«طراحی و پیاده سازی مدارات منطقی»

مدرس:
سمیرا تیموری

سیستم دهدهی اعداد

$491 = 4\times10^{2} + 9\times10^{1} + 1\times10^{0}$

$0.32 = 3\times10^{-1} + 2\times10^{-2}$

$491.32 = 4\times10^{2} + 9\times10^{1} + 1\times10^{0} \,.\, 3\times10^{-1} + 2\times10^{-2}$

تبدیل مبناهای اعداد

- توجه: با تغییر مبنای عدد، ماهیت آن عوض نمی شود بلکه فقط شکل نشان دادن آن تغییر می کند.

$$a = (a_n \dots a_2 a_1 a_0 . a_{-1} a_{-2} \dots a_{-m})_r$$

- مبنای ۲ (binary)
- مبنای ۱۰ (decimal)
- مبنای ۸ (octal)
- مبنای ۱۶ (hexadecimal) : شامل ارقام ۰ تا ۹ و a، b، c، d، e، f

تبدیل عدد از مبنای r به مبنای ۱۰

$$a = (a_n \dots a_2 a_1 a_0 . a_{-1} a_{-2} \dots a_{-m})_r$$

$$\sum_{i=-m}^{n} a_i (r)^i$$

$$(110.01)_2 = (1\times2^2 + 1\times2^1 + 0\times2^0 \, . \, 0\times2^{-1} + 1\times2^{-2})_{10}$$

تبدیل عدد از مبنای ۱۰ به مبنای r (روش تقسیمات متوالی)

- قسمت صحیح عدد را متوالیا به r تقسیم و قسمت اعشاری عدد را متوالیا در r ضرب می کنیم.
- مزیت روش تقسیمات متوالی بر مبنا، سادگیآن است.

$(301.2)_{10} = (?)_8$

$(?) = 455.1463....$

301	8	
296	37	8
5	32	4
	5	0
		4

8
0

$0.2 \times 8 = 1.6$

$0.6 \times 8 = 4.8$

$0.8 \times 8 = 6.4$

$0.4 \times 8 = 3.2$

$0.2 \times 8 =$ تکراری

تبدیل عدد از مبنای ۱۰ به مبنای r (روش افزودن وزن ها)

- بیشتر برای تبدیل اعداد دهدهی به اعداد دودویی کاربرد دارد.
- توان های صعودی ۲ را تا مقدار بزرگتر از عدد می نویسیم. با قرار دادن ۱ در زیر بزرگترین وزنی که مساوی یا کوچکتر از عدد دهدهی است شروع می کنیم. سپس آن وزن را از عدد کم می کنیم، این روال به همین ترتیب برای دیگر وزن ها تکرار می شود.

$(43)_{10} = (?)_2$

$43 - 32 = 11$

$11 - 8 = 3$

$3 - 2 = 1$

64	32	16	8	4	2	1
	1	0	1	0	1	1

تبدیل عدد از مبنای r^n به r و برعکس

- **تبدیل از مبنای r^n به r**

به ازای هر رقم، n رقم در مبنای r قرار می دهیم.

$$(257)_8 = (\ 010\ 101\ 111\ )_2$$

- **تبدیل از مبنای r به r^n**

قسمت صحیح را از سمت راست و قسمت اعشاری را از سمت چپ به صورت دسته های n رقمی جدا می کنیم، و معادل هر دسته را در مبنای می نویسیم.

$$(10101111)_2 = (2\ 5\ 7)_8 \qquad (0.101111)_2 = (\ 0.5\ 7)_8$$

مکمل ها (Complements)

- در مبنای r دو نوع مکمل مطرح می شود: مکمل r-1 (کاهش یافته) و مکمل r (مبنا)

مکمل کاهش یافته (r-1) در مبنای r

برای یافتن مکمل کاهش یافته عدد a (در مبنای r)، همه ارقام عدد a را از r-1 کم می کنیم.

$a = (256.73)_{10}$ → مکمل r-1 → [743.26]

مکمل کاهش یافته (مکمل یک) در مبنای ۲

- برای یافتن مکمل ۱ عدد a در مبنای ۲، همه بیت ها را عوض می کنیم.

$a = (10101)_2$ ← مکمل ۱ ← [01010]

مکمل مبنا (r)

- راه اول: مکمل r از جمع ۱ با مکمل r-1 حاصل می شود.
- راه دوم: صفرهای سمت راست عدد در صورت وجود تغییر نمی کند و اولین رقم غیر صفر را از r و سایر ارقام از r-1 کم می شوند.

$a = (450.27)_{10}$ ← مکمل r ← [549.73]

مکمل مبنا (مکمل دو) در مبنای ۲

- برای یافتن مکمل ۲ در مبنای ۲، صفرهای سمت راست و اولین یک را عوض نمی کنیم، سایر بیت ها را عوض می کنیم.

$a = (10100)_2$ ← مکمل ۲ ← [01100]

نمایش اعداد دودویی علامت دار (منفی)

۱) سیستم علامت و مقدار: بیت سمت چپ هر عدد علامت است، اگر صفر باشد، عدد مثبت و اگر یک باشد، عدد منفی است.
۲) سیستم مکمل۱: در روش مکمل۱ مقدار منفی عدد، مکمل۱ عدد است.
۳) سیستم مکمل۲: در روش مکمل۲ مقدار منفی عدد، مکمل۲ عدد است.

- نکته: همه اعداد منفی دارای ۱ در سمت چپ ترین بیت اند.
- امروزه عملا فقط از روش متمم دو استفاده می شود.

نمایش اعداد دودویی علامت دار

علامت و مقدار	مکمل ۱	مکمل ۲
0011 = +3	0011 = +3	0011 = +3
0010 = +2	0010 = +2	0010 = +2
0001 = +1	0001 = +1	0001 = +1
0000 = +0	0000 = +0	0000 = +0
1000 = -0	1111 = -0	-
1001 = -1	1110 = -1	1111 = -1
1010 = -2	1101 = -2	1110 = -2
1011 = -3	1100 = -3	1101 = -3

نمایش اعداد دودویی

	ماکزیمم عدد با n بیت	مینیمم عدد با n بیت
سیستم بی علامت	$(111\ldots1)_2 = 2^n - 1$	0
سیستم علامت و مقدار	$(011\ldots1)_2 = 2^{n-1} - 1$	$(111\ldots1)_2 = -(2^{n-1} - 1)$
سیستم مکمل۱	$(011\ldots1)_2 = 2^{n-1} - 1$	$(100\ldots0)_2 = -(2^{n-1} - 1)$
سیستم مکمل۲	$(011\ldots1)_2 = 2^{n-1} - 1$	$(100\ldots0)_2 = -2^{n-1}$

تفریق دو عدد بی علامت با کمک مکمل مبنا

- عدد اول را با مکمل مبنا عدد دوم جمع می کنیم.
  - اگر $a \geq b$، عمل جمع یک رقم نقلی انتهایی تولید می کند که باید چشم پوشی شود.
  - اگر $a < b$، عمل جمع هیچ گونه رقم نقلی انتهایی تولید نمی کند، برای یافتن جواب مکمل مبنا حاصل جمع را بدست می آوریم و سپس یک علامت منفی در جلوی آن می گذاریم.

$(3250)_{10} - (72532)_{10} = ?$

$(03250)_{10}$

$+ \quad [27468]_{10}$

30718 ← مکمل 10 ← $- 69282$

جمع دو عدد دودویی علامت دار در سیستم مکمل۲

- مشابه اعداد بی علامت جمع می کنیم و از رقم نقلی تولید شده از آخرین مکان، صرف نظر می کنیم.

$$
\begin{array}{rl}
 & 1 \\
 & (1101)_2 = -3 \\
+ & (0110)_2 = +6 \\
\hline
1 & (0011)_2 = +3
\end{array}
$$

سرریز(OverFlow)

- اعداد در کامپیوتر با طول محدود و تعداد بیت های مشخص به کاربرده می شوند. اگر نتیجه محاسبات خارج از این محدوده شود و بیت های بیشتر در دسترس نباشد، این بیت های اضافی حذف خواهد شد و نتیجه بدست آمده صحیح نخواهد بود. در این حالت گوییم در انجام محاسبه سرریز اتفاق افتاده است.

تشخیص سرریز دو عدد بدون علامت

- درجمع اعداد بدون علامت، اگر پس از جمع دو عدد بدون علامت رقم نقلی نهایی یک شود سرریز اتفاق افتاده است.

$$\begin{array}{rl} & 1 \\ & (1101)_2 = 13 \\ + & (1100)_2 = 12 \\ \hline \text{رقم نقلی نهایی} \quad (1) & (1001)_2 = 9 \end{array}$$

تشخیص سرریز دو عدد علامت دار در سیستم مکمل۲

- راه اول: اگر جمع دو عدد منفی، مثبت شود یا جمع دو عدد مثبت، منفی شود، سرریز است. دقت کنید جمع عدد منفی با عدد مثبت سریز ندارد.

- راه دوم: اگر رقم نقلی وارد شده به بیت سمت چپ با نقلی خارج شده از آن یکسان نباشد، سریز است.

تشخیص سرریز دو عدد علامت دار در سیستم مکمل۲

$$\begin{array}{ccccc} c_n & c_{n-1} & c_{n-2} & c_1 & c_0 \\ & a_{n-1} & a_{n-2} & a_1 & a_0 \\ + & b_{n-1} & b_{n-2} & b_1 & b_0 \\ \hline & s_{n-1} & s_{n-2} & s_1 & s_0 \end{array}$$

1) if $a_{n-1} = b_{n-1} \neq s_{n-1}$ then $v=1$

$$v = a_{n-1}\, b_{n-1}\, \overline{s_{n-1}} + \overline{a_{n-1}}\, \overline{b_{n-1}}\, s_{n-1}$$

2) if $c_n \neq c_{n-1}$ then $v=1$

$$v = c_n \oplus c_{n-1}$$

تفریق دو عدد دودویی علامت دار در سیستم مکمل۲

- عدد اول را با مکمل ۲ عدد دوم جمع می کنیم، و از رقم نقلی خروجی از مکان بیت علامت چشم پوشی می کنیم.

$(11111010)_2 - (11110011)_2 = ?$

$(-6)_{10}$ $(-13)_{10}$

$11111010 = -6$

$+ \quad 00001101 = +13$

$100000111 = +7$

$(00000110)_2 = +6$

کد کردن ارقام دهدهی

- کدها باید به صورت دودویی باشند زیرا کامپیوترها فقط قادرند صفرها و یک ها را نگه داری کنند.
- کدها فقط نماد یا سمبل نمایش اطلاعات را عوض می کنند و نه مفهوم آن ها را.
- یک کد دودویی n بیت، 2^n ترکیب ممکن از یک ها و صفرها را دارا است.
- به هر یک از کدهای دودویی نسبت داده شده کلمه کد (Code Word) گویند.

کدهای دودویی برای نمایش ارقام دهدهی

$(18)_{10}$

$(0001\ 1000)_{BCD}$

$(10010)_{2}$

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

کدهای وزن دار و خود مکمل

- کد وزن دار: به هر مکان از بیت وزنی تخصیص داده شده است.
- کد خود مکمل: مکمل ۹ عدد دهدهی مستقیما از تغییر صفرها به یک و یک ها به صفر در کد حاصل می شود.
- کدهای وزن داری که جمع وزن های آن ۹ نیست نمی تواند خود مکمل باشد.
- کدهای ۲۴۲۱ و افزونی ۳ از کدهای خودمکمل هستند، اما کد BCD خود مکمل نیست.

- مکمل ۹ ارقام کد BCD:
  - راه اول: رقم ها را با ۶ جمع کنیم سپس مکمل ۱ کنیم.
  - راه دوم: رقم ها را مکمل ۱ کنیم، سپس با ده جمع کنیم و از نقلی تولید شده صرفه نظر کنیم.

جمع اعداد در BCD

- جمع دو رقم دهدهی در BCD نمی تواند بزرگتر 1+9+9 باشد که در آن ۱، رقم نقلی قبلی است.
- ارقام BCD را به شکل دودویی جمع می کنیم، حاصل جمع بین 0 تا 19 خواهد بود، این مقادیر به دودویی برابرند با 0000 تا 10011، ولی به فرم BCD برابر با 0000 تا 11001 می باشد (اولین رقم نقلی است).

معتبر ← حاصل جمع ← 1001 ≤ حاصل جمع دودویی (بدون رقم نقلی)

معتبر ← + 0110 ← نامعتبر ← حاصل جمع ← 1010 ≤ حاصل جمع دودویی

کد گری (Gray)

- کدهای حلقوی: بین هر کلمه کد و کلمه کد بعدی تنها یک بیت تغییر کرده باشد.
- از معروف ترین کدهای حلقوی کد گری است.

رقم دهدهی	Gray
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100

رقم دهدهی	Gray
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

تبدیل کد باینری به گری (Gray)

- چپ ترین بیت را تغییر نمی دهیم و سایر بیت ها را دو به دو XOR می کنیم.

$$(b_{n-1}\, b_{n-2} \ldots b_1\, b_0) = (g_{n-1}\, g_{n-2} \ldots g_1\, g_0)_{Gray}$$

$$g_{n-1} = b_{n-1}$$

$$g_{n-2} = b_{n-1} \oplus b_{n-2}$$

$$g_{n-3} = b_{n-2} \oplus b_{n-3}$$

$$\vdots$$

$$g_0 = b_1 \oplus b_0$$

تبدیل کد گری (Gray) به باینری

- mod 2 باقیمانده تقسیم بر ۲ است.

$$(b_{n-1}\, b_{n-2} \ldots b_1\, b_0) = (g_{n-1}\, g_{n-2} \ldots g_1\, g_0)_{Gray}$$

$$b_{n-1} = g_{n-1}$$

$$b_{n-2} = g_{n-2} \oplus g_{n-1} = (g_{n-2} + g_{n-1}) \bmod 2$$

$$b_{n-3} = g_{n-3} \oplus g_{n-2} \oplus g_{n-1} = (g_{n-3} + g_{n-2} + g_{n-1}) \bmod 2$$

$$\vdots$$

$$b_0 = g_0 \oplus g_1 \oplus g_2 \ldots \oplus g_{n-1} = (g_0 + g_1 + g_2 \ldots + g_{n-1}) \bmod 2$$

کدهای تشخیص و تصحیح خطا

- هنگام انتقال داده ها ممکن است در آنها خطایی به علت تداخل های الکترومغناطیسی، حرارت زیاد و غیره به وجود آید. می توان کدهایی طراحی کرد که خطا را تشخیص و یا حتی تصحیح کنند.
- یکی از ساده ترین روشهای تشخیص خطا استفاده از بیت توازن (Parity) است. می توان به هر کلمه کد یک بیت اضافه کرد به طوری که تعداد بیت های یک آن مثلا فرد باشد.

تعریف فاصله و فاصله یک کد

- فاصله (Distance)، بین دو عدد دودویی برابر تعداد اختلاف بیت های آن دو عدد است.

a = 1011
b = 1000
فاصله
d (a,b) = 2

- فاصله یک کد، برابر مینیمم فاصله بین کلمات آن است.

- کدی که فاصله اش d باشد می تواند حداکثر d-1 خطا را تشخیص دهد و یا می تواند حداکثر $\left\lfloor \frac{d-1}{2} \right\rfloor$ خطا را تصحیح کند.
- کدی که فاصله اش d باشد آنگاه قادر به تصحیح t خطا و کشف همزمان s خطای دیگر می باشد و داریم: $2t + s + 1 \leq d$

بیت توازن (parity)

- کدی که فاصله اش یک است، با افزودن یک بیت توازن به داده های آن می توان یک خطا را تشخیص داد.

- توازن دو نوع است:
  - توازن زوج (even) بیتی است که به هر عدد اضافه می کنیم تا تعداد یک هایش زوج شود.
  - توازن فرد (odd) بیتی است که به هر عدد اضافه می کنیم تا تعداد یک هایش فرد شود.

- برای محاسبه بیت توازن زوج، باید تمامی بیت های داده را با هم XOR کنیم. برای آن که تشخیص دهیم خطا ایجاد شده یا خیر باید تمامی بیت های داده همراه با بیت توازن را XOR کنیم اگر حاصل یک شد خطا رخ داده است.

کد همینگ

- فاصله کد همینگ برابر ۳ است.
- اگر داده ها 2^n بیتی باشند، همینگ n بیت به آن ها اضافه می کند.
- بیت های اضافه شده در همینگ، در مکان هایی قرار می گیرند که شماره شان توانی از ۲ باشد (۱، ۲، ۴، ۸، ۱۶....) البته به شرطی که خانه ها را از چپ به راست و با شروع از عدد ۱ شماره گذاری کنیم.

0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

C_1 = XOR (3,5,7)

C_2 = XOR (3,6,7)

C_4 = XOR (5,6,7)

0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

C_1 = XOR (3,5,7,9,11,13,15)

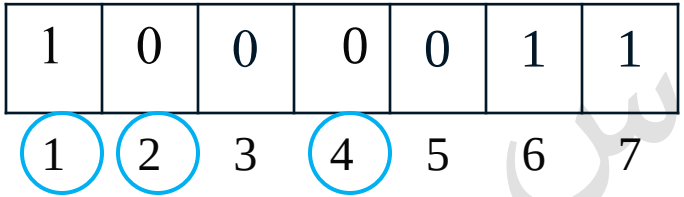
C_2 = XOR (3,6,7,10,11,14,15)

C_4 = XOR (5,6,7,12,13,14,15)

C_8 = XOR (9,10,11,12,13,14,15)

کد همینگ

داده اولیه: 0011



داده ارسالی: 1000011

داده دریافت شده توسط گیرنده : 1100011

$C_1 = XOR\ (3,5,7) = 1$

$C_2 = XOR\ (3,6,7) = 0$

$C_4 = XOR\ (5,6,7) = 0$

$D_1 = XOR\ (1,3,5,7) = 0$

$D_2 = XOR\ (2,3,6,7) = 1$

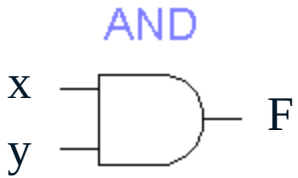
$D_4 = XOR\ (4,5,6,7) = 0$

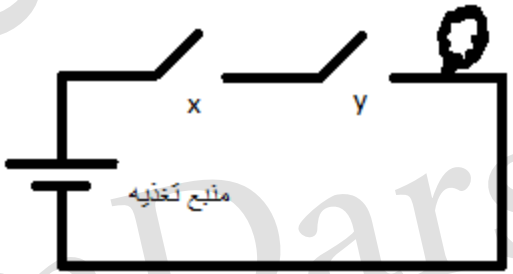
جبر بول دو ارزشی

- جبر بول، مدل ریاضی برای طراحی مدارهای منطقی است.
- در جبر بول فقط دو مقدار ۰ و ۱ وجود دارد.
- در جبر بول ۳ عملگر اصلی AND ، OR و not وجود دارد.

عملگر AND

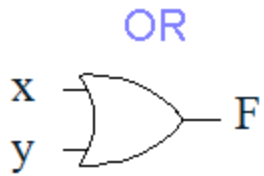
- بسته بودن کلید معادل یک و باز بودن کلید را معادل صفر فرض کنید.
- روشن بودن لامپ معادل یک و خاموش بودن لامپ را معادل صفر فرض کنید.

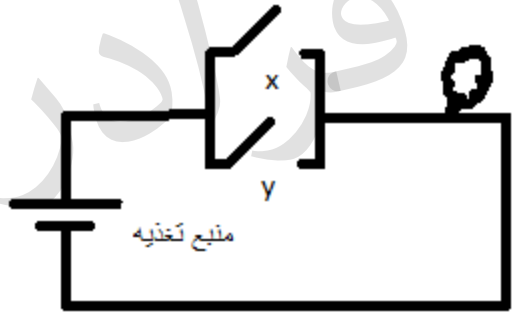




x y	F= x.y
0 0	0
0 1	0
1 0	0
1 1	1

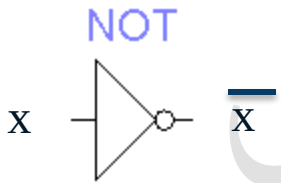
عملگر OR

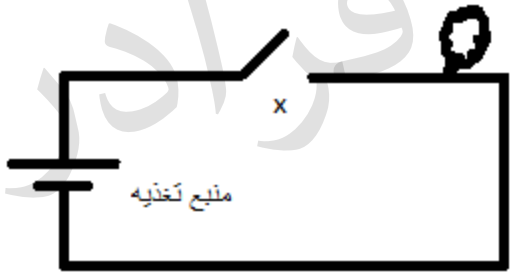




x y	F= x+y
0 0	0
0 1	1
1 0	1
1 1	1

عملگر NOT





x	F
0	1
1	0

جبر بول دو ارزشی

- متغیر بولی متغیری است که می تواند ۰ یا ۱ باشد.
- تابع بولی تابعی است که از تعدادی یا هیچ متغیر بولی تشکیل شده است.
- به هرمتغیر یا مکمل متغیر، لیترال گویند.
- تقدم عملگرها: پرانتز، not، AND، OR

توابع بولی متشکل از دو متغیر

- تعداد توابع بولی متفاوت که با n متغیر بولی می توان ساخت 2^{2^n}

ab	$\overline{a}$	$\overline{b}$	a.b	a+b	$a \uparrow b$	$a \downarrow b$	$a \oplus b$	$a \odot b$	$\overline{a}b$	$a+\overline{b}$	$a\overline{b}$	$\overline{a}+b$	a	b	0	1
00	1	1	0	0	1	1	0	1	0	1	0	1	0	0	0	1
01	1	0	0	1	1	0	1	0	1	0	0	1	0	1	0	1
10	0	1	0	1	1	0	1	0	0	1	1	0	1	0	0	1
11	0	0	1	1	0	0	0	1	0	1	0	1	1	1	0	1
	not	not	AND	OR	NAND	NOR	XOR	XNOR	a<b	a≥b	a>b	a≤b				

دوگان (dual) تابع

- در یک تابع بولی اگر AND به OR، OR به AND، 0 به 1 و 1 به 0 تبدیل شود، دوگان تابع بدست می آید.

- اصل Duality: اگر دو تابع با هم مساوی (هم ارز) باشند آن گاه دوگان آن دو تابع نیز با هم مساوی هستند.

$$a + 0 = a$$

$$a \cdot 1 = a$$

متمم تابع

- دوگان یک تابع را بدست می آوریم، سپس متمم هر متغیر را می نویسیم.

مثال:

$$F = \bar{x} \cdot y \cdot \bar{z} + \bar{x} \cdot \bar{y} \cdot z$$

دوگان تابع

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

متمم هر متغیر

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

جبر بول، جبر مجموعه ها، جبر گزاره ها

جبر بول	جبر مجموعه ها	جبر گزاره ها
AND .	اشتراک $\cap$	ترکیب عطفی $\wedge$
OR +	اجتماع $\cup$	ترکیب فصلی $\vee$
not $\overline{a}$	مکمل $\bar{A}$	نقیض $\sim A$
0	تهی $\emptyset$	False
1	مرجع M	True
XOR $\oplus$	تفاضل متقارن Δ	$\overline{\vee}$ $\underline{\vee}$ $\oplus$

خواص جبر بول		
عضو همانی (خنثی Identiy)	$a \cdot 1 = a$	$a + 0 = a$
غلبه	$a + 1 = 1$	$a \cdot 0 = 0$
خود توانی (Idempotency)	$a \cdot a = a$	$a + a = a$
عضو مکمل	$a \cdot \bar{a} = 0$	$a + \bar{a} = 1$
جابجایی (Commutative)	$a \cdot b = b \cdot a$	$a + b = b + a$
شرکت پذیری (Associatve)	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$	$a + (b + c) = (a + b) + c$
پخشی (توزیع پذیری)	$a + b \cdot c = (a + b) \cdot (a + c)$	$a \cdot (b + c) = a \cdot b + a \cdot c$
جذب	$a \cdot (a + b) = a$	$a + a \cdot b = a$
شبه جذب	$a \cdot (\bar{a} + b) = a \cdot b$	$a + \bar{a} \cdot b = a + b$
دمورگان	$\overline{(a \cdot b)} = \bar{a} + \bar{b}$	$\overline{(a + b)} = \bar{a} \cdot \bar{b}$

اثبات خاصیت جابه جایی با استفاده از جدول درستی

$$a \cdot b = b \cdot a$$

$$a + b = b + a$$

a	b	a . b	b . a	a + b	b + a
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	1	1	1

اثبات خاصیت جذب با استفاده از روابط جبری

$$a . (a+b) = a$$

بر اساس دوگانگی اثبات می شود.

$$a + a.b = a$$

$$a + a.b = a.1 + a.b$$

$$= a. (1+b)$$

$$= a.1$$

$$= a$$

ویژگی های برخی از عملگرها

- NAND و NOR جابه جا پذیر هستند ولی شرکت پذیر نیستند.
- XOR جابه جا پذیر و شرکت پذیر است.
- XOR روی هیچ عملی توزیع پذیر نیست. عمل NAND روی XOR و XNOR توزیع پذیر است.
- AND و OR حذف پذیر نیستند ولی XOR و XNOR حذف پذیرند.

$$a.b = a.c \not\rightarrow b = c$$

$$a \oplus b = a \oplus c \rightarrow b = c$$

قانون اجماع (Consensus)

1) $a.b + \overline{a}.c + b.c = a.b + \overline{a}.c$

2) $(a+b) . (\overline{a}+c) . (b+c) = (a+b) . (\overline{a}+c)$

قضیه شانون

$$1)\ f(x_1, x_2, \ldots, x_n) = \overline{x_1} \cdot f(0, x_2, \ldots, x_n) + x_1 \cdot f(1, x_2, \ldots, x_n)$$

$$2)\ f(x_1, x_2, \ldots, x_n) = [\overline{x_1} + f(1, x_2, \ldots, x_n)]\,[x_1 + f(0, x_2, \ldots, x_n)]$$

$$f(x_1, x_2, x_3 \ldots, x_n) = \overline{x_1} \cdot \overline{x_2} \cdot f(0, 0, x_3, \ldots, x_n) + \overline{x_1} \cdot x_2 \cdot f(0,1, x_3, \ldots, x_n) + x_1 \cdot \overline{x_2} \cdot f(1,0, x_3, \ldots, x_n) + x_1 \cdot x_2 \cdot f(1,1, x_3, \ldots, x_n)$$

فرم های نرمال

- (Sum of Products) SOP : اگر تابعی به صورت جمع حاصل ضرب ها باشد به آن SOP گویند.

$$F = a.b + \bar{a}.c + b.c$$

- (Product of Sums) POS : اگر تابعی به صورت ضرب حاصل جمع ها باشد به آن POS گویند.

$$G = (a+b) . (\bar{a}+c)$$

مینترم و ماکسترم

- مینترم (minterm) : جمله ای است به صورت ضرب که در آن همه لیترال ها دقیقا یک بار ظاهر شده باشد.
- مثال: با سه متغیر a، b، c جملات $a.b.c$ و $\bar{a}.b.c$ و ... مینترم هستند.
- با n متغیر بولی می توان 2^n مینترم نوشت.
- ماکسترم (Maxterm) : جمله ای است به صورت ضرب که در آن همه لیترال ها دقیقا یک بار ظاهر شده باشد.
- مثال: با سه متغیر a، b، c جملات $a+b+c$ و $\bar{a}+b+c$ و ... ماکسترم هستند.
- با n متغیر بولی می توان 2^n ماکسترم نوشت.

مینترم ها و ماکسترم ها برای ۳ متغیر

x	y	z	Minterm	Maxterm
0	0	0	$x'.y'.z' = m_0$	$x+y+z = M_0$
0	0	1	$x'.y'.z = m_1$	$x+y+z' = M_1$
0	1	0	$x'.y.z' = m_2$	$x+y'+z = M_2$
0	1	1	$x'.y.z = m_3$	$x+y'+z' = M_3$
1	0	0	$x.y'.z' = m_4$	$x'+y+z = M_4$
1	0	1	$x.y'.z = m_5$	$x'+y+z' = M_5$
1	1	0	$x.y.z' = m_6$	$x'+y'+z = M_6$
1	1	1	$x.y.z = m_7$	$x'+y'+z' = M_7$

مینترم ها و ماکسترم ها

- هر مینترم فقط در یک حالت برابر یک می شود.
- هر ماکسترم فقط در یک حالت برابر صفر می شود.
- ضرب مینترم در ماکسترم غیر هم شماره اش برابر مینترم است.
- ضرب مینترم در ماکسترم هم شماره اش برابر صفر است.
- جمع مینترم با ماکسترم غیر هم شماره اش برابر ماکسترم است.
- جمع مینترم با ماکسترم هم شماره اش برابر یک است.

جمع مینترم ها و ضرب ماکسترم ها

- جمع مینترم ها: هر تابعی را می توان به صورت منحصر به فرد به شکل جمع تعدادی مینترم نوشت که به آن Canonical SOP (CSP) یا SOP متعارف یا PDNF گویند و با $\sum$ نشان می دهیم.
- ضرب ماکسترم ها: هر تابعی را می توان به صورت منحصر به فرد به شکل ضرب تعدادی ماکسترم نوشت که به آن Canonical POS (CPS) یا POS متعارف یا PCNF گویند و با $\prod$ نشان می دهیم.

جمع مینترم ها و ضرب ماکسترم ها

- مثال: تابع $F(a,b,c) = a.\bar{b} + \bar{c}$ را بصورت جمع مینترم ها و ضرب ماکسترم ها بنویسید.

– روش اول:

$$F(a,b,c) = a.\bar{b} + \bar{c}$$

$$= a.\bar{b}\,(c+\bar{c}) + \bar{c}\,(a+\bar{a})\,(b+\bar{b})$$

$$= a.\bar{b}.c + a.\bar{b}.\bar{c} + a.b.\bar{c} + a.\bar{b}\,\bar{c}. + \bar{a}.b.\bar{c} + \bar{a}.\bar{b}.\bar{c}$$

$$= m_0 + m_2 + m_4 + m_5 + m_6 = \sum m\,(0,2,4,5,6)$$

$$= M_1.M_3.M_7 = \prod M\,(1,3,7)$$

جمع مینترم ها و ضرب ماکسترم ها

- روش دوم:

abc	F
000	$1 = m_0$
001	$0 = M_1$
010	$1 = m_2$
011	$0 = M_3$
100	$1 = m_4$
101	$1 = m_5$
110	$1 = m_6$
111	$0 = M_7$

- روش سوم:

$$F(a,b,c) = a.\bar{b} + \bar{c}$$

$$= m_0 + m_1 + m_4 + m_5 + m_6$$

ساده سازی توابع بولی

- روش های جبری
- روش نقشه (جدول) کارنو
- روش کوئین مک کلاسکی (Quine-McCluskey)
- استفاده از قضیه اجماع
- و روش های دیگر

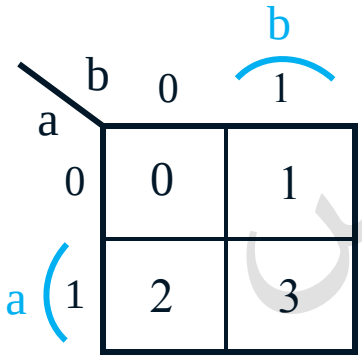
مراحل ساده سازی توابع بولی با روش جدول کارنو

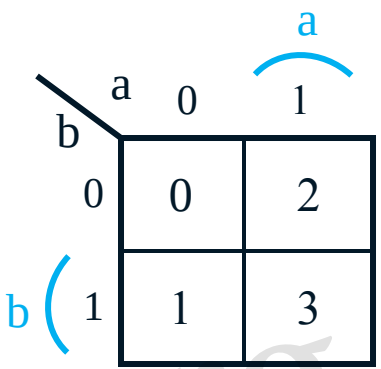
- گام اول: رسم جدول کارنو با توجه به تعداد متغیر های تابع
- گام دوم: برای ساده سازی توابع بصورت sop، مینترم های تابع را در جدول کارنو یک قرار می دهیم.
- گام سوم: همه یک های را دسته بندی می کنیم، طوری که تعداد دسته ها مینیمم باشد و دسته ها بزرگترین باشند.
- گام چهارم: تبدیل دسته ها به شکل جبری

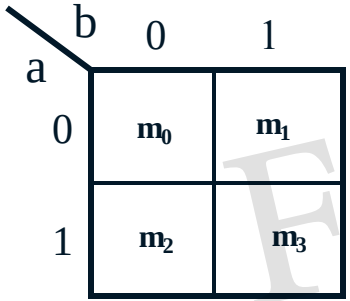
رسم جدول کارنو (دیاگرام ویچ)

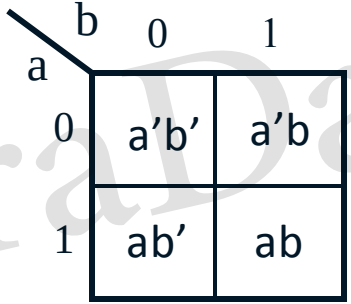
- برای ساده سازی توابع با حداکثر ۶ ورودی، می توان از جدول کارنو استفاده کرد.
- در این روش جدولی با توجه به تعداد ورودی ها در نظر گرفته می شود (تابعی که n متغیر دارد، جدولی با 2^n خانه خواهد داشت) و به هر مینترم یک خانه از این جدول اختصاص می یابد.
- در جدول کارنو، خانه های مجاور فقط یک بیت با هم اختلاف دارند (خانه ها به ترتیب کد گری شماره گذاری می شوند).

رسم جدول کارنو دو متغیره

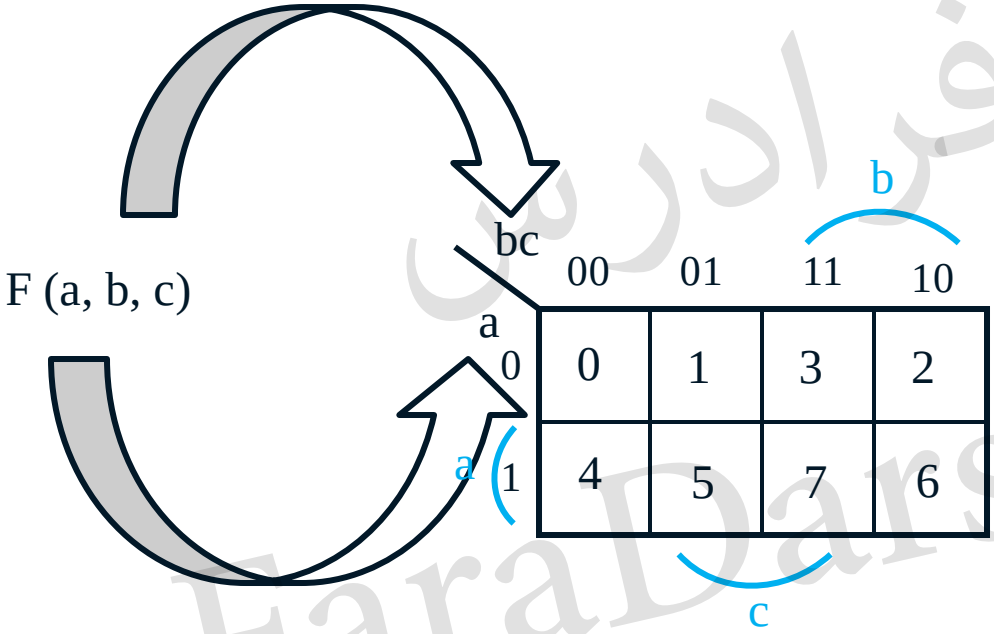




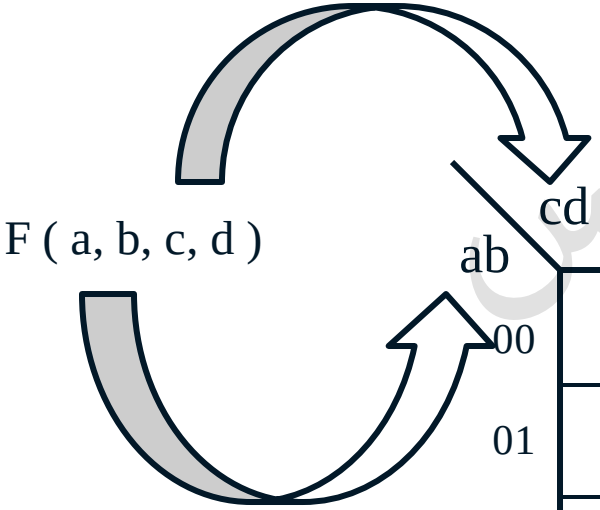




رسم جدول کارنو سه متغیره



رسم جدول کارنو چهار متغیره



ab \ cd	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

c

b

a

d

رسم جدول کارنو پنج متغیره

bc \ de	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

a=0

bc \ de	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

a=1

دسته بندی یک های جدول کارنو

- خانه های جدول را می توان به صورت سطری یا ستونی دسته بندی کرد.
- تعداد خانه های هر دسته باید توانی از دو باشد.
- در هر دسته حداقل یک، ۱ موجود باشد که جزء دسته های دیگر نباشد.

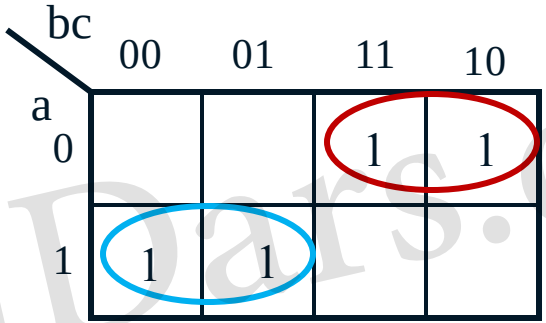
- اگر در جدول کارنو، صفرها را دسته بندی کنیم و مشابه یک ها، جملات را بنویسیم و جمع کنیم، مکمل تابع، به فرم sop ساده می شود که اگر مکمل کنیم، خود تابع به فرم pos حاصل می شود.

مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c) = \sum m\,(2,3,4,5)$$

$$F(a,b,c) = \bar{a}.b + a.\bar{b}$$

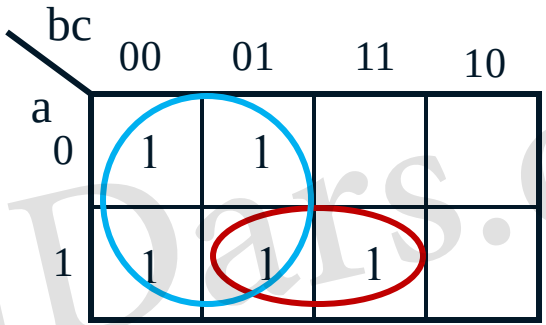


مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c) = \sum m\,(0,1,4,5,7)$$

$$F(a,b,c) = \overline{b} + a.c$$

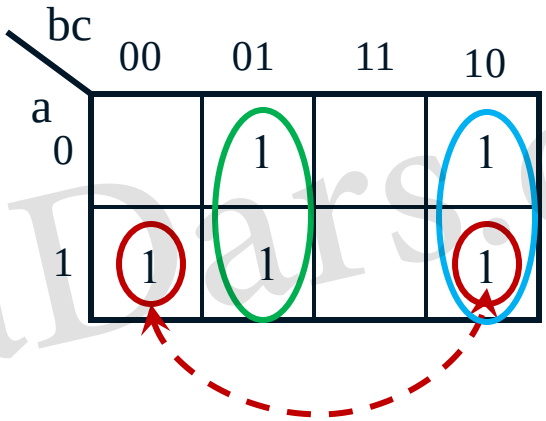


مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c) = \sum m\,(1,2,4,5,6)$$

$$F(a,b,c) = \overline{b}.c + b.\overline{c} + \overline{a}.c$$

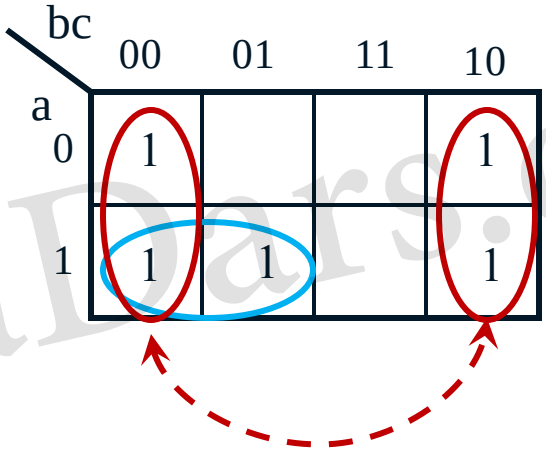


مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c) = \sum m\,(0,2,4,5,6)$$

$$F(a,b,c) = \overline{c} + a.\overline{b}$$



مثال های ساده سازی توابع بولی با روش جدول کارنو

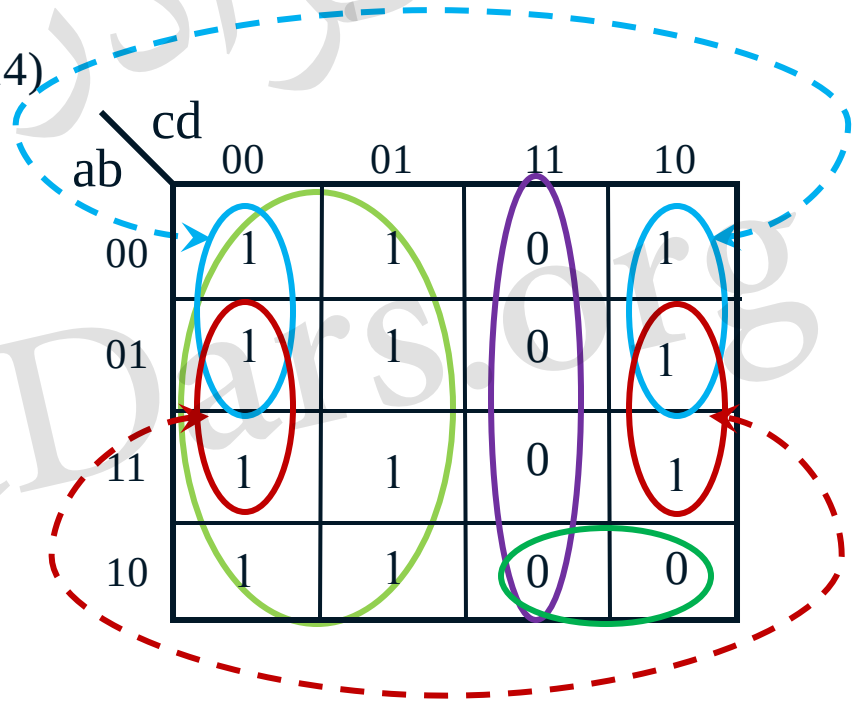
- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c,d) = \sum m\,(0,1,2,4,5,6,8,9,12,13,14)$$

$$F(a,b,c,d) = \bar{c} + \bar{a}.\bar{d} + b.\bar{d}$$

$$\overline{F}(a,b,c,d) = c.d + a.\bar{b}.c$$

$$F(a,b,c,d) = (\bar{c}+\bar{d}) . (\bar{a}+b+\bar{c})$$

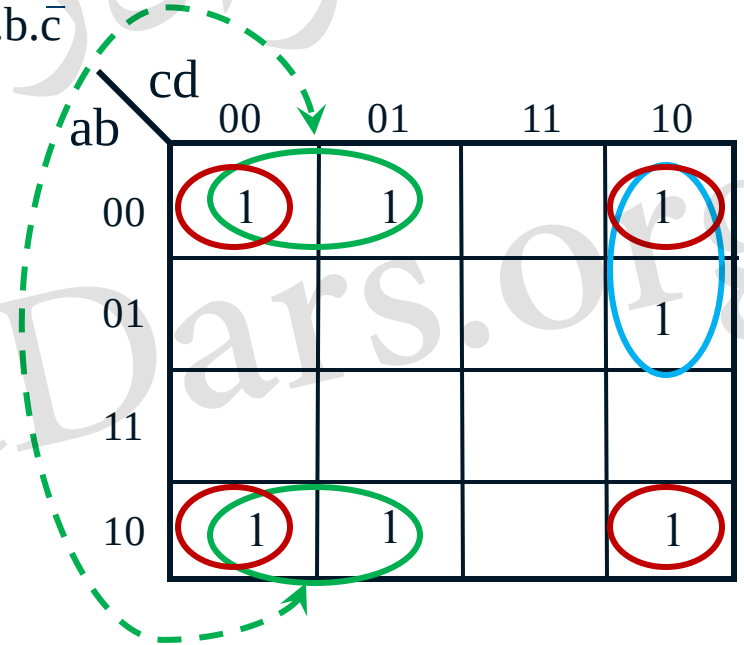


مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

$$F(a,b,c,d) = \bar{a}.b.\bar{c} + b.c.\bar{d} + \bar{a}.b.c.\bar{d} + a.\bar{b}.\bar{c}$$

$$F(a,b,c,d) = b.\bar{c} + \bar{a}.c.\bar{d} + \bar{b}\bar{d}$$

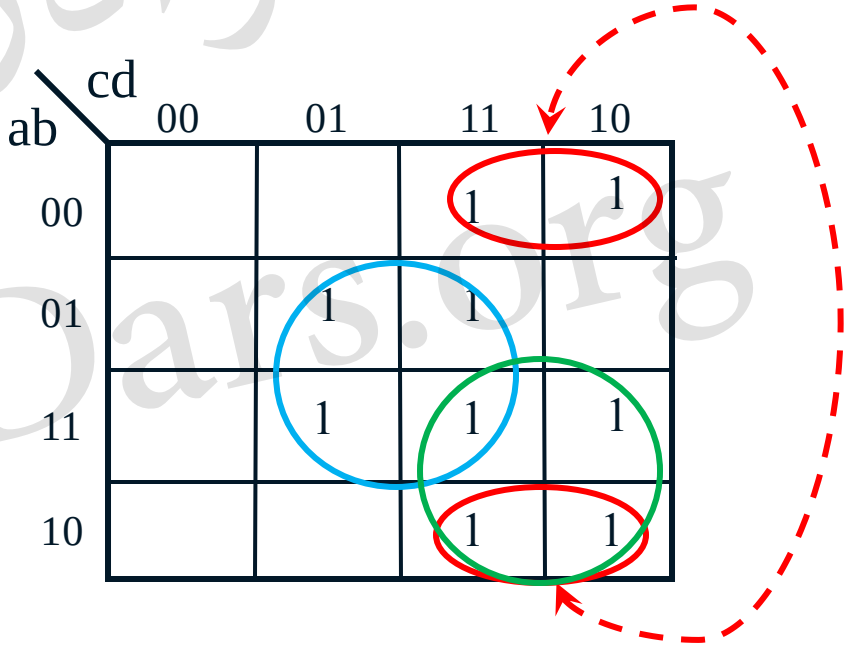


مثال های ساده سازی توابع بولی با روش جدول کارنو

- مثال: تابع داده شده را به صورت sop ساده کنید.

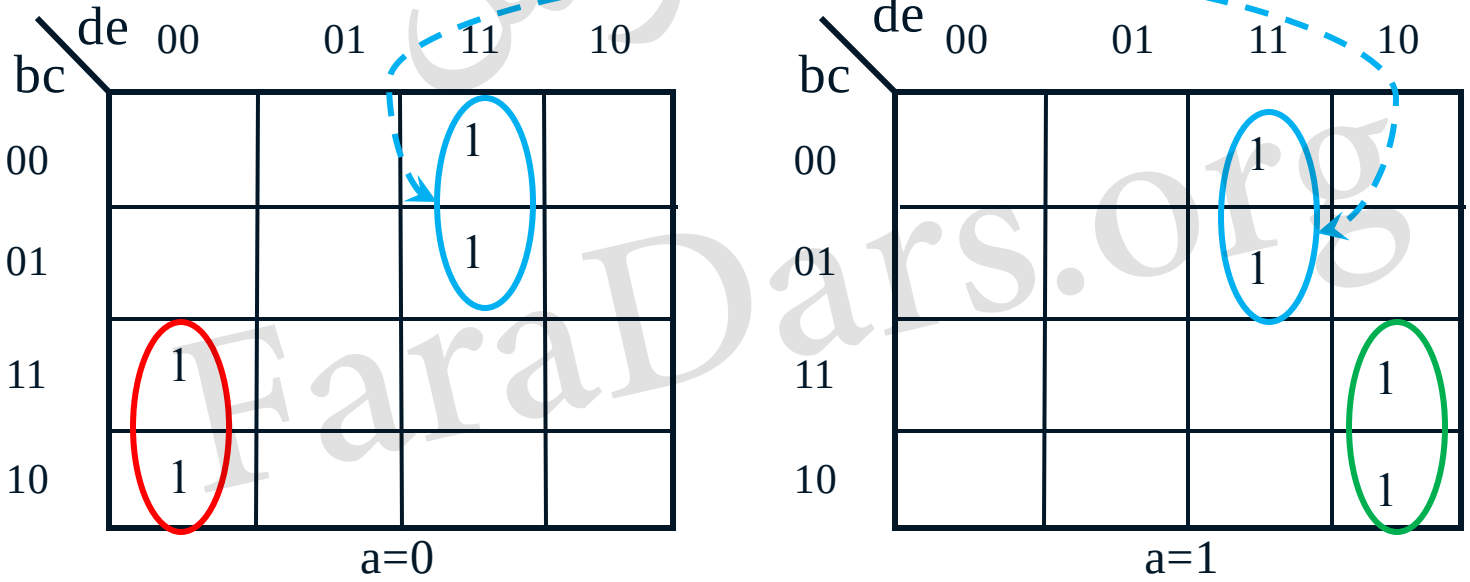
$$F(a,b,c,d) = a.b.c + c.d + b.\bar{c}.d + \bar{b}.c$$

$$F(a,b,c,d) = b.d + a.c + \bar{b}.c$$



مثال های ساده سازی توابع بولی با روش جدول کارنو

$$F(a,b,c,d,e) = \sum m\,(3,7,8,12,19,23,26,30) = \bar{a}.b.d.\bar{e} + a.b.d.\bar{e} + \bar{b}.d.e$$



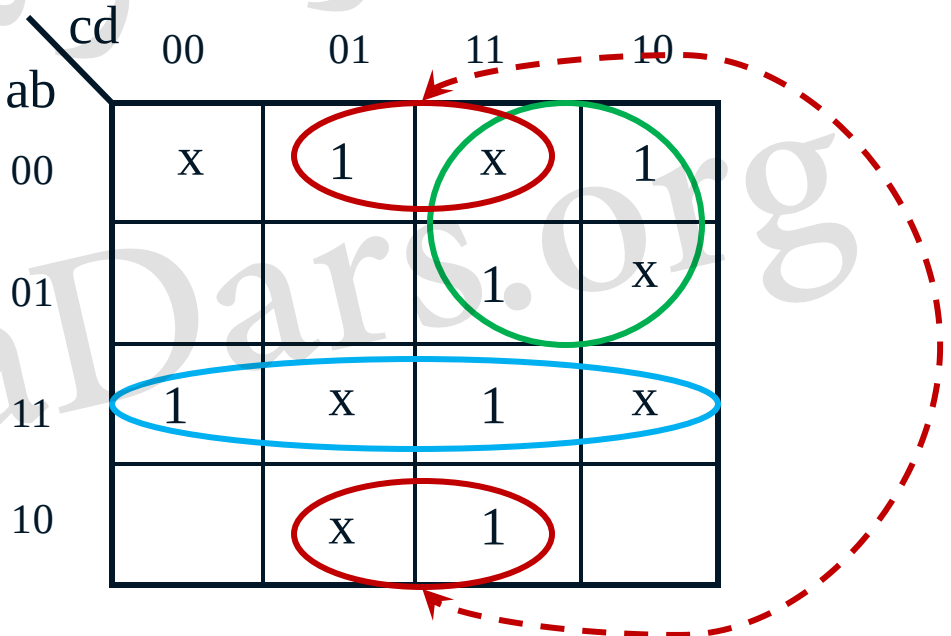
حالت بی اهمیت (Don’t Care) در جدول کارنو

- توابعی که در ازاء ترکیبی از ورودی ها خروجی های نامشخص دارند، تابع غیر کامل نامیده می شود.
- حالات بی اهمیت حالاتی هستند که می توان آن ها را یک یا صفر فرض کرد.
- حالات بی اهمیت کمک می کنند که دسته ها بزرگتر شوند در نتیجه تابع ساده تر شود.

ساده سازی توابع بولی با روش جدول کارنو با حالت های بی اهمیت

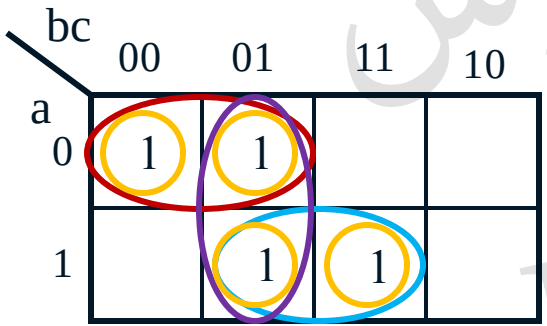
$$F(a,b,c,d) = \sum m(1,2,7,11,12,15) + d(0,3,6,9,13,14)$$

$$F(a,b,c,d) = \bar{a}.c + a.b + \bar{b}.d$$



ایجاب کننده (Implicant)

- جمله ای است به صورت ضرب که شامل یک یا چند مینترم باشد.

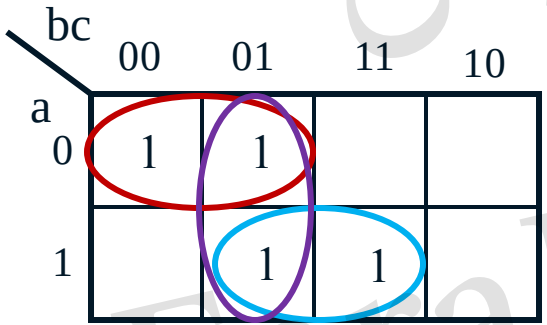


$\bar{a}.\bar{b}.\bar{c}$, $\bar{a}.\bar{b}.c$, $a.\bar{b}.c$, $a.b.c$,

$\bar{a}.\bar{b}$, $\bar{b}.c$, $a.\bar{c}$

ایجاب کننده اولیه (Prime Implicant:PI)

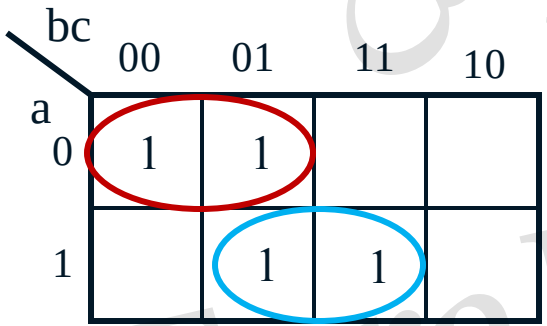
- ایجاب کننده ای است که توسط هیچ ایجاب کننده دیگری پوشش داده نشود، یعنی دسته ای است که درون دسته بزرگتری نباشد.



$\bar{a}.b,\ b.c,\ a.\bar{c}$

ایجاب کننده اولیه اساسی (Essential Prime Implicant :EPI)

- یک PI که حداقل یک مینترم را که توسط هیچ PI دیگری پوشش داده نشده است، پوشش دهد.



$\bar{a}.b, a.\bar{c}$

کوئین مک کلاسکی (Quine-McCluskey)

- ویژگی های روش QM:
  - روش QM یک روش الگوریتمی است که به راحتی قابل پیاده سازی ماشینی است.
  - بسیار زمان بر است.
  - برای هر تعداد متغیر جواب می دهد.
  - همزمان چندین تابع را می توان با روش QM ساده کرد.

ساده سازی توابع بولی با استفاده کوئین مک کلاسکی

- مثال: تابع داده شده را با روش QM به صورت sop ساده کنید.

$$F(a,b,c,d) = \sum m(2,4,6,8,9,10,12,13,15)$$

- گام اول: معادل دودویی همه مینترمهای تابع را پیدا کنید و در یک ستون عمودی زیر یکدیگر می نویسیم سپس مینترم ها را بر اساس تعداد یک هایشان دسته بندی می کنیم، مثلا مینترم ۲(۰۰۱۰) و مینترم ۴ (۰۱۰۰) هر یک دارای یک ۱ در شکل دودویی شان هستند و در گروه ۱ قرار دارند.

Minterms	a b c d
2	0 0 1 0
4	0 1 0 0
8	1 0 0 0
6	0 1 1 0
9	1 0 0 1
10	1 0 1 0
12	1 1 0 0
13	1 1 0 1
15	1 1 1 1

- گام دوم: مینترم های گروه های مجاور را با هم مقایسه می کنیم (گروه ۱ با ۲، گروه ۲ با ۳ و ...)، هر دو مینترمی که با هم یک بیت اختلاف دارند (مثل ۰۰۱۰ و ۰۱۱۰) را انتخاب می کنیم و محل اختلافشان علامت – را می گذاریم (۱۰-۰) و در ستون جدیدی می نویسیم و آن مینترم ها را علامت ✓ می زنیم. همین روند را برای ستون های جدید تکرار می کنیم.

Minterms	a b c d		Minterms	a b c d		Minterms	a b c d	
2	0010	✓	2,6	0-10	PI2	8,9,12,13	1-0-	PI1
4	0100	✓	2,10	-010	PI3			
8	1000	✓	4,6	01-0	PI4			
6	0110	✓	4,12	-100	PI5			
9	1001	✓	8,9	100-	✓			
10	1010	✓	8,10	10-0	PI6			
12	1100	✓	8,12	1-00	✓			
13	1101	✓	9,13	1-01	✓			
15	1111	✓	12,13	110-	✓			
			13,15	11-1	PI7			

- گام سوم: جملاتی که علامت ✓ ندارند، PI هستند، جملات تکراری را حذف و جدول پوشش (Coveing table) رسم می کنیم.
- جدول پوشش: در جدول پوشش تمام مینترم های تشکیل دهنده تابع را به صورت افقی بالای جدول و PI ها را به صورت عمودی سمت چپ جدول می نویسیم. در داخل جدول با علامت * مشخص می کنیم هر PI چه مینترمی را شامل می شود.

- گام چهارم: کمترین تعداد PI هایی که همه مینترم های تابع را در جدول پوشش شامل می شود، انتخاب می کنیم.

  - در جدول پوشش ستون هایی که فقط یک علامت * دارند (مثل ستون ۱۵ و ۹) به این معنی است که PI آن ها اساسی است.

	2	4	6	8 ✓	9 ✓	10	12 ✓	13 ✓	15 ✓
PI_1				*	*		*	*	
PI_2	*		*						
PI_3	*					*			
PI_4		*	*						
PI_5		*					*		
PI_6				*		*			
PI_7								*	*

$$f(a,b,c,d) = PI_1 + PI_3 + PI_4 + PI_7$$

$$= 1\text{-}0\text{-} + \text{-}010 + 01\text{-}0 + 11\text{-}1$$

$$= a.c'_+ b'.c.d'_+ a'.b.d'_+ a.b.d$$

ساده سازی توابع بولی با استفاده کوئین مک کلاسکی

- مثال: تابع داده شده را با روش QM به صورت sop ساده کنید.

$$F(A, B, C, D, E) = \sum m(2, 3,7, 10, 12, 15, 27) + d(5, 18, 19, 21, 23)$$

- توجه: مینترم ها و حالت های بی اهمیت با هم دسته بندی می شوند، ولی در جدول پوشش فقط مینترم ها را قرار می دهیم.

Minterms	ABCDE	
2	00010	✓
3	00011	✓
5	00101	✓
10	01010	✓
12	01100	PI7
18	10010	✓
7	00111	✓
19	10011	✓
21	10101	✓
15	01111	✓
23	10111	✓
27	11011	✓

Minterms	ABCDE	
2,3	0001-	✓
2,10	0-010	PI4
2,18	-0010	✓
3,7	00-11	✓
3,19	-0011	✓
5,7	001-1	✓
5,21	-0101	✓
18,19	1001-	✓
7,15	0-111	PI5
7,23	0-111	✓
19,23	10-11	✓
19,27	1-011	PI6
21,23	101-1	✓

Minterms	ABCDE	
2,3,18,19	-001-	PI1
~~2,18,3,19~~		
3,7,19,23	-0-11	PI2
~~3,19,7,23~~		
5,7,21,23	-01-1	PI3
~~5,21,7,23~~		

	2	3	7	10	12	15	27
PI_1	*	*					
PI_2		*	*				
PI_3			*				
PI_4	*			*			
PI_5			*			*	
PI_6							*
PI_7					*		

$$F(A,B,C,D) = PI_1 + PI_4 + PI_5 + PI_6 + PI_7$$

$$F(A,B,C,D) = PI_2 + PI_4 + PI_5 + PI_6 + PI_7$$

ساده سازی توابع بولی با استفاده قضیه اجماع

$a.b + \overline{a}.c + b.c = a.b + a.\overline{c}$

X. term1+ X’. term2 + term1 . term2 = X. term1+ X’. term2

termA = X. term1

termB = X. term2

term1.term2 = اجماع بین دو جمله

$a + a.c = a$ قانون جذب

X + X.term1 = X

روند ساده سازی توابع بولی با استفاده قضیه اجماع

- گام اول: تمام جملات حاصلضرب تابع را زیر هم می نویسیم.
- گام دوم: با شروع از اولین حمله و از بالا هر جمله را با تمام جمله های بالاترش مقایسه میکنیم.
- گام سوم: اگر هنگام مقایسه دو جمله قانون جذب قابل اعمال باشد ، با این قانون یکی از جملات را حذف می کنیم.
- گام چهارم: اگر هنگام مقایسه بین آنها اجماع وجود داشته باشد جمله اجماع بین انها را با تمام جملات موجود در فهرست مقایسه می کنیم. اگر در هیچ موردی قانون جذب قابل اعمال نبود این جمله را به انتهای فهرست اضافه می کنیم.
- گام پنجم: این روال را تا زمانی که هر جمله با تمام جمله های بالاتر خود مقایسه شود ادامه می دهیم.

مثال ساده سازی توابع بولی با استفاده قضیه اجماع

- مثال: تابع داده شده را با استفاده از قضیه اجماع ساده کنید.

$F(a,b,c,d,e) = ab'c'de + ab'de' + abde' + ab'cde$

1- ~~$ab'c'de$~~ } $ab'c'd$
2- ~~$ab'de'$~~ } ade'
3- ~~$abde'$~~
4- ~~$ab'cde$~~ } $ab'de$
5- ~~$ab'c'd$~~
6- ade' } $ab'd$
7- ~~$ab'de$~~
8- $ab'd$

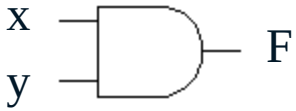
$F(a,b,c,d,e) = ade' + ab'd$

گیت های منطقی

- گیت های منطقی عناصر اصلی مدارهای منطقی هستند.
- هر تابع منطقی با ترکیبی از سه گیت AND ، OR و Not قابل ساخت و پیاده سازی است.
- گیت کامل : گیتی کامل است که همه توابع منطقی را بتوان با آن ایجاد کرد.

گیت های منطقی

AND

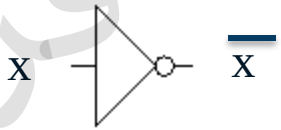


x y	F= x.y
0 0	0
0 1	0
1 0	0
1 1	1

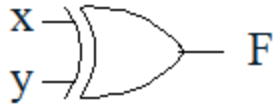
OR

x y	F= x+y
0 0	0
0 1	1
1 0	1
1 1	1

NOT



XOR

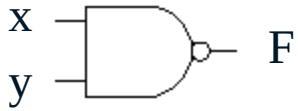


x y	F= x⊕y
0 0	0
0 1	1
1 0	1
1 1	0

گیت های منطقی

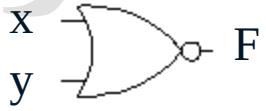
- علامت دایره کوچک در نمودارهای منطقی، نشانه متمم شدن متغیر مربوطه یا وجود یک گیت Not است.

NAND



x y	F= (x.y)’
0 0	1
0 1	1
1 0	1
1 1	0

NOR



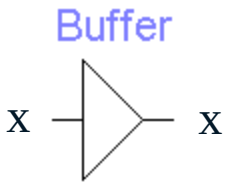
x y	F= (x+y)’
0 0	1
0 1	0
1 0	0
1 1	0

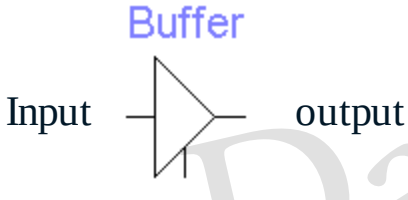
XNOR

x y	F= x ⊙ y
0 0	1
0 1	0
1 0	0
1 1	1

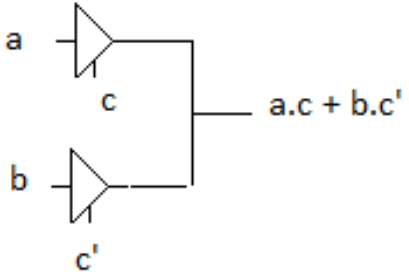
گیت های منطقی

- گیت Buffer هیچ تغییر منطقی ایجاد نمی کند. در عمل برای تطبیق امپدانس یا تقویت جریان استفاده می شود.
- همه گیت ها بجز NOT و Buffer می توانند هر تعداد ورودی داشته باشند.

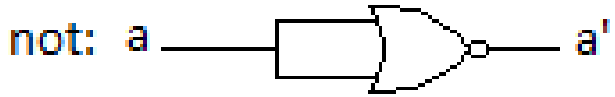


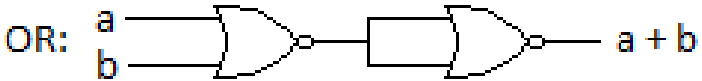


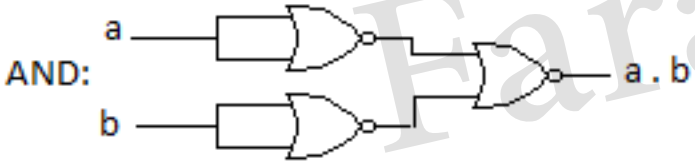
Output = Input. Control



گیت کامل NOR

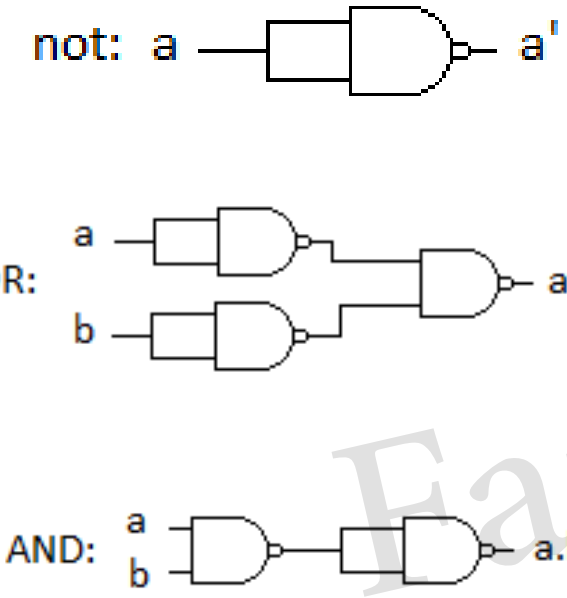






a	b	a NOR b	(a NOR b)’	a’	b’	a’ NOR b’
0	0	1	0	1	1	0
0	1	0	1	1	0	0
1	0	0	1	0	1	0
1	1	0	1	0	0	1

گیت کامل NAND



a	b	a NAND b	(a NAND b)’	a’	b’	a’ NAND b’
0	0	1	0	1	1	0
0	1	1	0	1	0	1
1	0	1	0	0	1	1
1	1	0	1	0	0	1

نکات گیت های منطقی

- اگر تمام ورودی های گیت NOR متمم شود خروجی آن معادل با خروجی گیت AND است.
- اگر تمام ورودی های گیت NAND متمم شود خروجی آن معادل با خروجی گیت OR است.
- اگر تمام ورودی های گیت OR متمم شود خروجی آن معادل با خروجی گیت NAND است.
- اگر تمام ورودی های گیت NAND متمم شود خروجی آن معادل با خروجی گیت NOR است.

انواع مدل های کامل

- کامل قوی: فقط با وجود متغیرهای $x_1, x_2, \ldots, x_n$ همه توابع با این n متغیر تولید کرد. مجموعه های {NAND}, {NOR}, {AND, NOT}, {OR, NOT} کامل قوی هستند.

- کامل ضعیف: با وجود متغیرهای $0, 1, x_1, x_2, \ldots, x_n$ همه توابع با این n متغیر تولید کرد. مجموعه های {XOR, OR}, {XOR, AND} کامل ضعیف هستند.

- کامل متممی قوی: ورودی عبارت است از $x_1, x_2, \ldots, x_n , \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$. مجموعه {AND, OR} کامل متممی قوی است.

- کامل متممی ضعیف: ورودی عبارت است از $0, 1 , x_1, x_2, \ldots, x_n , \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$

الگوریتم پیاده سازی تابع فقط با گیت NAND

- تابع را به صورت SOP ساده کنید.
- شکل تابع را با گیت های AND – OR رسم کنید.
- با مکمل کردن خروجی AND و ورودی OR، شکل تابع را به NAND – NAND تبدیل می کنیم.

الگوریتم پیاده سازی تابع فقط با گیت NOR

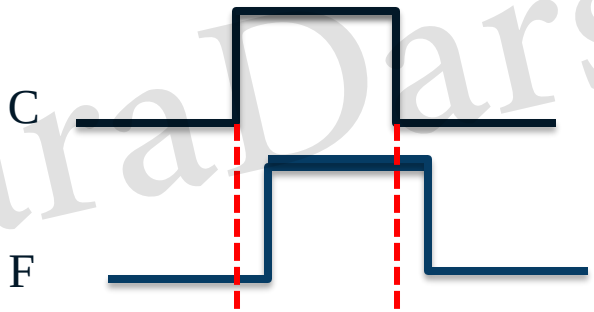
- تابع را به صورت POS ساده کنید.
- شکل تابع را با گیت های AND –OR رسم کنید.
- با مکمل کردن خروجی OR و ورودی AND، شکل تابع را به NOR – NOR تبدیل می کنیم.

نکات گیت های منطقی

- اگر ورودی ها Single Rail بودند یعنی فقط متغیرها (و نه not آنها) در ورودی ها هستند.
- اگر ورودی ها Double Rail بودند یعنی متغیرها و not آنها در ورودی ها هستند.

تاخیر انتشار (Propagation delay)

- تاخیر انتشار مدت زمانی است که طول می کشد تا تغییرات ورودی یک گیت به خروجی آن برسد.
- تاخیر انتشار تابعی از پیچیدگی مدار، تکنولوژی ساخت، دما، ولتاژ تراشه و تعداد ورودی های گیت های دیگری که خروجی گیت مورد نظر ما می تواند تغذیه کند، است.
- در برخی تکنولوژی ها، تاخیر زمانی که خروجی می خواهد از صفر به یک تغییر کند با زمانی که خروجی می خواهد از یک به صفر تغییر کند متفاوت است. معمولا تاخیر انتشار متوسط این دو زمان است.



مخاطره (Hazard)

- تاخیر انتشار در برخی از مدارات موجب بروز پدیده ناخواسته ای به نام مخاطره می شود.
- مخاطره منطقی (Logic Hazard) : فقط یک ورودی عوض می شود.
- مخاطره تابعی (Function Hazard) : بیش از یک ورودی همزمان عوض می شود.

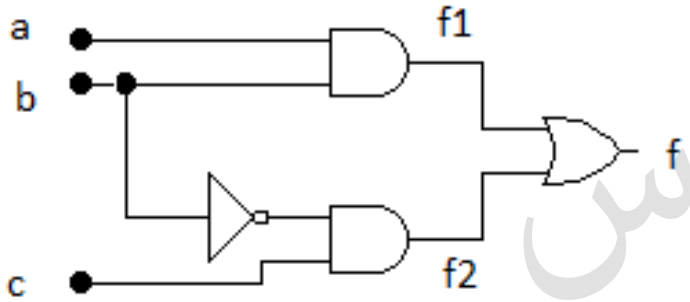
- مخاطره منطقی:
  - استاتیک:
    ۱) استاتیک سطح یک
    ۲) استاتیک سطح صفر
  - دینامیک

مخاطره ایستای سطح یک (Static Hazard)

- حالتی است که خروجی یک است و با تغییر ورودی نباید تغییر کند ولی برای مدت کوتاهی به طور ناخواسته صفر می شود و مجددا یک می شود. یعنی یک پالس ناخواسته منفی در خروجی ظاهر می شود.

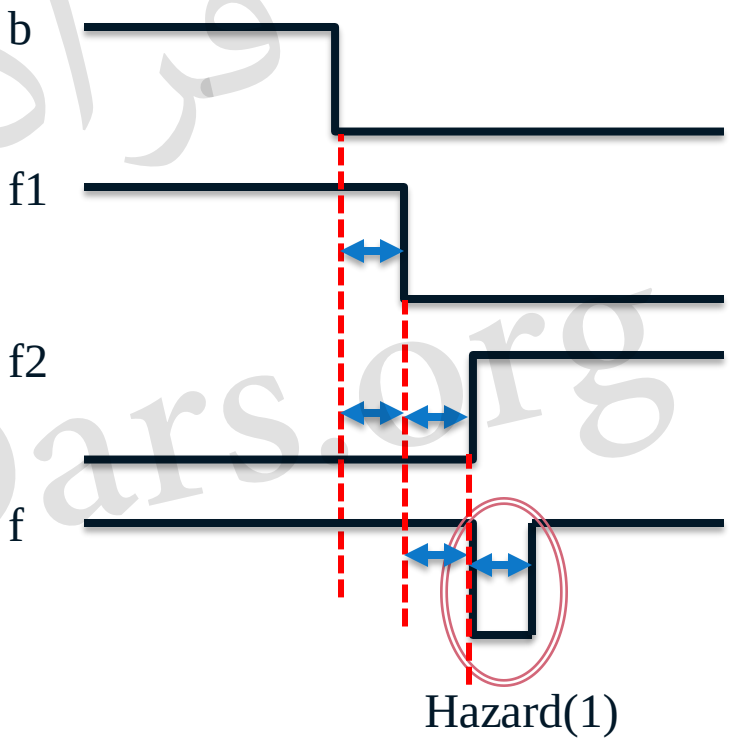
- مخاطره ایستای سطح یک معمولا در مدارات SOP (NAND-NAND, AND-OR) پیش می آید.

مثال: فرض می کنیم تاخیر هر گیت 2ns است.

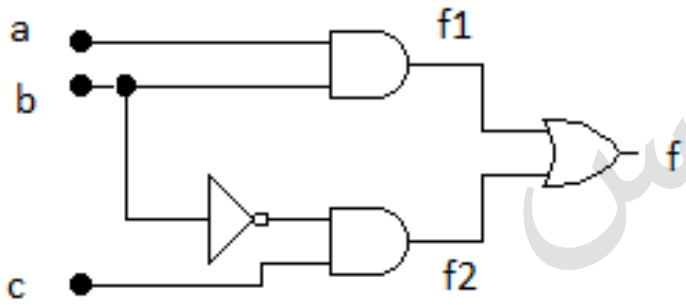


a b c = 1 1 1

f1= 1 , f2= 0 , f= 1

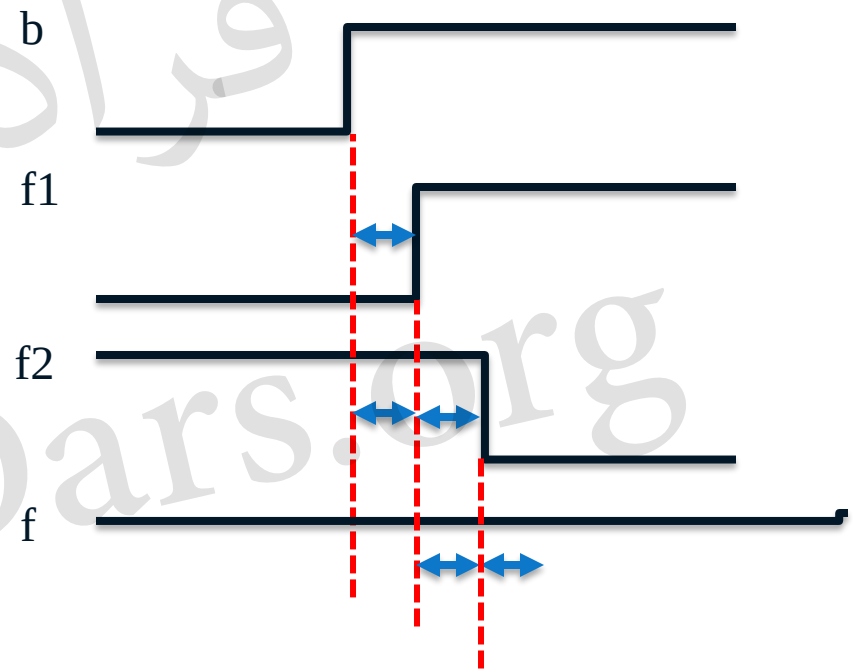


مثال: فرض می کنیم تاخیر هر گیت 2ns است.

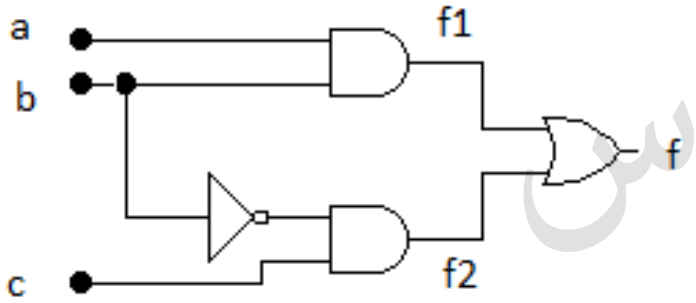


a b c = 1 0 1

f1= 0 , f2= 1 , f= 1

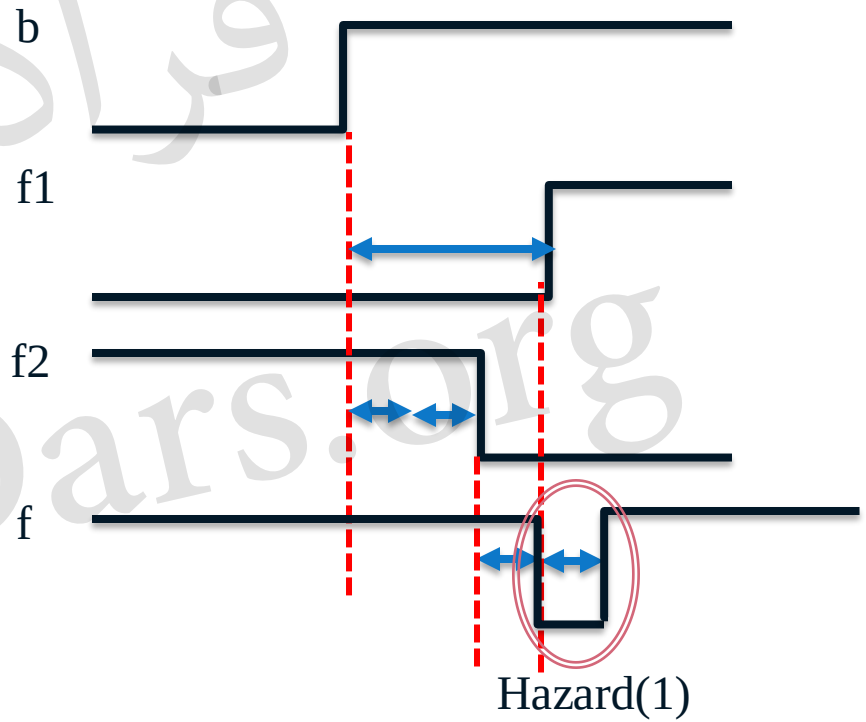


مثال: فرض کنید تاخیر گیت and مربوط به f1 ، 6ns و باقی گیت ها دارای تاخیر گیت 2ns باشند.



a b c = 1 0 1

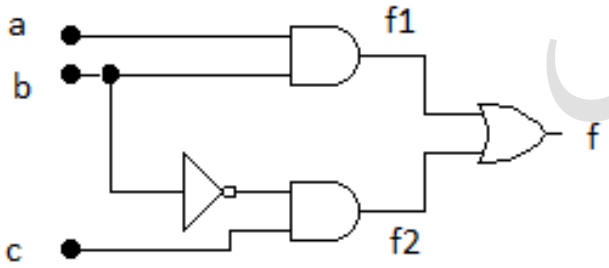
f1= 0 , f2= 1 , f= 1



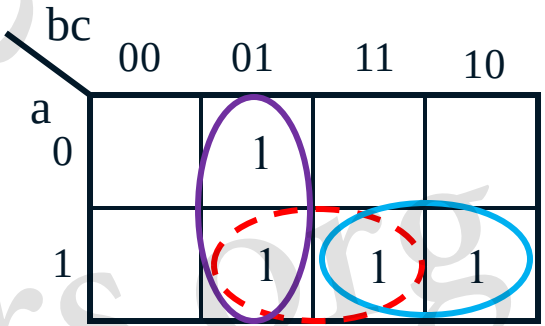
رفع مخاطره ایستای سطح یک (Static Hazard)

1. تابع خروجی را بنویسیم.
2. جدول کارنو را تشکیل دهیم.
3. اگر در جدول کارنو دو تا مینترم مجاور هم باشند و در یک دسته نباشند به معنی وجود مخاطره است، پس باید دسته ای به تابع اضافه کنیم که شامل آن دو مینترم باشد.

- مثال: رفع مخاطره سطح یک



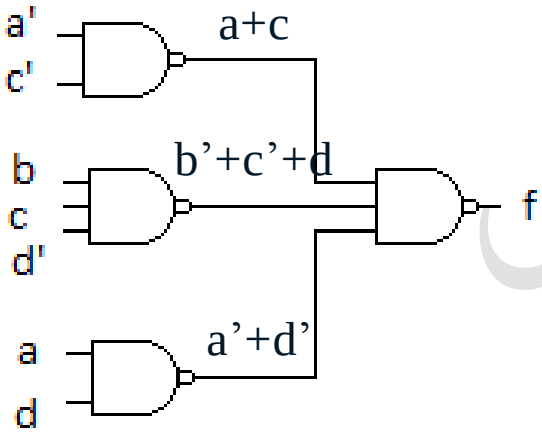
abc	f1	f2	F
000	0	0	0
001	0	1	1
010	0	0	0
011	0	0	0
100	0	0	0
101	0	1	1
110	1	0	1
111	1	0	1



$f = ab + b'c$

$f = ab + b'c + ac$

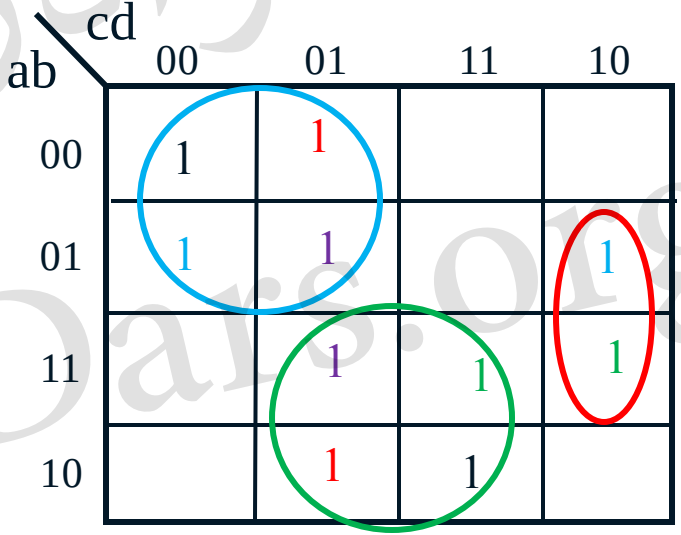
- مثال: مدار شکل مقابل در چند وضعیت مخاطره دارد و چگونه رفع مخاطره کنیم.



$F(a,b,c,d) = a'c' + bcd' + ad$

$+ abc + a'bd' + c'd$

c'd b , c'db'



مخاطره ایستای سطح صفر (Static - O - Hazard)

- حالتی است که خروجی صفر است و صفر نیز باید بماند ولی برای مدت کوتاهی ناخواسته یک می شود.

- مخاطره ایستای سطح صفر معمولا در مدارات POS (OR-AND , NOR– NOR) پیش می آید.

رفع مخاطره ایستای سطح صفر (Static - O - Hazard)

1. تابع خروجی را بنویسیم.
2. جدول کارنو را تشکیل دهیم.
3. اگر دو تا صفر مجاور هم در یک دسته نباشند به معنی وجود این نوع مخاطره است که آن ها را در یک دسته قرار می دهیم.

مخاطره دینامیک

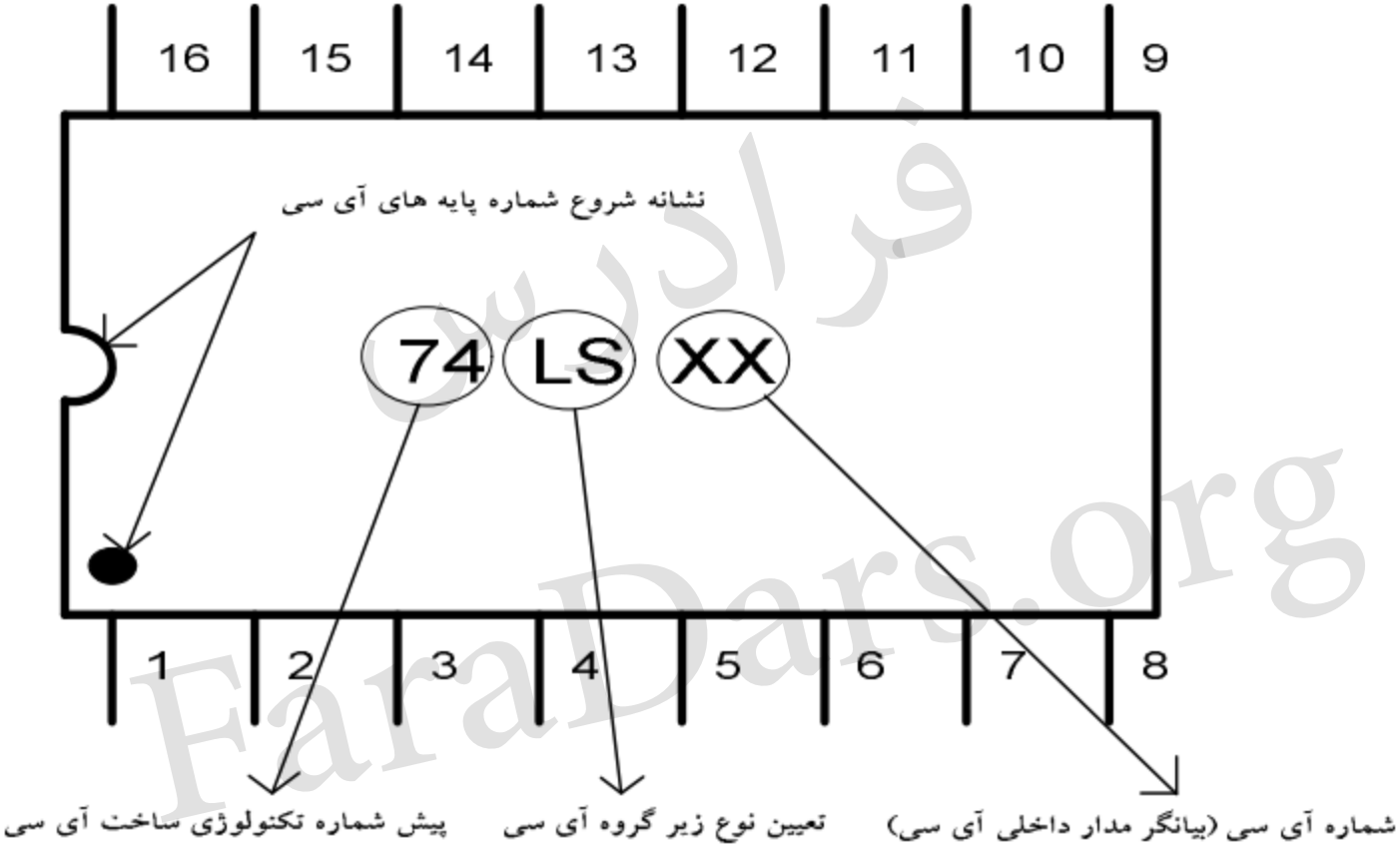
- خروجی بیش از یک تغییر ناخواسته دارد.

مدارهای مجتمع (Integrated Circuits)(IC)

- یک مدار مجتمع (IC) یک کریستال نیمه هادی از جنس سیلیکان است که به آن تراشه می گویند و حاوی اجزاء الکترونیکی در ساخت گیت های دیجیتال می باشد. انواع گیت ها در داخل تراشه به هم وصل می شوند تا مدار مورد نیاز ایجاد گردد. تراشه روی یک محفظه سرامیک یا پلاستیک نصب شده و اتصالات به پایه های بیرون برای ایجاد مدار مجتمع، متصل می گردد.

تقسیم بندی مدارات مجتمع براساس تعداد گیت های آنها

- مدارات مجتمع دارای سطوح فشردگی متفاوتی هستند:
  - SSI (Small Scale Integration) : کمتر از ۱۰ گیت در یک بسته.
  - MSI (Medium Scale Integration) : بین ۱۰ تا ۱۰۰۰ گیت در یک بسته. مانند دیکدر، مالتی پلکسر...
  - LSI (Large Scale Integration) : هزاران گیت در یک بسته. مانند حافظه ها، مدارات منطقی برنامه پذیر...
  - VLSI (Very Large Scale Integration) : صدها هزار گیت در یک بسته. مانند میکرو کامپیوترها، آرایه های حافظه پذیر ...



تقسیم بندی مدارات مجتمع براساس تکنولوژی ساخت

- RTL : Resistor Transistor Logic
- DTL: Diod Transistor Logic
- TTL: Transistor Transistor Logic
- ECL: Emitter Coupled Logic
- MOS: Metal Oxid Semiconductore
- CMOS: Complementary Mental Oxid Semiconductore

تقسیم بندی مدارات مجتمع براساس تکنولوژی ساخت

- RTL : Resistor Transistor Logic
- DTL: Diod Transistor Logic
- TTL: Transistor Transistor Logic
- ECL: Emitter Coupled Logic
- MOS: Metal Oxid Semiconductore
- CMOS: Complementary Mental Oxid Semiconductore

RTL (منطق مقاومت- ترانزیستور) و DTL (منطق دیود- ترانزیستور)

- فقط دارای ارزش تاریخی اند.
- گیت پایه در RTL گیت NOR است.
- گیت پایه در DTL گیت NAND است.

TTL (منطق ترانزیستور - ترانزیستور)

- یکی از رایج ترین انواع مدارهای مجتمع که با ولتاژ 5 ولت کار می کنند.
- IC های TTL تجاری با شماره هایی که اول آنها 74 است مشخص می شوند.
- تفاوت میان سری های TTL در منطق دیجیتال آنها نیست بلکه ساختار داخلی همه آنها بر مبنای گیت NAND است.
- آرایش خروجی همه گیت های TTL :
  1. خروجی کلکتور باز (open collector)
  2. خروجی توتم پل (totem - pole)
  3. خروجی سه حالته

انواع TTL و مشخصات آنها

نام سری TTL	پیشوند	گنجایش خروجی	توان مصرفی (mW)	تاخیر انتشار (ns)	حاصلضرب توان سرعت (pJ)
استاندارد	74	10	10	9	90
توان پایین	74L	20	1	33	33
سرعت بالا	74H	10	22	6	132
شوتکی	74S	10	19	3	57
شوتکی توان پایین	74LS	20	2	9.5	19
شوتکی پیشرفته	74AS	40	10	1.5	15
شوتکی پیشرفته توان پایین	74ALS	20	1	4	4
سریع	74F	20	4	3	12

ECL (منطق کوپلاژ امیتر)

- ECL سریع ترین خانواده است اما حد پارازیت و توان مصرفی آن بدترین است.
- ECL ها به علت تاخیر کم و مصرف زیاد در مدارهای فرکانس بالا کاربرد دارند.
- ECL معمولا با ولتاژهای تغذیه صفر و 5.2 – ولت کار می کنند.
- ECL ها دارای OR و NOR هستند.
- اگر خروجی دو گیت NOR این خانواده به هم متصل شوند اتصال معادل OR است.
- اگر خروجی دو گیت OR این خانواده به هم متصل شوند اتصال معادل AND است.

MOS (فلز – اکسید – نیمه هادی)

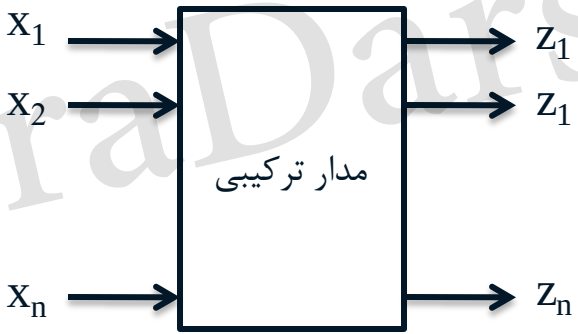
- ترانزیستورهای استفاده شده در MOS ، برخلاف ترانزیستورهای TTL و ECL که دو قطبی هستند، تک قطبی اند و سطح کمتری را اشغال می کنند.

CMOS

- CMOS در حالت سکون توان مصرفی اش خیلی کم است.
- مدار منطقی CMOS با یک منبع بین 3-18 V و معمولا 5 V تغذیه می شود.
- راه اندازی CMOS با منبع تغذیه بالاتر موجب کاهش زمان تاخیر انتشار و تصحیح حد پارازیت می شود اما توان مصرفی را افزایش می دهد.
- در مواقعی که توان مصرفی سیستم باید پایین باشد در حالی که تعداد گیت ها در داخل مدار مجتمع به دلیل پیچیده بودن طرح زیاد است از این خانواده استفاده می شود.

مدارات ترکیبی (Combinational)

- مدارهای منطقی:
  - ترکیبی
  - ترتیبی
- مدار ترکیبی: متشکل از تعدادی گیت منطقی است که خروجی آن ها در هر لحظه از زمان مستقیما به وسیله ورودی های همان لحظه معین می شود و به ورودی های قبلی بستگی ندارد.



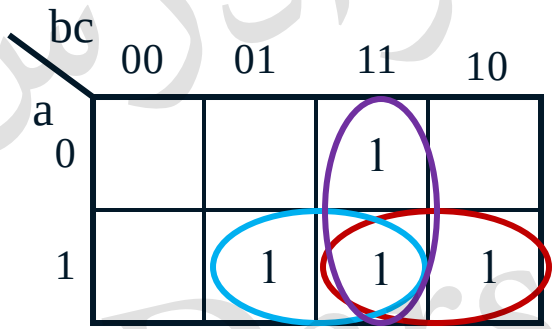
مراحل طراحی مدارات ترکیبی

- با توجه به توصیف مسئله تعداد ورودی ها و خروجی ها را مشخص کنید.
- رسم جدول درستی
- بدست آوردن تابع خروجی و ساده سازی آن (جدول کارنو، کوئین مک کلاسکی و ...)
- رسم دیاگرام منطقی

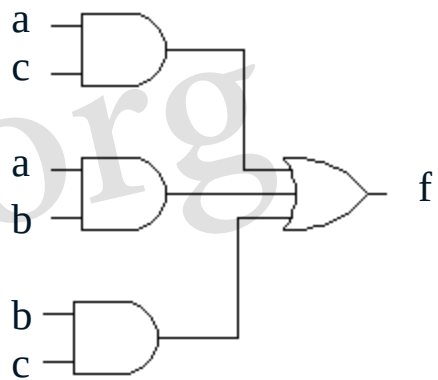
طراحی مدارات ترکیبی

- مثال1: مداری طراحی کنید که براساس اکثریت آرای ۳ نفر یک رای صادر کند.

abc	f
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

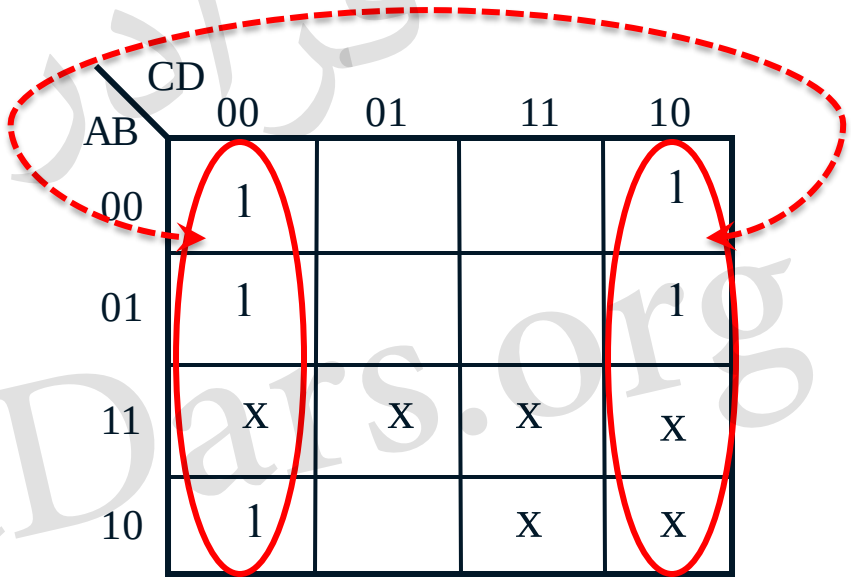


$f = a.c + a.b + b.c$

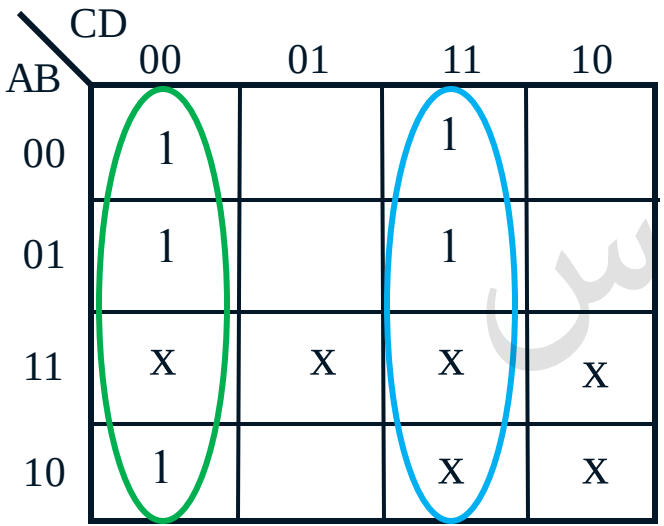


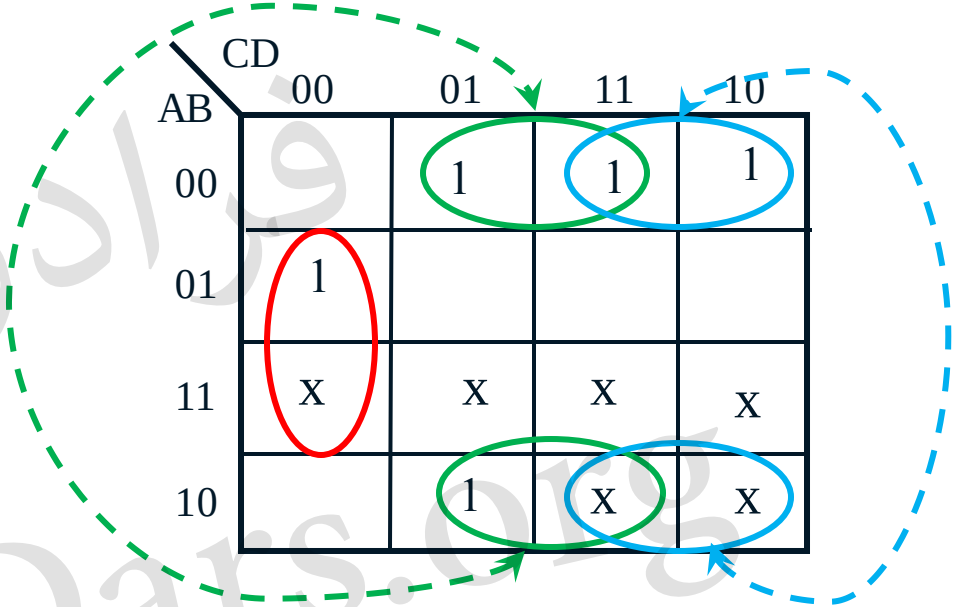
- مثال2: مدار مبدلی طراحی کنید که کد BCD را به کد افزونی ۳ تبدیل کند.

ورودی: کد BCD	خروجی: کد افزونی 3
A B C D	W X Y Z
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



$$Z = \overline{D}$$





$$Y = C.D + \overline{C}.\overline{D}$$

$$Y = C.D + \overline{(C+D)}$$

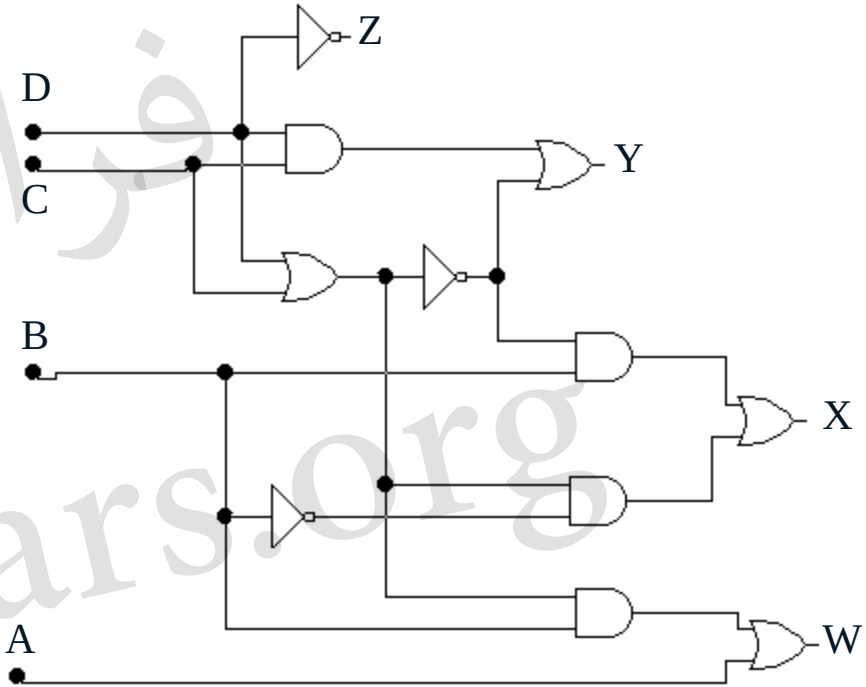
$$X = \overline{B}.C + \overline{B}.D + B\overline{C}\overline{D}$$

$$X = \overline{B}.(C + D) + B.\overline{(C+D)}$$

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

$W = A + BD + B.C$

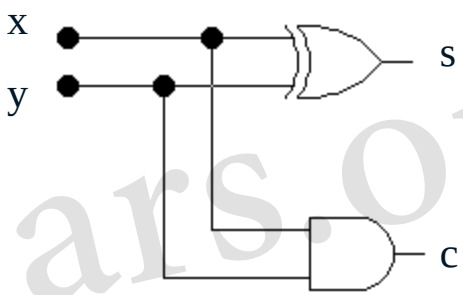
$W = A + B.(C + D)$



طراحی نیم جمع کننده (Half Adder)

- دو بیت را با هم جمع می کند، حاصل جمع و رقم نقلی تولید می کند.

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



طراحی تمام جمع کننده (Full Adder)

- سه بیت را با هم جمع می کند، حاصل جمع و رقم نقلی تولید می کند.

a b c	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

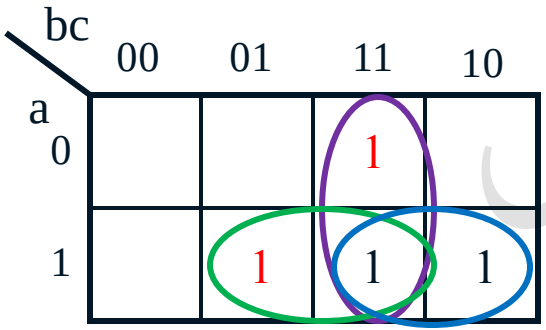
$$s(a, b, c) = \sum m(1,2,4,7)$$

$$c(a, b, c) = \sum m(3,5,6,7)$$

a \ bc	00	01	11	10
0	1			1
1	1		1	

$$s(a, b, c) = a \oplus b \oplus c$$

طراحی تمام جمع کننده (Full Adder)



$c(a, b, c) = \sum m(3,5,6,7)$

$c(a, b, c) = a.b + a.c + b.c$

$c(a, b, c) = a.b + a.b'.c + a'.b.c$

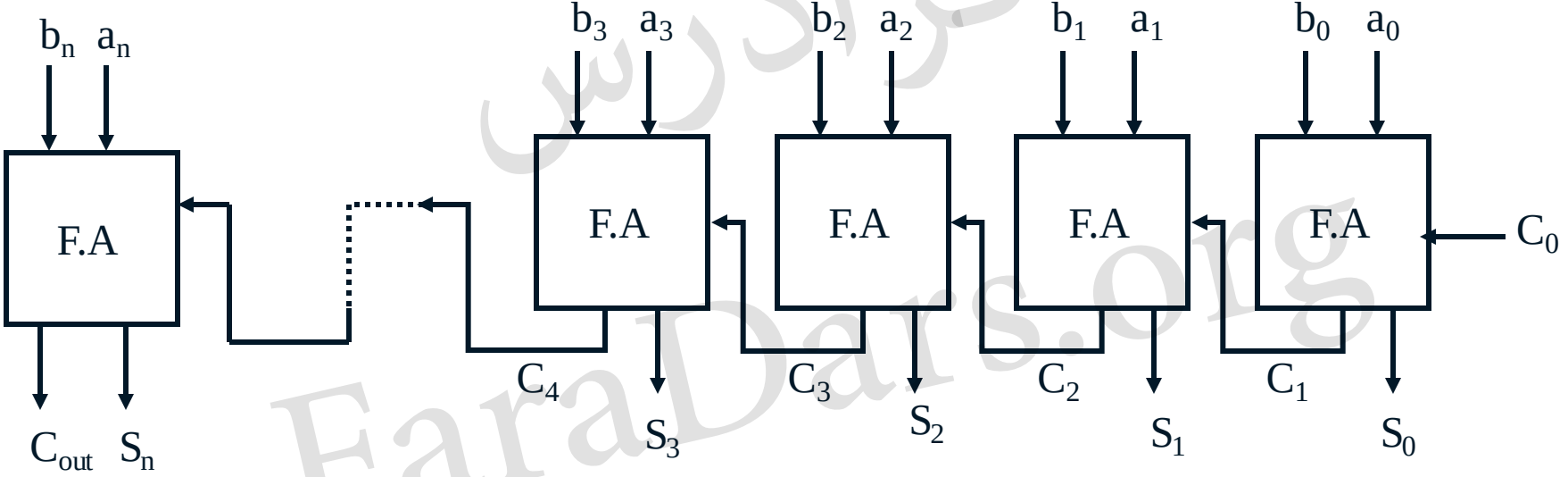
$c.(a.b' + a'.b)$

جمع کننده موج گونه (Ripple Carry Adder) یا موازی (Parallel) یا شبه موازی n بیت

- یک جمع کننده n بیتی، از به هم پیوستن متوالی n جمع کننده کامل ساخته می شود، که در آن هر خروجی نقلی از هر جمع کننده کامل به ورودی نقلی جمع کننده کامل بعدی زنجیر وار بسته می شود.

$$
\begin{array}{rccccccc}
 & c_{out} & c_{n-1} & c_{n-2} & \dots & c_1 & c_{in} & \\
 & & a_{n-1} & a_{n-2} & \dots & a_1 & a_0 & \\
+ & & b_{n-1} & b_{n-2} & \dots & b_1 & b_0 & \\
\hline
 & & s_{n-1} & s_{n-2} & \dots & s_1 & s_0 &
\end{array}
$$

جمع کننده موج گونه (Ripple Carry Adder) یا موازی (Parallel) یا شبه موازی n بیت



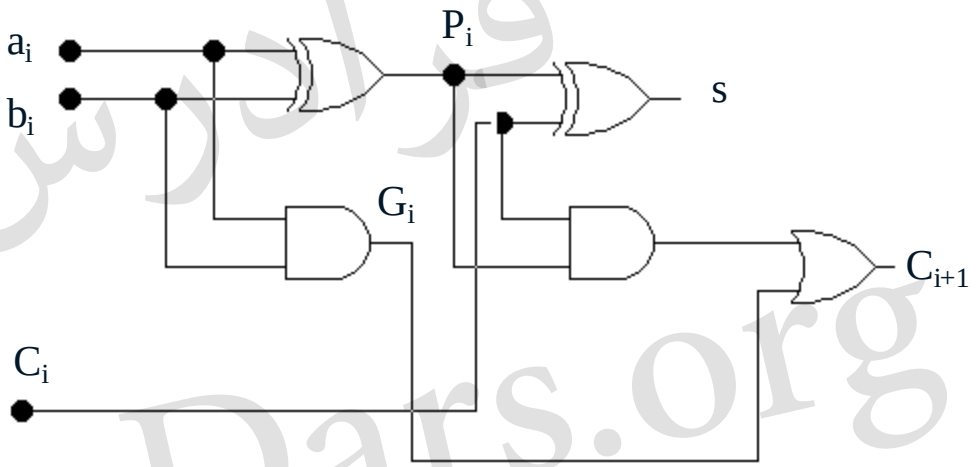
پیش بینی رقم نقلی

$P_i = a_i \oplus b_i$

$G_i = a_i \cdot b_i$

$s_i = P_i \oplus c_i$

$c_{i+1} = G_i + P_i \cdot c_i$



c_0 = نقلی ورودی

$c_1 = G_0 + P_0 \cdot c_0$

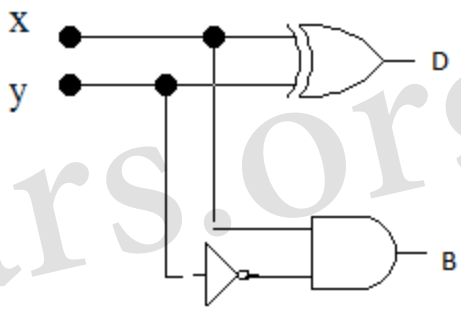
$c_2 = G_1 + P_1 \cdot c_1 = G_1 + P_1 \cdot (G_0 + P_0 \cdot c_0) = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0$

$c_3 = G_2 + P_2 \cdot c_2 = G_2 + P_2 \cdot (G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot c_0) = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot c_0$

طراحی نیم تفریق کننده

- در مدارهای تفریق کننده به جای رقم نقلی رقم قرضی وجود دارد.

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



جمع و تفریق کننده n بیت

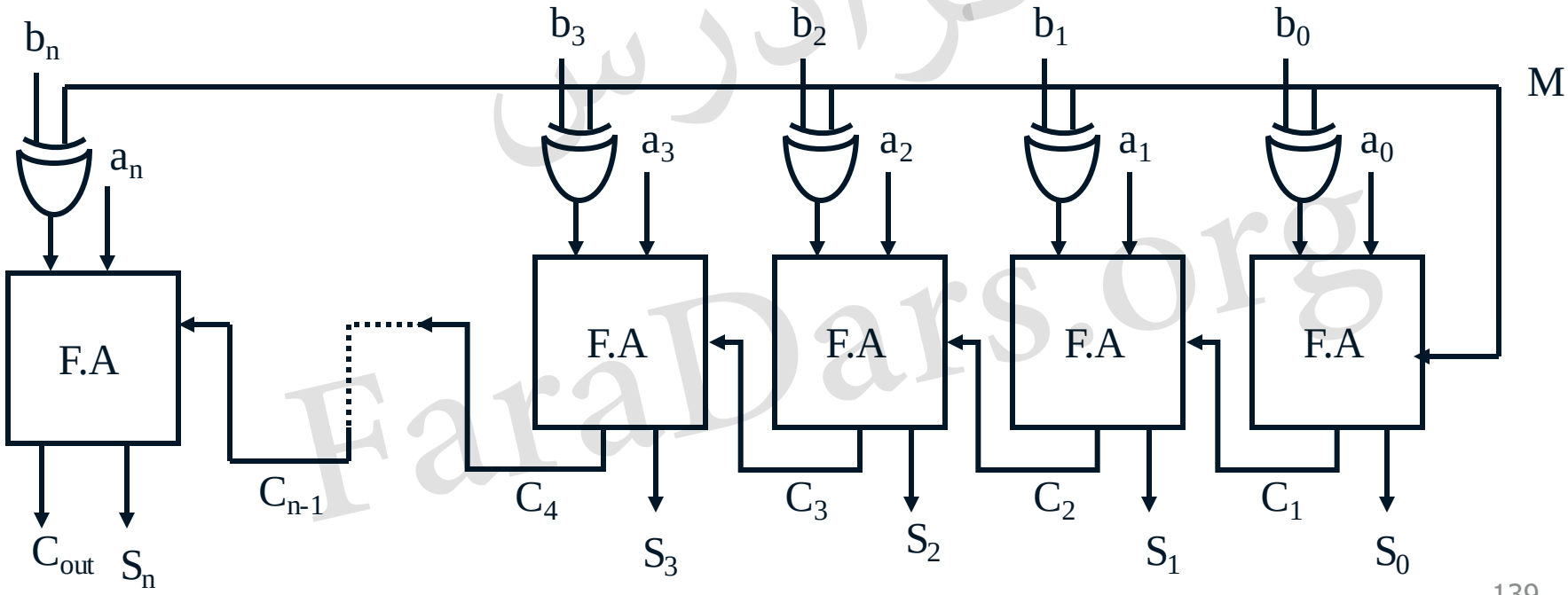
$$a - b = a + \overline{b} + 1$$

b a	F= b⊕ a
0 0	0
0 1	1
1 0	1
1 1	0

$$b \oplus 0 = b$$
$$b \oplus 1 = \overline{b}$$

جمع و تفریق کننده n بیت

$$a - b = a + \overline{b} + 1$$



مقایسه گر مقدار

- با مقایسه دو عدد، بزرگتر، کوچکتر یا مساوی بودن آنها تعیین می شود.
- مقایسه گر مقدار، مداری ترکیبی است که دو عدد را به عنوان ورودی دریافت و خروجی آن سه متغیر دودویی می باشد که بیانگر بزرگتر، کوچکتر و مساوی بودن عدد اول بنسبت به عدد دوم است.

طراحی مقایسه گر مقدار ۴ بیت

A_0 B_0	G E L
0 0	0 1 0
0 1	0 0 1
1 0	1 0 0
1 1	0 1 0

$$(A_0 = B_0) = A_0 B_0 + \overline{A_0}\,\overline{B_0}$$

$$(A > B) = A_0.\overline{B_0}$$

$$(A < B) = \overline{A_0}.B_0$$

$$A = A_3 A_2 A_1 A_0$$

$$B = B_3 B_2 B_1 B_0$$

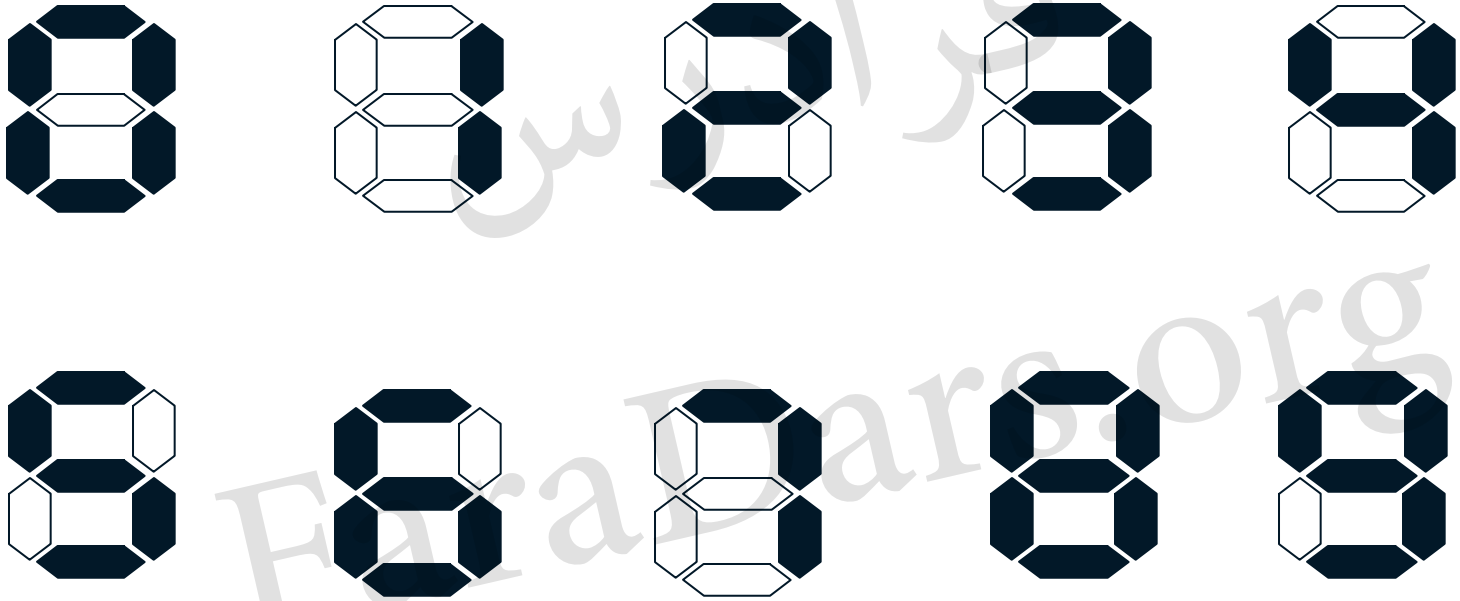
$$X_i = A_i B_i + \overline{A_i}\,\overline{B_i}$$

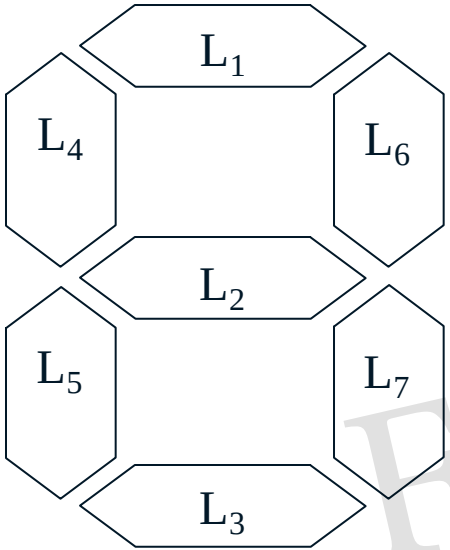
$$(A = B) = X_3 . X_2 . X_1 . X_0$$

$$(A > B) = A_3.\overline{B_3} + X_3.A_2.\overline{B_2} + X_3.X_2.A_1.\overline{B_1} + X_3.X_2.X_1.A_0.\overline{B_0}$$

$$(A < B) = \overline{A_3}.B_3 + X_3.\overline{A_2}.B_2 + X_3.X_2.\overline{A_1}.B_1 + X_3.X_2.X_1.\overline{A_0}.B_0$$

Seven Segment Display طراحی

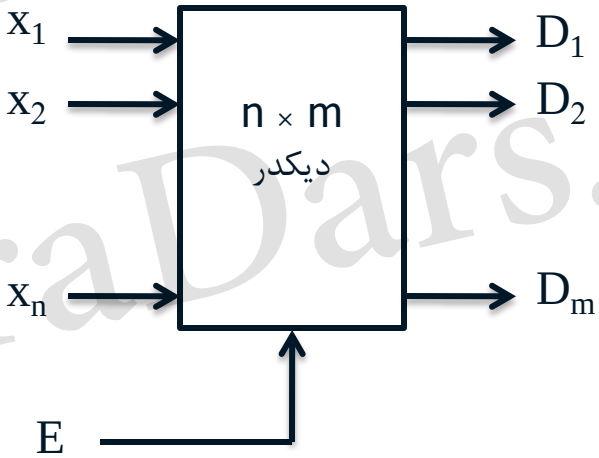




Val	B_3	B_2	B_1	B_0	L_1	L_2	L_3	L_4	L_5	L_6	L_7
0	0	0	0	0	1	0	1	1	1	1	1
1	0	0	0	1	0	0	0	0	0	1	1
2	0	0	1	0	1	1	1	0	1	1	0
3	0	0	1	1	1	1	1	0	0	1	1
4	0	1	0	0	0	1	0	1	0	1	1
5	0	1	0	1	1	1	1	1	0	0	1
6	0	1	1	0	1	1	1	1	1	0	1
7	0	1	1	1	1	0	0	0	0	1	1
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

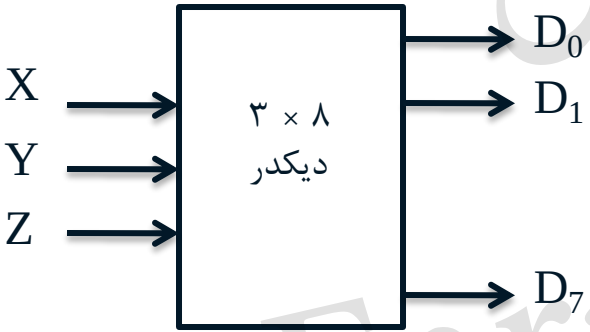
رمزگشا (دیکدر Decoder)

- مداری ترکیبی با n ورودی و حداکثر 2^n خروجی است.
- دیکدر تمامی مینترم ها را که با n متغیر دودویی می توان ساخت ایجاد می کند.
- دیکدر معمولا دارای یک یا دو ورودی فعال ساز برای کنترل کار مدار است.



دیکدر ۳ به ۸

- کاربرد رایج دیکدر ۳ به ۸ : تبدیل دودویی به هشت هشتی



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

دیکدر ۲ به۴ با یک ورودی فعال ساز

- بعضی از دیکدر ها با گیت های NAND ساخته می شوند، چون تولید مینترم ها در شکل متمم اقتصادی تر است.

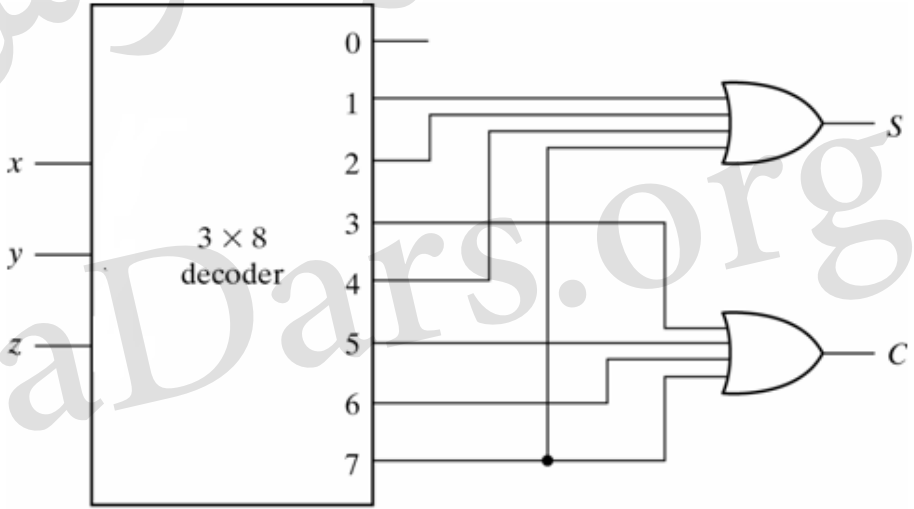
E	A_0	B_0	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

پیاده سازی مدار منطقی ترکیبی با دیکدر

- هر مدار ترکیبی n ورودی و m خروجی با یک دیکدر n به 2^n و m گیت OR قابل پیاده سازی است.
- مثال: مدار جمع کننده کامل را با استفاده از دیکدر پیاده سازی کنید.

$$s(x, y, z) = \sum m(1,2,4,7)$$

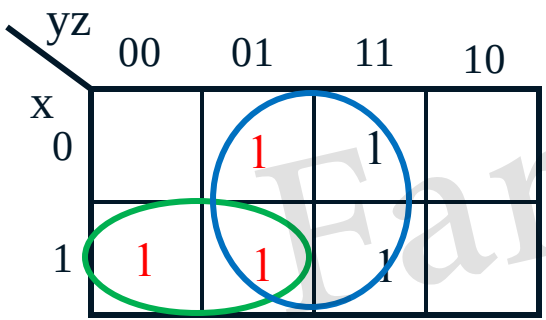
$$c(x, y, z) = \sum m(3,5,6,7)$$



پیاده سازی مدار منطقی ترکیبی با دیکدر

- مثال: مدار زیر را با استفاده از دیکدر پیاده سازی کنید.

$$F(x, y, z) = \sum m(1,3,4,5,7)$$



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

رمزگذار (Encoder)

- انکدر دارای 2^n یا کمتر خط ورودی و n خط خروجی است.
- انکدر مداری است که عمل عکس دیکدر را انجام می دهد.
- خطوط خروجی کد دودویی مربوط به مقدار دودویی ورودی را نشان می دهد.
- فرض می کنیم در هر لحظه از زمان تنها یک ورودی مقدار یک داشته باشد.
- اگر بیش از یک ورودی در هر لحظه از زمان فعال باشد از انکدر اولویت استفاده می کنیم.

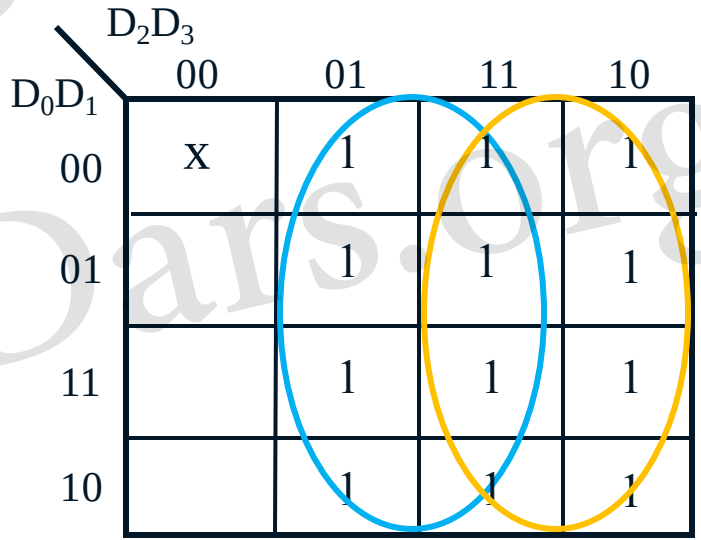
رمزگذار (Encoder)

ورودی								خروجی		
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

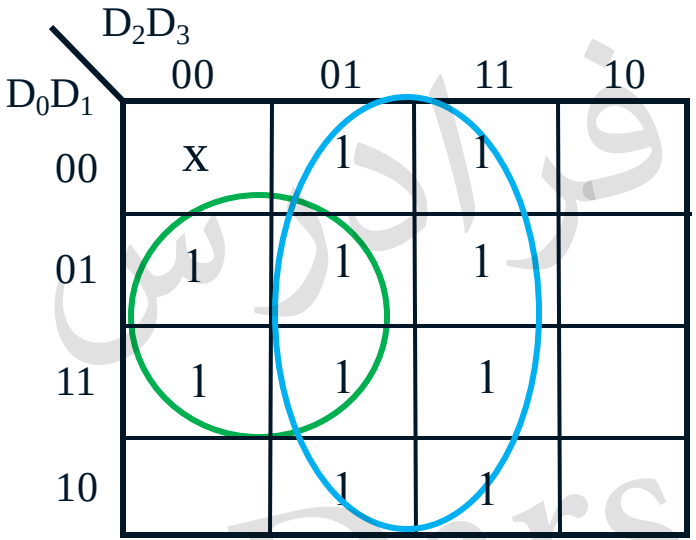
انکدر اولویت

- اینکدر اولویت اجازه می دهد تا چندین خط ورودی فعال شوند، اگر دو یا چند ورودی به طور همزمان برابر ۱ شوند، ورودی با اولویت بالاتر پیش خواهد افتاد.
- خروجی V مشخص می کند آیا هیچ یک از ورودی ها برابر ۱ هستند یا خیر (آیا خروجی معتبر است یا خیر)

ورودی				خروجی		
D_0	D_1	D_2	D_3	X	Y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1



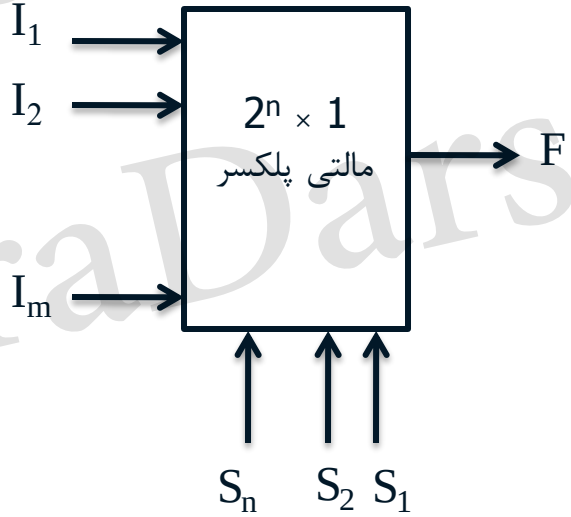
$X = D_2 + D_3$

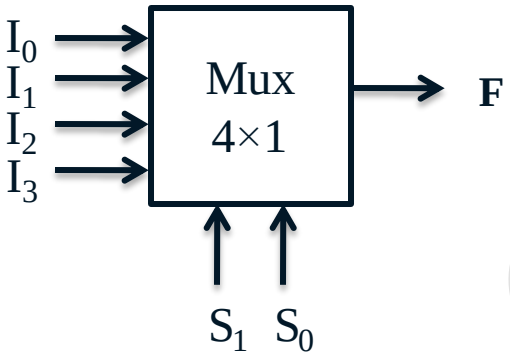


$$Y = D_3 + D_1 . \overline{D_2}$$

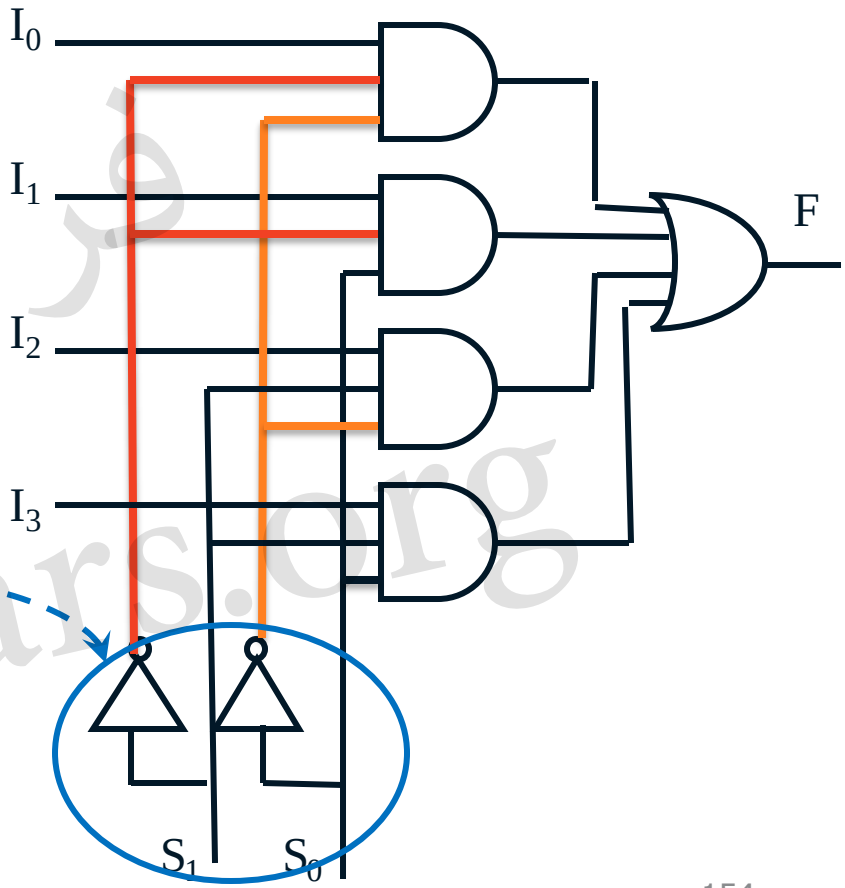
مالتی پلکسر (تسهیم کننده) (Data Selector)

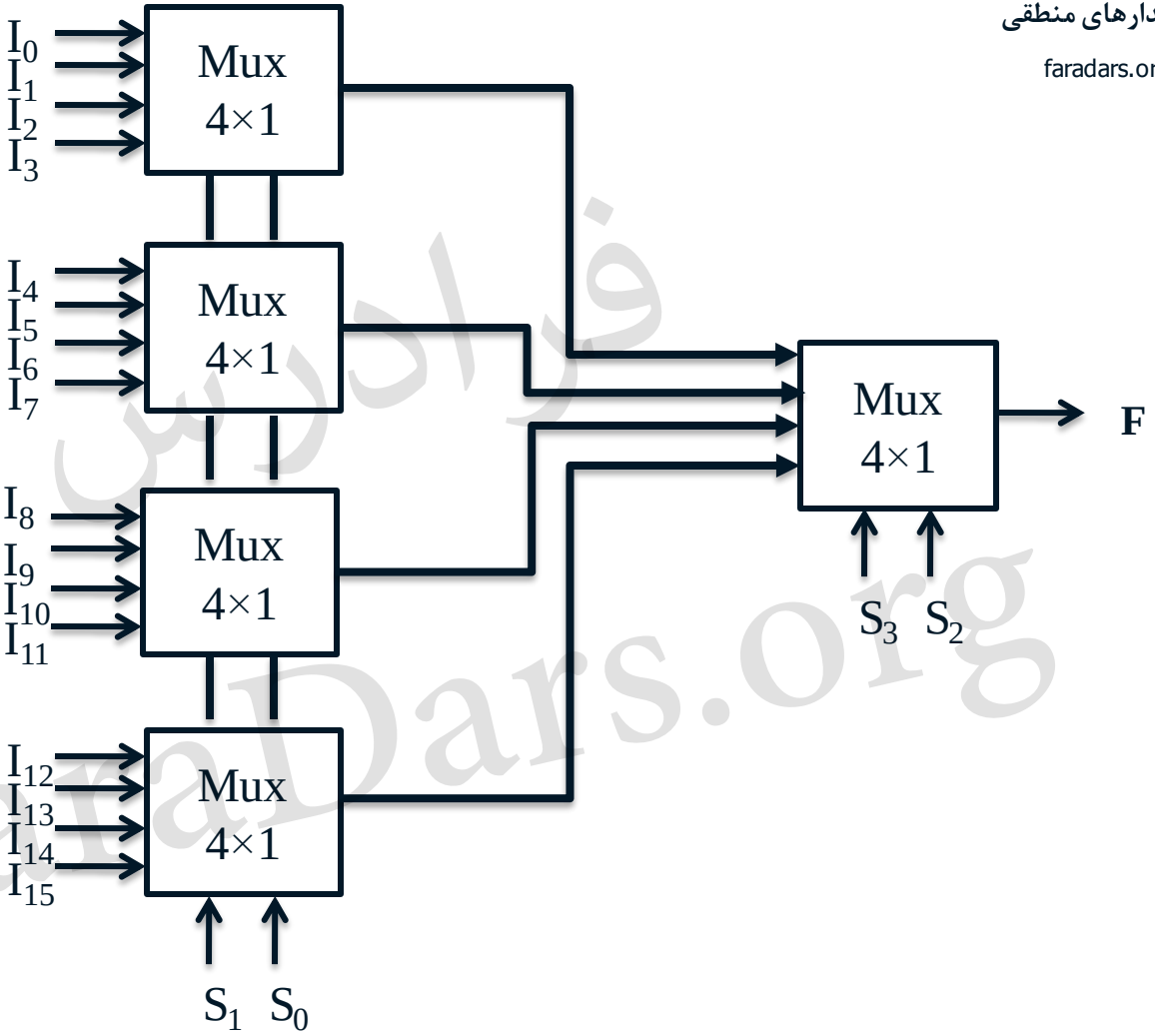
- یک مدار ترکیبی است که یکی از ورودی هایش را به تنها خروجی، منتقل می کند.
- انتخاب یک ورودی خاص به وسیله مجموعه خطوط انتخاب انجام می شود.
- معمولا 2^n خط ورودی و n خط انتخاب وجود دارد.





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

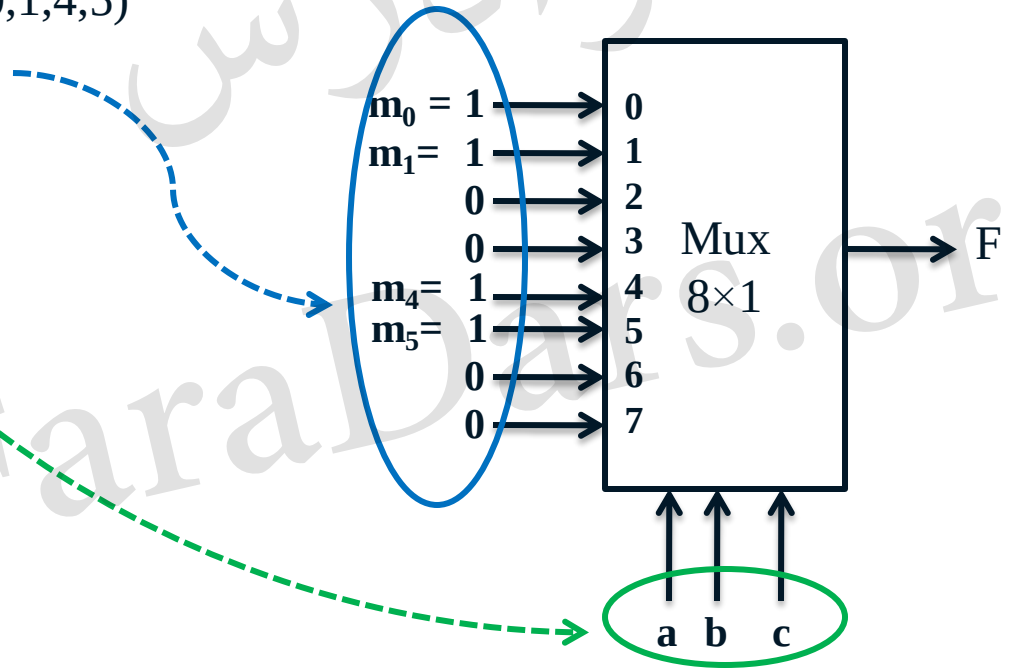




پیاده سازی توابع بول با مالتی پلکسر

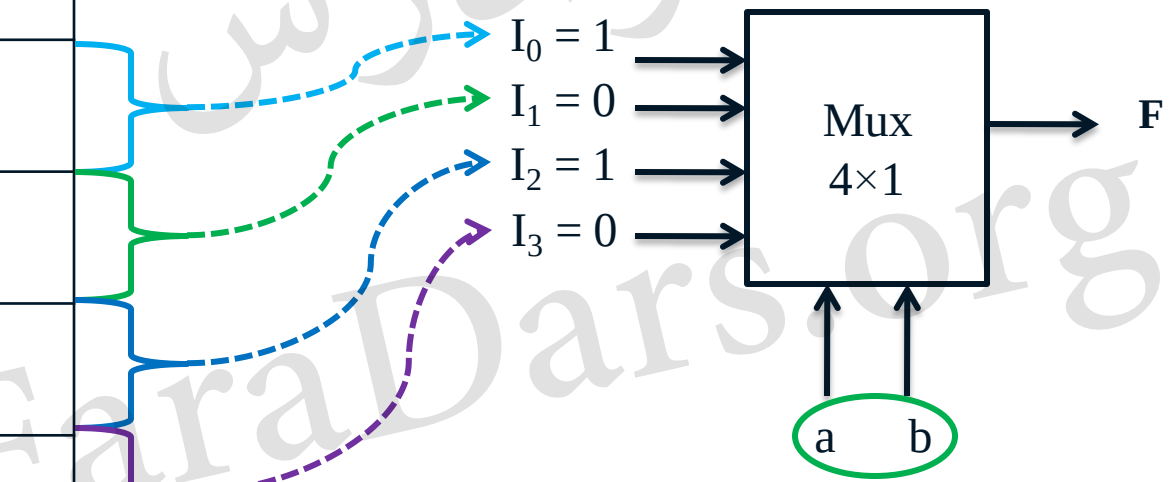
- مثال: تابع F را با مالتی پلکسر 8×1 ، مالتی پلکسر 4×1 و مالتی پلکسر 2×1 پیاده سازی کنید.

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



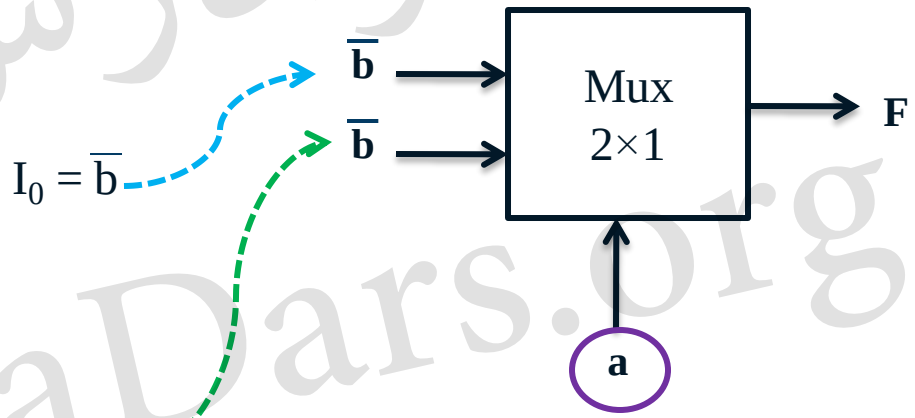
$F(a, b, c) = \sum m(0,1,4,5)$

a b c	F
0 0 0	1
0 0 1	1
0 1 0	0
0 1 1	0
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	0



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

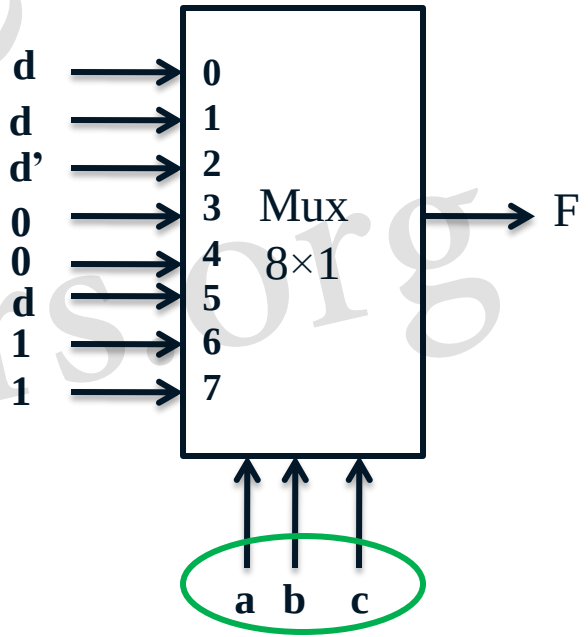
a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



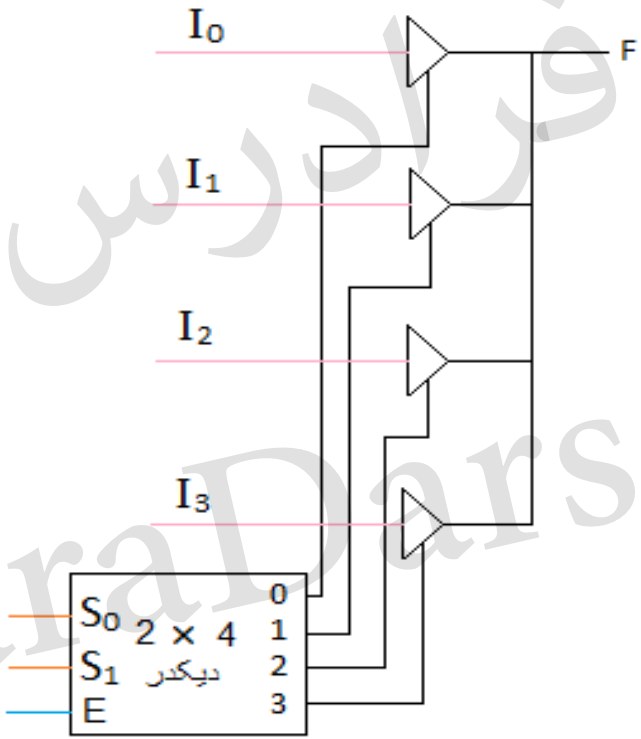
a b c d	F
0 0 0 0	0
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	0
0 1 1 0	0
0 1 1 1	0
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	1
1 1 0 0	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	1

- مثال: تابع F را با مالتی پلکسر 8×1 پیاده سازی کنید.

$$F(a, b, c, d) = \sum m(1, 3, 4, 11, 12, 13, 14, 15)$$

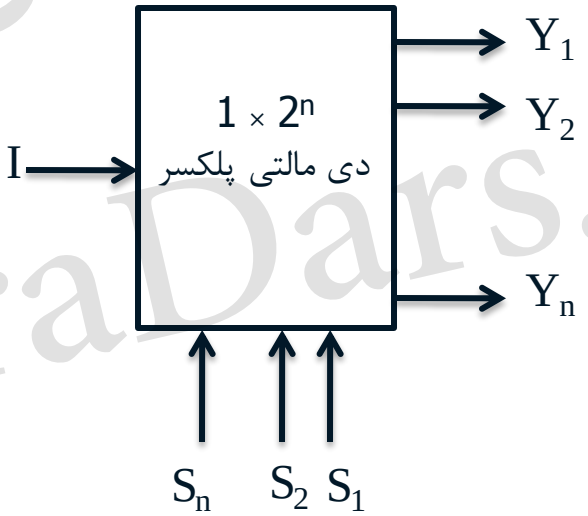


طراحی مالتی پلکسر با گیت های سه حالته



دی مالتی پلکسر (Demux)

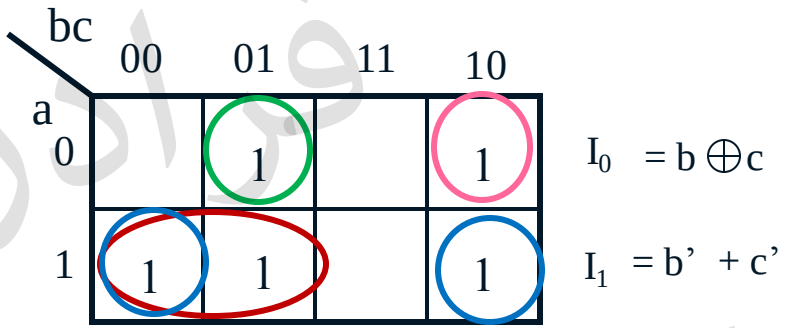
- یک مدار ترکیبی است که یک ورودی دارد و چندین خروجی، با توجه به مقدار انتخاب، ورودی را به یکی از خروجی ها منتقل می کند و سایر خروجی ها صفر می شوند.
- دی مالتی پلکسر عکس مالتی پلکسر است.

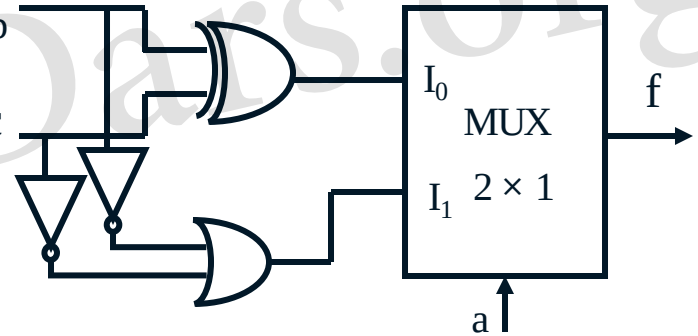


- مثال: تابع F را با مالتی پلکسر 2×1 پیاده سازی کنید.

$F(a, b, c) = \sum m(1, 2, 4, 5, 6)$

	abc	f
I_0	000	0
	001	1
	010	1
	011	0
I_1	100	1
	101	1
	110	1
	111	0

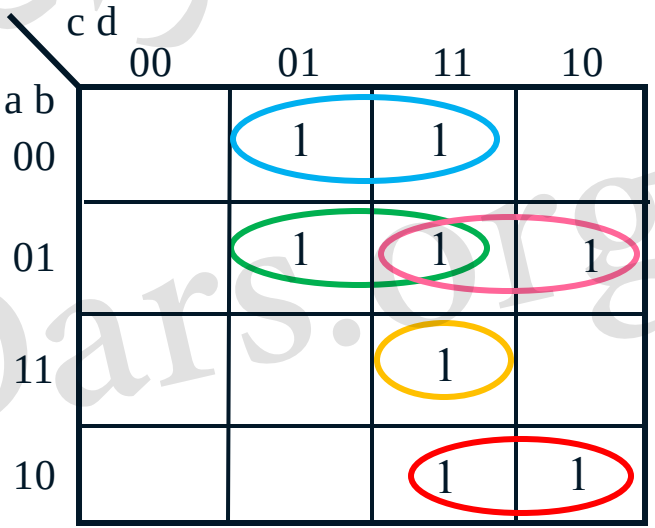


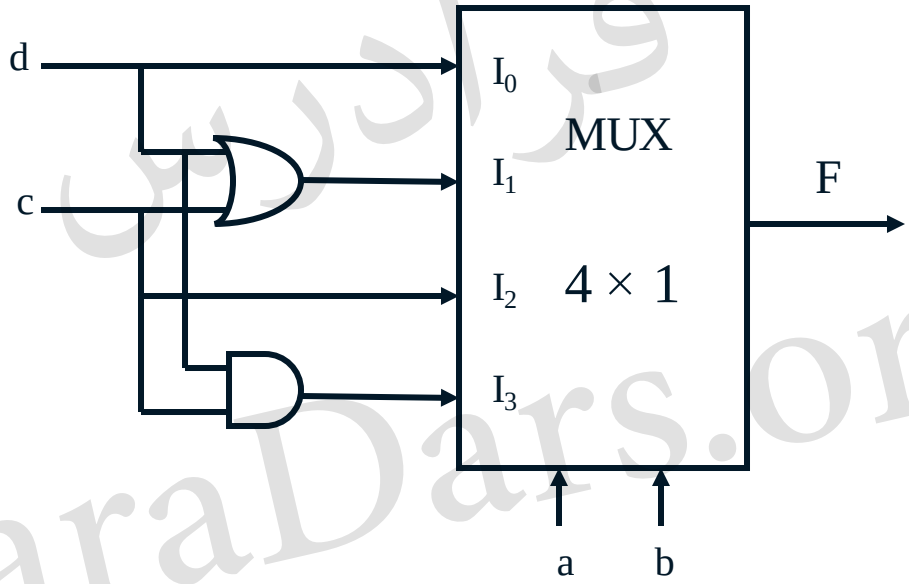


- مثال: تابع F را با مالتی پلکسر 4×1 پیاده سازی کنید.

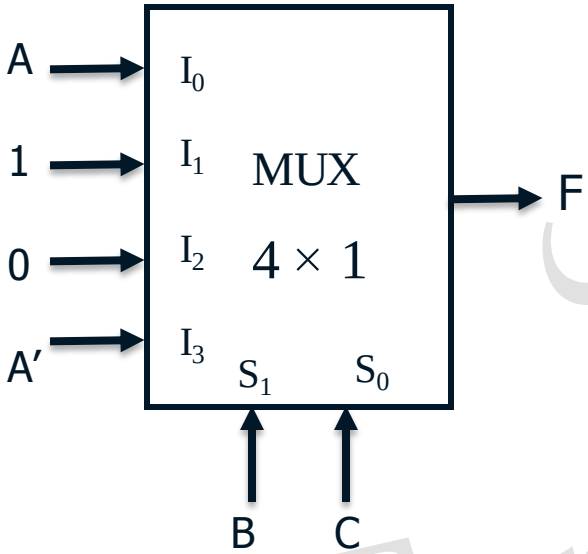
$$F(a, b, c, d) = \sum m(1, 3, 5, 6, 7, 10, 11, 15)$$

	a b c d	F
I_0	0 0 0 0	0
	0 0 0 1	1
	0 0 1 0	0
	0 0 1 1	1
I_1	0 1 0 0	0
	0 1 0 1	1
	0 1 1 0	1
	0 1 1 1	1
I_2	1 0 0 0	0
	1 0 0 1	0
	1 0 1 0	1
	1 0 1 1	1
I_3	1 1 0 0	0
	1 1 0 1	0
	1 1 1 0	0
	1 1 1 1	1





- مثال: با توجه به مالتی پلکسر تابع F (A,B,C) را مشخص کنید.



A BC	F
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1
1 0 1	1
1 1 0	0
1 1 1	0

$$F(A,B,C) = \sum(1, 3, 4, 5)$$

$$F(A,B,C) = \prod(0, 2, 6, 7)$$

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

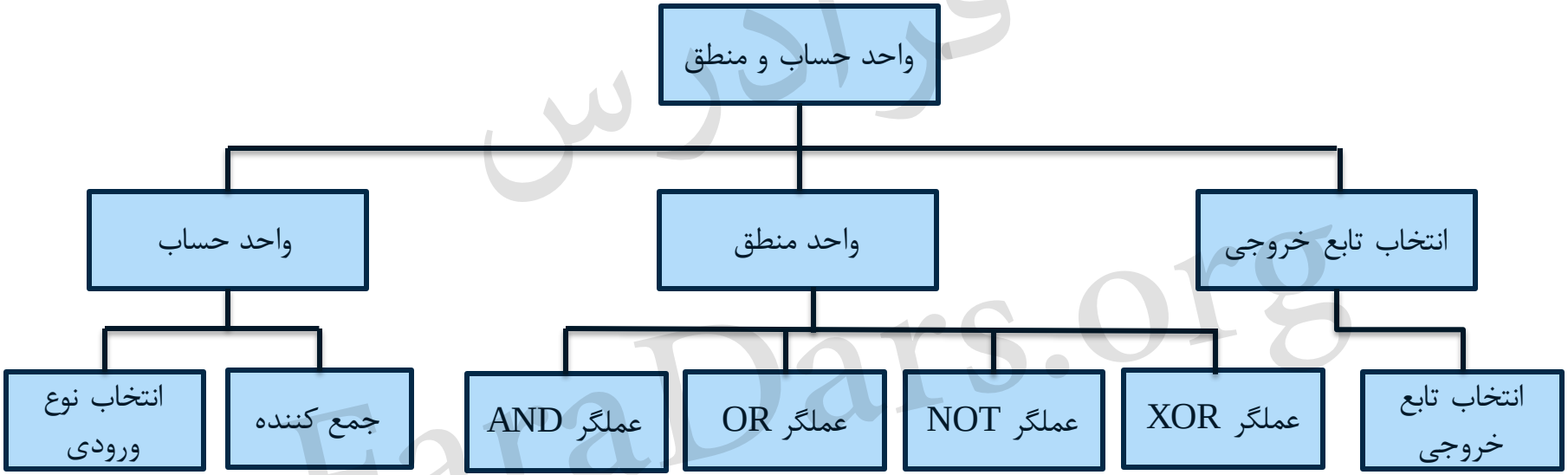
100 101 , 001 011

$$F(A,B,C) = \sum(4, 5, 1, 3)$$

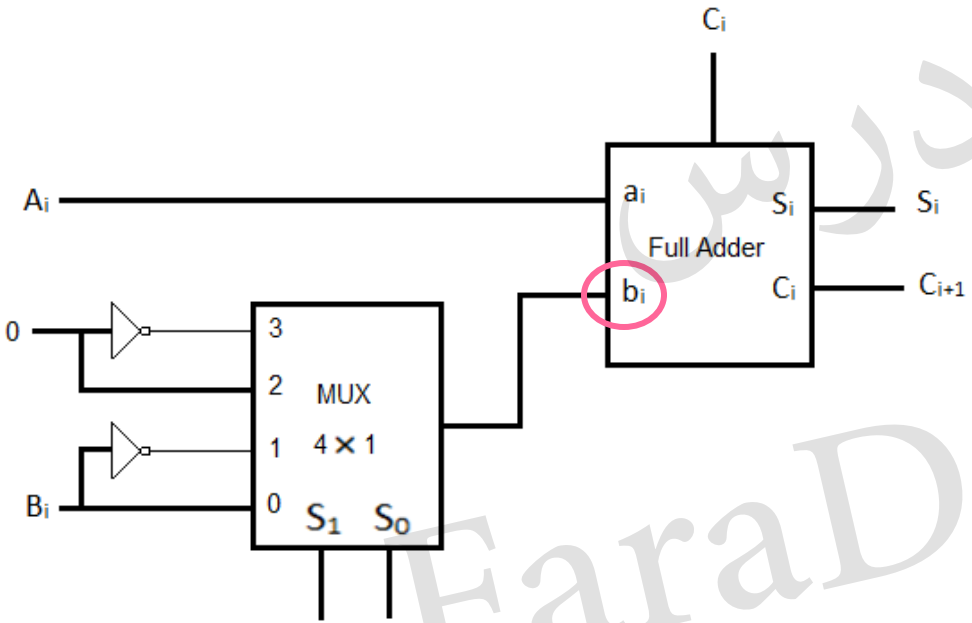
واحد حساب و منطق (Arithmetic Logic Unit)

- واحد حساب و منطق (ALU) در کامپیوترهای دیجیتال وظیفه انجام اعمال محاسباتی (جمع، تفریق و...) و منطقی (AND، OR و ...) را بر روی داده ها به عهده دارد.

طراحی درخت گونه واحد حساب و منطق

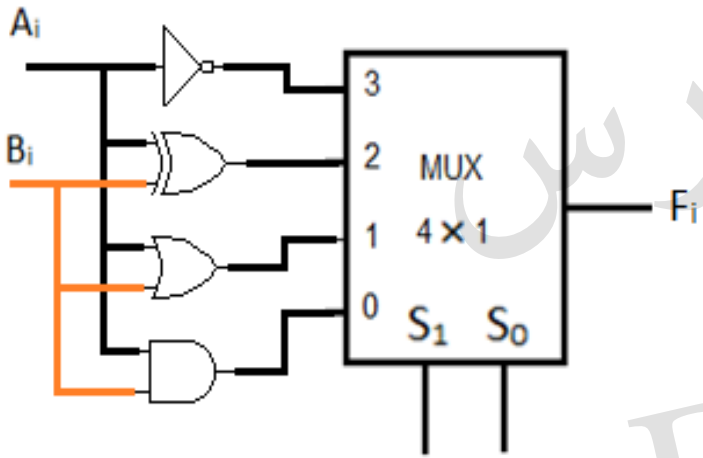


طراحی واحد حساب یک بیتی



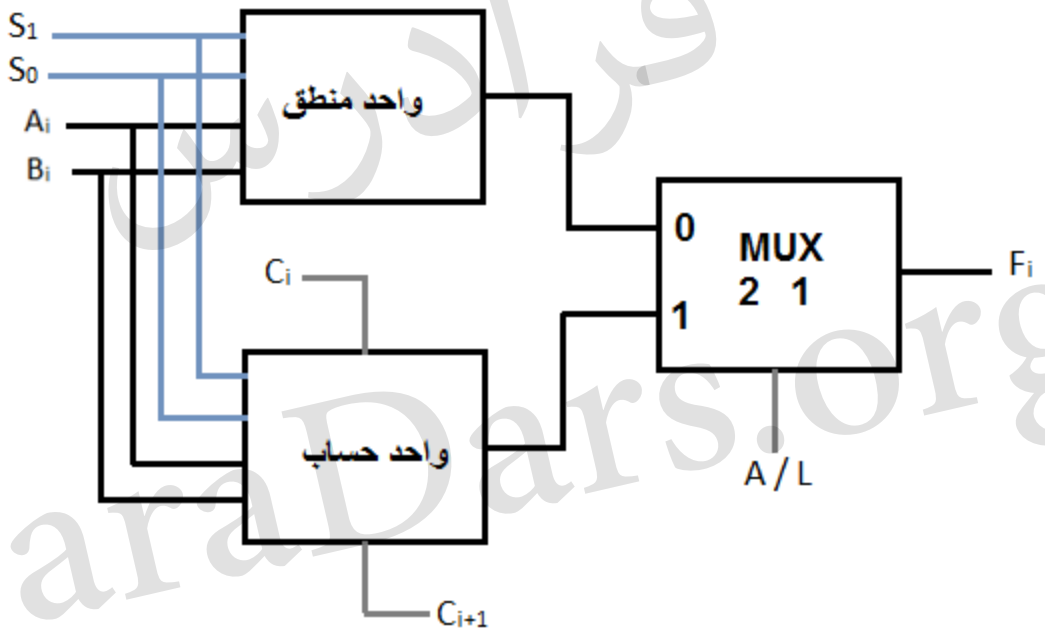
نام تابع	خروجی مدار	bi	Cin	S0	S1
جمع	A + B	B	0	0	0
جمع با رقم نقلی	A + B +1	B	1	0	0
تفریق با رقم قرضی	A + B’	B’	0	1	0
تفریق	A + B’ +1	B’	1	1	0
انتقال	A	0	0	0	1
یک واحد افزایش	A+1	0	1	0	1
یک واحد کاهش	A -1	1	0	1	1
انتقال	A	1	1	1	1

طراحی واحد منطق یک بیتی



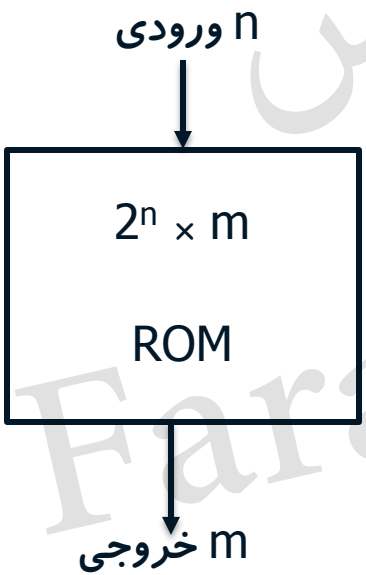
S1	S0	خروجی مدار	نام تابع
0	0	A . B	AND
0	1	A + B	OR
1	0	$A \oplus B$	XOR
1	1	A’	NOT

واحد حساب و منطق یک بیتی



حافظه فقط خواندنی (ROM)

- ROM : مداری که شامل گیت های OR همراه با دیکدر در یک IC است.

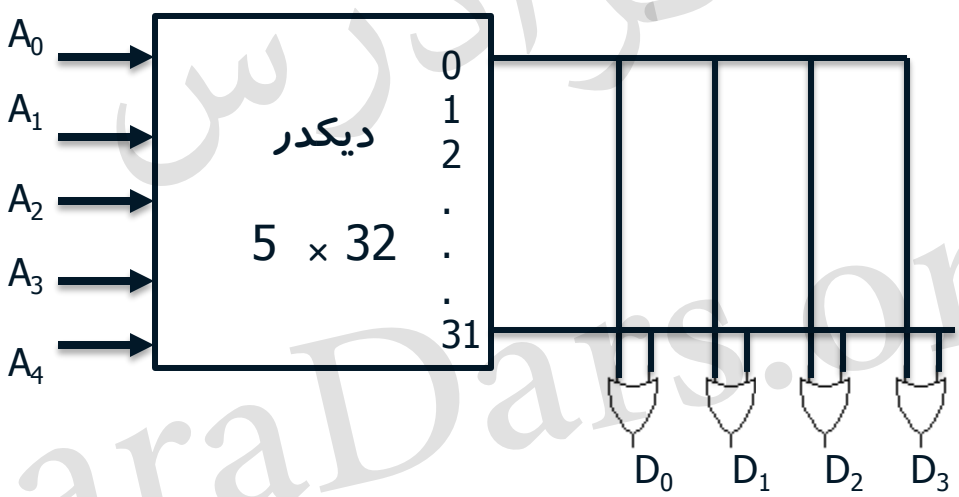


- آدرس هر ترکیب از متغیرهای ورودی است.
- n متغیر 2^n آدرس ایجاد می کند.

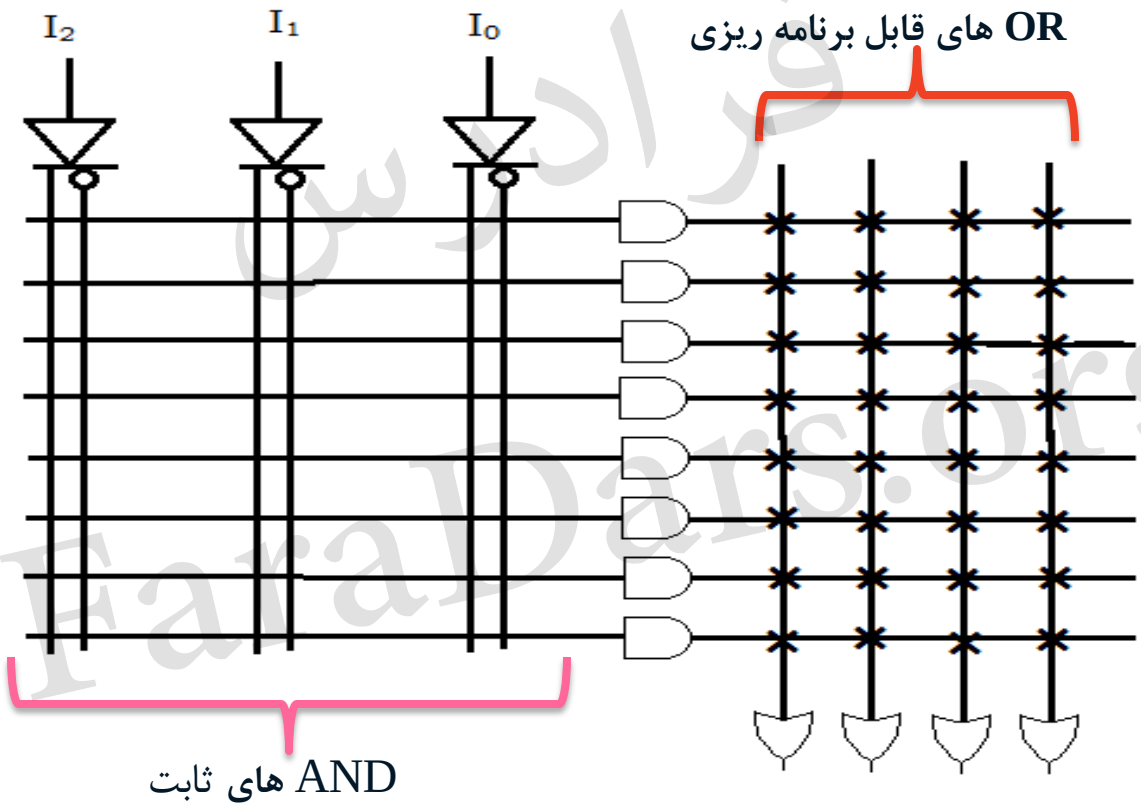
- کلمه هر ترکیب خروجی است.
- n متغیر 2^n کلمه ایجاد می کند.

ساختمان ROM

- نمایش ساختار یک ROM 32 × 4



حافظه فقط خواندنی قابل برنامه ریزی (PROM)

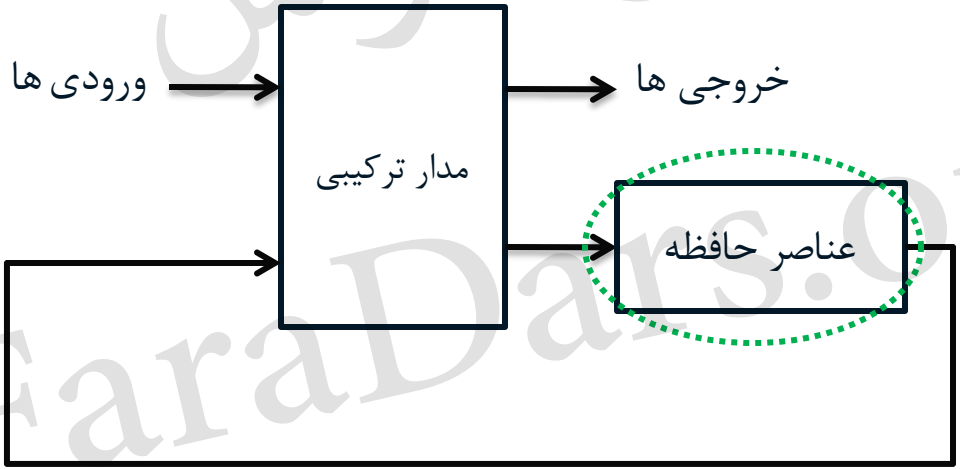


مدارات ترتیبی

- مدارهای سنکرون
- مدارهای پالسی
- مدارهای آسنکرون

مدارات ترتیبی

- خروجی مدارات ترتیبی علاوه بر ورودی های فعلی به ورودی های گذشته نیز وابسته است.

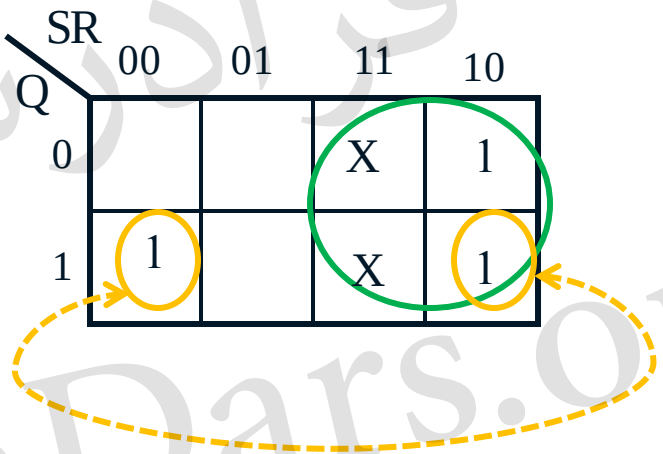


لچ ها

- عناصر ذخیره سازی در مدارات ترتیبی ساعت دار را فلیپ فلاپ می گویند.
- لچ ها (یا نگهدارها) مدارهای مبنایی هستند که همه فلیپ فلاپ ها با آنها ساخته می شود.

لچ SR

حالت فعلی		حالت بعدی	
S R	Q	Q^*	
0 0	0	0	بدون تغییر
0 0	1	1	
0 1	0	0	Reset
0 1	1	0	
1 0	0	1	Set
1 0	1	1	
1 1	0	0	غیر مجاز
1 1	1	0	

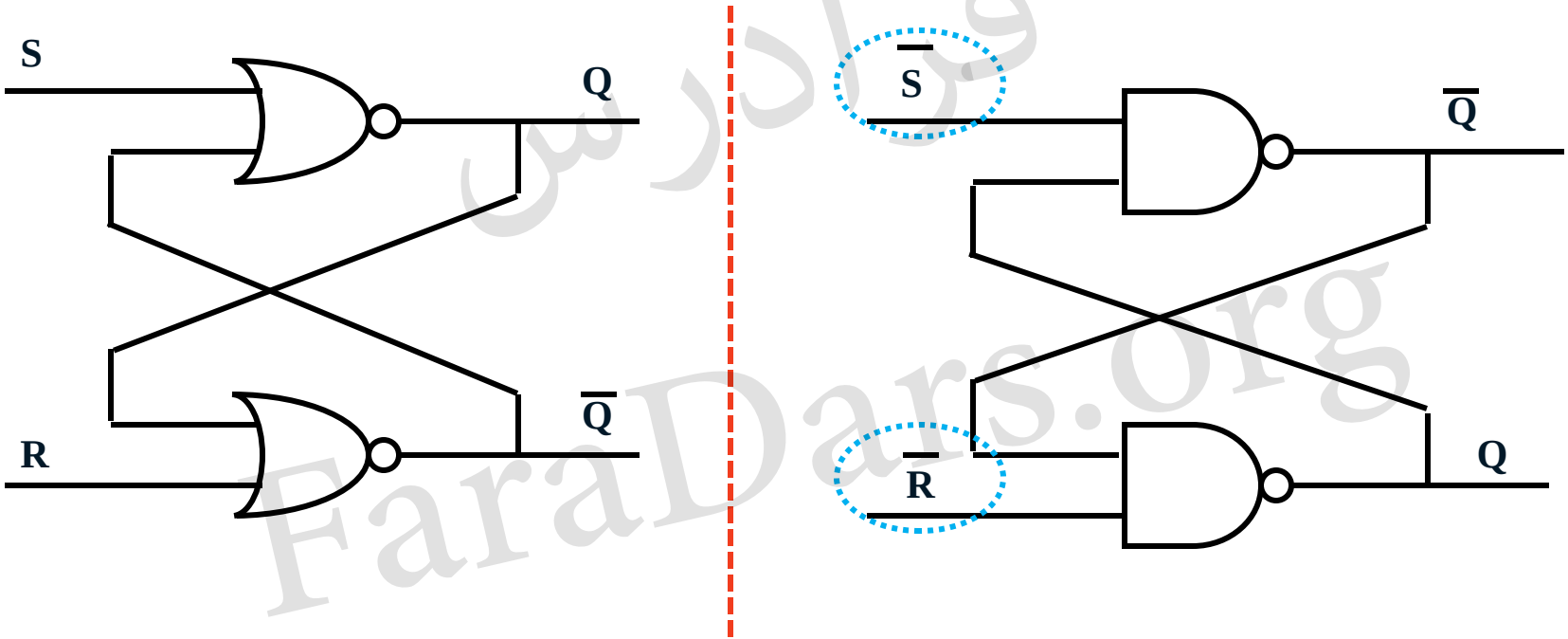


S	R	Q^*
0	0	Q
0	1	0
1	0	1
1	1	X

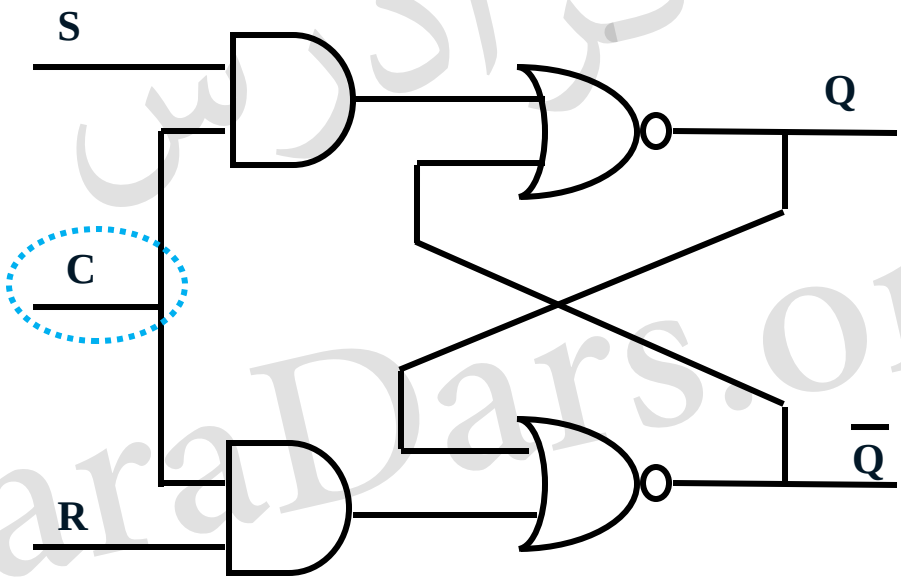
$$Q^* = S + \overline{R}.Q$$

$$S.R = 0$$

لچ SR

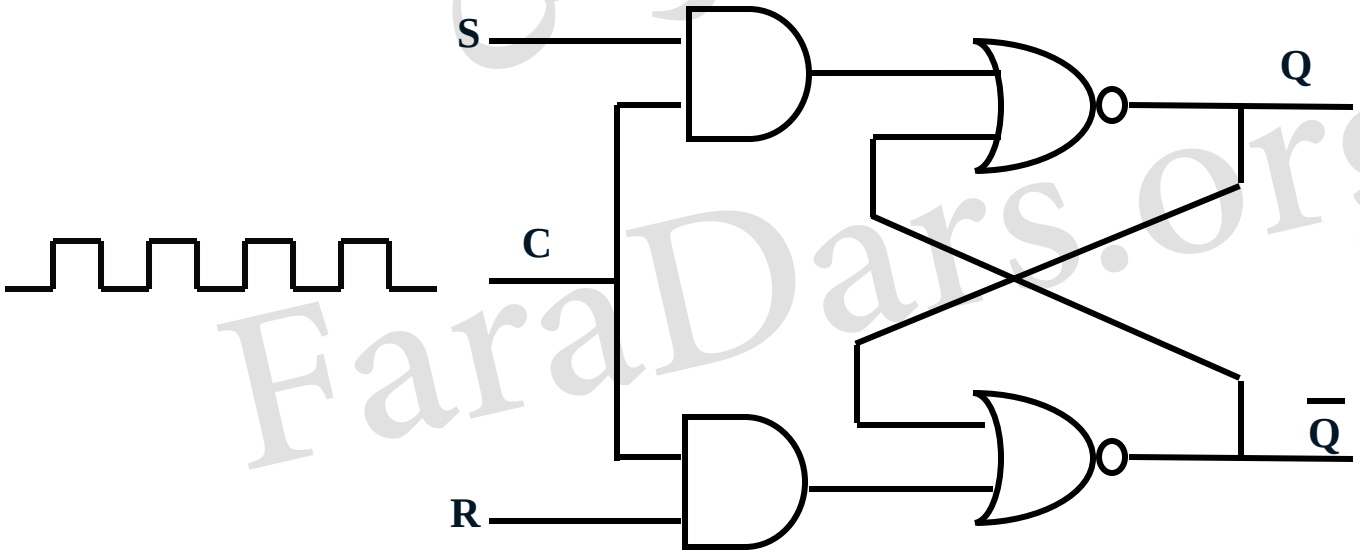


لچ SR با ورودی کنترل



فلیپ فلاپ

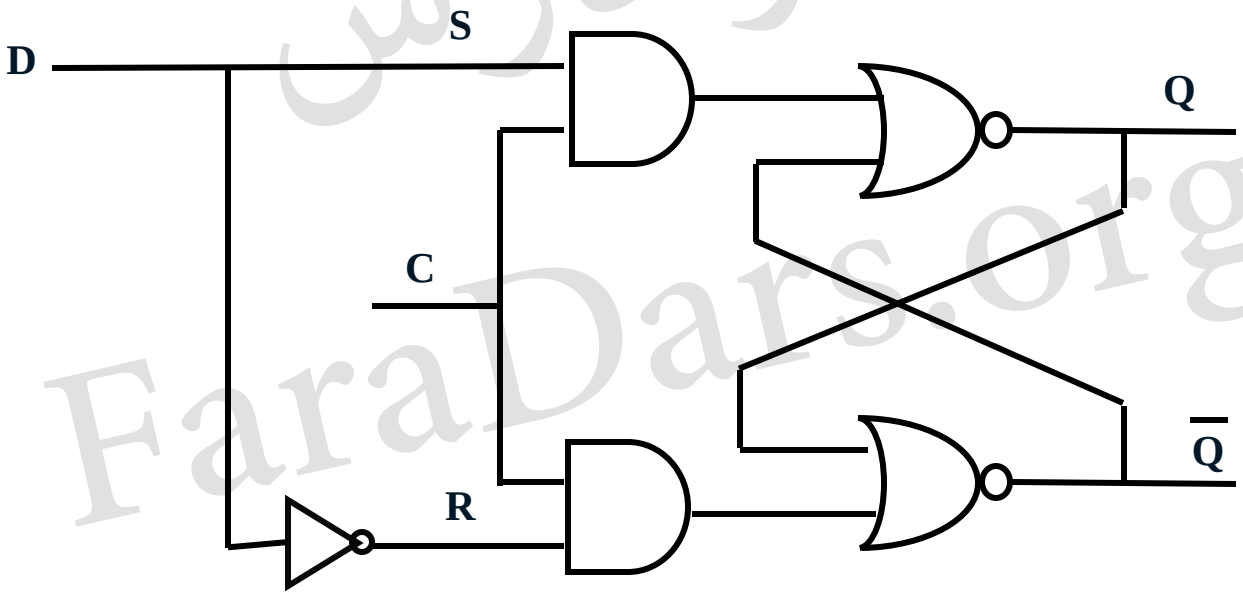
- اگر به ورودی کنترل لچ، کلاک (سیگنالی که متناوبا صفر و یک می شود) متصل کنیم به آن فلیپ فلاپ گفته می شود.
- فلیپ فلاپ، حساس به لبه کلاک است و لچ حساس به سطح کلاک است.



فلیپ فلاپ D

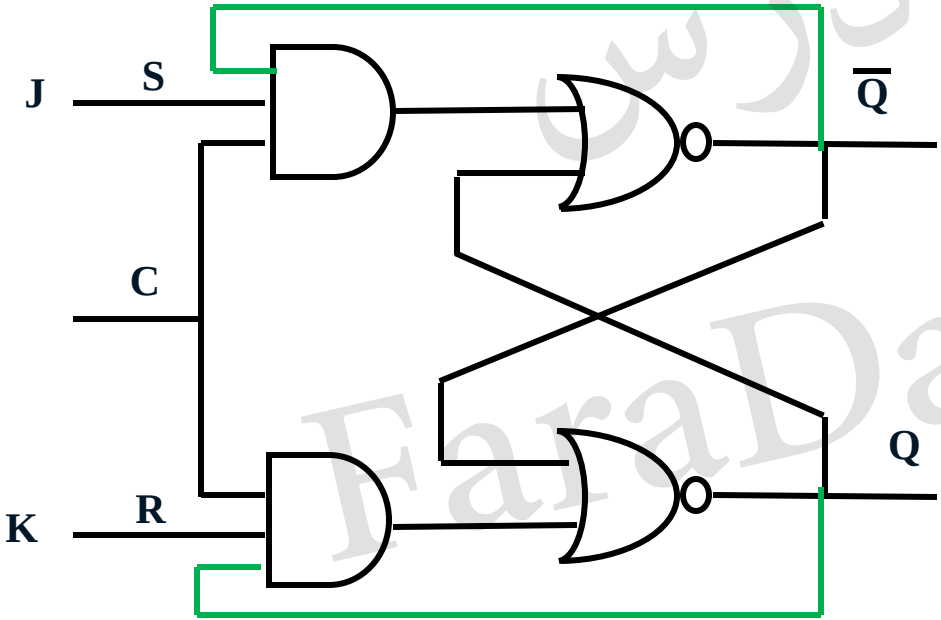
- از اتصال ورودی D به S و مکمل آن به R، فلیپ فلاپ D ساخته می شود.

$$Q^* = D$$



فلیپ فلاپ JK

- اگر در فلیپ فلاپ SR از Q فیدبکی به گیتی که ورودی R دارد و از Q به گیتی که ورودی S دارد بدهیم، فلیپ فلاپ JK ساخته می شود.



J	K	Q*	
0	0	Q	(حفظ حالت)
0	1	0	(Reset)
1	0	1	(Set)
1	1	$\overline{Q}$	(مکمل)

$$Q^* = J.\overline{Q} + \overline{K}.Q$$

فلیپ فلاپ T (Trigger)

- در فلیپ فلاپ JK اگر ورودی های J و K را به هم متصل کنیم، فلیپ فلاپ T بدست می آید.
- در فلیپ فلاپ T اگر T=0 حالت حفظ می شود ($Q^*= Q$)، و اگر T=1 حالت عوض می شود ($Q^*= \overline{Q}$).

- معادله مشخصه فلیپ فلاپ T:

$$Q^* = T \oplus Q$$

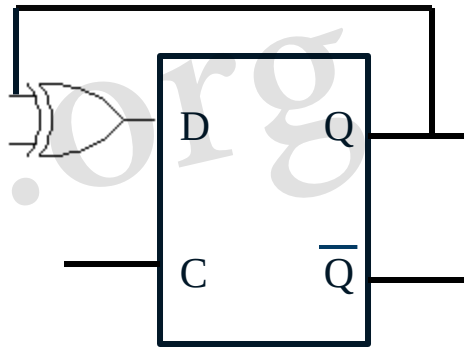
ساخت فلیپ فلاپ ها با کمک سایر فلیپ فلاپ ها

- مثال: با استفاده از فلیپ فلاپ D ، فلیپ فلاپ T بسازید.

معادله فلیپ فلاپ D : $Q^* = D$

معادله فلیپ فلاپ T : $Q^* = T \oplus Q$

$D = T \oplus Q$

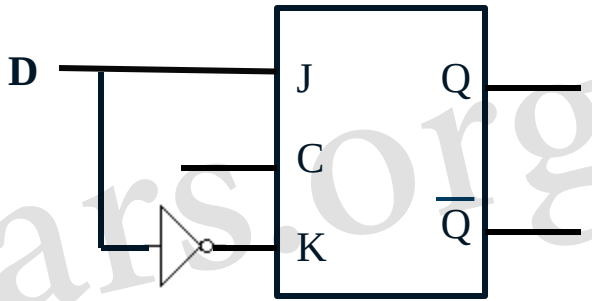


ساخت فلیپ فلاپ ها با کمک سایر فلیپ فلاپ ها

- مثال: با استفاده از فلیپ فلاپ JK ، فلیپ فلاپ D بسازید.

$D = 0 \rightarrow J = 0 , K = 1 \rightarrow Q^* = 0$

$D = 1 \rightarrow J = 1 , K = 0 \rightarrow Q^* = 1$



ساخت فلیپ فلاپ ها با کمک سایر فلیپ فلاپ ها

- مثال: با استفاده از فلیپ فلاپT ، فلیپ فلاپ JK بسازید.

معادله فلیپ فلاپ T :

$$Q^* = T \oplus Q$$

$$T = Q^* \oplus Q$$

معادله فلیپ فلاپ JK :

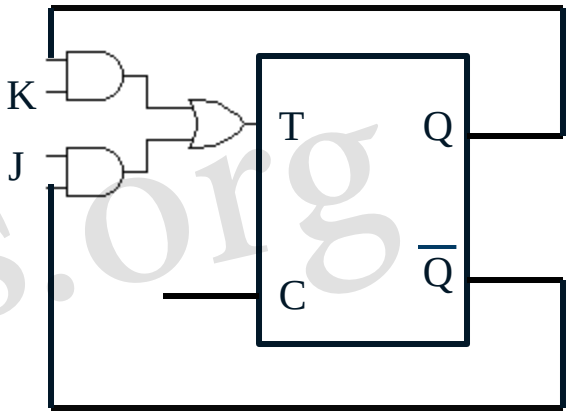
$$Q^* = J.\overline{Q} + \overline{K}.Q$$

$$T = (J.\overline{Q} + \overline{K}.Q) \oplus Q$$

$$T = \overline{(J.\overline{Q} + \overline{K}.Q)}.Q + (J.\overline{Q} + \overline{K}.Q).\overline{Q}$$

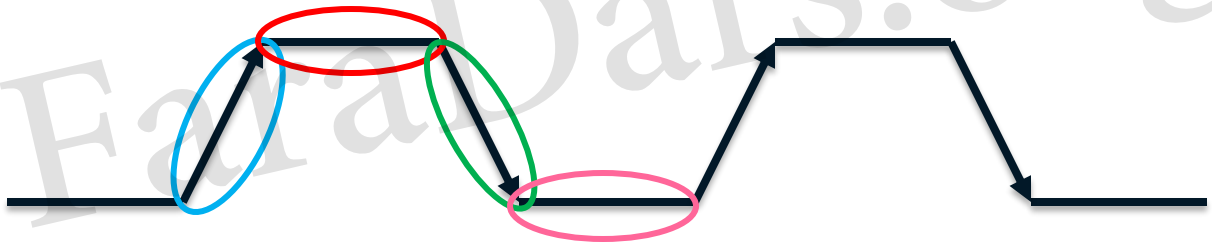
$$T = (\overline{J}+Q).(K+\overline{Q}).Q + J.\overline{Q}$$

$$T = K.Q + J.\overline{Q}$$

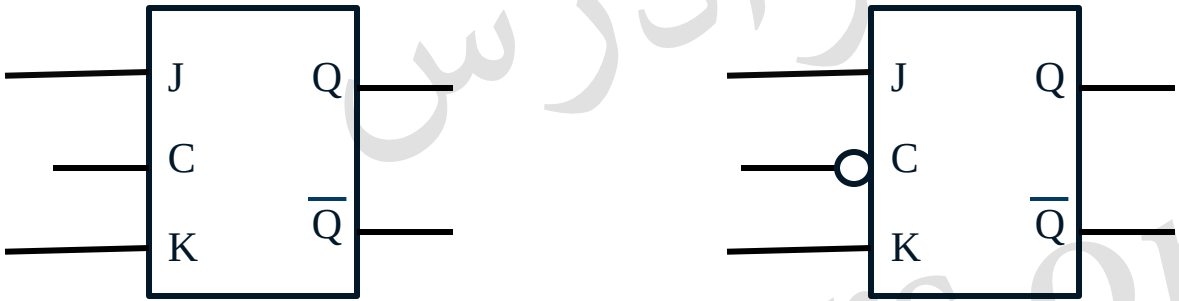


بخش های مختلف پالس ساعت (کلاک پالس)

1. لبه (Edge) مثبت کلاک: مدت زمانی که طول می کشد کلاک از صفر (منفی) به یک (مثبت) تغییر کند.
2. سطح (Level) مثبت کلاک: زمانی که کلاک مثبت است.
3. لبه منفی کلاک: مدت زمانی که طول می کشد کلاک از مثبت به منفی تغییر کند.
4. سطح منفی کلاک: زمانی که کلاک منفی است.



بلوک دیاگرام فلیپ فلاپ حساس به سطح

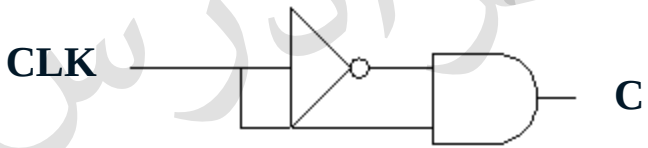


حساس به لبه کردن فلیپ فلاپ

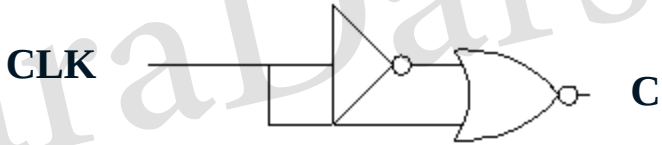
- استفاده از یک مدارایجاد کننده لبه (Edge Detector)
- استفاده از فلیپ فلاپ Master - Slave

Edge Detector

- اگر بر سر راه کلاک مدار نمودار زیر قرار گیرد آنگاه یک پالس کوچک مثبت ایجاد می شود.

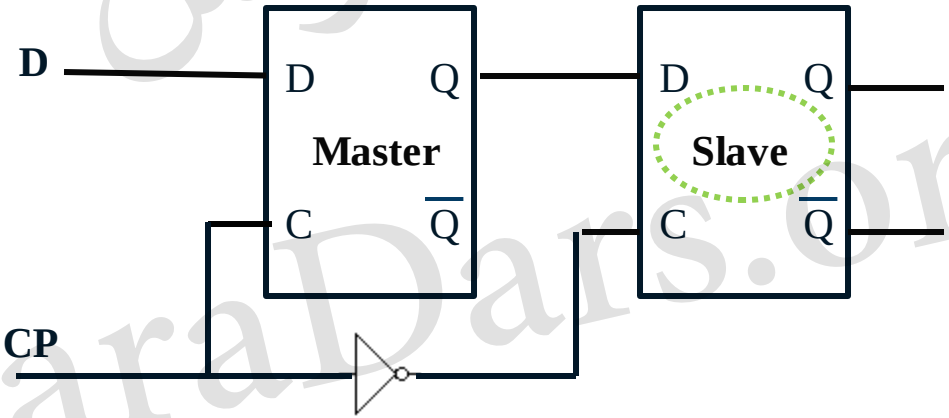


- اگر بر سر راه کلاک مدار نمودار زیر قرار گیرد آنگاه در لبه منفی کلاک یک پالس کوچک منفی ایجاد می شود.

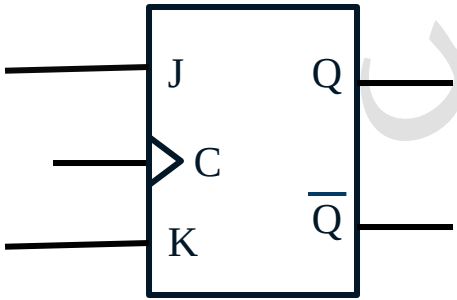


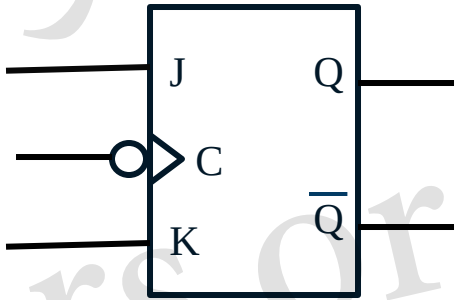
فلیپ فلاپ های تابع- حاکم (Master - Slave)

- یک فلیپ فلاپ تابع- حاکم از دو فلیپ فلاپ حساس به سطح ساخته می شود.

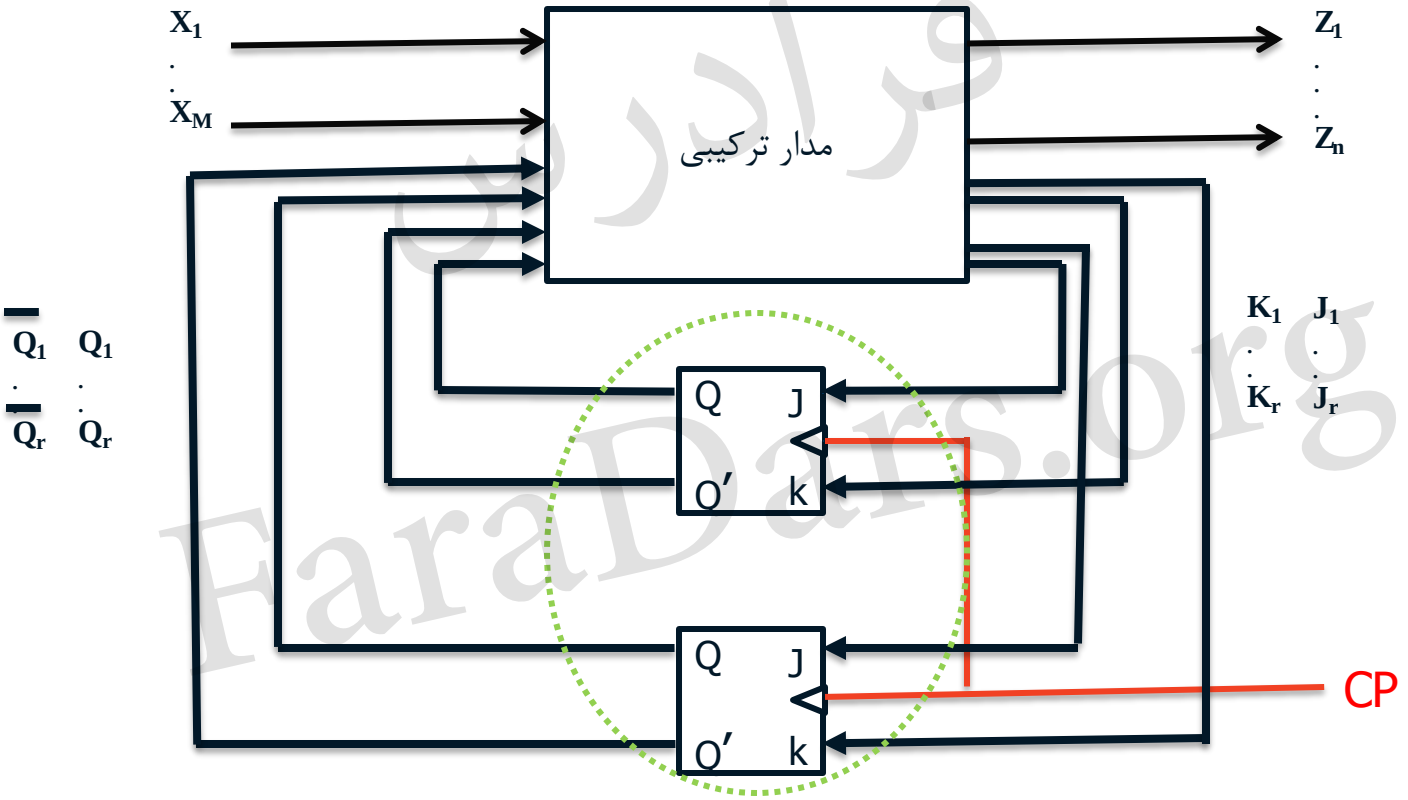


بلوک دیاگرام فلیپ فلاپ حساس به لبه





مدارهای ترتیبی سنکرون (Synchronous Sequential Circuit)



انواع مدارهای ترتیبی سنکرون

1. مدارهای مور (Moore) : اگر خروجی های مدار (z_1 .. z_n) تنها به حالت های مدار (Q_1 .. Q_n) وابسته باشد مدار ترتیبی از نوع مور است. بنابراین در مدارهای مور خروجی ها تابع ورودی های مدار نیستند و تنها به خروجی فلیپ فلاپ های مدار (حالت مدار) بستگی دارند.

2. مدارهای میلی (Mealy) : اگر در مداری، خروجی علاوه بر خروجی های فلیپ فلاپ ها (حالت مدار) به ورودی های مدار نیز وابسته باشد مدار ترتیبی از نوع میلی خواهد بود.

برخی نکات در مدارهای میلی و مور

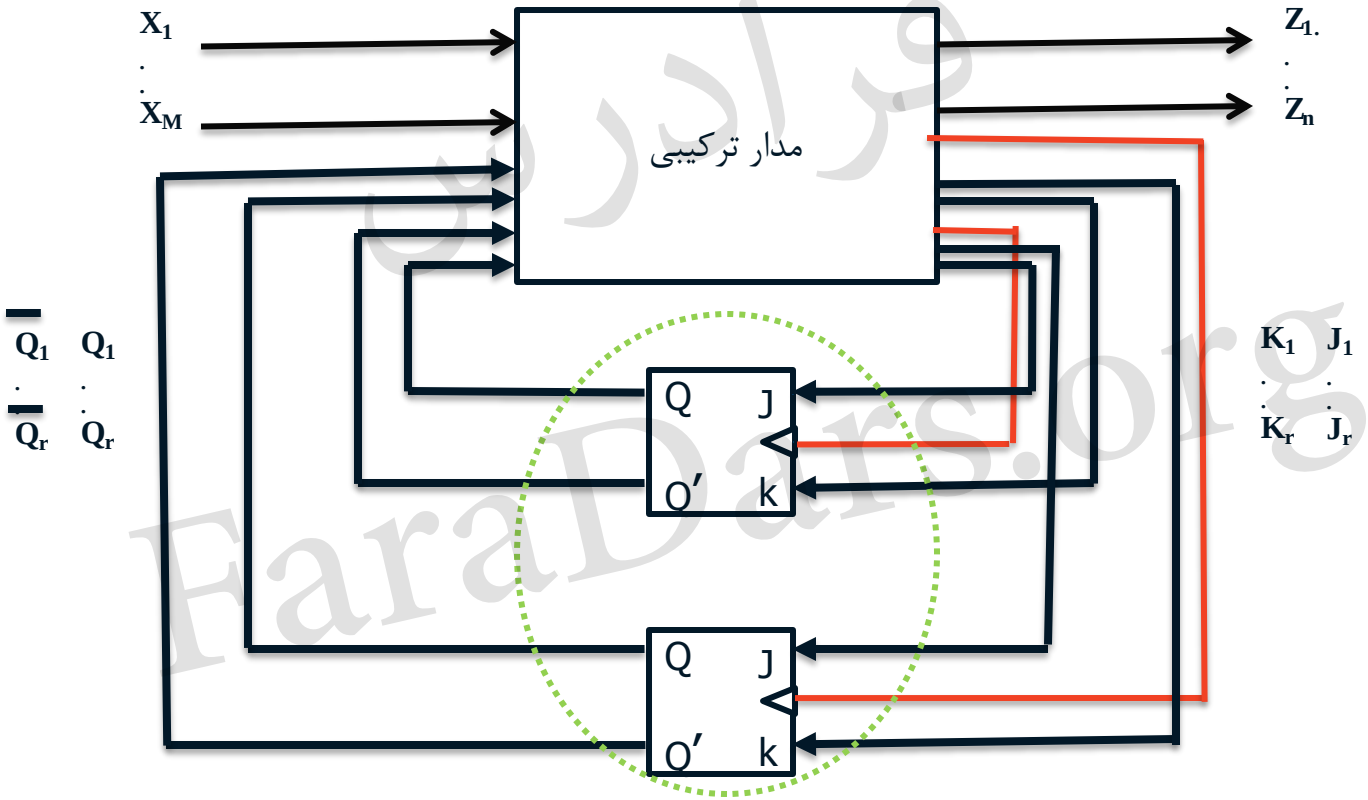
- در مدل مور، خروجی های مدار ترتیبی با ساعت همزمان است زیرا آنها تنها به خروجی فلیپ فلاپ ها بستگی دارند که به پالس ساعت وابسته اند.
- در مدل میلی، خروجی ها فقط هنگامی که ورودی ها در طول پالس ساعت تغییر کنند، عوض می شوند.
- در مدل میلی خروجی ها ممکن است مقادیر غلط لحظه ای داشته باشند، زیرا بین لحظه ورود داده ها و تغییر خروجی ها تاخیر وجود دارد.

برخی نکات در مدارهای میلی و مور

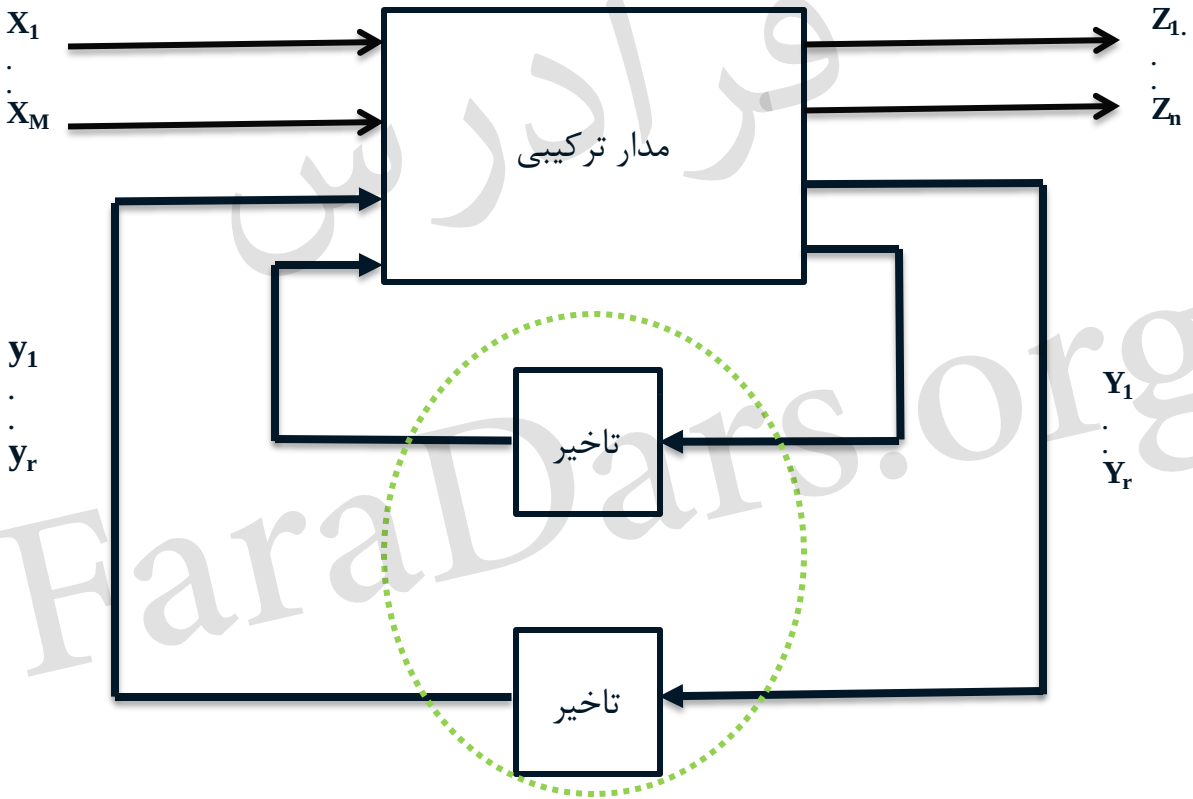
- برای همزمان کردن مدار نوع میلی، ورودی های مدار ترتیبی باید با پالس ساعت همزمان گردند و خروجی ها فقط باید در لبه پالس نمونه برداری شود.

- مزیت مدل میلی نسبت به مور، تعداد حالات کمتر آن و در نتیجه تعداد فلیپ فلاپ های آن کمتر است.

مدارهای پالسی



مدارهای آسنکرون



تحلیل مدارهای ترتیبی سنکرون

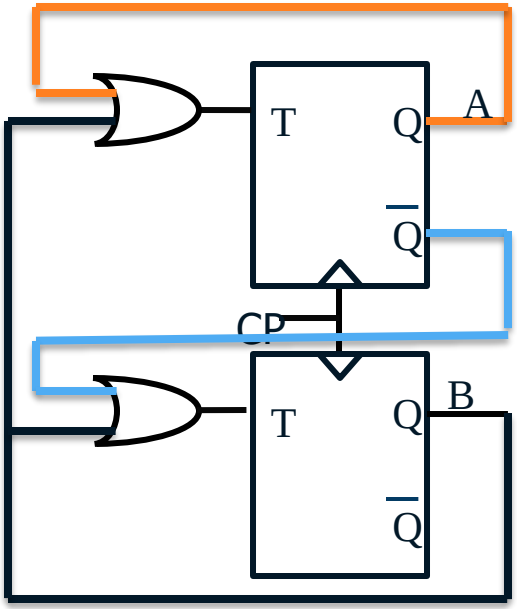
- تحلیل مدار یعنی بررسی تغییر حالت مدار

- هر فلیپ فلاپ یک حافظه تک بیتی (با مقدار ۰ یا ۱) است، که مقدارش در Q ذخیره می شود. پس هر فلیپ فلاپ ۲ حالت دارد، بنابراین مداری که دارای n فلیپ فلاپ است 2^n حالت دارد.

- مدار سنکرون مداری است که کلاک همه فلیپ فلاپ های آن مشترک باشد.

مراحل تحلیل مدارهای ترتیبی سنکرون

1. تابع ورودی همه فلیپ فلاپ ها مشخص کنید.
2. با توجه به تابع حالت بعدی فلیپ فلاپ های استفاده شده در مدار حالت بعدی فلیپ فلاپ ها را مشخص کنید.
3. با توجه به گام ۲ جدول حالت را تشکیل دهید.
4. با توجه به جدول حالت دیاگرام حالت را رسم کنید.



- مثال: مدار زیر یک مدار سنکرون است آن را تحلیل کنید.

$Q^* = T \oplus Q$

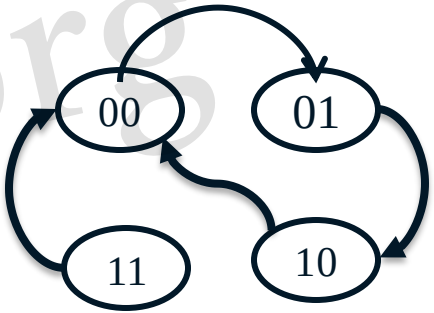
$A^* = T_A \oplus A = A + B \oplus A = A'B$

$B^* = T_B \oplus B = A' + B \oplus B = A'B'$

$T_A = A + B$

$T_B = A' + B$

A	B	A*	B*
0	0	0	1
0	1	1	0
1	0	0	0
1	1	0	0



جدول حالت

- جدول حالت: جدولی که متشکل از چهار بخش حالت فعلی، ورودی، حالت بعدی و خروجی

– یک مدار ترتیبی با m فلیپ فلاپ و n ورودی، نیاز به 2^{m+n} سطر در جدول حالت دارد، بخش حالت بعدی دارای m ستون، یعنی یک ستون در ازاء هر فلیپ فلاپ است. همچنین m ستون برای حالت فعلی و n ستون برای ورودی ها داریم.

- معادله حالت (معادله گذر) : حالت بعدی را بر حسب تابعی از حالات فعلی و ورودی ها بیان می کند.

نمودار حالت (دیاگرام حالت)

- اطلاعات جدول حالت را به صورت گرافیکی نشان می دهد.
- یک حالت با یک دایره نشان داده می شود، که عدد دودویی داخل هر دایره حالت فلیپ فلاپ را بیان می کند.
- گذر بین دو حالت با خطوط جهت داری که دو دایره را به هم وصل می کند نمایش داده می شود.
- خطوط جهت دار با دو عدد که با یک خط مورب از هم جدا شده اند برچسب خورده است.
- عدد سمت چپ بر روی خطوط جهت دار، مقدار ورودی در حالت فعلی است.
- عدد سمت راست بر روی خطوط جهت دار، مقدار خروجی در حالت فعلی در قبال ورودی مربوطه اش است.

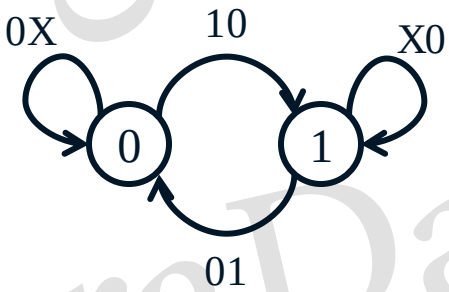
مراحل طراحی مدارات ترتیبی

1. توصیف مدار ترتیبی
2. رسم نمودار حالت (براساس توصیف مدار)
3. بدست آوردن جدول حالت (از روی نمودار حالت)
4. کاهش تعداد حالت
5. تخصیص حالت و بدست آودن جدول انتقال
6. انتخاب نوع فلیپ فلاپ و تعیین تعداد آن
7. رسم جدول های توابع ورودی فلیپ فلاپ ها و خروجی های مدار
8. ساده سازی توابع ورودی فلیپ فلاپ ها و خروجی های مدار
9. رسم مدار

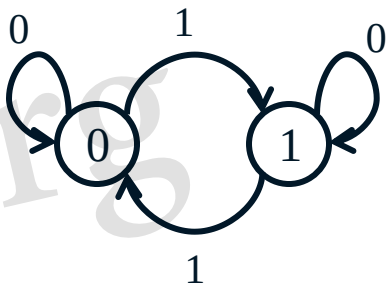
جدول تحریک (Excitation) فلیپ فلاپ ها

- جدول تحریک با توجه به حالت فعلی و حالت بعدی، ورودی فلیپ فلاپ ها را مشخص می کند.

Q	Q*	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

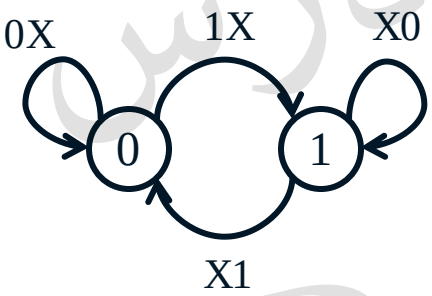


Q	Q*	T
0	0	0
0	1	1
1	0	1
1	1	0

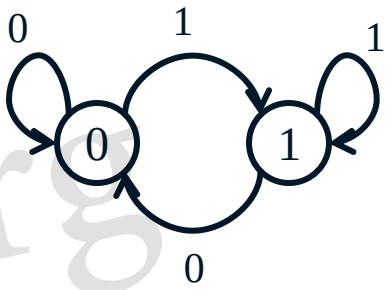


جدول تحریک (Excitation) فلیپ فلاپ ها

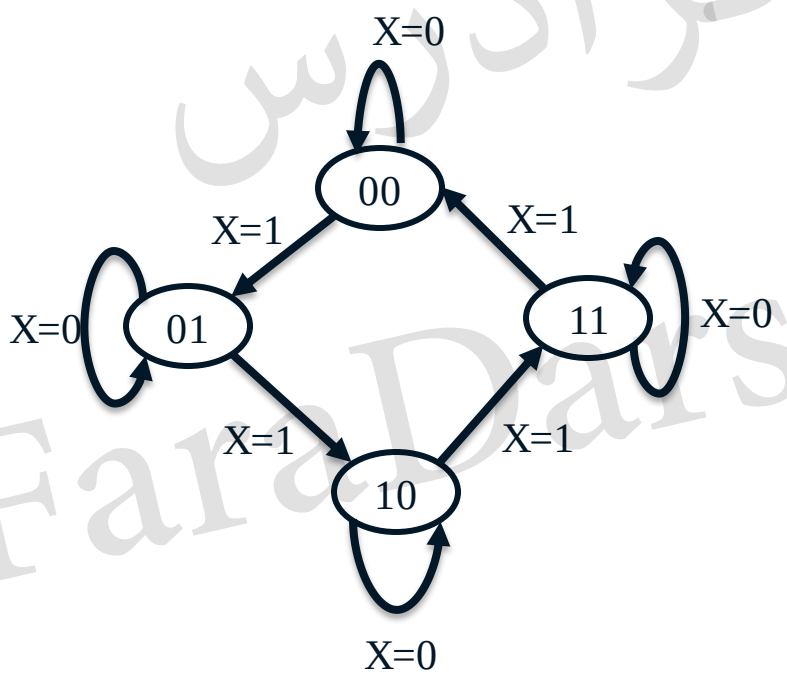
Q	Q*	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



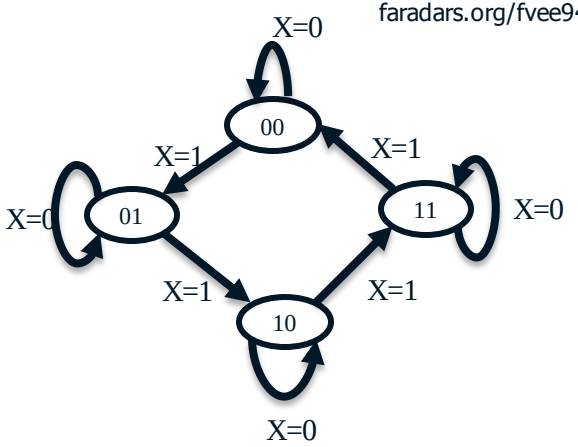
Q	Q*	D
0	0	0
0	1	1
1	0	0
1	1	1



- مثال: یک مدار ترتیبی طراحی کنید که هر وقت ورودی خارجی X برابر یک باشد رشته حالت دودویی 00 ، 01 ، 10 ، 11 را تکرار نماید. به ازء X=0 هم حالت مدار تغییری نکند (این مدار یک شمارنده دودویی دو بیتی است).



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



Q Q*	J K
0 0	0 X
0 1	1 X
1 0	X 1
1 1	X 0

A \ BX	00	01	11	10
0			1	
1	X	X	X	X

$J_A = BX$

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

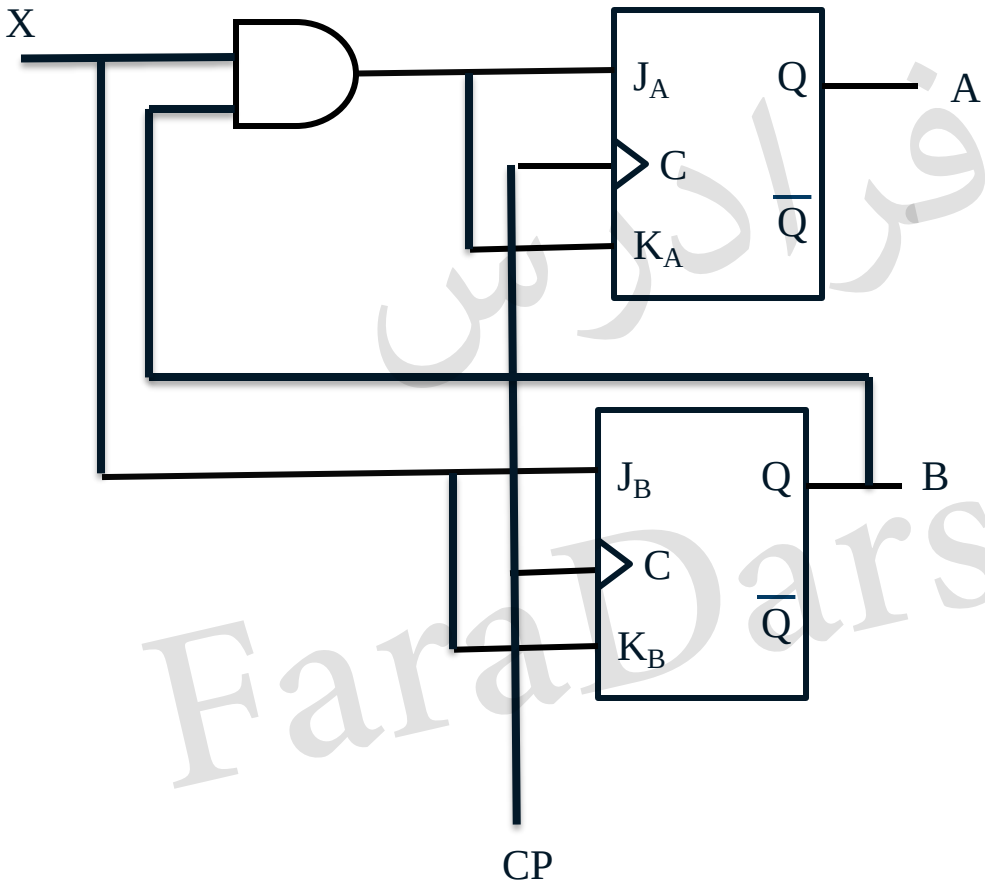
$K_A = BX$

A \ BX	00	01	11	10
0		1	X	X
1		1	X	X

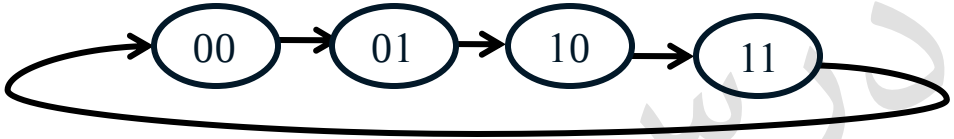
$J_B = X$

A \ BX	00	01	11	10
0	X	X	1	
1	X	X	1	

$K_B = X$



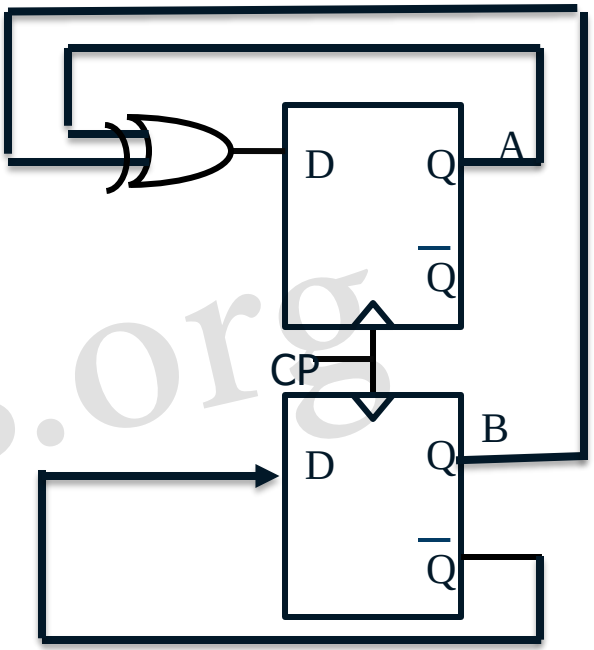
- مثال: با فلیپ فلاپ D یک شمارنده صعودی باینری ۲ بیتی طراحی کنید.



A	B	A*	B*
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

$$D_A = A^* = A \oplus B$$

$$D_B = B^* = \overline{B}$$



کاهش تعداد حالت ها

- دو حالت را معادل گویند هرگاه اولا خروجی های یکسان داشته باشند، و ثانیا حالات بعدی آنها نیز یا یکسان باشد و یا باهم معادل باشد.

- کاهش حالات در جدول فاقد حالات بی اهمیت:
  - روش اول: جدول ایجاب (Implication Table)
  - روش دوم: افراز(Partitioning)

جدول ایجاب (Implication Table)

- مثال: جدول حالت زیر را کاهش حالت دهید.
  - به صورت افقی همه حالات بجز آخری و بصورت عمودی همه حالات بجز اولی را می نویسیم.

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

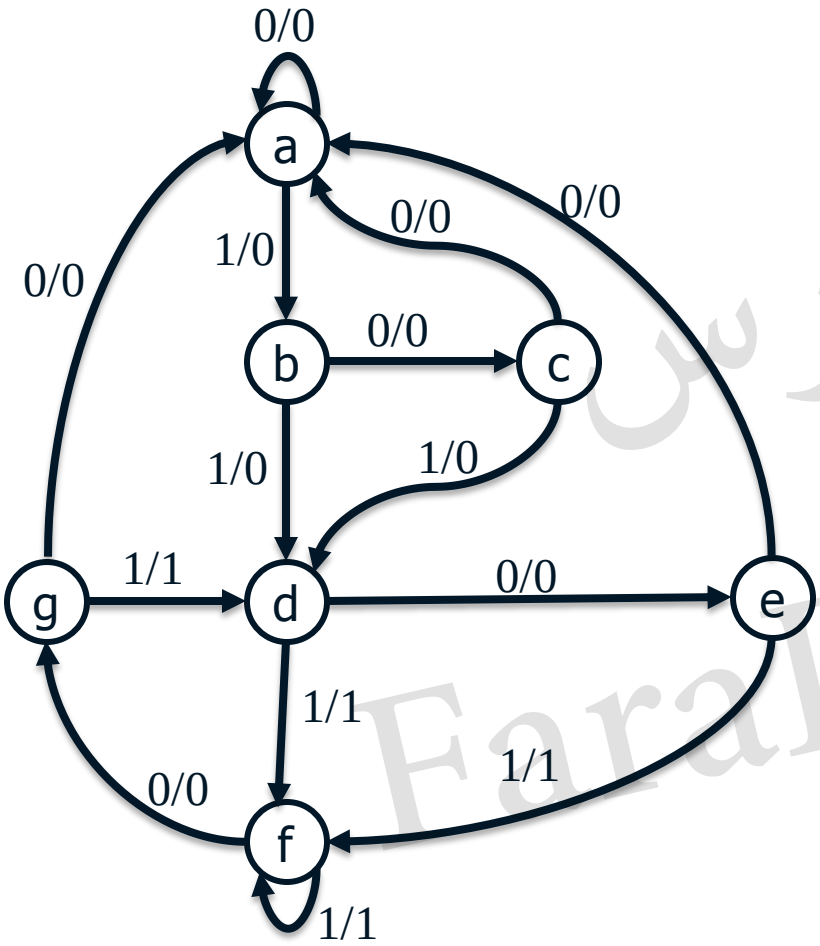
(a,b) (d,e) (d,g)

(d,e,g)

	a	b	c	d	e	f
b	de					
c	×	×				
d	×	×	×			
e	×	×	×	*		
f	cd	ce	×	×	×	
g	×	×	×	de	de	×

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

- مثال: دیاگرام حالت زیر را کاهش دهید.



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

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

(e,g) (d,f)

	a	b	c	d	e	f
b	ac bd					
c	bd	ac				
d	×	×	×			
e	×	×	×	ae		
f	×	×	×	eg	ag	
g	×	×	×	ae df	df	df ag

افراز(Partitioning)

- فقط برای جدول فاقد حالات بی اهمیت جواب می دهد و سریع تر از روش جدول ایجاب است.
- مثال: جدول حالت زیر را کاهش حالت دهید.

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

- افراز P_0 : فقط یک کلاس شامل همه حالات دارد. (a b c d e)
- افراز P_1 : دو کلاس است. در هر کلاس حالت هایی هستند که خروجی هایشان یکسان است.
(a b c) (d e)
- افراز P_2 : حالاتی که در افراز P_1 ، هم کلاس اند را بررسی می کنیم که آیا حالت بعدیشان نیز همکلاس اند یا خیر. اگر نبودند آن کلاس به کلاس های کوچکتری تقسیم می شود. (a) (b c) (d e)
- افراز P_3 : با توجه به افراز P_2 ساخته می شود. (a) (b c) (d) (e)
- افراز P_4 : با توجه به افراز P_3 ساخته می شود. $P_3 = P_4$

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

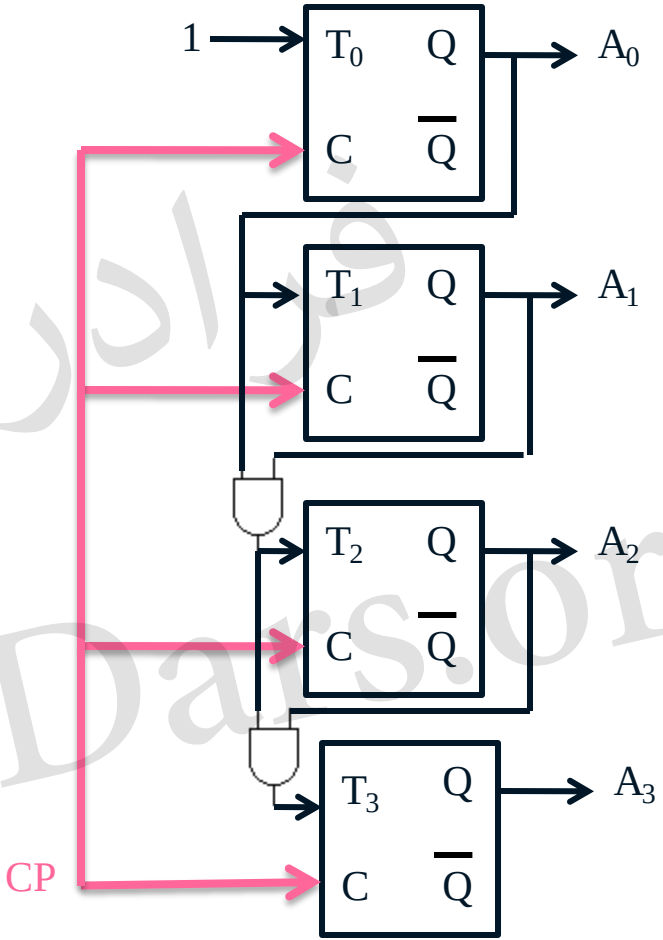
- هرگاه $P_k = P_{k+1}$ یعنی به جواب رسیده ایم.

شمارنده ها

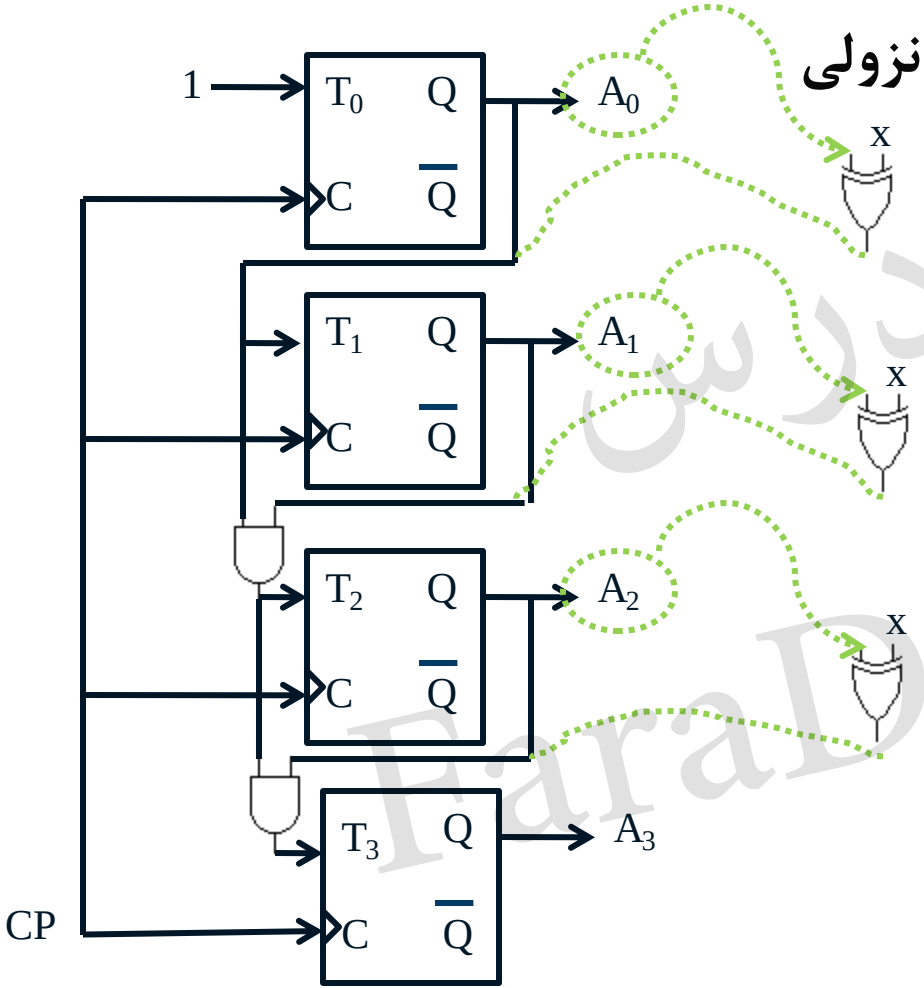
- مثال: یک شمارنده ۴ بیتی صعودی باینری سنکرون با T فلیپ فلاپ طراحی کنید.

  – مدار سنکرون: مداری که کلاک همه فلیپ فلاپ های آن مشترک است.

A_3	A_2	A_1	A_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



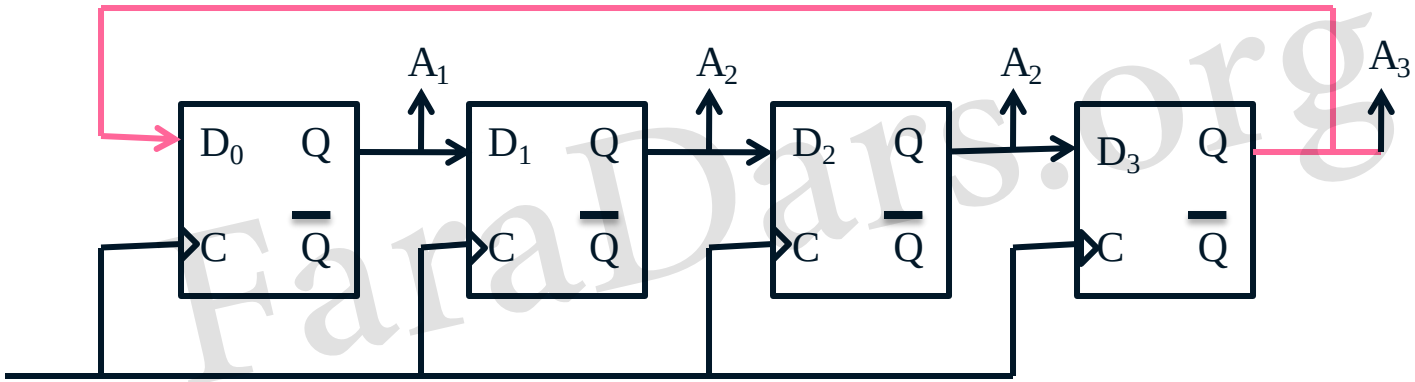
شمارنده صعودی- نزولی



X = 0	شمارش به صورت صعودی
X = 1	شمارش به صورت نزولی

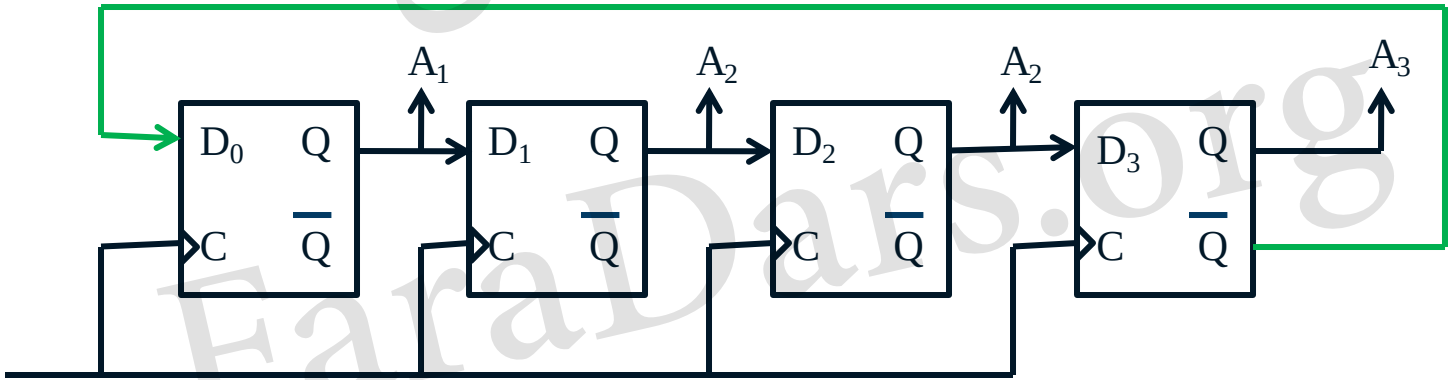
شمارنده حلقوی (Ring Counter)

- شامل تعدادی فلیپ فلاپ D است، که بصورت سریال به یکدیگر متصل شده اند و خروجی هر فلیپ فلاپ به ورودی بعدی متصل است و خروجی آخرین فلیپ فلاپ به ورودی اولین فلیپ فلاپ متصل است.
- شمارنده حلقوی با n فلیپ فلاپ n حالت را می شمارد.



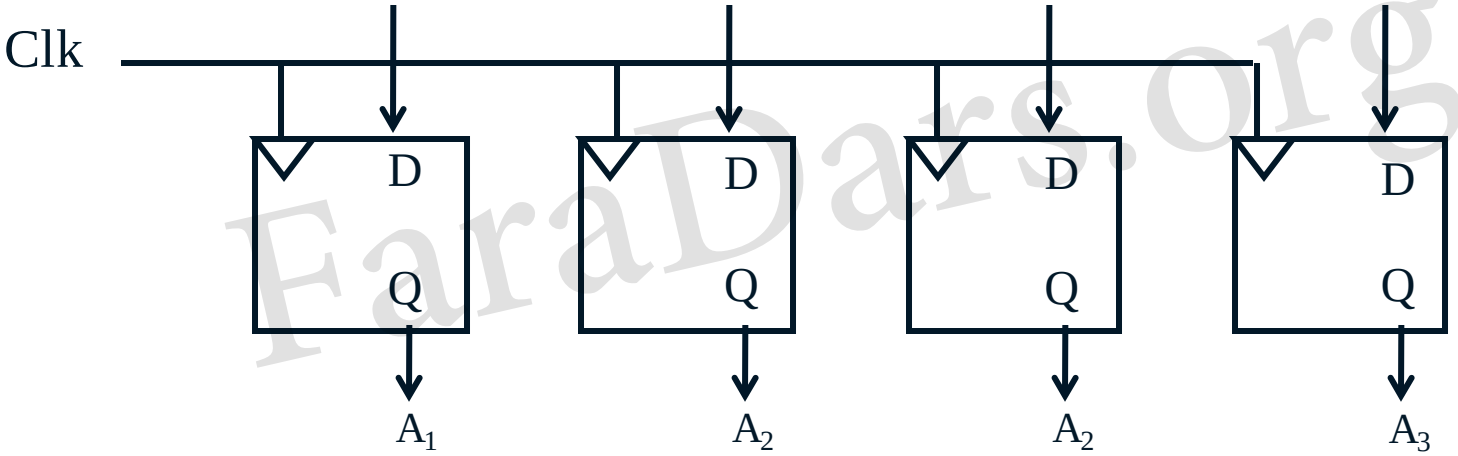
شمارنده جانسون

- مشابه شمارنده حلقوی با این تفاوت که مکمل خروجی آخرین فلیپ فلاپ به ورودی اولین فلیپ فلاپ متصل است.
- شمارنده جانسون با n فلیپ فلاپ 2n حالت را می شمارد.



ثبات ها (Registers)

- مجموعه ای از سلول های حافظه که اطلاعات دودویی را در خود نگه می دارند.
- یک ثبات n بیتی از n فلیپ فلاپ تشکیل شده است.
- معمولا فلیپ فلاپ های تشکیل دهنده ثبات از نوع حساس به لبه یا از نوع تابع- حاکم هستند.
- مثال: طرح ساده یک ثبات با فلیپ فلاپ D

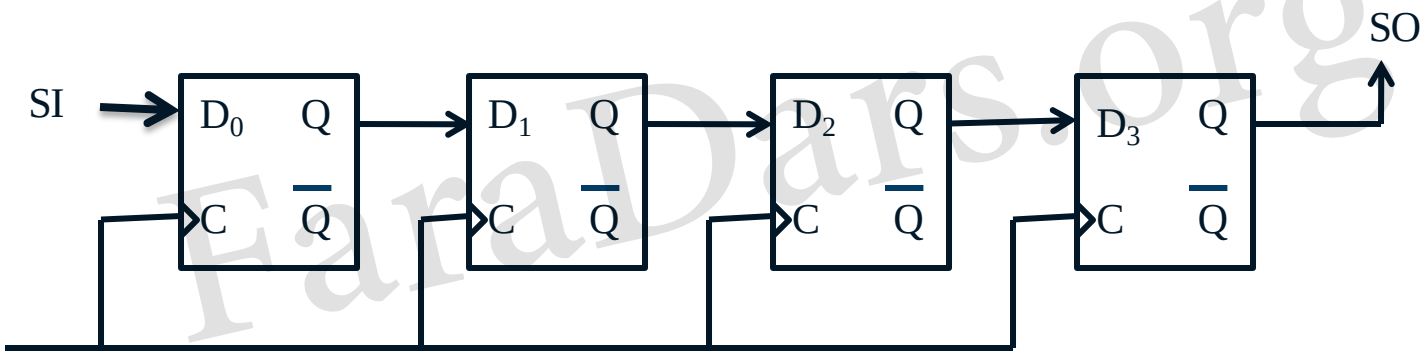


ثبات با امکان بار شدن موازی (Parallel Load)

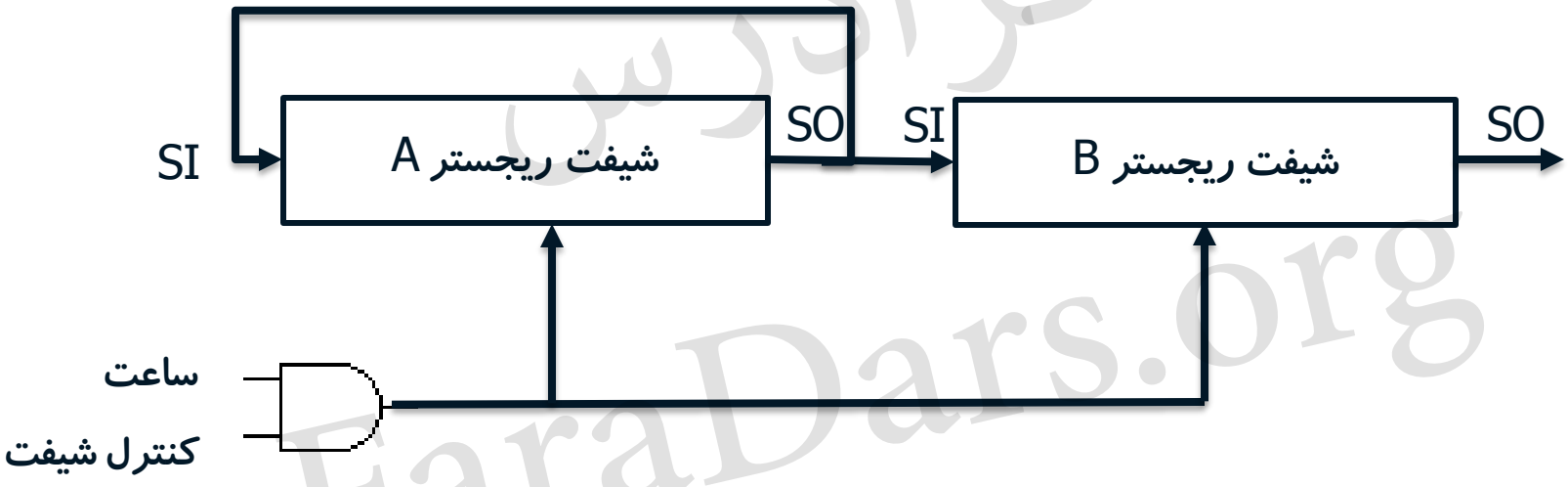
- انتقال اطلاعات جدید به داخل یک ثبات، بار شدن (Loading) اطلاعات به ثبات است.
- بار شدن موازی: اگر همه بیت های یک ثبات با یک پالس ساعت به طور همزمان بار شوند.

شیفت ریجستر

- یک ثبات که قادر است اطلاعات دودویی اش را به سمت چپ یا راست انتقال دهد.
- از زنجیره ای از فلیپ فلاپ های متصل به هم تشکیل شده است به طوری که خروجی یک فلیپ فلاپ به ورودی فلیپ فلاپ دیگر متصل است.
- همه فلیپ فلاپ ها پالس ساعت مشترک دریافت می کنند.



انتقال سریال

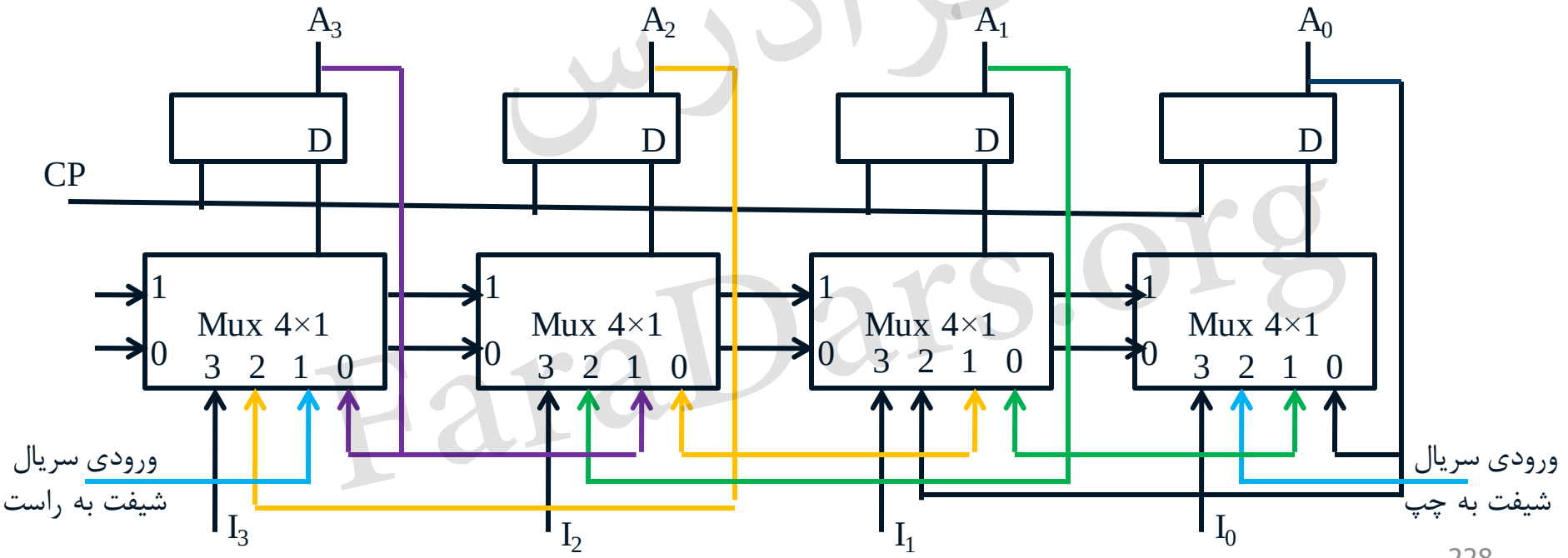


انواع شیفت ریجستر

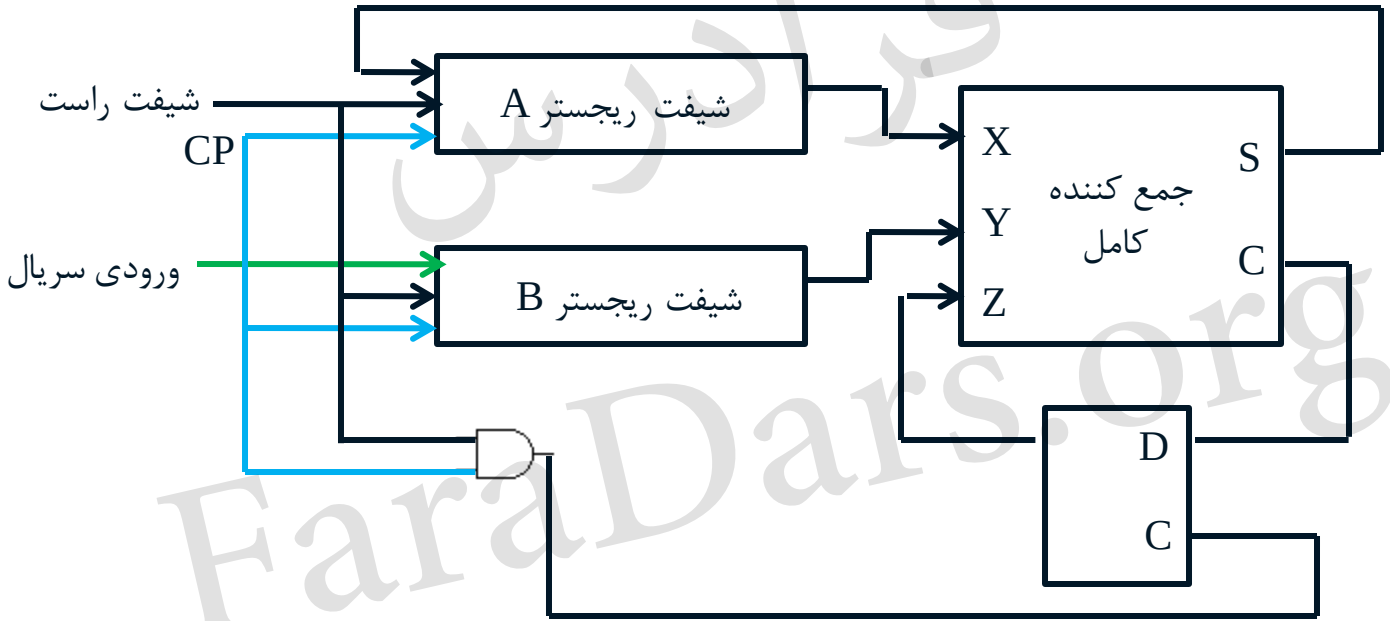
1. SISO (Serial In Serial Out) : یک ورودی سریال و یک خروجی سریال دارد.
2. SIPO (Serial In Parallel Out) : اگر همه خروجی فلیپ فلاپ ها را به عنوان خروجی انتخاب کنیم.
3. PISO (Parallel In Serial Out) : ورودی موازی به همه فلیپ فلاپ ها اعمال می شود و خروجی از یک فلیپ فلاپ گرفته می شود (کاربرد ندارد).
4. PIPO (Parallel In Parallel Out) : ورودی موازی خروجی نیز موازی است، یعنی ورودی ها به همه فلیپ فلاپ ها داده می شود و از همه فلیپ فلاپ ها گرفته می شود.

شیفت ریجستر یونیورسال

- ثباتی که قادر به جابه جایی داده در دو جهت و بار شدن موازی است.



جمع کننده سریال



این اسلاید ها بر مبنای نکات مطرح شده در فرادرس
«آموزش جامع مدارهای منطقی به صورت تئوری و عملی»
تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید.

faradars.org/fvee9403