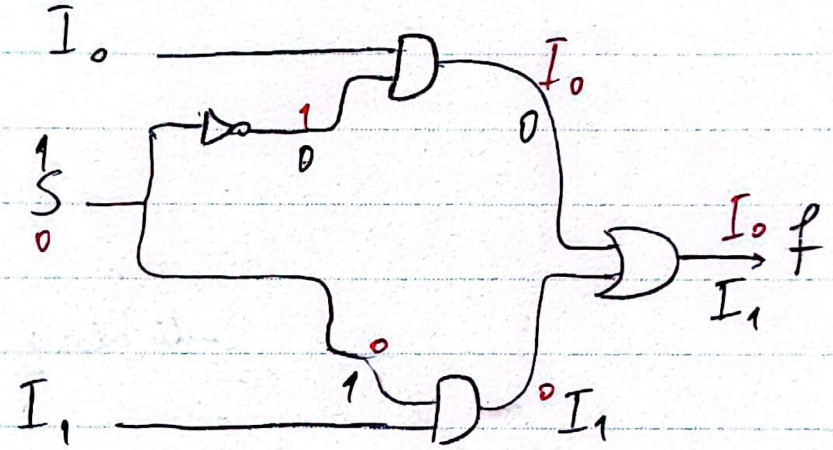
	S	F
براساس	0	I_0
خطانی ب	1	I_1



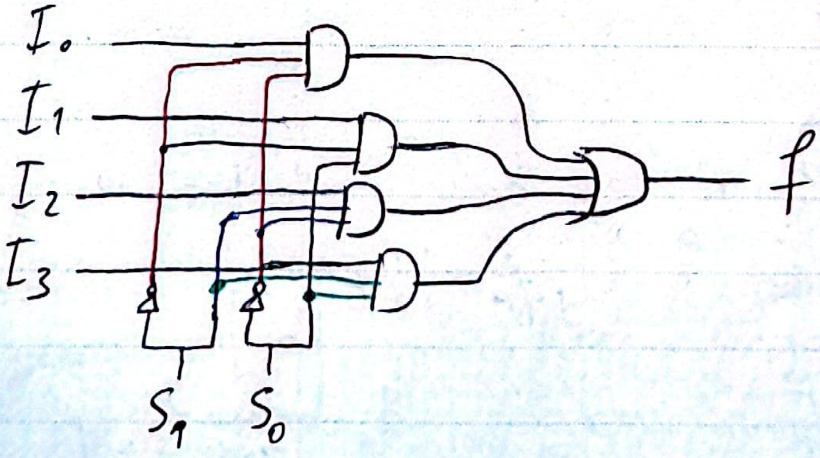
S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



اولی :

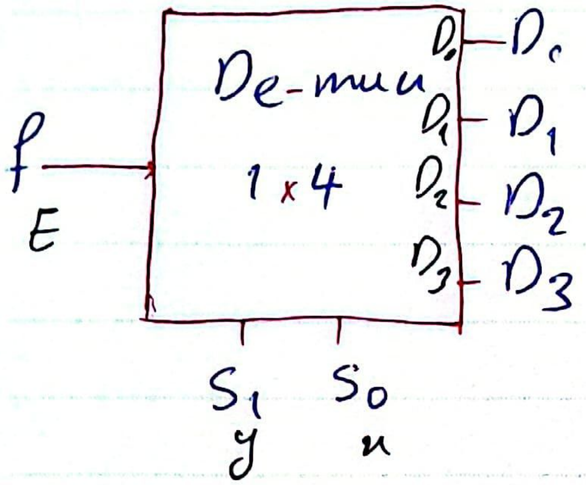


دری



Demux دی مالتیپلکسر

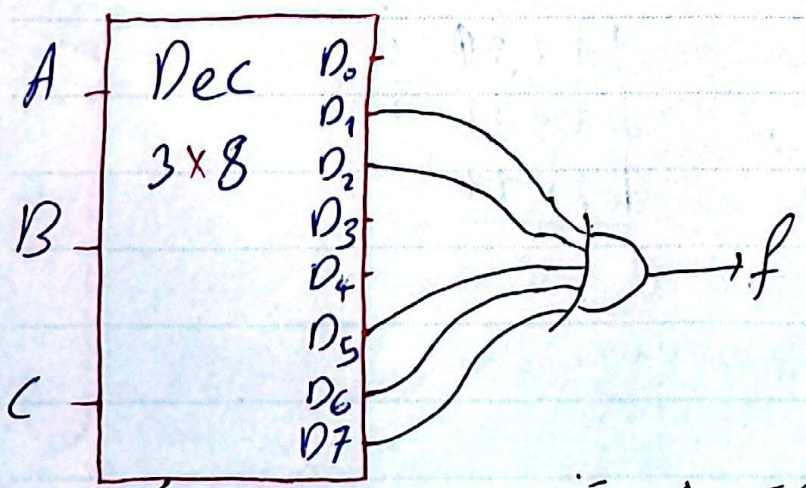
De-mux ≡ Decoder با خط E



پیاده سازی توابع با استفاده از Dec / mux

1: استفاده از Dec برای چند تابع با ورودی یکسان (مثلاً $f(A,B,C)$, $g(A,B,C)$)

$$f(A,B,C) = \Sigma(1,2,5,6,7)$$

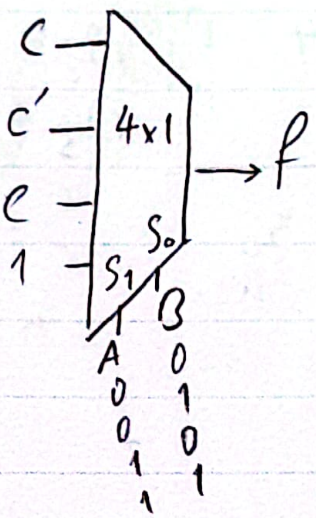


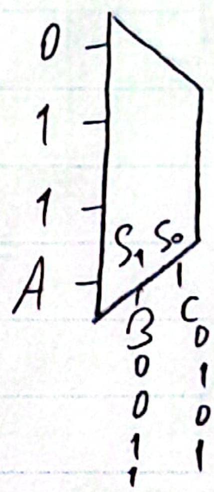
اما برای اینکار کافیست به تعداد یکی کمتر از متغیرها برای خط انتخاب جدا کرده و مالتی

پلکسرهای مربوط را می‌کشیم .

نکته : می‌توان همه متغیرها را در خط انتخاب گذاشت و به جای ورودی خود f را بدهیم

A	B	C	f	
0	0	0	0	C
0	0	1	1	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	C
1	0	1	1	
1	1	0	1	1
1	1	1	1	

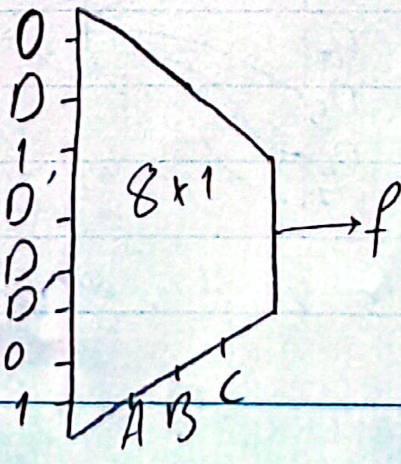


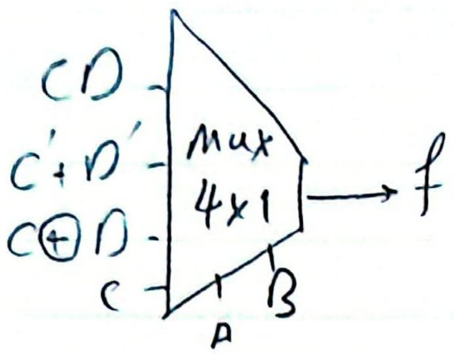


به طور کلی : برای تابع n متغیره استفاده از mux $2^{n-1}\times 1$

A	B	C	D	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
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	0

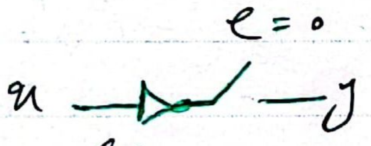
A	B	C	D	f
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1





بافر سه حالته

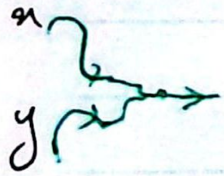
اضافه کردن تأخیر انتشار



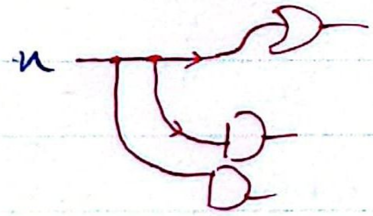
High Impedance

3- State

در حالت کلی می‌توانیم

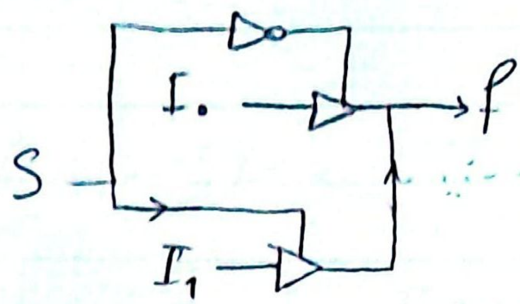


ولی نمی‌توانیم

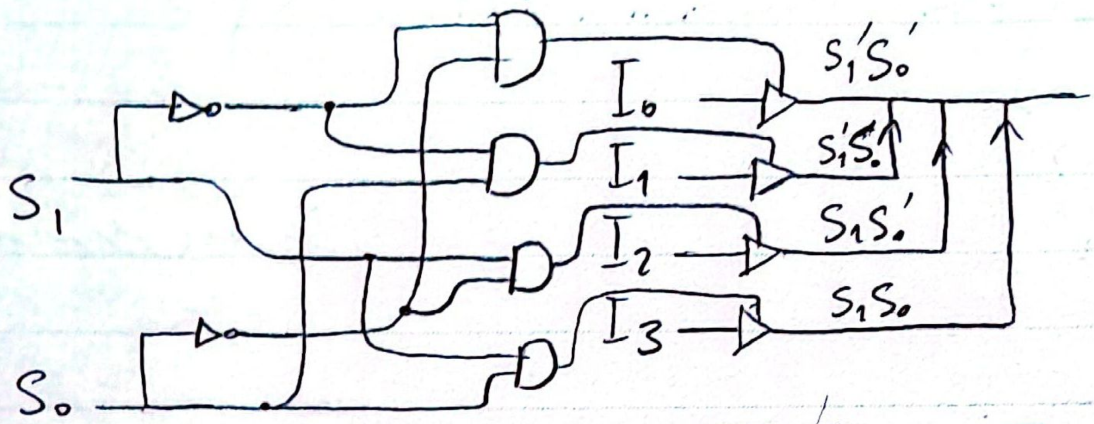


تذکر: تنها جایی که می‌توانیم این کار را انجام دهیم در بافر سه حالته است که $e = 0$ باشد

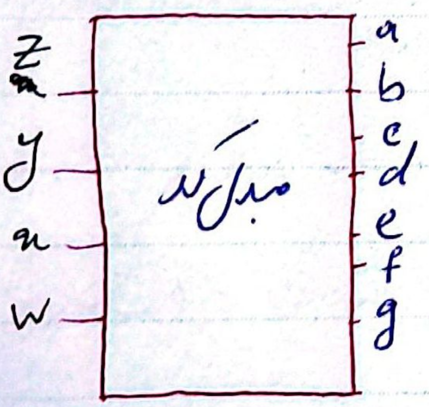
این دقیقاً همان multiplexer است

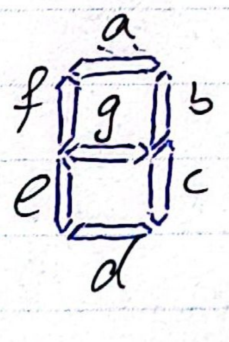


s	f
0	I_0
1	I_1



لامپ‌های 7 قسمتی 7Segment Lamp





رقم	کد باینری (w x y z)	7segment (g f e d c b a)	Hex
0	0000	0111111	3F
1	0001	0000110	06
2	0010	1011011	5B
3	0011	1001111	4F
4			
5			
6			
7			

Read Only Memory

f_1	f_2	f_3	f_4
0	1	1	0
1	1	0	1
1	1	1	1
⋮	⋮	⋮	⋮
1	1	1	1

4

5 + [illegible]

5×32

خط آدرس

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

ROM $2^n \times m$

خط آدرس

ورودی

$xy \rightarrow$ ROM $\rightarrow f_1 f_2$

0	0	0
1	0	1
2	0	1
3	1	0

ROM 4×2

:=

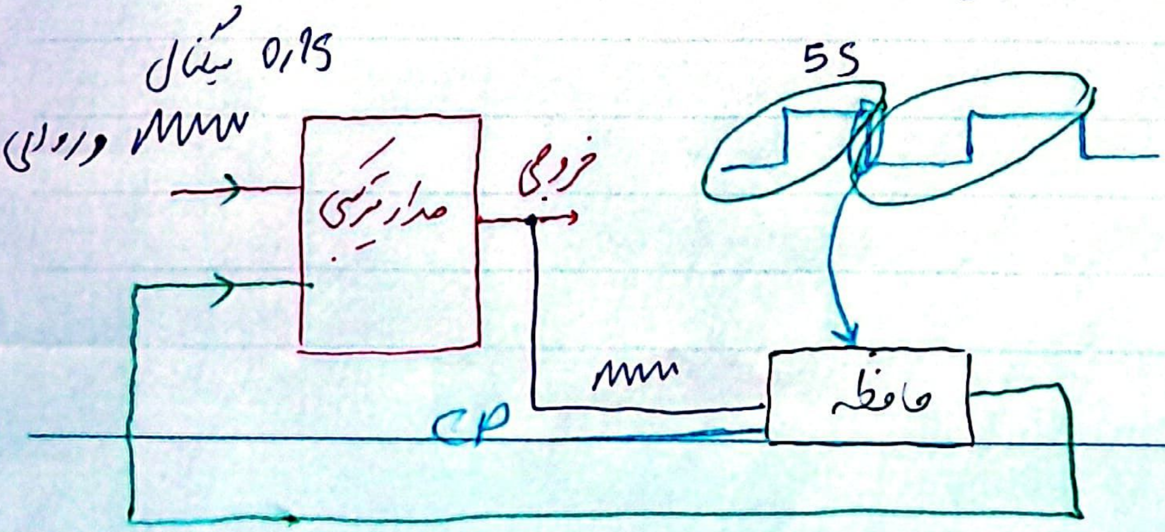
	f_1	f_2
0	0	0
1	0	1
2	0	1
3	1	0

فصل 6 : مدارهای ترتیبی

یک مدار ترتیبی مداری است که برخلاف مدارهای ترکیبی تنها وابسته به ورودی نیست بلکه باید وضعیت قبلی خود را نیز بداند که حافظه وارد کار می‌شود.

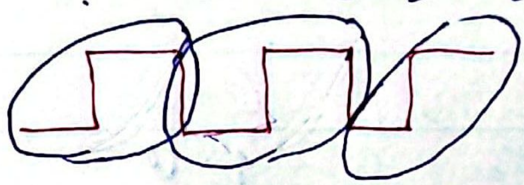
در واقع مدارهای ترتیبی مدارهایی هستند که مدارهای ترکیبی را با ورودی حافظه ترکیب کنیم!

در مدار ترکیبی به دلیل آنکه تنها وابسته به ورودی است لذا با تغییر سیگنال های ورودی به همان شدت سیگنال های خروجی نیز تغییر می کند که باعث تغییر در حافظه می‌شود (به صورت مدام) برای جلوگیری از آن و ثابت ماندن و از بین نرفتن اطلاعات حافظه از زمانی به نام پالس ساعت cp استفاده می‌شود. پالس ساعت کمک می کند که با تغییر سیگنال های ورودی مقادیر حافظه ثابت می ماند.



پالس ساعت

یک سیگنال ورودی که در بازه‌های زمانی مشخص ۰ سپس ۱ می‌شود. مثلاً در سیگنال



3 پالس ساعت وجود دارد

کاربردهای پالس ساعت

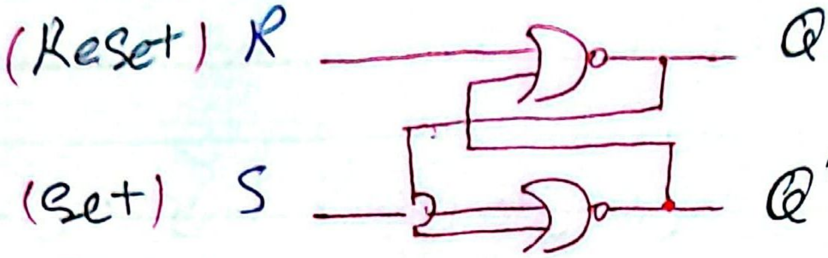
1. هر چه قدر ورودی تغییر کرد حافظه تغییر نکند

2. در مدار ترتیبی خروجی به حافظه برده می‌شود و از آن برای ورودی استفاده می‌شود. بعد از این مرحله خروجی بعدی بدست می‌آید و همین عمل دوباره تکرار می‌شود (به زبان ساده اگر خروجی 1، خروجی 2، ... که بدست می‌آیند به هم ربط دارند و خروجی‌های قبلی ثابت است) و این یک زمان مشخص دارد که از پالس ساعت بهره می‌برد.

3. مدارهای ترتیبی سنکرون هستند.

مدار پایه‌ای فلیپ فلاپ (لچ)

مدار: NOR



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

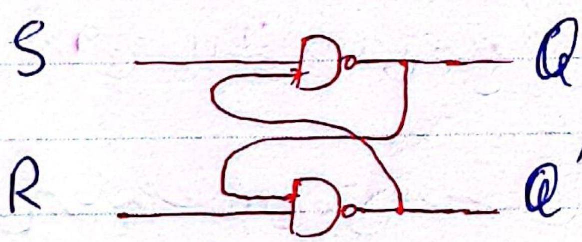
Set ← (1 0)

Reset ← (0 1)

مقدار قبلی را می‌دهد

نکته: از این حالت اغلب استفاده نمی‌شود چرا که می‌خواهیم یا Set کنیم یا Reset کنیم فلذا همزمان انجام نمی‌گیرند.

NAND:

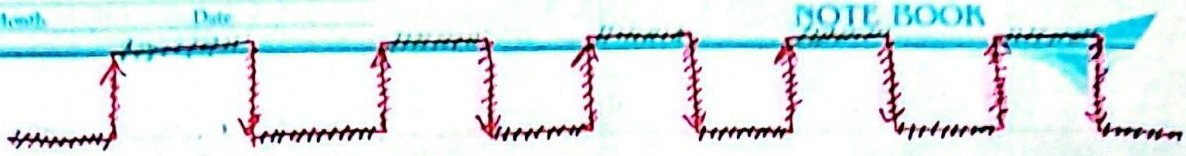


S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

فلیپ فلاپ

۱ بیت حافظه را فلیپ فلاپ گوییم.

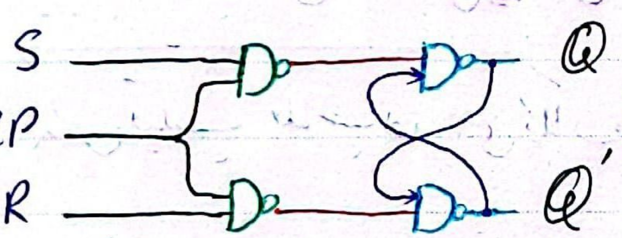
هر فلیپ فلاپ ترکیبی از گیت‌ها است.

CP


* واحدهایی ،،،،،، : لبه‌های منفی   در این حالت ورودی های R,S روی حافظه تأثیر ندارد

،،،،،، : لبه‌های مثبت

⇑ : لحظه 0 → 1 (لحظه منفی)   ⇓ : لحظه 1 → 0 (لحظه مثبت)



جدول تغییر فلیپ فلاپ

CP	S	R	Q_{t+1}
*0	X	X	Q_t
1	0	0	Q_t
*1	1	0	1
*1	0	1	0
1	1	1	?

نامشخص

① اگر از بالا به پایین بیاید 1
② اگر از پایین به بالا بیاید 0

اگر همگی (خروجی دو ورودی Q, Q') یک بشوند

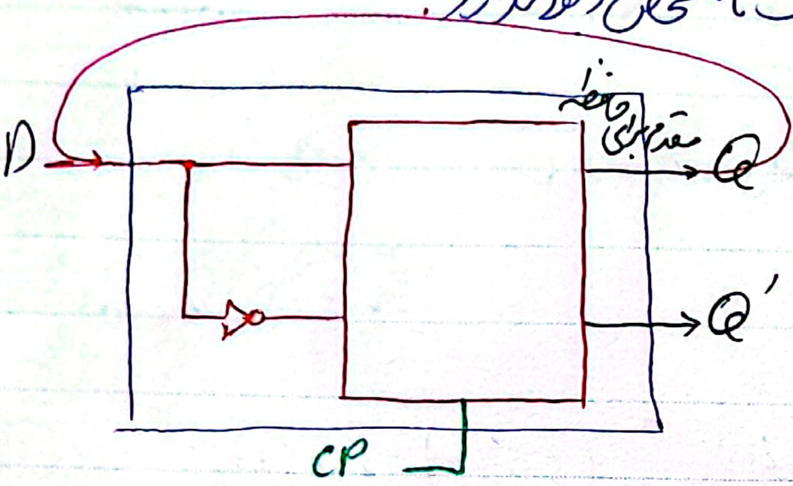
در حالت نامشخص در لحظه 0 → 1 شدن بستگی به سرعت سیستم دارد که

ببینند کدام یک از دو حالت بالا سریع‌تر به خروجی رسیده است.

نکته: حالت های * در جدول تغییر فلیپ فلاپ مربوط به فلیپ فلاپ D نیز می‌شود

فلیپ فلاپ D

در این نوع فیلیپ فلاپ حالت نامشخص وجود ندارد.

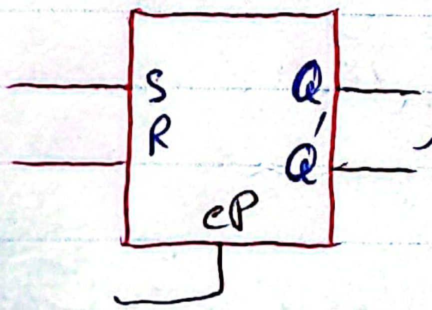


CP	D	Q_{t+1}
0	X	Q_t
1	0	0
1	0	1

فیلیپ فلاپ های حساس به لبه های منفی

در این فیلیپ فلاپ ها وقتی کلاک پالس در لبه های مثبت قرار دارد (1) ثابت می ماند ولی وقتی به لبه های منفی رسید ورودی های S, R روی حافظه تاثیر گذارند

وقتی CP مقدارش 1 است ← خروجی ثابت

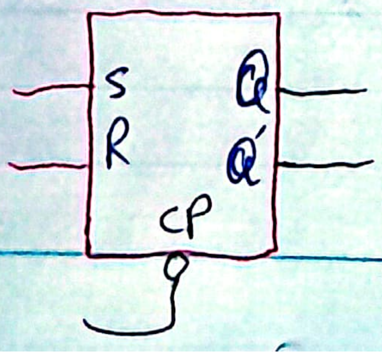


وقتی CP مقدارش 0 است ← خروجی تغییر می کند

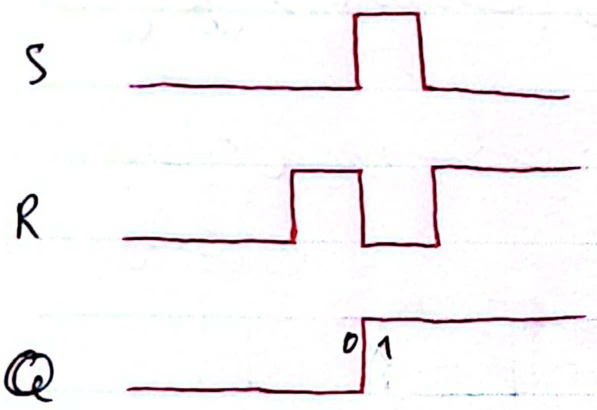
فیلیپ فلاپ های حساس به لبه های مثبت

در این حالت برعکس قبل وقتی کلاک پالس در لبه های منفی باشد ثابت ماندار

در لبه های مثبت تغییر می کند



در فیلیپ فلاپ ها مشکلی به وجود دارد آن است که ممکن است با تغییر S, R حتی با توجه به اینکه حساس به سطح های مثبت یا سطح های منفی است ممکن است که روی خروجی Q چندین تاثیر بگذارد



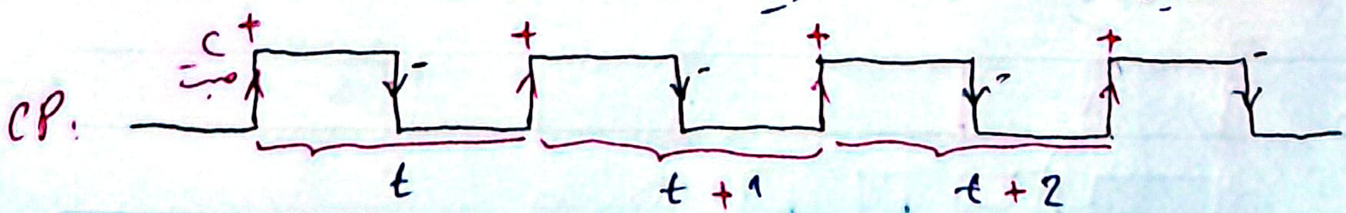
مثلاً اینها حساس به سطح های منفی اند

Q در حال تغییر است

در حالی که برای مدارهای ترتیبی قرار است که در یک لحظه تنها تغییر ایجاد شود و تا لحظه بعد این امواج ثابت بماند برای رفع این مشکل کلاک پالس را به جای آنکه حساس به سطح های مثبت یا منفی کنیم، حساس به لبه مثبت و منفی می کنیم.

لذا منظور از لبه مثبت لحظه از 0 به 1 شدن cp است و منظور از لبه منفی لحظه از 1 به 0 شدن cp است.

حال می توان یک لحظه کلاک پالس را بین دو لبه مثبت یا منفی در نظر گرفت یعنی:



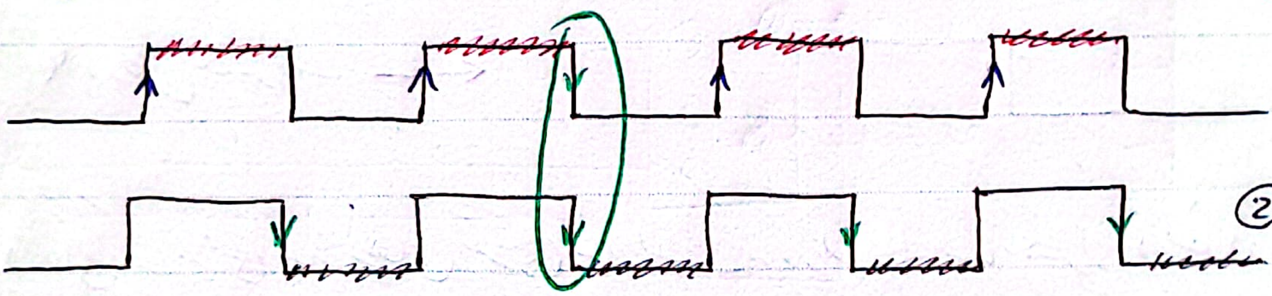
در این حالت خروجی آن مدام تغییر می‌کند؛ لذا تغییر تنها در لبه‌های مثبت هم اتفاق می‌افتد و تا لحظه مثبت بعدی آن ثابت می‌ماند (و بالعکس)

سؤال: حال چگونه باید کلاک پالس را حساس به لبه مثبت یا منفی کرد؟

دو کلاک پالس زیر را در نظر بگیرید:

③ حساس به لبه مثبت یا لبه بالارونده

حساس به لبه‌های مثبت ①

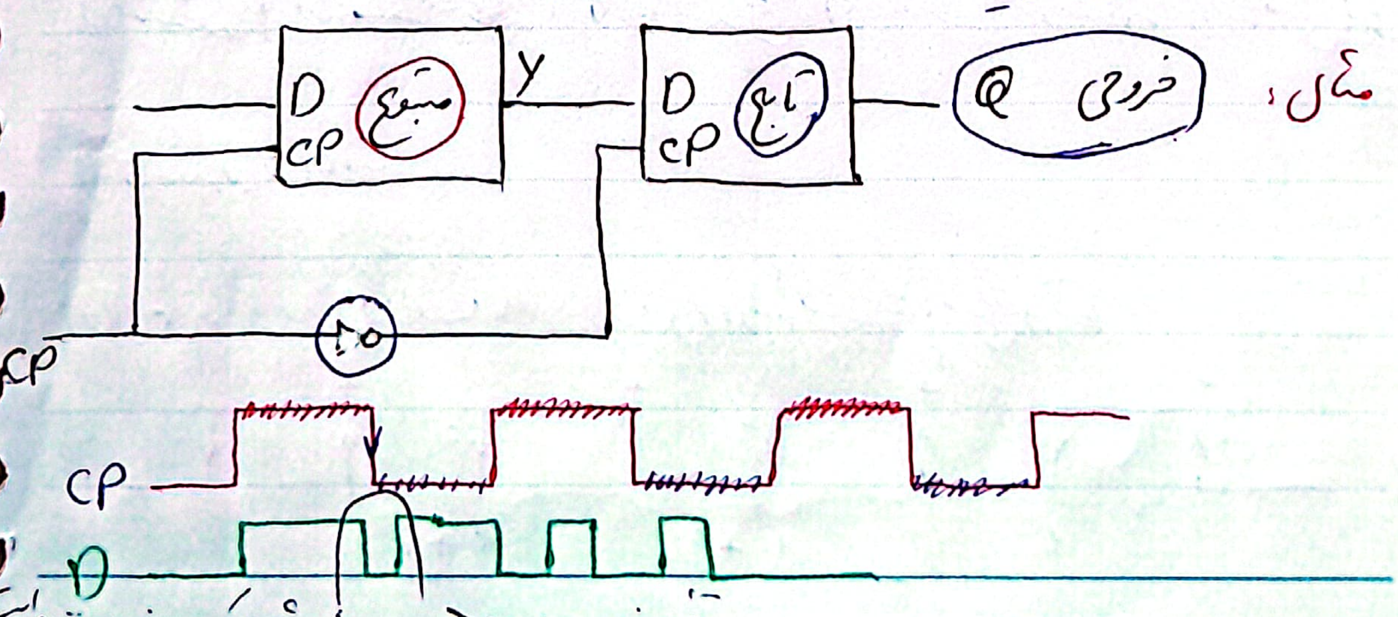


حساس به لبه‌های منفی ②

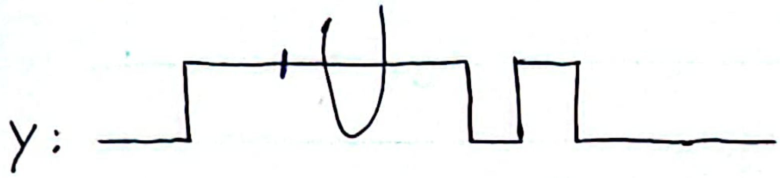
④ حساس به لبه منفی یا لبه پایین رونده

کافیست برای پیدا کردن حالات مطلوب اشتراک دو لبه منفی یا مثبت را بدست آوریم که به آن تابع متبوع می‌گوییم.

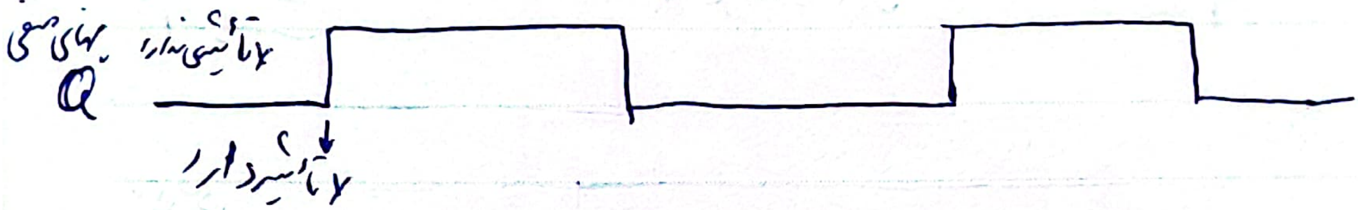
مثال:



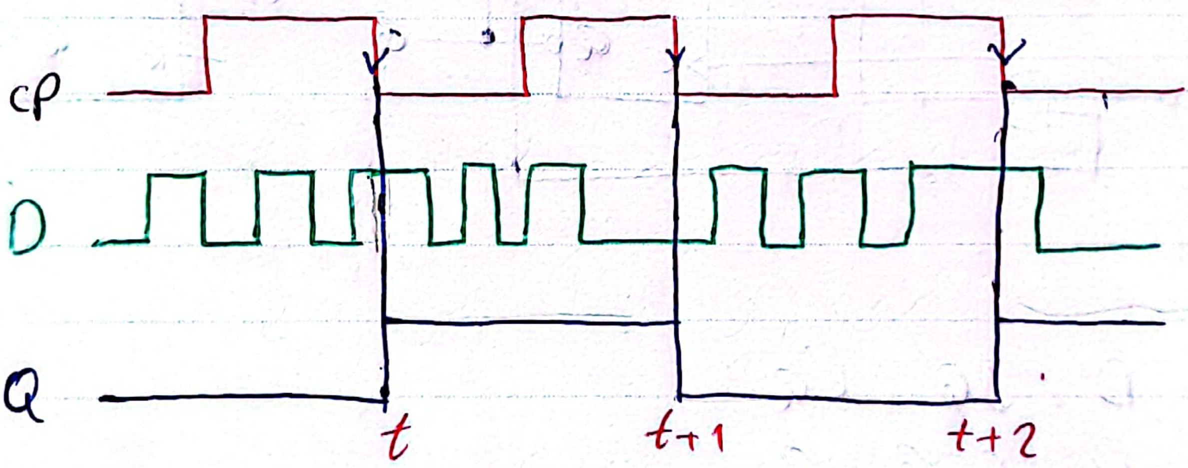
تغییر فلیپ فلاپ زمانی Y نداشته برقرار لبه‌های منفی هم است



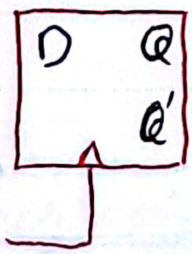
پس تغییر D روی خروجی y با توجه به اینکه حساس به پهنای مثبت است تأثیرگذار است.



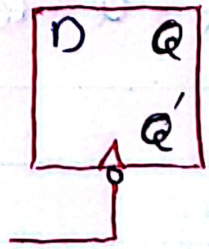
مثال ۲، اگر حساس به لبه باشد



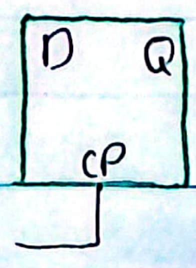
مدارهای فلیپ فلاپ های حساس:



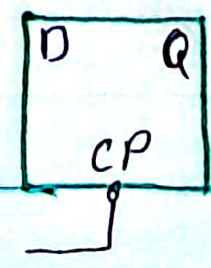
حساس به لبه مثبت



حساس به لبه منفی



حساس به پهنای مثبت



حساس به پهنای منفی

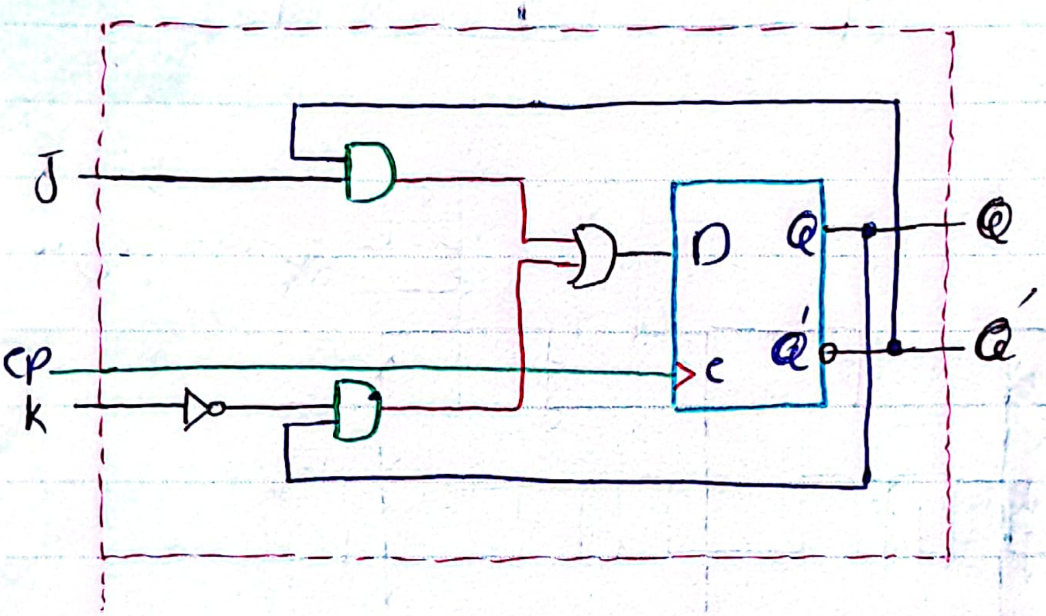
فلیپ فلاپ JK

	J	k	Q
*	0	0	ثابت
	0	1	0
	1	0	1
*	1	1	مکمل

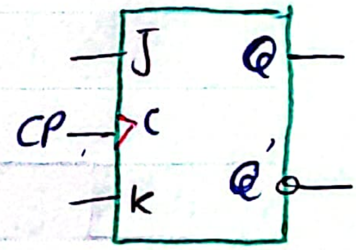
k (kill : 0)

J (jump : 1)

حالات * برای فلیپ فلاپ D نیز است



نمای کلی



$$D = JQ' + K'Q$$

J k

$00 \rightarrow D = 0 + Q = Q$

$01 \rightarrow D = 0 + 0 = 0$

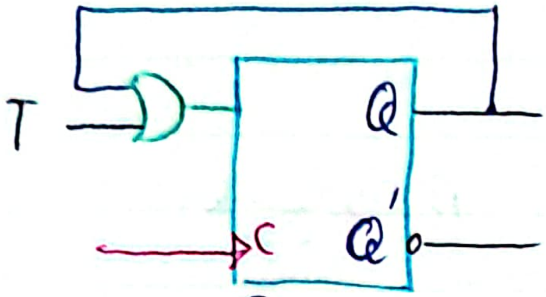
$10 \rightarrow D = Q' + Q = 1$

$11 \rightarrow D = Q' + 0 = Q'$

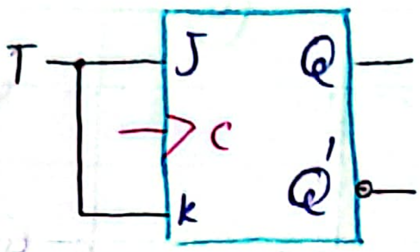
فلیپ فلاپ T

T	Q
0	ثابت
1	مکمل

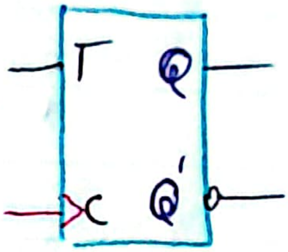
نماد کلی



با فلیپ فلاپ D



با فلیپ فلاپ JK



$D = T \oplus Q$

$x \oplus 0 = x$
$x \oplus 1 = x'$
یادآوری

if $D = Q$ ثابت

T

$0 \rightarrow D = 0 \oplus Q = Q$

$1 \rightarrow D = 1 \oplus Q = Q'$

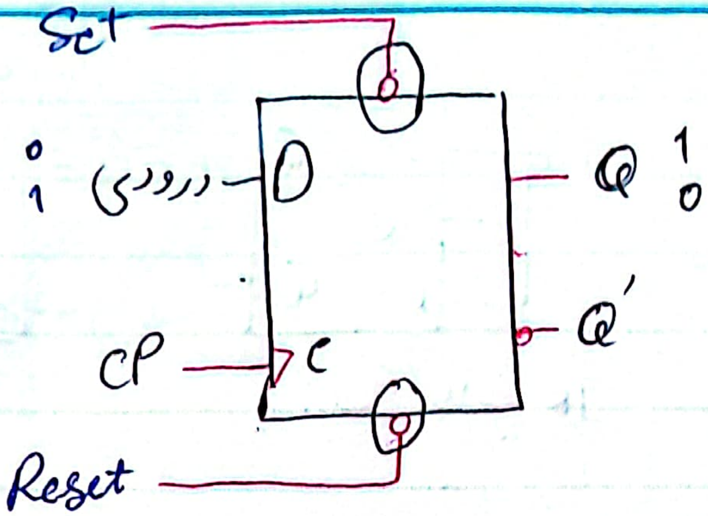
$D = Q'$ مکمل

ورودی های سنکرون و مستقیم

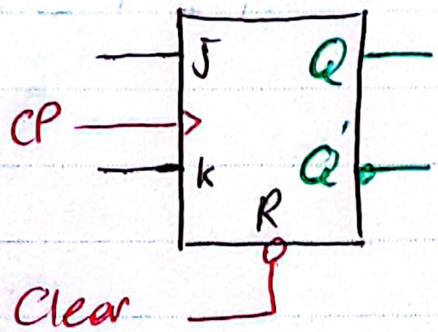
ورودی های سنکرون یا همگام باید هر دو ورودی با هم تغییر کنند مانند JK در واقع به یک کلاک پالس وصل هستند

ورودی های مستقیم یا آسنکرون یا غیرهمگام مستقیم روی خروجی تاثیر می گذارند مانند

Reset , Set , clear , Enable در واقع به یک کلاک پالس وصل نیستند



فلیپ فلاپ D

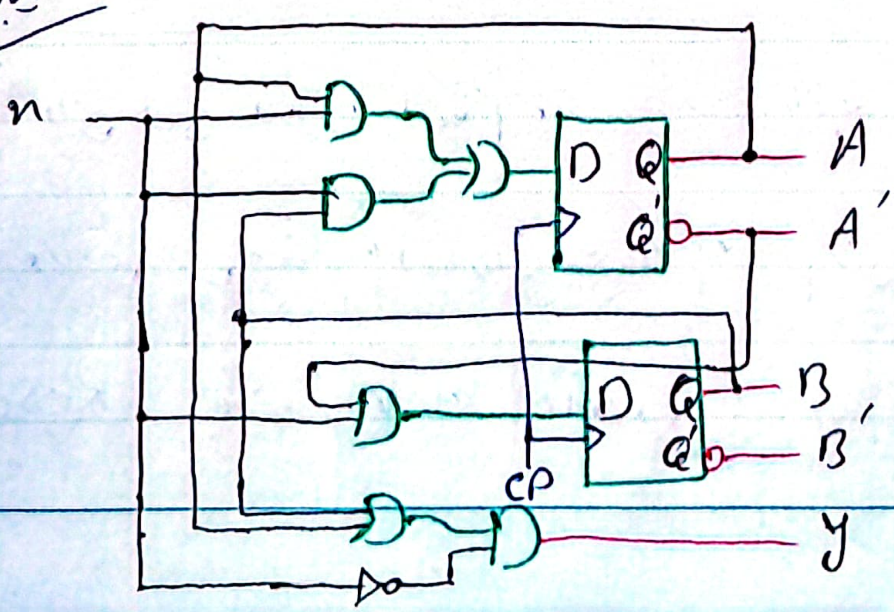


فلیپ فلاپ JK

Clear	CP	J	k	Q	Q'
0	X	X	X	0	1
1	↓	0	0	بدون تغییر	
1	↓	0	1	0	1
1	↓	1	0	1	0
1	↓	1	1	مکمل می‌شود	

مدار ترتیبی، معادله حالت، جدول حالت، دیاگرام حالت

مدار ترتیبی



معادله حالت

$$A(t+1) = A_t x + B_t x$$

$$B(t+1) = A'_t x$$

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

جداول حالت

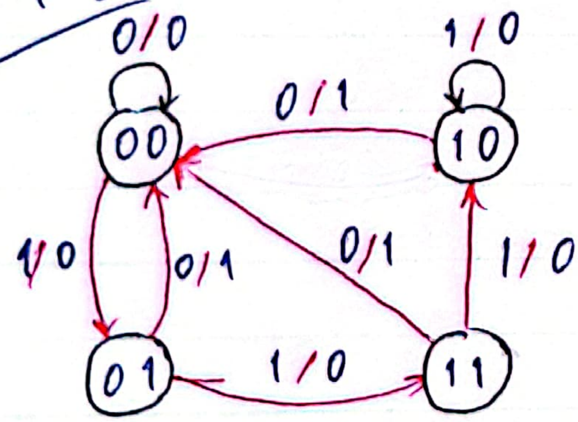
I

A_t	B_t	x	A_{t+1}	B_{t+1}	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

II

حالت فعلی	حالت بعدی		y	
	$x=0$	$x=1$	$x=0$	$x=1$
$A_t\ B_t$	$A_{t+1}B_{t+1}$	$A_{t+1}B_{t+1}$	y	y
0 0	0 0	0 1	0	0
0 1	0 0	1 1	1	0
1 0	0 0	1 0	1	0
1 1	0 0	1 0	1	0

دیاگرام حالت



جدول های مشخصه فلیپ فلاپ

J	K	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	Q'_t

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	?

غیر قابل پیش‌بینی

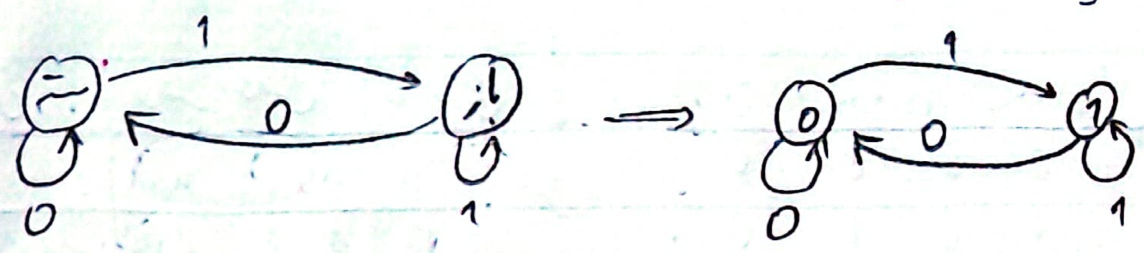
D	Q_{t+1}
0	0
1	1

T	Q_{t+1}
0	Q_t
1	Q'_t

NOTE BOOK

Year Month Date

مثال 2: دیاگرام باز و بسته شدن در

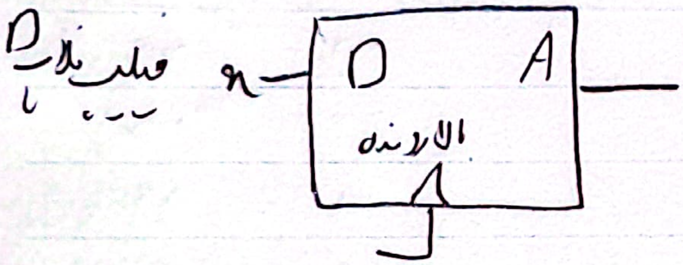


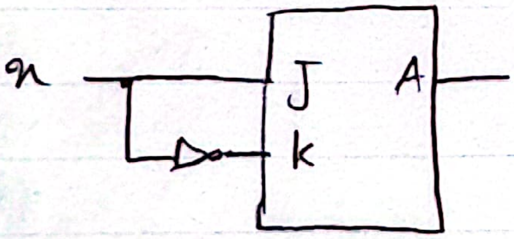
A_t	x	A_{t+1}	D_A	J_A	K_A
0	0	0	0	0	X
0	1	1	1	1	X
1	0	0	0	X	1
1	1	1	1	X	0

$D_A(A,x) = x$

$J_A(A,x) = x$

$K(A)(A,x) = x'$





جدول های تحریک فلیپ فلاپ ها

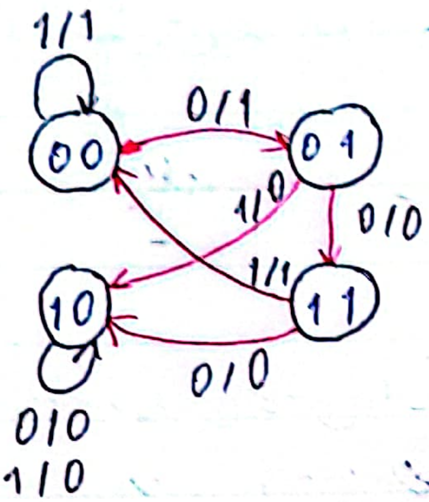
Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Q_t	Q_{t+1}	R	S
0	0	X	0
0	1	0	1
1	0	1	0
1	1	0	X

Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

مثال 3: با توجه به دیاگرام زیر مدار ترتیبی آن را رسم کنید



A	B	x	A	B	Y	J_A	k_A	R_B	S_B
0	0	0	0	1	1	0	X	0	1
0	0	1	0	0	1	0	X	X	0
0	1	0	1	1	0	1	X	0	X
0	1	1	1	0	0	1	X	1	0
1	0	0	1	0	0	X	0	X	0
1	0	1	1	0	0	X	0	X	0
1	1	0	1	0	0	X	0	1	0
1	1	1	0	0	1	X	1	1	0

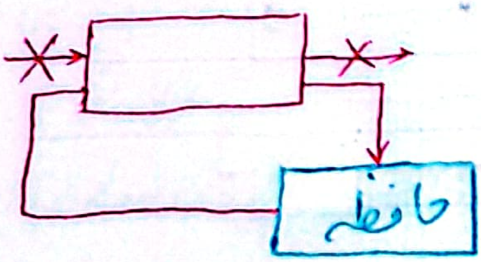
$Y = A'B' + ABx$

$J_A = B$ $\qquad$ $R_B = A + x$

$k_A = Bx$ $\qquad$ $S_B = A'x'$

طراحی شمارنده

مدارهای ترتیبی که ورودی و خروجی ندارند.

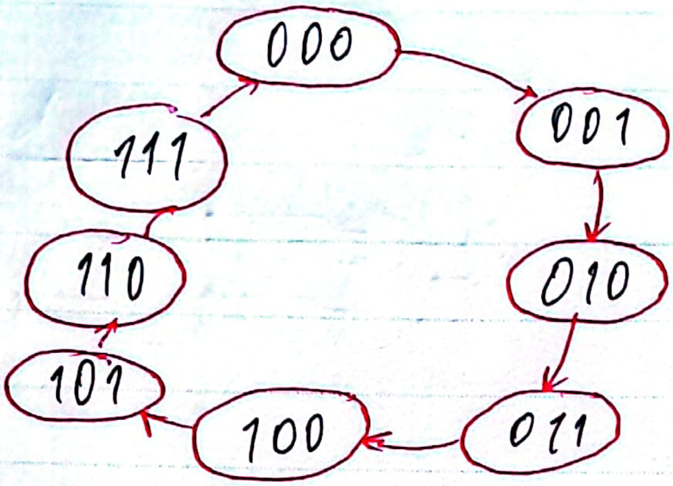


تنها به حافظه بستگی دارد

مثلاً در یک شمارنده 3 بیتی ورودی در هر پالس ساعت 0 به بعدی می‌شود 1 از

NOTE BOOK Year Month Date

۱ به بعدی می‌شود ۲ و ... نهایتاً به 7 که می‌رسد دوباره به صفر تبدیل می‌شود.



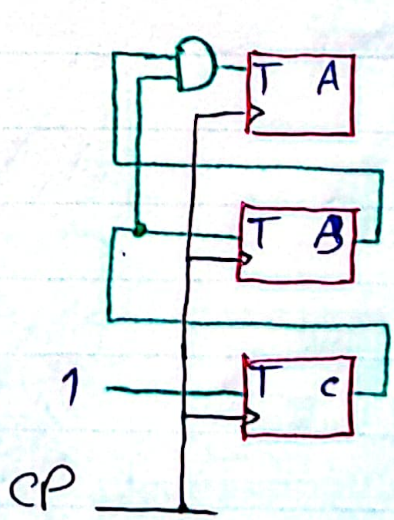
⟸ شمارنده دودویی ۳ بیتی (0 تا 2^n-1) می‌شمارد

ABC	ABC	T_A	T_B	T_C
000	001	0	0	1
001	010	0	1	1
010	011	0	0	1
011	100	1	1	1
100	101	0	0	1
101	110	0	1	1
110	111	0	0	1
111	000	1	1	1

$T_C = 1$

$T_B = C$

$T_A = BC$



لبه بالارونده

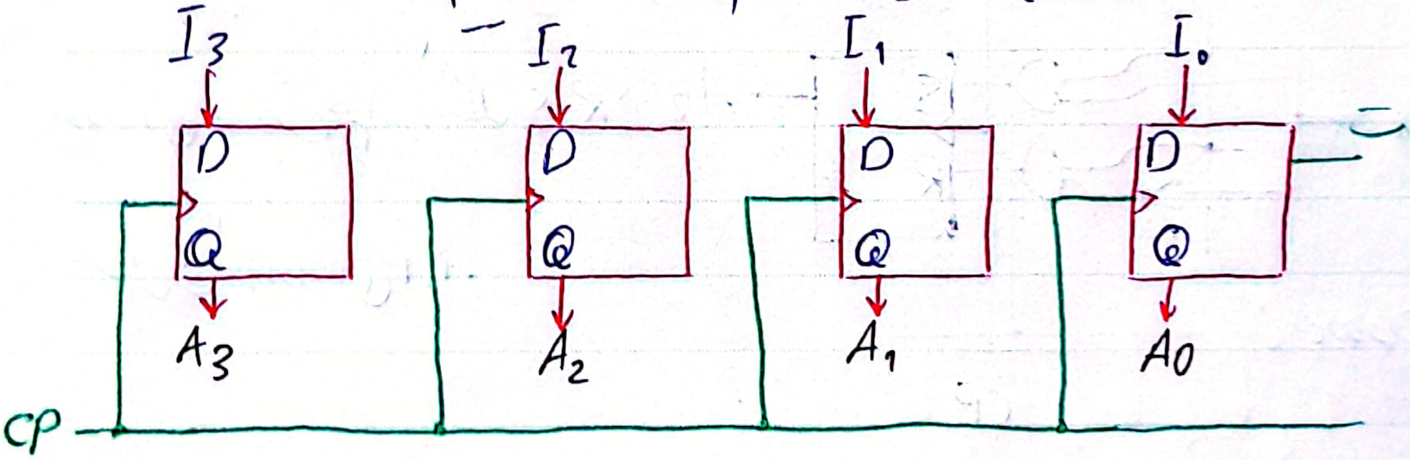
فصل 7: ثبات

ثبات شمارنده:

ثباتی که چندبیتی است و فقط می‌شمارد.

ثبات

مجموعه‌ای از فلیپ فلاپ‌ها را کنار هم را ثبات می‌گوییم.



ثبات 4 بیتی A

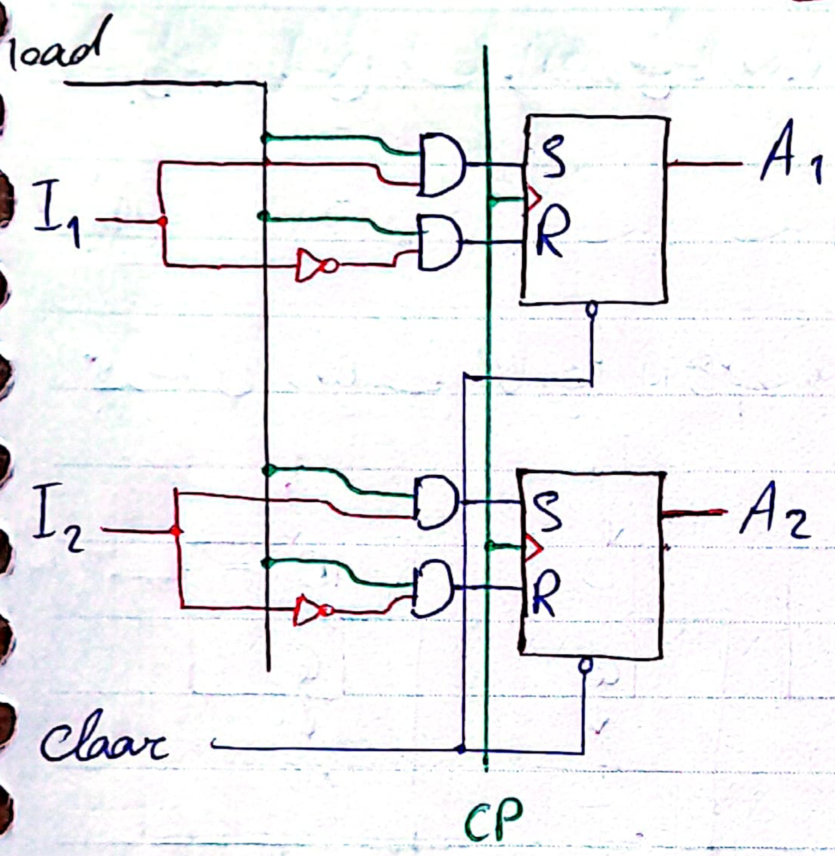
$A = A\ A\ A\ A$

به دلیل آنکه ما باید در هر کلاک پالس به آن مقدار دهیم (چون در هر CP در حال تغییر کردن است) مشکلی که به وجود می‌آید آن است که مدام با ورودی ما تغییر می‌کند پس برای رفع مشکل از ثبات با امکان بار شدن موازی استفاده می‌کنیم

اگر 0 باشد ورودی عوض نشود یا نشود مهم نیست (می‌خواهی بشود می‌خواهی نشود)

ولی اگر ۱ و ۰ شود تعویض شود

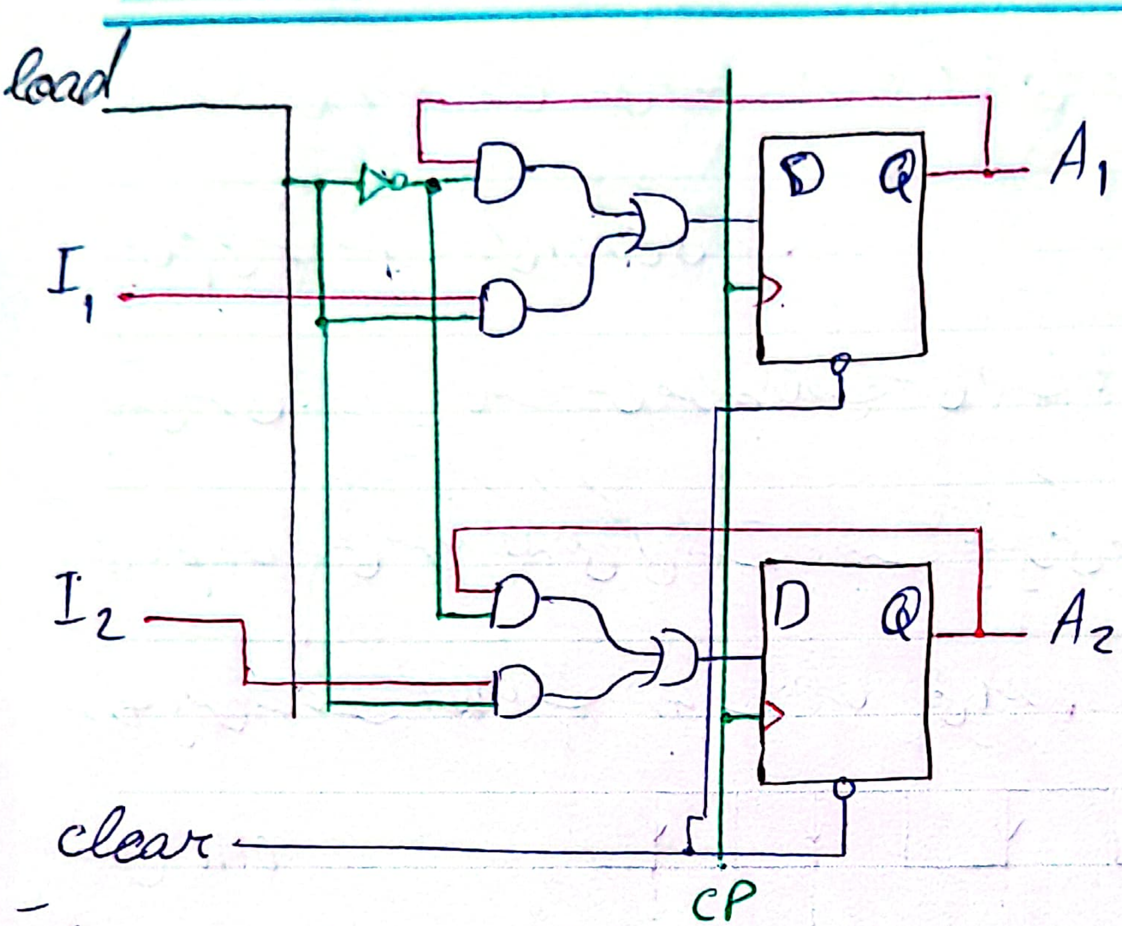
ثبات با قابلیت بارگذاری موازی



تذکر: در اینجا ورودی ۱ و ۱ نداریم زیرا دو ورودی نقیض هم اند و مشکل R,S نیز حل می شود در ورودی ۱،۱.

در اینجا دیتای ثابت نیازی به ورودی ثابت داشت اما با صفر کردن load دیگر ثابت می ماند.

Claar	CP	load	I_2 I_1	A_2 A_1
0	X	X	X X	0 0
1	X	0	X X	A_2 A_1
1	↑	1	0 0	0 0
1	↑	1	0 1	0 1
1	↑	1	1 0	1 0
1	↑	1	1 1	1 1



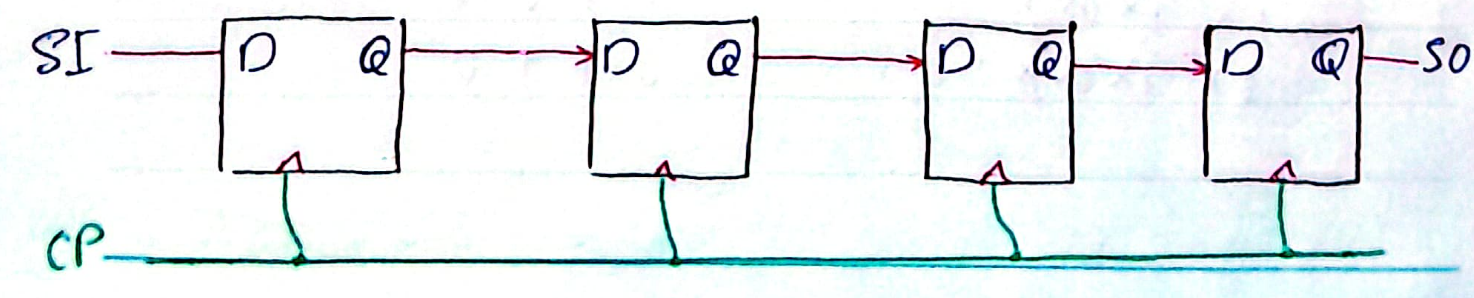
ورودی موازی : کل اطلاعات با هم در ثبات قرار می‌گیرد آن هم در یک لحظه یا پالس ساعت

شیفت دهنده

شیفت دهنده راست، دیتا را به سمت راست و شیفت دهنده چپ دیتا را به سمت چپ. و اگر شیفت دهنده از آخر به اولش وصل شود شیفت دهنده چرخشی می‌گوییم.

نکته: شیفت دهنده راست چپ ترینش ورودی دارد. (و بالعکس)

Serial Input



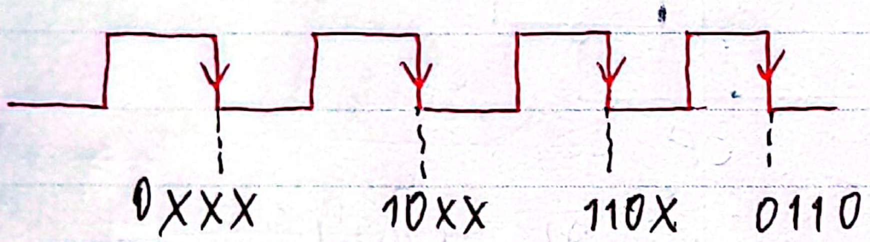
در لبه پایین رونده هر چی را اول هست در دری قرار می گیرد ولی چون در یک لحظه

انجام می شود مثلاً 0 اول در سوی می رد.

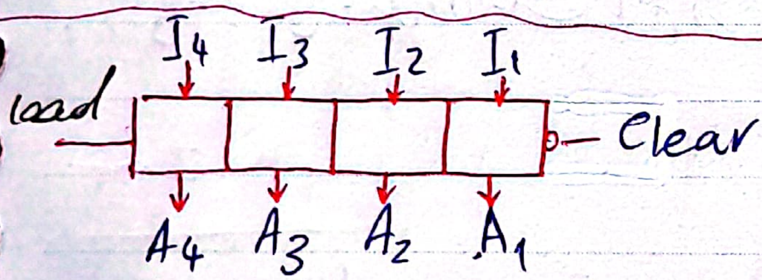
شیفت سری: در این حالت شیفت مدار به حالت سری است! زیرا کل اطلاعات

یکجا با هم مقدار نمی گیرند بلکه یکی یکی بیت به بیت تغییر می کنند مثلاً در مثال بالا

در 4 پالس ساعت اطلاعات در ثبات انتقال می یابد.

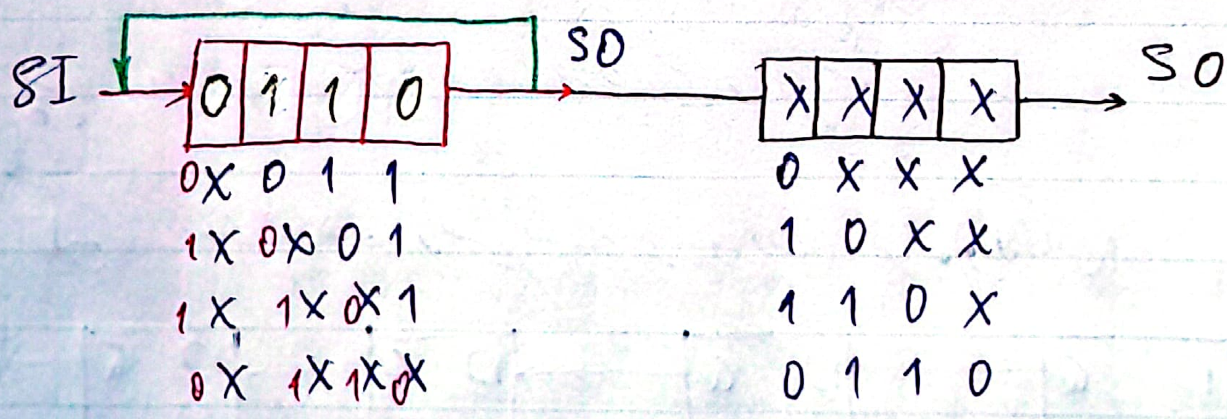


انتقال 0110

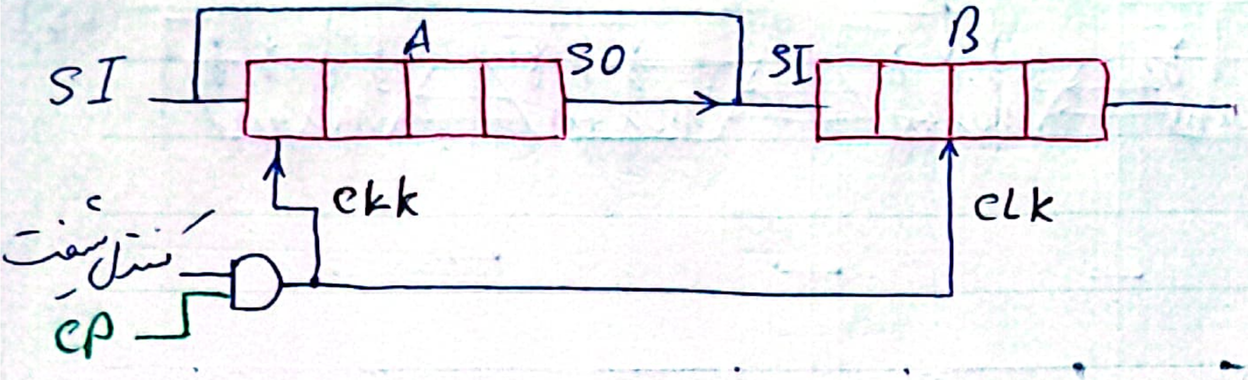


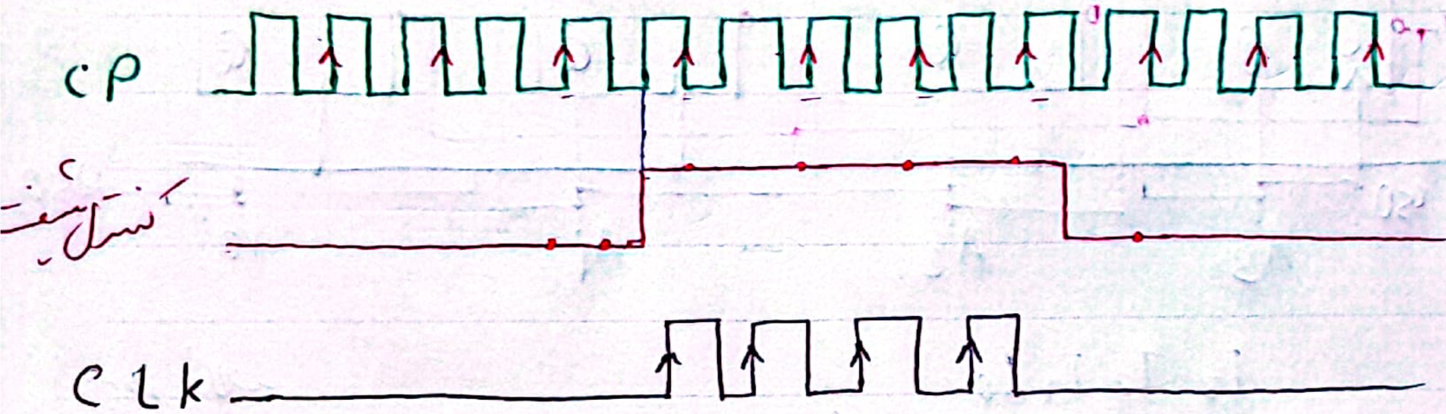
شیفت به صورت موازی

ادامه

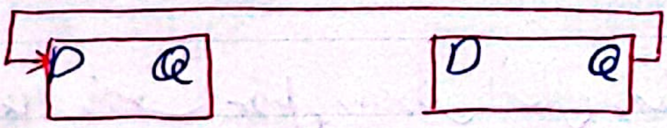


شیفت به راست

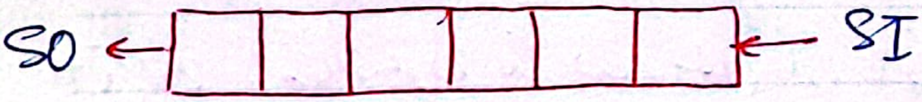




شیفت دهنده به چپ

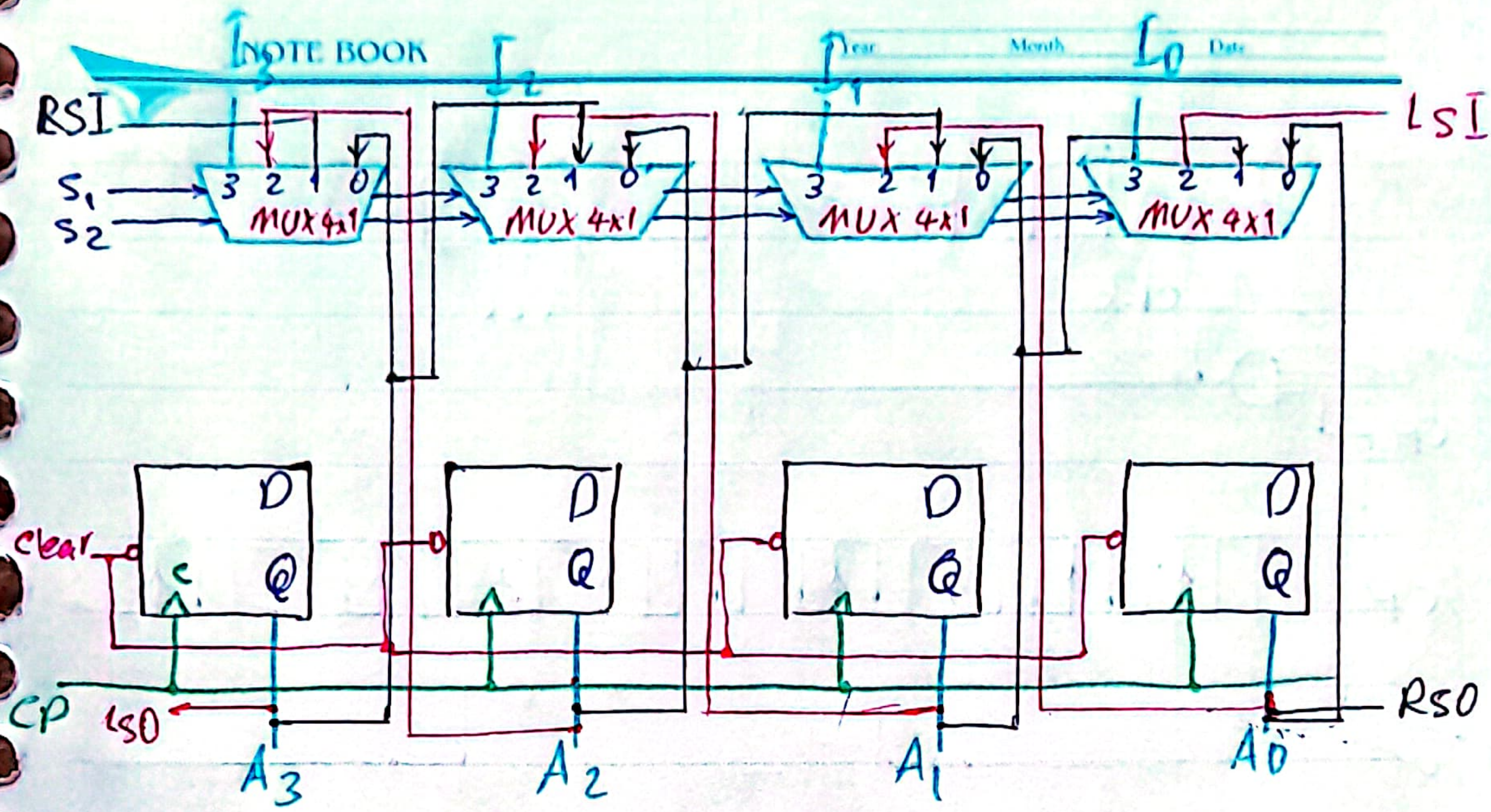


بقیه مانند بالا و مطالب قبل

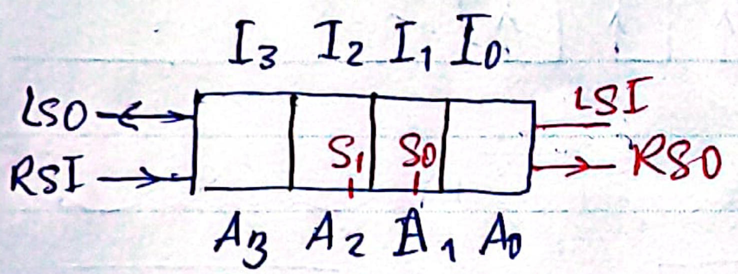


شیفت دهنده با امکان لود موازی (هم شیفت به چپ هم راست)

S_1	S_0	
0	0	ثابت
0	1	شیفت به راست
1	0	شیفت به چپ
1	1	load



به طور خلاصه



* با کمال تاسف به دلیل زیاد بودن اشکال و مدارهای ترتیبی لذا خواهشمندم که از اسلاید

11 تا 27 را مطالعه نموده لذا فقط مثل را اینجا توضیح خواهم داد: برای 0 تا 15

نکته: شمارنده باینری از 0 تا 15 می شمارد و شمارنده BCD از 0 تا 9 می شمارد