فصل اول

سیستم‌های دودوئی

مقدمه:

هدف از یادگیری این درس طراحی و ایجاد مدارهایی می‌باشد که وظیفه‌مندی خاصی را دنبال می‌کند، که طراح انتظار دارد. از این درس می‌توان در تهیه ساعتهای دیجیتال، شمارنده ها و ... استفاده کرد.

نمایش اعداد :

هر عدد را می‌توان با استفاده از ارقام آن به شکل زیر نمایش داد، همچنین می‌توان با استفاده از رابطه(۱-۱) و بدست آوردن مقدار Σ وزن آن عدد را بدست آورد.

رابطه ۱-۱:

$$N=(a_{n-1}a_{n-2}a_{n-3}\ \dots\ a_1a_0\ .\ a_{-1}a_{-2}a_{-3}\ \dots\ a_{-m})_r=\sum_{i=-m}^{n-1}a_i r^i$$

مثال:

وزن عدد $(365)_8$ را به دست آورید

$$(365)_8=(5\times8^0)+(6\times8^1)+(3\times8^2)=5+48+192=245$$

تبدیل مبنا:

تبدیل مبنای ده به دو:

برای این کار از دو راه می‌توان استفاده کرد:

استفاده از تقسیمات متوالی: به این صورت که عدد مبنای 10 را بر 2 تقسیم می‌کنیم و باقی مانده را بدست می‌آوریم. تا زمانی که خارج قسمت کوچکتر از 2 نشده به تقسیم کردن ادامه می‌دهیم. سپس آخرین خارج قسمت (که به هر شکل یک می‌باشد) را به عنوان سمت چپ‌ترین بیت در نظر می‌گیریم. و بعد از آن باقی مانده ها را که یا صفر و یا یک هستند به ترتیب از سمت راست انتخاب می‌کنیم و می‌نویسیم.

مثال:

عدد $(34)_{10}$ را به مبنای 2 تبدیل کنید:

```
34  |2
34  17  |2
 0  16   8  |2
     1   8   4  |2     ⇒ (100010)2
         0   4   2  |2
             0   2   1
                 0
```

با استفاده از تفریقات متوالی: به اینصورت که با در نظر گرفتن عدد مورد نظر باید یک n پیدا کنیم که 2^n بزرگترین عدد باشد که از عدد مورد نظر ما کوچکتر است. به عنوان مثال اگر عدد دلخواه ما 273 بود باید n را 8 در نظر بگیریم یعنی $2^8=256$ که از عدد ما کوچکتر است اگر n را 7 فرض می‌کردیم $2^7=128$ از عدد ما کوچکتر بود اما هنوز عدد دیگری مانند 256 وجود داشت که به عدد مورد نظر یعنی 273 نزدیکتر بود. پس از انتخاب n، حاصل 2^n را از عدد مورد نظر کم می‌کنیم.

$$273-2^8=273-256=17$$

بنابراین با توضیحات فوق به این نتیجه می‌رسیم که در عدد 273 یک 256 تا وجود داشته حال باید همین کار را در مورد باقی مانده انجام دهیم.

$17 \rightarrow n = 4 \rightarrow 17 - 2^4 = 1 \Rightarrow 1 \rightarrow n = 0$

$1 - 2^0 = 0$

حال با جایگذاری خواهیم داشت:

256	128	64	32	16	8	4	2	1	
(1	0	0	0	1	0	0	0	1)$_2$	

به بیان ساده تر برای تبدیل باید ببینیم که در عددی که می‌خواهیم از مبنای 10 به 2 ببریم تا 2^n، چند 2^{n-1}، چند 2^{n-2} و ... و بالاخره چند 2^0 وجود دارد.

تبدیل از مبنای دو به ده:

بااستفاده ازوزن اعدادصورت میگیرد؛ یعنی میتوان بااستفاده ازرابطه (۱-۱) تبدیل مبنای ۲ به ۱۰راانجام داد.

مثال:

عدد $(10010)_2$ را به مبنای 10 تبدیل کنید.

$(100010)_2 = (0\times 2^0)+(1\times 2^1)+(0\times 2^2)+(0\times 2^3)+(0\times 2^4)+(1\times 2^5) = 34$

نکته: در واقع در این روش باید وزن اراقم را از سمت راست بالای اراقم بنویسیم که از 2^0 شروع می‌شود و تا رسیدن به n رقم به 2^n می‌رسد. حال در انتها برای بدست آوردن وزن این اعداد باید فقط وزن‌هایی که پایهٔ یک دارند با هم جمع شوند.

مثال:

وزن $(100110)_2$ را بدست آورید.

وزن	2^6	2^5	2^4	2^3	2^2	2^1	2^0
پایه	1	0	0	0	1	1	0

$2^6 + 2^2 + 2^1 = 64 + 4 + 2 = 70$

یا به صورت

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

$= 64 + 4 + 2 = 70$

تبدیل از مبنای دَه به شانزده:

با استفاده از تقسیمات متوالی صورت می گیرد:

به اینصورت که باید عدد مبنای 10 را بر 16 تقسیم کنیم و این کار را تا جایی ادامه دهیم که خارج قسمت کوچکتر از 16 شود. سپس با انتخاب آخرین خارج قسمت و همچنین باقی مانده‌ها از راست به چپ شروع به نوشتن عدد می کنیم. منتها باید توجه داشت که در تقسیمات تا جایی که باقی‌مانده ها کوچکتر از 10 هستند مشکلی وجود ندارد اما برای باقی مانده‌های 10 تا 15 به شکل زیر عمل می کنیم.

یعنی باید حروف A و B و C و D و E و F را جایگزین کنیم.

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

مثال:

عدد 346 را به مبنای ۱۶ ببرید.

346	16	
336	21	16
10	16	1
	5	

$(1\ 5\ A)$ تبدیل می شود به $\rightarrow$ $(1\ 5\ 10)$

تبدیل از مبنای شانزده به ده:

با استفاده از وزن اعداد صورت می گیرد؛ یعنی با استفاده از رابطه (۱-۱)تبدیل مبنای 16 به 10 را انجام می‌دهیم.

مثال:

عدد $(15F)_{16}$ را به مبنای 10 تبدیل کنید.

$$(15F)_{16} = (F\times16^0)+(5\times16^1)+(1\times16^2) = 15+80+256 = 351$$

تبدیل از مبنای ده به هشت:

با استفاده از تقسیمات متوالی انجام می‌شود به این صورت که عدد مبنای 10 را بر 8 تقسیم می کنیم و باقی مانده را بدست می‌آوریم، سپس خارج قسمت را بر 8 تقسیم می کنیم و به همین شکل ادامه می‌دهیم تا زمانی که خارج قسمت کوچکتر از 8 شود. سپس از راست به چپ شروع به نوشتن ارقام می کنیم.

مثال:

عدد 245 را به مبنای 8 تبدیل کنید.

245	8	
240	30	8
5	24	3
	6	

$\Rightarrow \quad (365)_8 = (245)_{10}$

تبدیل مبنای هشت به ده:

این عمل با کمک وزن اعداد صورت می گیرد به این صورت که با استفاده از رابطه (۱-۱) می‌توان تبدیل مبنای هشت به 10 را انجام داد.

مثال:

$$(361)_8 = (1\times8^0)+(6\times8^1)+(3\times8^2) = 1+48+192 = 241$$

تبدیل مبنای دو به شانزده:

از سمت راست جداسازی بصورت 4 بیت 4 بیت انجام میدهیم سپس با استفاده از وزن اعداد به جواب می‌رسیم.

فقط باید توجه داشت که در هنگام جداسازی اگر در دسته آخر کمتر از ۴ بیت قرار گرفت به اندازه تعداد بیت کمتر سمت چپ صفر می گذاریم و دسته را کامل می کنیم. هر بیت رادر وزن متناظرش ضرب می‌کنیم و اعداد بدست آمده در یک دسته را با هم جمع می‌کنیم.

مثال:

$(100010)_2$ را به مبنای 16 ببرید.

$$\begin{array}{cccccccc} 8 & 4 & 2 & 1 & 8 & 4 & 2 & 1 \\ (0 & 0 & 1 & 0 & 0 & 0 & 1 & 0)_2 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \underbrace{0+0+2+0}_{2} & & & & \underbrace{0+0+2+0}_{2} & & & \end{array} \Rightarrow = (22)_{16}$$

تبدیل مبنای شانزده به دو:

برای تبدیل مبنای شانزده به دو باید هر رقم را به صورت چهار بیتی در مبنای دو نوشت لازم به ذکر است که جایگاه ارقام نباید تغییر کند.

مبنای 16	مبنای 2
0	0000
1	0001
2	0010
3	0011
4	0100
⋮	⋮
A ← 10	1010
B ← 11	1011
C ← 12	1100
D ← 13	1101
E ← 14	1110
F ← 15	1111

مبنای 16 = 4 بیت

مثال: عدد (1 6 D) در مبنای 16 را به مبنای 2 تبدیل کنید.

$$(1\ 6\ D)_{16} \rightarrow \begin{cases} 1 \rightarrow 0001 \\ 6 \rightarrow 0110 \\ D \rightarrow 1101 \end{cases} \rightarrow (000101101101)_2$$

تبدیل مبنای دو به هشت:

در این تبدیل جداسازی 3 بیتی انجام می‌دهیم:

به اینصورت که از سمت راست سه بیت، سه بیت جدا می کنیم سپس با استفاده از مقداردهی وزنی، وزن هر دسته را به دست می آوریم. باید توجه کرد اگر آخرین دسته تعداد بیتهایش کمتر از سه بود به اندازه تعداد بیت کمتر صفر در سمت چپ قرار می‌دهیم تا دسته کامل شود. هر بیت را در وزن متناظرش ضرب می کنیم و اعداد بدست آمده در یک دسته را با هم جمع می کنیم.

مثال:

عدد $(100010)_2$ را به مبنای 8 تبدیل کنید.

$$100010 \Rightarrow \begin{cases} \text{دسته اول: } 010 \Rightarrow \text{با استفاده از وزن‌ها} = 2 \\ \text{دسته دوم: } 100 \Rightarrow \text{با استفاده از وزن‌ها} = 4 \end{cases} \Rightarrow (42)_8$$

تبدیل مبنای هشت به دو:

برای تبدیل مبنای 8 به دو باید هر رقم را به صورت مبنای 2 اما به صورت سه بیتی بنویسیم. ضمناً باید توجه کرد که جایگاه ارقام نباید تغییر کند.

مبنای 8 = 3 بیت

مبنای 8	مبنای 2
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

مثال:

عدد $(712)_8$ را به مبنای 2 تبدیل کنید.

$$(712)_8 = (\ ?\ )_2 \Rightarrow \begin{cases} 2 \rightarrow 010 \\ 1 \rightarrow 001 \\ 7 \rightarrow 111 \end{cases} \Rightarrow (111001010)_2$$

تبدیل مبنای شانزده به هشت:

برای اینکه تبدیل مبنای 16 به 8 را انجام دهیم بهتر است ابتدا عدد در مبنای 16 را به مبنای 2 ببریم سپس عدد بدست آمده در مبنای 2 را به مبنای 8 تبدیل کنیم.

مثال:

عدد $(5D3)_{16}$ را به مبنای 8 ببرید.

$$(5D3)_{16} = (010111010011)_2 = (2723)_8$$

تبدیل مبنای هشت به شانزده:

برای تبدیل مبنای 8 به 16 می‌توان عمل عکس تبدیل 16 به 8 را انجام داد به این معنی که ابتدا عدد مورد نظر را به مبنای 2 ببریم سپس حاصل را به مبنای 16 بازگردانیم.

مثال:

عدد $(376)_8$ را به مبنای 16 تبدیل کنید.

$$(376)_8 = (011111110)_2 = (FE)_{16}$$

برای تبدیل عدد اعشاری در هر مبنا به مبنای 10 با استفاده از رابطه (۱-۱) باید وزن آن را بدست آوریم.

مثال:

عدد $(100010/110)_2$ را به مبنای 10 تبدیل کنید:

$$(100010/110)_2 = \begin{cases} \text{قسمت صحیح} & = (0\times2^0)+(1\times2^1)+(0\times2^2)+(0\times2^3)+(0\times2^4)+(1\times2^5)=34 \\ \text{قسمت اعشاری} & = (1\times2^{-1})+(1\times2^{-2})+(0\times2^{-3})=0.75 \end{cases}$$

بنابراین حاصل برابر است با:

$$(34.75)_{10}$$

تبدیل مبنای اعشاری:

برای تبدیل عدد اعشاری در مبنای ده به هر مبنا (به عنوان مثال مبنای r) باید عمل ضرب متوالی را انجام دهیم، عملیات به این شکل است که قسمت صحیح را به مانند قبل و با استفاده از روشهایی که قبلاً اشاره شد، بدست می‌آوریم. اما برای بدست آوردن قسمت اعشاری به شکل زیر عمل می‌کنیم:

ابتدا قسمت اعشار را جدا می‌کنیم و آن را در مبنا مورد نظر یعنی r ضرب می‌کنیم حاصل این ضرب یک قسمت صحیح و یک قسمت اعشاری خواهدداشت.حال قسمت صحیح راجدامی کنیم وقسمت اعشاری رادرمبنای r دوباره ضرب می کنیم،این عمل را تا جایی ادامه می‌دهیم تا به قسمت اعشاری صفر برسیم و یا اینکه به یک دوره گردش دست پیدا کنیم.

توجه: اگر در صورت سئوال خواسته شده بود که تا n رقم اعشار را حساب کنیم باید n بار عمل ضرب را انجام دهیم.

مثال:

عدد $(0.513)_{10}$ را به مبنای 2 و 8 تبدیل کنید.

$$(0.513)_{10} = \begin{aligned} &0.513\times2=1.026\rightarrow0.026 \\ &0.026\times2=0.052 \\ &0.052\times2=0.104 \\ &0.104\times2=0.208 \\ &0.208\times2=0.416 \\ &0.416\times2=0.832 \end{aligned}$$

$$(0.513)_{10} = (0.100000.....)_2$$

$$(0.513)_{10} = \begin{aligned} &0.513\times8=4.104 \\ &0.104\times8=0.832 \\ &0.832\times8=6.656 \\ &0.656\times8=5.248 \\ &0.248\times8=1.984 \end{aligned}$$

$$(0.513)_{10} = (0.40651....)_8$$

محاسبات:

اگر بخواهیم عملیات جمع را در مبنای r انجام دهیم به این ترتیب عمل می‌کنیم که، ابتدا ارقام از سمت راست زیر هم قرار می‌گیرند سپس از همان سمت راست شروع به جمع رقم به رقم می‌کنیم به این صورت که پس از جمع دو رقم دو حالت ممکن است برای حاصل جمع پیش بیاید. حالت اول زمانی است که حاصل جمع بزرگتر یا مبنای r باشد که در این صورت باید یک مبنای r از حاصل جمع کم شود باقی مانده نوشته می‌شود و یک عدد "1" به بالای رقم بعدی انتقال می‌یابد. اما اگر حاصل جمع کوچکتر از مبنا، بود تنها کافیست که همان حاصل نوشته شود.

Carry: در زمانی که دو عدد n رقمی را با هم جمع می‌کنیم انتظار داریم که حاصل نیز عدد n رقمی باشد اما زمان‌هایی به وجود می‌آید. که حاصل 1+ n رقمی می‌شود به این رقم جدید به وجود آمده رقم نقلی یا carry گفته می‌شود.

مثال:

اعداد $(457)_8$ و $(203)_8$ را جمع کنید.

457
+203

ابتدا 3 و 7 با یکدیگر جمع می‌شود حاصل عدد 10 خواهد بود که چون حاصل بزرگتر از مبنای 8 می‌باشد باید یک مبنای 8 از 10 کم شود.

10
− 8

2

حاصل بدست آمده را در ستون اول می‌نویسیم و یک عدد 1 در بالای رقم سمت چپ می‌نویسیم.

1
457
+ 203

2

حال باید 5 و صفر و یک جمع شود حاصل 6 می‌شود که از مبنای 8 کوچکتر است. بنابراین خودش را به عنوان حاصل جمع می‌نویسیم.

1
457
+ 203

662

حال اگر بخواهیم عملیات تفریق در مبنای r را انجام دهیم به این صورت عمل می‌کنیم که ابتدا ارقام از سمت راست زیرهم قرار گیرند سپس از همان سمت راست شروع به عملیات تفریق رقم به رقم می‌کنیم. در این حالت ممکن است رقم عدد بالائی بزرگتر از رقم عدد پائینی باشد می‌توان به راحتی عمل تفریق را انجام داد.

اما اگر رقم بالائی کوچکتر از رقم پائینی باشد در این صورت باید از رقم سمت چپ یکی قرض گرفت البته این عملیات در صورتی که رقم سمت چپ صفر باشد باید از رقم سمت چپ بعدی برای آن صفر قرض گرفت. پس از قرض گرفتن باید به رقم سمت راست یک مبنا اضافه شود و سپس عملیات تفریق انجام می‌شود.

مثال: تفریق زیر را در مبنای 16 انجام دهید.

A05
− 3B7

چون رقم سمت راست عدد اول کوچکتر از رقم سمت راست عدد دوم است بنابراین باید از رقم دوم قرض گرفته و چون این رقم صفر است باید از رقم سوم قرض گرفته شود. بنابراین عدد A تبدیل به 9 می‌شود و رقم دوم یعنی صفر تبدیل به 16 می‌شود (یک عدد مبنا به آن اضافه می‌شود) حال باید از این عدد یکی کم شود و رقم اول یعنی 5 تبدیل به 21 (21 = 5 + 16) می‌شود.

```
      15
    9 16 21
    A  0  5
-   3  B  7
-------------
    6  4  E
```

مثال: جمع و تفریق زیر را در مبنای 2 انجام دهید:

```
      1 1 1 1 1
    + 1 1 0 1 0 1
    + 0 1 1 1 1 1
-------------------
  1 0 1 0 1 0 0
```

```
        2 1 2
    0 1 2 0 2
    1 1 0 1 0 1
-   0 1 1 1 1 1
-------------------
    0 1 0 1 1 0
```

برای انجام عملیات ضرب در مبنای ۲ نیز باید مانند عملیات جمع و قوانین مربوط به ضرب را در نظر بگیریم.

مثال: ضرب زیر را در مبنای 16 انجام دهید.

```
  A2F
× 50E
```

ابتدا E را در عدد A2F ضرب می‌کنیم این کار را باید رقم انجام دهیم. یعنی ابتدا E را در F ضرب می‌کنیم در این صورت حاصل برابر 210 می‌شود حال باید ببینیم که چند مبنا در این حاصل وجود دارد. مشاهده می‌شود که در عدد 210 می‌توان 13 عدد 16 ایجاد کرد. حال باید 13 عدد 16 را از 210 کم کنیم. در این صورت خواهیم داشت:

```
  210
-13 × 16
---------
    2
```

باقی ماندهٔ 2 به عنوان رقم سمت راست قرار می‌گیرد و 13 یا D روی رقم سمت چپ قرار می‌گیرد و سپس E در 2 ضرب می‌کنیم حاصل 28 می‌شود این حاصل با 13 یا D که از مرحله قبل آمده جمع می‌شود. حاصل 41 می‌شود همان‌طور که مشاهده می‌شود دو مبنای 16 در 41 موجود است.

```
   41
- 2 × 16
---------
    9
```

حاصل 9 به عنوان جواب قرار می‌گیرد و 2 در روی رقم سمت چپ قرار می‌گیرد . حال با همین روش گفته شده E را در A ضرب می‌کنیم حاصل 140 می‌شود و با 2 جمع می‌کنیم. حاصل 142 می‌شود که در آن 8 بار مبنای 16 وجود دارد. باید از 142 این 8 مبنای 16 را کم کنیم.

```
   142
-  8 × 16
-----------
  E یا 14
```

حاصل 14 یا E می‌نویسیم و 8 را بالای رقم سمت چپ می‌بریم اما چون رقم دیگری وجود ندارد 8 پائین می‌آید.

```
   2 13
   A  2  F
×        E
-----------
8  E  9  2
```

حال یک رقم به سمت چپ شیفت می‌دهیم.

چون یک رقم صفر است می‌توان دوبار به سمت چپ شیفت دهیم.

```
      A  2  F
×     5  0  E
--------------
8  E  9  2
```

```
         4
      A  2  F
×           5
--------------
3  2  E  B
```

```
         1
      8  E  9  2
   1
+  0  0  0  0  0  0
+  3  2  E  B  0  0
-----------------
   3  3  7  9  9  2
```

قوانین جمع و تفریق و ضرب در مبنای 2 داریم:

$$\begin{array}{r} {\scriptstyle 0}\\ +\,0\\ \hline 0 \end{array} \qquad \begin{array}{r} {\scriptstyle 1}\\ +\,0\\ \hline 1 \end{array} \qquad \begin{array}{r} {\scriptstyle 0}\\ +\,1\\ \hline 1 \end{array} \qquad \begin{array}{r} {\scriptstyle 1}\\ +\,1\\ \hline \textcircled{1}0 \end{array}$$

carry

مثال:

حاصل جمع زیر را بدست آورید:

```
  1001101
+0111101
---------
1 0001010
```

carry (رقم نقلی)

تفریق:

در تفریق باینری داریم:

$$\begin{array}{r} {\scriptstyle 0}\\ -\,0\\ \hline 0 \end{array} \qquad \begin{array}{r} {\scriptstyle 0}\\ -\,1\\ \hline \textcircled{1}1 \end{array} \qquad \begin{array}{r} 1\\ -\,0\\ \hline 1 \end{array} \qquad \begin{array}{r} 1\\ -\,1\\ \hline 0 \end{array}$$

مثال:

حاصل تفریق زیر را بدست آورید:

```
  1100101
- 0111010
---------
  0101011
```

ضرب:

در ضرب باینری داریم:

$$\begin{array}{cccc} 1 & 1 & 0 & 0 \\ \times & \times & \times & \times \\ 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 \end{array}$$

مثال:

حاصل ضرب زیر را بدست آورید:

$$\begin{array}{r} 1011 \\ \times 1101 \\ \hline 1011 \\ + \ 0000 \\ + \ 1101 \\ + \ 1101 \\ \hline 10100111 \end{array}$$

قوانین جمع و ضرب در مبنای ۸:

قوانین مربوط به جمع

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

قوانین مربوط به ضرب

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

قوانین جمع و ضرب در مبنای 10:

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

قوانین مربوط به جمع

×	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	10	12	14	16	18
3	0	3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5	0	5	10	15	20	25	30	35	40	45
6	0	6	12	18	24	30	36	4	48	54
7	0	7	14	21	28	35	42	49	56	63
8	0	8	16	24	32	40	48	56	64	72
9	0	9	18	27	36	45	54	63	72	81

قوانین مربوط به ضرب

قوانین جمع و ضرب در مبنای 16:

قوانین مربوط به جمع

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

قوانین مربوط ضرب

×	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	10	12	14	16	18	1A	1C	1E
3	0	3	6	9	C	F	12	15	18	1B	1E	21	24	27	2A	2D
4	0	4	8	C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0	5	A	F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0	6	C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0	7	E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	0	8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	0	9	18	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	0	A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	0	B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	0	C	18	24	30	3C	48	54	60	6C	73	84	90	9C	A8	B4
D	0	D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	0	E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	0	F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

مکمل گیری:

دو نوع مکمل گیری وجود دارد:

- مکمل r
- مکمل r-1

اگر بخواهیم از عدد $(N)_r$ مکمل گیری کنیم، خواهیم داشت که:

r مبنای عدد n تعداد ارقام صحیح

m تعداد ارقام اعشاری

مکمل r-1 عدد N: $[N]_{r-1}$ مکمل r عدد N: $[N]_r$

رابطه (۲-۱)

$$[N]_r = r^n - (N)_r$$

رابطه (۳-۱)

$$[N]_{r-1} = r^n - r^{-m} - (N)_r$$

اگر عدد N،رقم اعشار نداشته باشد برای بدست آوردن مکمل r-1 آن باید m را صفر در نظر بگیریم.

$$m = 0 \Rightarrow [N]_{r-1} = r^n - 1 - (N)_r$$

اگر پایه مورد بحث ما دسیمال دهدهی (Decimal) بود یعنی r=10 می‌توان هم از روابط (۲-۱) و (۳-۱) برای بدست آوردن مکمل r و r-1 استفاده کرد و هم از روش زیر:

اگر بخواهیم مکمل r یک عدد در مبنای r را بدست آوریم کافیست بالای تمام ارقام آن عدد صفر بگذاریم و یک عدد یک در سمت چپ آن صفرها بگذاریم و سپس عمل تفریق را انجام دهیم. در ضمن اگر بخواهیم مکمل $(r-1)$ یک عدد در مبنای r را بدست آوریم باید بالای هر رقم آن یک $(r-1)$ قرار دهیم و سپس عمل تفریق را انجام دهیم.

مثال:

مکمل 10 عدد 256 را بدست آورید:

مکمل 10 عدد 256

$$\begin{array}{r} 1000 \\ -\ 256 \\ \hline 744 \end{array}$$

مثال:

مکمل 9 عدد 657 را بدست آورید:

مکمل 9 عدد 657

$$\begin{array}{r} 999 \\ -657 \\ \hline 342 \end{array}$$

مثال:

مکمل 8 عدد $(561)_8$ را بدست آورید.

مکمل 8 عدد 561

$$\begin{array}{r} 1000 \\ -561 \\ \hline 217 \end{array}$$

مثال:

مکمل r و r-1 اعداد زیر را بدست آورید:

مکمل ۱۰

$(234)_{10} = 10^3 - 234 = 766$

مکمل ۹

$10^3 - 1 - 234 = 765$

مکمل ۱۰

$(234.68)_{10} = 10^3 - 234.68 = 765.32$

مکمل ۹

$10^3 - 10^{-2} - 234.68 = 1000 - 0.01 - 234.68 = 765.31$

مثال: مکمل 16 و 15 عدد 3EAF را در مبنای 16 بدست آورید.

* ساده‌تر است که ابتدا مکمل $(r-1)$ را بدست آوریم سپس آن را با یک "1" جمع کنیم.

$$\begin{array}{r} FFFF \\ -\ 3EAF \\ \hline C150 \end{array}$$

مکمل 15 → 15

مکمل 15 + 1 = مکمل 16

$$\begin{array}{r} C150 \\ +\ \ 1 \\ \hline C151 \end{array}$$

مکمل 16 → 16

بدست آوردن مکمل 2 در مبنای 2:

ازسمت راست شروع می‌کنیم تا زمانی که به اولین یک برسیم تمام اعداد را بدون تغییر می‌نویسیم بعد از اولین ۱ به هر عددی که رسیدیم عکس آن را می‌نویسیم اگر صفر بود یک و اگر یک بود صفر.

مثال:

$(1001010010)_2$

مکمل 2:

0110101110

بدست آوردن مکمل 1 در مبنای 2:

برای به دست آوردن مکمل ۱ عدد اولیه را عیناً عکس می‌کنیم. یعنی یک به صفر و صفر به یک تبدیل می‌شود.

مکمل ۱ ← 011010110

مکمل ۱ $(011001110)_2 = 100110001$

راه دیگر بدست آوردن مکمل ۲ در مبنای 2:

ابتدا مکمل یک عدد را بدست می‌آوریم سپس به سمت راست‌ترین بیت، عدد یک را اضافه می‌کنیم.

مکمل ۲ = مکمل ۱ + ۱

مثال:

مکمل 2 عدد $(011010110)_2$ را بدست آورید.

مکمل ۱ 100101001

\+ 1

مکمل ۲ 100101010

توجه شود که اگر عدد اعشاری بود یک به سمت راست‌ترین بیت اضافه می‌شود.

مثال:

مکمل 2 عدد $(10011/101)_2$ را بدست آورید:

$(10011/101)_2 \rightarrow$ مکمل ۱ = (01100/010)

مکمل ۲ ← مکمل ۱ + ۱ ⇒

01100/010

\+ 1

01100/011

تفریق به کمک مکمل گیری:

دو روش برای این عملیات وجود دارد که عبارتند از:

۱- استفاده از مکمل r :

رابطه (۴-۱) : $M - N = M + (\text{مکمل } r \text{ عدد } N)$

با استفاده از رابطه (۴-۱) می‌توانیم حاصل تفریق M-N را بدست آوریم حاصل بدست آمده دارای دو حالت می‌باشد:

الف: زمانی که حاصل دارای Carry باشد که در این حالت تنها Carry را حذف کرده عدد باقیمانده حاصل مورد نظر ما می‌باشد.

مثال:

حاصل تفریق زیر را بدست آورید:

$4897 \rightarrow M$

$- \underline{1439 \rightarrow N}$ ⇒ داریم

?

M

$+ [N]_r$ ⇒ بنابراین از N مکمل r می‌گیریم

?

4897

\+ 8561

C

Carry 13458

ب: زمانی که حاصل بدست آمده فاقد Carry باشد ، از آن مکمل r می‌گیریم. سپس یک منفی جلویش می‌گذاریم. عدد بدست آمده حاصل مورد نظر می‌باشد.

$$\begin{array}{r} 1439 \\ +\ 5103 \\ \hline 6542 \end{array}$$

بنابراین باید از N مکمل r بگیریم $\Rightarrow$

$$\begin{array}{r} M \\ +\ [N]_r \\ \hline ? \end{array}$$

$$\begin{array}{r} 1439 \\ -\ 4897 \\ \hline ? \end{array} \Rightarrow$$

$-3458 \Rightarrow$ مکمل r می‌گیریم $\longrightarrow$

۲ـ استفاده از مکمل r-1 :

رابطه (۵ ـ ۱):

$$M-N = M+ (\text{مکمل } r\text{-}1 \text{ عدد } N)$$

با استفاده از رابطه (۵ ـ ۱) می‌توانیم حاصل تفریق M-N را بدست آوریم، حاصل بدست آمده دارای دو حالت می‌باشد.

الف: زمانی که حاصل دارای carry باشد، carry را حذف می‌کنیم و حاصل را با عدد یک جمع می‌کنیم، عدد بدست آمده حاصل مورد نظر می‌باشد.

مثال:

حاصل تفریق زیر را بدست آورید:

$$\begin{array}{rl} 4897 & \rightarrow M \\ -1439 & \rightarrow N \\ \hline ? & \end{array}$$

$$\begin{array}{r} M \\ +\ [N]_{r-1} \\ \hline ? \end{array}$$

باید از N مکمل r-1 بگیریم $\Rightarrow$ داریم $\Rightarrow$

$$\Rightarrow [1439]_{r-1} = 8560 \Rightarrow \begin{array}{rr} & 4897 \\ & +\ 8560 \\ \hline \text{Carry} \rightarrow & 3457 \end{array}$$

حال carry را حذف میکنیم و حاصل را با عدد " 1 " جمع می‌کنیم:

$$\begin{array}{r} 3457 \\ +\ 1 \\ \hline 3458 \end{array}$$

حاصل تفریق M,N $\rightarrow$

ب: زمانی که حاصل بدست آمده فاقد carry باشد، از آن مکمل r-1 می‌گیریم، سپس یک منفی جلوی آن می‌گذاریم. عدد بدست آمده حاصل مورد نظر می‌باشد.

مثال:

حاصل تفریق را بدست آورید:

$$\begin{array}{rl} 1439 & \rightarrow M \\ -\ 4897 & \rightarrow N \\ \hline ? & \end{array} \Rightarrow$$

بنابراین باید از N مکمل $\begin{array}{r} M \\ +\ [N]_{r-1} \\ \hline ? \end{array}$: داریم

r-1 بگیریم.

پس باید از حاصل دوباره مکمل r-1 بگیریم ← و یک منفی جلوی آن قرار می‌دهیم:

$\Rightarrow [4897]_{r-1} = 3458$

$= -3458$ حاصل تفریق M, N

کدهای عددی

خصوصیات کدگذاری اعداد:

وزن دار بودن: به این معنا که با استفاده از این وزنها بتوانیم مقدار دهدهی را بدست آوریم.

خود مکمل بودن: کدی خود مکمل است که مکمل یک آن در دودویی با مکمل 9 آن در دهدهی برابر باشد اگر یک کد وزن‌دار بخواهد خود مکمل باشد باید جمع وزنهایش با هم 9 شود.

خود مکمل نیست. $9 \neq 16+10+8+4+2+1$ مجموع وزنها (101110)

انعکاسی بودن: کدهایی دارای خاصیت انعکاسی هستند که کد قبل و بعد تنها در یک بیت با هم تفاوت داشته باشند. مثلاً 000 و 001 و ... تنها در یک بیت اختلاف دارند.

4 کد نمونه که دارای خاصیت انعکاسی هستند.

000
001
101
100

چند روش کدگذاری

کد BCD: (Binary Coded Decimal)

BCD یک کد چهاربیتی بصورت زیر است که می‌تواند اعداد 0 تا 9 را به وجودآورد این کد یک کد وزن‌دار می باشد که وزنهای آن 1 2 4 8 می‌باشد. در اصل باتوجه به مفهوم این کد می‌توان دریافت که در این کد هر رقم یک عدد دهدهی باید به باینری در آید.

برای بدست آوردن مکمل 9 یک عدد از کد BCD، دو راه وجود دارد که عبارتند از:

راه اول: ابتدا از آن عدد مکمل یک گرفته سپس آن را با 1010 جمع می کنیم.

0000
0001
⋮
1001

مثال:

مکمل 9 عدد 2 در کد BCD را بدست آورید:

$$2 \rightarrow 0010 \longrightarrow \begin{array}{r} 1101 \\ +1010 \\ \hline 1\ 0111 \end{array} \leftarrow \text{Carry}$$

carry را حذف می کنیم جواب حاصل مکمل ۹ عدد 2 در BCD است.

راه دوم: عدد مورد نظر را با (0110) جمع می کنیم یعنی در اصل عدد مورد نظر را با 6 جمع می کنیم از عدد حاصل مکمل یک می گیریم جواب حاصل مکمل ۹ عدد مورد نظر است. اگر حاصل را با یک جمع کنیم به مکمل ۱۰ میرسیم.

مثال:

مکمل 9 و 10 عدد یک را در کد BCD بدست آورید:

0001

مکمل 9 =1000 → مکمل یک می گیریم → +0110

0111

مکمل 9 +1= مکمل 10 =1000+1=1001

کد Excess:

کدافزونی به صورت Excess –k است یعنی k واحد به عدد باینری اضافه می‌کند. مهمترین ویژگی این کدها این است که وزن‌دارنیستند .متداولترین کد افزونی Excess-3 می باشد. برای بدست آوردن یک کد Excess –3 باید عدد باینری را با 3 جمع کنیم و همینطور برای تبدیل کد Excess – 3 به باینری باید 3 واحد از آن کم کنیم.

از دیگر خصوصیات این کد خود مکمل بودن آن می‌باشد. یعنی مکمل یک آن در دودویی با مکمل 9 آن در دهدهی برابر می‌باشد.

Binary	Excess
0000	0011
0001	0100
0010	0101
0011	0110
0100	0111

مثال:

1011 یک Excess-3 است برای بردن آن به باینری باید 3واحد از آن را کم کنیم این عدد برابر 11 در مبنای ده است و 3 واحد که از آن کم شود برابر 8 می‌شود .یعنی عدد ۱۰۰۰ دربایتری.

1011
+0011
1000

کد Gray:

مهمترین خاصیت این کد انعکاسی بودن آن است مثلاً در دو بیتی آن داریم:

00 → 0
01 → 1
11 → 2
10 → 3
⇒ این اعداد مربوط به کد Gray است.

همانطور که ملاحظه می‌شود هر کد با کد بعدی تنها در یک بیت متفاوت است. این کد را می‌توان به کمک آینه ایجاد کرد.

فرض کنید می‌خواهیم از عدد صفر تا 7 را در کد Gray ایجاد کنیم. برای ایجاد آن سه بیت نیاز می‌باشد. بنابراین به دو آینه نیاز داریم.

(توجه داریم که اگر بخواهیم یک کد n بیتی Gray ایجاد کنیم باید n -1 آینه استفاده کنیم.)

در شروع فقط می‌دانیم که صفر Gray همان "0" و یک Gray همان "1" می‌باشد. حال با قرار دادن یک آینه زیر این دو رقم انعکاس زیر را خواهیم داشت:

دهدهی	Gray
0	0
1	1
2	0
3	1

پس از تولید انعکاس باید این قانون را در نظر بگیریم که سمت چپ هر عدد بالای آینه یک بیت صفر و پائین آینه یک بیت یک قرار می‌دهیم. بنابرایم خواهیم داشت:

دهدهی	Gray
0	0 0
1	0 1
2	1 1
3	1 0

حال برای ایجاد چهار کد بعدی باید یک بار دیگر از آینه استفاده کنیم بنابراین:

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

حال طبق همان قانون قبلی برای اعداد بالای آینه یک بیت صفر سمت چپ و برای اعداد پائین آینه یک بیت یک سمت چپ قرار می‌دهیم بنابراین در انتها خواهیم داشت:

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

اما همان‌طور که مشاهده می‌شود اگر بخواهیم کدهای بزرگتری را در Gray ایجاد کنیم کار بسیار دشوای را پیش رو خواهیم داشت بنابراین بهتر است از روش‌های زیر برای این کار استفاده کرد.

تبدیل عدد دودویی به معادل Gray

برای تبدیل عدد دو دویی به معادل Gray می‌توان از رابطه (۵ ـ ۱) استفاده کرد. همانطور که ملاحظه می‌شود پرارزش‌ترین بیت در هر دو کد یکی است $(g_n = b_n)$ برای بدست آوردن کدهای بعد هم همانطور که می‌بینیم باید هر بیت را با بیت کناری آن X-OR کنیم تا حاصل یعنی معادل Gray تولید شود.

$$(b_n b_{n-1} \ \dots \ b_1)_2 = (g_n g_{n-1} \ \dots \ g_1)_G$$

رابطه (۵-۱):

$$\begin{cases} g_n = b_n \\ g_k = b_k \oplus b_{k+1} \quad k = 1,\dots,n-1 \end{cases}$$

تبدیل کد Gray به معادل دودویی:

رابطه (۶-۱):

$$\begin{cases} g_n = b_n \\ b_k = \sum_{i=k}^{n} g_i \ (\text{mod} \ \ 2) \end{cases}$$

برای تبدیل کد Gray به معادل دودویی باید از رابطه (۶-۱) استفاده کرد. اما به بیان دیگر می‌توان به این شکل عمل کرد که:

سمت چپ‌ترین رقم که در هر دو کد یکی می‌باشد، برای بدست آوردن بیت بعدی باید دو بیت آخر کد Gray را با هم $\oplus$ XOR کنیم تا بیت دوم بدست آید و سپس حاصل را بابیت سوم کد Gray، X-OR می‌کنیم تا بیت سوم بدست آید و به همین شکل تا آخرین بیت.

مثال:

عدد $(1001)_G$ را به معادل باینری تبدیل کنید:

g_4	g_3	g_2	g_1
(1	0	0	1)$_G$
	↓	↓	↓
	$\oplus$	$\oplus$	$\oplus$
↗	↓↗	↓↗	↓
b_4	b_3	b_2	b_1
(1	1	1	0)$_2$

	کد Gray	کد دهدهی
	0000	0
	0001	1
$00 \rightarrow 0$	0011	2
	0010	3
	0110	4
	0111	5
$01 \rightarrow 0$	0101	6
	0100	7
	1100	8
	1101	9
$11 \rightarrow 2$	1111	10
	1110	11
	1010	12
	1011	13
$10 \rightarrow 3$	1001	14
	1000	15

مثال:

عدد $(1010)_2$ را به معادل Gray تبدیل کنید.

$$\begin{array}{cccc} b_4 & b_3 & b_2 & b_1 \\ (1 & 0 & 1 & 0)_2 \\ \downarrow & \searrow\downarrow & \searrow\downarrow & \searrow\downarrow \\ & \oplus & \oplus & \oplus \\ \downarrow & \downarrow & \downarrow & \downarrow \\ (1 & 1 & 1 & 1)_G \\ g_4 & g_3 & g_2 & g_1 \end{array}$$

تبدیل مبنای ۱۰ به کد BCD :

هر رقم مبنای ۱۰ معادل با یک عدد چهار بیتی در کد BCD می‌باشد. حال اگر بخواهیم یک عدد مبنای ۱۰ را که از چندین رقم تشکیل شده است را به کد BCD تبدیل کنیم باید تک تک ارقام را به حالت باینری چهار بیتی بنویسم. یعنی با استفاده از ۱ وزن ۱ ۲ ۴ ۸ آن رقم را به باینری تبدیل کنیم.

مثال:

عدد ۳۴۵ در مبنای ۱۰ را به معادل BCD تبدیل کنید.

$$\begin{array}{ccc} 3 & 4 & 5 \\ \swarrow & \downarrow & \searrow \\ 0011 & 0100 & 0101 \end{array} \Rightarrow (001101000101)_{BCD}$$

تبدیل کد BCD به مبنای ۱۰ :

برای این کار کافیست از سمت راست دسته‌های چهار بیتی را تشکیل دهیم و بالای هر دسته وزن ۱ ۲ ۴ ۸ را قرار دهیم و وزن هر دسته را بدست آوریم یعنی هر بیت را در وزن متناظرش ضرب کنیم سپس اعداد یک دسته را با هم جمع کنیم تا حاصل یعنی معادل دهدهی بدست آید. در ضمن باید توجه داشت که اگر آخرین دسته کمتر از چهار بیت در خود داشت بایدسمت چپ آن صفر قراردهیم تادسته به صورت کامل درآید.

مثال:

عدد $(1101111001)_{BCD}$ را به معادل دهدهی تبدیل کنید.

$$\begin{array}{ccc} 0011 & 0111 & 1001 \\ 8421 & 8421 & 8421 \\ 0+0+2+1 & 0+4+2+1 & 8+0+0+1 \\ \Downarrow & \Downarrow & \Downarrow \\ 3 & 7 & 9 \end{array} \Rightarrow (379)_{10}$$

تبدیل مبنای ۱۰ به کد Excess-3 :

برای انجام این تبدیل باید هر رقم را به معادل Ex-3 تبدیل کنیم یعنی اینکه ابتدا به کمک وزن اعداد هر رقم مبنای ۱۰ را به معادل باینری تبدیل می‌کنیم سپس ۰۰۱۱ را (یعنی 3) به معادل باینری اضافه می‌کنیم تا معادل Ex-3 بدست آید.

باید توجه داشت که در این نوع تبدیل هم باید از وزن ۱ ۲ ۴ ۸ استفاده شود.

مثال:

عدد ۳۵۷ در مبنای ۱۰ را به معادل Ex-3 تبدیل کنید.

3	5	7	
↙	↓	↘	
0011	0101	0111	← مرحله اول تبدیل هر رقم به معادل باینری
+ 0011	0011	0011	← مرحله دوم جمع با ۰۰۱۱
0110	1000	1010	$\Rightarrow (011010001010)_{Ex-3}$

تبدیل کد Ex-3 به مبنای ۱۰:

برای این کار ابتدا از سمت راست شروع به دسته‌بندی چهار بیتی می‌کنیم ،با استفاده از وزن ۸ ۴ ۲ ۱ برای هر دسته مقدار وزن آن را پیدا می‌کنیم سپس عدد بدست آمده را منهای ۳ می‌کنیم تا حاصل در مبنای ۱۰ بدست آید. باید توجه داشت که اگر آخرین دسته کمتر از چهار بیت بود سمت چپ آن،به تعداد بیتهای کمترصفر قرار می‌دهیم تا دسته کامل شود.

مثال:

عدد $(10110011100)_{Ex-3}$ را به معادل دهدهی تبدیل کنید.

0101	1001	1100	
8421	8421	8421	
0+4+0+1	8+0+0+1	8+4+0+0	
⇓	⇓	⇓	
5	9	12	← مرحله اول بدست آوردن وزن هر دسته با کمک ۸۴۲۱
−3	3	3	← مرحله دوم تفریق حاصل از 3 و بدست آوردن معادل دهدهی
2	6	9	$\Rightarrow (269)_{10}$

سایر کدهای وزن دار:

1 2 4 8 معمولترین وزنهاست که کد BCD را تشکیل می‌دهد ولی وزن‌های دیگری مانند 1 2 4 2 , 1 2 1 4 , ... وجود دارد. این وزنها در واقع قراردادی است و قانون خاصی ندارد. برخی از این کدها از جمله $\bar{1}\ \bar{2}\ 4\ 8$ کدی می‌باشد که قابلیت ساختن اعداد منفی را نیز دارد. علامت "-" روی برخی از وزن‌ها به معنی منفی بودن آن وزن می‌باشد در اصل در صورت استفاده از آن وزن مقدارش از وزن کل کم خواهد شد.

مثال:

عدد 1111 در کد $1\ 2\ \bar{4}\ 8$ چند است؟

$$1111 = (1\times 1) + (1\times 2) + (1\times(-4)) + (1\times 8) = 1+2+(-4)+8 = 7$$

مثال:

عدد 22 را با کد 1 2 4 8 16 بنویسید؟

در عدد 22 یک 16 وجود دارد پس زیر وزن 16 عدد یک می‌گذاریم سپس مقدار 16 را از 22 کم می‌کنیم. $22-16=6$

در عدد 6، مقدار 8 وجود ندارد پس زیر وزن 8 عدد صفر می‌گذاریم. اما در 6 یک 4 وجود دارد. زیر مقدار 4 عدد یک قرار می‌دهیم. بعد 4 را از 6 کم می‌کنیم. $6-4=2$. در عدد 2 یک 2 وجود دارد پس زیر وزن 2 هم یک قرار می دهیم و مقدار وزن 1 نیز صفر خواهد شد.

$$\begin{array}{r} 22 \\ -16 \\ \hline 6 \\ -4 \\ \hline 2 \\ -2 \\ \hline 0 \end{array} \qquad \begin{array}{ccccc} 16 & 8 & 4 & 2 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{array}$$

یکی از این کدها $\bar{1}\ 2\ \bar{4}\ 8$ می‌خواهیم ببینیم که اعداد در این کد چگونه ساخته می‌شود. کوچکترین عددی که در این کد ساخته می‌شود عدد 5 - می‌باشد که آن 0101 می‌باشد. یعنی:

$$(0\times 8)+(0\times(-4))+(1\times 2)+(0\times(-1))=10$$

حال فرض کنید بخواهیم عدد 5 را در این کد بسازیم به این صورت عمل می‌کنیم که:

به کمک اعداد مثبت نمی‌توان 5 را ساخت زیرا عدد 8 که بزرگتر از 5 است و عدد 2 که کوچکتر از 5 می‌باشد. بنابراین باید با بازی با وزن‌ها عدد 5 را باسازیم. ابتدا در زیر وزن 8 عدد یک را قرار می‌دهیم در این حالت با فرض صفر بودن سایر وزن‌ها به عدد 8 خواهیم رسید. (یعنی $\begin{matrix}8 & \bar{4} & 2 & \bar{1}\\ 1 & 0 & 0 & 0\end{matrix}=8$) بنابراین برای رسیدن به 5 باید سه واحد از 8 کم کنیم. و چون وزن 3- وجود ندارد مجبور هستیم از 4- کمک بگیریم. (یعنی $\begin{matrix}8 & \bar{4} & 2 & \bar{1}\\ 1 & 1 & 0 & 0\end{matrix}=4$) در این صورت به عدد 4 خواهیم رسید همان‌طور که ملاحظه می‌شود این عدد کوچکتر از عدد مورد نظر ما یعنی 5 می‌باشد پس باید به آن یک واحد اضافه کنیم اما می‌بینیم که وزن مثبت "1" وجود ندارد و فقط برای اضافه کردن می‌توان از 2+ استفاده کرد. با استفاده از این وزن حاصل برابر 6 می‌شود ($\begin{matrix}8 & \bar{4} & 2 & \bar{1}\\ 1 & 1 & 1 & 0\end{matrix}=6$) حال کافیست که تنها یک واحد از آن کم شود بنابراین که به صورت $\begin{matrix}8 & \bar{4} & 2 & \bar{1}\\ 1 & 1 & 1 & 1\end{matrix}=5$ در خواهد آمد.

حال با کمی تلاش می‌توان جدول زیر که مربوط به اعداد این کد می‌باشد را بدست آورد.

دهدهی	8	$\bar{4}$	2	$\bar{1}$	
-5	0	1	0	1	مجموعه اعداد منفی و صفر تولید شده توسط کد $\bar{1}$ 2 $\bar{4}$ 8
-4	0	1	0	0	
-3	0	1	1	1	
-2	0	1	1	0	
-1	0	0	0	1	
0	0	0	0	0	
1	0	0	1	1	مجموعهٔ اعداد مثبت
2	0	0	1	0	
3	1	1	0	1	
4	1	1	0	0	
5	1	1	1	1	
6	1	1	1	0	
7	1	0	0	1	
8	1	0	0	0	
9	1	0	1	1	
10	1	0	1	0	

کدهای تشخیص و تصحیح خطا:

اولین موضوع مورد بحث فاصله بین دو کد است فاصله دو کد عبارتست از میزان اختلاف بین دو کد یعنی تعداد تفاوت در صفرها و یک‌ها در جایگاههای یکسان مثلاً بین دو کد $I = 0010$ و $J = 1110$ فاصله چند است؟

$0010 \rightarrow I$

$\underline{1110} \rightarrow J$

$d(I, J) = 2 \Rightarrow$ 2 تا اختلاف وجود دارد.

زیرا اولین بیت از سمت راست ($I = 0, j = 0$)ودومین بیت($I = 1, j = 1$)باهم برابرند.پس اختلاف وجودندارد.اما دربیتهای سوم وچهارم مقدار i, j متفاوت است.پس این دوعدد(۲ کد) در ۲ بیت باهم اختلاف دارند.

مورد دیگر وزن یک کد است که با w نشان داده می‌شود مثلاً درمثال بالا $w(I)$ برابر است با تعداد یکهای موجود در آن کد :

$w(I) = 2$

مورد دیگر تعداد خطاهای قابل کشف است که با S نشان داده می‌شود.

T : تعداد خطاهای قابل تصحیح می‌باشد.

کد parity (توازن):

این کد،کد تشخیص خطاهاست parity دو نوع دارد : زوج و فرد.

پریتی زوج به این معناست که تعداد یک‌های داده باید زوج باشد.

پریتی فرد به این معناست که تعداد یک‌های داده باید فرد باشد.

پریتی مورد استفاده در بحث مورد نظر پریتی یک بیتی است که به آن افقی پریتی گفته می‌شود.

در ضمن parity ماتریسی نیز وجود دارد که به سطر وستون parity اضافه می‌کند.

فرض کنید اطلاعات پنج بیتی 10110 فرستاده می‌شود، اگر با پریتی فرد فرستاده شود چون تعداد یکها فرد است به آخر آن صفر اضافه می‌شود که درست باقی بماند ولی اگر با پریتی زوج فرستاده شود چون باید تعداد یکهایش زوج باشد به بیت آخر یک، یک اضافه می‌شود تا تعداد یک‌ها زوج شود.

$$10110 \begin{cases} \text{فرد} \rightarrow 10110\ 0 \\ \text{زوج} \rightarrow 10110\ 1 \end{cases}$$

یکی دیگر از کدهایی که برای عملیات تشخیص و تصحیح خطا به کار برده می‌شود که همینگ می‌باشد نحوهٔ کدگذاری در این روش کمی متفاوت با کد Parity است. حال در این قسمت به بررسی یک دادهٔ 4 بیتی می‌پردازیم که روی آن سه بیت کد همینگ نیز اضافه می‌شود.

کد همینگ

اگر یک کد ارسالی در همینگ به صورت $C_1C_2b_3C_4b_5b_6b_7$ باشد داریم:

اطلاعات اصلی $\rightarrow$ $b_3b_5b_6b_7$

Parity های اضافه شده در کد همینگ $\rightarrow$ $C_1C_2C_4$

$$C_1 = b_3 \oplus b_5 \oplus b_7$$
$$C_2 = b_3 \oplus b_6 \oplus b_7$$
$$C_4 = b_5 \oplus b_6 \oplus b_7$$

یک کدهمینگ با فاصله d قابلیت کشف S خطا $(S = d-1)$ و تصحیح T خطا $\left(T = \left\lfloor \frac{d-1}{2} \right\rfloor\right)$ را دارا است.

مثال:

کد همینگی را در نظر بگیرید که حداقل فاصله در آن 6 باشد تعداد خطای قابل تشخیص و تصحیح را بدست آورید.

$d = 6$

$$S = d-1 = 6-1 = 5$$
$$T = \left\lfloor \frac{d-1}{2} \right\rfloor = \left\lfloor \frac{6-1}{2} \right\rfloor = 2$$

مثال: اگر فرض کنیم بخواهیم دادهٔ 1011 را توسط همینگ کد کنیم به چه داده‌ای خواهیم رسید؟

حل:

همان‌طور که ملاحظه می‌شود $\left.\begin{matrix} b_3=1 \\ b_5=0 \\ b_6=1 \\ b_7=1 \end{matrix}\right\}$ می‌باشد. بنابراین داریم:

$$C_1 = b_3 \oplus b_5 \oplus b_7 = 1 \oplus 0 \oplus 1 = 0$$

$$C_2 = b_3 \oplus b_6 \oplus b_7 = 1 \oplus 1 \oplus 1 = 1$$

$$C_4 = b_5 \oplus b_6 \oplus b_7 = 0 \oplus 1 \oplus 1 = 0$$

بنابراین کد ایجاد شده به صورت زیر در می‌آید:

011011

کدهای Ascii :

برای نشان دادن کاراکترهای موجود در یک سیستم کامپیوتر می‌توان از یک نوع کدگذاری استفاده کرد که یک عدد را به آن کاراکتر تخصیص دهد به این نوع کدها، کدهای ASCII گفته می‌شود که توسط استانداردهای مختلف نمایش داده شده است.

به عنوان مثال برای نمایش A از عدد 1000001 استفاده می‌شود که معادل 65 می‌باشد. یا برای کاراکتر a از عدد 1100001 می‌توان استفاده کرد.

برای نمایش برخی کاراکترهای خاص نیز میتوان از کدهای ASCII استفاده کرد، مانند %(0100101) یا $(0100010).

نکته:

اگر به یک کد پریتی زوج اضافه کنیم حداقل اختلاف 2 بیت می‌باشد $(d=2)$ که میتوان یک خطا را تشخیص داد ولی نمی‌توان هیچ خطایی را تصحیح کرد.

نکته:

مزیت کد وزن‌دار 1 2 4 2 این است که عمل جمع هر دو عدد در این کد با مکمل گیری بیت به بیت انجام می‌شود این کد یک کد خود مکمل است. چون

$$2+4+2+1=9$$

نکته:

در جمع دو عدد باینری n بیتی که حاصل n بیتی داشته باشد برای بدست آوردن مقدار overflow از رابطه (۷-۱) استفاده می‌کنیم.

$$\begin{array}{l} x_7\, x_6 \ldots x_0 \\ \underline{y_7\, y_6 \cdots y_0} \\ p_7\, p_6 \cdots p_0 \end{array}$$

رابطه (۷-۱):

$V = x_7 y_7 \bar{p}_7 + \bar{x}_7 \bar{y}_7 p_7$ سرریز یا over flow

مجموعه مسائل فصل اول:

۱- تغییرات مبناهای زیر را حل کنید:

الف)

$(273)_{10} = (100010001)_2$

273	136	68	34	17	8	4	2	1
1	0	0	0	1	0	0	0	1

2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
256 128 64 32 16 8 4 2 1
$(1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1)_2$

273
−256
17
−16
1

ب)

$(3124)_{10} = (\quad)_2$

2048 1024 512 256 128 64 32 16 8 4 2 1
$(1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 0)_2$

3124
−2048
1076
−1024
52
−32
20
−16
4
−4
0

ج)

$(365)_{10} = (\quad)_8$

365 |8
45 |8
5
5

8^2 8^1 8^0
64 8 1
$(5\ 5\ 5)_8$

د)

64 32 16 8 4 2 1
$(1\ 1\ 0\ 0\ 0\ 1\ 0)_2 = (\quad)_{10}$

هـ)

8 1
$(4\ 7)_8 = (\quad)_{10}$
$(7\times1)+(4\times8)=(39)_{10}$

و)

256 16 1
$(1\ A\ F)_{16} = (431)_{10}$
↓ ↓
10 15

ی)

$(257)_8 = (\quad)_2$

$(257)_8 \rightarrow \begin{cases} 7 \rightarrow 111 \\ 5 \rightarrow 101 \\ 2 \rightarrow 010 \end{cases} \rightarrow (10101111)_2$

ن)

$(100111011101)_2 = (4735)_8$

۲- مکمل r و r – 1 اعداد زیر را بدست آورید.

$(25)_{10}$

مکمل 10: 100 − 25 = 75

مکمل 9: 99 − 25 = 74

$(347)_8$

مکمل 8: 1000 − 347 = 431

مکمل 7: 777 − 347 = 430

$(100100)_2$

مکمل 2: (011100)

مکمل 1: (011011)

۳- تبدیل کدهای زیر را انجام دهید.

$(011)_2 = (110)_{EX-3}$

$011 \rightarrow 3$

$+ \quad 3$

$6 \rightarrow 110$

$(11000101)_B = (10100111)_G$

$(10100111)_G = (11000101)_B$

۴- جمع و تفریق و ضرب‌های زیر را انجام دهید.

الف)

مبنای 16

		1	1	
	A	2	F	4
+	5	0	D	E
	F	3	D	2

مبنای 16

		15	20	
	9	~~16~~	~~4~~	20
	~~A~~	~~0~~	~~5~~	~~4~~
−	5	2	D	E
	4	D	7	6

مبنای 16

			4	4	
			9	8	
			2	F	E
	×		5	0	A
		1	D	E	C
+	E	B	6	0	0
	E	D	3	E	C

ب)

		2	1	1	
		1	1	0	1
+		1	1	1	0
+		0	1	1	1
1	0	0	0	1	0

توضیح:

اگر 4 عدد "1" در مبنای دو با هم جمع شوند از حاصل باید 2 بار مبنای 2 کم شود و 2 به عنوان Carry مرحله بعد تولید شود. پس خواهیم داشت:

اما چون در مبنای 2 رقم "2" وجود ندارد باید به جای آن معادلش یعنی 10 را قراردهیم.

2
+1
+1
+1
+1
100

مجموعه تست‌های فصل اول:

۱- کدام گزینه غلط است؟

۱) عدد 308 در مبنای شانزده برابر است با 788 در مبنای ده

۲) عدد 10111001 در مبنای دو برابر است با B9 در مبنای شانزده

۳) عدد 377 در مبنای هشت برابر است با 111111 در مبنای دو

۴) عدد 242 در مبنای ده برابر است با 242 به مرز (کد) BCD

۲- معادل عدد 19 – در مبنای ده به صورت مکمل دو (Two's complement) کدام است؟

۱) 101110

۲) 011110

۳) 101101

۴) 110101

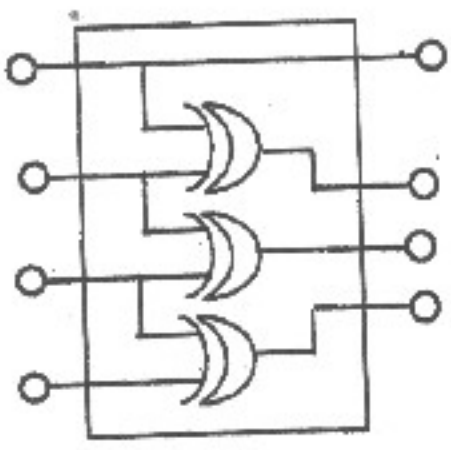
۳- معادل اکتال عدد $(A0F)_{16}$ کدام است؟

۱) 5117 ۲) 61171 ۳) 5017 ۴) 6017

۴- عدد 10010011 به صورت BCD بیان شده است. معادل این دو کدام است؟

۱) 10100010 ۲) 01011101 ۳) 11101100 ۴) 01101100

۵- شکل زیر کدام مبدل است؟



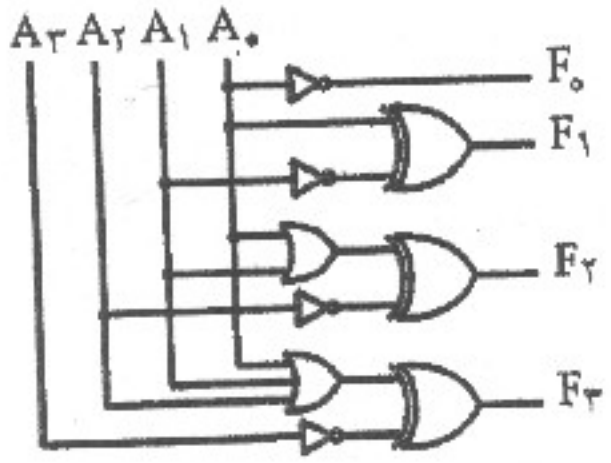
۱) کد سه افزون به کد باینری

۲) کد باینری به گری

۳) کد گری به باینری

۴) کد باینری به کد سه افزون

۶- با توجه به مدار زیر و به فرض این که اعداد a و f در سیستم مکمل دو باشند رابطه بین این دو عدد چهار بیتی $\left(f = F_3F_2F_1F_0 \ , \ a = A_3A_2A_1A_0\right)$ چیست؟



۱) f مکمل a است.

۲) f برابر با منفی a است.

۳) f برابر $a + 1$ است.

۴) f برابر $a - 1$ است.

فصل دوم

جبر بول، ساده‌سازی و اصول اولیه

اصول اولیه و قوانین جبر بول:

شماره اصل	اپراتور ضرب ×	اپراتور جمع +	
1	$a+\bar{a}b=a+b$	$a.(\bar{a}+b)=a.b$	شبه جذب
2	$a.1=a$	$a+0=a$	عضو خنثی
3	$a.b=b.a$	$a+b=b+a$	جابه‌جایی
4	$a.(b+c)=(a.b)+(a.c)$	$a+(b.c)=(a+b).(a+c)$	پخشی
5	$a.\bar{a}=0$	$a+\bar{a}=1$	عضو وارون
6	$a.a=a$	$a+a=a$	خود توانی
7	$\overline{a+b}=\bar{a}.\bar{b}$	$a+ab=a$	جذب
8	$\overline{a.b}=\bar{a}+\bar{b}$	$\overline{a+b}=\bar{a}.\bar{b}$	دمورگان

انواع جملات در جبربول

مجموع حاصلضرب‌ها:
sop : sum of products
ضرب حاصل جمع‌ها:
pos : product of sums

به عنوان مثال تابع f_1 تشکیل شده از sopها می‌باشد.

$$f_1(A,B,C)=A'B+B'C+AC'+ABC$$

تابع f_2 تشکیل شده از posها می‌باشد.

$$f_2(A,B,C)=(A'+B).(B'+C).(A'+B'+C')$$

فرمهای استاندارد:

minterm برای هر تابع n متغیره،اگر عبارت ضرب شامل تمام n متغیر باشد و هر متغیر بصورت مکمل یا غیر مکمل (a,a') فقط یکبار در عبارت ظاهر شوند عبارت ضرب را minterm گویند. هر کدام از n متغیر استفاده شده در این مینترم نشانگر یا یک بودن متغیر می‌باشد. به این صورت که اگر در یک مینترم متغیری خودش ظاهر شود نشان دهندهٔ یک بودن آن متغیر است و اگر مکملش ظاهر شود نشانهٔ صفر بودن آن می‌باشد.

$$F(A,B,C)=\Sigma(0,1,5,7)$$

$0 \rightarrow 000 \rightarrow A'B'C'$ minterm شماره صفر

$1 \rightarrow 001 \rightarrow A'B'C$ minterm شماره یک

$5 \rightarrow 101 \rightarrow AB'C$ minterm شماره 5

$7 \rightarrow 111 \rightarrow ABC$ minterm شماره 7

$$F(A,B,C)=A'B'C'+A'B'C+AB'C+ABC$$

اگر n متغیر داشته باشیم 2^n minterm داریم.

Maxterm: برای هر تابع n متغیره اگر عبارت جمع شامل تمام n متغیر باشد و هر متغیر بصورت مکمل یا غیر مکمل (a, a') فقط یکبار در عبارت ظاهر شوند عبارت جمع راmaxterm گویند برای یک تابع n متغیر 2^n تا maxterm داریم.

$$F(A,B,C) = \Pi(2,3,4,6)$$

$$2 \rightarrow 010 \rightarrow A + B' + C$$

$$3 \rightarrow 011 \rightarrow A + B' + C'$$

$$4 \rightarrow 100 \rightarrow A' + B + C$$

$$6 \rightarrow 110 \rightarrow A' + B' + C$$

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

در حالت کلی بدون در نظر گرفتن خروجی، 2^n یعنی 8 تا minterm و 8 تا maxterm داریم:

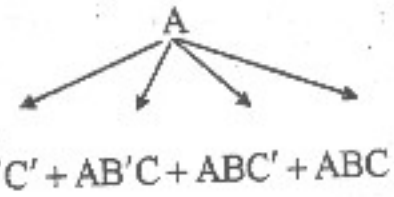
$$\underbrace{A'B'C' + A'B'C + \ \ldots\ + ABC}_{8 \text{ تا}}$$

نکته: اگر تعداد یک‌ها و صفرهای خروجی برابر بود minterm را می‌نویسیم اگر تعداد صفرها کمتر بود maxterm را می‌نویسیم اگر تعداد یک‌ها کمتر بود minterm را می‌نویسیم.

تبدیل SOP و POS به فرمهای استاندارد (Minterm و Maxterm):

اگر بخواهیم در یک تابع (f) که از n متغیر تشکیل شده است یک sop یا یک pos که شامل m متغیر می باشد $(m < n)$ را به فرم استاندارد تبدیل کنیم، باید تمامی ترکیبات مختلف (n-m) متغیر را که در آن sop یا pos کم دارد را اضافه کنیم. تفاوت sop و pos با minterm و maxterm در این است که در minterm و maxterm باید در هر جمله تمام متغیرها ظاهر شوند ولی در sop و pos این گونه نیست). یعنی یک sop یا pos تبدیل به 2^{n-m} ، minterm یا maxterm خواهد شد.

فرض می‌کنیم در تابع f که دارای سه متغیر A، B و C می‌باشد، A یک POS و یا یک SOP است ولی Minterm یا Maxterm نیست زیرا B و C را ندارد. حال برای اینکه بخواهیم آن را به فرم استانداردر آوریم داریم:



$$AB'C' + AB'C + ABC' + ABC$$

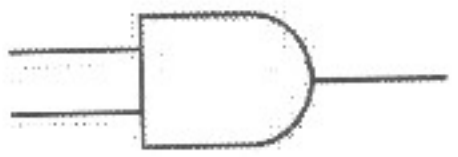
ترکیبات مختلف از n- m متغیر باقی مانده	BC	
	0 0	B'C'
	0 1	B'C
	1 0	BC'
	1 1	BC

اگر در تبدیل آن به فرم استاندارد حالت‌های شبیه به هم دیده شده باید یکی را حذف کرد.

تذکر: ترتیب قرار گرفتن متغیرها باید حفظ شود یعنی اگردر تابع $f(A, B, C)$ می‌خواهیم یک sop یا pos را تبدیل به حالت استاندارد کنیم و متغیرهای D و B را کم داشت پس از تولید minterm یا maxterm دقت می‌کنیم که ترتیب D و C و B و A حفظ شود.

گیت‌های منطقی:

AND: خروجی یک AND تنها در زمانی یک می‌باشد که هر دوورودی مقدار یک داشته باشند.



A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

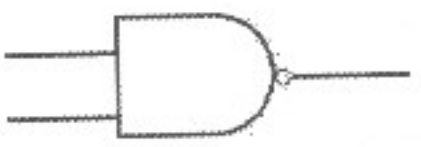
$f = A.B$

OR: خروجی یک OR تنها زمانی صفر است که هر دو ورودی صفر باشد.

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

$f = A + B$

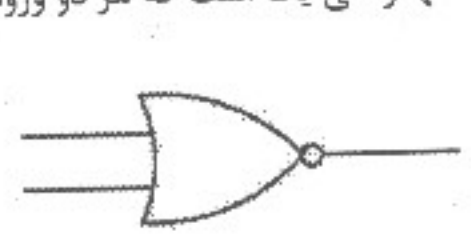
NAND: خروجی یک NAND تنها زمانی صفر است که هر دو ورودی یک باشد.



A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

$f = A' + B'$

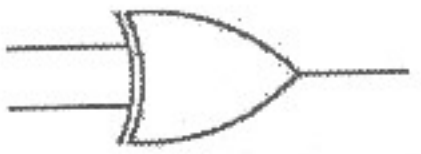
NOR: خروجی یک NOR تنها زمانی یک است که هر دو ورودی صفر باشد.



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

$f = A'B'$

XOR: خروجی XOR تنها زمانی یک است که ورودیها با یکدیگر متفاوت باشد.



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

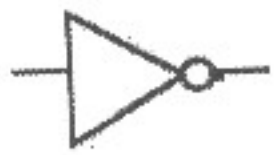
$f = A \oplus B = A'B + AB'$

X-NOR: خروجی X-NOR زمانی یک است که ورودیها هر دو یکسان باشند.

A	B	X-NOR
0	0	1
0	1	0
1	0	0
1	1	1

$f = A \odot B = A'B' + AB$

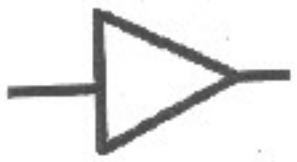
Not: ورودی را مکمل می‌کند. یعنی صفر را به یک و یک را به صفر تبدیل می‌کند.



A	NOT
0	1
1	0

$f = A'$

Buffer خروجی همان ورودی است.



A	Buffer
0	0
1	1

$f = A$

جداول درستی:

A	B	AND	OR	NAND	NOR	XOR	X-NOR	not A	Buffer A
0	0	0	0	1	1	0	1	1	0
0	1	0	1	1	0	1	0	1	0
1	0	0	1	1	0	1	0	0	1
1	1	1	1	0	0	0	1	0	1

منطق مثبت ومنفی:

در منطق مثبت High به معنی یک است و low به معنی صفر منطقی است.

در منطق منفی High به عنوان صفر منطقی است و low به عنوان یک منطقی است.

به عنوان مثال جدول منطق مثبت و منفی گیت AND را با هم مقایسه می کنیم.

A	B	AND
H	H	H
H	L	H
L	H	H
L	L	L

منطق منفی ⇒ و

A	B	AND
L	L	L
L	H	L
H	L	L
H	H	H

⇒ منطق مثبت

همانطور که ملاحظه می‌شود می توان دریافت که جدول AND در منطق منفی همان جدول AND در منطق مثبت است با این تفاوت که جای H و L با یکدیگر عوض شده است.

نحوهٔ ایجاد جدول درستی:

جدول درستی جدولی است که تمامی حالات ممکن برای n متغیر موجود در تابع را تولید می‌کند. این جدول از دو قسمت ورودی و خروجی تشکیل شده است که قسمت ورودی تعداد ردیف‌های جدول را نیز مشخص می‌کند.

برای ایجاد یک جدول درستی برای یک تابع n متغیره باید 2^n ردیف به وجود آورد. هر کدام از این ردیف‌ها یک شمارهٔ مخصوص به خود را نیز دارند. شمارهٔ ردیف‌های جدول درستی از صفر شروع می‌شود و تا $2^n - 1$ نیز ادامه پیدا می‌کند.

در اصل این شمارهٔ ردیف‌ها شماره‌های مربوط به mintermها و maxtermها را نیز مشخص می‌کندو در ضمن با استفاده از همین شمارهٔ ردیف‌ها می‌توان حالت‌های متفاوت برای n متغیر را ایجاد کرد. به عنوان مثال اگر یک تابع 5 متغیره متشکل از متغیرهای A , B , C , D , E داشته باشیم و بخواهیم در ستون ورودی‌ها و ردیف 5 مقدار بنویسیم داریم:

16 8 4 2 1
A B C D E
0 0 1 0 1

ردیف	a	minterm	maxterm
0	0	a′	a
1	1	a	a

جدول درستی یک متغیره

ورودی

ردیف	a b	minterm	maxterm
0	0 0	a′b′	a + b
1	0 1	a′b	a + b′
2	1 0	ab′	a′ + b
3	1 1	ab	a′ + b′

جدول درستی 2 متغیره

ورودی

ردیف	a b c	minterm	maxterm
0	0 0 0	a′b′c′	a + b + c
1	0 0 1	a′b′c	a + b + c′
2	0 1 0	a′bc′	a + b′ + c
3	0 1 1	a′bc	a + b′ + c′
4	1 0 0	ab′c′	a′ + b + c
5	1 0 1	ab′c	a′ + b + c′
6	1 1 0	abc′	a′ + b′ + c
7	1 1 1	abc	a′ + b′ + c′

جدول درستی سه متغیره

نکته: در جدول درستی در ستون خروجی آنهایی که مقدارشان یک است مشخص‌کننده تابع خروجی براساس minterm هاست و آنهاییکه مقدارشان صفر است مشخص کننده تابع خروجی براساس maxterm هاست.

A	B	C	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$f(A, B, C) = \Sigma(0,1,5,7) = \Pi(2,3,4,6)$

مثال: جدول درستی طراحی کنید که مربوط به یک میز رأی گیری باشد. این میز رأی گیری 3 نفره می‌باشد به این صورت که هر سه نفر قدرت رأی مساوی دارند. این میز دارای یک چراغ است و چراغ زمانی روشن می‌شود که حداقل دو نفر از رأی دهندگان جواب مثبت داده باشند. (پذیرفته شدن طرح باعث روشن شدن چراغ و یک شدن خروجی می‌باشد.)

رأی دهندگان

ردیف	a b c	چراغ f
0	0 0 0	0
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	1
6	1 1 0	1
7	1 1 1	1

تابع به صورت minterm
$$\begin{cases} f(a,b,c) = \sum(3,5,6,7,) \\ f(a,b,c) = a'bc + ab'c + abc' + abc \end{cases}$$

یا

$$\begin{cases} f(a,b,c) = \Pi(0,1,2,4) \\ f(a,b,c) = (a+b+c)\times(a+b+c')\times(a+b'+c)\times(a'+b+c) \end{cases}$$

مثال: جدول درستی مربوط به یک میز رأی گیری چهار نفره را که دارای دو چراغ سبز و آبی است را طراحی کنید. چراغ آبی زمانی روشن می‌شود که نفر دوم و سوم جواب مثبت داده باشند و چراغ سبز زمانی روشن می‌شود که حداقل سه نفرجواب مثبت داده باشند. تابع خروجی مربوط به چراغ سبز و آبی را به صورت minterm و maxterm بنویسید.

	نفر چهارم نفر اول	چراغ آبی	چراغ سبز
ردیف	a b c d	B	G
0	0 0 0 0	0	0
1	0 0 0 1	0	0
2	0 0 1 0	0	0
3	0 0 1 1	0	0
4	0 1 0 0	0	0
5	0 1 0 1	0	0
6	0 1 1 0	1	0
7	0 1 1 1	1	1
8	1 0 0 0	0	0
9	1 0 0 1	0	0
10	1 0 1 0	0	0
11	1 0 1 1	0	1
12	1 1 0 0	0	0
13	1 1 0 1	0	1
14	1 1 1 0	1	1
15	1 1 1 1	1	1

$B(a,b,c,d) = \sum(6,7,14,15) = a'bcd' + a'bcd + abcd' + abcd$

یا

$B(a,b,c,d) = \Pi(0,1,2,3,4,5,8,9,10,11,12,13)$

$G(a,b,c,d) = \sum(7,11,13,14,15) = a'bcd + ab'cd + abc'd + abcd' + abcd$

یا

$G(a,b,c,d) = \Pi(0,1,2,3,4,5,6,7,8,9,10,12)$

مثال: جدول درستی طراحی کنید یک عدد سه بیتی ورودی را به توان 2 برساند.

حل:

در این گونه مسائل باید ابتدا بزرگترین عدد ورودی را در نظر بگیریم و آن را به توان 2 برسانیم. در این مساله 7^2 برابر 49 می‌باشد حال باید 49 را به صورت باینری در آوریم تا بفهمیم که خروجی باید چند بیتی باشد.

ردیف	x	y	z	32 a	16 b	8 c	4 d	2 e	1 f
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	1
2	0	1	0	0	0	0	1	0	0
3	0	1	1	0	0	1	0	0	1
4	1	0	0	0	1	0	0	0	0
5	1	0	1	0	1	1	0	0	1
6	1	1	0	1	0	0	1	0	0
7	1	1	1	1	1	0	0	0	1

$a(x,y,z) = \sum(6,7)$ یا $\Pi(0,1,2,3,4,5)$

$b(x,y,z) = \sum(4,5,7)$ یا $\Pi(0,1,2,3,6)$

$c(x,y,z) = \sum(3,5)$ یا $\Pi(0,1,2,4,6,7)$

$d(x,y,z) = \sum(2,6)$ یا $\Pi(0,1,3,4,5,7)$

$e(x,y,z) = \sum(\quad)$ یا $\Pi(0,1,2,3,4,5,6,7)$

$f(x,y,z) = \sum(1,3,5,7)$ یا $\Pi(0,2,4,6)$

به دست آوردن توابع از روی جدول درستی

برای نوشتن توابع از روی جدول درستی ابتدا ستون خروجی را در نظر می‌گیریم. سپس با توجه به اینکه می‌خواهیم تابع را به صورت minterm یا maxterm بنویسیم، به یک یا صفر بودن خروجی توجه می‌کنیم.

اگر بخواهیم تابع را به صورت minterm بنویسیم باید یک‌های ستون خروجی را در نظر بگیریم. و minterm متناظر با هر، یک، خروجی را بنویسیم. برای این کار باید در نظر داشته باشیم که اگر در ستون ورودی‌ها مقدار هر متغیر یک بود باید خودش را بنوسیسم و اگر صفر بود باید مکملش را بنویسیم. (به عنوان مثال مقدار A در ردیف ششم یک است و B مقدارش صفر و C دارای ارزش یک می‌باشد بنابراین minterm آن به شکل AB'C در می‌آید.) پس از نوشتن تمامی minterm‌ها باید آنها را با هم جمع کنیم تا تابع بدست آید.

حال اگر بخواهیم تابع را به صورت maxterm بنویسیم باید صفرهای ستون خروجی را در نظر بگیریم و maxtermهای متناظر با هر، یک، از ردیفهای خروجی با ارزش صفر را بنویسیم. برای این کار باید در نظر داشت که اگر در ستون ورودی ها مقدار هر متغیر یک بود باید مکمل آن متغیر را بنویسیم و اگر صفر بود باید خود آن متغیر را بنویسیم. (به عنوان مثال در ردیف پنجم جدول درستی می‌بینیم که مقدار A یک می باشد و B صفر را به خود اختصاص داده و C نیز ارزش صفر دارد بنابراین maxterm مربوط به این ردیف برابر $A'+B+C$ می‌باشد.) پس از نوشتن تمامی maxtermها باید آنها را در هم ضرب کنیم تا تابع بدست آید.

$$x \begin{cases} 0 \to x' \\ 1 \to x \end{cases} \quad \text{min term}$$

$$x \begin{cases} 0 \to x \\ 1 \to x' \end{cases} \quad \text{max term}$$

	ورودی			خروجی	
	A	B	C	Z	
0	0	0	0	0	$\to A+B+C$
1	0	0	1	0	$\to A+B+C'$
2	0	1	0	0	$\to A+B'+C$
3	0	1	1	1	$\to A'BC$
4	1	0	0	0	$\to A'+B+C$
5	1	0	1	1	$\to AB'C$
6	1	1	0	1	$\to ABC'$
7	1	1	1	1	$\to ABC$

$$\Rightarrow Z(A,b,c) = \Sigma(3,5,6,7)$$
$$Z = A'BC + AB'C + ABC' + ABC$$

$$\Rightarrow Z(A,b,c) = \Pi(0,1,2,4)$$
$$Z = (A+B+C)\times(A+B+C')\times(A+B'+C)\times (A'+B+C)$$

مثال:

جدول درستی طراحی کنید که ۳ ورودی داشته باشد و ۴ خروجی به شکلی که خروجی ها از جمع ورودی با عدد ۳ بدست آمده باشد. سپس تابع هر کدام از خروجی ها را بدست آورید.

	ورودی‌ها			خروجی‌ها			
	A	B	C	x	y	w	z
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	1
3	0	1	1	0	1	1	0
4	1	0	0	0	1	1	1
5	1	0	1	1	0	0	0
6	1	1	0	1	0	0	1
7	1	1	1	1	0	1	0

$x = AB'C + ABC' + ABC$

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

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

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

تقدم عملگرها

منظور از تقدم این است که ارجحیت با کدام عملگر می‌باشد. با توجه به دانش قبلی که در ریاضیات داشتیم می‌دانیم که بالاترین تقدم با اپراتور پرانتز می‌باشد یعنی در صورت وجود پرانتز باید ابتدا داخل آن محاسبه شود سپس به بقیه عملگرها رسیدگی کنیم.

۱) پرانتز

۲) not (x′)

۳) and

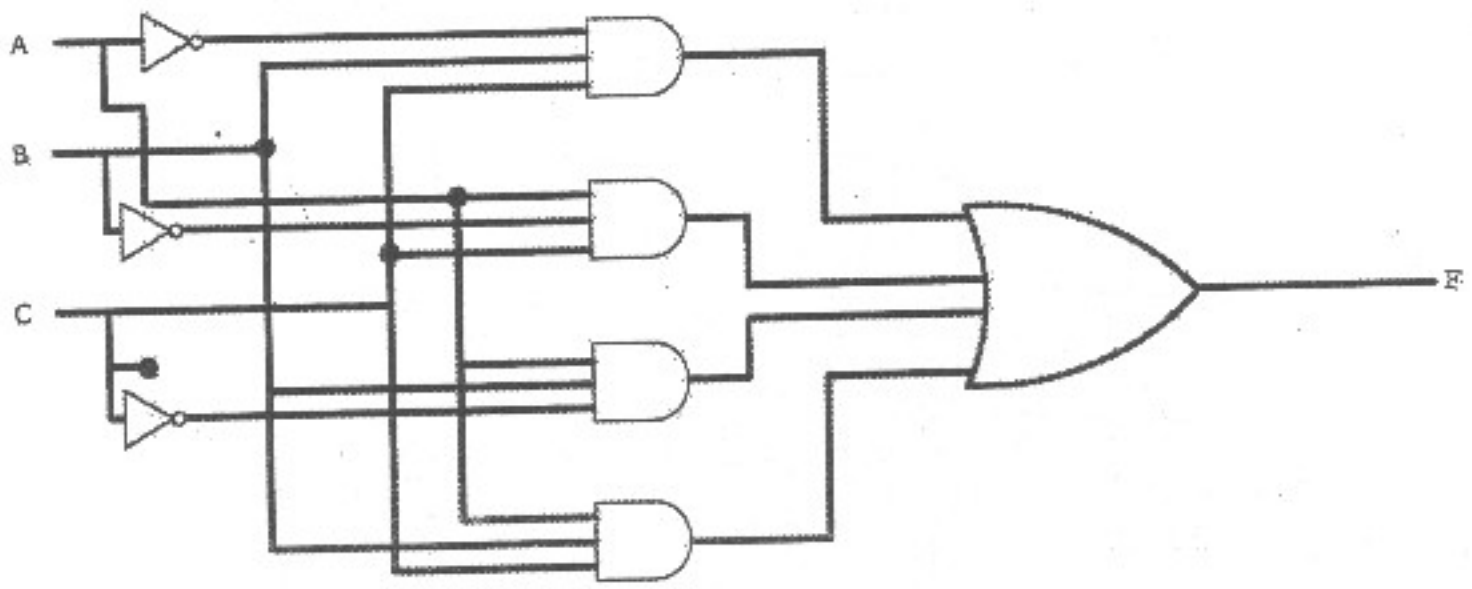
۴) or

پیاده سازی توابع بول باگیت‌ها

برای پیاده سازی توابع به کمک گیتها باید ابتدا با استفاده از جدول درستی که در اختیار داریم تابع را بدست آوریم. سپس با توجه به شکل تابع بدست آمده آن را به کمک گیتهای مورد نیاز پیاده سازی کنیم. البته در ادامه خواهیم دید که به کمک روشهای مختلف باید ابتدا تابع را ساده کنیم و سپس آن را پیاده سازی کنیم. به عنوان مثال اگر بخواهیم از روی جدول درستی زیر عمل پیاده سازی را انجام دهیم. باید ابتدا تابع را به فرم minterm‌هایش (یا maxterm‌هایش) در آوریم سپس به کمک گیتهای AND، OR، NOT به کمک ورودی‌های a , b , c می‌توان عمل پیاده سازی را به فرم زیر انجام دهیم.

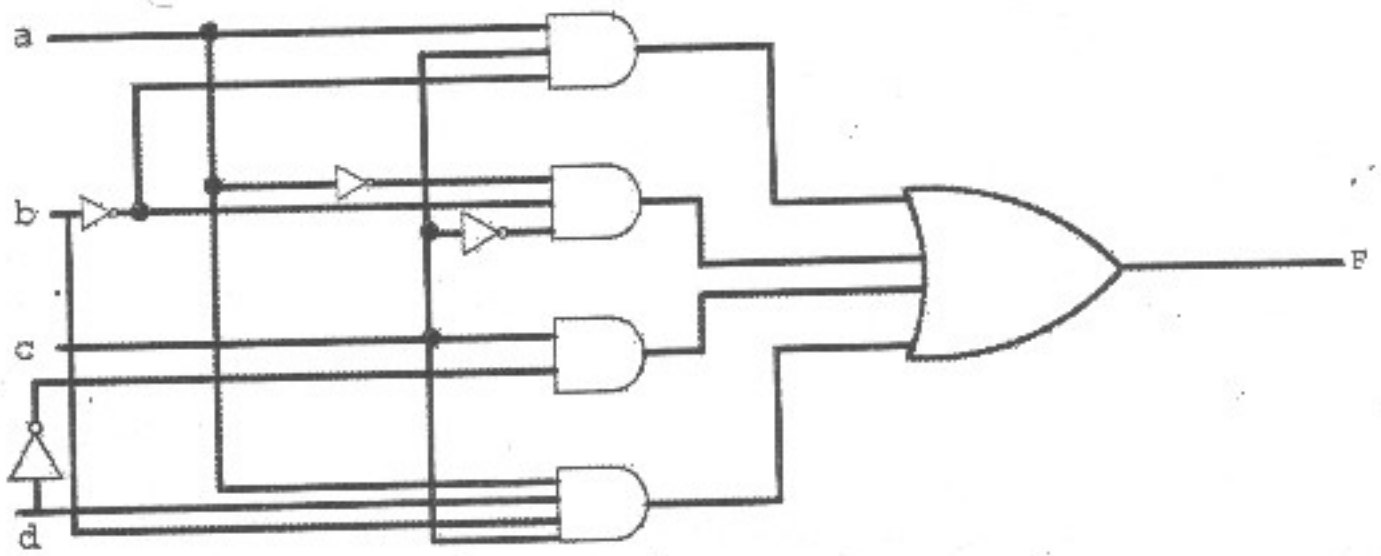
	a	b	c	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

$F(a,b,c) = \Sigma(3,5,6,7) = a'bc + ab'c + abc' + abc$



مثال:
تابع زیر را به کمک گیتها پیاده سازی کنید.

$$F(a,b,c) = ab'c + a'b'c' + cd' + abcd$$



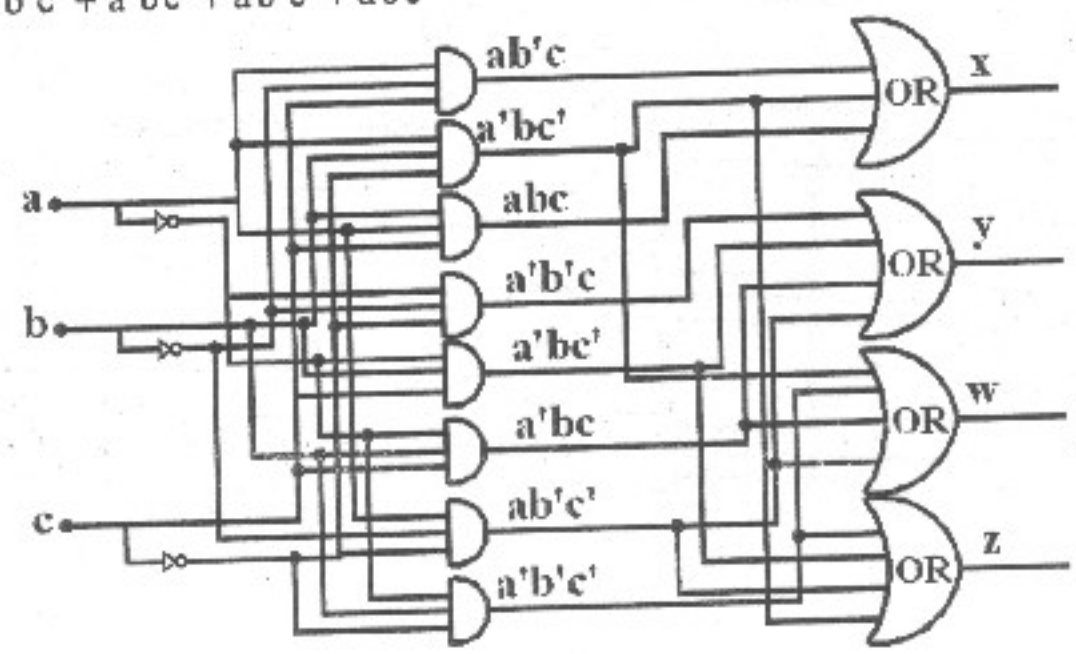
مثال: مداری طراحی و پیاده‌سازی کنید که دارای سه ورودی باشد و خروجی معادل 3 – Ex ورودی باشد.

$2^n = 2^3 = 8$

(با استفاده از جدول کارنو ساده کنید)

	4 a	2 b	1 c	x	y	w	z
0	0	0	0	0	0	1	1
1	0	0	1	0	1	0	0
2	0	1	0	0	1	0	1
3	0	1	1	0	1	1	0
4	1	0	0	0	1	1	1
5	1	0	1	1	0	0	0
6	1	1	0	1	0	0	1
7	1	1	1	1	0	1	0

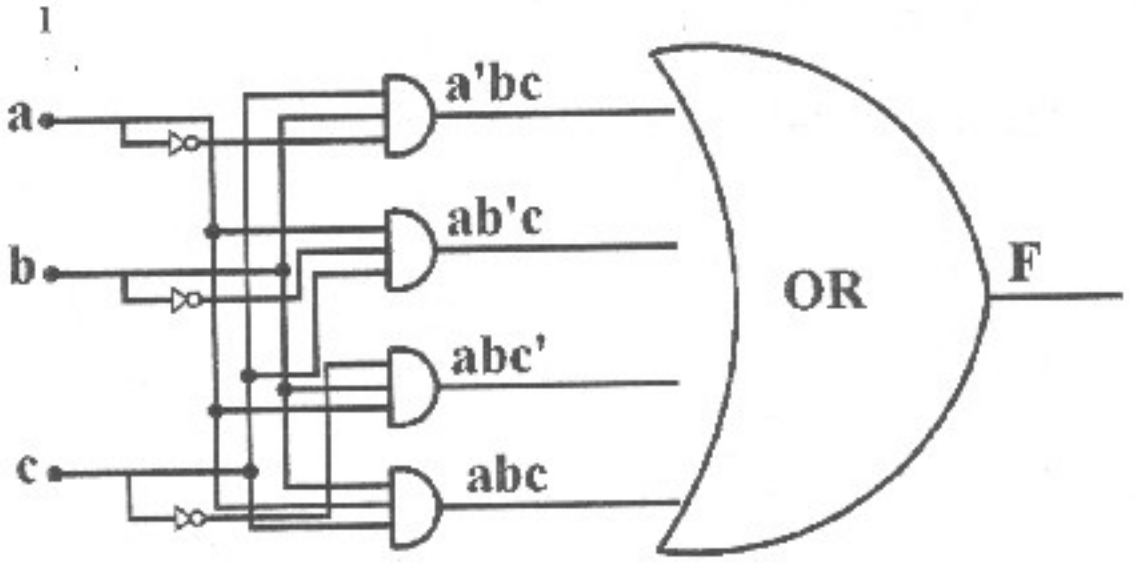
$$x = \sum(5,6,7) = ab'c + abc' + abc$$
$$y = \sum(1,2,3,4) = a'b'c + a'bc' + a'bc + ab'c'$$
$$w = \sum(0,3,4,7) = a'b'c' + a'bc + ab'c' + abc$$
$$z = \sum(0,2,4,6) = a'b'c' + a'bc' + ab'c' + abc'$$



مثال : به کمک مدارهای منطقی میز رأی‌گیری سه نفره‌ای را طراحی کنید که دارای یک چراغ باشد. این چراغ در صورتی روشن می‌شود که یک طرح پذیرفته شود در غیر اینصورت خاموش خواهد بود. در صورتی یک طرح مورد قبول وارد می‌شود که حداقل دو نفر از سه رأی‌دهنده موجود جواب مثبت بدهند (جواب مثبت یعنی اینکه یک باشد و جواب منفی یعنی صفر باشد) مدار را طراحی و پیاده‌سازی کنید.

	a	b	c	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

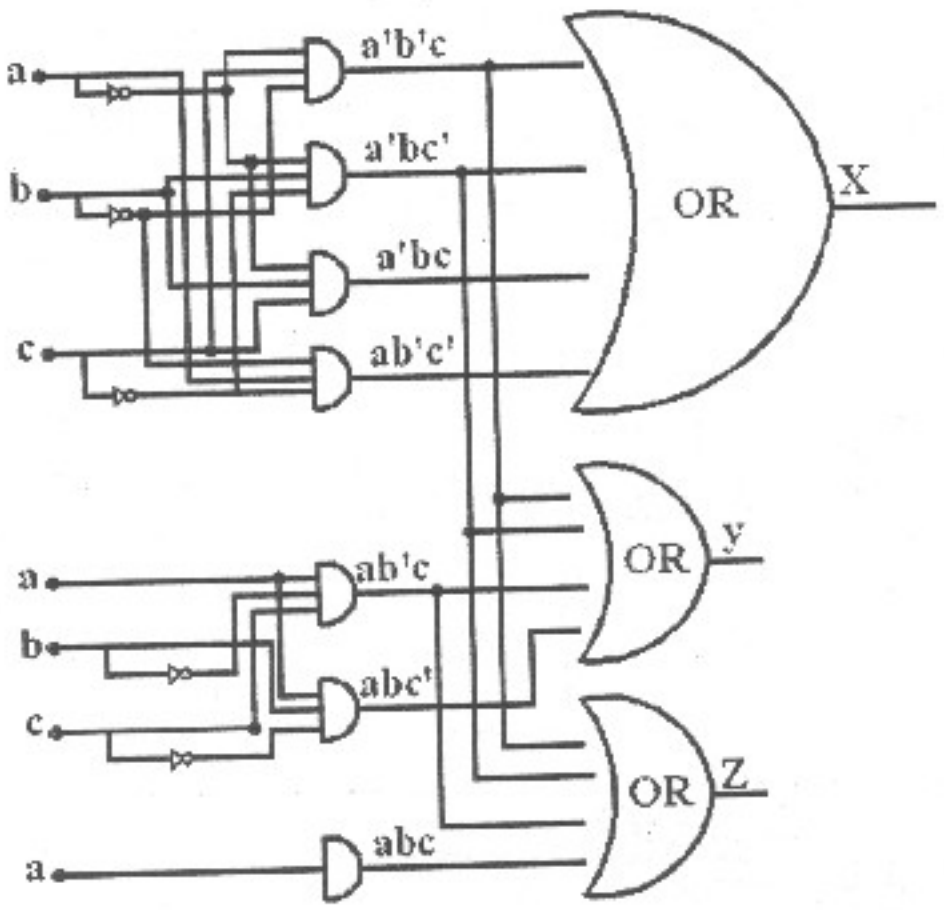
$$F = \sum (3 , 5 , 6 , 7) = a'bc + ab'c + abc' + abc$$



مثال : مداری طراحی و پیاده‌سازی کنید که ورودی آن یک عدد سه بیتی باشد و خروجی آن مکمل ۲ ورودی. (مکمل ۲ یعنی به اولین یک که رسیدیم خودش را نوشته و بقیه را عکس می‌کنیم.)

	a	b	c	x	y	z
0	0	0	0	0	0	0
1	0	0	1	1	1	1
2	0	1	0	1	1	0
3	0	1	1	1	0	1
4	1	0	0	1	0	0
5	1	0	1	0	1	1
6	1	1	0	0	1	0
7	1	1	1	0	0	1

$$x = \sum (1 , 2 , 3 , 4) = a'b'c + a'bc' + a'bc + ab'c'$$
$$y = \sum (1 , 2 , 5 , 6) = a'b'c + a'bc' + ab'c + abc'$$
$$z = \sum (1 , 3 , 5 , 7) = a'b'c + a'bc + ab'c + abc$$



مکمل گیری از توابع:

$x' \rightarrow x$
$x \rightarrow x'$
$1 \rightarrow 0$
$0 \rightarrow 1$
$\times \rightarrow +$
$+ \rightarrow \times$

برای مکمل گیری باید به جای 1 صفر و به جای صفر 1 قرار می‌دهیم و به جای ضرب، جمع و به جای جمع هم ضرب قرار می‌دهیم

ساده سازی با استفاده از جدول کارنو:

مرحله اول: باید تمام مینترم‌ها را به صورت کامل درآورد.

مرحله دوم: تعیین نمودن تعداد خانه‌های جدول کارنو با استفاده از فرمول 2^n که n در آن تعداد ورودی‌ها می‌باشد.

مرحله سوم: باید قسمت بندی ورودی‌ها را روی جدول انتخاب کنیم مهمترین نکته در این مرحله وجود همپوشانی برای ورودی‌ها است.

مرحله چهارم: باید جدول را با استفاده از مینترم‌ها و قرار دادن یک‌ها در خانه‌های جدول پر کنیم.

مرحله پنجم: ساده کردن جدول که خود شامل دو قسمت است:

قسمت اول: پیدا کردن همسایه‌ها (همسایه به خانه‌هایی گفته می‌شود که حداقل یک ضلع مشترک داشته باشند)

قسمت دوم: باید ناحیه همسایه‌ها را بخوانیم و تشخیص دهیم محدوده مورد نظر کجا قرار دارد.

* همسایه‌ها باید بزرگترین مضرب 2 باشند. (2^n).

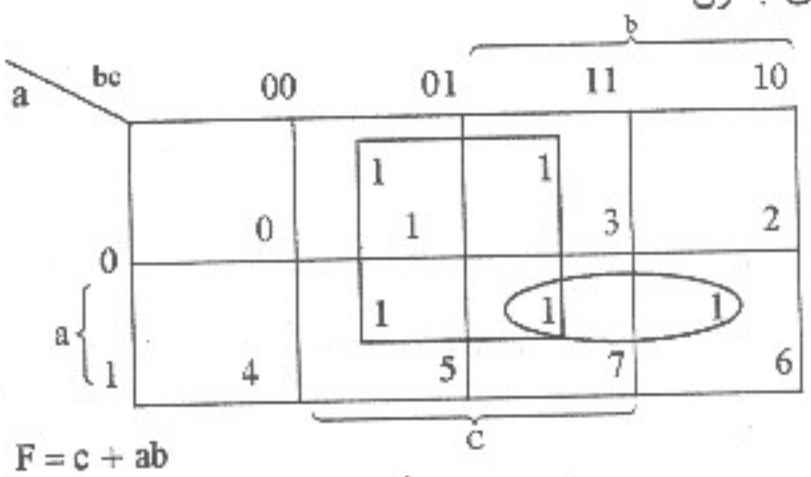
مثال:

با توجه به درستی زیر تابع f را به کمک جدول کارنو ساده کنید.

	a	b	c	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

تعداد خانه‌های جدول $= 2^3 = 8$

$F = a'bc + ab'c + abc' + abc$



$F = c + ab$

$F = a'bc + ab'c + abc' + a'b'c + abc$

مثال:

تابع F را به کمک جدول کارنو ساده کنید:

$F = (a,b,c,d) = \Sigma(1,2,3,4,7,11,12,14,15)$

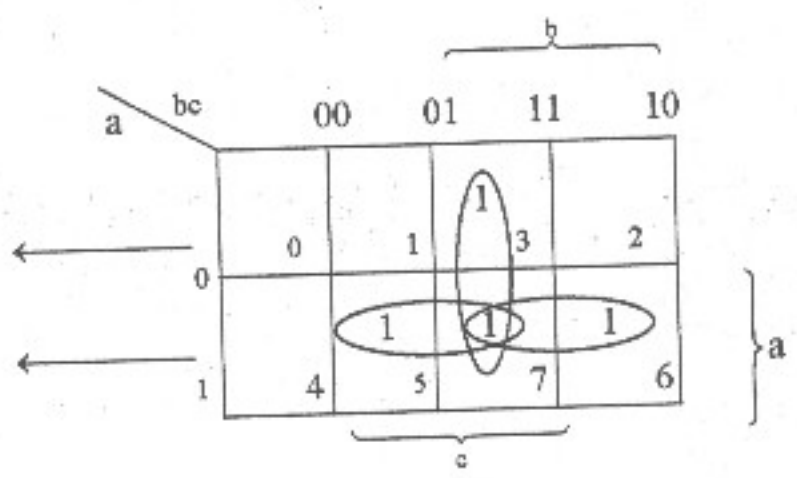
	a	b	c	d	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
A	1	0	1	0	0
B	1	0	1	1	1
C	1	1	0	0	1
D	1	1	0	1	0
E	1	1	1	0	1
F	1	1	1	1	1

ab \ cd	00	01	11	10
00		1	1	1
01	1		1	
11	1		1	1
10			1	

$F = cd + abc + a'b'c + a'b'd + bc'd'$

مثال : جدول کارنو را برای ساده‌سازی مربوط به میز رأی‌گیری سه نفره را رسم کنید.

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



مرحله اول تبدیل جملات

ضرب به فرم استاندارد

مثال:

تابع F را وارد جدول کارنو کنید:

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

تعداد خانه‌های جدول $= 2^4 = 16$

حل: باید ابتدا جملات را به فرم استاندارد (minterm) درآوریم:

$y'z = xy'z + x'y'z \Rightarrow wy'z + w'xy'z + wx'y'z + w'x'y'z$

$wxy' = wxy'z + wxy'z'$

$wxz' = wxyz' + wxy'z'$

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

$F = wxy'z + w'x'y'z + wx'y'z + w'xy'z + wxy'z' + wxyz' + w'x'yz$

حال با توجه به اینکه تابع به صورت استاندارد (minterm ها) درآمده می‌توان آن را وارد جدول کارنو کرد. با توجه به اینکه تابع f به صورت f (w,x,y,z) می‌باشد. با اختصاص خانه‌های جدول و نیز مشخص کردن صفر یا یک بودن هر متغیر به این صورت که اگر متغیر خودش بود یک و اگر متممش بود صفر قرار می‌دهیم، می‌توان جدول را پر کنیم. به عنوان مثال در $w'xyz'$ در واقع به جای w,z باید صفر و به جای x و y و یک قرار دهیم و minterm به شکل ۰۱۱۰ در می‌اید که دو بیت سمت چپ ردیف و دو بیت سمت راست ستون را مشخص می‌کند. سپس از تقاطع آنها می‌توان به خانه مورد نظر هست یافت.

۰۱ : ردیف.

حال تمامی minterm ها را به شکل زیر پر می‌کنیم.

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

شماره‌گذاری خانه‌های جدول کارنو:

در شماره‌گذاری خانه‌های جدول شماره‌های داخل جدول همان شماره mintern ها می‌باشد. وشماره بیرونی مربوط به وجود و عدم وجود متغیر می‌باشد. یعنی صفر به معنای آن است که متغیر وجود ندارد و یک به معنای وجود متغیر می‌باشد.

x \ y	0	1
0	0	1
1	2	3

x \ yz	00	01	11	10
0	0	1	3	2
1	4	5	7	6

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

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

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

مثال:

تابع F را وارد جدول کارنو کنید.

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

001 011 101 111

x \ yz	00	01	11	10
0	0	1 1	1 3	2
1	4	1 5	1 7	6

F = z

مثال:

تابع زیر را در جدول کارنو وارد کنید و همسایه‌گیری کنید.

0000 0001 0010 0101 1000 1001 1010

1111

$$t(x,y,w,z)=\sum(0,1,2,5,8,9,10,15)$$

$$t=y'w'+y'z'+x'w'z+xywz$$

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

مثال:

با استفاده از کارنو 5 متغیره تابع زیر را ساده کنید:

$$F(A,B,C,D,E)=\Sigma(0,2,6,7,8,10,14,15,16,18,20,21,22,23,24,26,28,29,30,31)$$

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

A = 0

	00	01	11	10
	1 16	17	19	1 18
	20	21	23	22
	28	29	1 31	1 30
	1 24	25	27	1 26

A = 1

$$f=C'E'+AC+CD$$

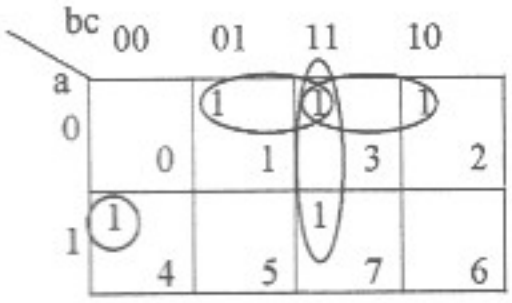
مثال: مداری طراحی و به کمک جدول کارنو ساده کنید که دارای ۳ ورودی باشد برای ورودی‌های صفر تا چهار خروجی، ورودی به اضافه ۳ باشد و از ۵ تا ۷ خروجی، ورودی منهای ۳ باشد.

	a	b	c	x	y	z
0	0	0	0	0	1	1
1	0	0	1	1	0	0
2	0	1	0	1	0	1
3	0	1	1	1	1	0
4	1	0	0	1	1	1
5	1	0	1	0	1	0
6	1	1	0	0	1	1
7	1	1	1	1	0	0

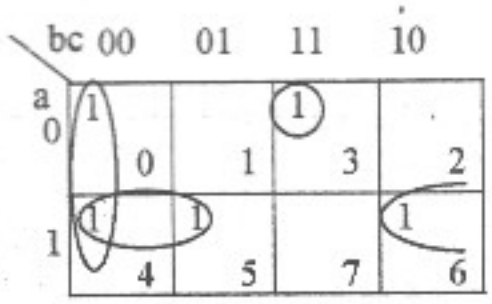
$$x = \Sigma(1,2,3,4,7) = a'b'c + a'bc' + a'bc + ab'c' + abc$$

$$y = \Sigma(0,3,4,5,6) = a'b'c' + a'bc + ab'c' + ab'c + abc'$$

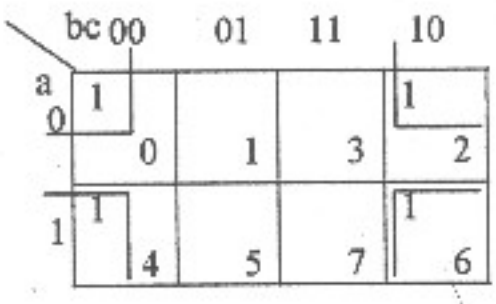
$$z = \Sigma(0,2,4,6) = a'b'c' + a'bc' + ab'c' + abc'$$



$$x = a'c + a'b + bc + ab'c'$$



$$y = ab' + b'c' + ac' + a'bc$$

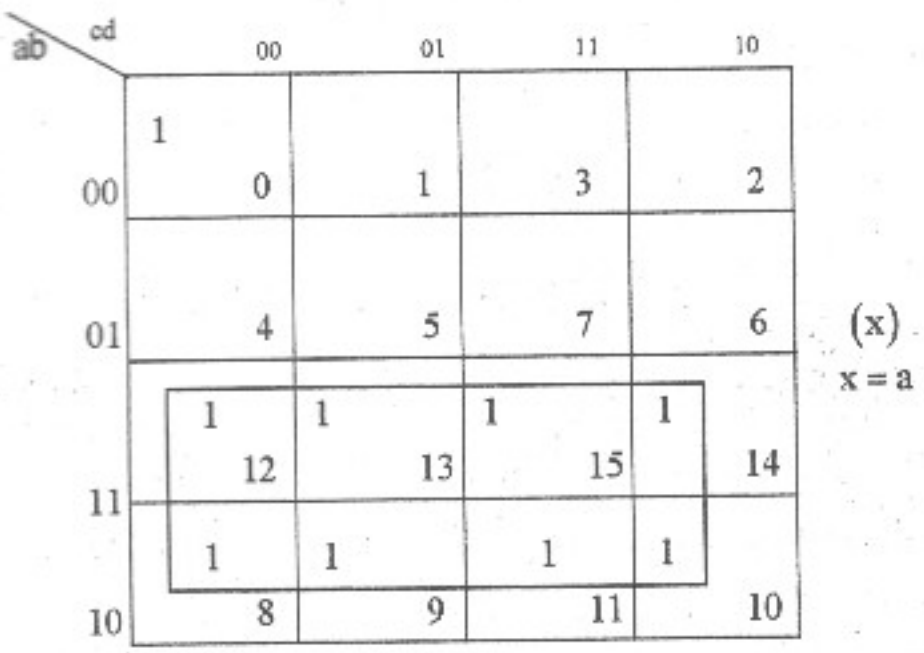


$$z = c'$$

مثال: مداری طراحی کنید و آن را به کمک جدول کارنو ساده کنید که ورودی آن یک عدد 4 بیتی باشد و خروجی‌اش معادل Gray ورودی باشد.

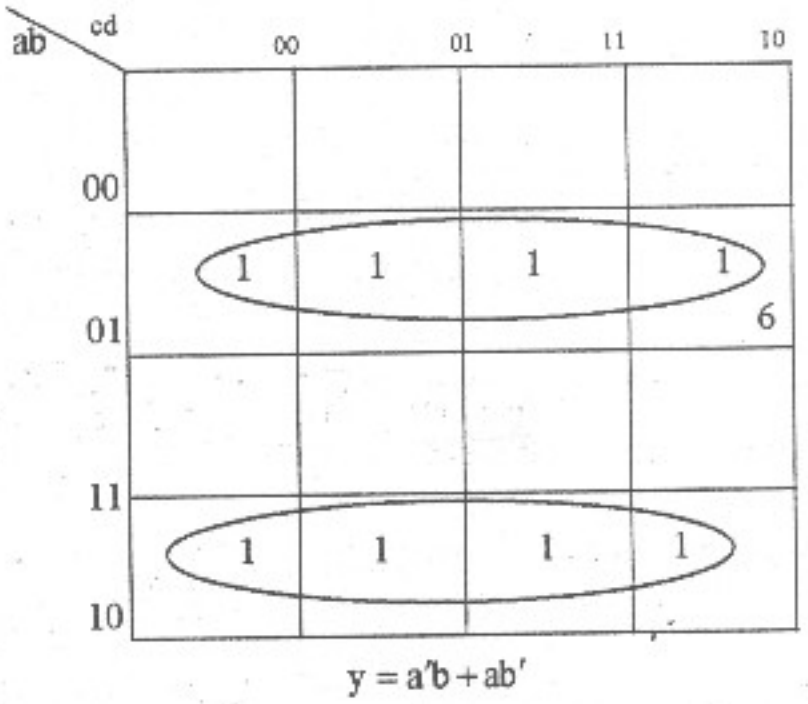
$2^4 = 16$

	a	b	c	d	x	y	w	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



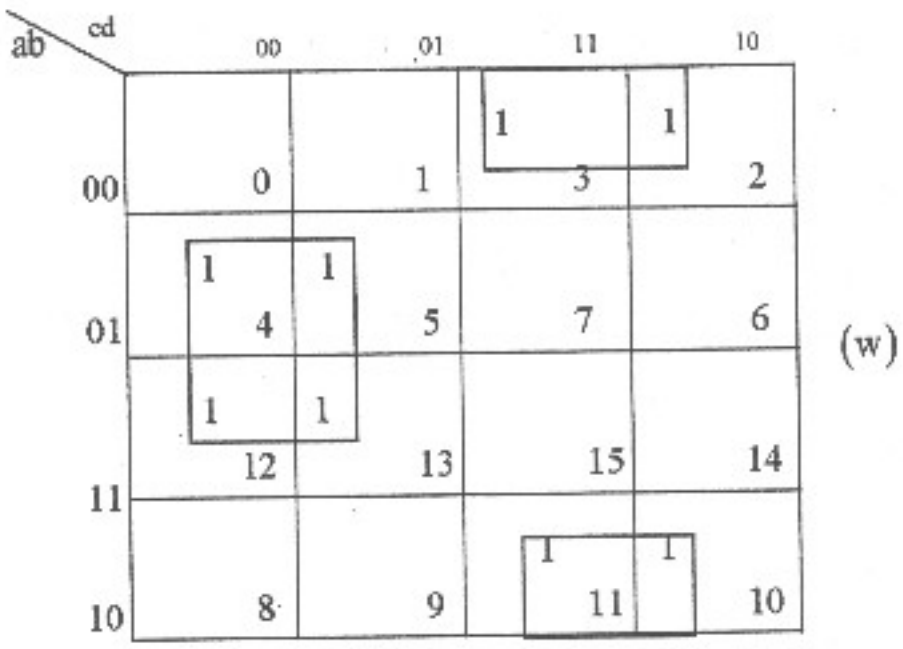
(x)

$x = a$

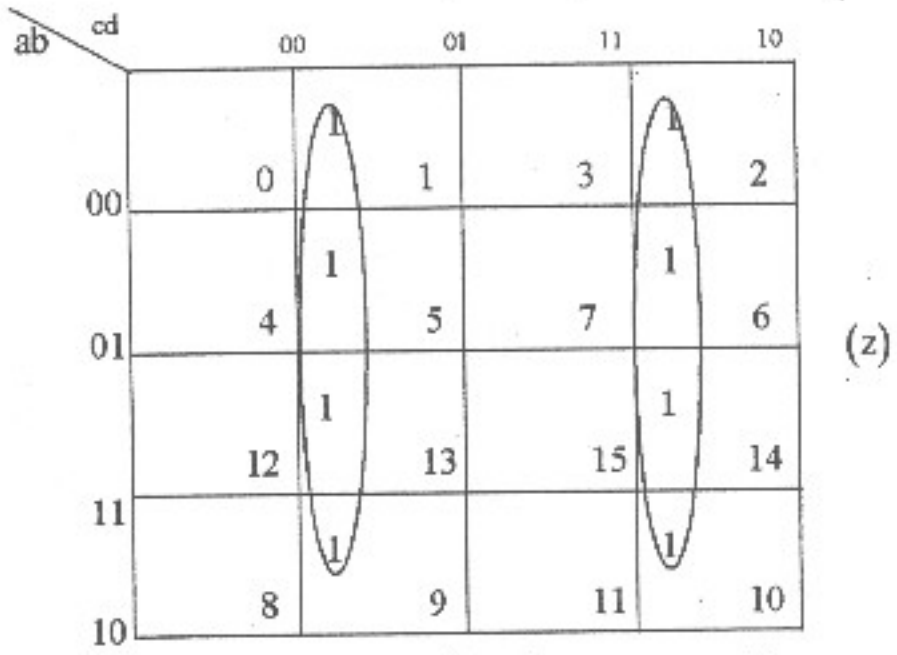


(y)

$y = a'b + ab'$



$w = b'c + b'c$



$z = c'd + cd'$

شکل دیگر شماره‌گذاری جدول کارنو:

شماره‌گذاری جدول کارنو (یا در اصل اختصاص خانه‌های جدول کارنو به متغیرها) انواع متفاوتی دارد. اولین نکته مهم این است که باید اصل همپوشانی در شماره‌گذاری یا اختصاص خانه‌های جدول رعایت شود. دومین نکته این است که باید شماره‌گذاری جدول به نحوی باشد که یک فرم استاندارد را رعایت کرده باشد. به این مفهوم که ساده شده یک تابع به کمک دو جدول کارنو که از شماره‌گذاری‌های متفاوت اما استاندارد استفاده می کنند نباید تفاوت داشته باشند.

شکل دیگر شماره‌گذاری جدول کارنو

اگر جای متغیرها را در جدول کارنو تغییر دهیم به شکل جدید شماره گذاری می‌رسیم. به این مفهوم اگر در یک کارنوی چهار متغیره که تابع F(x,y,w,z) را نشان می دهد به جای اینکه به مانند شماره‌گذاری در حالت قبل xy را به ردیفها و wz را به ستونها اختصاص دهیم، برعکس عمل کنیم یعنی xy را به ستونها و wz را به ردیفها اختصاص دهیم، شماره‌های داخل جدول کارنو به فرم زیر تغییر می‌کنند.

xy / wz	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	9	14	10

ملاحظه می‌شود تنها تفاوت این روش با روش قبلی در نحوه ناحیه بندی آن می‌باشد. توجه داشته باشید که تفاوتی ندارد که از کدام نوع شماره‌گذاری برای جدول کارنو خود استفاده می‌کنید، تنها این مسئله مهم است که باید حتماً از استانداردهای شماره‌گذاری استفاده کنید تا در پایان ساده شده تابع به درستی به وجود آید.

نحوه شکل گیری شماره‌های داخلی جدول کارنو:

هر خانه از جدول کارنو یک شماره دودویی مختص به خود را داراست. به این شکل که در کنار هر ردیف و هر ستون یک شماره دودویی دیده می‌شود (مثلاً 00 یا 10) و با توجه به نام متغیری که به آن اختصاص داده شده به مفهوم وجود یا عدم وجود آن متغیر می‌باشد.

به عنوان مثال در جدول کارنوی زیر که مربوط به تابع F(A,B,C) می‌باشد داریم:

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

کنار ردیف اول عدد صفر نوشته شده که به مفهوم عدم وجود A در این ردیف می‌باشد و ردیف دوم عدد یک به معنای اختصاص این ردیف به A است.

یا عدد 01 در ستون دوم به معنای وجود C و عدم وجود B می‌باشد.

حال اگر بخواهیم یک خانه را شماره‌گذاری کنیم با توجه به ارزش مکانی متغیرها (به این مفهوم که در مثال فوق A بیشترین ارزش مکانی و C کمترین ارزش را داراست) مقدار دودویی تبدیل به معادل دهدهی می‌باشد. به عنوان مثال در جدول فوق اگر بخواهیم خانه‌ای در ردیف اول و در ستون چهارم را شماره‌گذاری کنیم داریم:

$$\begin{cases} A=0 \\ BC=10 \end{cases} \rightarrow ABC=010$$ تبدیل به معادل دهدهی $\rightarrow$ 2

C \ AB	00	01	11	10
0	0	2	6	4
1	1	3	7	5

A	B	C	شماره
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

این روش را برای تمامی خانه‌های جدول کارنو می‌توان پیاده کرد.

بنابراین ملاحظه می‌شود که با تغییر در جای متغیرها، باز هم می‌توان خانه های جدول را با توجه به ارزش مکانی آنها شماره گذاری کرد.

به عنوان مثال برای F(x,y,z) داریم:

x \ yz	00	01	11	10
0	0	1	3	2
1	4	5	7	6

y \ xz	00	01	11	10
0	0	1	5	4
1	2	3	7	6

y \ xz	00	01	11	10
0	0	2	6	4
1	1	3	7	5

x

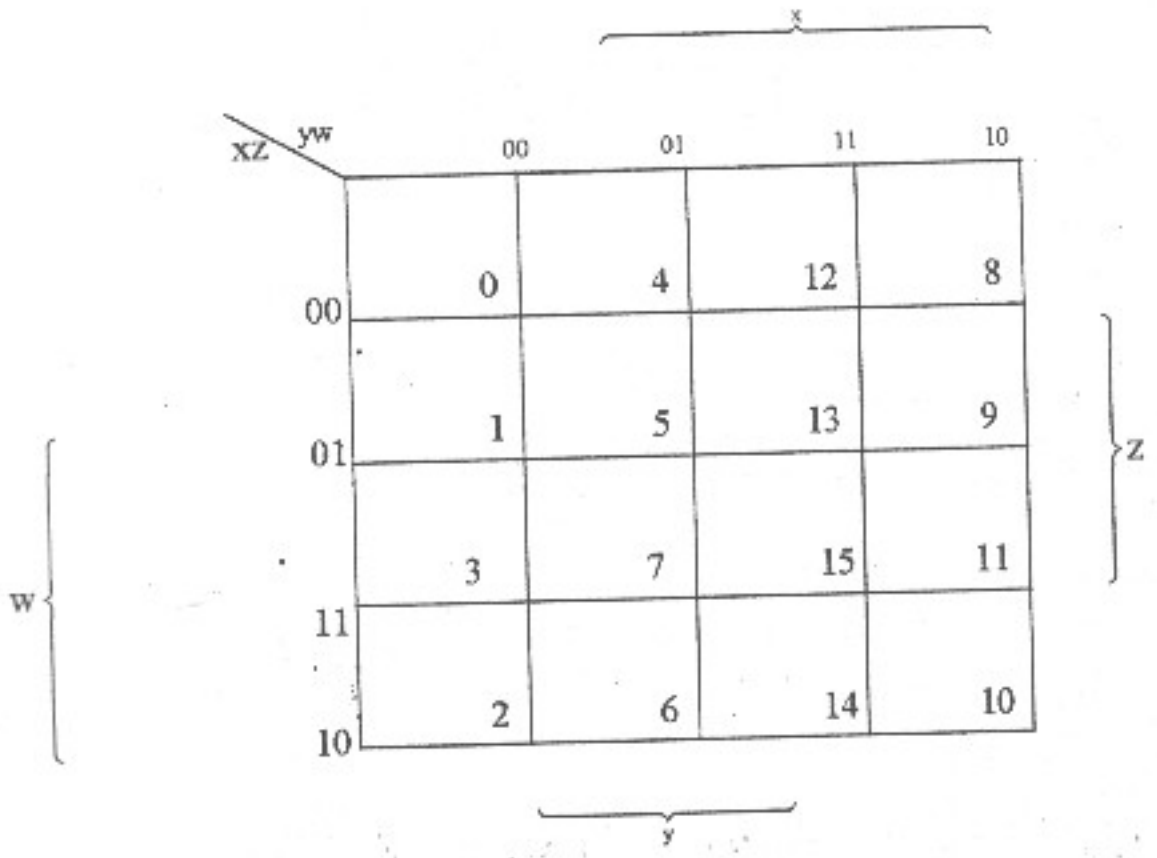
xz \ yw	00	01	11	10
00	0	2	6	4
01	1	3	7	5
11	9	11	15	13
10	8	10	14	12

x y w z
1 1 0 0

f(x,y,w,z)

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

f(x,y,w,z)



f(x,y,w,z)

حالت‌های بی‌اهمیت یا Dontcare:

به حالتهایی گفته می‌شود که یک بودن یا صفر بودن آنها تفاوت نمی‌کند. و طراح مطابق با نیاز خود آن را صفر یا یک در نظر می گیرید.

در جدول کارنو به جای حالات don't care باید X گذاشته شود.

ویژگی اصلی حالتهای بی‌اهمیت در این است که می توان آنها را هم با صفرها و هم با یک‌ها همسایه گرفت.

* اگر در یک جدول درستی حالت‌ها فقط یک و don't care یا صفر و don't care باشند احتیاج به جدول کارنو نداریم زیرا در اولی حاصل برابر یک و در دومی حاصل برابر صفر است.

مثال:

تابع f را با حالات بی‌اهمیت d در نظر بگیرید، آن را ساده کنید:

$$f(a,b,c)=\sum(1,3,7) \qquad d=\sum(0,2,5)$$

a \ bc	00	01	11	10
0	X 0	1 1	1 3	X 2
1	 4	X 5	1 7	 6

F = C

شماره‌های صفر، و 2 را در نظر نمی گیریم، چون تابع را مشکل‌تر می کنند. اما شماره 5 را به عنوان یک در نظر می گیریم و می بینیم که باعث ساده‌تر شدن تابع می‌شود.

حال فرض کنیم که همه حالات بی اهمیت را صفر در نظر بگیریم، خواهیم دید که تابع به فرم $F=a'c+bc$ در می‌آید. که از فرم قبلی $(F=c)$ پیچیده‌تر می‌باشد بنابراین در این مثال استفاده از حالت بی‌اهمیت به سود ما بود.

اما نمی‌توان گفت که حالات بی‌اهمیت همیشه برای ما مفید خواهند بود، بلکه با توجه به مکان این حالات می‌توان از آنها استفاده بهینه کرد.

نکته:

اگر خانه‌های 1,2,4,7 جدول کارنو یک بود ⟵ XOR سه ورودی

اگر خانه‌های 0,3,5,6 جدول کارنو یک بود ⟵ XNOR سه ورودی

یا اگر در خانه‌های جدول خانه‌هایی که مقدارشان یک است در موقعیت یکی در میان بودند و شروع از خانه صفر بود تابع X-NOR ورودی‌ها پیاده سازی شده و اگر شروع از خانه یک بود تا X-OR پیاده سازی شده است.

چهار ورودی XNOR

wz \ xy				
	1		1	
		1		1
	1		1	
		1		1

چهار ورودی X-OR

wz \ xy				
		1		1
	1		1	
		1		1
	1		1	

ایجاب کننده (Implicant):

در یک تابع n متغیره به صورت $f(x_1, x_2, ..., x_n)$ اگر جمله حاصلضرب P حداقل یک نقطه یک تابع را بپوشاند و هیچ نقطه صفر با نپوشاند ایجاب کننده تابع F نامیده می‌شود. در اصل یک عبارت ضرب می‌باشد که یک یا چند Minterm را شامل شود.

ایجاب کننده نخستین (Prime Implicant) یا (PI):

ایجاب کننده P از تابع F را ایجاب کننده نخستین گویند هر گاه برای هر ایجاب کننده دیگر مانند P نقطه‌ای وجود داشته باشد که به وسیله P پوشانده نشود ولی به وسیله P پوشانده شود.

مثال:

با توجه به جدول درستی زیر نوع Pها را مشخص کنید.

A	B	C	F	
0	0	0	1	ایجاب کننده است → $P_1 = A'B'$
0	0	1	1	
0	1	0	0	
0	1	1	1	ایجاب کننده نیست → $P_2 = A'B$
1	0	0	-	
1	0	1	1	ایجاب کننده است → $P_3 = AB'$
1	1	0	0	
1	1	1	0	

حال می‌توان P_4 را به وجود آورد که از ترکیب P_3, P_1 تشکیل شده است:

$P_1, P_3 \Rightarrow P_4 = B'$

همانطور که ملاحظه می‌شود P_4 توانایی پوشش چهار مینترم شمارههای (0,1,4,5) را داراست. در اصل P_4 یک ایجاب کننده نخستین می‌باشد زیرا نمی‌توان پوشش دیگری پیدا کرد که چهار مینترم فوق را بپوشاند.

نکته: ملاحظه می‌شود که هر ایجاب کننده در واقع یک جمله حاصلضرب است که ساده شده minterm ها می‌باشد. بنابراین اگر بتوان یک تابع را به صورت ایجاب کننده‌هایش در آوریم آن را به ساده‌ترین صورت نوشته‌ایم.

حال اگر تابع را به شکل PI هایش بنویسیم در اصل ساده‌ترین صورت آن تابع را بدست آورده‌ایم.

بدست آوردن PIهای توابع:

می‌دانیم که هر minterm یک نقطه از تابع را می‌پوشاند و از ترکیب دو مینترم که فقط در یک بیت (یک متغیر) با هم تفاوت دارند می‌توان جمله حاصلضربی ایجاد نمود که دو نقطه از تابع را می‌پوشاند. متغیری که متفاوت بود حذف شده و جای آن می‌توان علامت "–" به معنی بی‌اهمیت را قرارداد حال جمله حاصلضرب بدست آمده را می‌توان با بقیه جملات حاصلضرب که آنها هم از ساده شدن minterm های دیگر تشکیل شده ، مقایسه کرد و عمل ساده کردن را انجام داد، این کار را تا جایی ادامه می‌دهیم تا امکان هیچ ترکیب دیگری برای ساده سازی وجود نداشته باشد. با انجام عملیات فوق در اصل بد PIهای یک تابع رسیده‌ایم.

روش QM برای ساده‌سازی توابع (کوئین مک کلاسکی):

با استفاده از الگوریتم QM می‌توان توابع را به راحتی ساده کرد:

مرحله ۱ : دسته‌بندی mintermها:

منظور از دسته بندی mintermها جداسازی آنها با توجه به تعداد یک‌های آنها می‌باشد به عنوان مثال دو؟؟ که به صورت ABC'D و A'BCD هستند در اصل به صورت 1101 و 0111 هستند که در هر دو آنها سه عدد "1" وجود دارد بنابراین هر دو در یک دسته قرار خواهند گرفت.

بنابراین در این مرحله جدولی را تشکیل می‌دهیم که در آن دسته‌های مختلف minterm ها وجود دارند.

مرحله ۲: مقایسه mintermها :

در این مرحله باید عمل مقایسه بین minterm را انجام دهیم به این شکل که هر minterm از یک دسته را با minterm در دسته بعدی مقایسه می‌کنیم اگر تنها در یک بیت تفاوت داشتند آن بیت متفاوت را حذف می‌کنیم. عمل مقایسه را تا زمانی انجام می‌دهیم که هیچ دو minterm از دو دسته متفاوت وجود نداشته باشند که با هم مقایسه نشده باشند به عنوان مثال در مقایسه دو minterm شماره‌های ۵ و ۷ که در دو دسته متفاوت قرار دارند خواهیم داشت:

$$\left.\begin{array}{ll} 101 \rightarrow & \text{شماره ۵ minterm} \\ 111 \rightarrow & \text{شماره ۷ minterm} \end{array}\right\} \rightarrow 1-1$$

همانطور که ملاحظه می‌شود درminterm فوق در دومین بیت با هم متفاوت هستند بنابراین آن بیت حذف می‌شود و به جای آن علامت "–" به معنای بی‌اهمیت یا Dont care قرار می‌گیرد.

پس از انجام عمل مقایسه بین هر دو minterm کنار شماره آن علامت " " را می‌زنیم . در انتها باید کنار تمام minterm ها علامت" " گذاشته شده باشد.

مرحله ۳:مرحله دوم ساده سازی:

با انجام مرحله ۲ به یک ستون جدید در جدول رسیده‌ایم حال باید به آنها نیز به چشم minterm نگاه کنیم و مانند مرحله اول شروع به دسته‌بندی آنها کنیم سپس همان مراحل که برای مقایسه mintermها انجام داده بودیم را اعمال کنیم.

باید توجه داشت که برای انجام دسته بندی تمامی حالات بی‌اهمیت را صفر در نظر می‌گیریم و عمل دسته بندی را انجام می‌دهیم.

پس از انجام عمل مقایسه می‌بینیم که ممکن است در هر دسته minterm ی وجود داشته باشد که جلوی آن علامت " " وجود ندارد، در اصل این جمله حاصلضرب یکی PI های ما خواهد بود که با هیچ جمله حاصلضرب دیگری قابل ساده شدن نیست.

پس از انجام مراحل فوق در انتها به یک جمله حاصلضرب می‌رسیم که آن جمله هم یک PI خواهد بود.

مثال:

تابع $f(A,B,C,D)=\Sigma(0,1,2,5,8,10,14,15)$ را به کمک روش M ساده کنید.

مرحله اول: دسته‌بندی

	D	C	B	A	شماره minterm
دسته صفر	0	0	0	0	0
دسته یک	1	0	0	0	1
	0	1	0	0	2
	0	0	0	1	8
دسته دو	1	0	1	0	5
	0	1	0	1	10
دسته سه	0	1	1	1	14
دسته چهار	1	1	1	1	15

همانطور که ملاحظه می‌شود minterm‌ها به پنج دسته تقسیم شدند که در دسته iام تعداد یک‌ها i تا می‌باشد.

مرحله دوم: مقایسه بین دو minterm در دسته‌های متفاوت

شماره minterm	A	B	C	D	شماره minterm مقایسه شده	A	B	C	D
0	0	0	0	0	(0,1)→	0	0	0	–
1	0	0	0	1	(0,2)→	0	0	-	0
2	0	0	1	0	(0,8)→	-	0	0	0
8	1	0	0	0	(1,5)→	0	-	0	1
5	0	1	0	1	(2,10)→	-	0	1	0
10	1	0	1	0	(8,10)→	1	0	-	0
14	1	1	1	0	(10,14)→	1	-	1	0
15	1	1	1	1	(14,15)	1	1	1	-

حال باید حاصل minterm های ساده شده رادر دسته‌بندی کنیم:

نام	D	C	B	A	
دسته صفر (P_2)	-	0	0	0	(0,1)→
(P_3)	0	-	0	0	(0,2)→
(P_4)	0	0	0	-	(0,8)→
دسته یک (P_5)	1	0	-	0	(1,5)→
(P_6)	0	1	0	-	(2,10)→
(P_7)	0	-	0	1	(8,10)→
دسته دو	0	1	-	1	(10,14)→
دسته سه	0	1	1	1	(14,15)→

حال باز هم باید عمل مقایسه را انجام دهیم:

$$\left.\begin{matrix}(P_3,P_7)\rightarrow - \ 0 \ - \ 0\\ (P_4,P_6)\rightarrow - \ 0 \ - \ 0\end{matrix}\right\}\Rightarrow$$

مینترم‌های شماره $(0,2,8,10)$ با یکدیگر ساده شده‌اند و تبدیل به:

A B C D
\- 0 - 0

شده است بنابراین $B'D'$ یک PI می‌باشد و تابع به مجموع PIهای زیر تبدیل شده است:

$$\{A'B'C'+A'C'D+ACD'+ABCD'+B'D'\}$$

انتخاب پوشش مینیمم از مجموعه PIهای تابع:

پوشش: انتخاب حداقل تعدد از PI های تابع با کمترین هزینه به طوری که هر نقطه "1" تابع حداقل به وسیله یکی از PI های تابع پوشانده شود را پوشش گویند.

جدول پوشش: جدولی می‌باشد که به ازای هر PI یک سطر و به ازای یک‌های تابع (minterm) ستون دارد.

نحوه پر کردن جدول پوشش: در تقاطع سطر ام و ستون ام یک درج می‌کنیم اگر ایجاب کننده نخستین P_i مینترم M_j را بپوشاند.

نکته: نقاط Don't Case در جدول درج نمی‌شوند زیرا نیازی به پوشش آن‌ها نیست.

برای بدست آوردن پوشش مینیمم کافیست جدول پوشش را ساده کرد برای ساده کردن جدول پوشش کافیست از الگوریتم زیر پیروی کنیم:

۱ـ حذف PIهای ضروری

در صورتی که مینترم m_j تنها به وسیله یک P_i پوشانده شود آن را PI ضروری گویند که باید در هر پوشش مینیمم ظاهر شده باشد جدول را با حذف سطر P_i و کلید ستون‌های پوشیده شده به وسیله آن را حذف می‌کنیم.

۲ـ حذف سطرهایی مغلوب:

اگر P_i کلید نقاط متناظر با P_j را بپوشاند در این صورت گوئیم P_i غالب بر سطر P_j است. در این صورت سطر مغلوب یعنی P_j را حذف می‌کنیم.

۳ـ حذف ستون‌های غالب:

اگر هر سطری که m_j را نیز بپوشاند و نسبت به m_j نقاط بیشتری داشته باشد گوئیم m_i غالب (Dominate) نیز m_j است.
در این صورت ستون غالب یعنی m_i را حذف می‌کنیم.

مثال:

جدول پوشش تابع

$$f(x_1,x_2,x_3,x_4)=\Sigma(0,1,2,5,8,10,14)+d(7)$$

برای بدست آوردن مجموعه PIهای تابع f از روش QM می‌توان سود برد:

$$PI=\{x_1'x_2'x_3',x_1'x_3'x_4',x_1'x_2x_4,x_1x_3x_4',x_1x_2x_3,x_2x_3x_4,x_2'x_4'\}$$

شماره PI ها		m_0	m_1	m_2	m_5	m_8	m_{10}	m_{14}	m_{15}
P_1	$x_1'x_2'x_3'$	1	1						
P_2	$x_1'x_3'x_4'$		1		1				
P_3	$x_1'x_2x_4$						1	1	
P_4	$x_1x_3x_4'$							1	1
P_5	$x_1x_2x_3$								1
P_6	$x_2x_3x_4$					1	1		
P_7	$x_2'x_4'$	1		1					

مرحله اول حذف PIهای ضروری:

P_7 ضروری است بنابراین سطر P_7 و ستون‌های m_0,m_2,m_8,m_{10} از جدول حذف می‌شود.

مرحله دوم حذف سطرهای مغلوب:

همانطور که دیده می‌شود P_2 بر P_3 و P_1 و P_5 بر P_6,P_4 غالب است.

بنابراین P_6,P_4,P_3,P_1 باید حذف شوند.

		m_1	m_5	m_{14}	m_{15}
P_1	$x_1'x_2'x_3'$	1			
P_2	$x_1'x_3'x_4$	1	1		
P_3	$x_1'x_2x_4$		1		
P_4	$x_1x_3x_4'$			1	
P_5	$x_1x_2x_3$			1	1
P_6	$x_1x_2x_3$				1
P_7	$x_2x_3x_4$				

بنابراین تنها سطرهای P_5,P_2 باقی می‌مانند که به همراه PI7 تابع را به صورت کامل پوشش می‌دهند.

$$f=P_7+P_2+P_5=x_2'x_4'+x_1'x_3'x_4+x_1x_2x_3$$

تاخیر انتشار:

مدت زمان تاخیر بین تغییر ورودی و خروجی یک گیت را گویند به این مفهوم که چه مدت زمانی طول می‌کشد که تغییر در ورودی باعث تغییر در خروجی شود و دو نوع تاخیر انتشار وجود دارد:

T_{PHL}: مدت زمان تاخیری که خروجی از یک به صفر تغییر می‌کند.

T_{PLH}: مدت زمان تاخیری که خروجی از صفر به یک تغییر می‌کند.

$$\text{تاخیر انتشار } T_{PD} = \frac{T_{PHL} + T_{PLH}}{2}$$

پیاده سازی با گیت‌های NAND و NOR:

برای پیاده سازی NOT بصورت تمام NAND از یک NAND تک ورودی و برای پیاده‌سازی تمام NOR از یک NOR تک ورودی استفاده می‌کنیم.

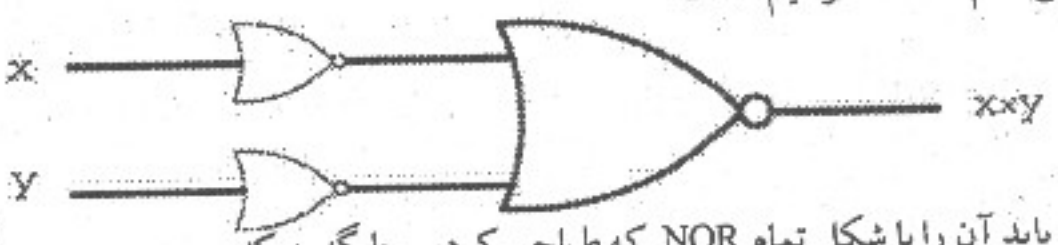
در اصل دو راه برای به وجود آوردن NOR یا NAND تک ورودی داریم که به صورت NOT عمل کند، راه اول این است که دو ورودی را به یکدیگر وصل کنیم و راه دوم این است که یک ورودی را حتماً یک کنیم.

باید توجه داشت که اگر در NAND تک ورودی و یا در NOR تک ورودی پشت سر هم باشند چون عمل NOTکردن دو بار انجام می‌شود به معنی این است که هیچ عملی انجام نشده و در اصل این دو NOT پشت سر هم کار یک سیم را انجام می‌دهند و یکدیگر را خنثی می‌کنند بنابراین می توان آنها را حذف کرد و یک سیم به جای آنها گذاشت.

برای پیاده سازی AND در طراحی تمام NAND خواهیم داشت:

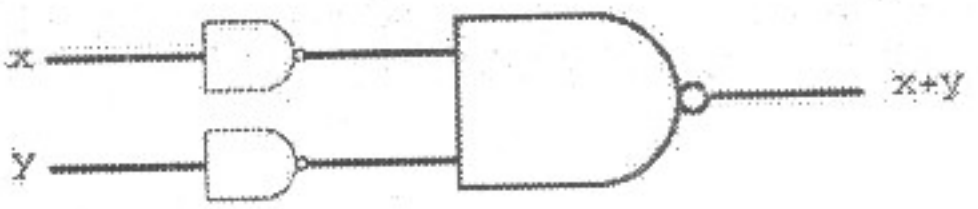
یعنی باید هر کجا که گیت AND دیدیم آن را حذف کنیم و به جای آن شکل تمام NAND آن را قرار دهیم.

برای پیاده سازی AND در طراحی تمام NOR خواهیم داشت:



یعنی هر کجا گیت AND داشتیم باید آن را با شکل تمام NOR که طراحی کردیم جایگزین کنیم.

برای پیاده سازی OR در طراحی تمام NAND خواهیم داشت:



برای پیاده سازی OR در طراحی تمام NOR خواهیم داشت:

برای پیاده سازی Not درطراحی تمام NAND و NOR خواهیم داشت:

دلیل استفاده از طراحی مدارهای تمام NAND و NOR:

به علت استفاده از IC ها طراحی تمام NAND و NOR انجام می‌شود. از آنجا که در هر IC ممکن است 2 یا 4 عدد از آن گیت وجود داشته باشد بنابراین مقرون به صرفه است که از یک IC استفاده لازم را ببریم.

به عنوان مثال فرض کنید بخواهیم تابع $F = AB + C'$ را پیاده سازی کنیم در حالت عادی باید از یک گیت OR و یک گیت AND و یک NOT استفاده کنیم که برای پیاده سازی به کمک ICها باید 3 عدد IC متفاوت را به کار ببریم. با توجه به اینکه در هر IC چهار گیت از آن نوع وجود دارد بنابراین در پایان پیاده سازی، سه گیت OR، سه گیت AND، و 5 گیت NOT بدون استفاده خواهیم داشت. که نشاندهنده عدم استفاده صحیح (طراحی صحیح) می باشد.

حال اگر فرض کنیم تابع را بخواهیم به کمک NAND پیاده سازی کنیم خواهیم داشت:

ملاحظه می‌شود که می‌توان تابع فوق را به کمک دو گیت NAND پیاده سازی کرد. بنابراین لازم است یک IC از نوع NAND خریداری کرد و از نیمی از آن استفاده کرد.

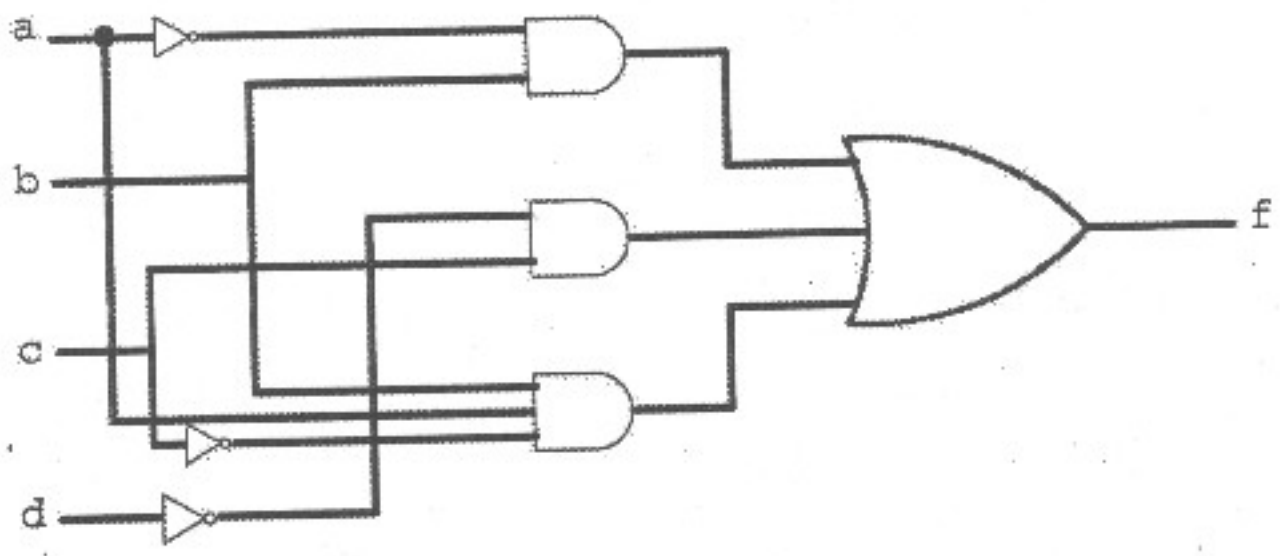
می‌بینیم که با طراحی جدید هم پیچیدگی مدار پایین آمده و هم سخت افزار کمتری استفاده شده است.

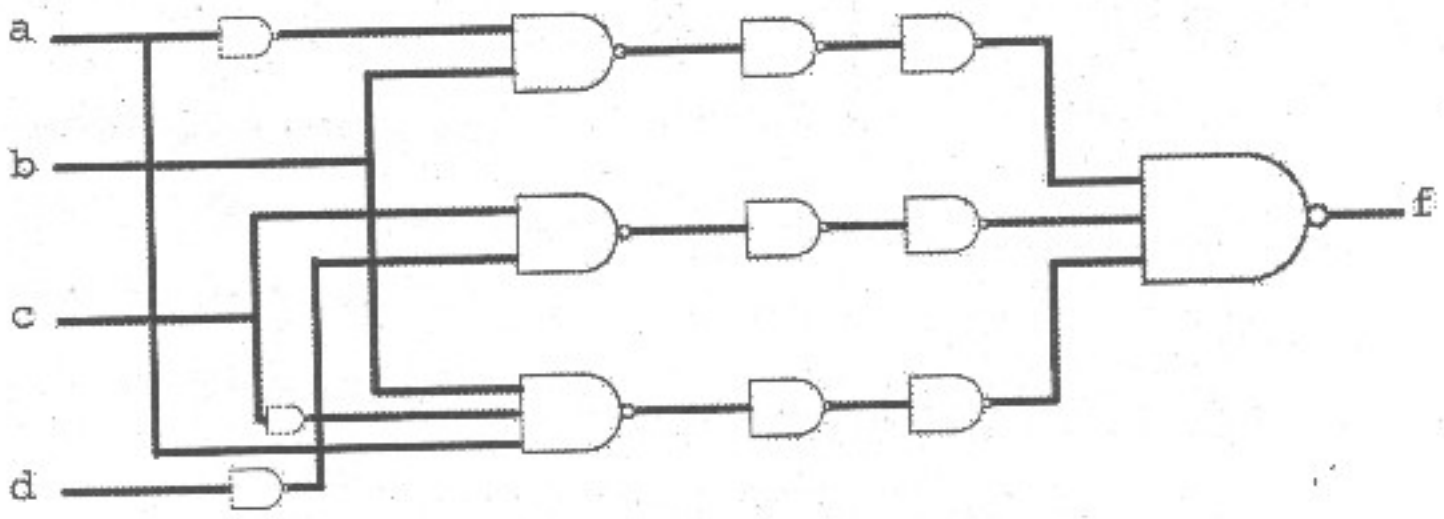
دلیل دوم استفاده از گیتهای NAND و NOR این است که این گیتها به تنهایی قادر به پیاده سازی تمامی مدارات ترکیبی می‌باشند به این مفهوم که این گیتها از نظر عملیاتی کامل هستند.

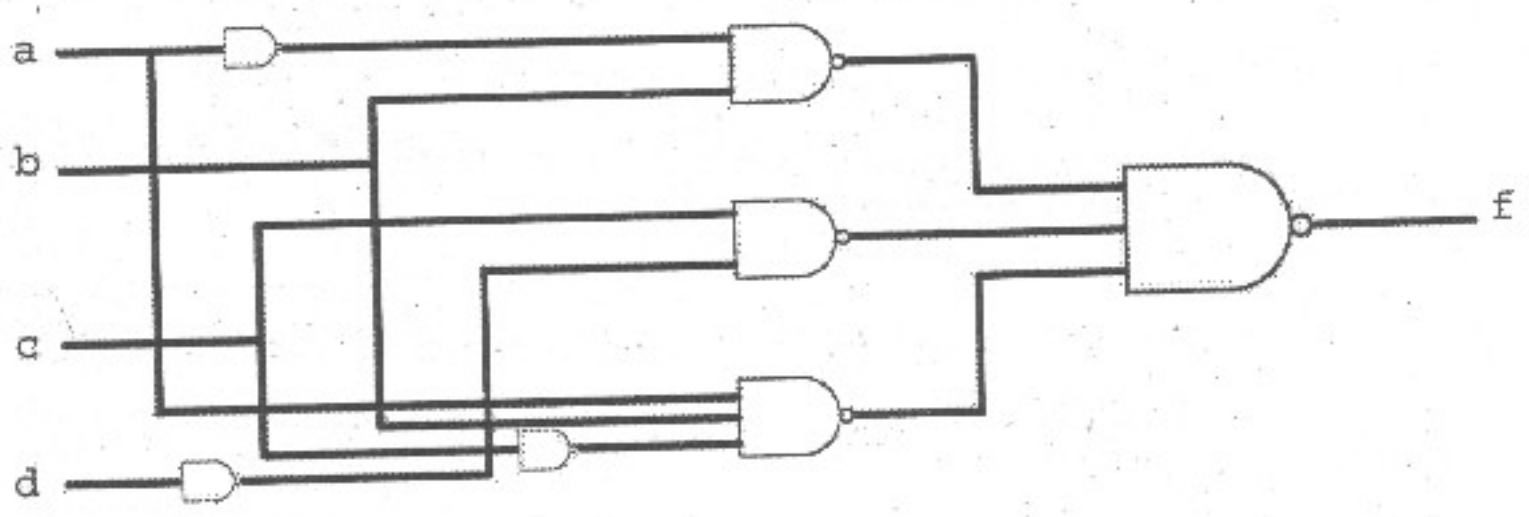
- NAND و NOR از نظر عملیایت کامل هستند به این معنا که تمام مدارها را می‌توان به کمک NAND و یا NOR پیاده‌سازی کرد.

مثال:

تابع $F = A'B + CD' + ABC'$ را به صورت تمام NAND پیاده سازی کنید.



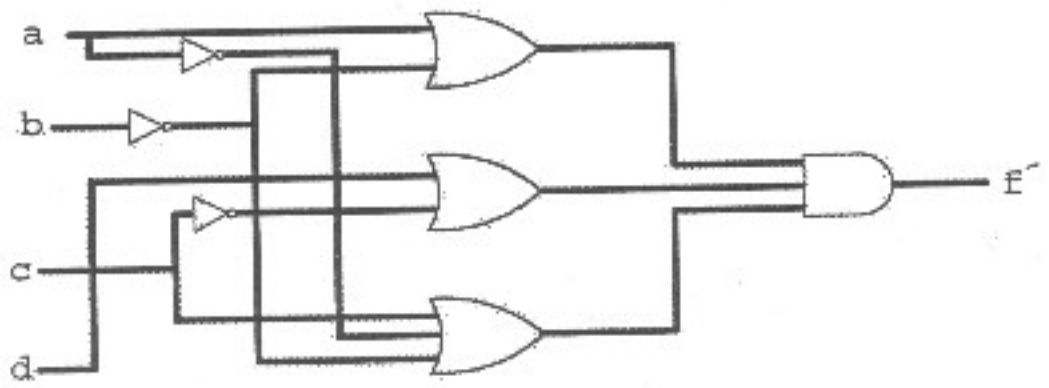


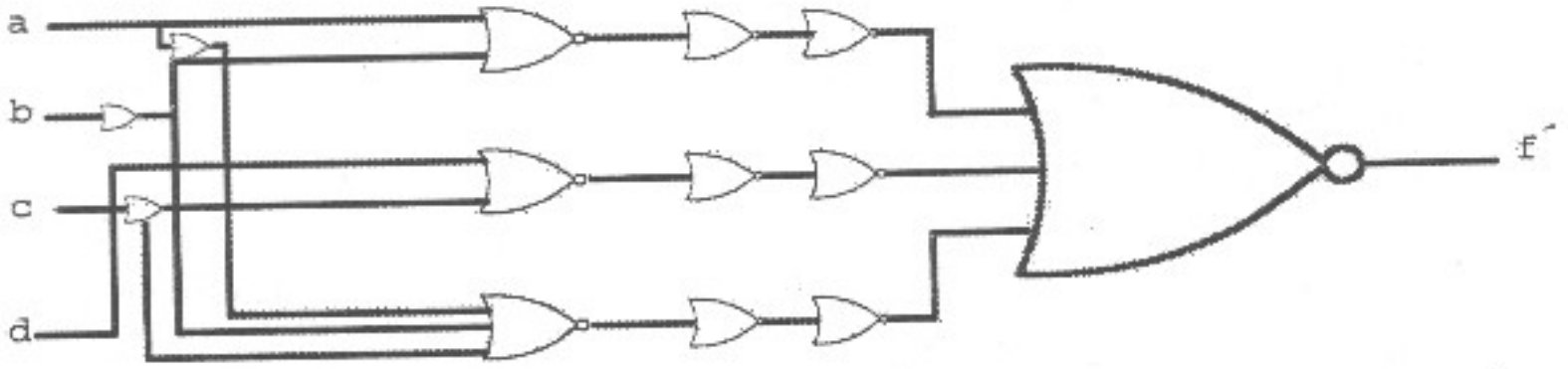


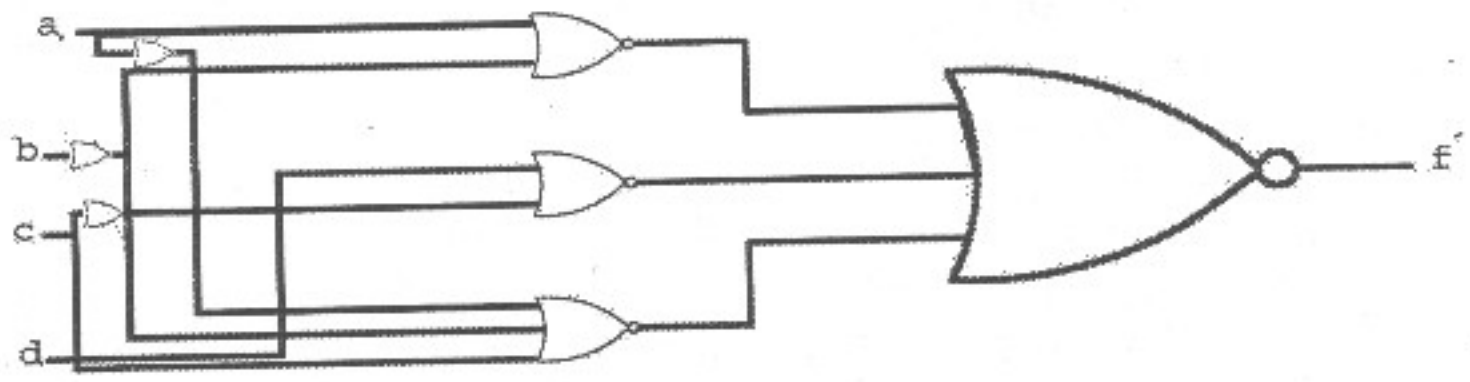
مثال:

تابع $F = A'B + CD' + ABC'$ در نظرگرفته و F' را به صورت تمام NOR پیاده سازی کنید.

* این را در نظر داشته باشید گاهی پیاده‌سازی F' را حت‌تر و به صرفه‌تر است زیرا ممکن است از گیت‌های کمتری استفاده کنیم و در انتها می‌توان با قرار دادن یک Not جلوی F' آن را به F تبدیل کرد.







مثال:

توابع بولی زیر را به صورت ضرب حاصل جمع‌ها ساده کنید.

الف) $F(w,x,y,z)=\sum(0,2,5,6,7,8,10)$

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

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

ب) $F(A,B,C,D)=\Pi(1,3,5,7,13,15)$

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

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

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

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

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

مثال:

عبارت زیر را ساده کنید و آن را با مدارهای دو طبقه گیت NAND پیاده‌سازی کنید.

$$F(ABCD)=AB'+ABD+ABD'+A'C'D'+A'BC'$$

$AB'(C+C')=AB'C(D+D')+AB'C'(D+D')=AB'CD+AB'CD'+AB'C'D+AB'C'D'$

$ABD(C+C')=ABCD+ABDC'$

$ABD'(C+C')=ABD'C+ABD'C'$

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

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

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

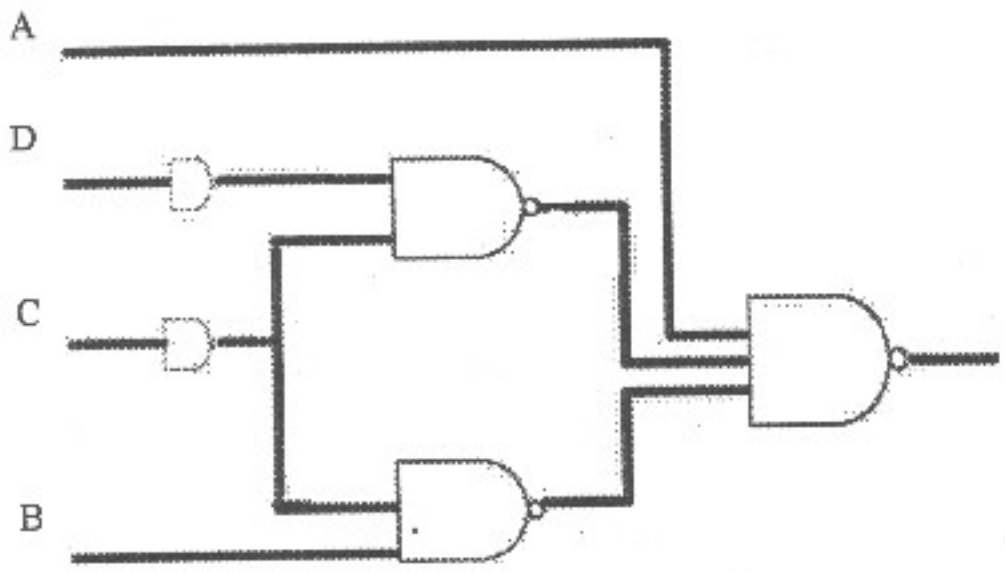
1011 1010 1001 1000 1111 1101

$+ABCD'+ABC'D'+A'BC'D'+A'B'C'D'+A'BC'D+A'BC'D'$

1110 1100 0100 0000 0101 0100

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

$F=A+BC'+C'D'$



مثال:

یک نمودار گیت NAND رسم کنید به نحوی که تابع زیر را پیاده‌سازی کرده آن را متمم نمائید.

$$F(A,B,C,D) = \sum(0,1,2,3,4,8,9,12)$$

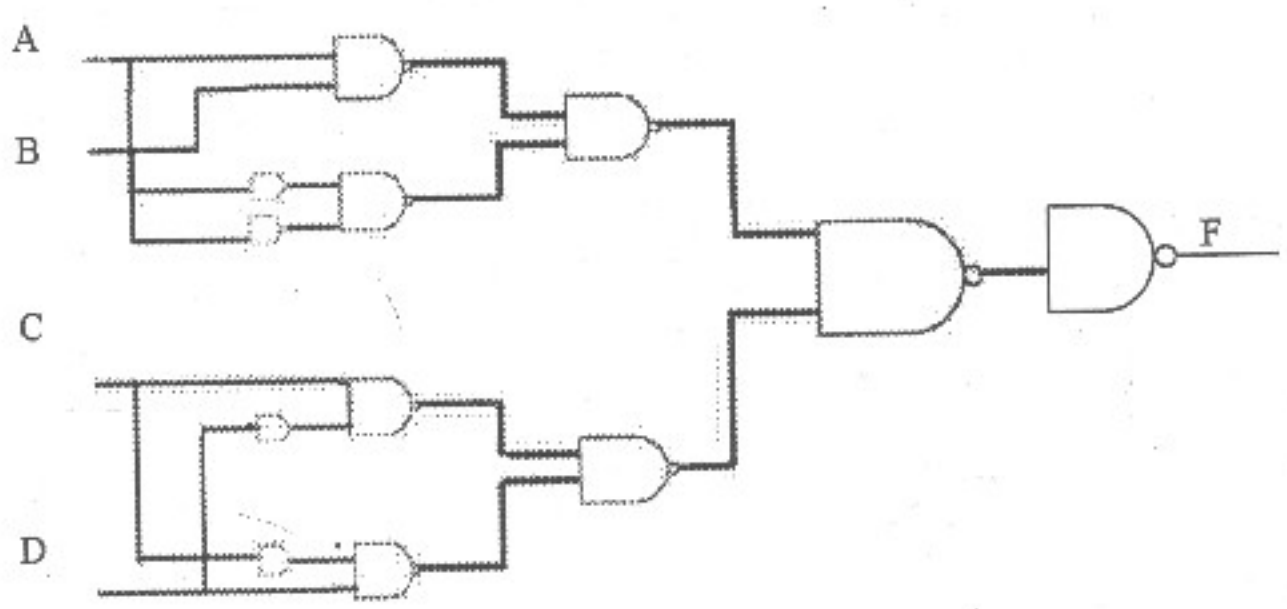
AB \ CD	00	01	11	10
00	1	1	1	1
01	1			
11	1			
10	1	1		

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

مثال:

یک نمودار منطقی با استفاده از گیت‌های NAND دو ورودی برای پیاده‌سازی عبارت زیر رسم کنید.

$$(AB + A'B')(CD' + C'D)$$



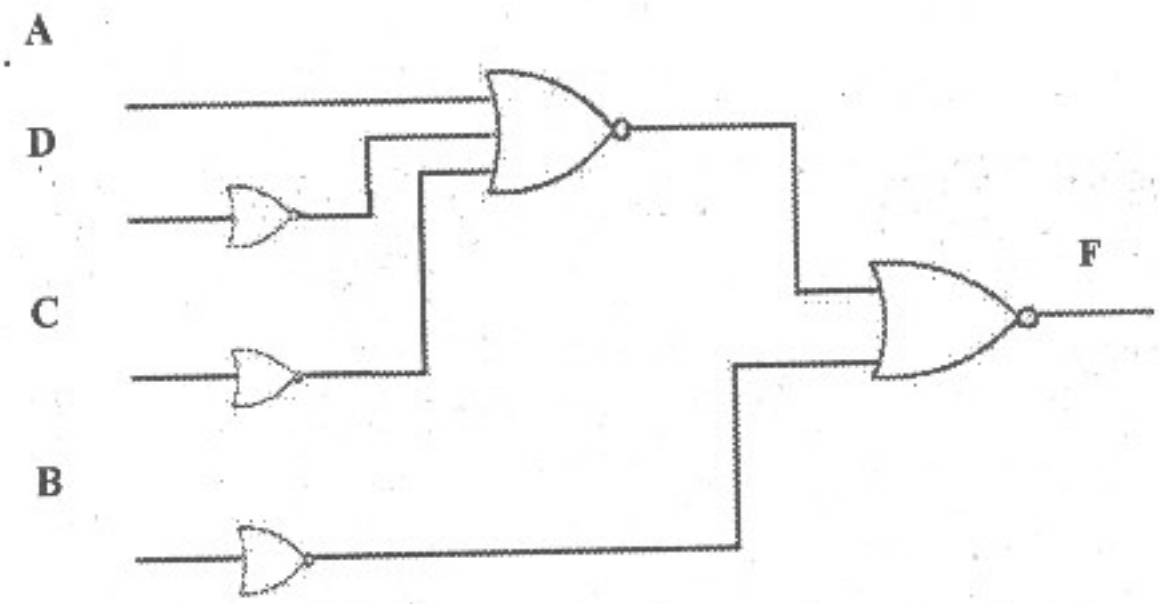
مثال:

تابع بول F را همراه با حالات بی‌اهمیت d با کمتر از دو گیت NOR پیاده‌سازی کنید.

$$F(A,B,C,D) = \sum(0,1,2,9,11)$$
$$d(A,B,C,D) = \sum(8,10,14,15)$$

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

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



مثال:

با استفاده از نقشه چهار متغیره توابع زیر را ساده نمائید.

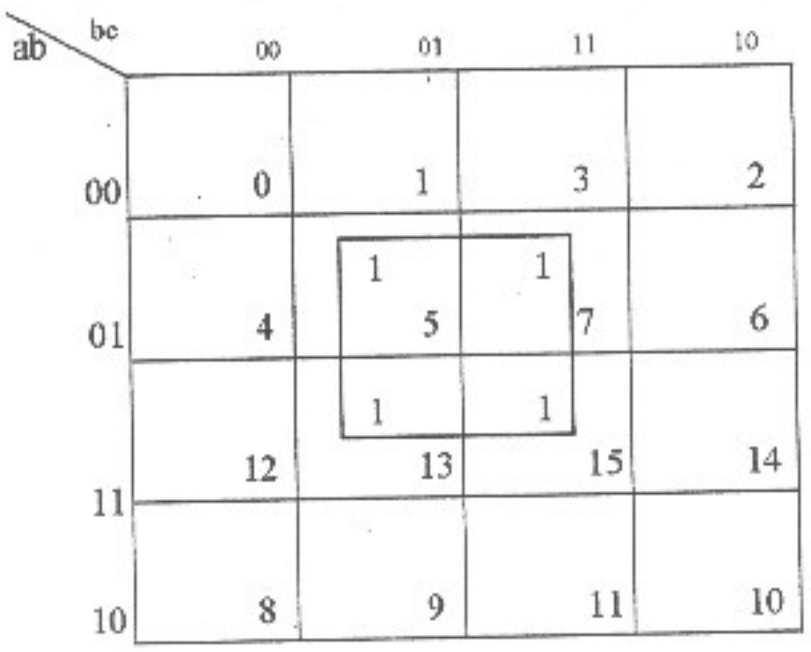
$$F(A,B,C,D) = \sum(0,2,4,5,6,7,8,10,13,15)$$
$$F = BD + A'B + B'D'$$

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

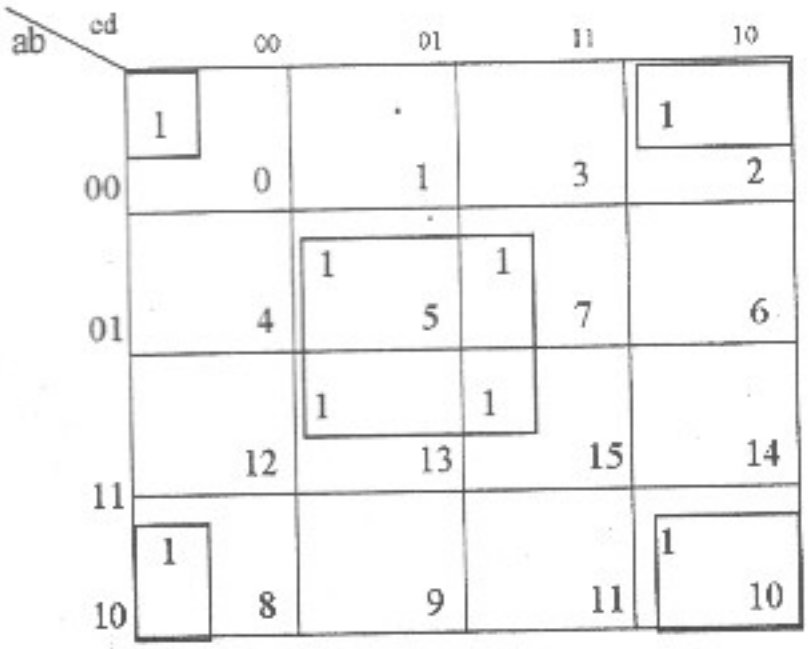
$$F = AB' + B'C' + B'D'$$
$$F = B'(A + C' + D')$$

مجموعه مسائل فصل دوم:

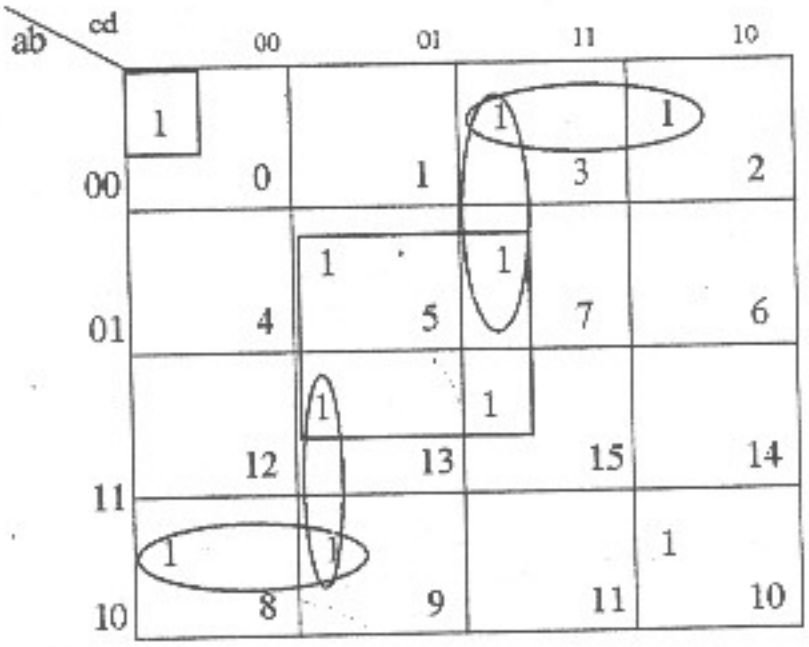
۱- کارنوهای زیر را ساده کنید.



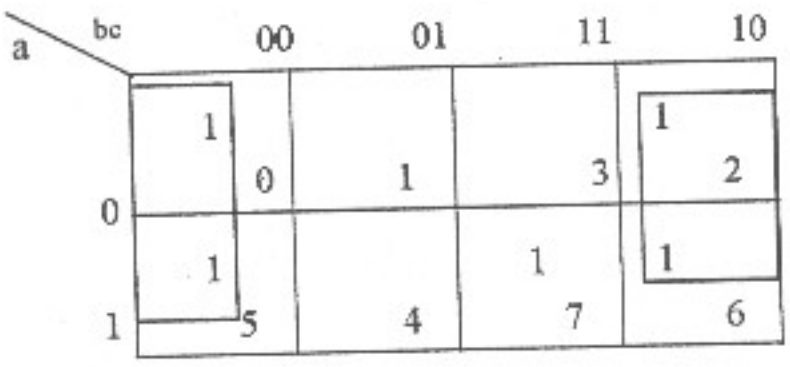
$f = bd$



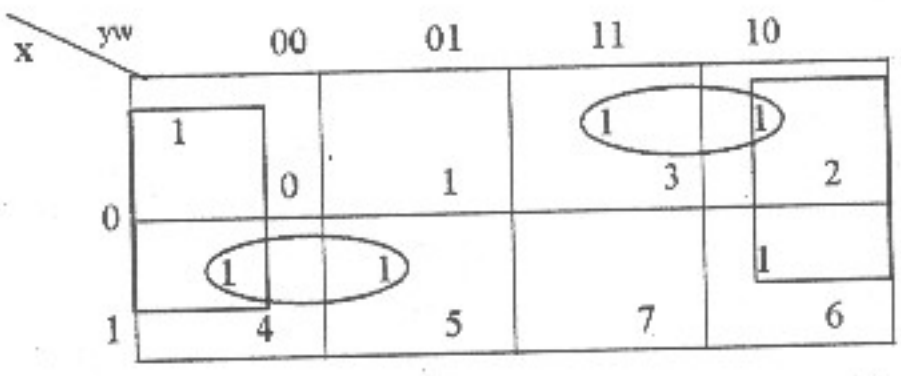
$f = bd + b'd'$



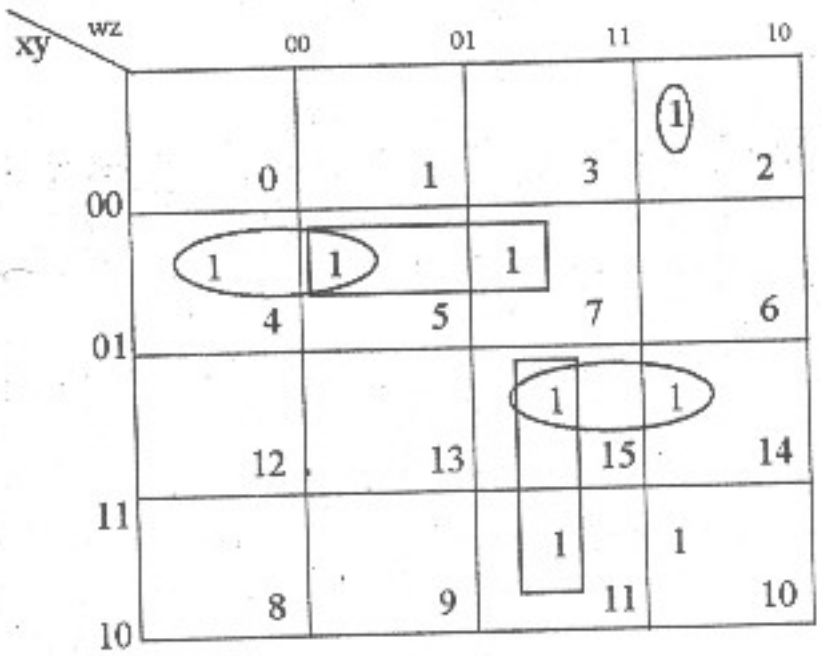
$f = bd + b'd' + a'b'c + ac'd$



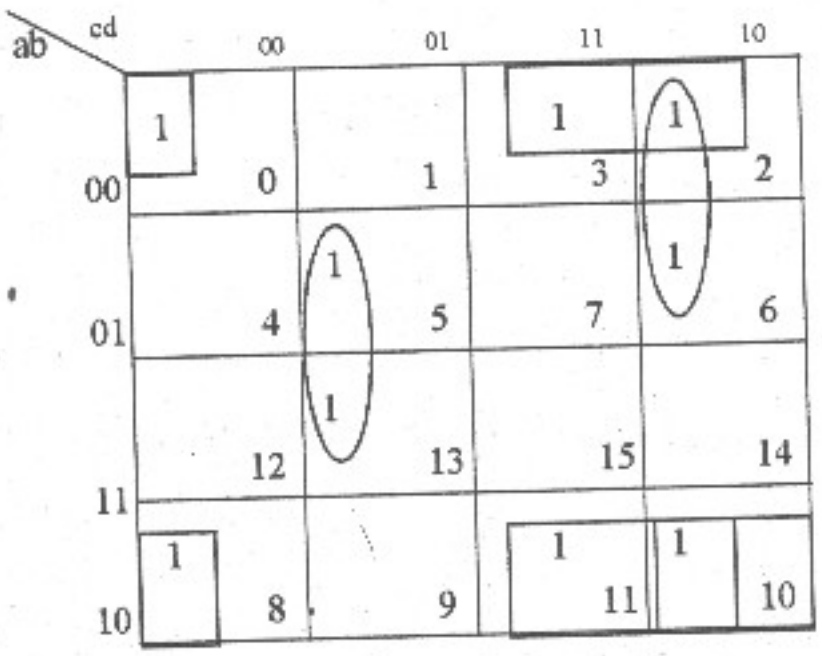
$F = w'$



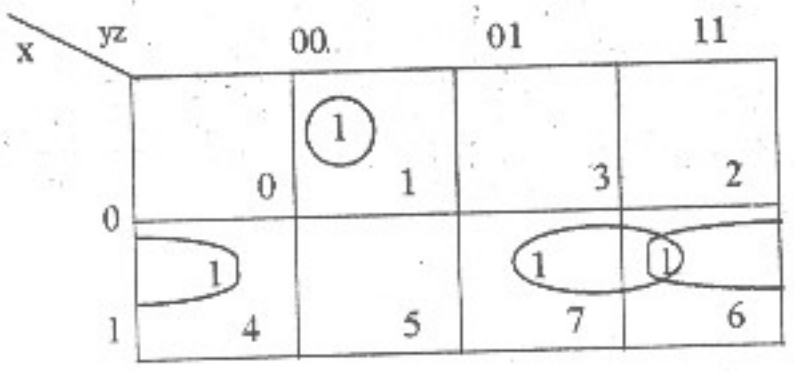
$f = w' + x'y + xy'$



$f = x'yw' + x'yz + xyw + xwz + x'y'wz'$

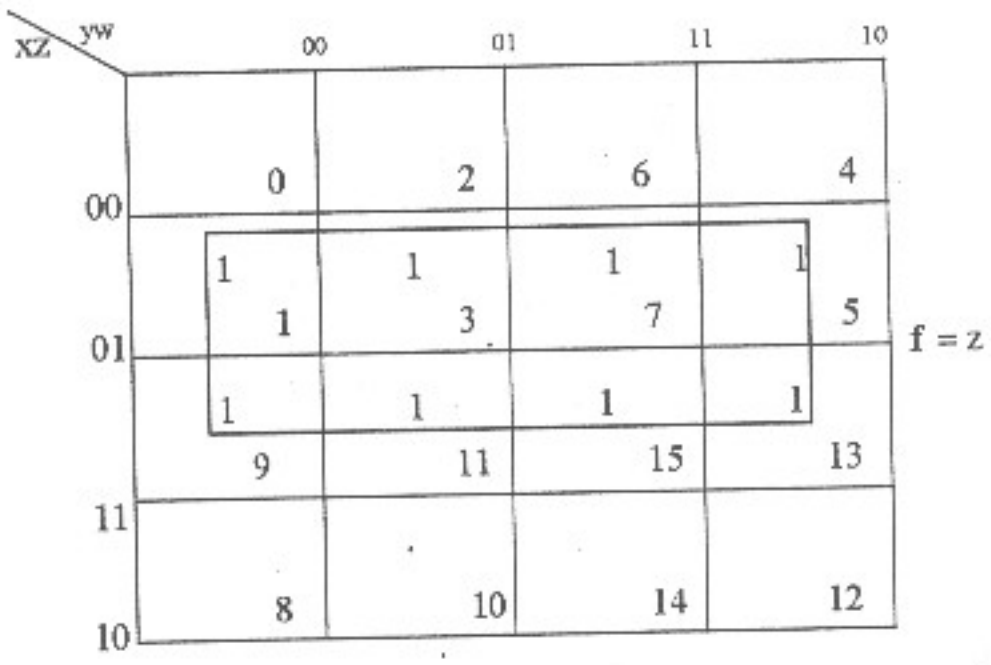


$f = bc'd + b'c + b'd' + a'cd'$



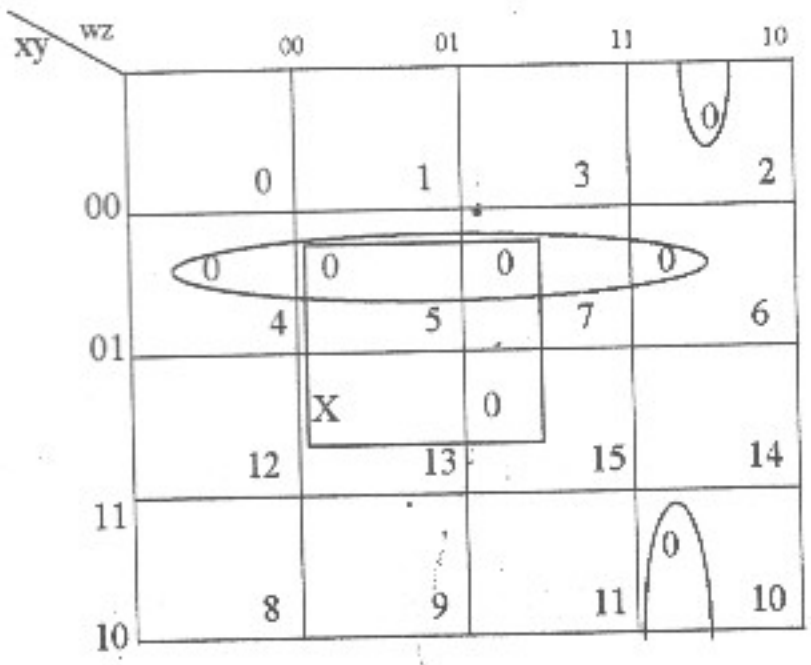
$F = xz' + xy + x'y'z$

۲- به کمک جدول کارنوی زیر تابع $f(x,y,w,z)=\pi(0,2,4,6,8,10,12,14)$ را ساده کنید.



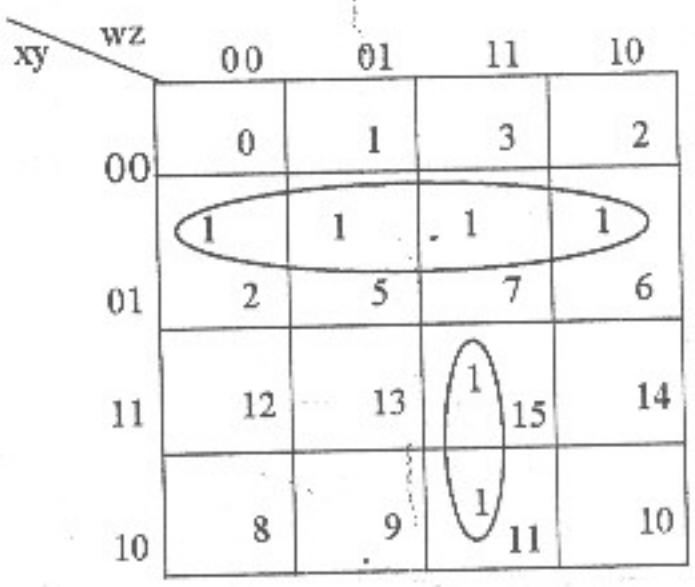
$$f(x,y,w,z)=\pi(0,2,4,6,8,10,12,14)=\Sigma(1,3,5,7,9,11,13,15)$$

۳- جدول کارنوی زیر را به کمک Maxterm ساده کنید.

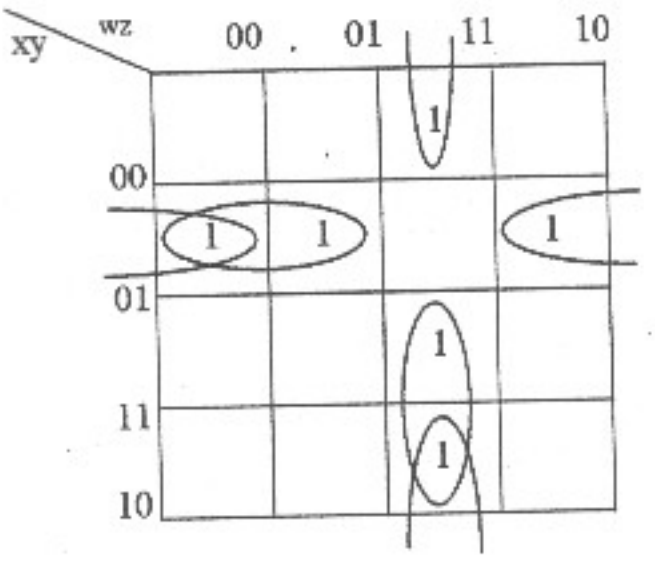


$$f(x,y,w,z)=(x+y').(y'+z').(y+w'+z)$$

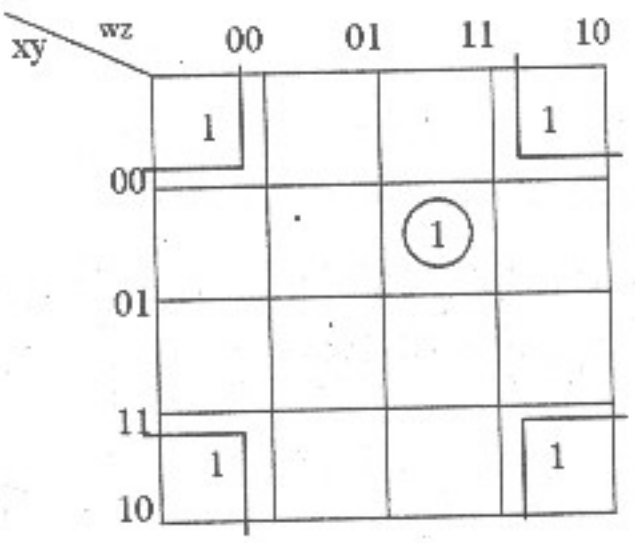
۴- جداول کارنوزیر را همسایه‌گیری کنید.



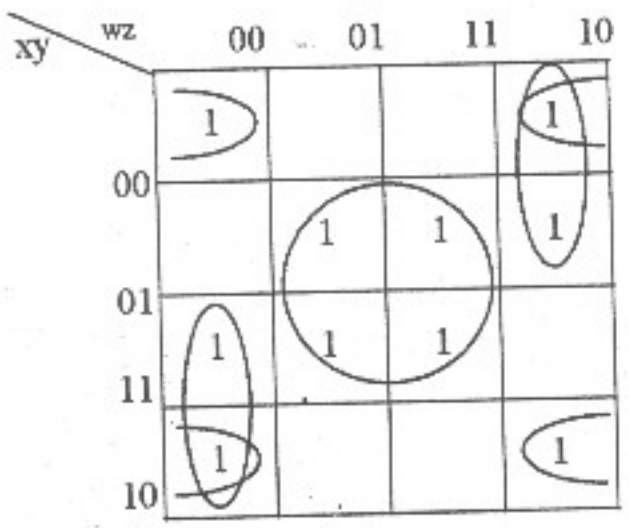
$$F = x'y + xwz$$



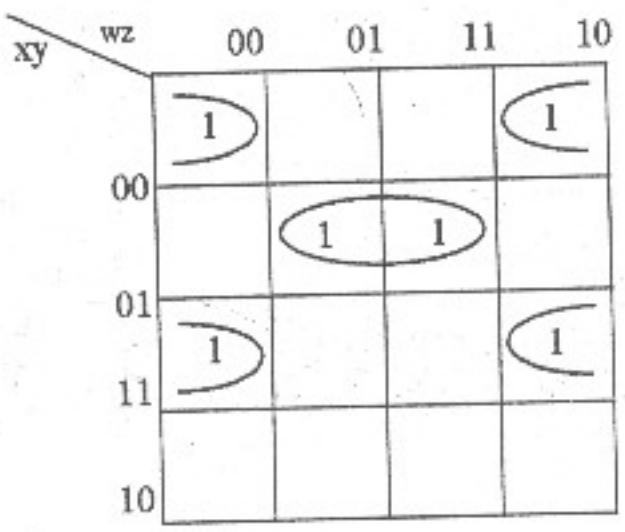
$$F = xwz + x'yw' + x'yz' + y'wz$$



$$F = y'z' + x'ywz$$



$$F = yz + y'z' + x'wz' + xw'z'$$

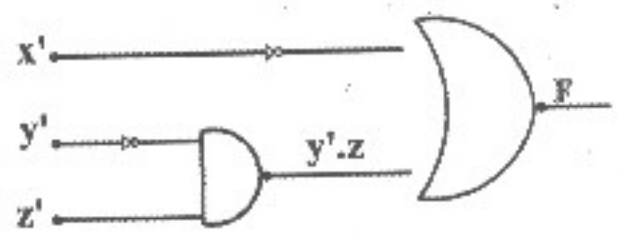


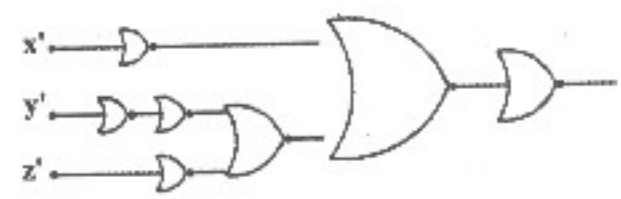
$$F = x'y'z' + x'yz + xyz'$$

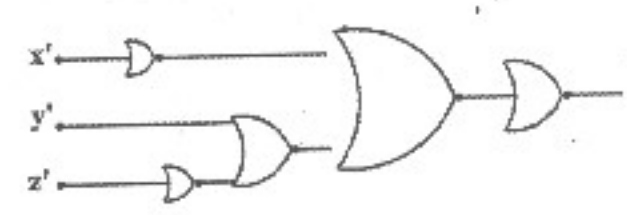
۵- تابع زیر را به صورت تمام NAND و تمام NOR پیاده سازی کنید.

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

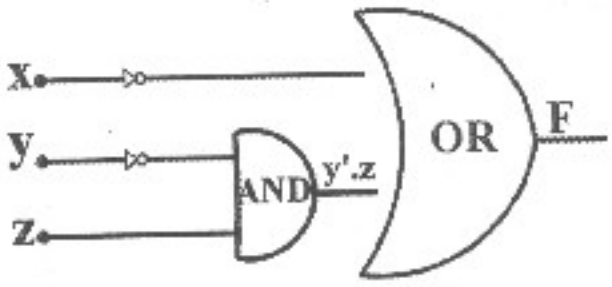
تمام NOR:

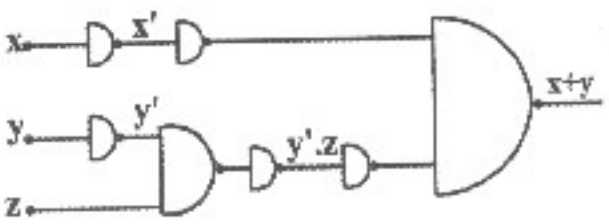


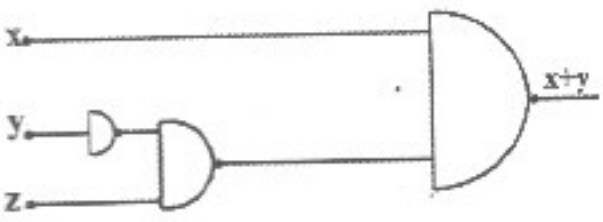




تمام NAND:



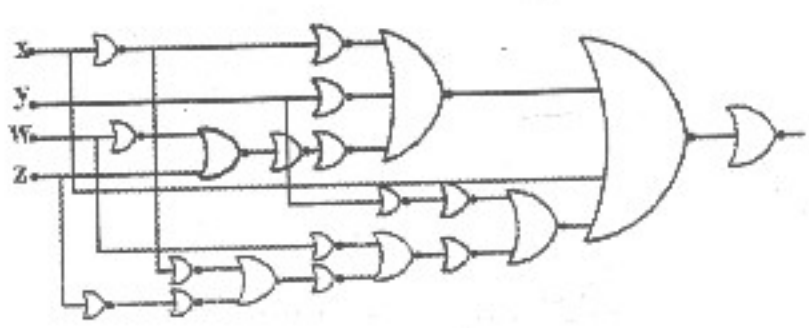




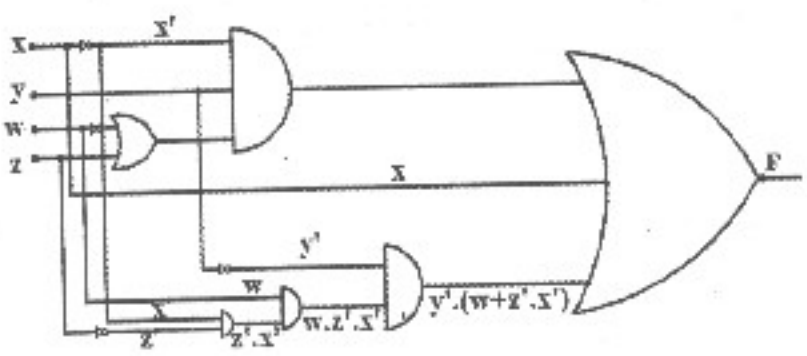
۶- تابع زیر را بصورت تمام NAND پیاده‌سازی کنید و همچنین پیاده‌سازی تمام NOR.

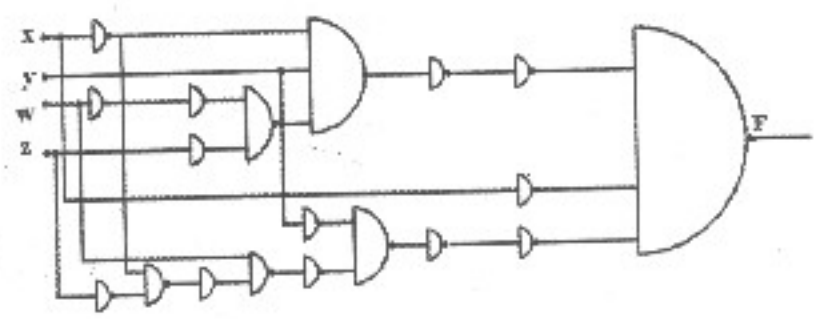
$F = x'y(\omega' + z) + x + y'(\omega.z'.x')$

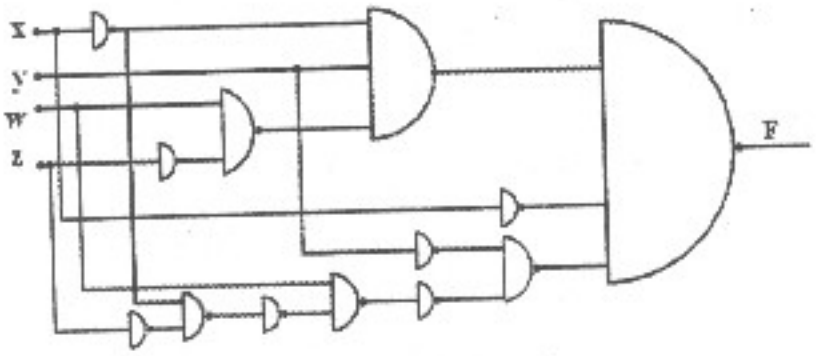
تمام NOR:

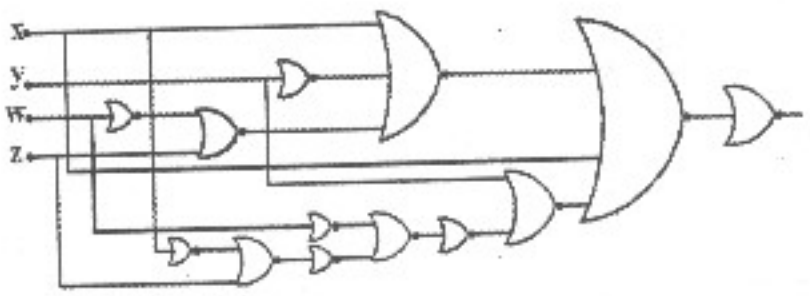


تمام NAND:

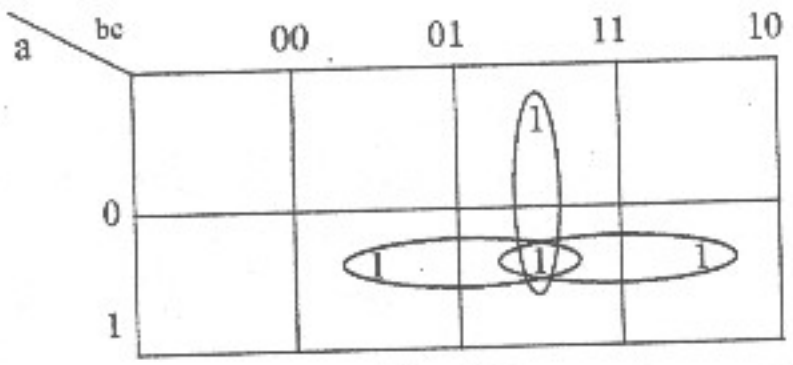




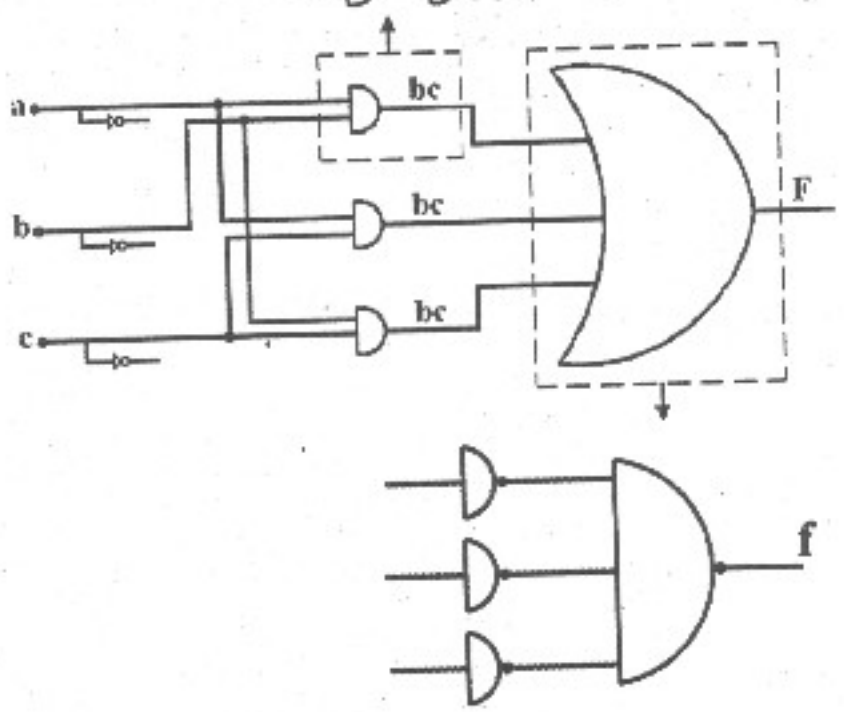


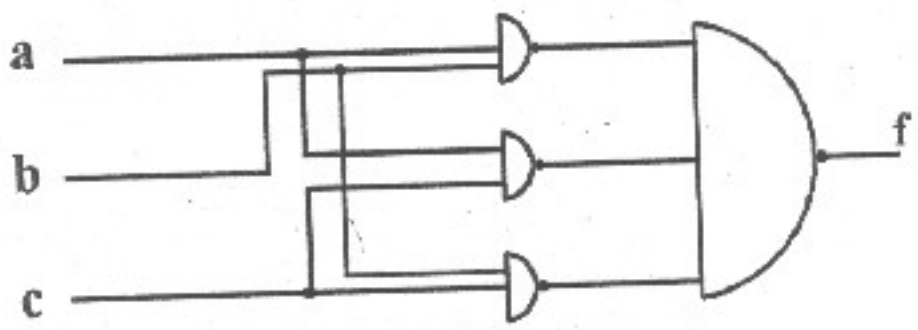


۷- می‌خواهیم مدار مربوط به میز رأی گیری سه نفره را پیاده‌سازی تمام NAND انجام دهیم.



$$F = a\,c + a\,b + b\,c$$

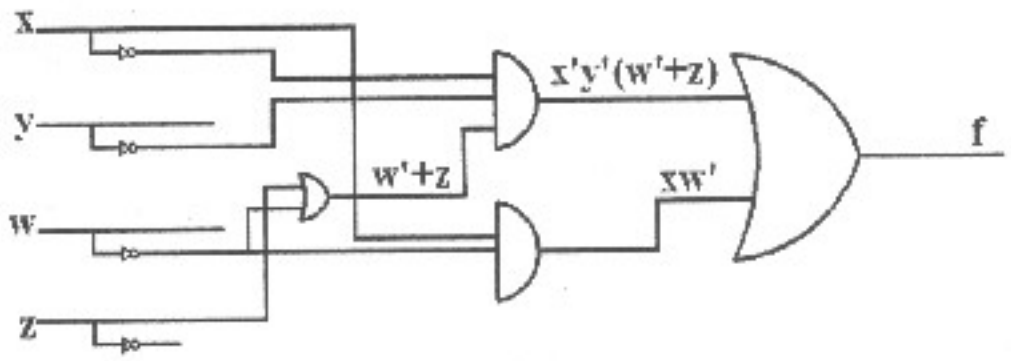


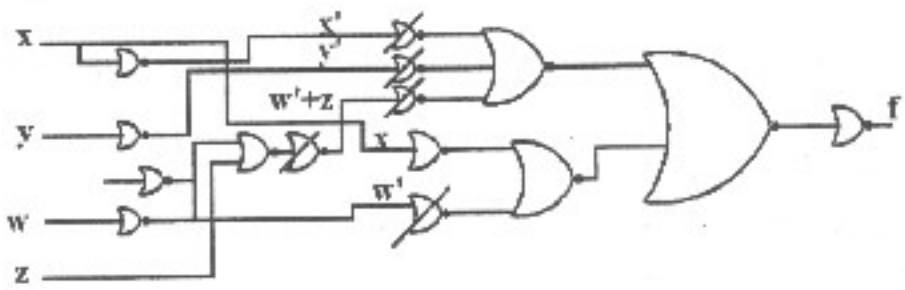


هر جا دو NAND یا NOR یک ورودی پشت سرهم قرار گیرند به جای هر دو سیم قرار می‌دهیم.

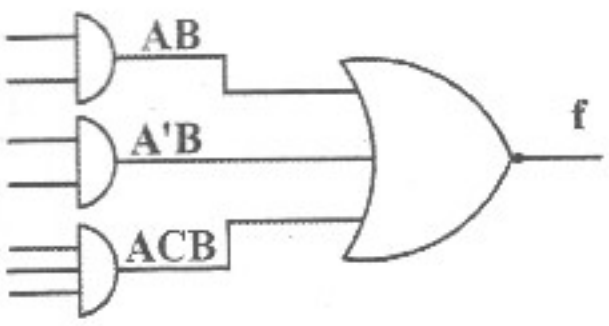
۸- می‌خواهیم تابع زیر را بصورت تمام NOR پیاده‌سازی کنیم.

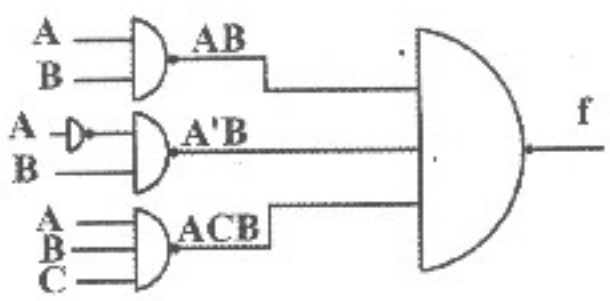
$$F = x'\,y'(w' + z) + x\,w'$$





۹- تابع $F = AB + ACD + A'B$ را به صورت تمام NAND پیاده‌سازی کنید.





۱۰- مداری طراحی کنید که ورودی آن یک رقم BCD و خروجی آن معادل gray ورودی باشد.

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

$X = AB'C'$

AB \ CD	00	01	11	10
00	0	1	3	2
01	1 4	1 5	1 7	1 6
11	X 12	X 13	X 15	X 14
10	1 8	9	X 11	X 10

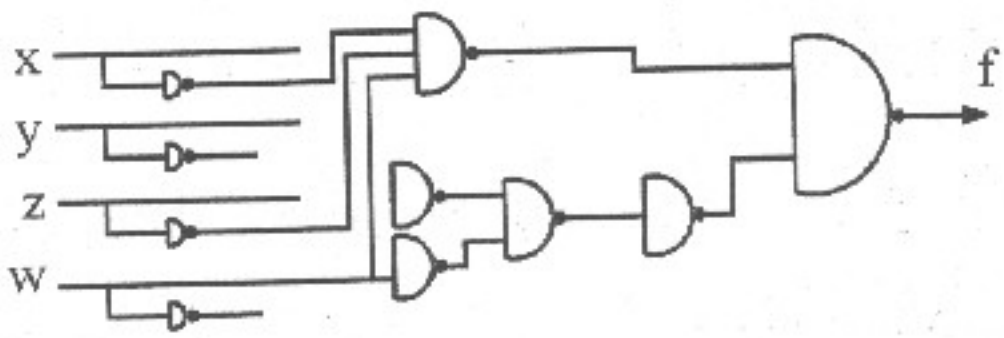
$y = B + A$

AB \ CD	00	01	11	10
00	0	1	1 3	1 2
01	1 4	1 5	7	6
11	X 12	X 13	X 0 15	X 0 14
10	8	9	X 11	X 10

$w = B'C + BC' = B \oplus C$

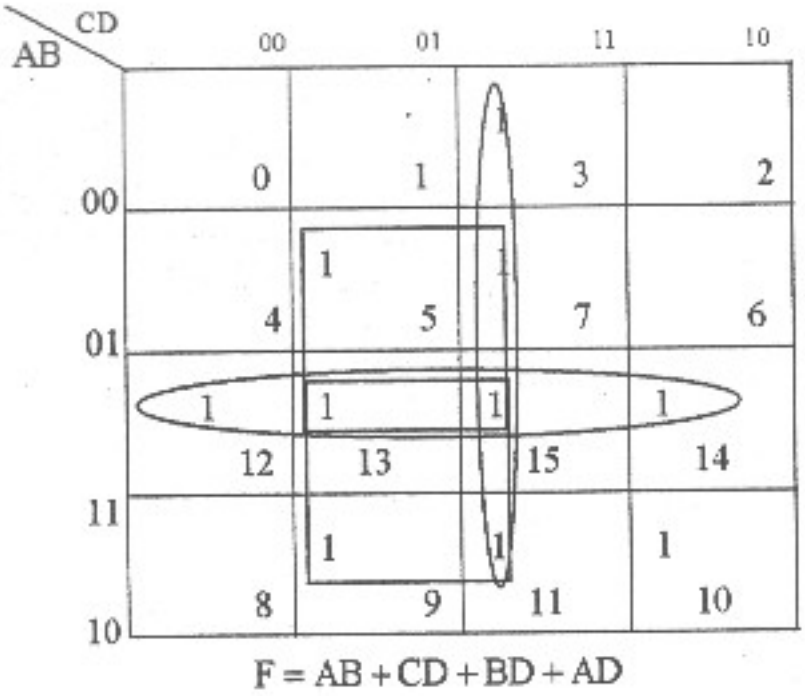
AB \ CD	00	01	11	10
00	0	1 1	3	2
01	1 4	5	7	6
11	X 12	X 13	X 0 15	X 0 14
10	8	9	X 11	X 10

$Z = C'D + CD' = C \oplus D$

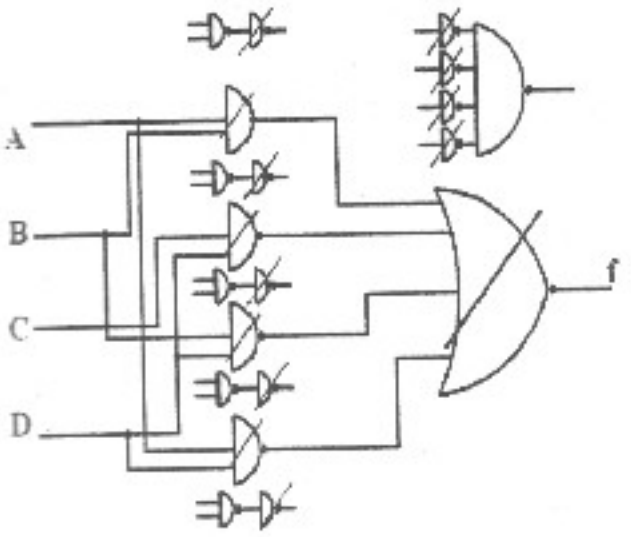


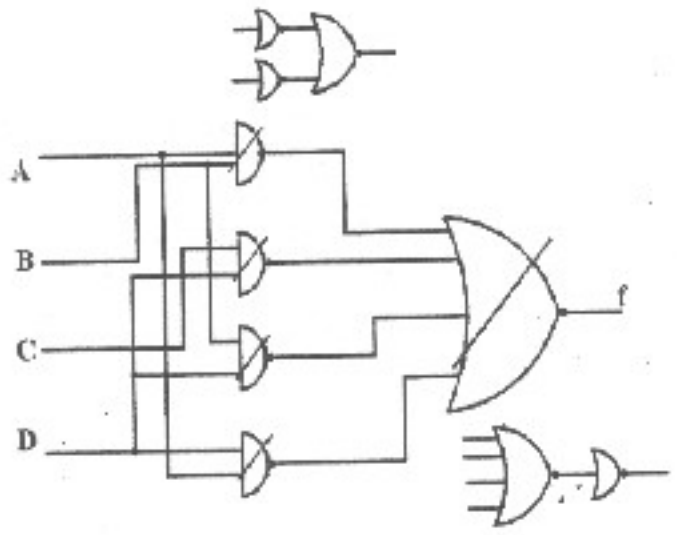
۱۱- مداری طراحی کنید که یک میز رأی‌گیری چهارنفره را شبیه‌سازی کند. به صورتی که نفر اول دارای سه حق رأی، نفر دوم دارای چهار حق رأی، نفر سوم دو حق رأی، نفر چهارم دارای پنج حق رأی باشد. خروجی مدار در زمانی یک است که مجموع حق رأی‌ها 7 یا بیشتر باشد.

	^3A	^4A	^2C	^5D	F
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1



$F = AB + CD + BD + AD$





۱۲- تابع F را به صورت minterm و maxterm بنویسید:

	a	b	c	d	F
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
A	1	0	1	0	0
B	1	0	1	1	1
C	1	1	0	0	1
D	1	1	0	1	0
E	1	1	1	0	1
F	1	1	1	1	1

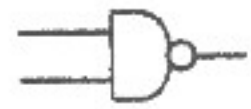
$F(a,b,c,d) = \Sigma(1,4,7,11,12,14,15) = a'b'c'd + a'bc'd' + a'bcd + ab'cd + abc'd' + abcd' + abcd$

$F(a,b,c,d) = \Pi(0,,2,3,5,6,8,9,10,13) = (a+b+c+d).(a+b+c'+d).(a+b+c'+d').(a+b'+c+d').$
$(a+b'+c'+d).(a'+b+c+d).(a'+b+c+d').(a'+b+c'+d).(a'+b'+c+d')$

مجموعه تست‌های فصل دوم:

۱- در کدامیک از گزینه‌های زیر، دروازه‌های منطقی معادل نیستند؟

۱)  معادل با 

۲)  معادل با 

۳)  معادل با 

۴)  معادل با 

۲- کدامیک از گزینه‌های زیر قوانین حاکم بر مدارهای منطقی را نقض می‌کنند؟

۱) $A.B + A'.C + B.C = A.B + A'.C$

۲) $A' + A = 1$

۳) $A + A'.B = A + B$

۴) $A.B + A.C = A + B.C$

۳- درکدامیک از گزینه‌های زیر ساده‌ترین رابطه برای پیاده کردن جدول کارنو مقابل، در قالب یک مدار منطقی را نشان می‌دهد؟

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

۱) $A'B + BD + B'C'D' + AC'D$

۲) $A'B + BD + A'C'D' + AB'C'$

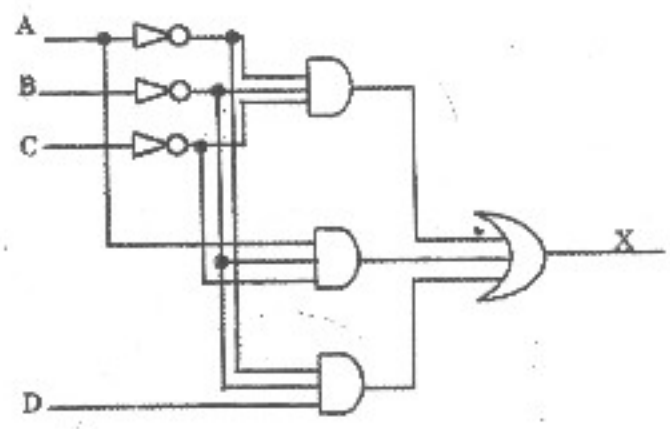
۳) $A'B + BD + B'C'D' + AB'C'$

۴) $A'B + BD + C'D' + B'C'$

۱۳- ساده شده تابع $F = A(B+\bar{C}A)+(BC)(B+\bar{C}A)$ کدام است؟

۱) $AB+A\bar{C}+BC$ ۲) $\overline{AB}+A\bar{C}+BC$ ۳) $\bar{A}\bar{B}+\bar{A}C+\bar{B}C$ ۴) $AB+A\bar{C}+\bar{B}C$

۱۴- خروجی مدار شکل مقابل کدام است؟



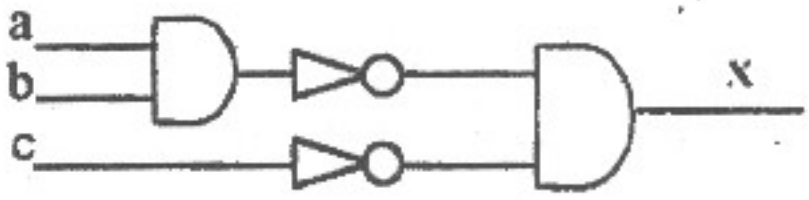
۱) $\overline{ABC} + A\overline{BC} + \overline{AB}C$

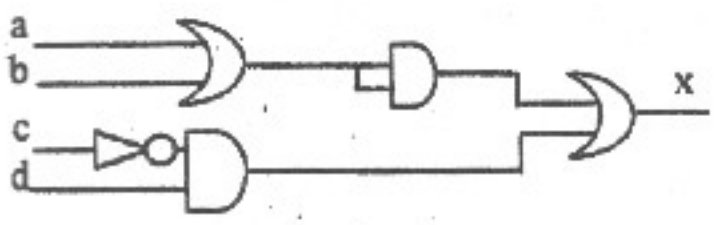
۲) $\bar{A}\bar{B}\bar{C} + \overline{ABC} + \overline{AB}D$

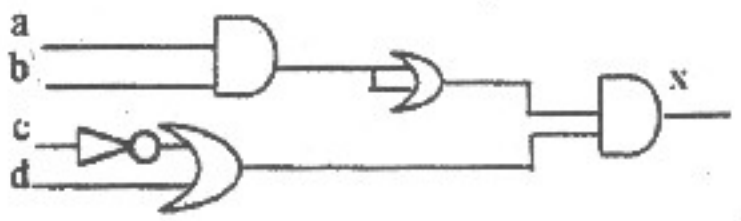
۳) $\overline{A\bar{B}C} + A\bar{B}\bar{C} + \bar{A}\bar{B}D$

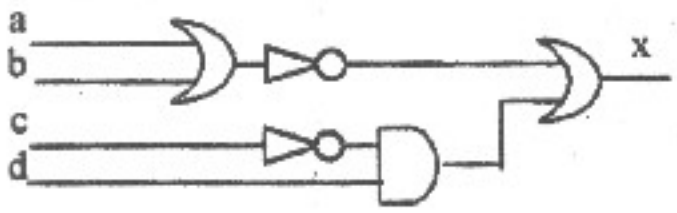
۴) $ABC + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}D$

۱۵- مدار معادل رابطه $X = \overline{(a+b+\bar{c}d)}$ کدام می‌باشد؟

۱)


۲)


۳)


۴)


۱۶- برای تابع $f(A,B,C,D,E) = \sum m(1,4,9,11,13,15,17,19,22,25,27,29,30,31) + \sum d(3,12,20)$ (d حالت بی اهمیت است) کدام گزینه ساده شده زیر صحیح است:

۱) $f(A,B,C,D,E) = \bar{B}\bar{E} + \bar{C}E + \bar{A}CD\bar{E} + ACD\bar{E}$

۲) $f(A,B,C,D,E) = BE + \bar{C}E + \bar{A}CD\bar{E} + ACD\bar{E}$

۳) $f(A,B,C,D,E) = BE + \bar{C}E + \bar{A}CD\bar{E} + \bar{A}CD\bar{E}$

۴) $f(A,B,C,D,E) = \bar{B}\bar{E} + \bar{C}E + ACD\bar{E} + ACDE$

۱۷- برای یک مدار مبدل که جدول درستی آن در زیر داده شده است برای خروجی x کدام پاسخ صحیح است؟

D	F	E	D	C	B	A	Z	J	X
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	1	0	1	0
0	0	0	0	1	1	1	0	1	1
0	0	0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	1
0	1	1	1	1	1	1	1	1	0
1	1	1	1	1	1	1	1	1	1

۱) $x = \overline{(A \odot B \odot C)}$

۲) $x = A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G$

۳) $X = A \oplus B \odot C \oplus D$

۴) $x = B \oplus D \oplus C$

۱۸- کدامیک از حاصلضرب maxtermها نمایش Canonical یا Standard تابع زیر می‌باشد.

$f(W,,X,Y,Z) = X\bar{Y} + \bar{W}Y + WXY$

۱) $\Pi M(0,1,8,9,10,11)$

۲) $\Pi M(0,1,2,3,7,12,13,15)$

۳) $\Pi M(1,3,4,5,7,12,13,14,15)$

۴) $\Pi M(0,1,2,6,8,9,11)$

۱۹- برای تابع بولی 5 متغیره زیر ساده‌ترین صورت ضرب حاصل جمع‌ها کدام است؟

$$F(A,B,C,D,E) = \Pi(0,8,9,10,12,16,17,25,29,), d(2,7,13,23)$$

۱) $(A+\bar{B})(A+\bar{C})(A+\bar{D}+E)(\bar{B}+\bar{C}+E)(B+\bar{C}+\bar{E})$

۲) $(A+C+E)(A+\bar{B}+D)(\bar{A}+B+C+D)(\bar{B}+D+\bar{E})$

۳) $(D+B+E)(A+C+\bar{D}+\bar{E})(\bar{A}+C+E)(\bar{A}+C+D)$

۴) $(A+\bar{B})(C+\bar{E})(\bar{A}+C+\bar{D}+\bar{E})(\bar{A}+C+D+\bar{E})$

۲۰- ساده‌ترین صورت تابع مقابل کدام است؟

$$F(A,B,C,D,E) = \sum(0,2,6,8,9,11,13,15,16,18,22) + d(4,10,12,14,20,26,28,30)$$

۱) $F = A'E' + A'B + AB'E'$

۲) $F = A'E' + AB'D'E' + A'B + ADE$

۳) $F = A'E' + AB'$

۴) $F = A'B + B'E'$

۲۱- در تابع بولی که دارای ترم‌های زیر می‌باشد، ساده‌ترین فرم حاصل جمع حاصلضرب‌ها کدام است؟

$$f(A,B,C,D,E) = \sum m(0,2,4,9,11,12,14,18,20,21,27,29), d(6,10,16,22,25)$$

۱) $\bar{A}D\bar{C} + A\bar{B}D + BCD + B\bar{C}\bar{D} + B\bar{C}E$

۲) $AC\bar{D} + A\bar{B}C + \bar{A}D\bar{C} + B\bar{C}\bar{E}$

۳) $ABC + AB\bar{D} + BC\bar{D} + BC\bar{E} + ABE$

۴) $\bar{B}\bar{E} + B\bar{C}E + \bar{A}C\bar{E} + AC\bar{D}E$

۲۲- برای تابع بولی 5 متغیرهٔ زیر ساده‌ترین صورت حاصل جمع حاصلضرب‌ها کدام است؟

$$F(A,B,C,D,E) = \sum(0,3,8,14,15,16,18,24,26,27,29), D(6,7,9,19,22)$$

۱) $\bar{C}\bar{D}\bar{E} + A\bar{B}C + \bar{A}\bar{B}\bar{C}E + ABCD\bar{E}$

۲) $\bar{A}CD + \bar{C}\bar{D}E + \bar{B}\bar{C}D + \bar{A}\bar{B}DE + ABC\bar{D}E$

۳) $\bar{A}CD + \bar{C}\bar{D}\bar{E} + A\bar{C}D + \bar{A}\bar{B}\bar{C}E + ABC\bar{D}E$

۴) $\bar{A}CD + A\bar{C} + \bar{A}\bar{B}\bar{C}\bar{E} + ABCD\bar{E} + \bar{C}\bar{D}E$

۲۳- تابع $F = \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$ برحسب حاصل ضرب مجموعه‌ها کدام است؟

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

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

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

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

۲۴- صورت مینیمم تابع زیر کدام است؟

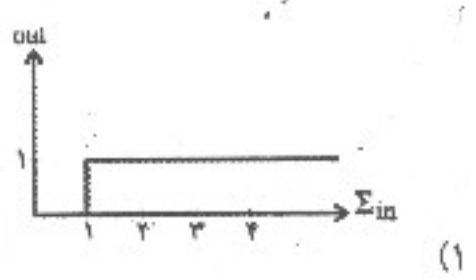
$$F(X_1, X_2, X_3) = \bar{X}_1\bar{X}_2\bar{X}_3 + X_1\bar{X}_2\bar{X}_3$$

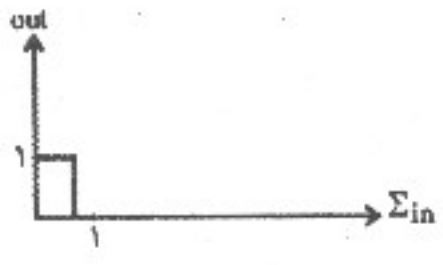
۱) $\bar{X}_1\bar{X}_2\bar{X}_3$ ۲) $\bar{X}_2\bar{X}_3$ ۳) $X_1\bar{X}_2\bar{X}_3$ ۴) $\bar{X}_1 + X_1$

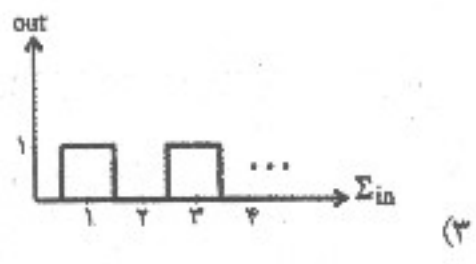
۲۵- در منطق منفی هر دروازه، OR به چه دروازه منطقی در منطق مثبت تبدیل می‌شود؟

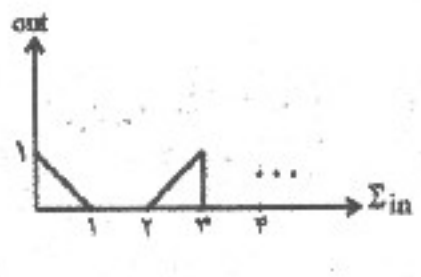
۱) AND ۲) NOR ۳) NAND ۴) XNOR

۲۶- کدامیک از اشکال زیر یک XOR را مشخص می‌سازد؟ (Σ_{in} جمع ورودی‌ها را نشان می‌دهد و out خروجی مدار است)

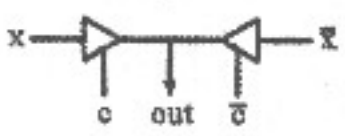
۱)


۲)


۳)


۴)


۲۷- کدام عملیات مربوط به مدار زیر است؟



۱) CX ۲) $\bar{C}\bar{X}$ ۳) $C \oplus X$ ۴) $\overline{C \oplus X}$

۲۸- ساده شده جدول کارنوی زیر کدام است؟

CD \ AB	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	d	d	d	d
10	1	1	d	d

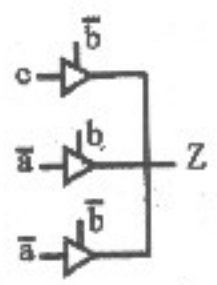
۱) $A' + AD + BC$

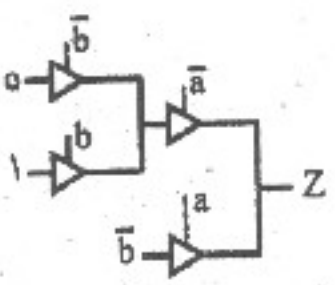
۲) $A' + AD' + BD'$

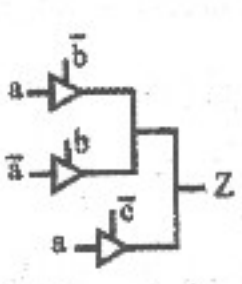
۳) $C + AD' + BD$

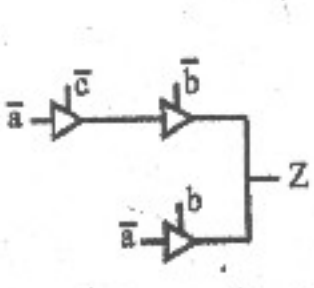
۴) $C + AD + BD$

۲۹- با استفاده از گیت‌های سه حالته کدام مدار دارای خروجی $z(a \oplus b) + \bar{a}c$ می‌باشد؟

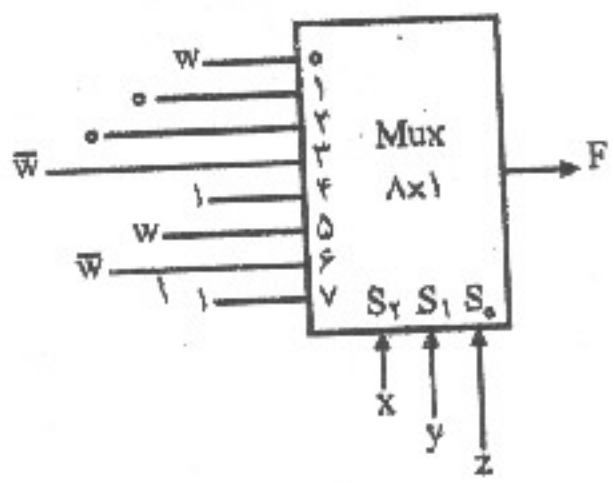
۱)


۲)


۳)


۴)


۳۰- مدار زیر، پیاده‌سازی کدامیک از روابط زیر است؟



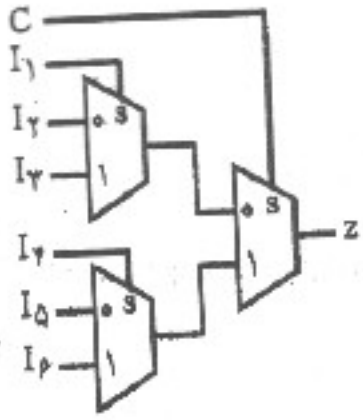
۱) $F=(w,x,y,z)=\Sigma(0,1,3,5,7,11,14)$

۲) $F=(w,x,y,z)=\Sigma(1,3,6,7,8,11,12)$

۳) $F=(w,x,y,z)=\Sigma(3,4,6,7,8,12,13,15)$

۴) $F=(w,x,y,z)=\Sigma(3,5,7,9,11,13,15)$

۳۱- در شکل زیر با ارتباط دادن ورودی‌های a و b به خطوط I_1,I_2,I_3,I_4,I_5,I_6 مدار را به گونه‌ای بسازید که خروجی Z معادل تابع $Z=abc+a\bar{c}+b\bar{c}$ شود؟



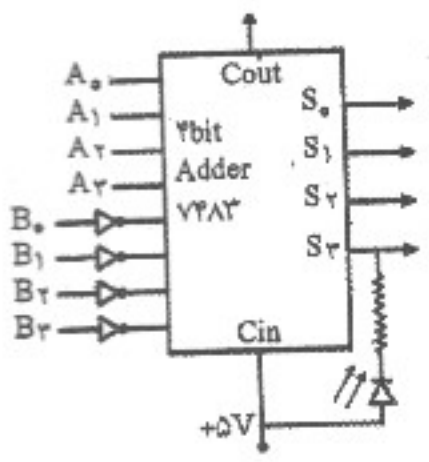
۱) $I_1I_2I_3I_4I_5I_6=ab0\ ba1$

۲) $I_1I_2I_3I_4I_5I_6=ab1\ a0b$

۳) $I_1I_2I_3I_4I_5I_6=abb\ 1ab$

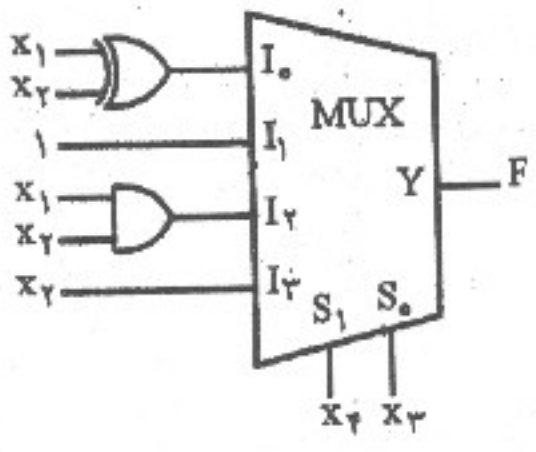
۴) $I_1I_2I_3I_4I_5I_6=01ab10$

۳۲- به فرض اینکه اعداد A و B در سیستم مکمل 2 (2's Complement) بین 7- و 7 + هستند در مدار زیر روشن شدن LED بیانگر چیست؟ A و B هم علامت هستند.



۱) $A<B$ ۲) $A\leq B$ ۳) $A=B$ ۴) $A\geq B$

۳۳- تابع خروجی برای مدار روبرو به چه صورت است؟



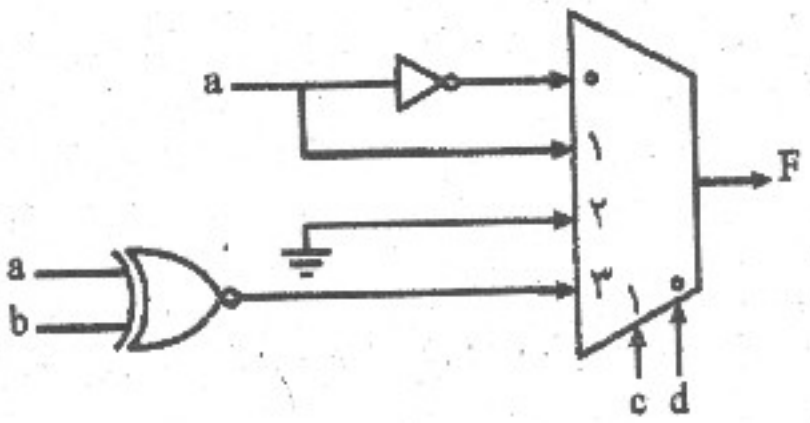
۱) $F(x_4,x_3,x_2,x_1)=\sum m(1,2,4,5,6,7,11,14,15)$

۲) $F(x_4,x_3,x_2,x_1)=\sum m(1,4,5,9,10,11,12,13,15)$

۳) $F(x_4,x_3,x_2,x_1)=\sum m(0,3,4,5,6,7,10,11,15)$

۴) $F(x_4,x_3,x_2,x_1)=\sum m(1,2,3,4,5,7,11,13,15)$

۳۴- مدار زیر، پیاده‌سازی کدام رابطه است؟



۱) $f(a,b,c,d)=\Sigma 0,1,3,5,7$

۲) $f(a,b,c,d)=\Sigma 1,3,5,7,11,15$

۳) $f(a,b,c,d)=\Sigma 0,3,4,9,13,15$

۴) $f(a,b,c,d)=\Sigma 0,3,5,7,13,15$

فصل سوم

مدارهای ترکیبی

پایه‌ترین مبحث در مورد مدارهای منطقی و یا شاید هم اصلی‌ترین بحث همین مدارهای ترکیبی می باشند. همانطور که از اسم آنها به نظر می‌رسد این مدارات از ترکیب گیتهای متفاوت ایجاد شده اند که یک عملکرد خاص را انجام می‌دهند.

در واقع طراحی مدارات ترکیبی با استفاده از مطالب گفته شده در فصول یک و دو انجام می‌شود. بنابراین قبل از شروع این فصل حتماً فصول قبل به صورت کامل مطالعه شود.

الگوریتم برای طراحی مدارهای ترکیبی:

۱ـ بدست آوردن جدول درستی از روی صورت مسئله

۲ـ بدست آوردن توابع بولی از روی جدول درستی

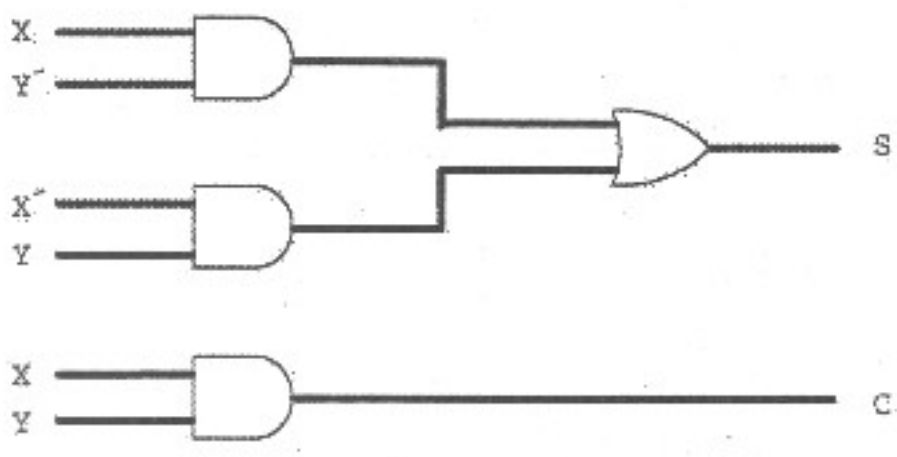
۳ـ ساده کردن در صورت نیاز

۴ـ پیاده‌سازی تابع ساده شده به کمک گیت‌ها

نیم جمع کننده Half Adder:

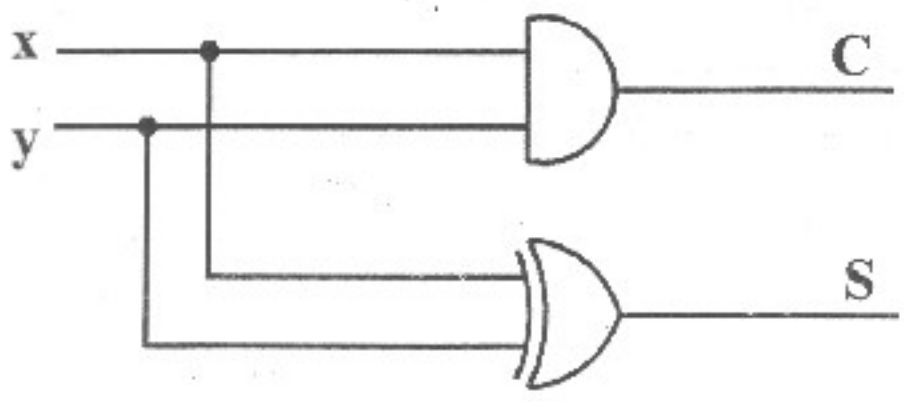
دو بیت را با هم جمع می‌کند حاصل جمع را در S و رقم نقلی را در C نشان می‌دهد یعنی مداری است با دو ورودی و دو خروجی.

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0
		AND	XOR



$S = x \oplus y$

$c = x \times y$



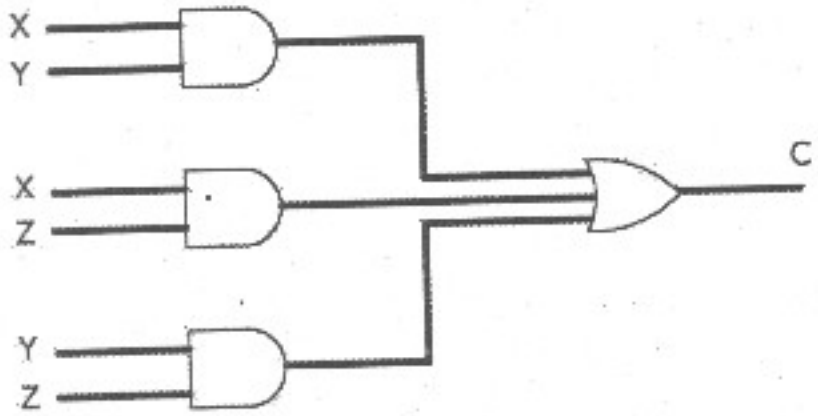
تمام جمع کننده Full Adder:

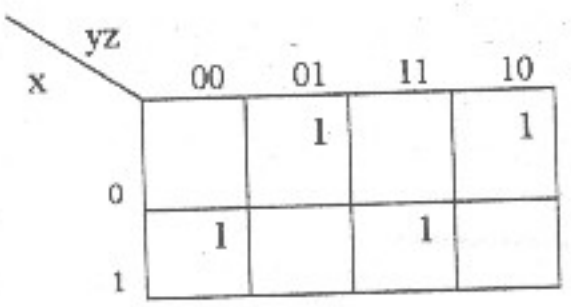
دو بیت ورودی و یک بیت carry مرحله قبل را با هم جمع می‌کند و یک بیت به عنوان حاصل جمع (S) و یک بیت به عنوان Carry یا رقم نقلی (C) را به خروجی منتقل می‌کند. یعنی مداری است با سه ورودی و دو خروجی.

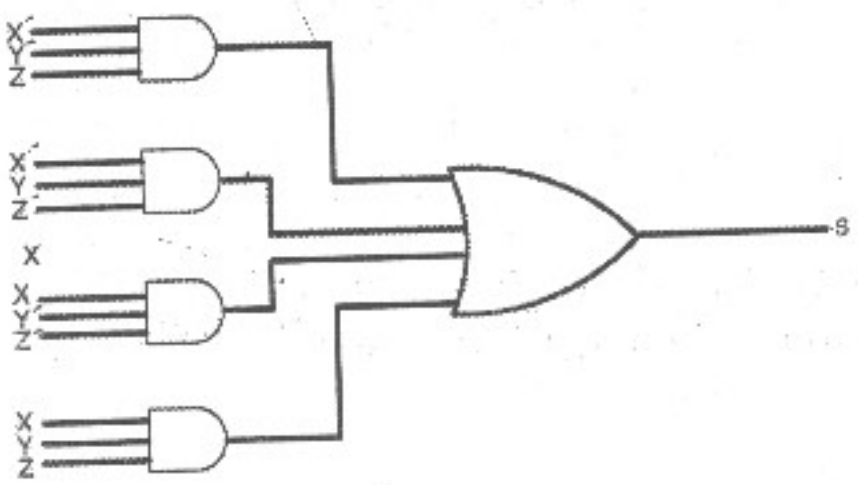
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

$$C = xy + xz + yz$$
$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = xy + xz + yz$$

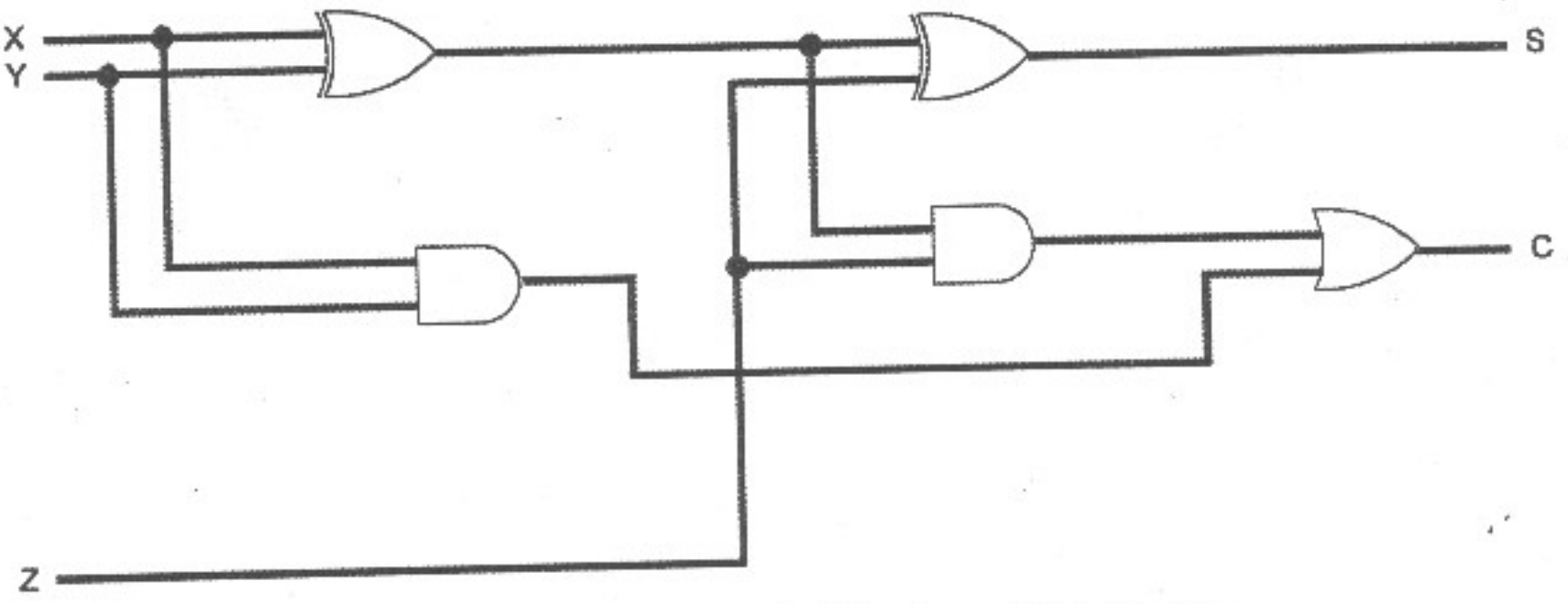




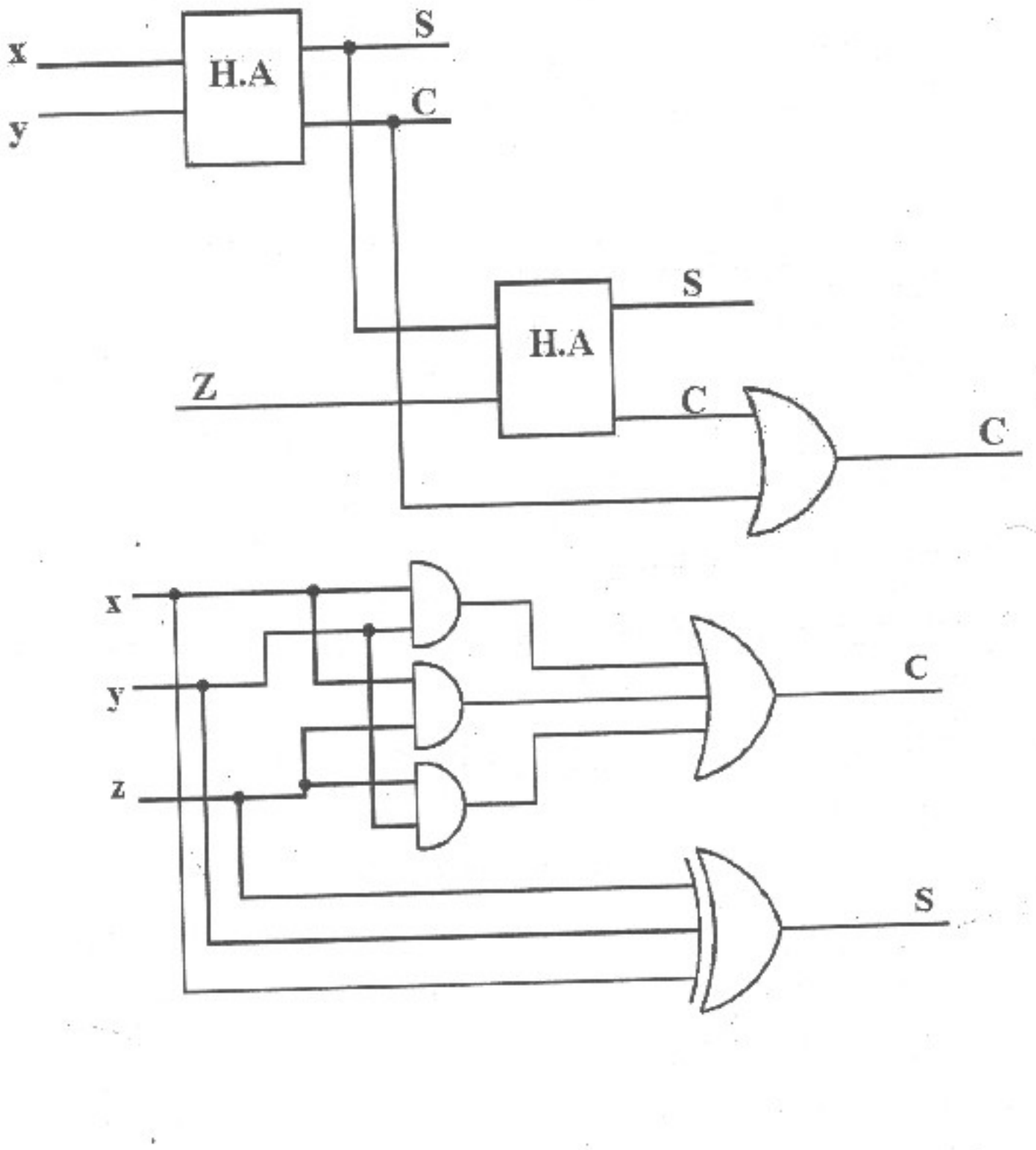


$$S = x'y'z + x'yz' + xy'z' + xyz$$

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



پیاده سازی یک تمام جمع کننده کامل با دو نیم جمع کننده و یک گیت OR



نیم تفریق کننده Half Subtractor:

دو بیت را از هم کم می‌کند، حاصل تفریق را در D و رقم قرضی را در B قرار می‌دهد. یعنی مداری است با دو ورودی و دو خروجی.

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

$B = x'y$

$D = x \oplus y$

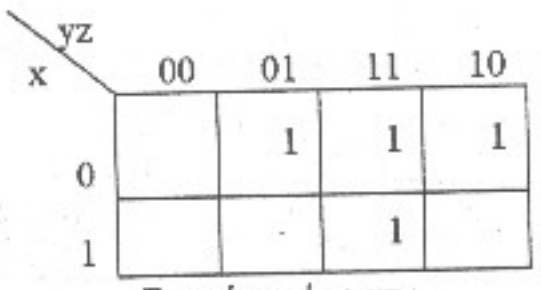
تمام تفریق کننده:

دو بیت و رقم قرضی مرحله قبل را از هم کم می‌کند، حاصل تفریق را در D و رقم قرضی را در B قرار می‌دهد یعنی مداری است با سه ورودی و دو خروجی.

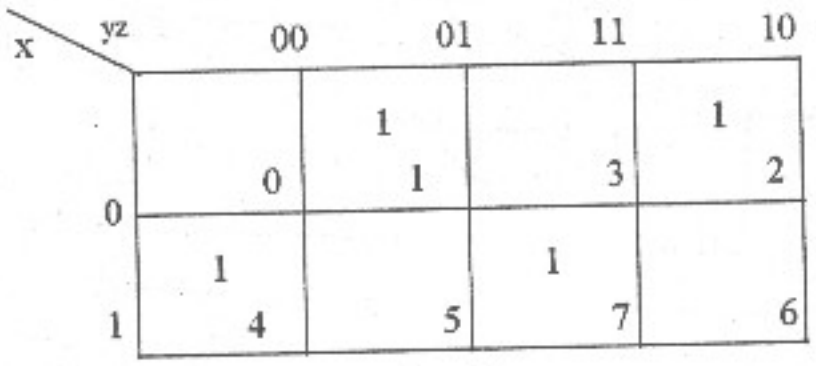
x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

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

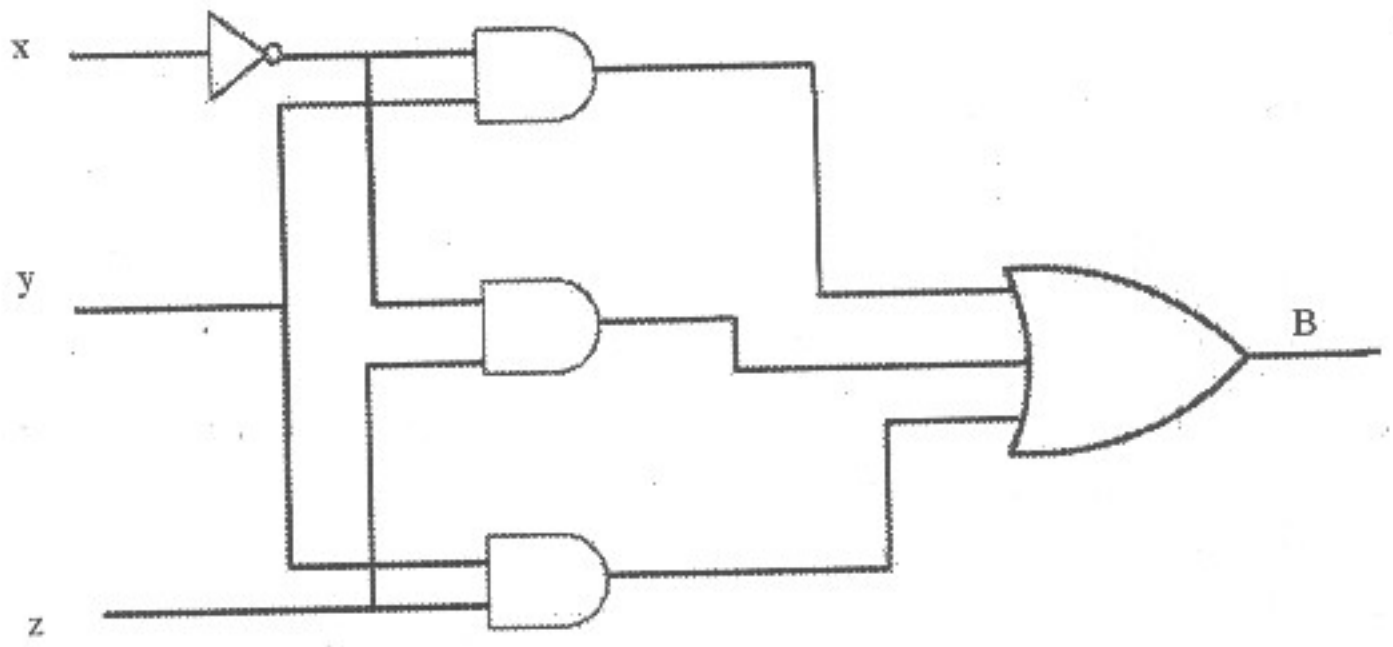
$D = x \oplus y \oplus z$

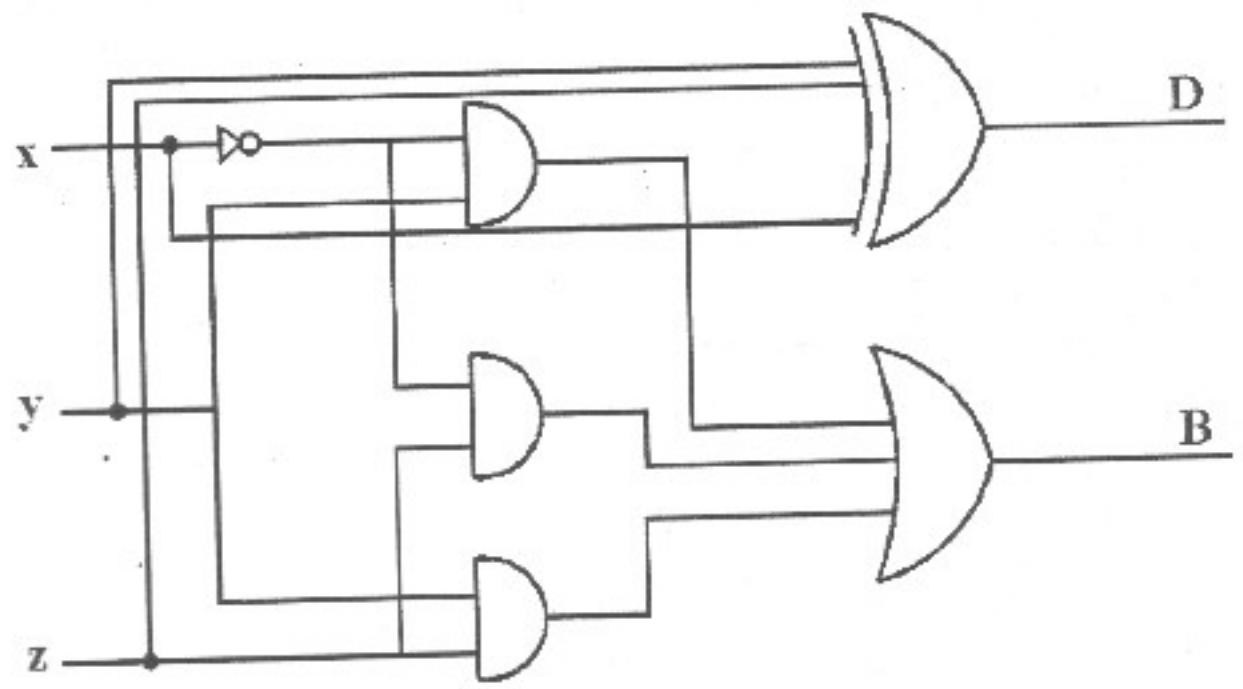


$B = x'z + x'y + yz$



$D = x \oplus y \oplus z$





طراحی مدار تولید کننده بیت Parity:

دلیل استفاده از بیت Parity برای تشخیص خطاست.

Parity فرد اگر تعداد یک‌ها فرد بود به سمت راست صفر اضافه می‌شود و اگر تعداد یک‌ها زوج بود به سمت راست یک اضافه می‌شود.

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

تشخیص خطا با بیت پریتی برای ۳ ورودی

این مدار دارای سه ورودی x و y و z می‌باشد، در واقع اطلاعات رسیده می‌باشند و P بیت پریتی است که فرستنده روی آن گذاشته، حال بررسی می‌کنیم که p گذاشته شده صحیح است اگر درست باشد مقدار C یک است و در غیر اینصورت صفر. به عنوان مثال:

X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

در دوخط اول و دوم X و Y و Z صفر هستند.

پس باید Parity آن ۱ باشد در نتیجه خط دوم درست است.

$C = x \oplus y \oplus z \oplus p$

Parity زوج اگر تعداد یک‌ها فرد بود سمت راست یک اضافه می‌کنیم و اگر تعداد یکها زوج بود سمت راست صفر اضافه می‌کنیم.

Parity فرد و زوج برای دو ورودی

x	y	P فرد	P زوج
0	0	1	0
0	1	0	1
1	0	0	1
1	1	1	0
		XNOR	XOR

نکته: مطالب فوق به این معنی است که اگر parity فرد را برای یک تابع بدست آوریم می‌توانیم با قرار دادن یک گیت Not در خروجی آن را به parity زوج تبدیل کنیم و برعکس.

مدار parity زوج و فرد:

	a	b	c	P فرد	P زوج
0	0	0	0	1	0
1	0	0	1	0	1
2	0	1	0	0	1
3	0	1	1	1	0
4	1	0	0	0	1
5	1	0	1	1	0
6	1	1	0	1	0
7	1	1	1	0	1

a \ bc	00	01	11	10
0	1 (0)	1 (1)	1 (3)	1 (2)
1	1 (4)	0 (5)	1 (7)	0 (6)

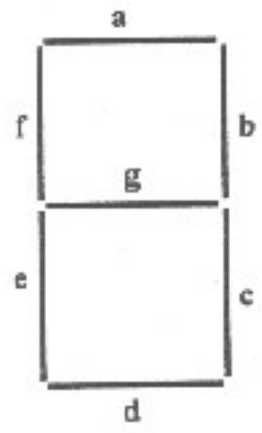
$p = A \odot B \odot C$ فرد

a \ bc	00	01	11	10
0	0 (0)	1 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	1 (7)	0 (6)

$p = A \oplus B \oplus C$ زوج

نمایشگرهای 7-segment:

	xywz	a	b	c	d	e	f	g
0	0000	1	1	1	1	1	1	0
1	0001	0	1	1	0	0	0	0
2	0010	1	1	0	1	1	0	1
3	0010	1	1	1	1	0	0	1
4	0100	0	1	1	0	0	1	1
5	0101	1	0	1	1	0	1	1
6	0110	1	0	1	1	1	1	1
7	0111	1	1	1	0	0	0	0
8	1000	1	1	1	1	1	1	1
9	1001	1	1	1	1	0	1	1



تبدیل کد BCD به 7-Segment معادله تمام حروف با جدول کارنو به همراه گیت

$a = x'y'w'z' + x'y'wz' + x'y'wz + x'yw'z + x'ywz' + x'ywz + xy'w'z' + xy'w'z$

$b = x'y'w'z' + x'y'w'z + x'y'wz' + x'y'wz + x'yw'z' + x'ywz + xy'w'z' + xy'w'z$

$c = x'y'w'z' + x'y'w'z + x'y'wz + x'yw'z' + x'yw'z + x'ywz' + x'ywz + xy'w'z' + xy'w'z$

$d = x'y'w'z' + x'y'wz' + x'y'wz + x'yw'z + x'ywz' + xy'w'z' + xy'w'z'$

$e = x'y'w'z' + x'y'wz' + x'ywz' + xy'w'z'$

$f = x'y'w'z' + x'yw'z' + x'yw'z + x'ywz' + x'ywz' + xy'w'z' + xy'w'z$

$g = x'y'wz' + x'y'wz + x'yw'z' + x'yw'z + x'ywz' + xy'w'z' + xy'w'z$

xy \ wz	00	01	11	10
00			1	1
01	1	1		1
11	×	×	×	×
10	1	1	×	×

این جدول تنها برای g رسم شده می‌توان برای بقیه خروجی‌ها نیز رسم کرد.

نکته:

در کد BCD تا عدد 9 را در نظر می‌گیریم پس در جدول کارنو خانه‌های 9 به بعد don't care به حساب می‌آیند.

توضیحات:

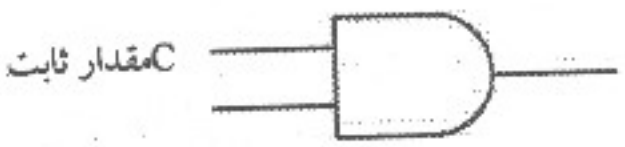
منظور از گیت یک ورودی به این معنی نمی‌باشد که آن گیت دو ورودی ندارد بلکه منظور اینست که یک ورودی آن را مقدار ثابت می‌دهیم و خروجی را در قبال آن می‌سنجیم.

مثلاً اگر یک ورودی گیت AND را ثابت و برابر یک قرار دهیم به آن هر ورودی دیگر بدهیم در خروجی ظاهر خواهد شد.

اطلاعاتی در مورد گیت‌ها:

باید توجه داشت که تمام گیتهای زیر دارای یک ورودی ثابت می‌باشند که با C نمایش داده شده است.

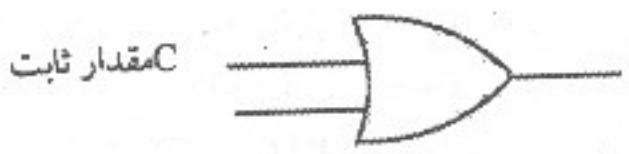
AND یک ورودی:



	x	F	
C = 0	0	0	→ F = 0
	1	0	
C = 1	0	0	→ F = x
	1	1	

نکته: در گیت AND اگر یک ورودی صفر باشد خروجی حتماً صفر است. اما اگر یک ورودی یک باشد خروجی برابر ورودی دیگر است.

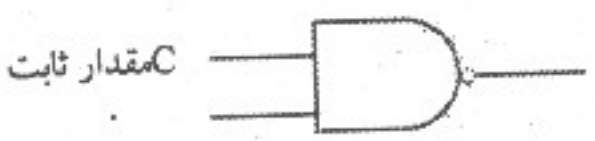
OR یک ورودی:



	x	F	
C = 0	0	0	→ F = x
	1	1	
C = 1	0	1	→ F = 1
	1	1	

* اگر یک ورودی یک باشد خروجی حتماً یک است. اگر یک ورودی گیت OR صفر باشد خروجی برابر ورودی دوم است.

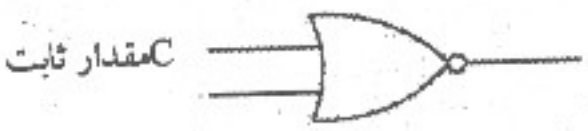
NAND یک ورودی:



	x	F	
C = 0	0	1	→ F = 1
	1	1	
C = 1	0	1	→ F = x′
	1	0	

* در NAND اگر یک ورودی صفر باشد خروجی حتماً یک است و اگر یک ورودی یک باشد خروجی NOT شده ورودی دوم است.

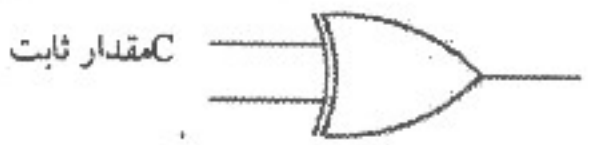
NOR یک ورودی:



	x	F	
C = 0	0	1	→ F = x′
	1	0	
C = 1	0	0	→ F = 0
	1	0	

* اگر یک ورودی یک باشد خروجی حتماً صفر است. اگر یک ورودی صفر باشد خروجی NOT ورودی دوم است.

XOR یک ورودی:



	x	F	
C = 0	0	0	→ F = x
	1	1	
C = 1	0	1	→ F = x′
	1	0	

X-NOR یک ورودی:

	x	F	
C = 0	0	1	→ F = x′
	1	0	
C = 1	0	0	→ F = x
	1	1	

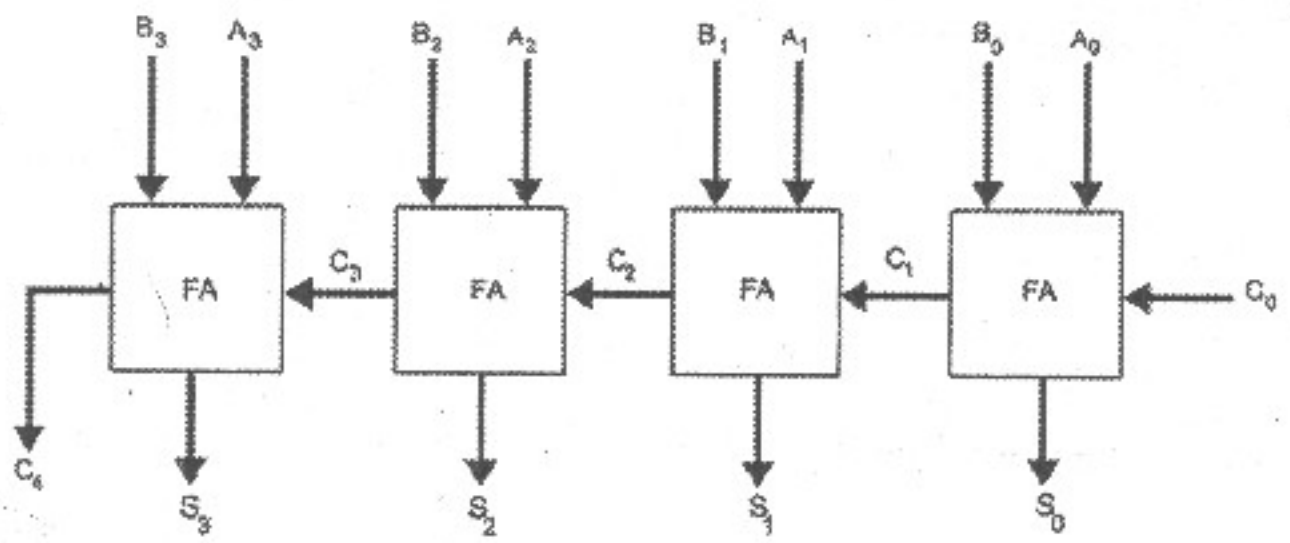
جمع کننده دو به دو موازی

برای جمع دو عدد n بیتی از n عدد، full Adder استفاده می‌شود که هر یک شامل یک C_{in} ، C_{out} و حاصل جمع (S)می باشند. (n بیتی) carry خروجی هر مرحله، C_{in} برای مرحله بعدی است.

$A = (A_{n-1}A_{n-2}...A_2A_1A_0)$ عدد n بیتی اول

$B = (B_{n-1}B_{n-2}...B_2B_1B_0)$ عدد n بیتی دوم

$S = (S_{n-1}S_{n-2}...S_2S_1S_0)$ حاصل جمع A و B



جمع کننده 4 بیت

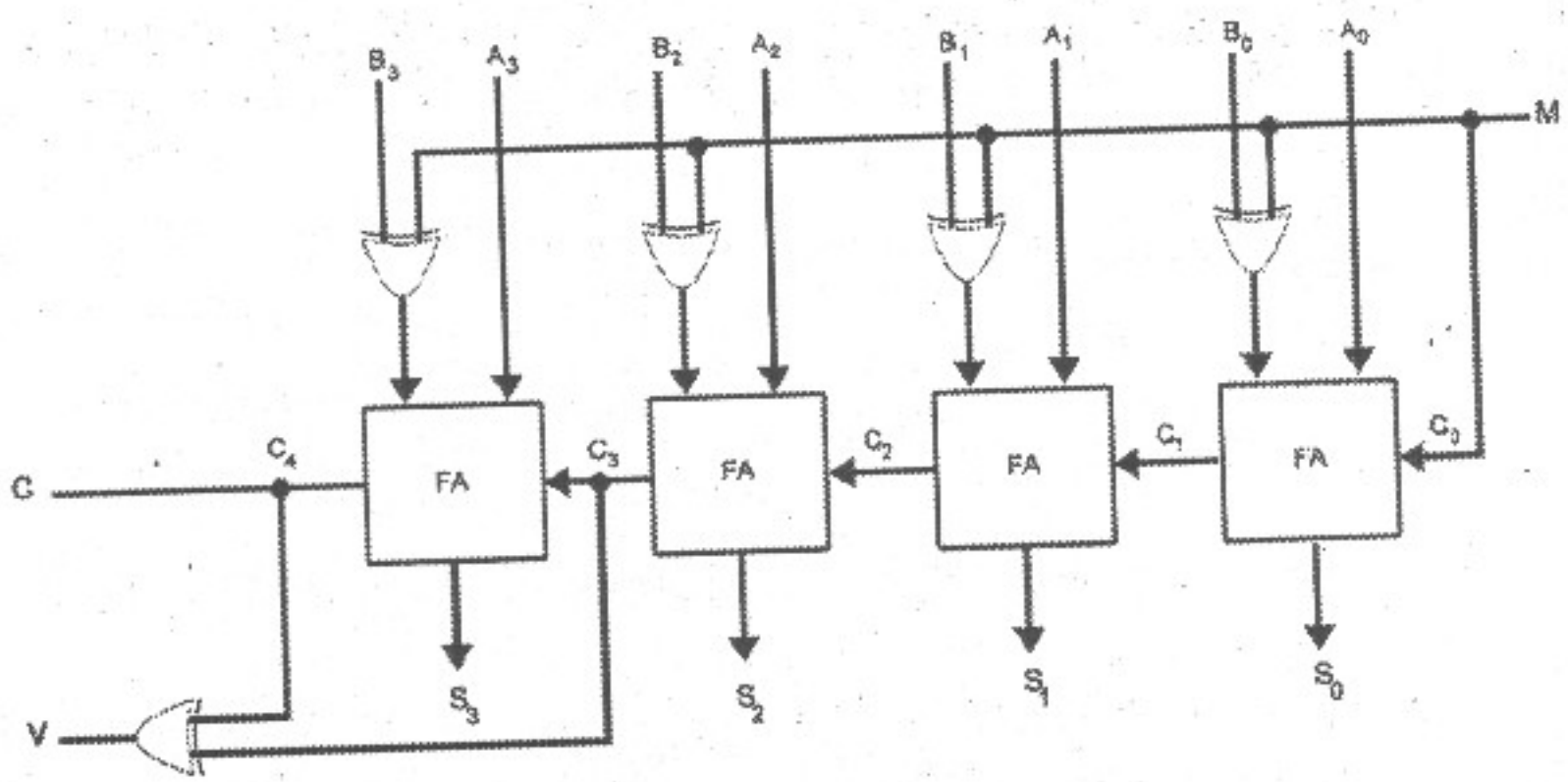
تفریق دودویی

برای تفریق دو عدد A و B (A-B) ابتدا از B مکمل ۲ می‌گیریم سپس A را با مکمل ۲ عدد B جمع می‌کنیم.

اما برای این کار از گیت X-OR استفاده می‌کنیم یعنی یک ورودی X-OR را ثابت وبرابر با مقدار Logic یک قرار می‌دهیم. و ورودی دیگر را همان عدد B می‌گذاریم. با این عمل در واقع مکمل یک عدد B را بدست آورده‌ایم حال باید آن را با یک جمع کنیم تا حاصل همان مکمل 2 شود.

در واقع ورودی هر Full Adder یک بیت از عدد A و یک خروجی X-OR که از بیت متناظر B بدست آمده می‌باشد. باید توجه داشت دیگر ورودی X-OR مورد نظر خط M می‌باشد که هنگامی که مقدارش صفر است عمل جمع B و A را انجام می‌دهد و اگر M مقدارش یک باشد عمل تفریق صورت می‌گیرد.

برای اینکه عمل تفریق به درستی انجام شود و A با مکمل 2 عدد B جمع شود باید ورودی M را به C_{in} وصل کرد تا زمانی که M=1 است و عمل تفریق انجام می‌شود، A با مکمل یک عدد B و عدد یک جمع شود تا حاصل تفریق درست باشد.



جمع - تفریق گر

انتشار رقم نقلی

با توجه به شکل فوق، می‌بینیم که برای انجام عمل جمع، هر مرحله منتظر Carry مرحله قبل می‌ماند تا آن را دریافت کند بعد عمل جمع را انجام دهد. این تاخیر تا مراحل بعدی ادامه دارد به همین دلیل در انتها تاخیر زیادی خواهیم داشت. لذا برای از بین بردن این تاخیر با توجه به شکل F.A به پیش‌بینی رقم نقلی می‌پردازیم.

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i$$

ورودی C_0 =

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

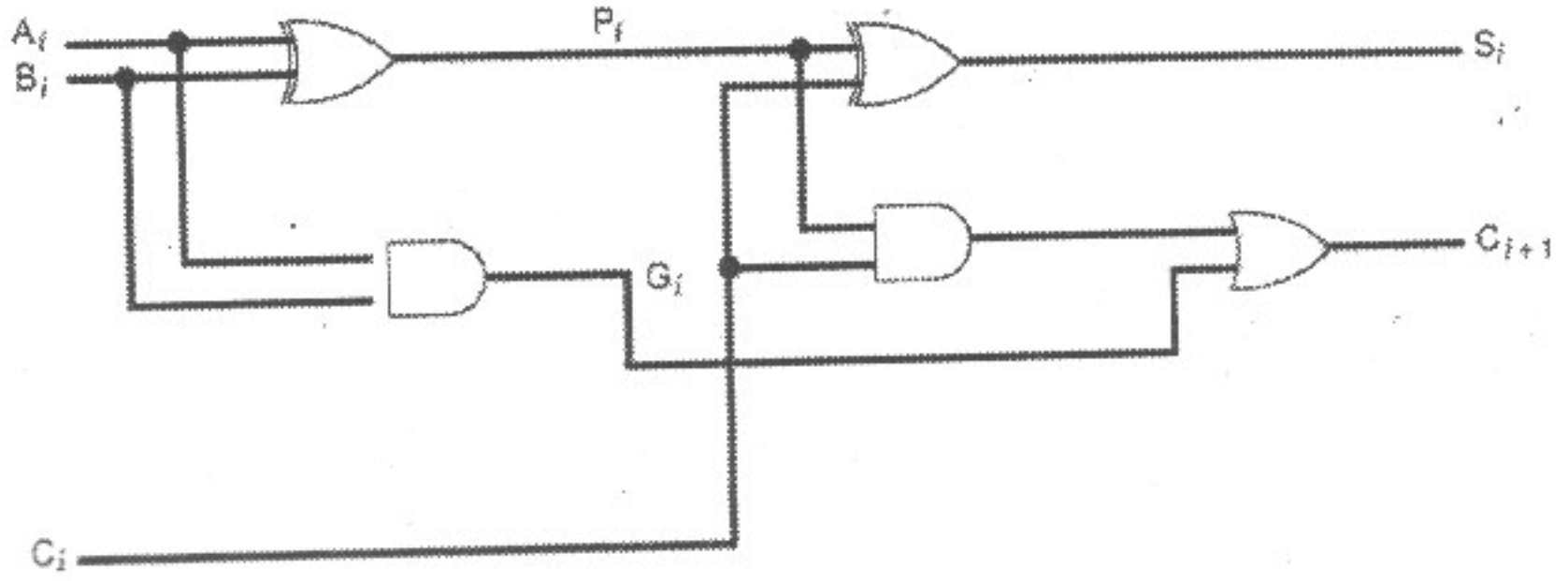
$$C_1 = G_0 + P_0 C_0$$

$$C_2 = G_1 + P_1 C_1$$

$$G_1 + P_1 (G_0 + P_0 C_0)$$

$$C_3 = G_2 + P_2 C_2$$

$$G_2 + P_2 (G_1 + P_1 (G_0 + P_0 C_0))$$



جمع کننده کامل با P و G

حال می‌توانیم C_n را نیز برحسب C_1 یعنی C_{in} بدست آوریم و لازم نیست منتظر محاسبه آن باشیم.

تنها مشکل این روش وجود محاسبات بسیار بالای آن می باشد، اما لازم به ذکر است که اگر دو عددی که با هم جمع می‌شوند تعداد بیتهایشان زیاد باشد این روش خیلی سریعتر از عدم استفاده از آن می باشد.

جمع کننده BCD

همانطور که میدانیم اعداد BCD از صفر شروع وبه 9 ختم می‌شوند و می‌بینیم که:

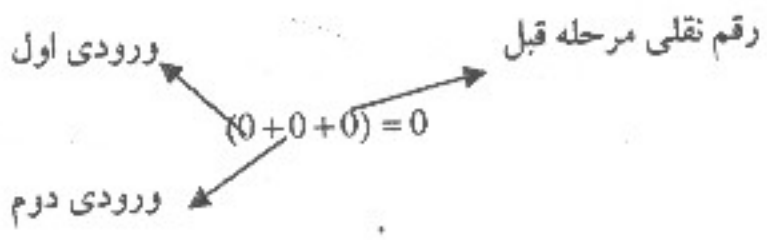
$$13 = \begin{cases} \text{BCD} & = 1001 \\ \text{باینری} & = 1101 \end{cases}$$

بزرگترین حاصل جمع در BCD

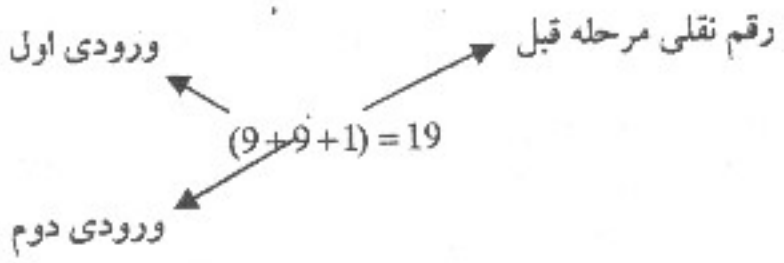
$$9 + 9 + 1 = 19$$

1 1001

بدیهی است که کوچکترین حاصلجمع نیز صفر می باشد یعنی هر دو ورودی صفر باشند و همچنین از مرحله قبل نیز Carry وجود نداشته باشد.



با توجه به توضیحات بالا به این نتیجه می‌رسیم که بزرگترین حاصلجمع مکمل در BCD جمع دو عدد 9 و جمع حاصل با کری مرحله قبل می‌باشد.



مشکل موجود در این نوع جمع استفاده از جمع کننده دو به دو موازی می‌باشد.

که اعداد خروجی را به صورت باینری تولید می‌کند. بنابراین باید اعداد صفر تا 19 را به معادل BCD تبدیل کنیم. برای این کار از جدول زیر استفاده می‌کنیم.

دهدهی	جمع BCD					جمع دودویی				
	S_1	S_2	S_4	S_8	C	Z_1	Z_2	Z_4	Z_8	K
0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	1	0	0	0	0
2	0	1	0	0	0	0	1	0	0	0
3	1	1	0	0	0	1	1	0	0	0
4	0	0	1	0	0	0	0	1	0	0
5	1	0	1	0	0	1	0	1	0	0
6	0	1	1	0	0	0	1	1	0	0
7	1	1	1	0	0	1	1	1	0	0
8	0	0	0	1	0	0	0	0	1	0
9	1	0	0	1	0	1	0	0	1	0
10	0	0	0	0	1	0	1	0	1	0
11	1	0	0	0	1	1	1	0	1	0
12	0	1	0	0	1	0	0	1	1	0
13	1	1	0	0	1	1	0	1	1	0
14	0	0	1	0	1	0	1	1	1	0
15	1	0	1	0	1	1	1	1	1	0
16	0	1	1	0	1	0	0	0	0	1
17	1	1	1	0	1	1	0	0	0	1
18	0	0	0	1	1	0	1	0	0	1
19	1	0	0	1	1	1	1	0	0	1

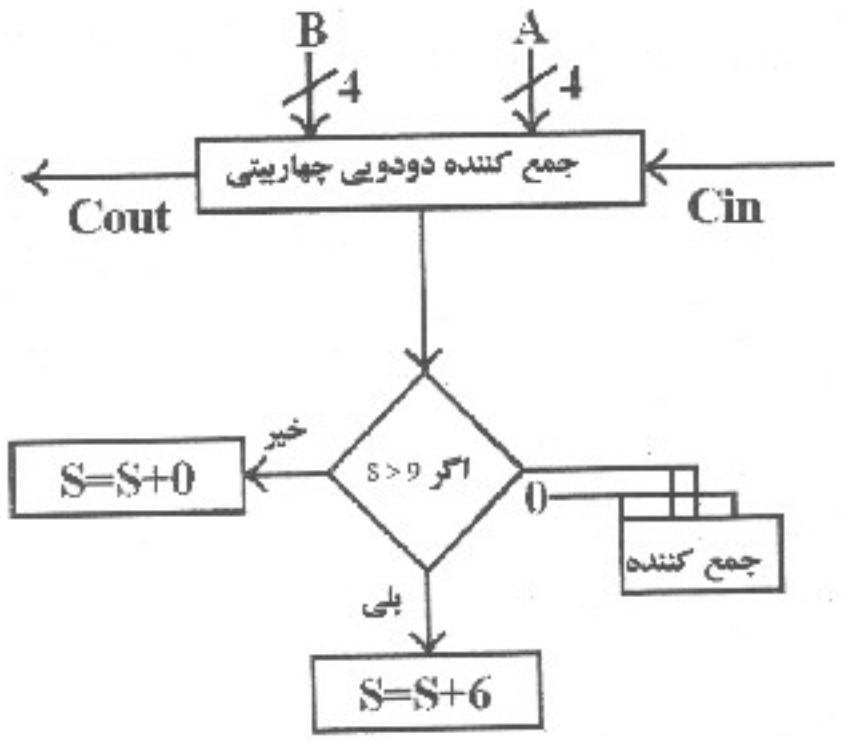
با توجه به جدول فوق مشاهده می‌شود که اعداد 0 تا 9 در باینری معادل BCD هستند اما از اعداد 10 تا 19 باید به معادل BCD تبدیل شوند.

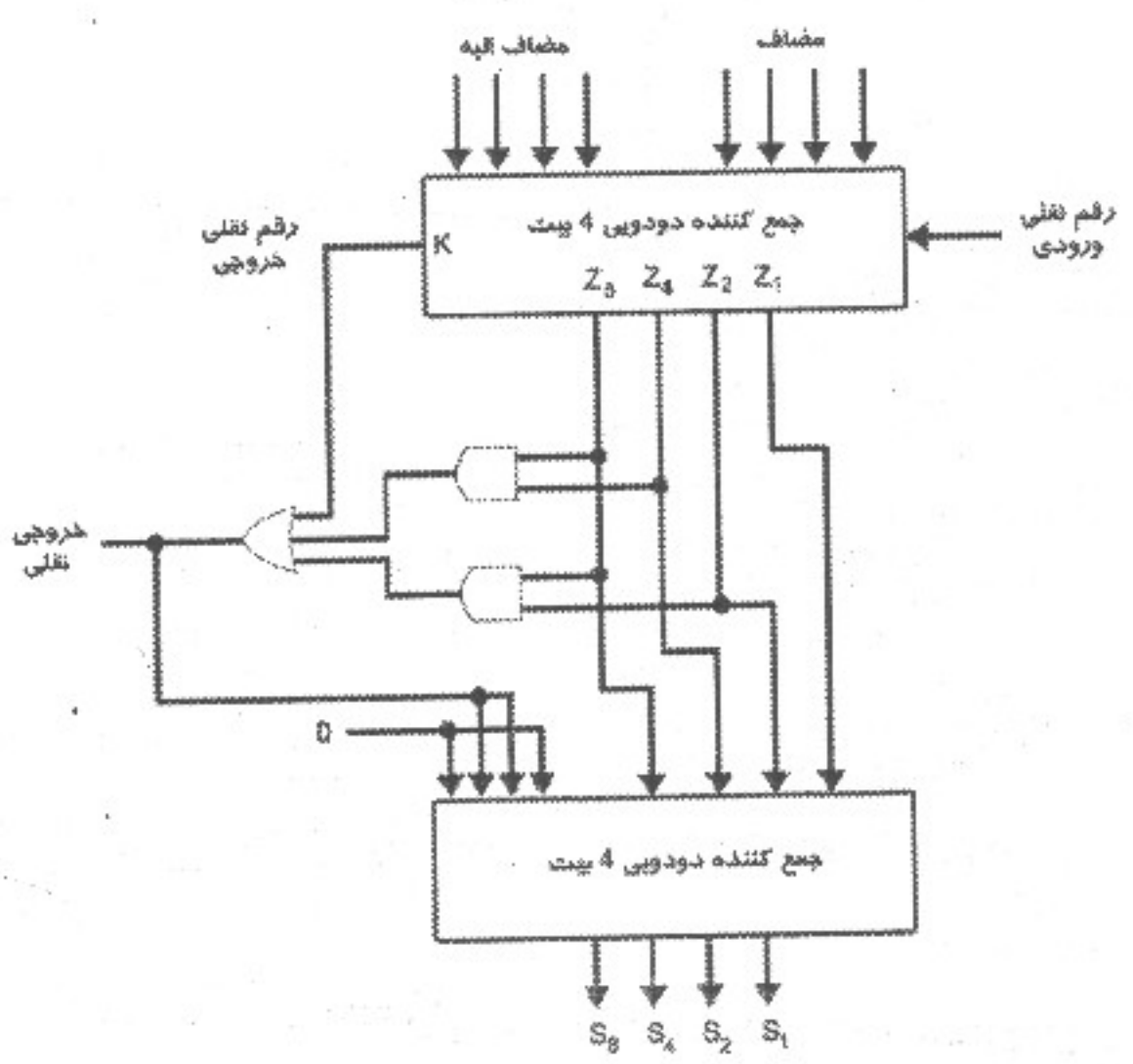
با توجه به جدول (با استفاده از جدول کارنو) می‌توان خروجی‌های ساده شده را بدست آورد. اما با توجه به شکل جدول می‌توان دید که به اعداد باینری مقدار 6 اضافه شده تا حاصل به صورت BCD درآید. به عنوان مثال عدد 16 در باینری معادل 10000 می‌باشد. برای تبدیل آن به 16 در BCD باید به آن (0110) را اضافه کرد یعنی به صورت (10110) که 4 بیت سمت راست معادل 6 و سمت چپ‌ترین بیت معادل 1 است.

برای این کار از دو جمع کننده دو دویی 4 بیتی باید استفاده کرد به این صورت که دو عددی را که می‌خواهیم با هم جمع کنیم را به جمع کننده اول وارد می‌کنیم حاصل را به جمع کننده دوم اعمال می‌کنیم و با توجه به جدول و به دست آوردن معادله خروجی C مربوط به کری خواهیم داشت:

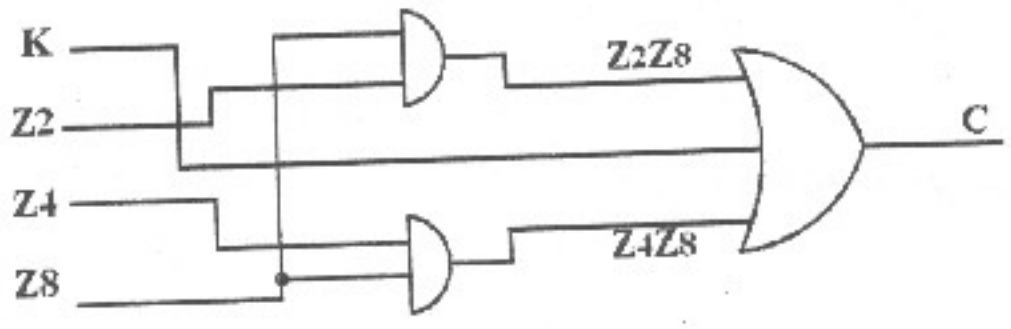
بنابراین با استفاده از خروجی‌های جمع کننده اول خروجی C را تولید می‌کنیم. و باز با توجه به جدول می‌بینیم زمانی باید مقدار عددی 6 را به عدد باینری اضافه کنیم که مقدار C، یک است.

حال باید دیگر ورودی جمع کننده دوم را عدد 6 بگذاریم (0110) می‌بینیم که این مقدار زمانی فعال است که مقدار خروجی C یک باشد.





جمع کننده BCD



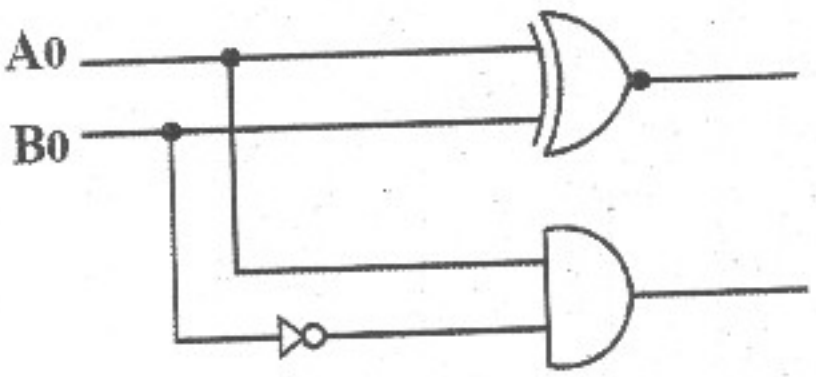
در مقایسه دو عدد باینری باید مقایسه بیت به بیت انجام شود. می دانیم با گیت X-NOR (یا X-OR فرقی نمی کند) می‌توان عمل مقایسه را انجام داد. در واقع تنها می‌توان با یک X-NOR مساوی بودن بیت‌ها را چک کرد و برای یک داده n بیتی می‌توان از n عدد X-NOR استفاده کرد. تا عمل مقایسه انجام شود. به این شکل که اگر تمام خروجی X-NOR ها، یک شود یعنی دو عدد n بیتی با یکدیگر مساوی هستند. اما در غیر اینصورت باید بتوان تشخیص داد کدامیک بزرگتر هستند.

رابطه ۹-۱

$$x_i = A_i \odot B_i \quad i = 0,\ldots,n-1$$

$$\Rightarrow x_1 = A_1 \odot B_1$$
$$x_2 = A_2 \odot B_2$$
$$\vdots$$
$$x_n = A_n \odot B_n$$

A	B	⊙
0	0	1
0	1	0
1	0	0
1	1	1



① اگر $A_0 = B_0 \leftarrow f = 1$
اگر $A_0 \neq B_0 \leftarrow f = 0$

② اگر $A_0 > B_0 \leftarrow f = 1$
اگر $A_0 \not> B_0 \leftarrow f = 0$

② , ① ← اگر با هم اتفاق بیفتد ← $A_0 < B_0$

در صورتی $A \not> B$ است که بیت پر ارزش آن از بیت پر ارزش B بزرگتر باشد یعنی بیت nام، A باید یک و بیت nام، B باید صفر باشد. اما زمانی که بیت های nام مساوی بودند باید به بیت n-1 ام توجه کرد و به همین ترتیب اگر این بیت‌ها هم مساوی بودند باید به بیت‌های کم ارزش‌تر برسیم.

اگر دو عدد چهاربیتی $A = A_3A_2A_1A_0$ و $B = B_3B_2B_1B_0$ را در نظر بگیریم می‌بنیم که اگر دو عدد با هم متساوی باشند حاصل x_i برابر یک خواهد بود $(x_i = A_i \odot B_i)$ و در غیر اینصورت خواهیم داشت:

به معنی یک بودن A و صفر بودن B می‌باشد: AB' و به معنی یک بودن A است :A

به معنی صفر بودن A است: A'

بنابراین ملاحظه می‌شود که:

رابطه (۱۰-۱):

$$f(A>B) \Rightarrow A_3B_3' + x_3.A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0' \rightarrow f\begin{cases}1 \rightarrow A>B\\ 0 \rightarrow A \not> B\end{cases}$$

رابطه (۱۱-۱):

$$f(A<B) \Rightarrow A_3'B_3 + x_3.A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0 \rightarrow f\begin{cases}1 \rightarrow A<B\\ 0 \rightarrow A \not< B\end{cases}$$

مثال:

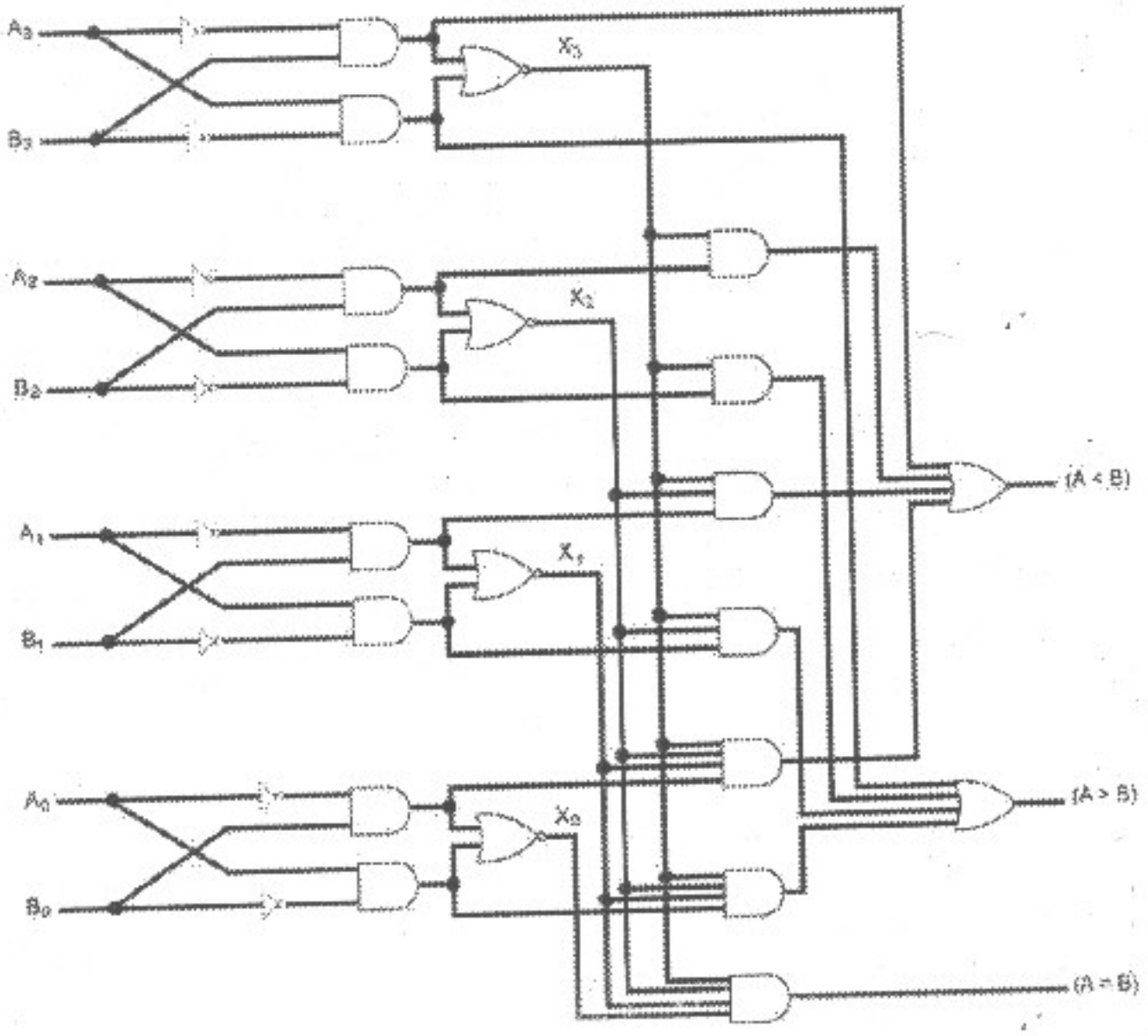
دو عدد $A=1111$ و $B=1110$ را با روش فوق با یکدیگر مقایسه کنید.

با استفاده از رابطه ۱۱-۱ داریم:

$$\begin{matrix}A=1111\\ B=1110\end{matrix} \Rightarrow \begin{cases}A_3=1 & B_3=1\\ A_2=1 & B_2=1\\ A_1=1 & B_1=1\\ A_0=1 & B_0=0\end{cases} \quad \begin{matrix}A_3B_3' + x_3.A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'\\ 0 \quad +(1)(0) \quad +(1)(1)(0) \quad +(1)(1)(1)(1)\end{matrix}$$

در صورتیکه حاصل ۱ باشد $A>B$ اما در غیر اینصورت نمی‌توان گفت که حتماً $A<B$ است زیرا در صورت تساوی A و B نیز حاصل محاسبات فوق صفر خواهد بود و برای اینکه بتوانیم تشخیص دهیم که $A<B$ است باید حتماً از رابطه ۱۱-۱ استفاده کنیم.

$A_3B_3' = 0$
$x_3 = A_3.B_3 = 1$
$A_2B_2' = 0$
$x_2 = A_2.B_2 = 1$
$A_1B' = 0$
$x_1 = 1$
$A_0B_0' = 1$



مقایسه گر مقدار چهار بیتی

دیکدر (Decoder):

n ورودی

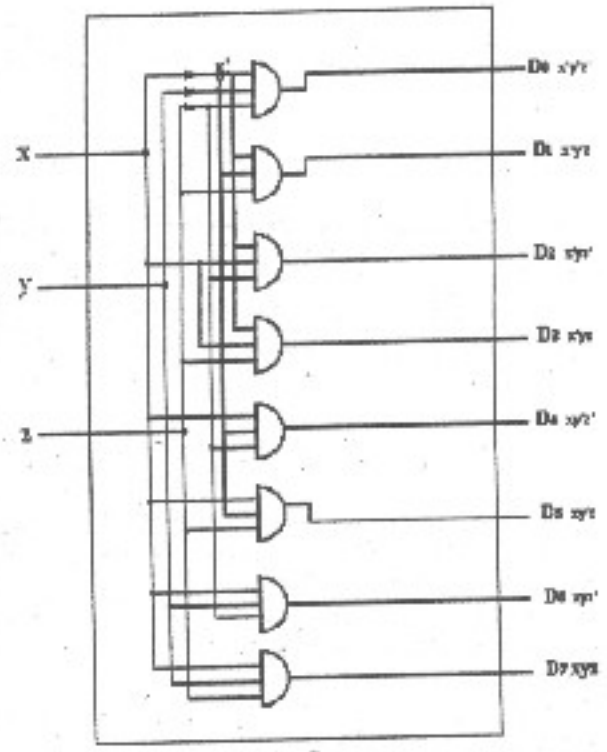
2^n خروجی

یک دیکدر شامل n ورودی و 2^n خروجی می‌باشد. در واقع با اعمال ورودی ها می‌توان تمام 2^n مینترم موجود را پیاده‌سازی کرد. به این شکل که با اعمال ورودی و با توجه به مقدار باینری آن همان شماره در خط خروجی (شماره مینترم) فعال می‌شود. یعنی همان خط مقدارش یک و بقیه صفر می شود بنابراین مشاهده می‌شود با یک دیکدر می توان مسائل ترکیبی مطرح شده در قبل را به راحتی پیاده‌سازی کرد. باید توجه داشت که در یک Decoder در هر لحظه تنها یک خروجی یک می‌باشد.

ورودی‌ها			خروجی‌ها							
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

ساختمان داخلی یک دیکدر

	x	y	z	
0	0	0	0	x′y′z′
1	0	0	1	x′y′z
2	0	1	0	x′yz′
3	0	1	1	x′y z
4	1	0	0	xy′z′
5	1	0	1	xy′z
6	1	1	0	xyz′
7	1	1	1	xyz



3x8

- طراحی به کمک دیکدر

۱ـ یک دیکدر Active high و گیت OR یک‌های خروجی

۲ـ یک دیکدر Active high و گیت NOR صفرهای خروجی

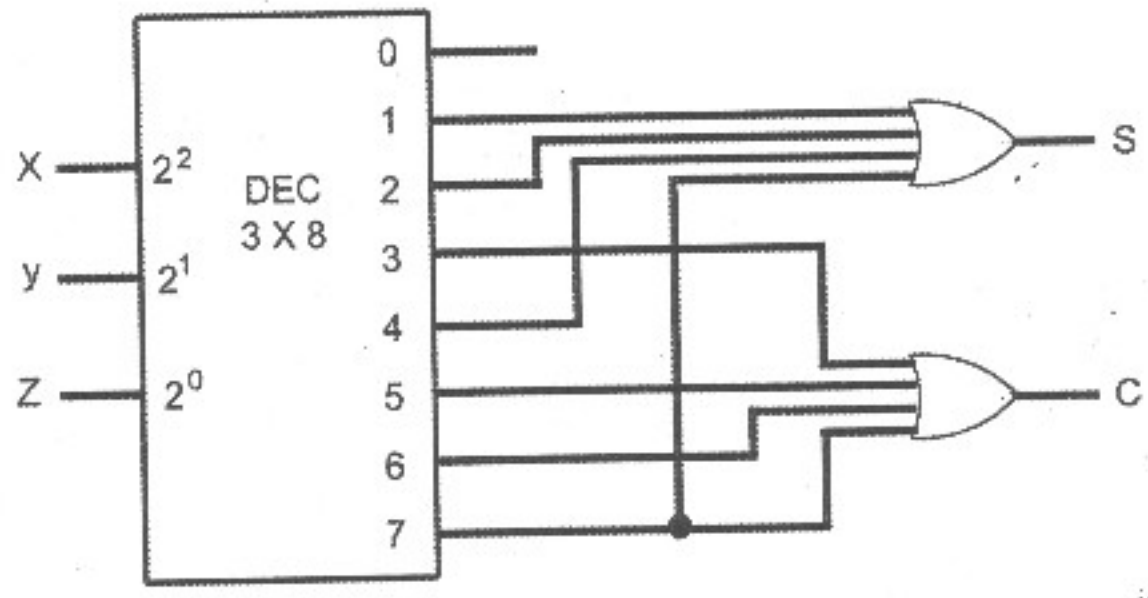
۳ـ یک دیکدر Active low و گیت AND صفرهای خروجی

۴ـ یک دیکدر Active low و گیت NAND یک‌های خروجی

مثال:

یک مدار تمام جمع کننده با استفاده از یک دیکدر و گیت OR طراحی کنید. (به تعداد خروجی گیت OR داریم)

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

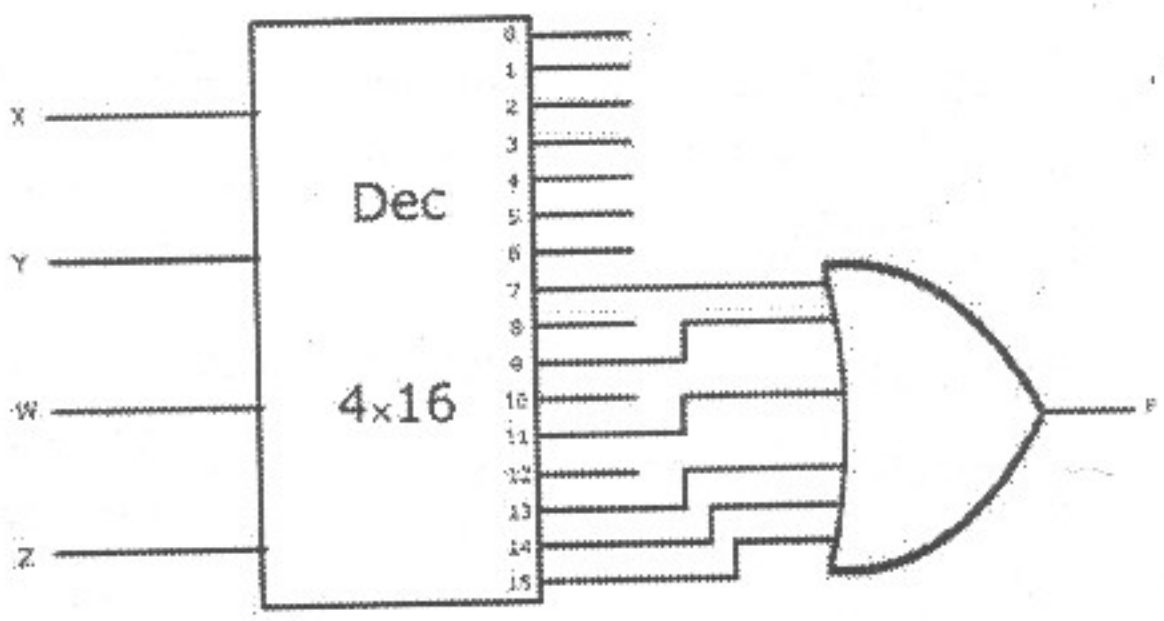


پیاده سازی یک جمع کننده کامل با دیکدر

مثال:

یک میز رای گیری که ۴ رای دهنده وجوددارد (x,y,z,w) زمانی چراغ رای‌گیری روشن می‌شود که یا سه نفر جواب مثبت داده باشند یا اینکه نفر اول وچهارم رای مثبت داده باشند.

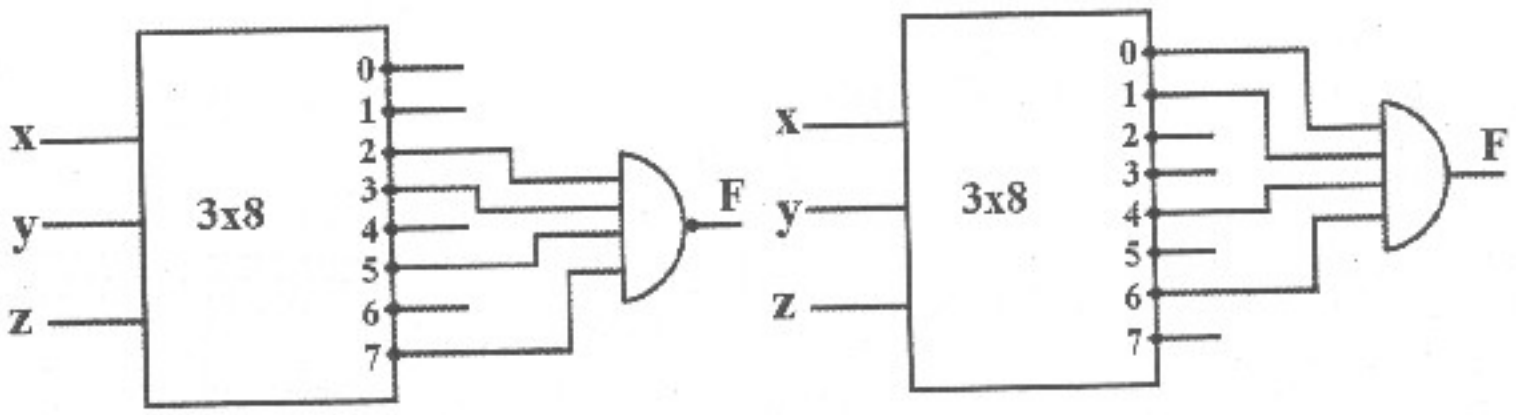
x	y	w	z	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



مثال- فرض کنید تابع f به صورت زیر تعریف شده می‌خواهیم به کمک یک دیکدر Active low و گیت‌های AND و NAND تابع F را پیاده‌سازی کنیم.

صفرهای خروجی → AND → Active low

	x	y	z	f
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	0	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1



دیکدر همراه با خط Enable

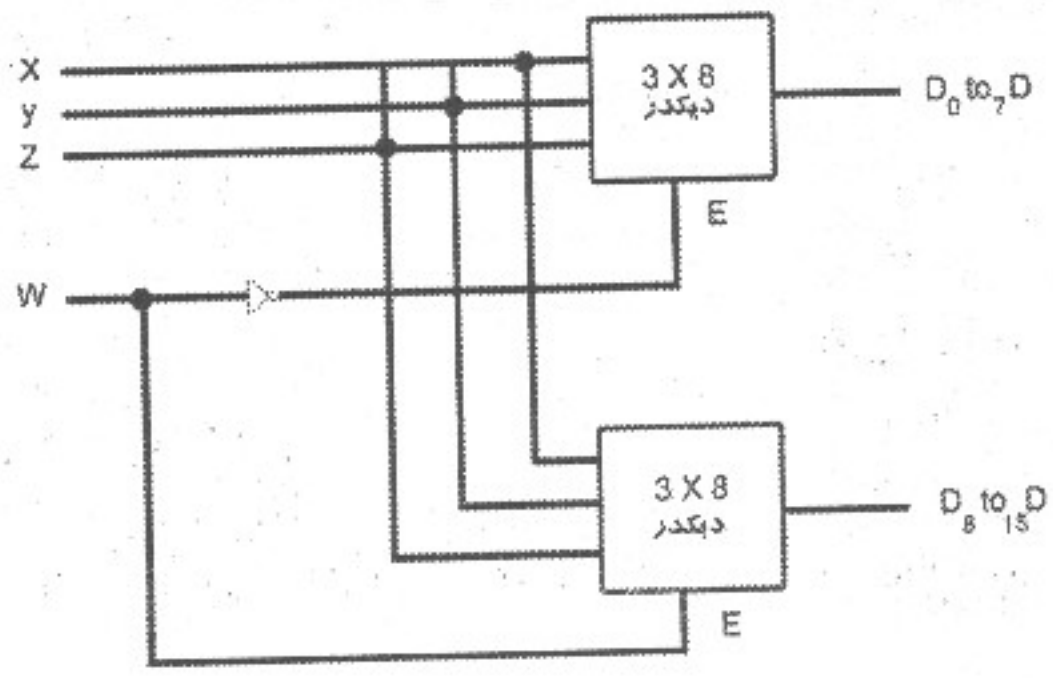
این دیکدر به وسیله یک خط به نام Enable قادر به کار کردن می‌باشد یعنی زمانی که خط $E = 1$ باشد دیکدر فعال و در زمان صفر بودن E دیکدر غیرفعال است. غیرفعال بودن آن به این معنی است که تمام خروجی‌ها مقدارشان صفر است.

ورودی‌ها								خروجی‌ها		
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

ترکیب دیکدرها:

می توان با استفاده از دیکدرها با ورودی کمتر دیکدرهایی با خروجی بیشتر ساخت به عنوان مثال با استفاده از دو دیکدر 3×8 می‌توان یک دیکدر 4×16 ساخت.

با توجه به شکل ملاحظه می‌شود که دیکدر به وجود آمده دارای 4 ورودی و 16 خروجی می‌باشد، نحوه کار ان به این شکل است که در زمانی که E مقدارش یک است دیکدر پایین فعال است و دیکدر بالا غیر فعال بنابراین خروجی‌های دیکدر بالائی همگی صفر هستند و بسته به مقداری که به x و y و z داده می‌شود که خط خروجی در دیکدر پایینی یک می‌شود.بنابراین می‌توان با مقداردهی به x و y و z و E تمامی 16 مینترم موجود را تولید کرد.



دیکدر 16 X 4 ساخته شده با دو دیکدر 3 X 8

خط E در شکل فوق به صورت Active high می‌باشد در این حالت هنگامی که E مقدارش صفر باشد دیکدر به صورت غیرفعال می‌باشد به این معنا که تمام خروجی‌ها غیرفعال است (اگر دیکدر Active high بود خروجی‌ها همه صفر و اگر Active low بود خروجی‌ها همه یک می‌شود).

اما زمانی که E مقدار یک داشته باشد دیکدر فعال می‌شود و با توجه به عدد ورودی یک شمارهٔ خروجی فعال می‌شود..

انکدرها (Encoder)

انکدر برعکس دیکدر می‌باشد یعنی 2^n ورودی و n خروجی است. در هر لحظه تنها یک ورودی فعال است که باعث نمایش یک مقدار باینری در خروجی است.

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

انکدرهای اولویت

در انکدر اولویت به جای فعال بودن تنها یک بیت از ورودی‌ها می تواند چند بیت فعال باشد. اما باید توجه داشت برای نمایش خروجی به باارزش‌ترین بیت‌ها توجه می‌شود یعنی اگر در ورودی دو خط فعال بود آن خطی در نظر گرفته می‌شود که ارزش بیتی آن بیشتر باشد. به این معنی که برای نشان دادن خروجی ابتدا باارزشترین بیت ورودی چک می‌شود و اگر مقدارش یک بوده به دیگر ورودی ها اهمیتی نمی‌دهد و اگر مقدار باارزش‌ترین بیت ورودی صفر بود سراغ بیتهای بعدی می‌رود.

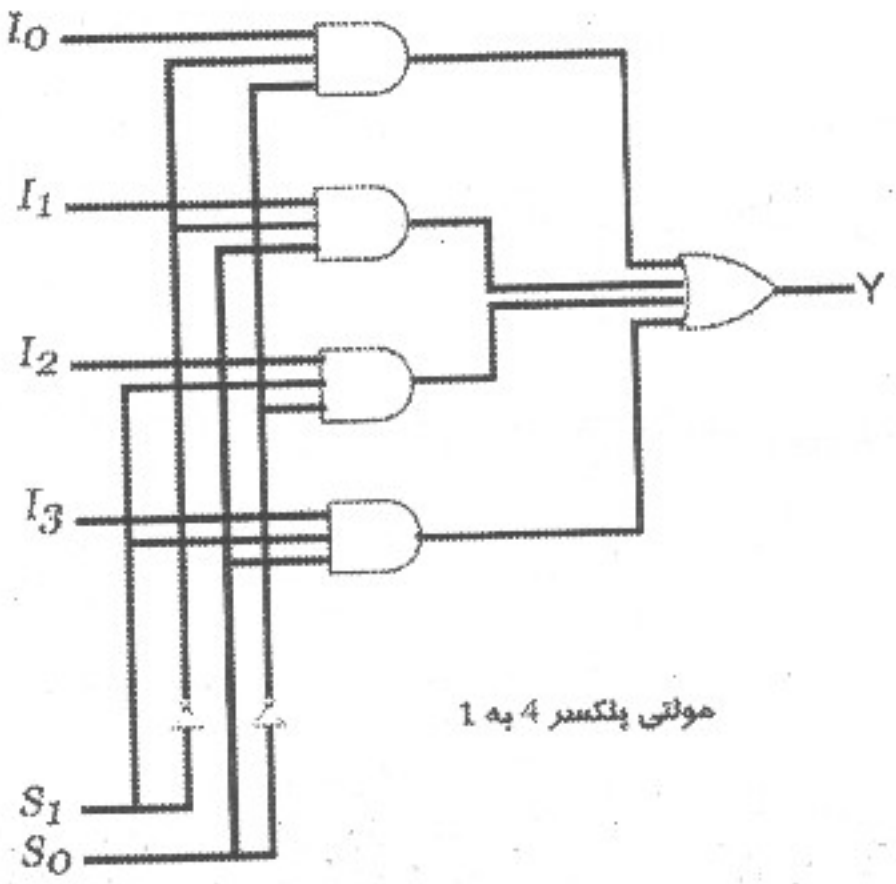
ورودی‌ها				خروجی‌ها		
D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

مولتی پلکسر MUX (یا Multi Plexer)

در مولتی پلکسر 2^n خط ورودی داریم و n خط select و یک خروجی. در اصل مقدار وارد شده به خطوط select باعث فعال شدن یکی از خطوط ورودی و انتقال آن به خروجی می‌شود. برای پیاده‌سازی توابع ترکیبی می‌توان از مولتی پلکسر استفاده کرد.

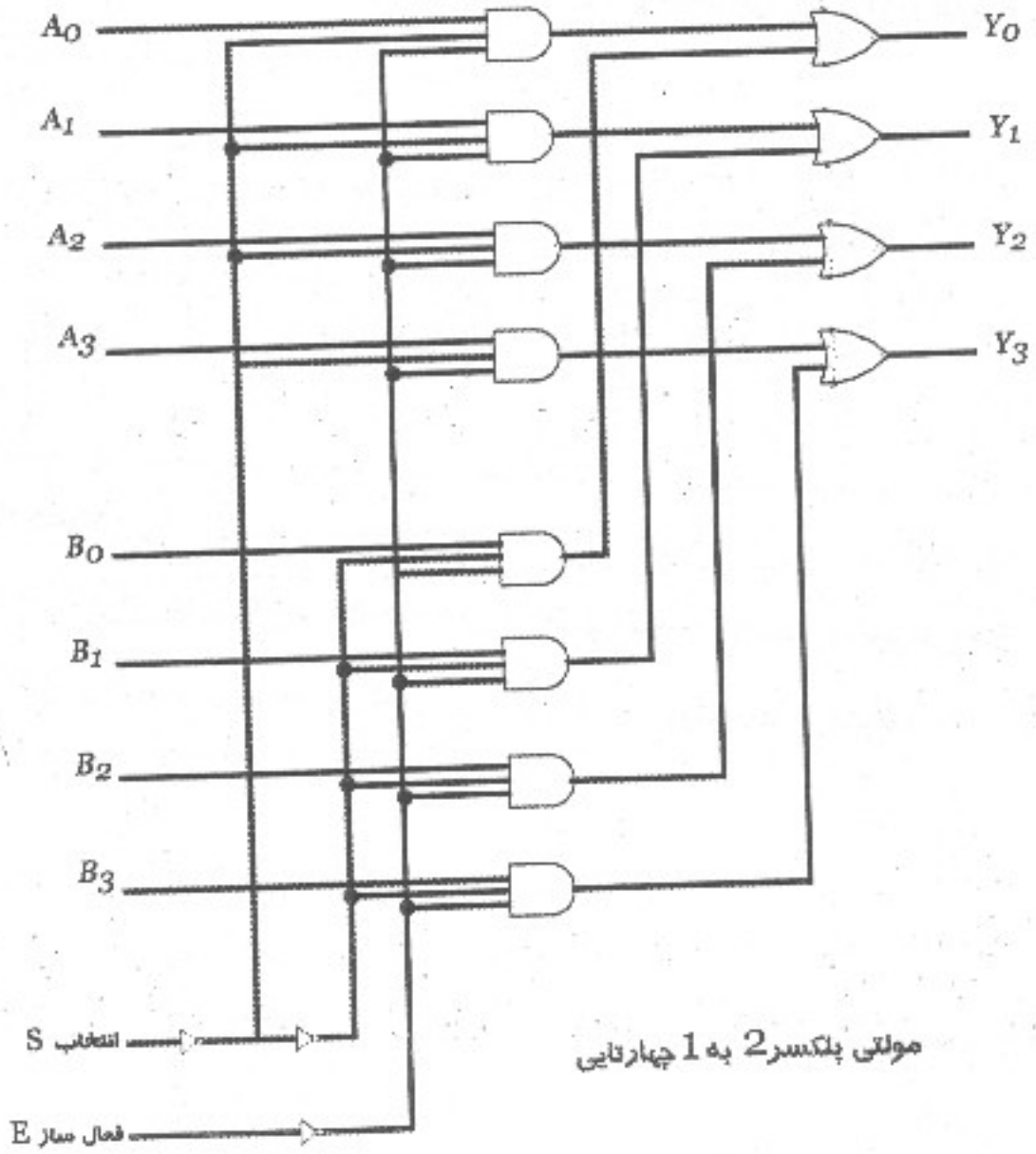
S_1	S_0	y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

2^n ورودی
1 خروجی

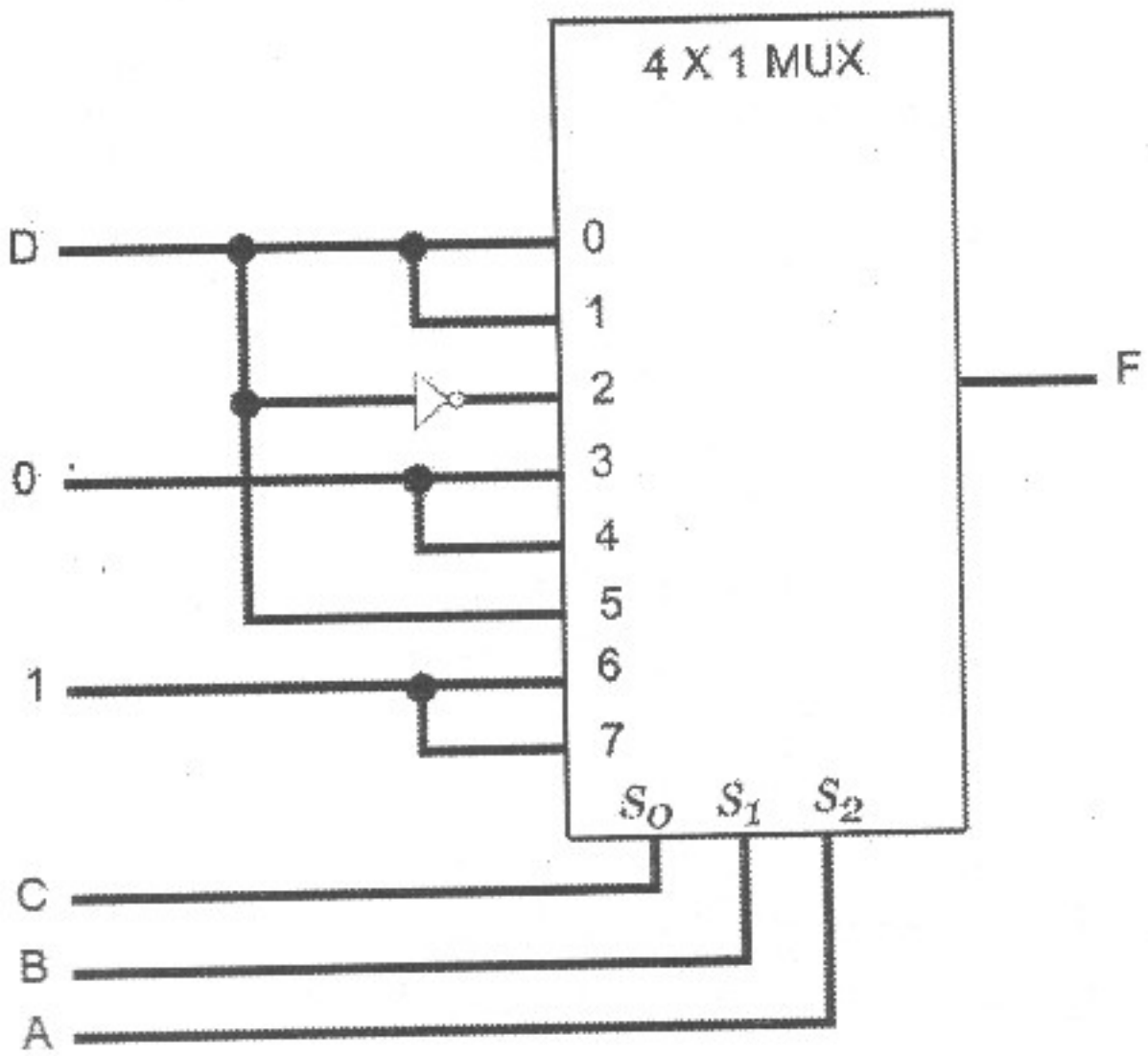


مولتی پلکسر 4 به 1

* AND شمارهٔ 1 زمانی خروجی‌اش برابر I_0 می‌شود که هر دو S_0 و S_1 برابر صفر باشد.
* AND شمارهٔ 3 زمانی مقدار خروجی‌اش برابر I_2 می‌شود که $S_0 = 0$ و $S_1 = 1$ باشد.



مولتی پلکسر2 به 1 چهارتایی



پیاده‌سازی یک تابع۴ ورودی با یک مولتی پلکسر

طراحی به کمک multiplexer

چهار را ه متفاوت برای طراحی به کمک mux وجود دارد که عبارتند از:

الف) طراحی به کمک بیت با ارزش (روش n-1)

$$(A, B, C, D, \ldots)_{n+1} \quad \left.\begin{matrix} A \\ A' \\ 0 \\ 1 \end{matrix}\right\} \rightarrow \text{ورودی‌های MUX}$$

اگر بخواهیم با مولتی پلکسر یک تابع n متغیره را تولید کنیم (طراحی کنیم) باید n-1 متغیر به خطوط select اختصاص دهیم و ورودی‌های مولتی پلکسر را به صورت A و 'A و 0 و 1 در آوریم بنابراین برای پیاده سازی یک تابع n متغیره باید از یک مولتی پلکسر $2^{n-1} \times 1$ استفاده کنیم. قبل از هر چیز باید توجه داشت که تابع باید به صورت مینترم ها در آورده شود.

باید توجه داشت که هیچ لزومی ندارد که با ارزش‌ترین بیت را به ورودی بدهیم و n-1 متغیر باقی مانده را به خطوط select بلکه منظور از توضیحات فوق این است که باید یکی از متغیرها را به ورودی بدهیم و مابقی را به خطوط select

نکته: برای طراحی مدار به کمک مالتی پلکسر، به تعداد خروجی‌ها احتیاج به مولتیپلکسر داریم.

مثال:

یک میز رای‌گیری داریم که سه نفر رای دهنده دارد (A,B,C) زمانی چراغ رای‌گیری روشن می‌شود که حداقل دو نفر جواب مثبت بدهند. به کمک یک MUX آن را طراحی کنید.

$n = 3 \rightarrow n - 1 = 2$ تعداد متغیرها

$2^{n-1} = 2^2 = 4$ تعداد خطوط select

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

	I_0	I_1	I_2	I_3
A'	0	0	0	1
A	0	1	1	1
	0	A	A	1

اگر مقادیر ستون I_n برای A و A' هر دو یک بود مقدار واقعی $I_A = 1$ است.

اگر مقادیر ستون I_n برای A و A' هر دو صفر بود مقدار واقعی $I_n = 0$ است.

ب) طراحی به کمک بیت کم ارزش (روش n-1)

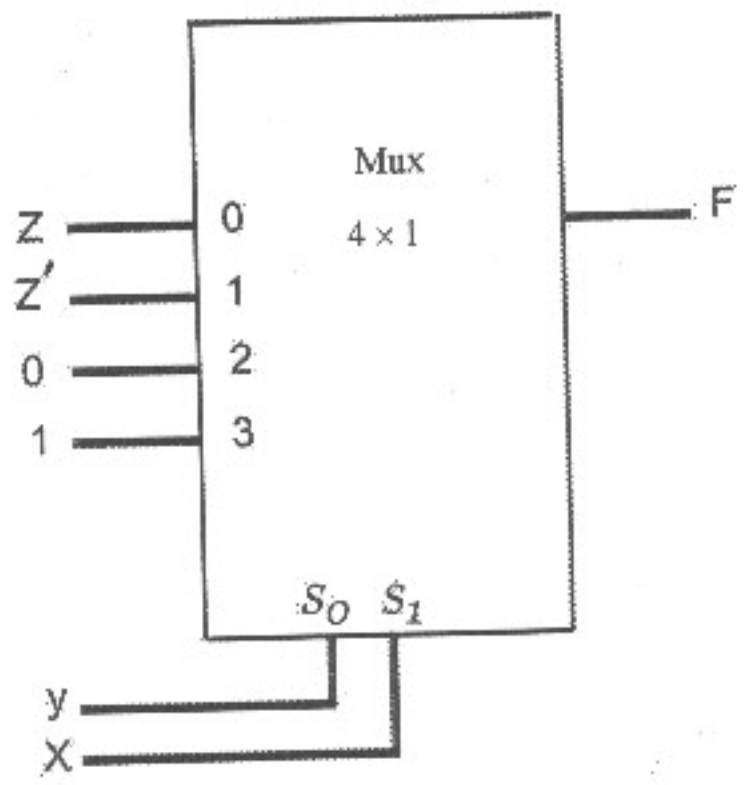
تذکر: در این روش‌ها تنها مشکل شماره‌گذاری جدول تغییر می‌کند.

مثال:

تابع $f(x,y,z) = \Sigma(1,2,4,5)$ را به کمک Mux پیاده‌سازی کنید. می‌خواهیم این بار به جای این‌که x را به ورودی‌ها بدهیم (با ارزش‌ترین بیت) به جایش z (کم ارزش‌ترین بیت) را به ورودی‌ها اختصاص دهیم.

بنابراین در تشکیل جدول باید در شماره‌گذاری تغییراتی بدهیم.

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



ج) طراحی با بیت انتخابی: (روش n – 1)

در این روش ابتدا یک بیت را برای اختصاص به خطوط ورودی انتخاب می‌کنیم. حال باید حالت‌ها مختلف n بیت باقی مانده را از صفر تا $2^n - 1$ در نظر بگیریم. و برای هر حالت ایجاد شده در جدول درستی دو ردیف وجود دارد که باید در این دو ردیف مقدار بیت انتخاب شده به عنوان ورودی را با خروجی مقایسه کنیم در این مقایسه چهار حالت ممکن است پیش بیاید که مقادیر مربوط به خطوط ورودی را ایجاد می‌کنند (در واقع همان I_0 تا I_{2^n-1}) این حالات عبارتند از:

۱ـ اگر بیت انتخابی و خروجی با هم برابر باشند باید به خط ورودی mux بیت انتخابی اختصاص داده شود.

۲ـ اگر بیت انتخابی و خروجی و مخالف هم باشند به خط ورودی mux باید not شده بیت انتخابی را به خط ورودی اختصاص دهیم.

۳ـ اگر در مقایسه بیت خروجی و بیت انتخابی مشاهده شد که در هر دو ردیف خروجی یک شده باید به خط ورودی mux مقدار منطقی یک اختصاص داده شود.

۴ـ اگر در مقایسه بیت خروجی و بیت انتخابی مشاهده شود که در هر دو ردیف مقدار خروجی صفر شده باید به خط ورودی mux (I_0 تا I_{2^n-1}) مقدار منطقی صفر اختصاصی داده شود.

مثال: به کمک mux تابع $f(a, b, c) = \sum(1, 5, 6, 7)$ را طراحی کنید.

حل:

در این مثال متغیر b را برای اختصاص به خطوط ورودی mux در نظر می‌گیریم. بنابراین باید برای دو متغیر باقی ماندهٔ حالت‌های مختلف را در نظر بگیریم. و با توجه به این حالات عملیات مقایسه را انجام می‌دهیم.

ابتدا حالت صفر به صفر را برای ac در نظر بگیرید: (تعیین ورودی I_0) برای این حالت دو ردیف صفر و دو را می‌توان در نظر گرفت. اگر b که به عنوان بیت انتخابی محسوب می‌شود را با خروجی در آن دو ردیف مقایسه کنیم می‌بینیم که در هر دو ردیف بدون توجه به مقدار b خروجی برابر صفر است بنابراین باید به خط ورودی I_0 مقدار صفر اختصاص داده شود.

حال حالت صفر یک را در نظر می گیریم یعنی در اصل می‌خواهیم مقدار ورودی I_1 را بدست آوریم:

همان‌طور که مشاهده می‌شود باید ردیف‌های یک و سه با هم مقایسه شوند در این مقایسه می‌بینیم که در ردیف یک مقدار b با خروجی f به عکس می‌باشد و در ردیف سه نیز همین حالت برقرار است بنابراین باید به ورودی I_1 مقدار کار را اختصاص دهیم.

با همین منوال مشخص می‌شود که مقدار I_2 برابر b و I_3 برابر یک می‌باشد. بنابراین:

	a	b	c	f
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

د: در این روش تمامی متغیرها به خطوط selsct اختصاص داده می‌شود. و با توجه به جدول درستی ورودی‌ها مقادیری را به خود اختصاص می‌دهند به این صورت که مقدار خروجی ردیف صفر به ورودی I_0 اختصاص داده می‌شود و مقدار خروجی $2^n - 1$ نیز به I_{2^n-1} اختصاص داده می‌شود.

مثال: تابع $f(a, b, c, d) = \pi(0, 3, 4, 7, 10; 11, 15)$ را به کمک mux پیاده‌سازی کنید.

	a	b	c	d	f
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

به کمک مولتی پلکسر و چهار روش آن مداری طراحی کنید که زوج و فرد بودن یک عدد سه بیتی را مشخص کند.

روش الف: (روش n-1) (به کمک بیت با ارزش)

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

روش ب: (روش n -1) (به کمک بیت کم ارزش)

	I_0	I_1	I_2	I_3
C′	⓪	②	④	⑥
C	1	3	5	7
	C′	C′	C′	C′

روش ج: (روش n – 1) (به کمک بیت انتخابی)

در این روش B را به عنوان بیت انتخابی در نظر گرفتیم حال باید حالات مختلف A و C را تحلیل کنیم.

$I_0 = 1 \leftarrow AC = 00$
$I_1 = 0 \leftarrow AC = 01$
$I_2 = 1 \leftarrow AC = 10$
$I_3 = 0 \leftarrow AC = 11$

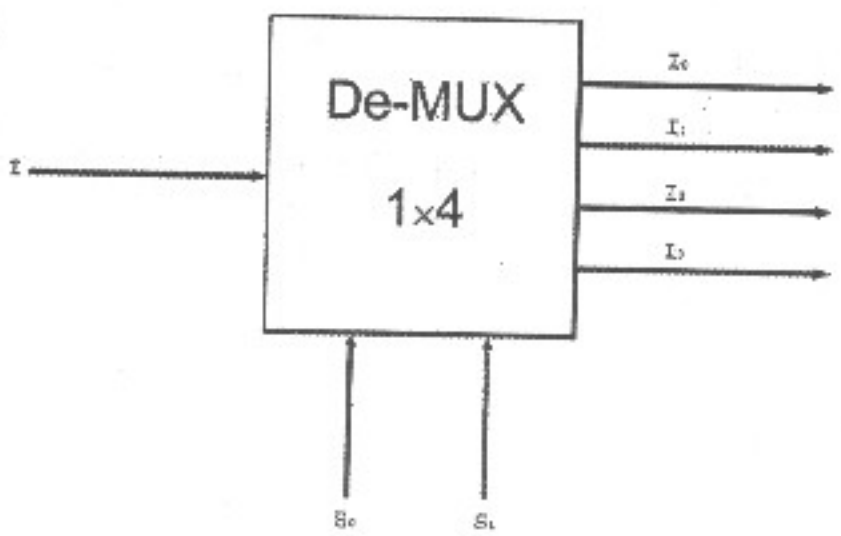
	A	B	C	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

روش د:

	A	B	C	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

دی مالتی پلکسر DMUX یا De Multi Plexer :

عکس MUX می‌باشد به این معنی که تنها یک خط ورودی دارد و 2^n خط خروجی. در یک دی مالتی پلکسر 1×2^n از n خط select استفاده می‌شود. تا یک ورودی را به خروجی منتقل کند. یعنی با ورودی که به خطوط select می‌دهیم مشخص می‌کنیم که مقدار ورودی باید در کدامیک از خروجی‌ها نمایش داده شود.



حافظه و منطق برنامه‌ریز

ROM

ROM از عناصر مدارهای ترکیبی است. مداری است شامل n ورودی و m خروجی مثلاً (8 . 32) ROM دارای 5 وردی $\left(32\ 2^5\right)^n$ و 8 خروجی است. به ازای هر n ورودی، 2^n ترکیب متناوب وجود دار که هر کدام آنها یک آدرس گویند.

هر ترکیبی از خطوط خروجی یک نامیده می‌شود. یعنی با توجه به آدرسی که در ورودی به آن می‌دهیم در خروجی تعداد صفر و یک وارد می‌شود.

PLA

هنگامی که تعداد حالات بی‌اهمیت در یک مدار با تعداد خروجی زیاد بالا باشد به جای استفاده از ROM می‌توان از PLAاستفاده کرد. در ROM تمام ترکیباتُ ورودی (مینترم‌ها) به صورت کامل موجود بود اما در PLA تنها جملات حاصل ضرب وجود دارد یعنی ها به صورت کامل تولید نمی‌شود. منظور از جملات حاصلضرب SOP ها هستند. به جای دیکدر در PLA از گیت‌های AND استفاده می‌وشد و از هر کدام از این گیت‌ها می‌توان برای تولید جملات حاصلضرب متغیرهای ورودی استفاده کرد. گیت‌های AND و OR موجود در PLA از ابتدا همراه با اتصالات ماینشان ساخته می‌شوند تا قطع اتصالات مناسب توابع موردنیاز به فرم حاصلضرب پیاده‌سازی می‌شوند یک PLA شامل n خط ورودی و m خط خروجی و k جمله حاصلضرب و m جمله حاصلجمع می‌باشد تعداد اتصالات در یک PLA از رابطه زیر بدست می‌آید.

$$(2n\times k)+(m\times k)+m$$

برمبنای اینکه از یک ROM(8×6) استفاده کنیم می‌توانیم از یک ROM(8×4) استفاده کنیم.

انواع ROM

سه نوع ROM داریم : EEPROM , EPROM , PROM

PROM : نقشه موردنظر را به شرکت سازنده داده می‌شود. شرکت اتصالات را می‌سوزاند و مدار مورنظر را می‌سازد. در PROM نمی‌شود برای بار دوم برنامه‌ریزی کرد.

EPROM : PROM ی که قابلیت یک شدن دارد یعنی می‌توان یکبار اتصال برقرار کرد بار بعدی حذف کرد.

EEPROM : در EEPROM دیگر اتصالات را با اشعه ماوراءبنفش قطع می‌کنیم تا که با جریان الکتریکی این کار را انجام می‌دهیم.

a	b	c	f_1	f_2	f_3	f_4	f_5	f_6
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

پیاده‌سازی منطق ترکیبی با استفاده از منطق ROM

برای پیاده‌سازی یک مدار ترکیبی با n ورودی و n خروجی بر یک ROM از نوع $2^n \times n$ نیاز داریم.

برنامه‌ریزی ROM

منظور قطع کردن اتصالات در ROM می‌باشد برای این کار باید جدول برنامه ROM را تهیه کرد. منظور از تهیه جدول برنامهROM این است که باید مشخص کنیم کدامیک از خطوط خروجی دیکدر باید وارد OR شوند. در یک ROM در ابتدا تمام خطوط خروجی وارد تمام گیت‌های OR می‌شوند به این منظور که برای هر یک از m گیت OR ، 2^n ورودی خواهیم داشت. اما پس از بدست آوردن جدول برنامه ROM خطوطی را که اتصالشان را لازم نداریم قطع می‌کنیم (این کار را برای تمام گیت‌های OR انجام می‌دهیم). اما در صورتی که بخواهیم پیاده‌سازی را به کمک IC ، ROM انجام دهیم تنها کافی است جدول برنامه بدست آمده را به شرکت سازنده ROM ارائه دهیم تا مدار ترکیبی موردنظر ما را به وجود آورد.

مثال: به کمک یک ROM مداری طراحی کنید که عدد ورودی 3 بیتی را دریافت کند و در خروجی مربع آن را نمایش دهد.

قوه تکمیل جدول به این صورت است که جمله های ضرب SOP ها را به کمک ورودی‌ها می‌سازیم یعنی در شکل 100 و یا حالت بی‌اهمیت. به این معنا که اگر جمله ضرب متغیر ورودی را در خود داشت به جای آن 1 و اگر مکملش را داشت به جایش صفر و درغیر این صورت (-) حالت بی‌اهمیت را می‌گذاریم. در ستون خروجی‌ها اگر تابع f_1 از SOP های k , L , ... تشکیل شده بود $(f_1 = SOP_k + SOP_L + ...)$ در آن ردیف‌ها (k,L,...) می‌گذاریم و در ردیف‌های باقی مانده (-) می‌گذاریم. اگر تابع f_i خودش را استفاده کردیم در ردیف انتها مقدار T را می‌نویسیم و اگر مکملش را استفاده کردیم در آن ردیف c می‌نویسیم تنها باید توجه داشت که اگر در ستونی c نوشته بود باید در انتهای پیاده‌سازی، آن خروجی را متمم کنیم حال برای قسمت نهایی پیاده‌سازی باید تمام ورودی را هم خودشان و هم مکملشان را ایجاد کنیم.

سپس به تعداد جملات حاصلضرب، گیت AND و به تعداد خروجی، گیت OR قرار دهیم. در انتها خروجی OR ها همان توابع موردنیاز ما می‌باشند.

PAL

قطعه منطقی برنامه‌پذیری است با ارائه OR بدون تغییر و ارائه AND برنامه‌پذیر مشاهده می‌شود.

تفاوت PLA , ROM

در این است که در ROM مینترم ها تولید می‌شود اما در PLA ، SOPها تولید می‌شود. در طراحی با ROM خود تابع را می‌خواهیم اما در PLA تابع را به صورت ساده شده خواهیم داشت ROM و PLA در این است که هر دو توابع پیچیده با تعداد خروجی بالا را پیاده‌سازی می‌کنند.

طراحی به کمک PLA :

چون در طراحی به کمک PLA از SOP ها استفاده می‌کنیم بنابراین برای طراحی باید ابتدا تابع موردنظر را ساده کنیم. باید توجه داشت که در این نوع طراحی هم خود تابع و هم مکملش باید پیاده شود (سازده‌سازی توسط کارنوو ...) سپس فرضی را انتخاب می‌کنیم که کمترین تعداد SOP را در خود داشته باشد. سپس جملات مجزا را در تمام توابعی که می‌خواهیم به کمک PLA پیاده‌سازی کنیم، انتخاب می‌کنیم و جدولی به شکل زیر طراحی می‌کنیم.

اگر c بود به این معنی که تابعی که پیاده‌سازی کردیم مکملش است و اگر T بود به این معنی است که خود تابع را پیاده‌سازی می‌کنیم.

	متغیرهای ورودی	خروجی
SOP	A B C ...	F_1 F_2 F_3 ...
SOP_1		
SOP_2		
		مقدار آن یا C است یا T

پیاده‌سازی به کمک PAL با مشخصات فوق بسیار آسان‌تر از پیاده‌سازی به کمک PLA می‌باشد. اما انعطاف‌پذیری PAL بسیار کمتر از PLA می‌باشد.

مشخصات PAL

۱ـ کلیه ورودی‌ها و مکمل آنها موجود است.

۲ـ دارای بردار AND قابل برنامه‌ریزی می‌باشد (تعدادشان کمتر از تعداد ؟ ها است)

۳ـ دارای بردار OR با ورودی‌های ثابت می‌باشد. باید توجه داشت در طراحی به کمک PAL برخلاف PLA نمی‌توان یک جمله ضرب را بین دو یا چند OR به اشتراک گذاشت.

بنابراین هر تابع بدون توجه به توابع دیگر ساده می‌شود.

مثال:

توابع f_1 , f_2 را به کمک PLA پیاده‌سازی کنید.

SOP	A	B	C	f_1	f_2
A B	1	1	-	1	1
A C	1	-	1	1	1
B C	-	1	1	1	-
A′B′C′	0	0	0	-	1
				C	T

ساختمان داخلی ROM

یک ROM از یک دیکدر و تعدادی گیت OR تشکیل شده است. دیکدر هم به وسیله تعدادی AND پیاده‌سازی شده می‌خواهیم یک ROM(16×4) طراحی کنیم این ROM ، 4 ورودی و 4 خروجی دارد. باید از یک دیکدر 16×4 استفاده کنیم هر کدام از انتقالات یک بوز سر راه خود دارند.

ر کدام از مینترم‌ها که لازم باشد ارتباط برقرار است.

مجموعه مسائل فصل سوم:

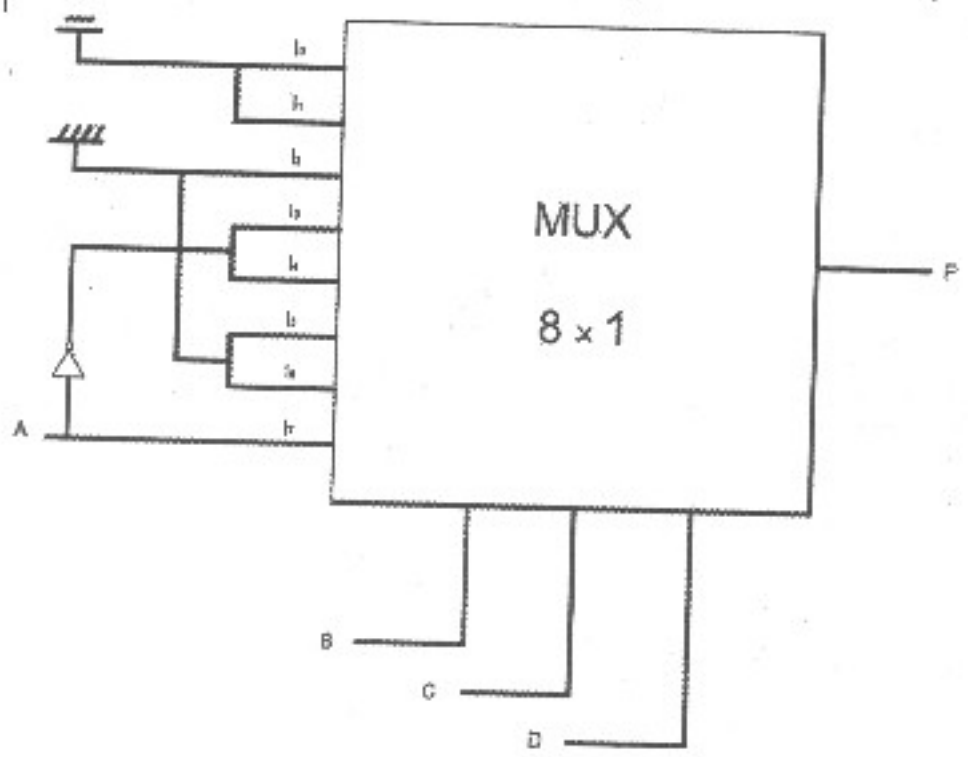
۱ـ تابع را به کمک یک MUX پیاده سازی کنید.

$$F(A,B,C,D) = \Sigma(0,1,3,4,8,9,15)$$

$n+1=4 \Rightarrow n=3$

$2^3 = 8$ تعداد ورودی

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	1	1	0	1	1	0	0	0
A	1	1	0	0	0	0	0	1
	1	1	0	A′	A′	0	0	A



۲ـ به کمک یک دیکدر و یک گیت OR جدول درستی زیر را پیاده‌سازی کنید.

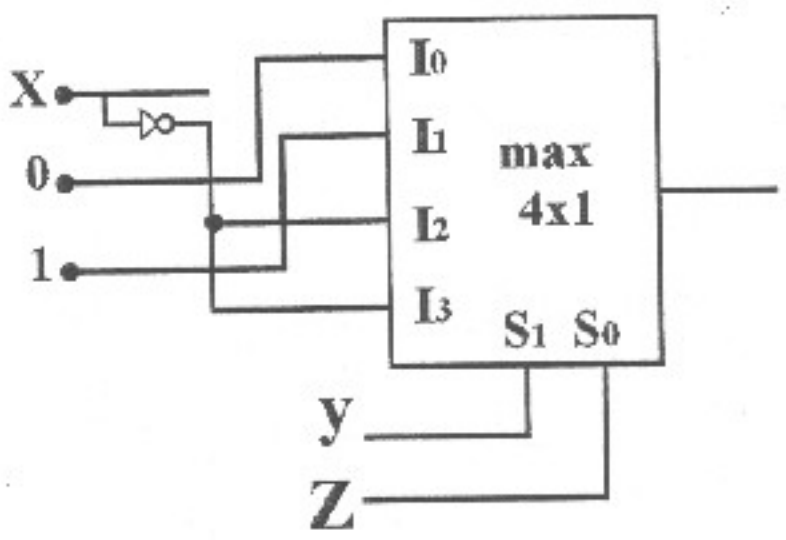
	x	y	z	F
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	0

x \ yz	00	01	11	10
0	1 0	1 1	3	1 2
1	1 4	5	7	1 6

$F = x'y' + z'$

۳- به کمک یک مالتی پلکسر تابع f را طراحی کنید.

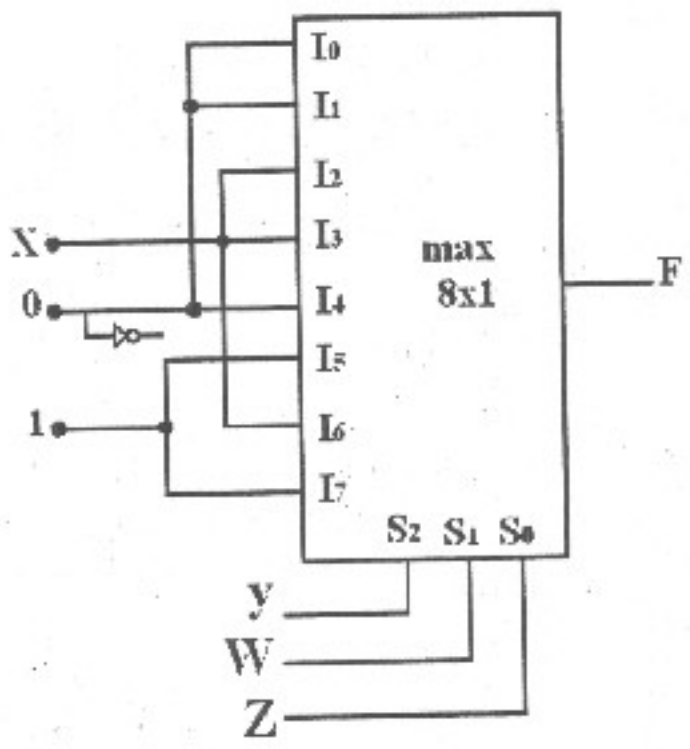
x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



ورودی‌ها	I_0	I_1	I_2	I_3
X′	0	1	2	3
X	4	5	6	7
	0	1	X′	X′

۴- مداری با چهار ورودی طراحی کنید که یک خروجی داشته باشد. خروجی در زمانی یک می‌شود که یا w , x یک باشند و یا زمانیکه y , z یک باشند این مدار را به کمک مولتی‌پلکسر طراحی کنید.

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

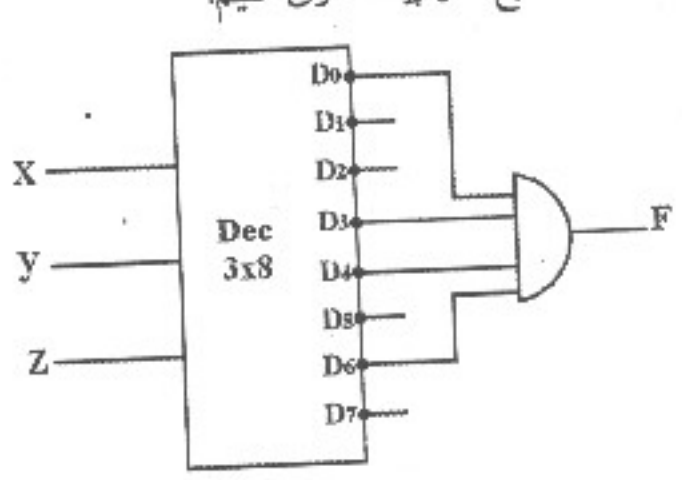


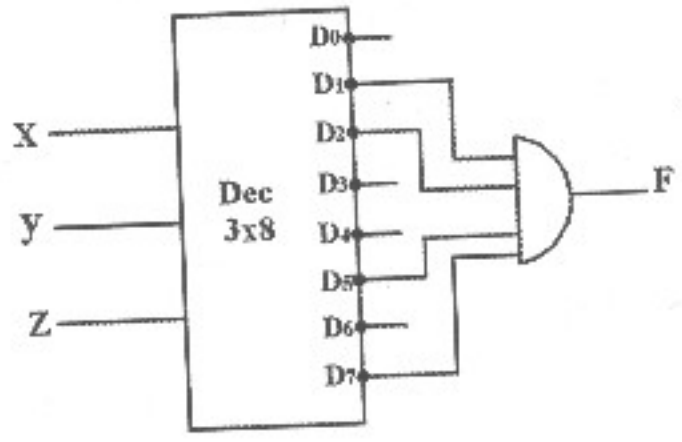
ورودی‌ها	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
X′	0	1	2	3	4	5	6	7
X	8	9	10	11	12	13	14	15
	0	0	X	X	0	1	X	1

۵- راجع به طراحی زیر تحقیق کنید: فرض کنید تابع F بصورت زیر تعریف شده باشد:

می‌خواهیم به کمک یک دیکدر Active low و گیت‌های AND و NAND تابع F را پیاده‌سازی کنیم.

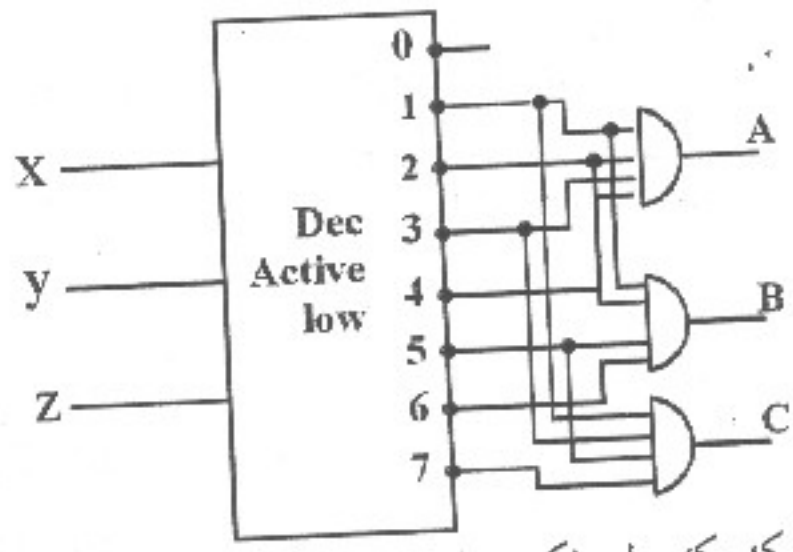
	x	y	z	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	1





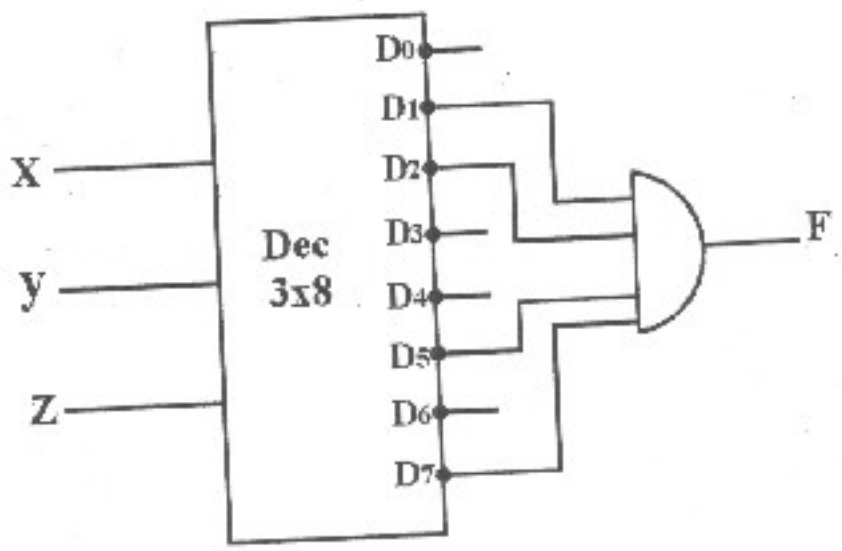
۶- به کمک یک دیکدر Active low و گیت NAND مداری طراحی کنید که مکمل 2 ورودی‌ها را در خروجی نمایش دهد؟ (مدار دارای 3 ورودی باشد)

	x	y	z	A	B	C
0	0	0	0	0	0	0
1	0	0	1	1	1	1
2	0	1	0	1	1	0
3	0	1	1	1	0	1
4	1	0	0	1	0	0
5	1	0	1	0	1	1
6	1	1	0	0	1	0
7	1	1	1	0	0	1



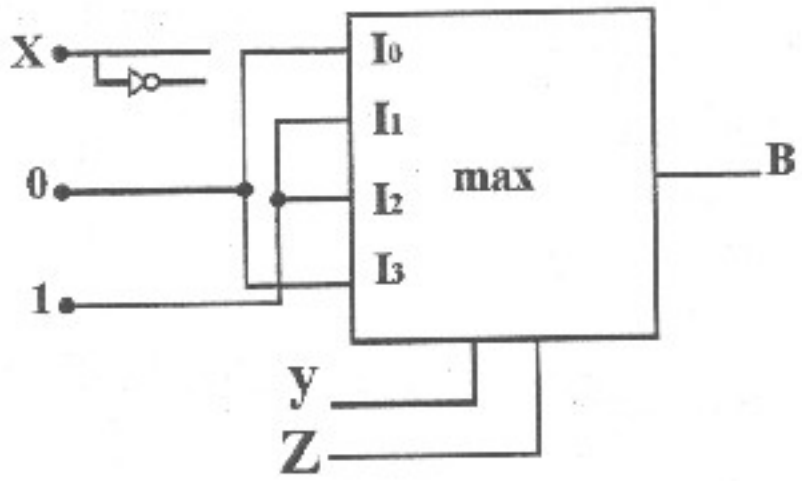
۷- به کمک یک مولتی‌پلکسر مداری طراحی کنید که خروجی، ورودی به اضافه یک باشد.(**توضیحات:** اگر ورودی 7 بود خروجی صفر می‌شود. مدار دارای 3 ورودی باشد.)

	x	y	z	A	B	C
0	0	0	0	0	0	1
1	0	0	1	0	1	0
2	0	1	0	0	1	1
3	0	1	1	1	0	0
4	1	0	0	1	0	1
5	1	0	1	1	1	0
6	1	1	0	1	1	1
7	1	1	1	0	0	0

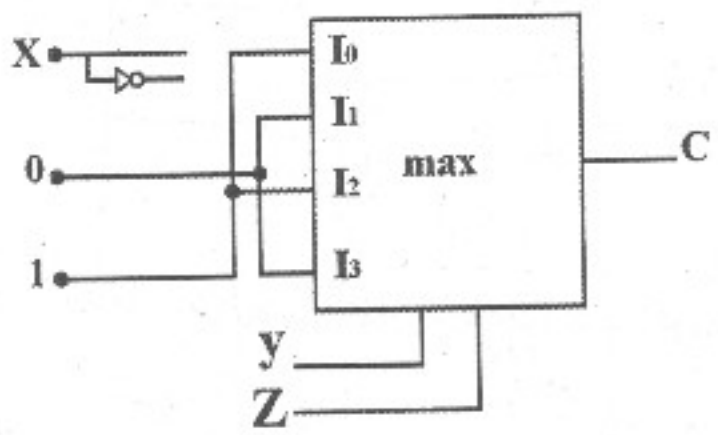


ورودی‌ها	I_0	I_1	I_2	I_3
X′	0	1	2	[3]
X	[4]	[5]	[6]	7
	X	X	X	X′

ورودی‌ها	I_0	I_1	I_2	I_3
X'	0	[1]	[2]	3
X	4	[5]	[6]	7
	0	1	1	0

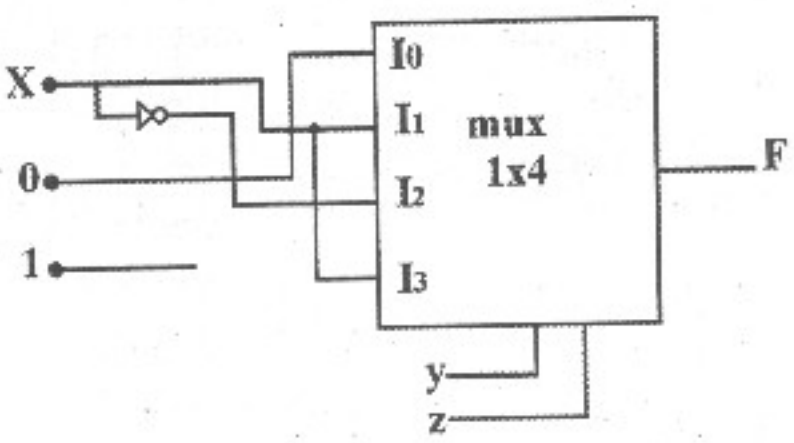


ورودی‌ها	I_0	I_1	I_2	I_3
X'	[0]	1	[2]	3
X	[4]	5	[6]	7
	1	0	1	0



۸- به کمک یک mux تابع f را طراحی کنید.

آنهایی که f آنها مساوی یک است دورشان خط می‌کشیم.



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

	I_0	I_1	I_2	I_3
x'	(0)	1	(2)	3
x	(4)	(5)	6	(7)
	1	x	x'	x

۹- به کمک یک Mux تابع F را طراحی کنید.

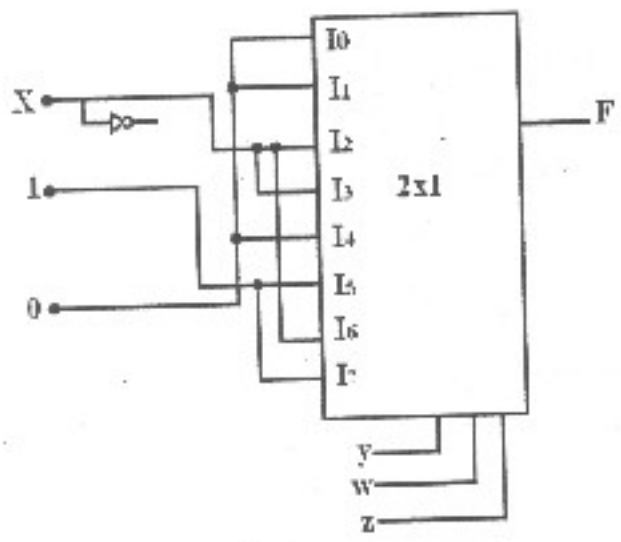
F مداری است با چهار ورودی و یک خروجی، خروجی در زمانی یک می‌شود که یا x , w هر دو یک باشند و یا y , z هر دو یک باشند

ترتیب ورودی‌ها نیز به شکل زیر می‌باشد. x y w z

$2^4 = 16$

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

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
x'	0	1	2	3	4	5	6	7
x	8	9	10	11	12	13	14	15
	0	0	X	X	0	1	X	1



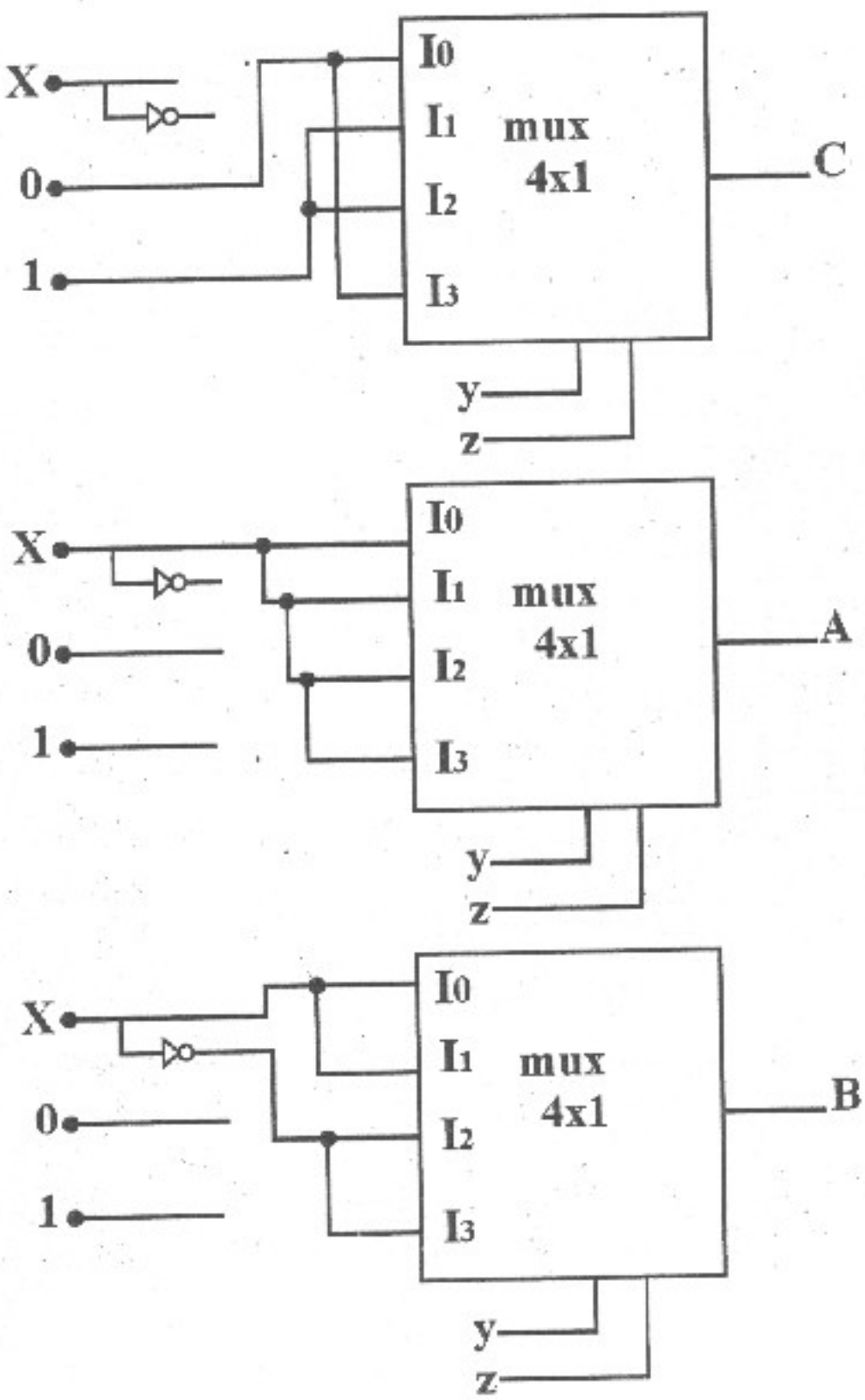
۱۰ـ مدار تبدیل کُد باینری به کد gray را به سه بیت ورودی را به کمک Mux طراحی کنید؟

	x	y	z	A	B	C
0	0	0	0	0	0	0
1	0	0	1	0	0	1
2	0	1	0	0	1	1
3	0	1	1	0	1	0
4	1	0	0	1	1	0
5	1	0	1	1	1	1
6	1	1	0	1	0	1
7	1	1	1	1	0	0

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

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

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



۱۱ - مداری طراحی کنید که ورودی آن یک عدد سه بیتی باشد و خروجی آن مربع ورودی باشد.

مطلوب است:

الف - پیاده‌سازی (عادی)

ب - پیاده‌سازی تمام NAND

ج - طراحی به کمک دیکدر Active high و گیت OR

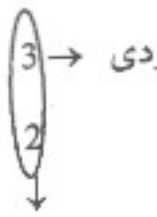
د - طراحی به کمک دیکدر Active low و گیت NAND

ه - طراحی به کمک مولتی پلکسر (Mux)

x	y	z	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0
0	1	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	1
1	1	0	1	0	0	1	0	0
1	1	1	1	1	0	0	0	1

$7^2 = 49 \rightarrow 110001$

به تعداد خروجی برای طراحی جدول کارنو رسم می کنیم.



تعداد خانه‌های جدول کارنو

- برای خروجی A جدول کارنو احتیاج نداریم زیرا تنها یک‌ها برای عدد 6 و7 است که تنها در یک بیت اختلاف دارند و ورودی متناظر آن بیت حذف می‌شود.

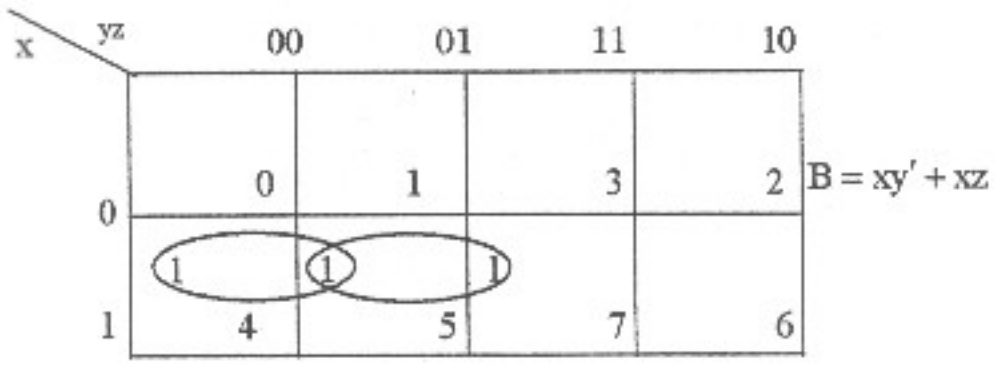
- خروجی D هم مانند A است.

- خروجی E هم چون همه صفر است پس مساوی صفر است.

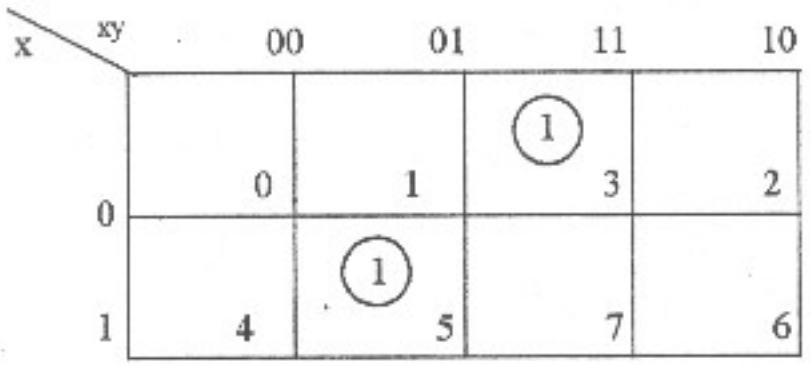
- خروجی F هم دقیقاً مانند ستون ورودی z است پس مساوی z می‌شود.

- برای خروجی B و C هم جدول کارنو رسم می کنیم.

$A = xy$
$D = yz'$
$E = 0$
$F = z$

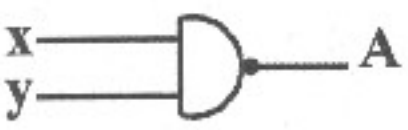


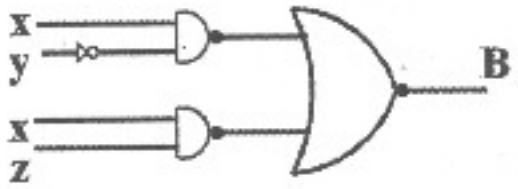
$B = xy' + xz$

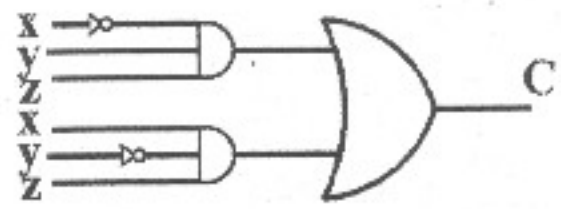


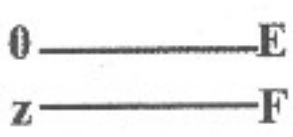
$c = xy'z + x'yz$

الف ـ پیاده‌سازی (عادی)

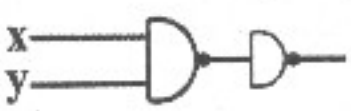


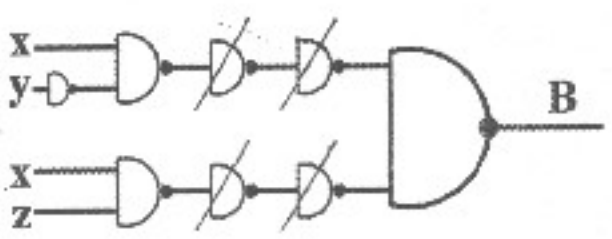


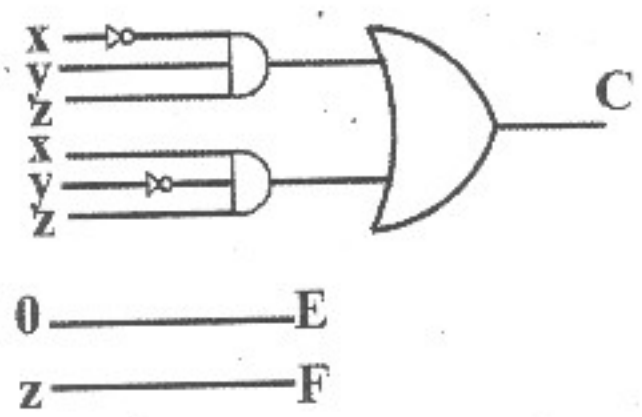


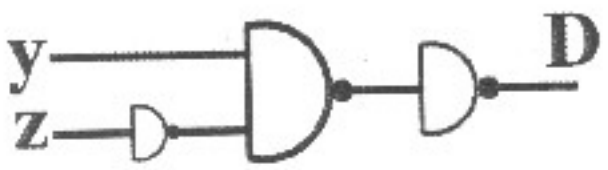


ب ـ پیاده‌سازی تمام NAND





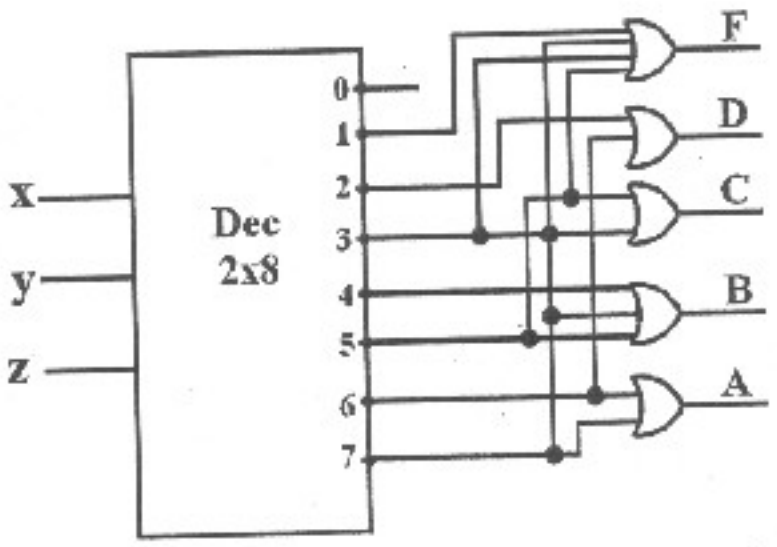




ج - طراحی به کمک دیکدر Active high و گیت OR

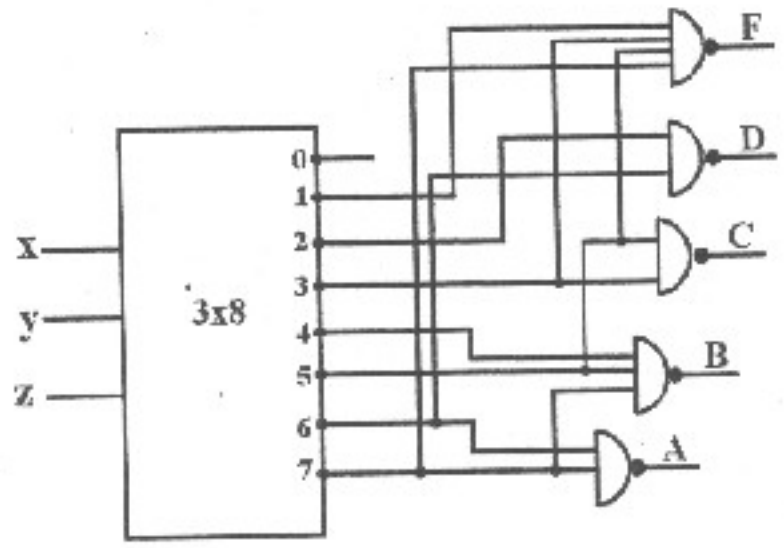
نکته:

Dec
- Actrive high → OR → یک‌های خروجی
- Active high → NOR → صفرهای خروجی
- Active low → AND → صفرهای خروجی
- Active low → NAND → یک‌های خروجی



نکته: اگر بخواهیم به کمک دیکدر مداری طراحی کنیم که n خروجی داشته باشد باید از n گیت استفاده کنیم اما تنها یک دیکدر برای طراحی کافی است.

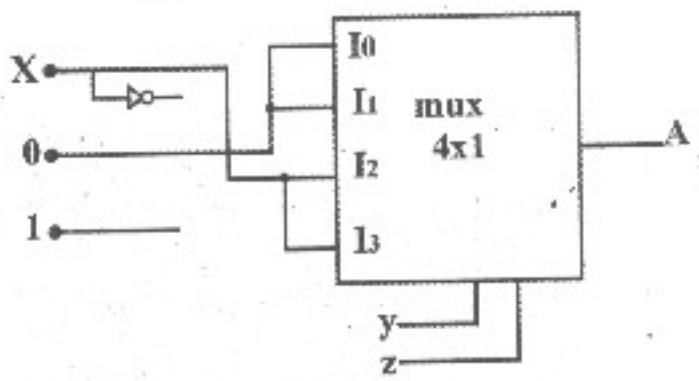
د - طراحی به کمک دیکدر Active low با گیت NAND



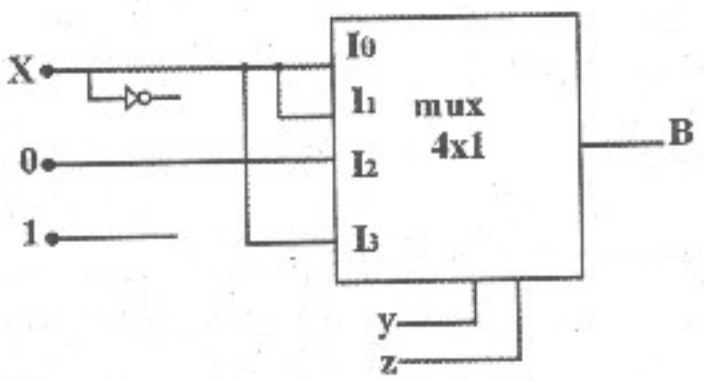
ه ـ طراحی به کمک مولتی‌پلکسر (Mux).

نکته: به تعداد خروجی مدار باید Mux داشته باشیم.

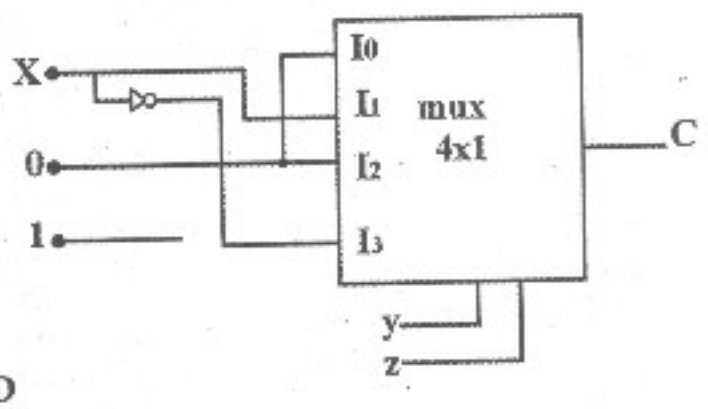
	I_0	I_1	I_2	I_3
x′	0	1	2	3
x	4	5	⑥	⑦
	0	0	x	x



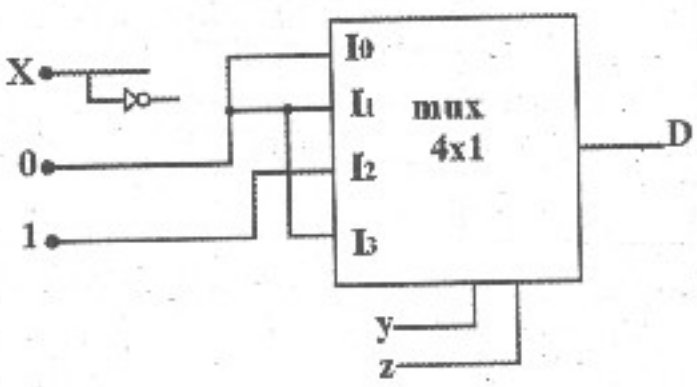
	I_0	I_1	I_2	I_3
x′	0	1	2	3
x	④	⑤	6	⑦
	x	x	0	x



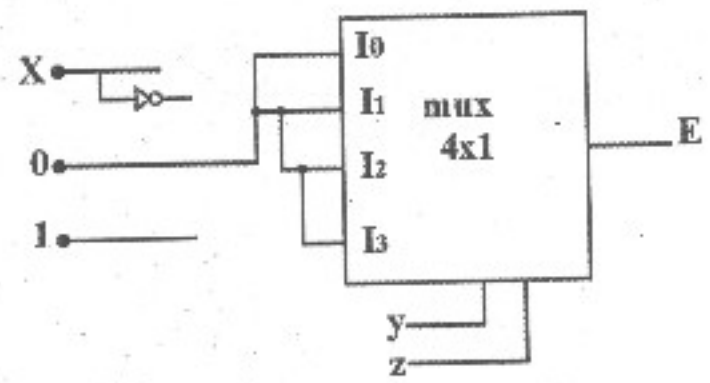
	I_0	I_1	I_2	I_3
x′	0	1	2	③
x	4	⑤	6	7
	0	x	0	x′



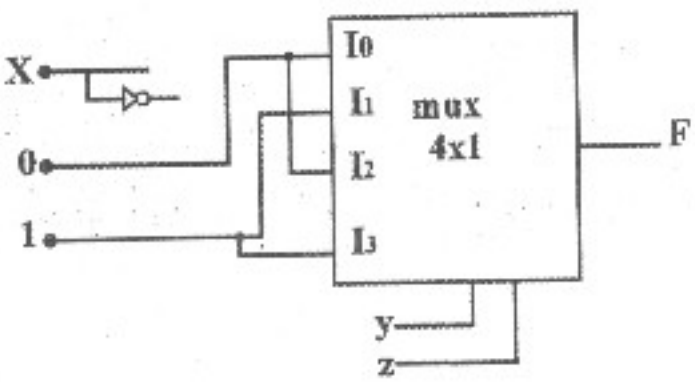
	I_0	I_1	I_2	I_3
x′	0	1	②	3
x	4	5	⑥	7
	0	0	1	0



	I_0	I_1	I_2	I_3
x′	0	1	2	3
x	4	5	6	7
	0	0	0	0

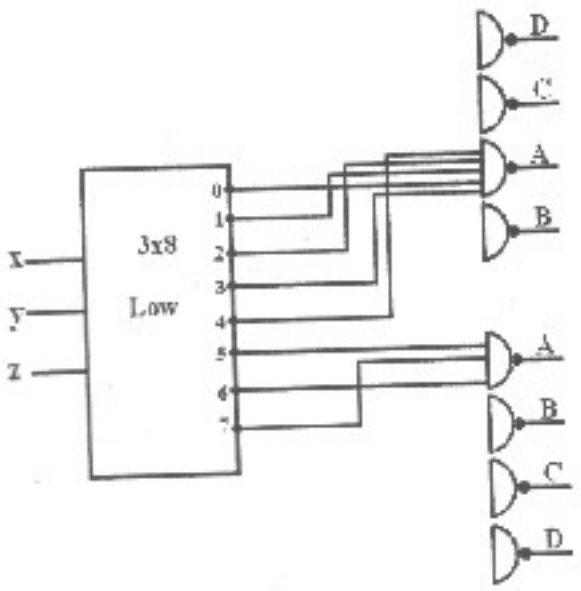


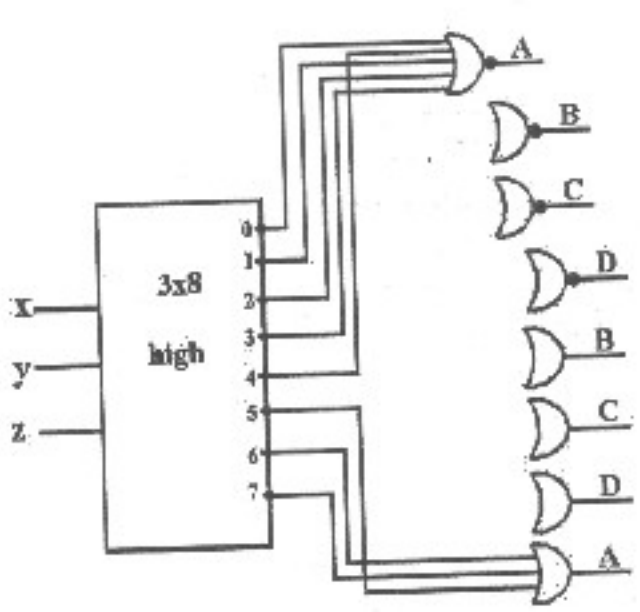
	I_0	I_1	I_2	I_3
x′	0	①	2	③
x	4	⑤	6	⑦
	0	1	0	1



۱۲- به کمک دیکدرهای Active low و Active high و مالتی‌پلکسر مداری طراحی کنید که ورودی را (به صورت باینری) به معادل EX -3 آن تبدیل کند.

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





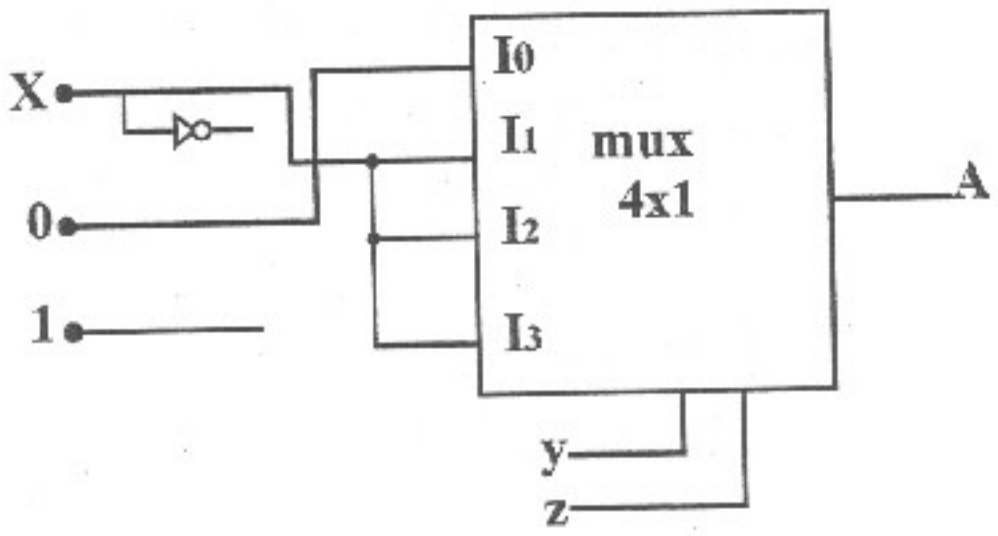
	I_0	I_1	I_2	I_3
x'	0	1	2	3
x	4	⑤	⑥	⑦
	0	x	x	x

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

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

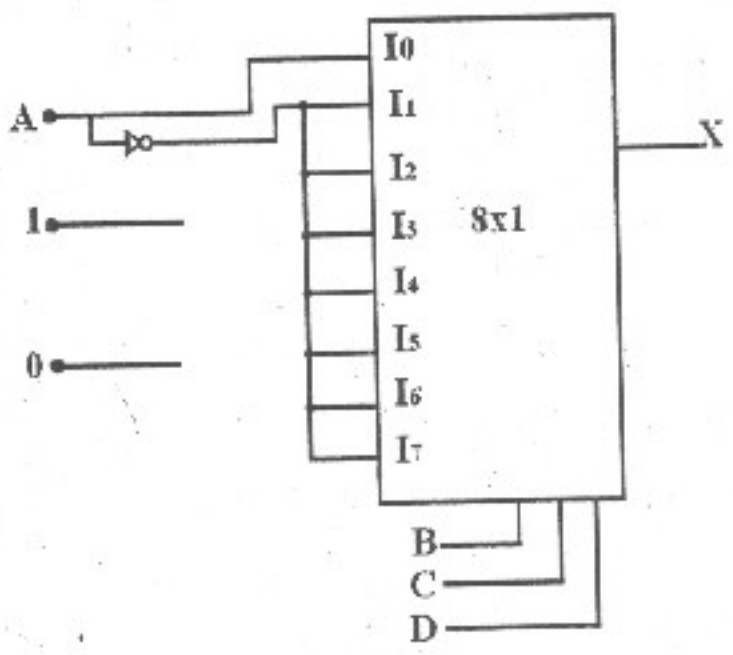
	I_0	I_1	I_2	I_3
x'	⓪	1	②	3
x	④	5	⑥	7
	1	0	1	0

برای D , C , B هم Mux رسم می کنیم.



۱۳ - مداری با چهار ورودی به کمک Mux طراحی کنید که خروجی مکمل 2 ورودی باشد.

	A	B	C	D	x	y	w	z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1
2	0	0	1	0	1	1	1	0
3	0	0	1	1	1	1	0	1
4	0	1	0	0	1	1	0	0
5	0	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	1	0	0	0
9	1	0	0	1	0	1	1	1
A	1	0	1	0	0	1	1	0
B	1	0	1	1	0	1	0	1
C	1	1	0	0	0	1	0	0
D	1	1	0	1	0	0	1	0
E	1	1	1	0	0	0	1	0
F	1	1	1	1	0	0	0	1



برای z , w , y هم شکل رسم شود.

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	0	①	②	③	④	⑤	⑥	⑦
A	⑧	9	10	11	12	13	14	15
	A	A′	A′	A′	A′	A′	A′	A′

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	0	①	②	3	4	⑤	⑥	7
A	8	⑨	⑩	⑪	12	⑬	⑭	15
	0	1	1	0	0	1	1	0

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	0	(1)	(2)	(3)	(4)	5	6	7
A	8	(9)	(10)	(11)	(12)	13	14	15
	0	1	1	1	1	0	0	0

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
A′	0	(1)	2	(3)	4	(5)	6	(7)
A	8	(9)	10	(11)	12	(13)	14	(15)
	0	1	0	1	0	1	0	1

مجموعه تست‌ها:

۱- مدار داخلی یک مالتی پلکسر 4×1 و Active High از چه گیت‌هایی تشکیل شده است؟

۱) 4 گیت AND و یک گیت OR و 2 گیت NOT

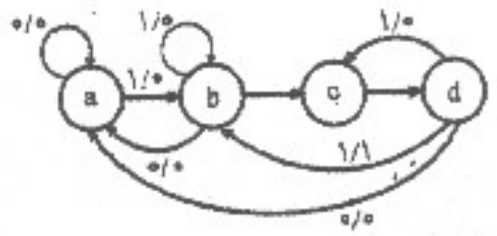
۲) 4 گیت AND و یک گیت OR و 4 گیت NOT

۳) 4 گیت AND و دو گیت OR و 2 گیت NOT

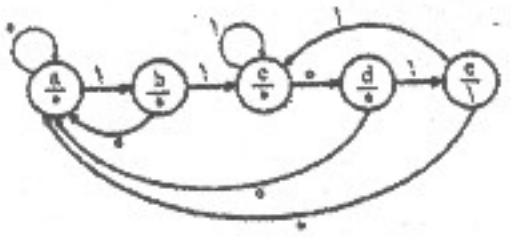
۴) 4 گیت AND و دو گیت OR و 4 گیت NOT

۲- یک State Diagram به صورت Moore رسم کنید که 1101 را روی ورودی پیدا می‌کند. وقتی این Sequence پیدا شد خروجی یک می‌شود.

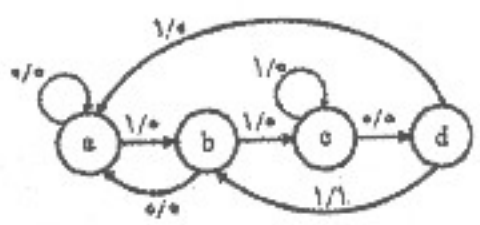
۱)



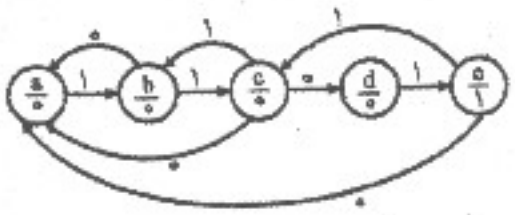
۲)



۳)



۴)



۳- نمودار حالت مقابل را با استفاده از دو فلیپ‌فلاپ D پیاده‌سازی کنیم. تابع فلیپ‌فلاپ مربوط به بیت با ارزش بیشتر را مولتی پلکسر MUX_1 و تابع ورودی فلیپ‌فلاپ دیگر را MUX_0 می‌سازد. کدام گزینه ورودی‌های مولتی پلکسرها را مشخص می‌کند؟

ورودی‌های $select_0$ و $select_1$ مربوط به مولتی پلکسر را به ترتیب Q_0 و Q_1 وصل می‌کنیم.

۱) $MUX_1 : I_0 = x \quad I_1 = \bar{x} \quad I_2 = \bar{x} \quad I_3 = \bar{x}$
$MUX_0 : I_0 = \bar{x} \quad I_1 = x \quad I_2 = 1 \quad I_3 = \bar{x}$

۲) $MUX_1 : I_0 = 0 \quad I_1 = 0 \quad I_2 = x \quad I_3 = \bar{x}$
$MUX_0 : I_0 = x \quad I_1 = \bar{x} \quad I_2 = 1 \quad I_3 = 1$

۳) $MUX_1 : L_0 = 0 \quad L_1 = x \quad L_2 = x \quad L_3 = \bar{x}$
$MUX_0 : L_0 = x \quad L_1 = \bar{x} \quad L_2 = 1 \quad L_3 = \bar{x}$

۴) $MUX_1 : I_0 = x \quad I_1 = 0 \quad I_2 = x \quad I_3 = \bar{x}$
$MUX_0 : I_0 = 1 \quad I_1 = x \quad I_2 = 1 \quad I_3 = \bar{x}$

۴- جدول Prime Implicant زیر برای تابع f بدست آمده است. ساده‌ترین فرم تابع عبارت است از:

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

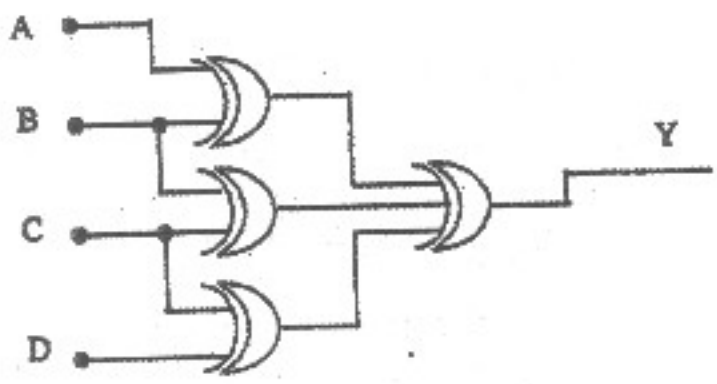
۱) $f = PI_1 + PI_3 + PI_4 + PI_6$

۲) $f = PI_4 + PI_1 + PI_3 + PI_7$

۳) $f = PI_3 + PI_7 + PI_1 + PI_5$

۴) $f = PI_2 + PI_3 + PI_6 + PI_7$

۵- کدام گزینه رابطه حاکم بر مدارشکل زیر را درست تعریف می‌کند؟



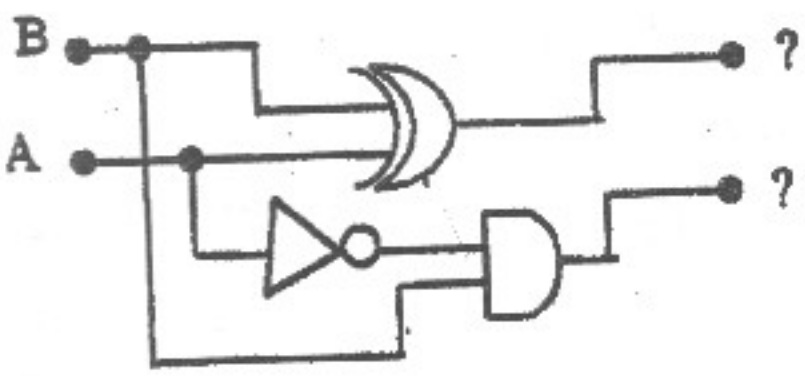
۱) $Y = (A \oplus B) + (C \oplus D)$

۲) $Y = A \oplus B \oplus C \oplus D$

۳) $Y = (A \oplus B) + (B \oplus C) + (C \oplus D)$

۴) $Y = (A+B) \oplus (B+C) \oplus (C+D)$

۶- مداری که در شکل زیر ملاحظه می‌کنید چه عملی را انجام می‌دهد؟



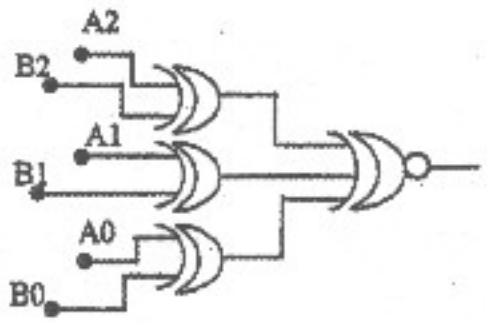
۱) یک جمع کننده تک بیتی است با احتساب دو بر قبلی (FA)

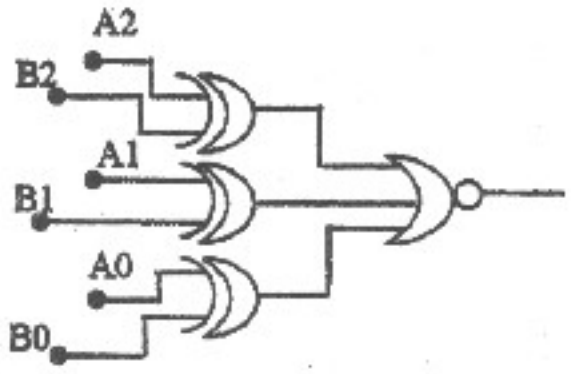
۲) یک تفریق کننده تک بیتی است با احتساب دو بر یک قبلی (FS)

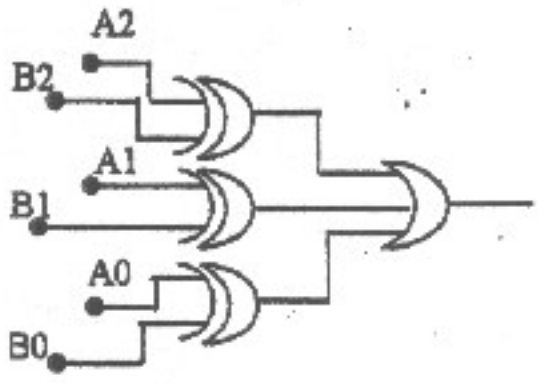
۳) یک تفریق کننده تک بیتی است بدون احتساب دو بر قبلی (HS)

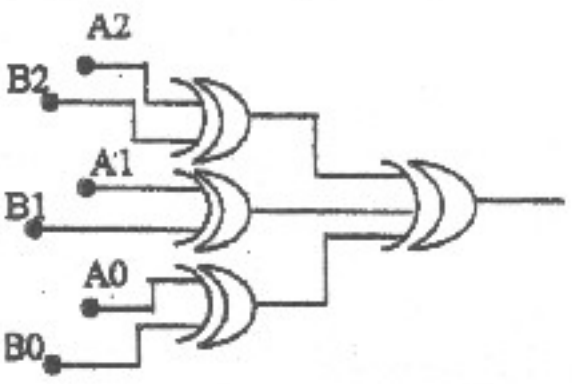
۴) یک جمع کننده تک بیتی است بدون احتساب دو بر یک قبلی (HA)

۷- کدامیک از مدارهای زیر یک مقایسه‌کننده 3 بیتی را نشان می‌دهد که عدم تساوی A و B باعث 0 شدن خروجی آن می گردد؟

۱)


۲)


۳)


۴)


۸- می‌خواهیم یک رمزگشای 3 به 8 یا (3 TO 8 Decoder) را بسازیم. مدار این رمز گشا دارای سه ورودی، 8 خروجی و یک پایه کنترل است، و فقط یکی از خروجی‌ها تحت شرایط زیر در سطح 0 فعال می‌شود (Active Low):

الف : چنانچه یک پایه فعال کننده مدار به نام E' در سطح 0 فعال شود.

ب : سه ورودی مدار در سطوحی متناسب با یک عدد بین 7 و 0 فعال شده باشند.

تعداد دروازه‌های لازم برای این مدار کدامند؟

۱) 4 دروازه NOT و 8 دروازه NAND چهار ورودی
۲) 3 دروازه NOT و 8 دروازه AND چهار ورودی
۳) 4 دروازه NOT و 8 دروازه NAND سه ورودی
۴) ۳ دروازه NOT و 8 دروازه AND سه ورودی

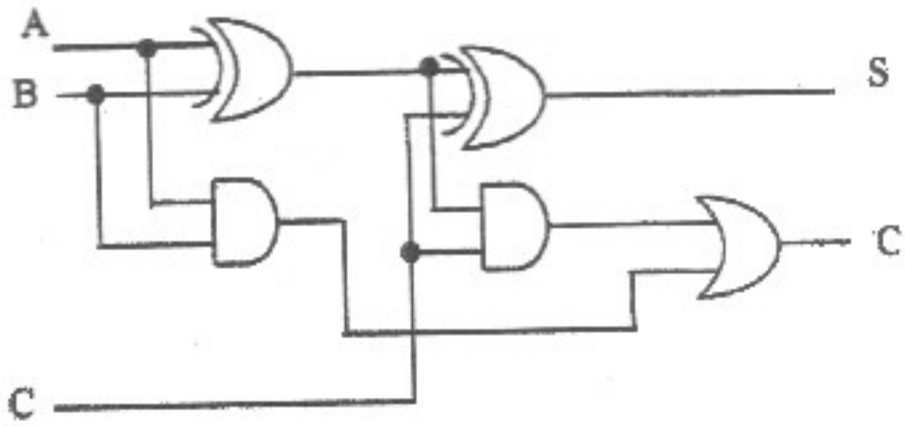
۹- در کدام گزینه تعداد دروازه‌های منطقی مورد نیاز برای یک مالتی پلکسر 4 به 1 (یک از چهار) درست معرفی شده است؟

۱) دو دروازه NOT ، چهار دروازه OR سه ورودی، یک دروازه AND چهار ورودی
۲) دو دروازه NOT ، چهار دروازه AND سه ورودی، یک دروازه OR چهار ورودی
۳) یک دروازه NOT ، چهار دروازه OR سه ورودی، یک دروازه AND چهار ورودی
۴) یک دروازه NOT ، چهار دروازه AND سه ورودی، یک دروازه OR چهار ورودی

۱۰- برای فرستادن 55 خط اطلاعات به صورت Multiplexer نیاز به چند خط کنترل (Select input) داریم.

۱) 6 ۲) 5 ۳) 8 ۴) 4

۱۱ـ شکل زیر چیست؟

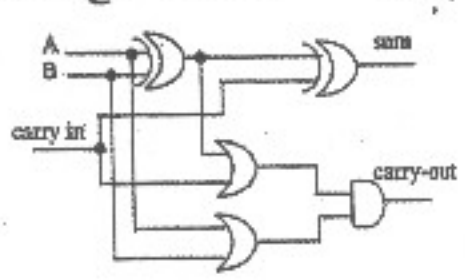


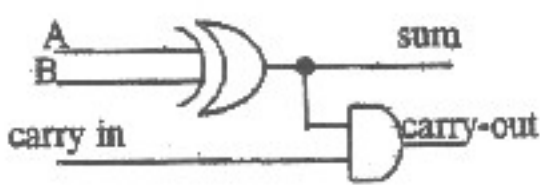
۱) جمع کننده ناقص
۲) تفریق کننده کامل
۳) تفریق کننده ناقص
۴) جمع کننده کامل

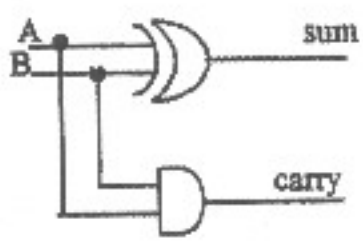
۱۲ـ از دو دیکودر 3×8 با پایه Enable کدام دیکودر را می‌توان ساخت؟

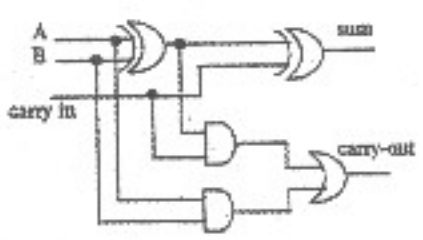
۱) 6×8 ۲) 4×16 ۳) 4×8 ۴) 6×16

۱۳ـ کدام گزینه کاملترین واحد جمع کننده دوبینی (Full Adder) قبل توسعه به تعداد بیت بالاتر را نشان می‌دهد؟

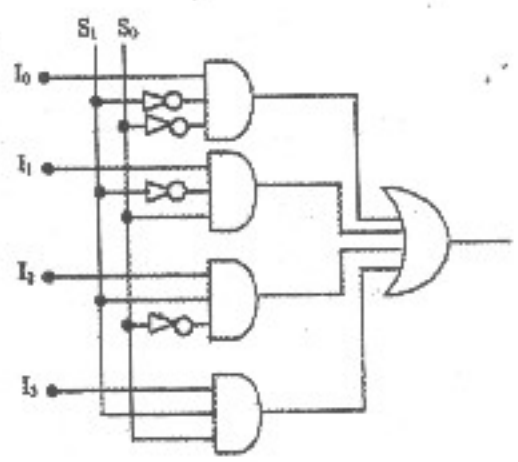
۱)


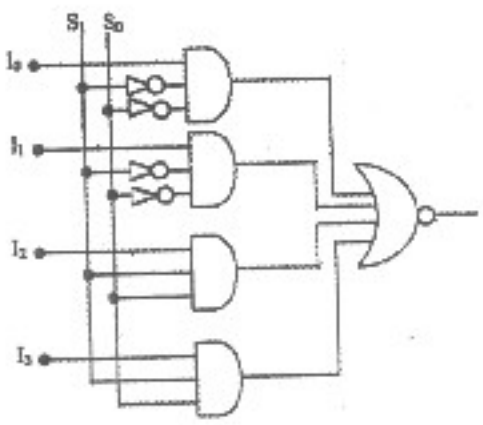
۲)


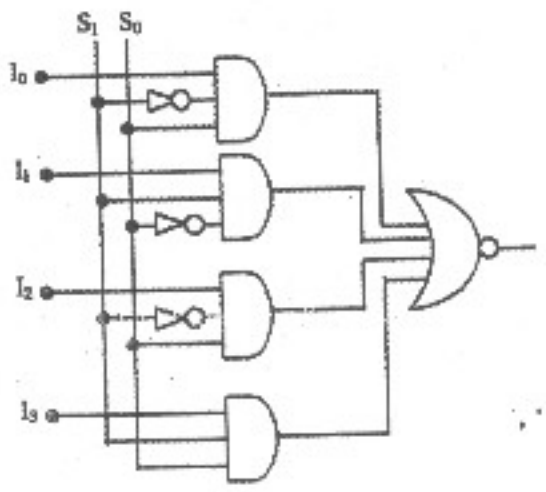
۳)


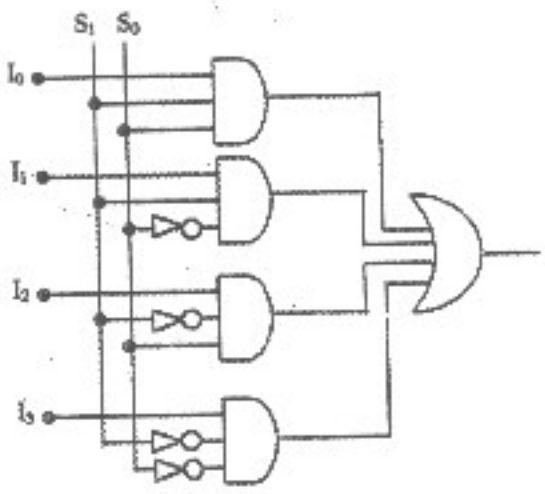
۴)


۱۴ـ کدام مدار منطقی می‌تواند بعنوان یک گزینشگر چهار بیتی (4 – bit Multiplexer) بکار رود. لازم است متناسب با عدد مبنای دو در پایه‌های گزینشگر S_1 S_0 ورودی متناظر با آن به خروجی مربوط شود.

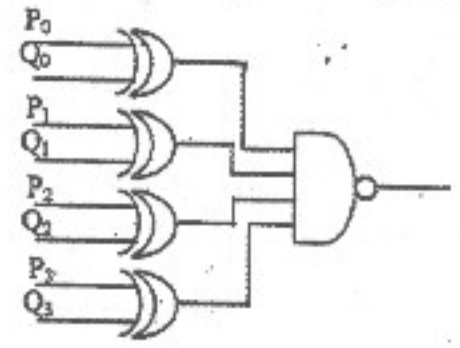
۱)


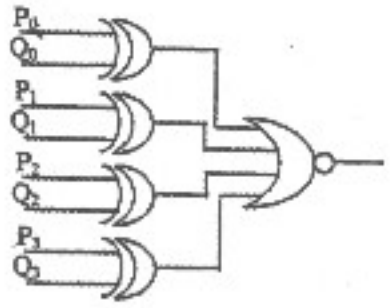
۲)


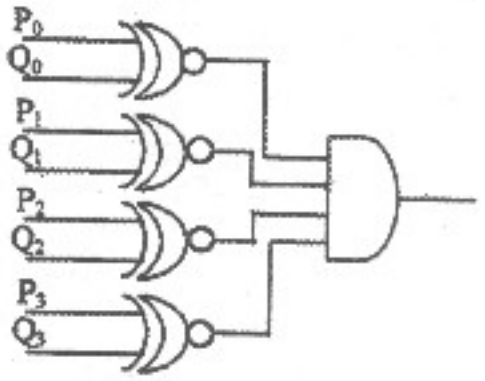
۳)


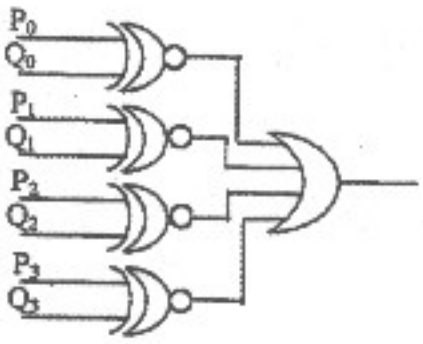
۴)


۱۵ ـ برای مقایسه دو عدد چهاربیتی P و Q و تعیین تساوی و یا عدم تساوی آن‌ها کدام مدار مناسب است. لازم است در صورت تساوی P و Q خروجی مدار 1 شود.

۱)


۲)


۳)


۴)


۱۶ ـ با استفاده از مدارهای مجتمع با مقیاس متوسط چگونه می‌توان یک $\frac{1}{23}$decoder ایجاد نمود؟

۱) یک $\frac{1}{8}$decoder و هشت $\frac{1}{4}$decoder

۲) یک $\frac{1}{8}$decoder و شانزده $\frac{1}{2}$dec

۳) یک $\frac{1}{4}$dec و چهار $\frac{1}{8}$dec

۴) یک $\frac{1}{4}$dec وهشت $\frac{1}{4}$dec

۱۷ ـ با استفاده از مدارهای MUX 4 به 1 و گیت‌های منطقی لازم یک مدار مقایسه‌گر 2 بیتی می‌خواهیم طراحی کنیم. (مقایسه دو عدد 2 بیتی) با کدام عناصر زیر می‌توان عمل را انجام داد، در صورتی که اگر $A \geq B$ باشد خروجی مدار 1 و اگر $A < B$ باشد خروجی صفر می‌باشد؟

(فرض کنید همه ورودی‌ها و خروجی MUX از نوع active high هستند)

۱) یک مدار MUX و یک NOT

۲) دو مدار MUX 4 به 1 و یک NOR

۳) یک NAND ، یک NOT و یک MUX 4 به 1

۴) دو مدار MUX 4 به 1 و یک NAND

۱۸ ـ در یک سیستم کنترلی که دارای چهار ورودی وچهار خروجی و دو ورودی کنترل می‌باشد و طبق جدول زیر عمل می‌نماید توابع خروجی E و F کدام است؟

I_1	I_2	E	F	G	–
0	0	1	1	1	1
0	1	A	B	C	D
1	0	$\bar{A}$	$\bar{B}$	$\bar{C}$	$\bar{D}$
1	1	0	0	0	0

۱) $E=\bar{I}_1\bar{I}_2+\bar{I}_1 I_2 A+I_1\bar{I}_2\bar{A}$
$F=\bar{I}_1\bar{I}_2+\bar{I}_1 I_2 B+I_1\bar{I}_2\bar{B}$

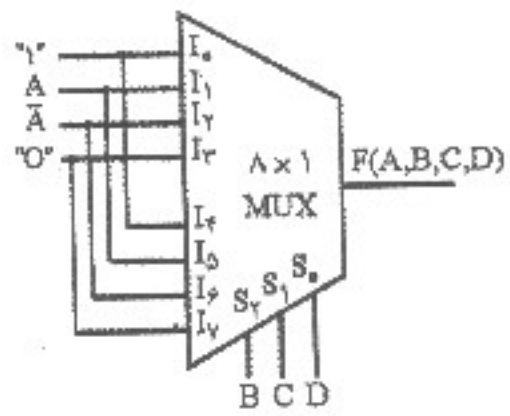
۲) $E=\bar{I}_1\bar{I}_2 A+I_1\bar{I}_2\bar{A}+I_1 I_2$
$F=\bar{I}_1\bar{I}_2 B+I_1\bar{I}_2\bar{B}+I_1 I_2$

۳) $E=A+\bar{I}_1\bar{I}_2+I_1 I_2$
$F=B+\bar{I}_1\bar{I}_2+I_1 I_2$

۴) $E=I_1 I_2+\bar{I}_1 I_2+AI_1\bar{I}_2+\bar{A}\bar{I}_1\bar{I}_2$
$F=I_1 I_2+\bar{I}_1 I_2 BI_1\bar{I}_2+\bar{B}\bar{I}_1\bar{I}_2$

۱۹- مداری دارای 4 ورودی A,B,C,D و خروجی F با استفاده از یک مالتی پلکسر 1 × 8 طبق شکل زیر طراحی گردیده است، ترم‌های این مدار که به ازاء آن خروجی high می‌باشد را بنویسید؟

	0	1	2	3	4	5	6	7
$\bar{a}$	0	1	2	3	4	5	6	7
a	8	4	10	11	12	13	14	15



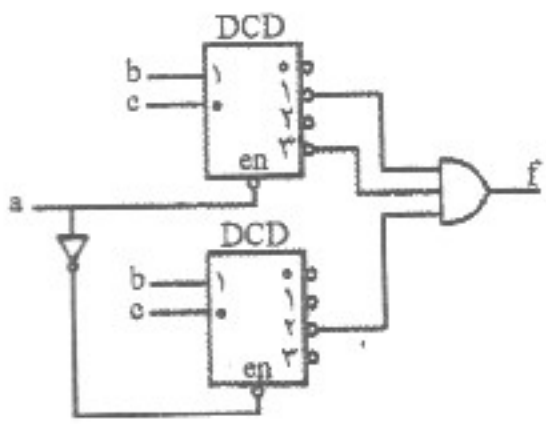
۱) $f(a,b,c,d)=\Sigma 0,4,8,12,14$

۲) $f(a,b,c,d)=\Sigma 1,3,5,7,9,11,13$

۳) $f(a,b,c,d)=\Sigma 0,2,4,6,8,9,12,13$

۴) $f(a,b,c,d)=\Sigma 0,2,6,8,9,12$

۲۰- شکل مقابل از دو دیکودر دو ورودی تشکیل شده که دارای active – low enable می‌باشند. خروجی‌های دیکودرها نیز active – low (فعال - پایین) هستند. تابع خروجی صحیح کدام است؟



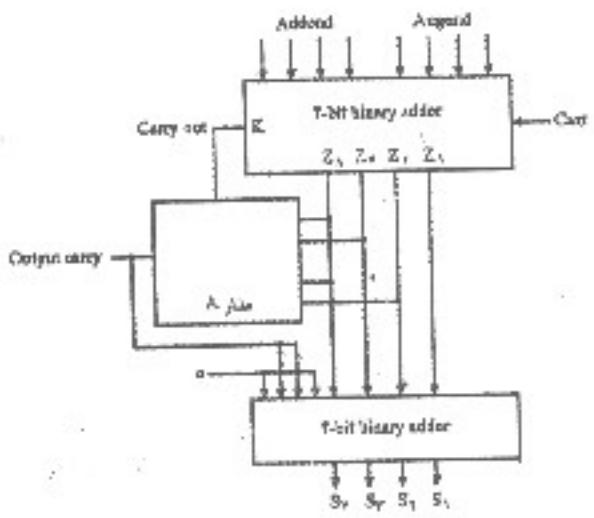
۱) $f(a,b,c)=\Sigma m(1,3,6)$

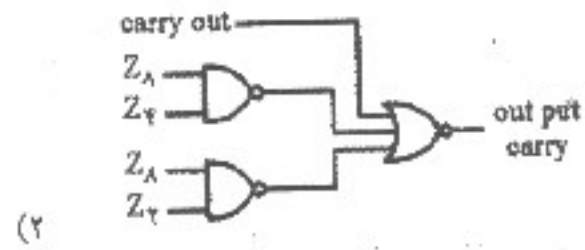
۲) $f(a,b,c)=\Sigma m(2,5,7)$

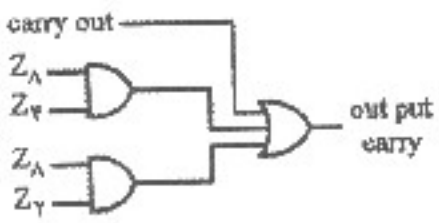
۳) $f(a,b,c)=\Sigma m(0,1,3,4,6)$

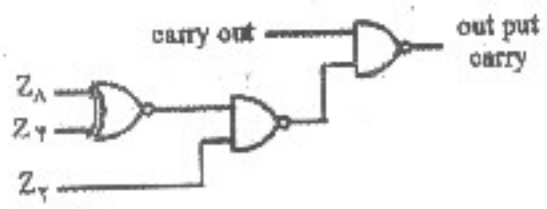
۴) $f(a,b,c)=\Sigma m(0,2,4,5,7)$

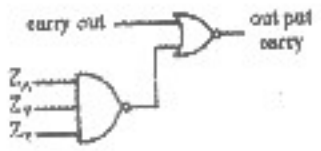
۲۱- توسط یک مدار (A) و دو تمام جمع کننده می‌توان یک جمع کنندهٔ BCD طراحی نمود (مطابق شکل مقابل). مدار A را طراحی نمائید. (کدام گزینه را می‌توان به جای مدار A قرار داد.)



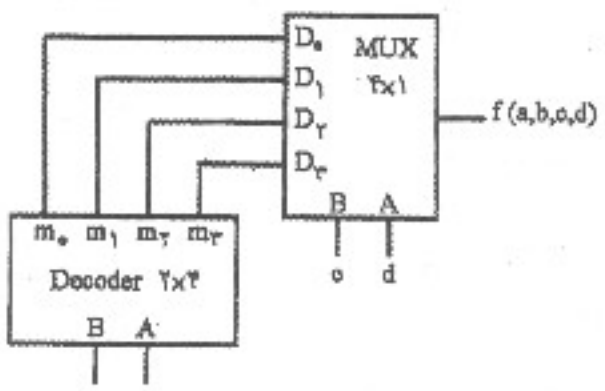
۱)


۲)


۳)


۴)


۲۲ـ مدار مقابل را در نظر بگیرید. B ورودی پرارزش‌تر دیکدر (active high) و مالتی پلکسر است کدام گزینه بیانگر تابع F (a , b , c, d) است؟



۱) F=0

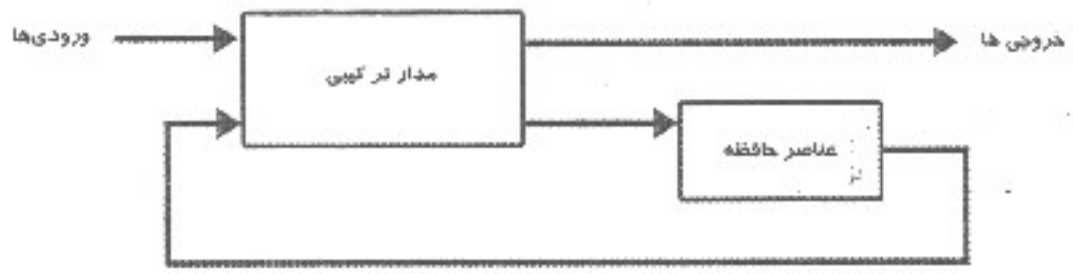
۲) $F=\sum m(0,5,10,15)$

۳) $\overline{F}=\overline{a}\,\overline{b}+ab+\overline{c}\,\overline{d}+cd$

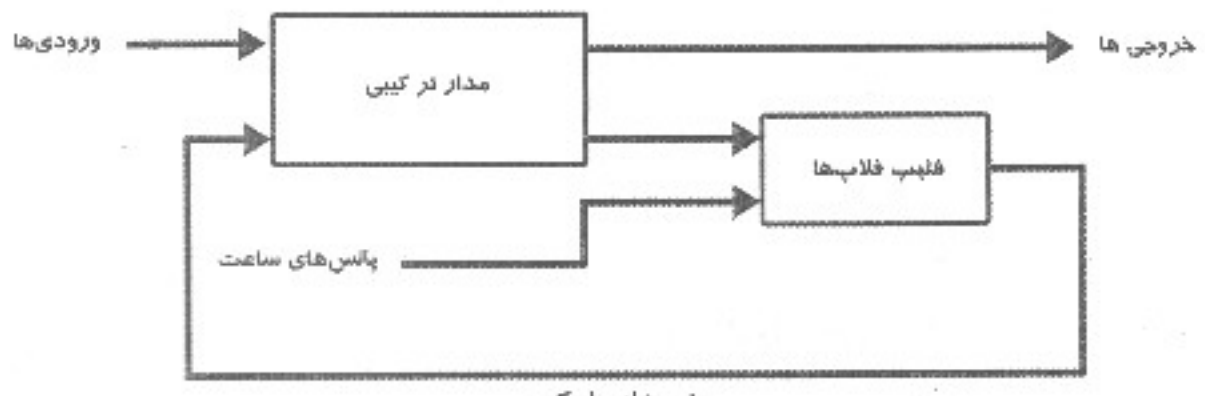
۴) F = 1

مدارهای ترتیبی

مدارهای ترتیبی در اصل همان مدارهای ترکیبی هستند که به آنها حافظه اضافه شده این حافظه‌ها می توانند از فلیپ فلاپ‌ها تشکیل شده باشند.



نمودار بلوکی یک مدار ترتیبی



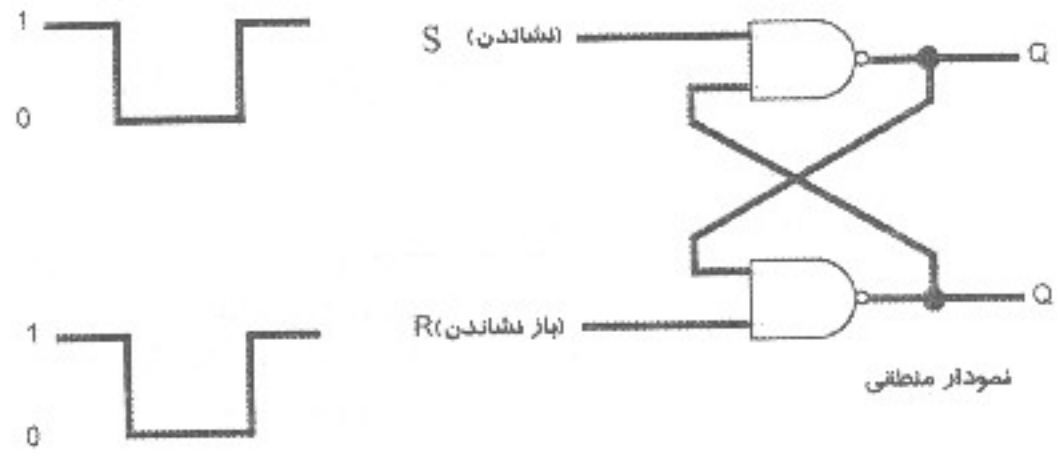
نمودار بلوکی

مدار ترتیبی ساعت‌دار همزمان

فلیپ فلاپ‌ها Flip Flop

یکی از عناصر حافظه هستند که برای نگهداری اطلاعات از آنها استفاده می‌شوند. فلیپ فلاپ‌ها در اصل مهمترین عناصر موجود در مدارهای ترتیبی می‌باشند.

S	R	Q	Q′
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1



لچ SR با گیت‌های NAND

در ساختار فیلپ فلاپ SR که در آن گیت‌های NAND شکل بالا به کار رفته باید توجه داشت که ورودی‌های ان به صورت Active low می‌کنند یعنی جدول درستی آن به شکل زیر تغییر می‌کند.

R	Q	Q′
0	0	1
1	0	1
1	1	0
1	1	0
0	1	1

همان‌طور که ملاحظه می‌شود (0 , 0) برای ورودی‌های (S , R) هیچ تغییری در خروجی مرحله قبل ایجاد نمی‌کند و همچنین (1, 1) برا ورودی‌های (S , R) باعث به وجود آمدن مشکل در آن می‌شود زیرا خروجی‌های (Q′ , Q) که باید همیشه با هم متفاوت باشند هر دو برا یک می‌شوند.

$Q' = \text{Not}\ (Q)$

S	R	Q	Q′
0	1	0	1
0	0	0	1
1	0	1	0
0	0	1	0
1	1	–	–

SR-Flip Flop

خروجی درلحظه t Q_t

خروجی در لحظه t+1 $Q_{(t-1)}$

نشاندن یا set کردن S

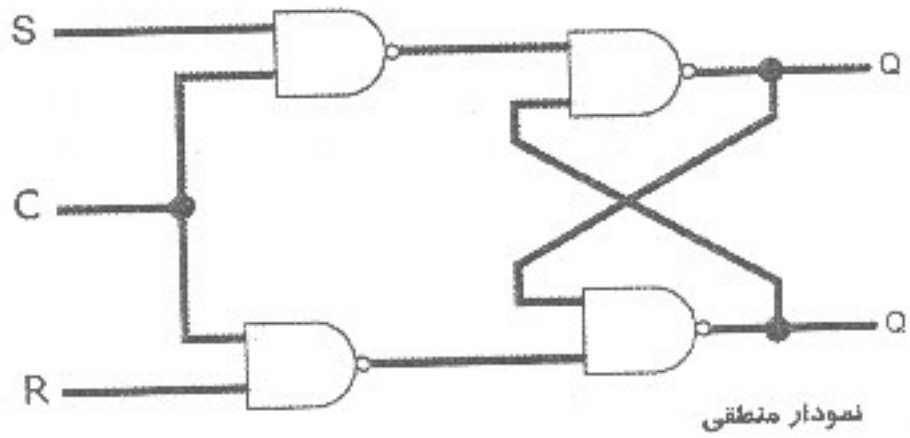
بازنشاندن یا Reset کردن R

S	R	Q_t	$Q_{(t-1)}$	
0	0	0	0	→ Q_t بدون تغییر
0	0	1	1	
0	1	0	0	→ 0 Reset
0	1	1	0	
1	0	0	1	→ 1 Set
1	0	1	1	
1	1	0	-	→ – نامعین
1	1	1	-	

زمانی که SR هر دو یک هستند حالت نامعین یا حالت ناخواسته است. که در این نوع فلیپ فلاپ به وجود می‌آید.

Q \ SR	00	01	11	10
0			×	1
1	1		×	1

$Q_{t-1} = S + QR' \Rightarrow \begin{cases} S.R = 0 \\ S.R \neq 1 \end{cases}$ S و R هر دو با هم نمی‌تواند یک باشند.



نمودار منطقی

لچ SR با ورودی کنترل

در شکل بالا وقتی CLOCK صفر است فلیپ فلاپ عمل نمی‌کند و هر چه از قبل در خروجی مانده نشان می‌دهد ولی اگر یک باشد به شکل یک S . R فلیپ فلاپ عمل می‌کند و خروجی‌ها مطابق جدول صفحهٔ قبل خواهد بود.

Jk - Fliplop

برای از بین بردن حالت ناخواسته در SR – FF فلیپ فلاپ جدیدی به وجود آوردند به نام Jk- FF به کمک خطوط فیدبک که از خروجی Q', Q گرفته شده و به همراه ورودی‌ها به NAND های اولیه وارد شده، مشکل مقدار (1 , 1) را حل کردند.

Q_t خروجی در لحظه t

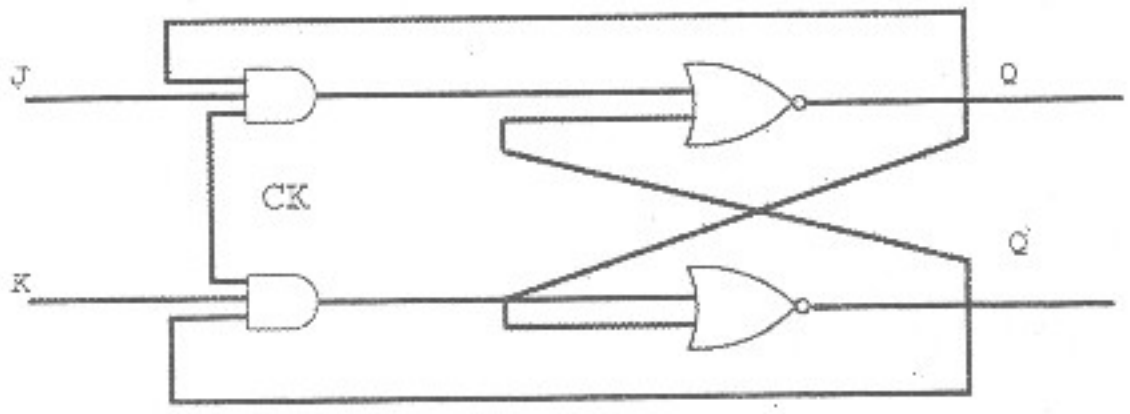
$Q_{(t+1)}$ خروجی در لحظه t+1

نشاندن یا Set کردن J

J	K	Q_t	$Q_{(t+1)}$		خروجی	
0	0	0	0	→	Q_t	بدون تغییر
0	0	1	1			
0	1	0	0	→	0	Reset
0	1	1	0			
1	0	0	1	→	1	Set
1	0	1	1			
1	1	0	1	→	Q'_t	متمم
1	1	1	0			

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

$\Rightarrow Q_{t+1} = jQ'_t + k'Q_t$



D-Flip Flop

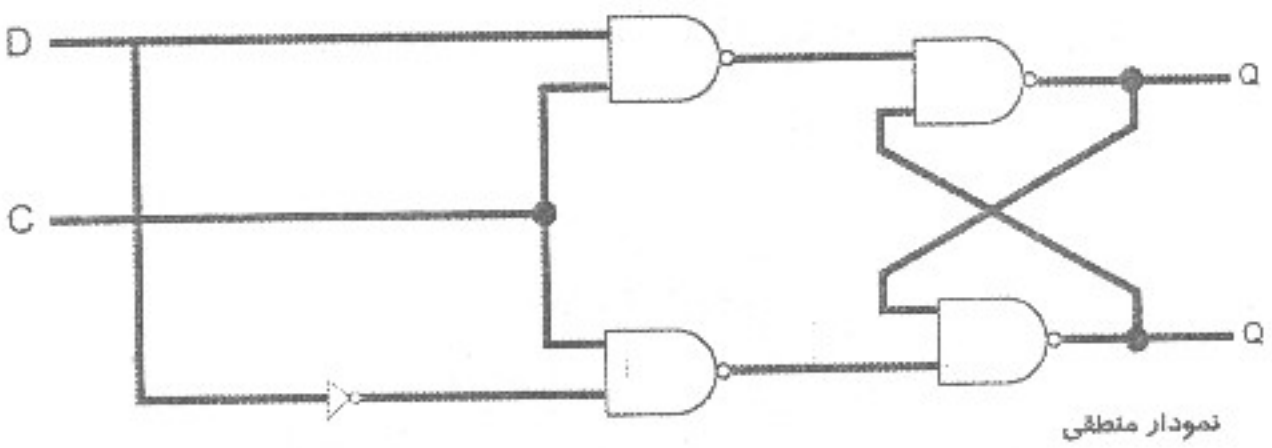
حالتی خاص از SR-FF است.

به آن Data F.F نیز گفته می‌شود. که کارش نگهداری اطلاعات در طول یک CLK می‌باشد. در واقع همان داده‌ای که به ورودی آن اعمال می‌شود را به خروجی انتقال می‌دهد.

$$\begin{cases} D=0 & Q_{t+1}=0 \\ D=1 & Q_{t+1}=1 \end{cases} \Rightarrow Q_{t+1}=D$$

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

همانطور که ملاحظه می‌شود خروجی مرحله قبل هیچ تاثیری در عملکرد D-F.F ندارد و تنها زمانی که CLK فعال شود هر مقداری که روی ورودی قرار دارد به خروجی منتقل می‌شود.



نمودار منطقی

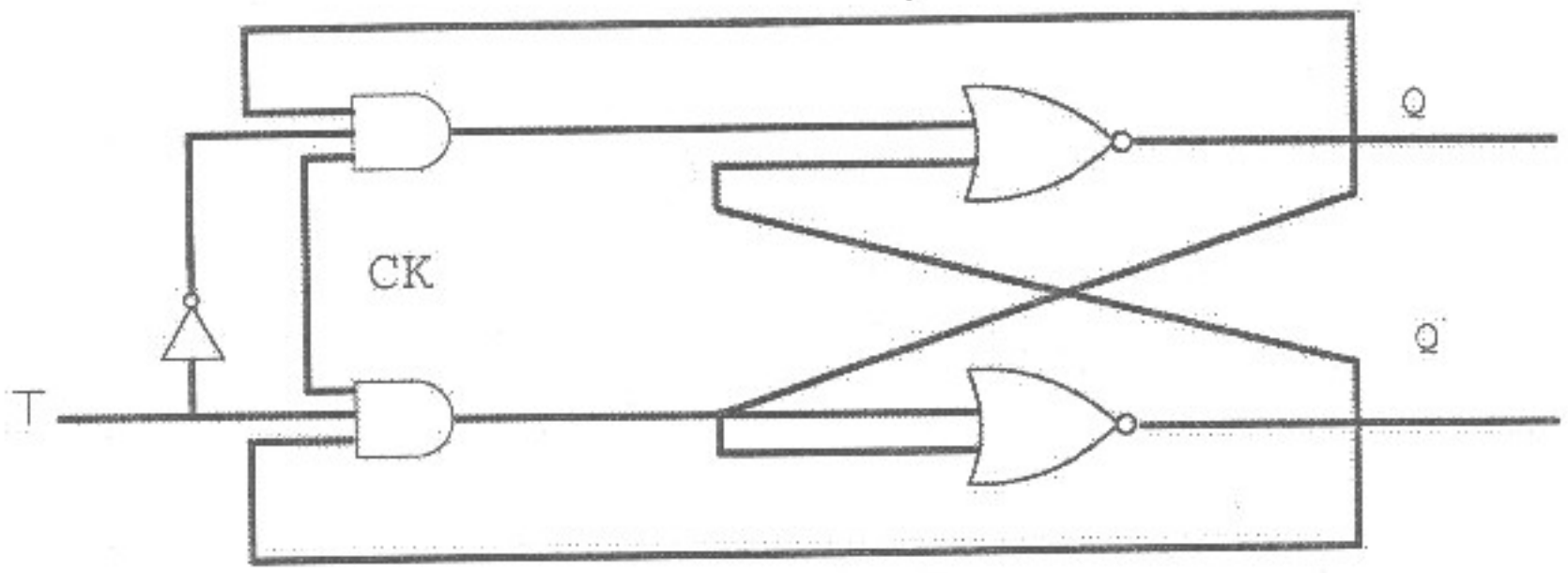
لچ D

T-Flip Flop

T	$Q_{(t)}$	$Q_{(t-1)}$
0	0	0
0	1	1
1	0	1
1	1	0

وقتی T=0 باشد خروجی مرحله بعدی همان خروجی مرحله قبل است.

$$\Rightarrow Q_{t+1}=Q_t \qquad (1)$$

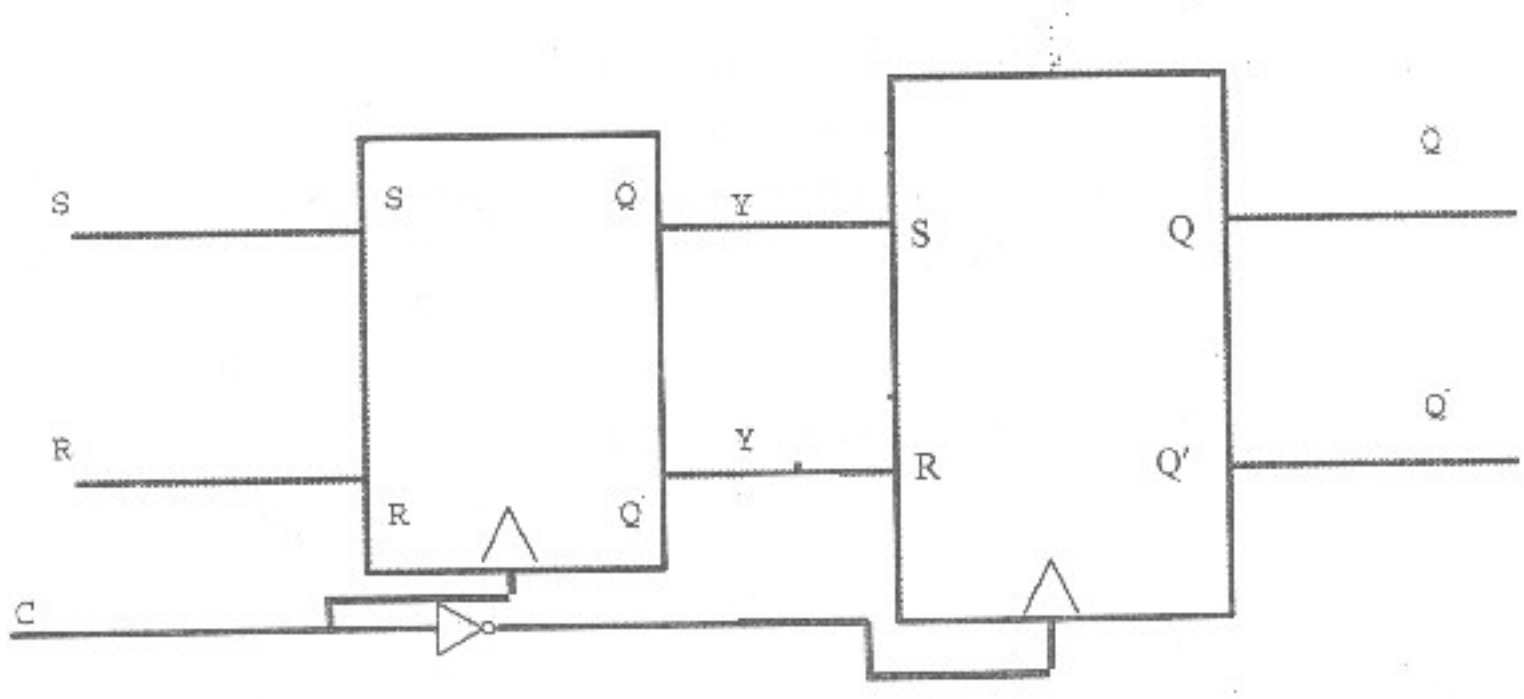


وقتی T=1 باشد خروجی مرحله بعد متمم خروجی مرحله قبل است

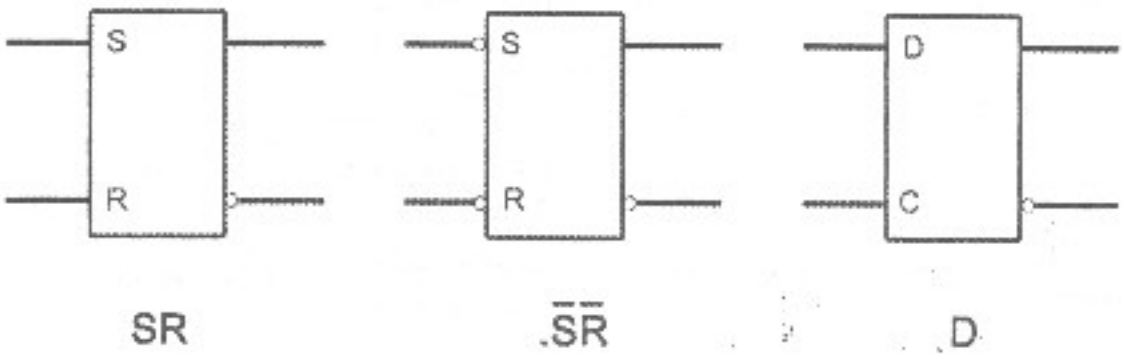
$$T = 1 \Rightarrow Q_{t+1} = Q'_t \qquad (II)$$

تابع متبوع Master Slave

از دو فلیپ فلاپ پشت سر هم که هر دو از یک نوع هستند (D,T,Jk,RS) تشکیل شده است . خروجی فلیپ فلاپ اول وارد فلیپ فلاپ دوم می‌شود و در ورودی آن می‌ماند تا هنگامی که کلاک باعث فعال شدن فلیپ فلاپ دوم شود و خروجی آن فعال شود. و در هر زمان تنها یک فلیپ فلاپ کار می‌کند به این مفهوم که کلاک وارد شده در فلیپ فلاپ دوم، متمم شده کلاک وارد شده به فلیپ فلاپ اول است. یعنی در زمانی که مقدار CLK یک باشد تنها فلیپ فلاپ اول کار می کند و در زمانی که صفر است اولین فلیپ فلاپ غیرفعال می‌شود و فلیپ فلاپ کار می کند.



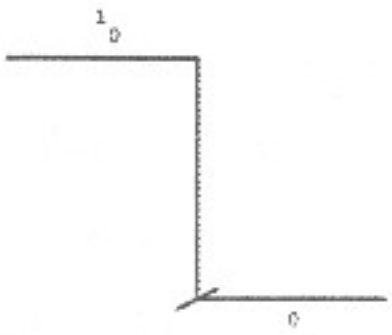
سمبل‌ها گرافیکی

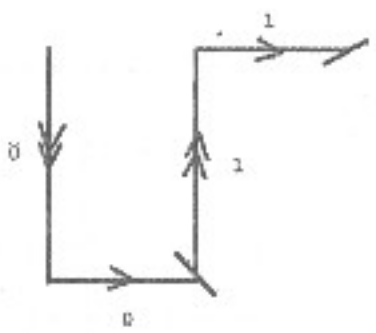


سمبل‌های گرافیکی لچ

انواع کلاک پالس:

کلاک پالس یا حساس به لبه است یا حساس به سطح





حساس به لبه:

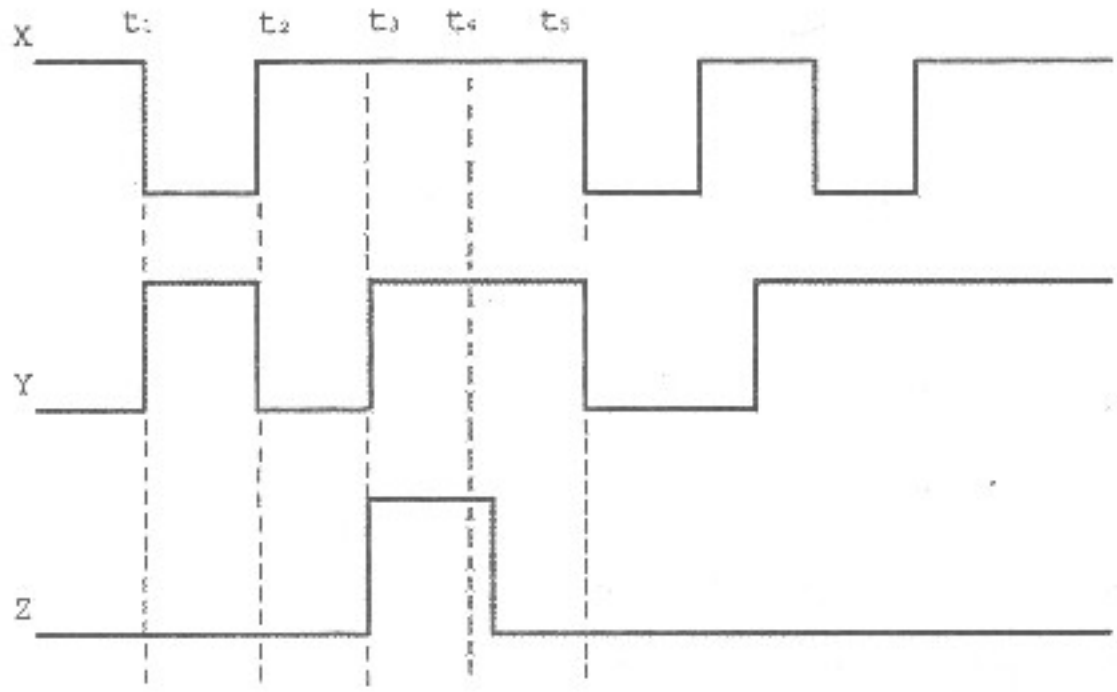
در کلاک پالس حساس به لبه زمانی که حرکت در لبه به سمت پایین است عدد صفر تولید می‌شود تا زمانی که تغییر سمت صورت گیرد به محض تغییر جهت حرکت به سمت بالا عدد یک تولید می‌شود

حساس به سطح:

در این نوع کلاک با شروع سطح مقادیر صفر و یک تولید می‌شود به این مفهوم که وقتی که سطح بالا شروع می‌شود مقدار کلاک 1 و وقتی که سطح پایین شروع می‌شود مقدار کلاک صفر می‌شود این مقادیر تا شروع سطح بعدی به همین شکل می‌ماند.

ورودی‌های مدار ترتیبی:

در لحظه خاصی مثل t=1 نمی‌توان تشخیص داد مقدار x,y,z چقدر است برای مشخص کردن مقدار z,y,x باید از بازه‌های زمانی استفاده کرد.



در بازه $0 < t \le 1$ $\begin{cases} x = 1 \\ y = 0 \\ z = 0 \end{cases}$

در بازه $1 < t \le 2$ $\begin{cases} x = 0 \\ y = 1 \\ z = 0 \end{cases}$

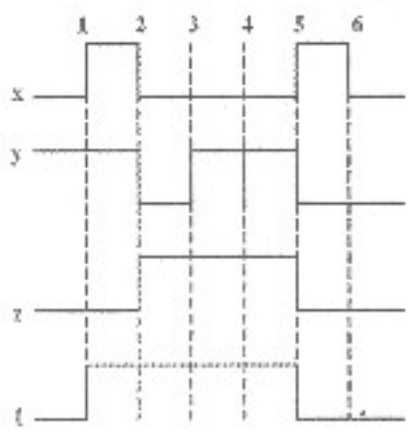
در بازه $2 < t \le 3$ $\begin{cases} x = 1 \\ y = 0 \\ z = 0 \end{cases}$

در بازه $3 < t \le 4$ $\begin{cases} x = 1 \\ y = 1 \\ z = 1 \end{cases}$

در بازه $4 < t \le 5$ $\begin{cases} x = 1 \\ y = 1 \\ z = 0 \end{cases}$

	x	y	z
0<t≤1	1	0	0
1<t≤2	0	1	0
2<t≤3	1	0	0
3<t≤4	1	1	1
4<t≤5	1	1	0

مثال: شکل تابع $F = A + BC'$ را رسم کنید.(با توجه به اینکه کلاک ورودی A و B و C)؟

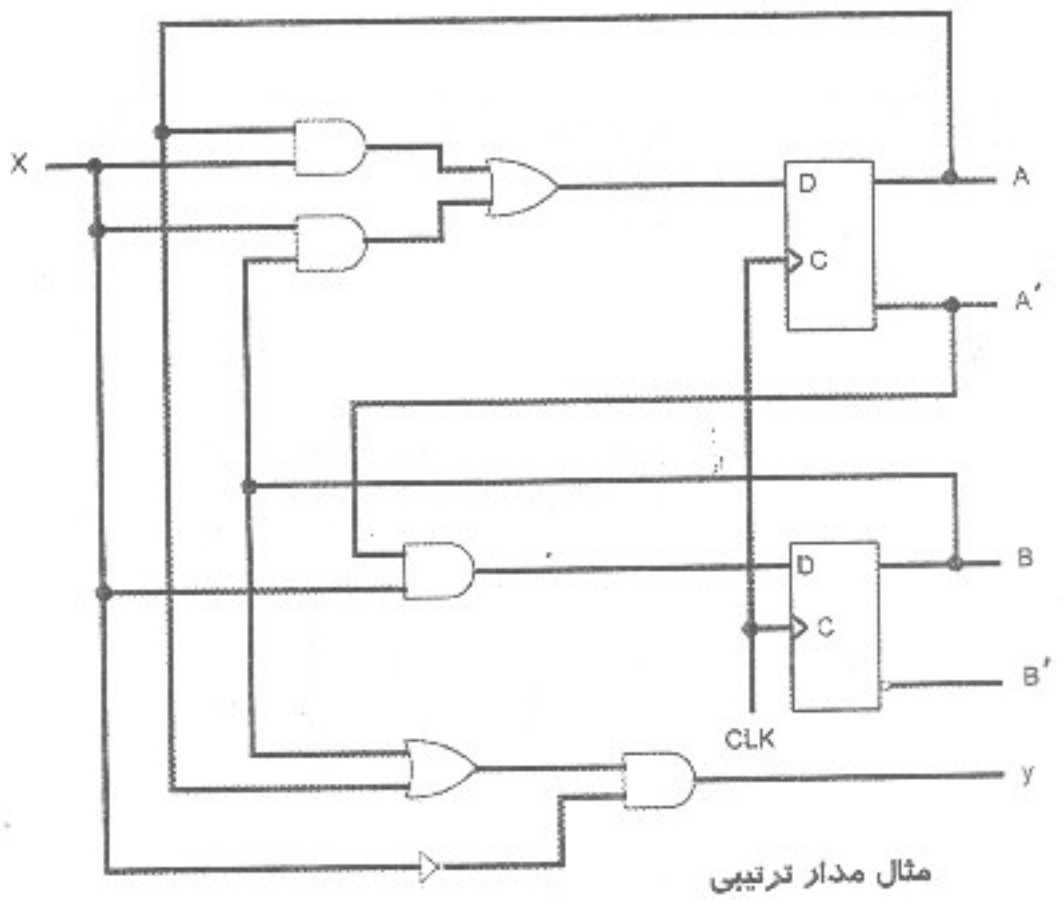


تحلیل مدارهای ترتیبی ساعت‌دار

مرحله ۱ـ به دست آوردن معادلات خروجی بر حسب ورودی و خروجی‌های مرحله قبل

مثال:

با توجه به شکل مدار زیر تحلیل و طراحی را انجام دهید.



مثال مدار ترتیبی

هر ورودی که بدهیم همان را در خروجی می‌گیریم زیرا هر دو فیلیپ فلاپ از نوع D-FFهستند

$$y = x'.(A+B)$$

$$A_{(t+1)} = (A.x)+(B.x)$$

$$B_{(t+1)} = A'.x$$

مرحله ۲ـ بدست آوردن جدول

حالت فعلی			حالت بعدی		
$A_{(t)}$	$B_{(t)}$	x	$A_{(t+1)}$	$B_{(t+1)}$	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

$B_{(t+1)} = A_{(t+1)}$ ، y را بر اساس معادلات خروجی که در مرحله قبل بدست آوردیم محاسبه می‌کنیم.

مرحله ۳ــ دیاگرام حالات

۱ـ تعداد فلیپ فلاپ‌ها را مشخص می‌کنیم تا تعداد حالات مشخص شود .(تعداد حالات در واقع برابر است با 2 به توان تعداد فلیپ فلاپ‌ها)

حالات

A	B
0	0
0	1
1	0
1	1

تعداد فلیپ فلاپ‌ها : n

2^n = تعداد حالات

۲ـ تعداد ورودی‌ها را مشخص می‌کنیم و حالات ممکن را برایش می‌نویسیم . برای یک ورودی (x) دو حالت داریم $\begin{cases} x=0 \\ x=1 \end{cases}$

اگر دو ورودی داشتیم (x,y) چهار حالت دارد $\begin{cases} x=0 \\ x=1 \\ y=0 \\ y=1 \end{cases}$ و برای هر کدام مقدار AB را به دست می‌آوریم.

به این مفهوم که اگر m خط ورودی داشتیم 2^m حالت مختلف برای ورودی‌ها به وجود می‌آید.

در مثال قبل یک ورودی داریم

حالت فعلی		حالت بعدی				خروجی	
		x = 0		x = 1		x = 0	x = 1
A	B	A	B	A	B	y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

با توجه به جدول حالات که در مرحله 2 بدست آوردیم:

وقتی A,B حالت فعلی (۰,۰) باشند و $x=0 \Leftarrow$ AB حالت بعدی (0,0) هستند و y=0

وقتی A,B حالت فعلی (۰,۰) باشند و $x=1 \Leftarrow$ AB حالت بعدی (0,1) هستند و y=0

وقتی A,B حالت فعلی (0,1) باشند و $x=0 \Leftarrow$ AB حالت بعدی (0,0) هستند و y=1

وقتی A,B حالت فعلی (0,1) باشند و $x=1 \Leftarrow$ AB حالت بعدی (1,1) هستند و y=0

وقتی A,B حالت فعلی (1,0) باشند و $x=0 \Leftarrow$ AB حالت بعدی (0,0) هستند و y=1

وقتی A,B حالت فعلی (1,0) باشند و $x=1 \Leftarrow$ AB حالت بعدی (1,0) هستند و y=0

وقتی A,B حالت فعلی (1,1) باشند و $x=0 \Leftarrow$ AB حالت بعدی (0,0) هستند و y=1

وقتی A,B حالت فعلی (1,1) باشند و $x=1 \Leftarrow$ AB حالت بعدی (1,0) هستند و y=0

حال اگر بر فرض بخواهیم خروجی (y) و حالات بعدی (AB) را به کمک گیت‌ها پیاده‌سازی کنیم با توجه به جدول و با توجه به مقادیر یک به دست آمده در ستون خروجی‌ها و حالات بعدی به مانند طراحی مدارهای ترکیبی عمل می‌کنیم.

با توجه به این‌که مقدار خروجی y در زمانی که x=1 است صفر می‌باشد می‌توان دریافت که x به مانند یک کنترل برای y است. به این معنا که می‌توان y را از خروجی گیت ANDی گرفت که یک ورودی آن متمم شده x می‌باشد و با توجه به ستون y در زمان x=0 به مانند خروجی یک OR می‌باشد که B,A ورودی‌های آن هستند.

۳ـ دیاگرام حالات

در بدست آوردن دیاگرام حالت می‌دانیم که باید به تعداد حالات موجود دایره‌ای کشید. و هر کدام را با یک حالت نام‌گذاری کنیم.

سپس با استفاده از جدول حالات شروع به ترسیم خطوط می‌کنیم. به این صورت که با توجه به حالت فعلی و حالت بعدی و همچنین مقادیر ورودی و خروجی می‌توان طراحی را انجام داد به عنوان نمونه در مثال قبل زمانی که در حالت فعلی (1,0) می‌باشیم با گرفتن مقدار ورودی x=0 به حالت بعدی (0,0) می‌رویم و مقدار خروجی ما y=1 می‌شود.

مدل‌های میلی و مور

مدل‌های میلی مدل‌هایی (مدارهایی) هستند که خروجی تابعی از ورودی و حالت فعلی است.

مدل‌های مور مدل‌هایی (مدارهایی) هستند که خروجی فقط تابعی از حالت فعلی است.

مدل‌های میلی و مور دو مدار ترتیبی هستند و برای تشخیص میلی یا مور بودن یک مدار باید حتماً آن را تحلیل کرد.

در مدارهای مور با تغییر ورودی خروجی تغییر نمی‌کند بلکه تنها با تغییر کلاک عوض می‌شود. مزیت مدل‌های میلی انعطاف‌پذیری آن‌ها و معمولاً تعداد حالات کمتر آن‌ها می‌باشد.

شمارنده‌ها جزو مدارهای مور هستند که قابل تبدیل به مدل میلی نمی‌باشند.

دیاگرام حالت مدل میلی

دیاگرام حالت مدل مور

تبدیل مدارهای مور به میلی و میلی به مور:

مثال قبل نمونه‌ای از مدل میلی است مدار علاوه بر AB حالت فعلی به x (ورودی) نیز بستگی دارد.

تحلیل به کمک فلیپ فلاپ‌ها

وقتی در مسئله‌ای گفته می‌شود به صورت ترتیبی حل شود و یا از طریق فلیپ فلاپ‌ها طراحی شود به سود ما است که از فلیپ فلاپ‌ها استفاده کنیم. در مدارهای ترتیبی بیشترین عنصری که استفاده می‌شود فلیپ فلاپ‌ها هستند.

آسان‌ترین فلیپ فلاپ‌ها برای طراحی jk و T فلیپ فلاپ است.

RS فلیپ فلاپ مانند jk است تنها تفاوت در حالت (1,1) در SR است که سیستم حالت نامعین خواهد داشت از SR در زمانی استفاده می‌کنند که بخواهیم ورودی‌ها را به صورت دستی وارد کنیم. در این‌صورت اگر ورودی اشتباه وارد شود فلیپ فلاپ به حالت (1,1) می‌رود و سیستم متوقف می‌شود.

الویت استفاده از فیلپ فلاپ‌ها در طراحی:

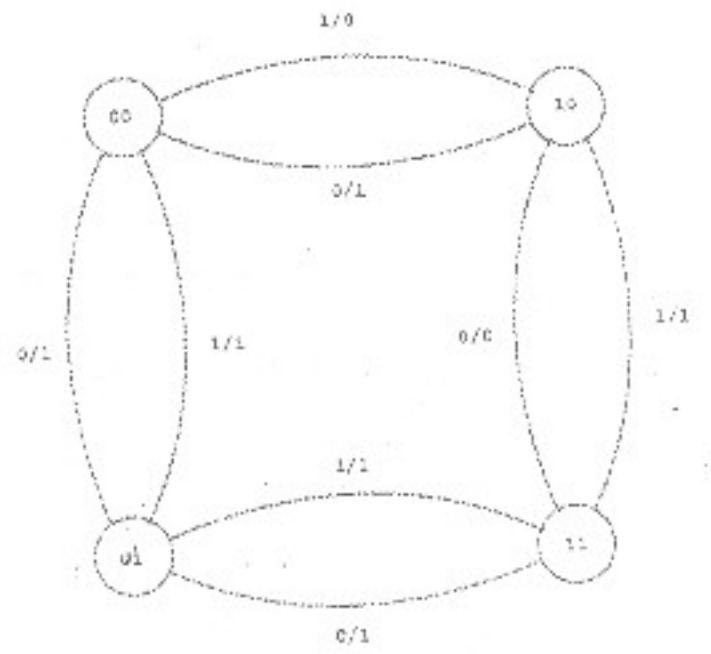
1-Jk
2-SR
3-T
4-D

برای طراحی با فلیپ فلاپ باید ورودی‌های آن‌ها را (SR , Jk) به کمک توابع ترکیبی تنظیم (طراحی) می‌کنیم

برای این کار می‌توان از جدول حالات استفاده کرد به این صورت که باید حالات فعلی ورودی را بنویسیم و از روی دیاگرام حالات و یا شکل مسئله حالت‌های بعدی و خروجی را بدست آوریم. سپس با استفاده از جدول بدست آمده و با مقایسه حالت فعلی و حالت بعدی و همچنین استفاده از جدول تحریک فلیپ فلاپ‌ها ورودی‌های لازم را برای اعمال به فلیپ فلاپ‌ها به دست می‌آوریم.

به عنوان مثال برای بدست آوردن جدول حالات به شکل فوق از روی دیاگرام حالات (۱۳-۴) خواهیم داشت.

حالات فعلی		ورودی	حالات بعدی		خروجی
A	B	X	A	B	Y
0	0	0	0	1	1
0	0	1	1	0	0
0	1	0	1	1	1
0	1	1	0	0	1
1	0	0	0	0	1
1	0	1	1	1	1
1	1	0	1	0	0
1	1	1	0	1	1



جدول تحریک فلیپ فلاپ‌ها

خروجی فلیپ فلاپ‌ها		ورودی فلیپ فلاپ‌ها					
$Q_{(t)}$	$Q_{(t+1)}$	S	R	J	K	D	T
0	0	0	x	0	x	0	0
0	1	1	0	1	x	1	1
1	0	0	1	x	1	0	1
1	1	x	0	x	0	1	0

در این جدول با توجه به $Q_{(t+1)}$, $Q_{(t)}$ که خروجی فلیپ فلاپ‌ها در حالت فعلی و بعدی هستند ورودی‌ها را می‌نویسیم.

به عنوان مثال:

برای SR فلیپ فلاپ داریم:

اگر به ورودی‌های SR , (0,0) بدهیم هیچ تغییری در خروجی مرحله قبل به وجود نمی‌آید.

اگر به ورودی‌های SR , (1,1) بدهیم حالت نامعین است.

اگر به ورودی SR, (1,0) بدهیم یعنی Set, R=0,S=1 می‌کند.

اگر به ورودی SR , (0,1) بدهیم یعنی Reset , R=1,S=0 می‌کند.

در حالت اول وقتی $Q_t = 0$ بوده $Q_{(t-1)} = 0$ شده است این حالت در SR نشان دهنده این است که یا Reset شده یا بدون تغییر بوده است.

$$(0,0)\text{ خروجی}\left\{\begin{array}{lll} \rightarrow \text{بدون تغییر} & 0 & 0 \\ \rightarrow \text{Reset} & 0 & 1 \end{array}\right\} \Rightarrow SR' + S'\not{R} = S' \Rightarrow S' = 0 \ , \ R = X$$

برای بدون تغییر بودن باید ورودی‌های R,S هر دو صفر باشند و برای Reset شدن باید R=1,S=0 باشد حال مینترم ورودی‌ها را می‌نویسیم یعنی $S'R' + S'R$ این دو در یک بیت متفاوت هستند بیت متفاوت حذف می‌شود و S' می‌ماند یعنی S باید صفر باشد و R حالت بی‌اهمیت است چون فرقی نمی‌کند صفر باشد یا یک.

در حالتی که $Q_t = 0$, $Q_{t+1} =$ است یعنی Set شده پس ورودی SR باید به صورت (1,0) خواهد بود برای حالت $Q_t = 1$, $Q_{t+1} =$ یعنی Reset شده پس ورودی‌ها S,R به صورت (0,1) خواهد بود.

وقتی $Q_t = 1$, $Q_{t+1} =$ است دو حالت داریم یا بدون تغییر مانده یا set شده است.

$$(1,1)\left\{\begin{array}{ll} \rightarrow \text{بدون تغییر} & \\ & 0 \quad 0 \\ & 1 \quad 0 \\ \rightarrow \text{Set} & \end{array}\right\} \Rightarrow \not{S}R' + \not{S}R' = R'$$

وقتی ورودی S,R صفر باشد خروجی مرحله بعدی همان خروجی مرحله قبلی است و هنگامی Set می‌شود که ورودی S , R , (1 , 0) باشد برای دو حالت مینترم می‌نویسیم به صورت $S'R' + SR'$ این دو یک بیت متفاوت دارند که حذف می‌کنیم در نهایت به R' می‌رسیم و ورودی S نیز حالت بی‌اهمیت است.

برای JK فلیپ فلاپ داریم:

وقتی j=k=1 است خروجی متمم خروجی مرحله قبل است.

وقتی j=k=0 است خروجی همان مرحله قبل است.

وقتی k=0,j=1 خروجی Set می‌شود یعنی اگر خروجی مرحله قبل 0 یا 1 باشد، خروجی مرحله بعد 1 است.

وقتی k=1,j=0 خروجی Reset می‌شود یعنی اگر خروجی مرحله قبل 0 یا 1 باشد، خروجی مرحله بعد 0 است.

با توجه به توضیحات بخش SR برای JK-F.F نیز داریم:

$$(0,0)\left\{\begin{array}{ll} & \begin{array}{cc} j & k \end{array} \\ \rightarrow \text{بدون تغییر} & \begin{array}{cc} 0 & 0 \\ 0 & 1 \end{array} \\ & \begin{array}{cc} 0 & x \end{array} \\ \rightarrow \text{Reset} & \end{array}\right\} \Rightarrow j'\not{K}' + j'\not{K} = j'$$

(0,1)
→ Set
$$\begin{array}{cc} j & k \\ \hline 1 & 0 \\ 1 & 1 \\ \hline 1 & x \end{array} \Rightarrow jk' + jk' = j$$
→ متمم

(1,0)
→ Reset
$$\begin{array}{cc} j & k \\ \hline 0 & 1 \\ 1 & 1 \\ \hline x & 1 \end{array} \Rightarrow j'k + jk = k$$
→ متمم

(1,1)
→ بدون تغییر
$$\begin{array}{cc} j & k \\ \hline 0 & 0 \\ 1 & 0 \\ \hline x & 0 \end{array} \Rightarrow j'k' + jk' = k'$$
→ Set

برای D فلیپ فلاپ داریم:

هر چه به ورودی بدهیم همان را در خروجی می‌بینیم.

Q_{t+1} آخرین خروجی ما است پس هر مقداری که دارد همان هم در ورودی باید باشد.

برای T فلیپ فلاپ داریم:

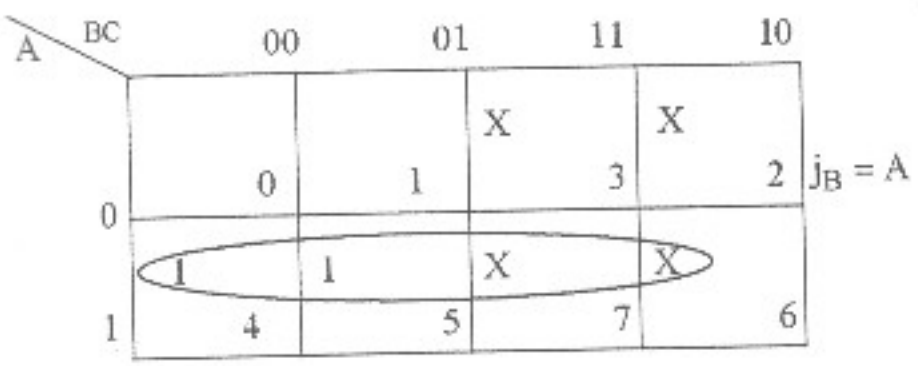
اگر T=0 بود خروجی هیچ تغییری با خروجی مرحله قبل ندارد.

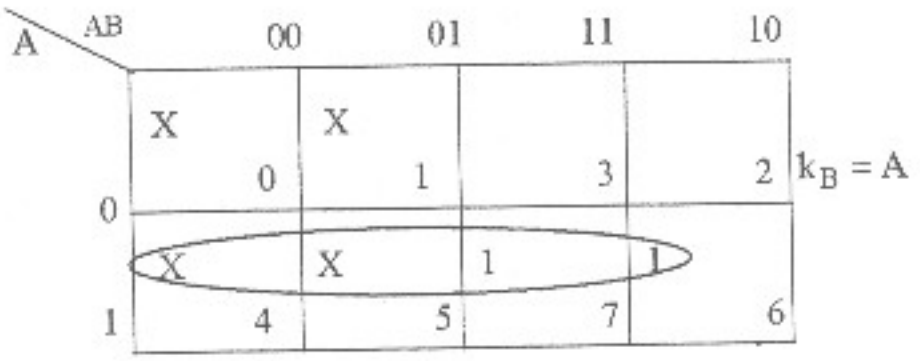
اگر T=1 بود خروجی متمم خروجی مرحله قبل می‌شود.

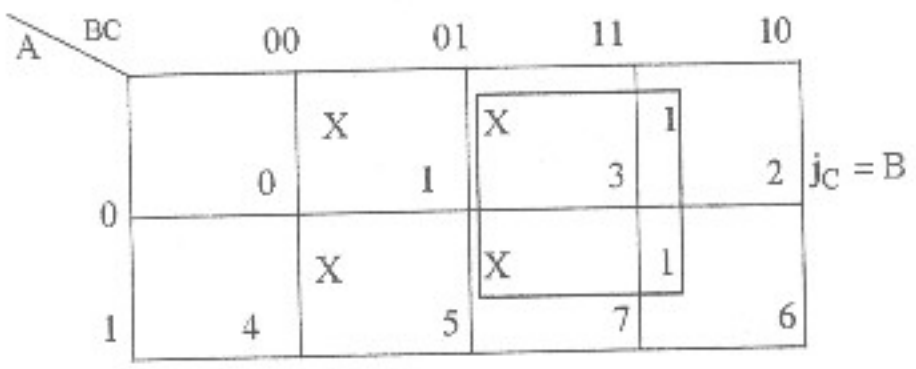
مثال: به کمک فیلیپ فلاپ J.k مداری طراحی کنید که یک عدد دو دویی، سه بیتی را به معادل gray آن تبدیل کند.

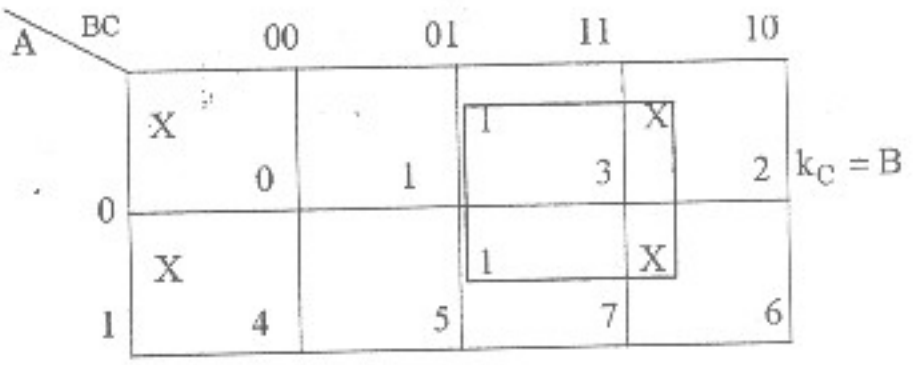
ورودی / حالات فعلی			خروجی / حالات بعدی								
A_t	B	C	A_{t+1}	B	C	j_A	k_A	j_B	k_B	j_C	k_C
0	0	0	0	0	0	0	X	0	X	0	X
0	0	1	0	0	1	0	X	0	X	X	0
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	1	1	X	0	1	X	X	0
1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	0	0	X	0	X	1	X	1
						0	0				

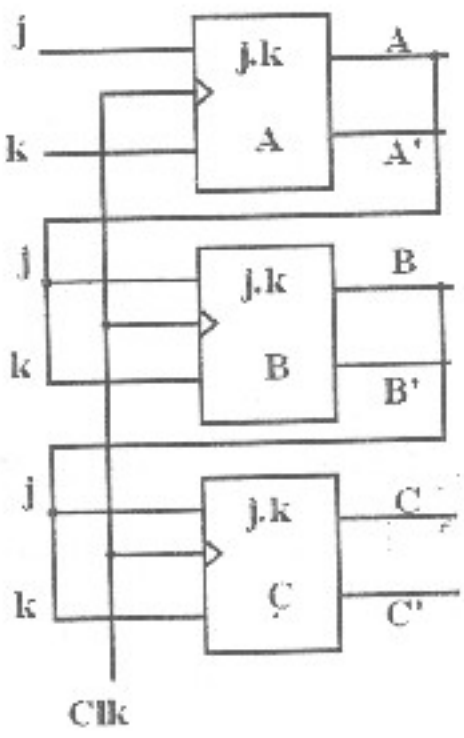
جدول کارنو











کاهش حالات

در طراحی‌ها از فلیپ فلاپ‌ها استفاده می‌کنیم که نسبت به تجهیزات مدارهای ترکیبی گران‌تر هستند.برای این‌که مدار از لحاظ هزینه بهینه باشد باید حالات معادل را پیدا کنیم تا تعداد فلیپ فلاپ‌ها کاهش یابد

حالات معادل

به حالاتی گفته می‌شود که ورودی خروجی و حالت بعدی در هر دو State یکی باشد.

(State: به هر کدام از دایره‌ها در دیاگرام حالات و یا حالت‌های متفاوت موجود در مدار گفته می‌شود)

در حالات معادل اگر ورودی و خروجی و مقصد یکی باشد (خروجی / ورودی با هم برابر) دو حالت با هم معادل هستند چون هر دو با ورودی و خروجی مشابه به State رفتند.

مراحل کاهش حالات:

۱ـ تشکیل جدول حالات (به کمک توضیحاتی که در قبل آمده)

۲ـ حذف حالات معادل:

بعد از پیدا کردن حالات معادل یکی از آن‌ها را حذف می‌کنیم و در هر کجای جدول حالات که از حالت حذف شده استفاده شده بود حالت معادل آن را قرار می‌دهیم.

۳ـ تشکیل جدول حالات جدید برای طراحی ورودی‌های جدید

کاهش حالات با استفاده از جدول:

نحوه پر کردن جدول به این صورت است که در خانه تقاطع 2 حالت یا x یا ✓یا یک شرط می‌گذاریم

x: به این معنی که دو حالت با هم معادل نیست

b	×			
e	✓			
d	a,b		×	
e	a,c		✓	
	a	b	c	d

✓ :به این معنی که دو حالت با هم معادل است

حالات معادل حالاتی هستند که ورودی و خروجی و حالت بعدی آن‌ها یکسان نامند

x زمانی است که خروجی دو حالت با یکدیگر مساوی نبود. اگر هم خروجی یکی بود و هم حالت بعدی ✓ می‌گذاریم

اگر خروجی یکی باشد ولی حالت‌های بعدی متفاوت باشد حالت فعلی یک شرط است وقتی خروجی‌ها یکی بود و حالت بعدی متفاوت a,d در صورتی معادلند که a,b (شرط) با هم معادل باشد در جدول به محل تقاطع ab توجه می‌کنیم چون در محل تقاطع x است پس a,b معادل نیستند در نتیجه a,b هم معادل نیستند و در جدول به جای شرط x می‌گذاریم

A,c هم در صورتی معادلند که a,c معادل باشد در جدول تقاطع خانه‌های c,a علامت ✓ وجود دارد پس c,a معادلند در نتیجه a,e هم معادلند و به جای شرط ✓ قرار می‌دهیم.

نام‌گذاری حالات

می‌توان به هر یک از حالات، نامی اختصاص داد که از حروف تشکیل شده باشد. به عنوان مثال می‌توان $(a, b, c, \ldots, z)$ را به عنوان یک حالت در نظر گرفت.

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

حالت‌های d,a دارای ورودی حالت بعدی خروجی یکسان هستند پس $a \equiv d$ می‌باشد.

بنابراین در جدول هر جا a بود می‌توانیم d قرار می‌دهیم. اگر در جدول به جای a,d را قرار دهیم حالت‌های e,c نیز با هم معادل هستند یعنی ورودی حالت بعدی و خروجی آن‌ها یکی خواهد بود پس $c \equiv e$ خواهد شد به این ترتیب می‌توانیم 5 فلیپ فلاپ را به 3 فلیپ فلاپ کاهش دادیم.

کاهش حالات به کمک جدول:

می‌خواهیم جدولی طراحی کنیم که در آن تمام حالات یک مسئله با هم مقایسه شوند اگر تعداد حالات را n فرض کنیم می‌خواهیم یک جدول $(n-1) \times (n-1)$ بدست آوریم که در آن تقاطع برای هر دو حالت تنها یک بار وجود دارد (برای دو حالت d,a تنها یک خانه تقاطع وجود دارد که هم برای d,a و هم برای a,d می‌باشد.)

تخصیص حالات

به هر حالت یک کد اختصاص می‌دهیم در این‌جا چون 5 حالت داریم به هر کدام یک کد سه بیتی اختصاص می‌دهیم. لزومی ندارد این کدها به ترتیب باشند برای n حالت متفاوت باید از m بیت استفاده کرد.

$$m = \lceil \mathrm{Log}_2^n \rceil$$

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

a = 000
b = 001
c = 100
d = 111
e = 100

حالت فعلی	ورودی	حالت بعدی	خروجی
A B C	x	A B C	y
0 0 0	0	0 0 1	1
0 0 1	0	1 0 0	0
1 0 0	0	1 1 1	1
1 1 1	0	0 0 1	1
1 0 0	0	0 0 0	1

کاهش حالت با استفاده از جدول:

حالت فعلی	ورودی x	حالت بعدی	خروج
a	0	b	1
b	0	e	0
c	0	d	1
d	0	b	1
e	0	a	1

* 5 تا حالت داریم پس جدول ما 4×4 $(n-1) \times (n-1)$ است

	b	X			
	c	b , d	X		
(۱)	d	✓	X	d , b	
(۲)	e	a , b	X	d , a	b , a
		a	b	c	d

* (۱) خروجی‌های a و b با هم برابرند پس به سراغ حالت بعدی آن‌ها می رویم که هر دو b است پس معادلند.

* (۲) در صورتی a و b با هم برابرند یعنی معادلند که a و b با هم معادل باشند.

b	X			
c	X	X		
d	✓	X	X	
e	X	X	✓	X
	a	b	c	d

$\begin{cases} c \equiv e \\ a \equiv d \end{cases}$

حالت فعلی	حالت بعدی	خروجی
a	b	1
b	c	0
c	a	1

خلاصه برای پیاده‌سازی مدارهای ترتیبی:

۲ـ با تشریح مسئله به یک جدول حالت می‌رسیم.

۲ـ حالت را کاهش می‌دهیم.

۳ـ به هر حالت یک عدد باینری تخصیص می‌دهیم و جدول حالت را دوباره می‌سازیم.(تخصیص حالت)

۴ـ جدول تحریک فلیپ فلاپ موردنیاز را می‌نویسیم.

۵ـ معادلات را از روی جدول تحریک بدست می‌آوریم

۶ـ دیاگرام منطقی را رسم می‌کنیم.

مثال: به کمک FF . jK مدار مربوط به دیاگرام حالات زیر را طراحی کنید.

حالت فعلی	x	حالت بعدی	y
A	0	B	1
A	1	E	0
B	0	E	1
B	1	D	0
C	0	B	1
C	1	E	0
D	0	B	0
D	1	C	0
E	0	E	0
E	1	D	0

$A \equiv B \xrightarrow{\text{اگر}} \begin{cases} B \not\equiv E \\ D \equiv E \end{cases} \rightarrow$ معادل نیست

$A \equiv C$ ✓ — x $A \not\equiv D$ ×

x $A \not\equiv E$ × — x $C \not\equiv D$ ×

x $C \not\equiv E$ × — x $B \not\equiv D$ ×

$B \equiv C \xrightarrow{\text{اگر}} \begin{cases} B \not\equiv E \\ D \equiv E \end{cases}$ ×

$D \equiv E \xrightarrow{\text{اگر}} \begin{cases} B \not\equiv E \\ C \equiv D \end{cases}$ ×

فقط A و C با هم معادلند پس به جای C می‌توان A را گذاشت و ستون مربوط به C را حذف کرد.

$A \rightarrow 00$

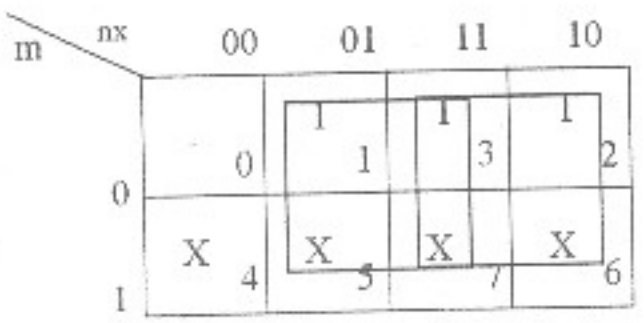
$B \rightarrow 01$

$D \rightarrow 10$

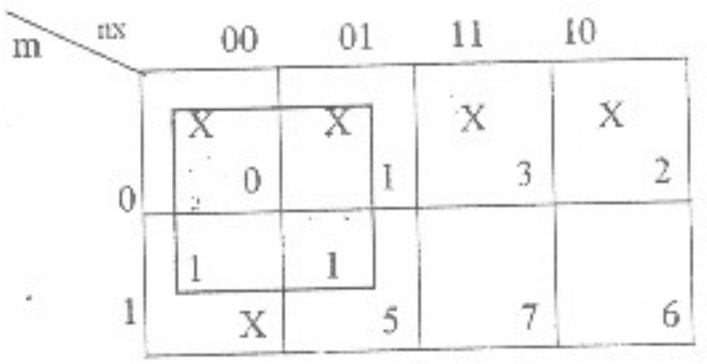
$E \rightarrow 11$

حالت فعلی — حالت بعدی

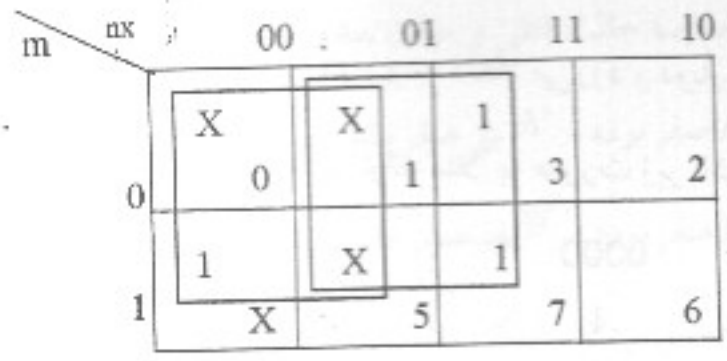
	m	n	x	m	n	y	j_m	k_m	j_n	k_n
A	0	0	0	0	1	1	0	X	1	X
	0	0	1	1	1	0	1	X	1	X
B	0	1	0	1	1	1	1	X	X	0
	0	1	1	1	0	0	1	X	X	1
D	1	0	0	0	1	0	X	1	1	X
	1	0	1	0	0	0	X	1	0	X
E	1	1	0	1	1	0	X	0	X	0
	1	1	1	1	0	0	X	0	X	1



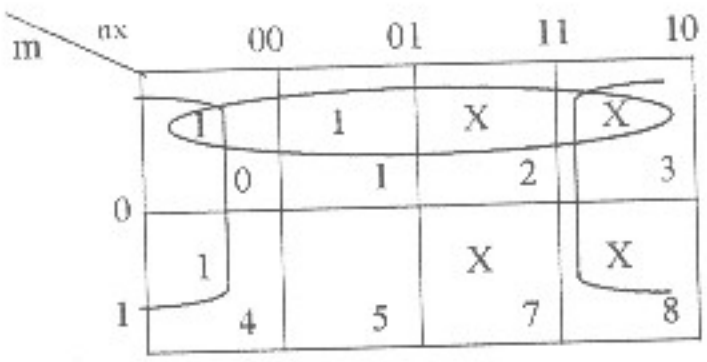
$j_m = n + x$



$k_m = n'$



$k_n = n' + x$



$j_n = m' + x'$

طراحی شمارنده

دو نوع شمارنده داریم:

۱)با ترتیب

۲) بدون ترتیب

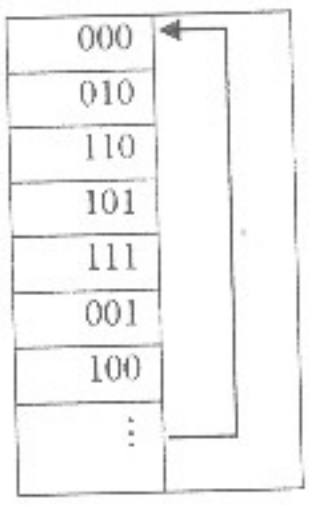
شماره با ترتیب: یعنی این‌که اگر صفر را شمرد،بعد یک را بشمارد، بعد ۲ و همین‌طور ادامه پیدا کندتا به یک x برسد که بعد از x دور بزند و صفر را بشمارد یا اگر از 5 شروع کرد بعد 6 و بعد 7 را بشمارد تا عدد x، سپس دور بزند و باز از 5 شروع به شمارش بکند یعنی لزومی ندارد از صفر شروع به شمارش کند ولی از هر کجا که شروع به شمارش کرد به ترتیب جلو برود.

شماره بدون ترتیب: می‌تواند از هر عددی شروع شود ولی اعداد را به ترتیب نمی‌شمارد وقتی تا x را بشمارد دور می‌زند و از عدد اول شروع به شمارش می‌کند.

● شمارنده‌ها یا دو دویی هستند یا غیر دو دویی.

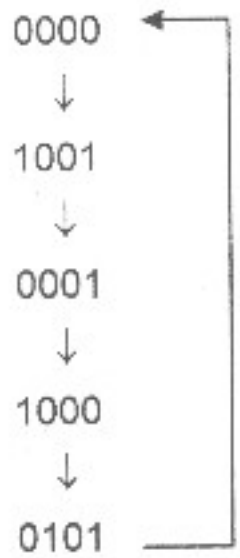
شمارنده دودویی با ترتیب: یعنی اگر سه بیتی است از 0 0 0 شروع کند به شمارش تا به 1 1 1 برسد.

شمارنده دودویی بدون ترتیب: از 0 0 0 شروع کند بعد 0 1 0 بعد 1 1 0 , 1 0 1 , 1 1 1 را بشمارد دوباره بازگردد به اول.



برای مجموعه $\{0,1,2,3,...,8,9\}$ شمارنده بدون ترتیب این مجموعه به صورت زیر است:

شمارنده غیر دودویی: شمارنده کدهای BCD یک شمارنده غیر دودویی است که از (0000) شروع می‌شود و به 1001 می‌رود و دوباره به اول باز می‌گردد این شمارنده به 1111 نمی‌رسد پس دودویی نیست حال اگر این شمارنده بدون ترتیب باشد می‌تواند مثلاً به صورت زیر باشد.



مشاهده می‌شود که در یک شمارنده دودویی n بیتی تعداد اعداد قابل شمارش 2^n می‌باشد یعنی به عنوان مثال از صفر تا $2^n - 1$ اما در یک شمارنده غیر دودویی با n بیت هیچ دلیلی برای شمارش 2^n عدد وجود ندارد. به عنوان مثال شمارنده اعداد BCD از 4 بیت استفاده می‌کند اما 2^4 یعنی 16 عدد را نمی‌شمارد بلکه تنها 10 عدد می‌شمارد.

در یک شمارنده دودویی از n فلیپ فلاپ استفاده می‌شود تا 2^n عدد را بشمارد اما در یک شمارنده غیر دودویی n بیتی از n فلیپ فلاپ استفاده می‌شود اما ممکن است 2^n عدد را نشمارد یعنی از فلیپ فلاپ‌ها استفاده بهینه نمی‌شود.

مثال:

یک شمارنده دودویی (با ترتیب) سه بیتی طراحی کنید که از صفر تا 7 را بشمارد (با استفاده از T فلیپ فلاپ)

در یک شمارنده وقتی از فلیپ فلاپ استفاده می‌کنیم باید حالت فعلی و حالت بعدی را به وجود بیاوریم حالت فعلی در شمارنده، همان ترکیبات متغیرها می باشد یعنی برای n متغیر 2^n حالت فعلی به وجود می‌آید. حالت بعدی عددی است که به وجود می‌آید

حالت فعلی			حالت بعدی			ورودی فلیپ فلاپ		
A	B	C	A′	B′	C′	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

شمارنده ترتیبی است پس از 000 شروع می‌کند به شمارش می‌کند تا به 7 برسد بعد برگردد به صفر برمی‌گردد، چون با استفاده از T فلیپ فلاپ می‌خواهیم طراحی کنیم، باید ورودی‌های T.FF را به دست آوریم. اگر خروجی مرحله بعدی متمم خروجی مرحله قبل بود T=1 است و اگر خروجی مرحله بعد با مرحله قبل یکی بود T=0 است.

در مقایسه حالت فعلی و حالت بعدی داریم:

A صفر بوده ، A' نیز صفر مانده پس T_A نیز صفر است.
B صفر بوده ، B' نیز صفر مانده پس T_B نیز صفر است.
C صفر بوده، C' یک شده پس T_C یک است.
} وقتی حالت فعلی (000) است و حالت بعدی (001)

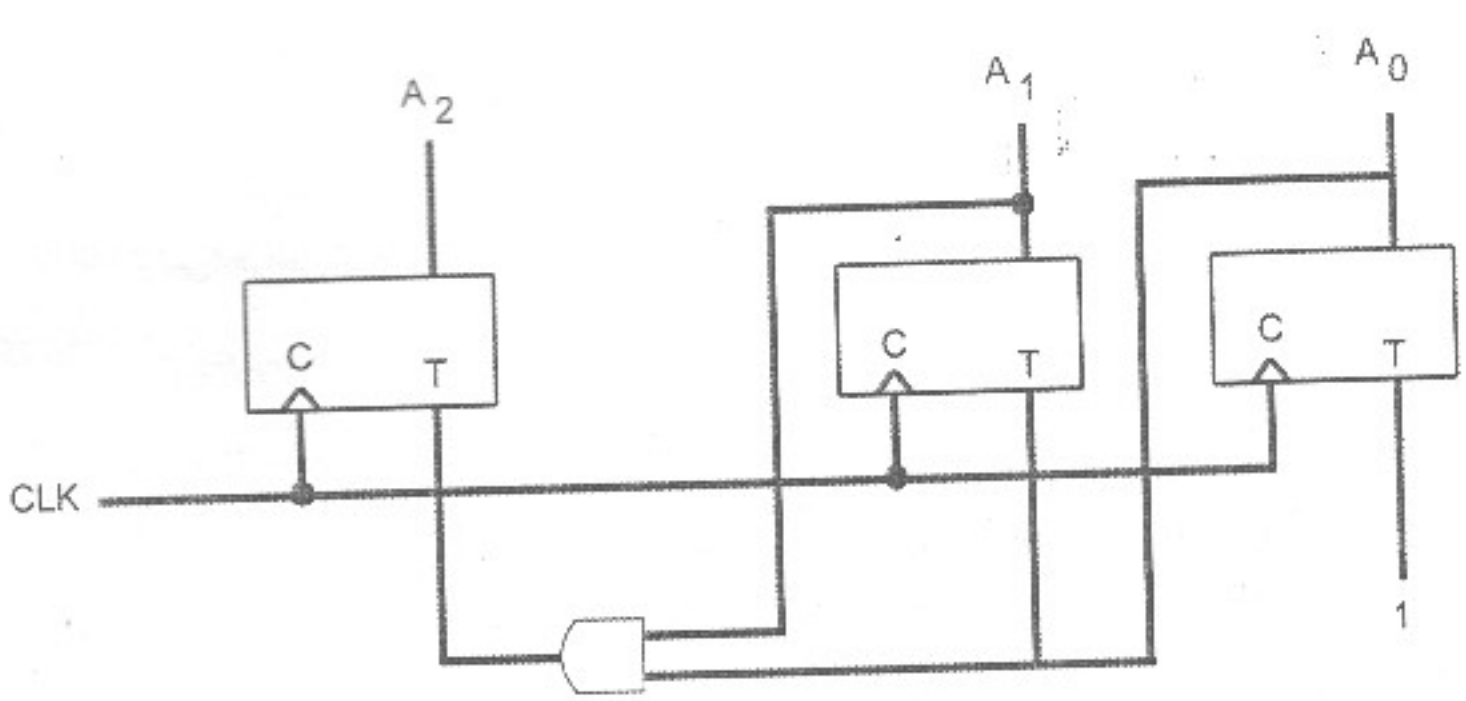
A صفر بوده ، A' نیز صفر مانده پس T_A صفر است.
B صفر بوده، B' یک شده پس T_B یک است.
C یک بوده، C' صفر شده پس T_C یک است.
} وقتی حالت فعلی (001) بوده و حالت بعدی (010)

بدین ترتیب بقیه جدول را نیز کامل می کنیم بعد مینترم‌ها را می‌نویسیم.

$T_a = bc$

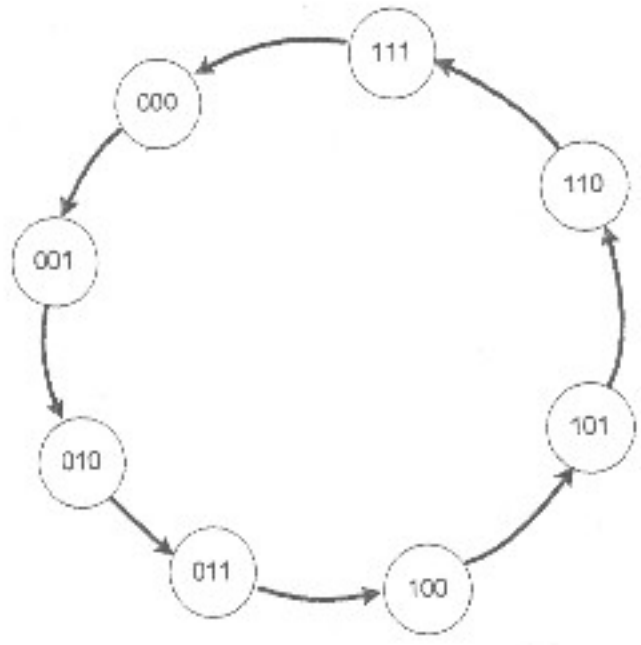
$T_b = c$

$T_c = 1$ چون هر چه اعمال کرده‌ایم، در T_c تاثیری نداشته و همان مقدار یک مانده است بنابراین:

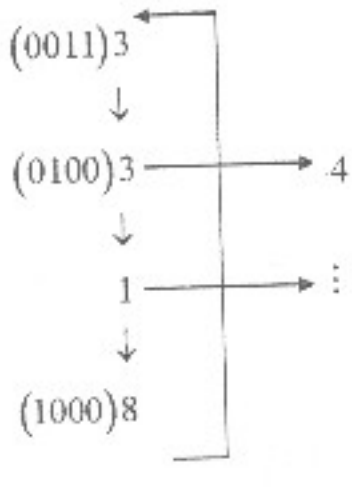


نمودار منطقی شمارنده دو دویی 3 بیت

این دیاگرام با دیاگرام حالت‌های قبلی متفاوت است قبلاً برای هر State یک ورودی داشتیم اما این‌جا ورودی خط clk است. اگر خط clk نباشد ورودی تغییر نمی‌کند. یعنی ورودی به صورت دستی نداریم.



در شمارنده غیر دودویی می‌توانیم یک مقدار مشخص را به وجود بیاوریم که از آن‌جا شروع به شمارش بکنیم و یک مقدار مشخص (دلخواه) هم به وجود بیاوریم که به آن ختم شود مثلاً در BCD از 3 (0011) شروع به شمارش کنیم تا 8 (1000) بشمارد بعد به اول بازگردد.

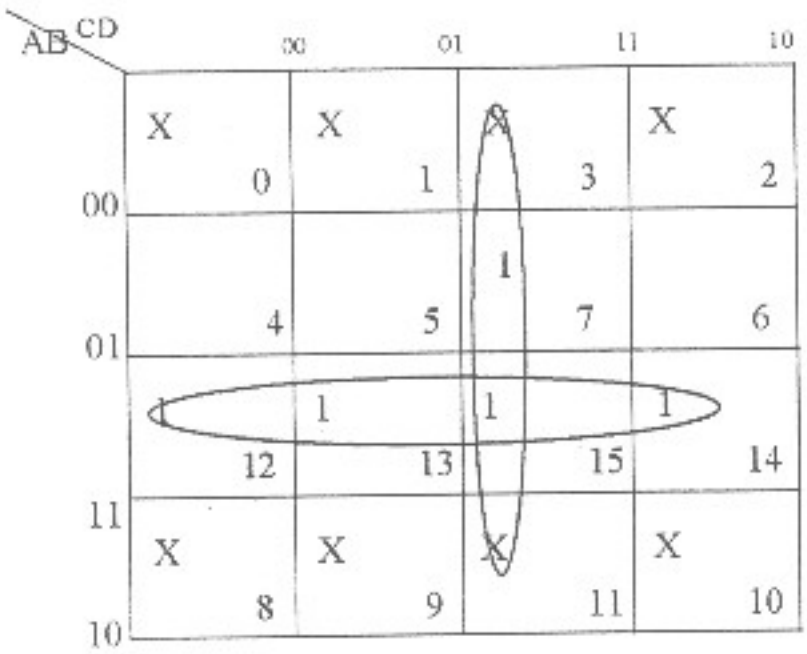


- یک شمارنده BCD (ترتیبی) طراحی کنید (با استفاده از jk فلیپ‌فلاپ)

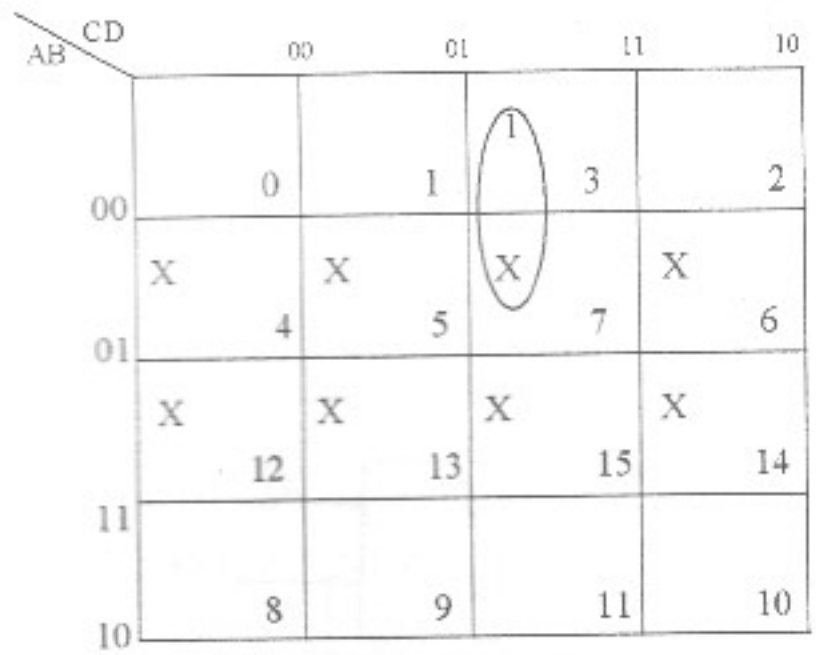
اگر مدار به یک حالت ناخواسته و نامعین رسید حالت بعدی را صفر در نظر بگیرید.

حالات فعلی				حالات بعدی				ورودی‌های فلیپ‌فلاپ							
A	B	C	D	A	B	C	D	J_A	k_A	J_B	k_B	J_C	K_C	J_D	k_D
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	1	X	0	X	0	X	1	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1
1	0	1	0	0	0	0	0	X	1	0	X	X	1	0	X
1	0	1	1	0	0	0	0	X	1	0	X	X	1	X	1
1	1	0	0	0	0	0	0	X	1	X	1	0	X	0	X
1	1	0	1	0	0	0	0	X	1	X	1	0	1	X	1
1	1	1	0	0	0	0	0	X	1	X	1	X	1	0	1
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1

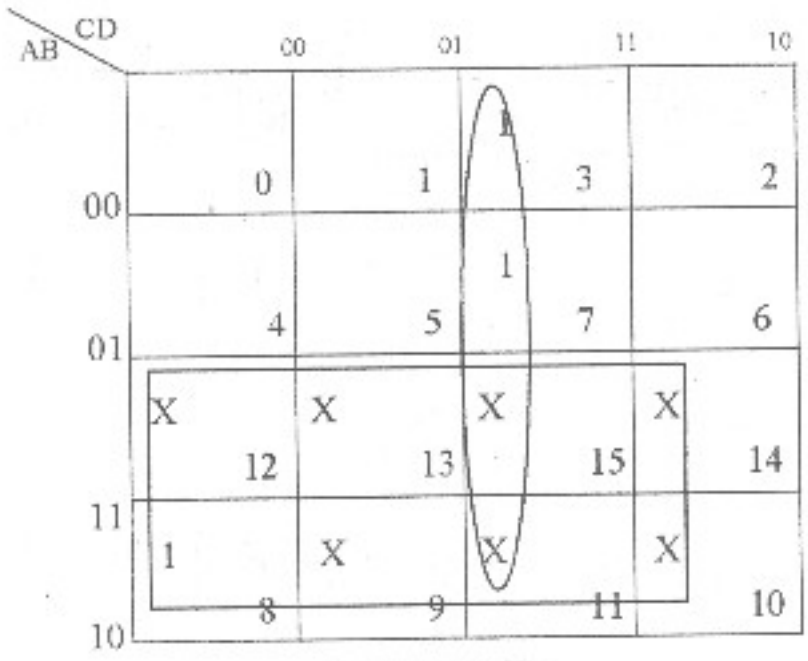
جدول کارنو



$k_B = CD + AB$



$J_B = A'CD$



$J_A = A + BCD$

$K_A = 1$

AB \ CD	00	01	11	10
00	1 (0)	X (1)	X (3)	1 (2)
01	1 (4)	X (5)	X (7)	1 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	X (9)	X (11)	X (10)

$$J_D = A' + B'C'$$

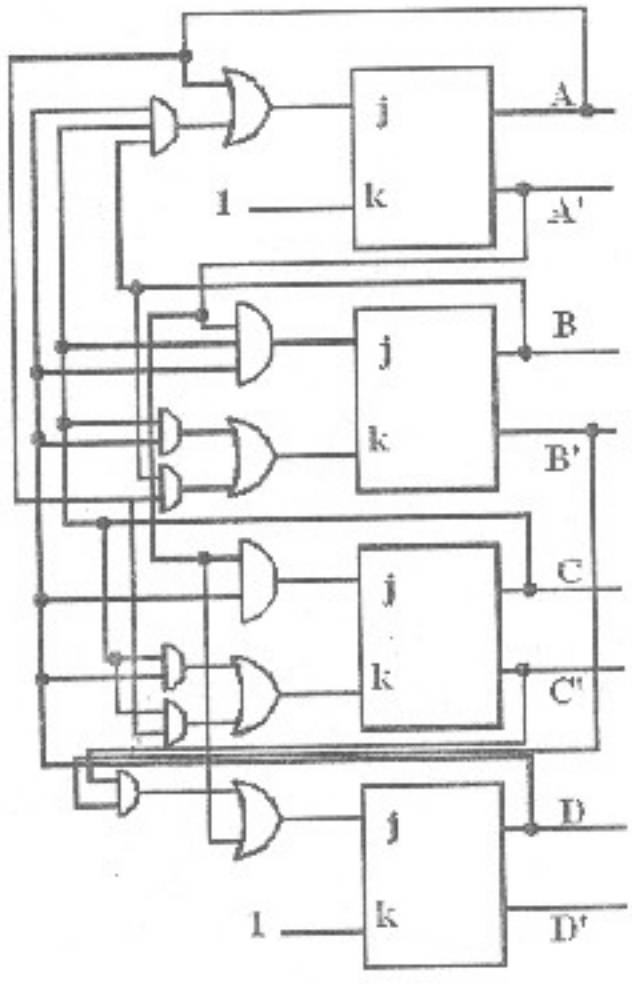
AB \ CD	00	01	11	10
00	X (0)	X (1)	1 (3)	(2)
01	X (4)	X (5)	1 (7)	(6)
11	X (12)	X (13)	1 (15)	1 (14)
10	X (8)	X (9)	1 (11)	1 (10)

$$K_C = CD + AC$$

AB \ CD	00	01	11	10
00	1 (0)	X (1)	(3)	X (2)
01	1 (4)	X (5)	(7)	X (6)
11	(12)	(13)	X (15)	X (14)
10	(8)	(9)	X (11)	X (10)

$$J_C = A'D$$

$$K_D = 1$$



مثال:

یک شمارنده دودویی (سه بیت) با ترتیب با استفاده از RS فلیپ فلاپ طراحی کنید؟

A	B	C	A	B	C	S_A	R_A	S_B	R_B	S_C	R_C
	0	0	0	0	1	0	x	0	x	1	0
0	0	1	0	1	0	0	x	1	0	0	1
0	1	0	0	1	1	0	x	x	1	1	0
0	1	1	1	0	0	1	0	0	1	0	1
1	0	0	1	0	1	x	1	0	x	1	0
1	0	1	1	1	0	x	1	1	0	0	1
1	1	0	1	1	1	x	1	x	1	1	0
1	1	1	0	0	0	0	1	0	1	0	1

$S_A = A'BC$

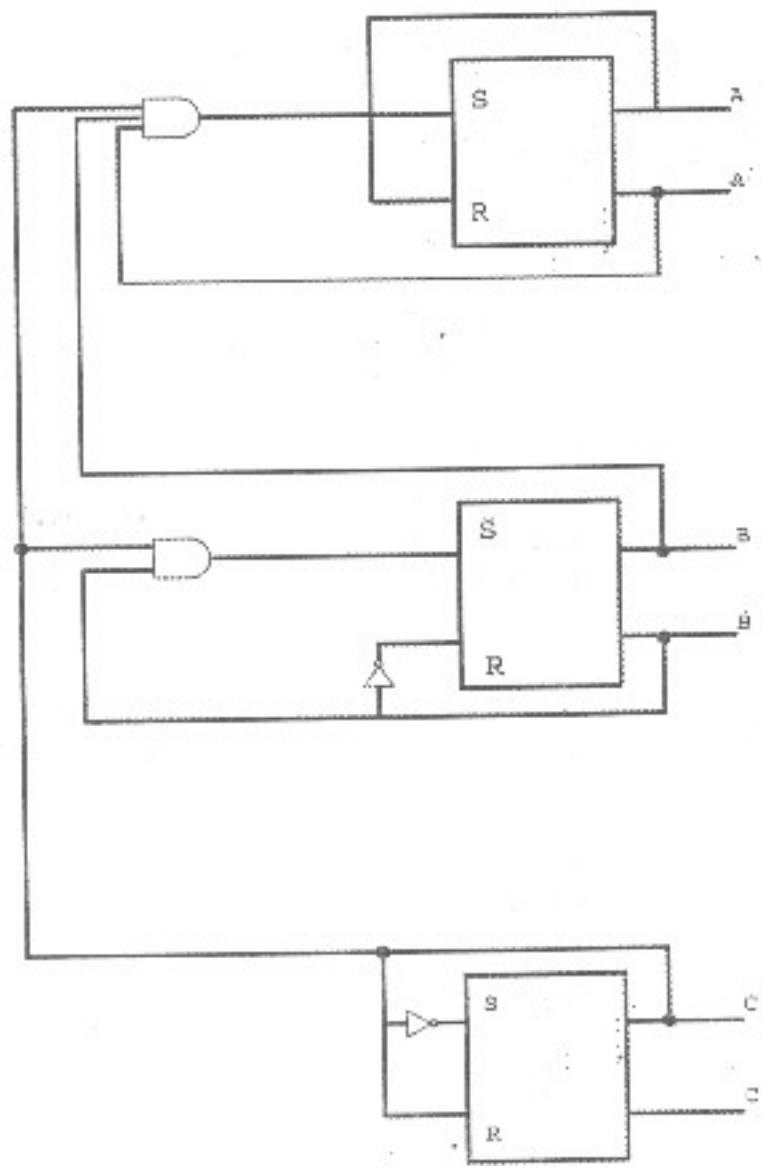
$R_A = A$

$S_B = B'C$

$R_B = B$

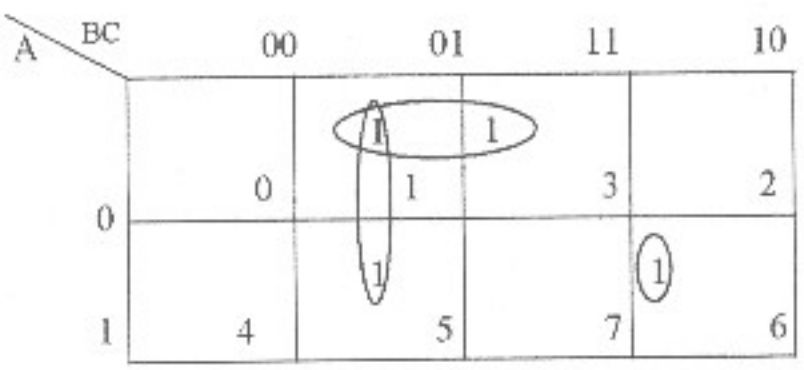
$S_C = C'$

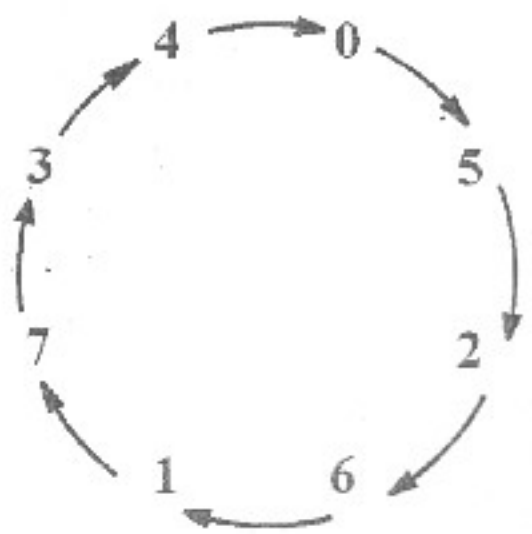
$R_C = C$



شمارنده غیر ترتیبی سه بیتی

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

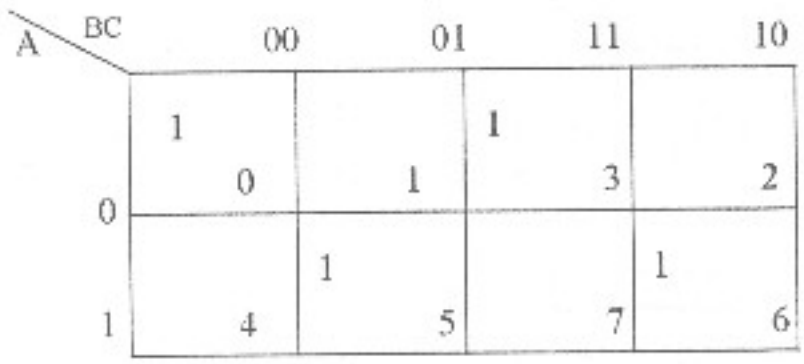




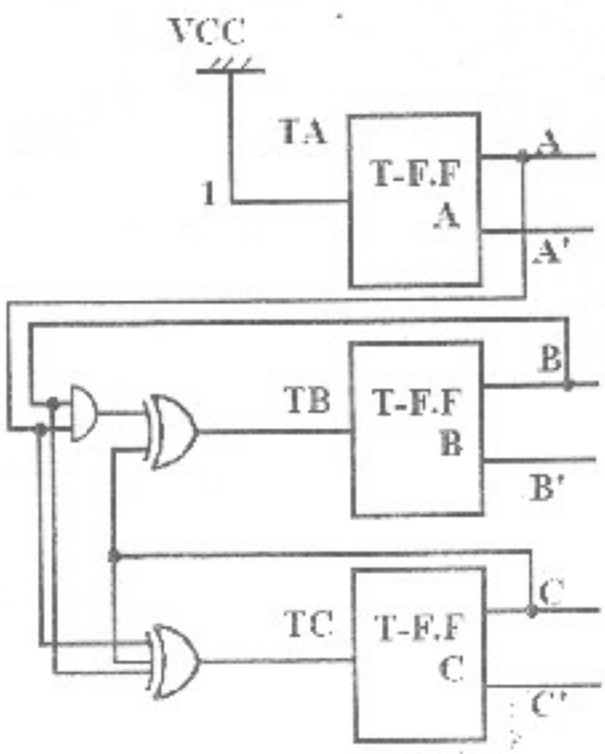
$T_A = 1$

$T_B = \underline{A'C} + \underline{B'C} + ABC'$

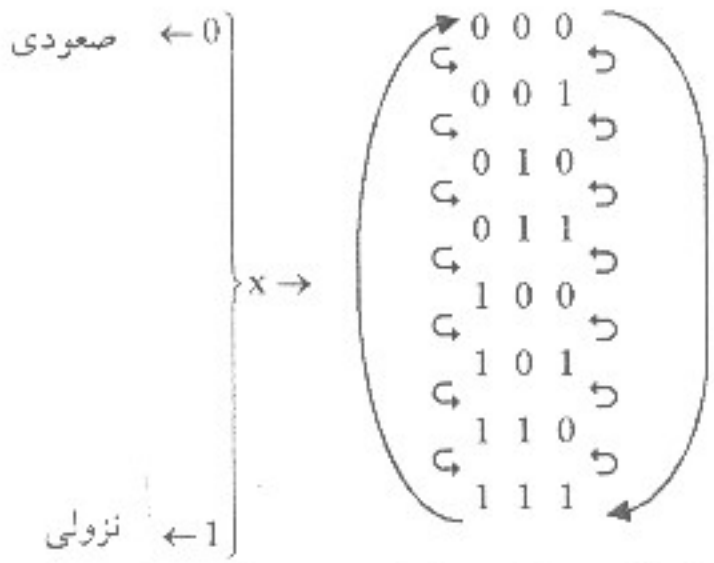
$C(A'+B') + C'(AB) = C\left(\overline{AB}\right) + C'(AB) = C \oplus (AB)$



$T_C = A \odot B \odot C$



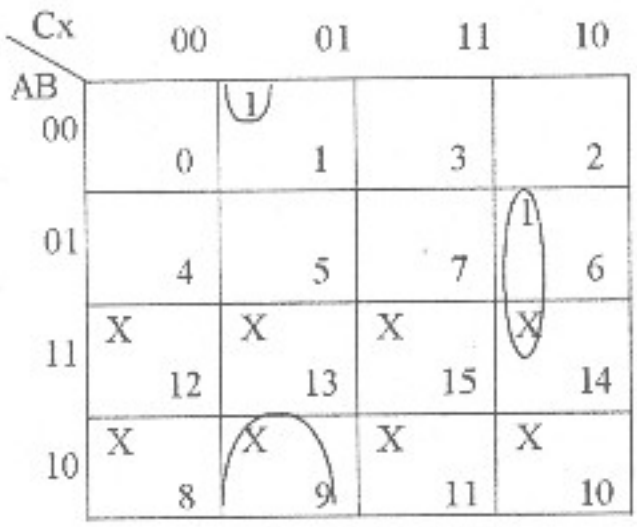
مثال: به کمک j.K، RS، D و T فلیپ فلاپ مداری طراحی کنید که عملیات شمارش صفر تا 7 را به صورت صعودی و 7 تا صفر را به صورت نزولی انجام دهد.



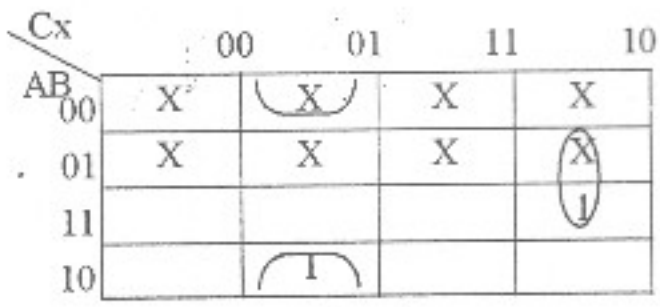
تعداد FF = $\left\lceil \log_2^8 \right\rceil = 3$

A	B	C	x	A	B	C	j_A	k_A	j_B	k_B	j_C	k_C	S_A	R_A	S_B	R_B	S_C	R_c
0	0	0	0	0	0	1	0	X	0	X	1	X	0	X	0	X	1	X
0	0	0	1	1	1	1	1	X	1	X	1	X	1	0	1	X	1	X
0	0	1	0	0	1	0	0	X	1	X	X	1	0	X	1	X	X	1
0	0	1	1	0	0	0	0	X	0	X	X	1	0	X	0	X	X	1
0	1	0	0	0	1	1	0	X	X	0	1	X	0	X	X	0	1	X
0	1	0	1	0	0	1	0	X	X	1	1	X	0	X	X	1	1	X
0	1	1	0	1	0	0	1	X	X	0	X	1	1	0	X	1	X	1
0	1	1	1	0	1	0	0	X	X	0	X	1	0	X	X	0	X	1
1	0	0	0	1	0	1	X	0	0	X	1	X	X	0	0	X	1	X
1	0	0	1	0	1	1	X	1	1	X	1	X	0	1	1	X	1	X
1	0	1	0	1	1	0	X	0	1	X	X	1	X	0	1	X	X	1
1	0	1	1	1	0	0	X	0	0	X	X	1	X	0	0	X	X	1
1	1	0	0	1	1	1	X	0	X	0	1	X	X	0	X	0	1	X
1	1	0	1	1	0	1	X	0	X	1	1	X	X	0	X	1	1	X
1	1	1	0	0	0	0	X	1	X	1	X	1	0	1	X	1	X	1
1	1	1	1	1	1	0	X	0	X	0	X	1	X	0	X	0	X	1

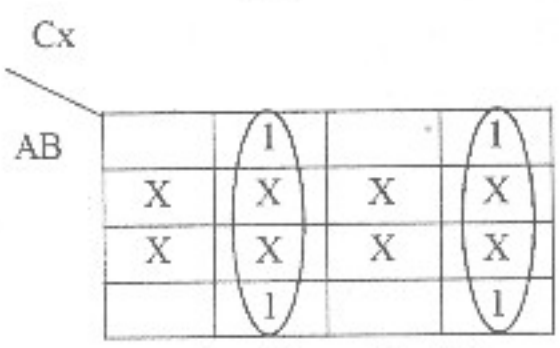
T_A	T_B	T_C	D_A	D_B	D_C
0	0	1	0	0	1
1	1	1	1	1	1
0	1	1	0	1	0
0	0	1	0	0	0
0	0	1	0	1	1
0	1	1	0	0	1
1	1	1	1	0	0
0	0	1	0	1	0
0	0	1	1	0	1
1	1	1	0	1	1
0	1	1	1	1	0
0	0	1	1	0	0
0	0	1	1	1	1
0	1	1	1	0	1
1	1	1	0	0	0
0	0	1	1	1	0



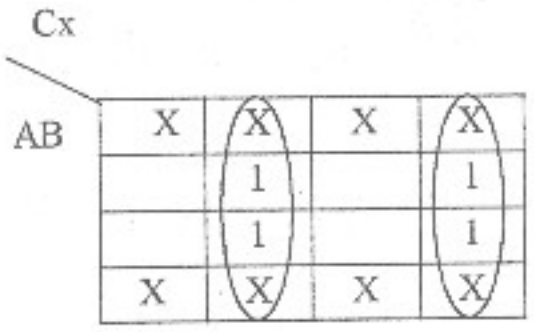
$j_A = Bcx' + B'c'x$



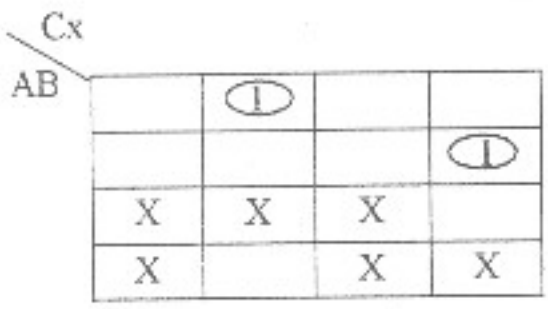
$k_A = BCx' + B'C'x$



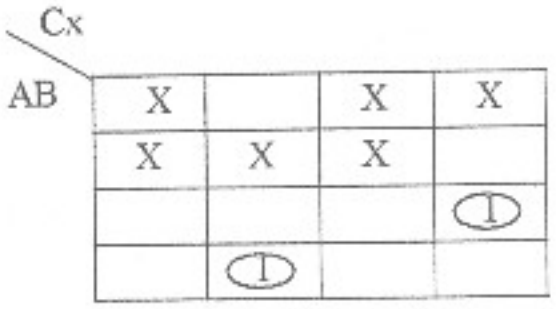
$j_B = c'x + cx' = C \oplus x$



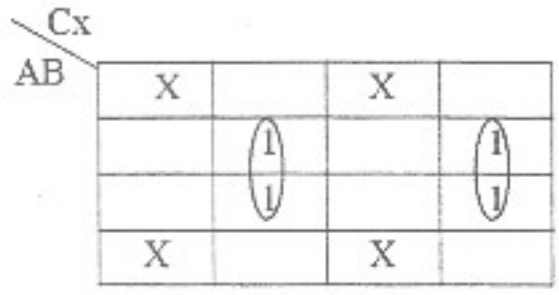
$k_B = c'x + cx' = C \oplus x$



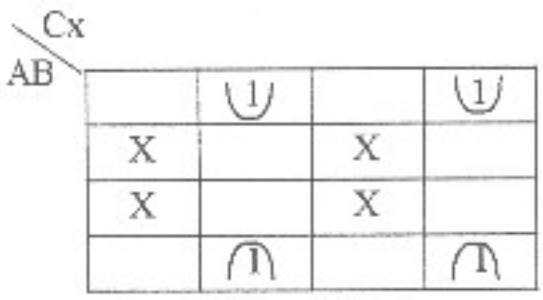
$S_A = A'B'C'x + A'BCx$



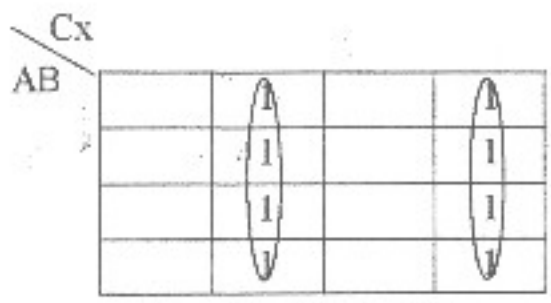
$R_A = AB'C'x + ABCx'$



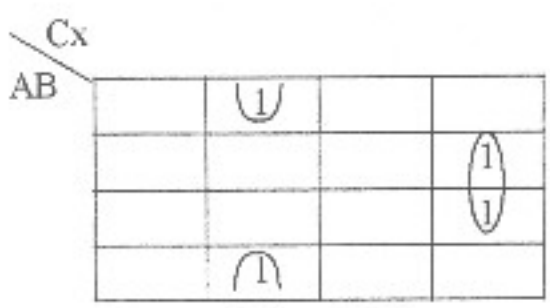
$$R_B = BCx' + BC'x$$



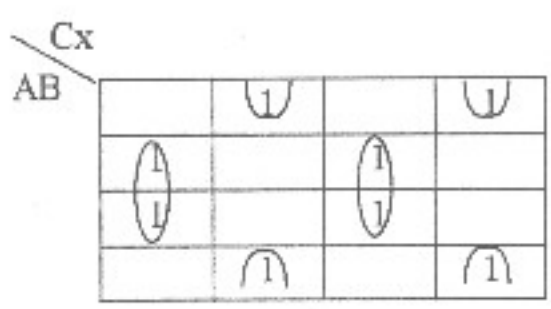
$$S_B = B'C'x + B'Cx'$$

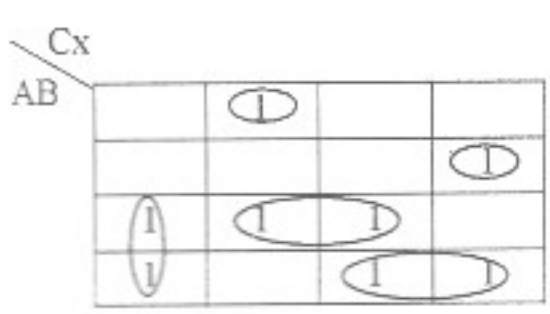


$$T_B = C'x + Cx' = C \oplus x$$



$$T_A = B'C'x + BCx'$$





$$D_A = AC'x' + ABx' + AB'C + A'BCx' + A'B'C'x$$

$$D_B = BC'x' + BCx + B'Cx' + B'C'x = B(C \odot x) + B(C \oplus x) = B \oplus (C \oplus x)$$

$$B(C \odot x) = B\underbrace{\overline{(C \oplus x)}}_{\alpha'} + B\underbrace{(C \oplus x)}_{\alpha} = B\alpha' + B\alpha = B \oplus C \oplus x$$

بدست آوردن فلیپ فلاپ‌ها از روی یکدیگر

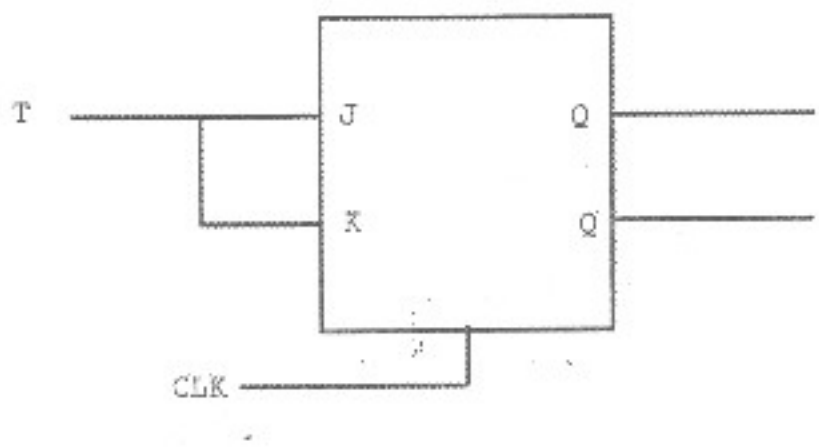
می‌خواهیم T فلیپ فلاپ‌ها را با استفاده از jk فلیپ فلاپ بدست آورد می‌توانیم ورودی‌های j,k فلیپ فلاپ را هم وصل کنیم تا T فلیپ فلاپ ساخته می‌شود.

به این معنی که اگر یک jk فلیپ فلاپ را به صورت تک ورودی در آوریم، توانسته‌ایم یک T فلیپ فلاپ بسازیم.

j	k	Q_{t+1}
0	0	Q_t
0	1	Reset
1	0	Set
1	1	Q'_t

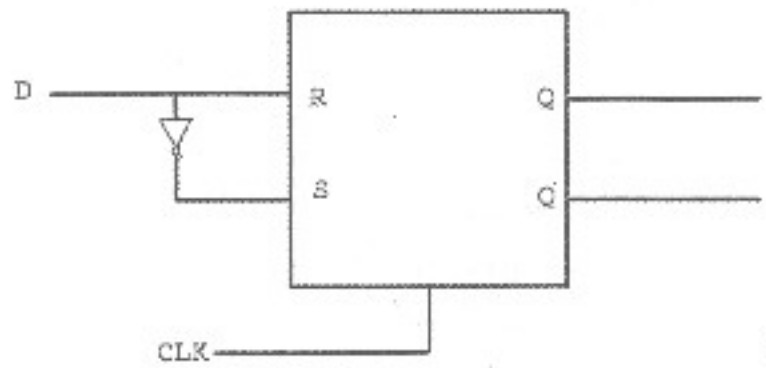
$\Rightarrow$

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



اگر بخواهیم D فلیپ فلاپ را با استفاده از SR فلیپ فلاپ بسازیم باید در ورودی‌های S,R را به هم وصل می کنیم به این صورت که خود S و متمم R را به هم وصل می کنیم. یعنی اینکه کافیست از ورودی S یک خط بگیریم و آن را مکمل کنیم و وارد R کنیم.

S	R	Q_{t+1}
0	0	Q_t
0	1	Reset
1	0	Set
1	1	نامعین



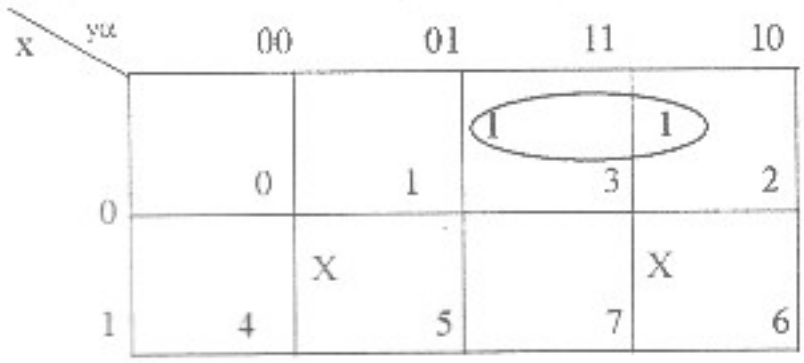
ولی با SR فلیپ فلاپ به راحتی نمی‌توانیم T فلیپ فلاپ بسازیم چون در SR حالت (1,1) نامعین است. و اگر R و S را به هم وصل کنیم و تشکیل یک ورودی بدهیم آن وقت زمانی که به ورودی مقدار یک را اختصاص دهیم حالت نامعین به وجود آورده‌ایم.

اما با (JK) فلیپ فلاپ می‌توان یک D فلیپ فلاپ بسازیم کافیست که یک ورودی را یک بار خودش و یک بار متممش را بسازیم و خودش را به خط J و متممش را به خط K وصل کنیم.

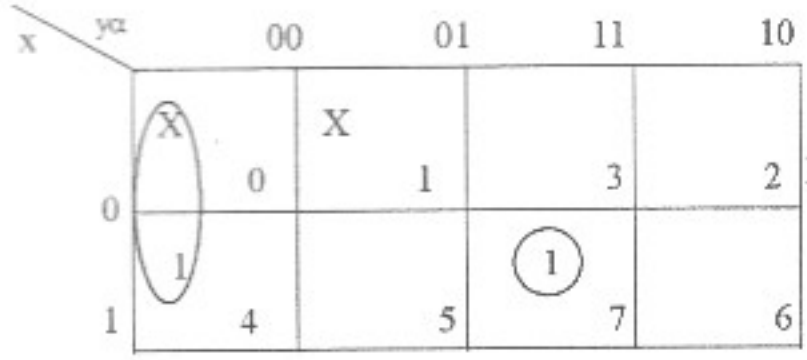
مجموعه مسائل فصل ۴:

۱ـ با توجه به دیاگرام حالات زیر به کمک فلیپ‌فلاپ RS مدار را طراحی کنید.

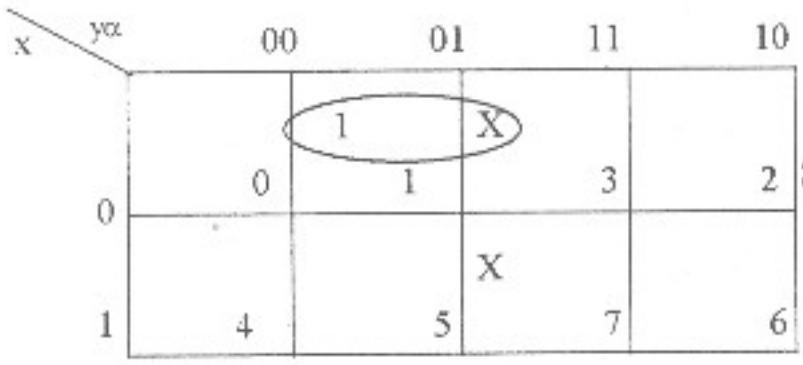
x	y	α	x	y	β	S_x	R_x	S_y	R_y
0	0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	0	X	1	0
0	1	0	1	0	1	1	0	0	1
0	1	1	1	1	1	1	0	X	0
1	0	0	0	0	1	0	1	0	X
1	0	1	1	0	0	X	0	0	X
1	1	0	1	0	0	X	0	0	1
1	1	1	0	1	1	0	1	X	0



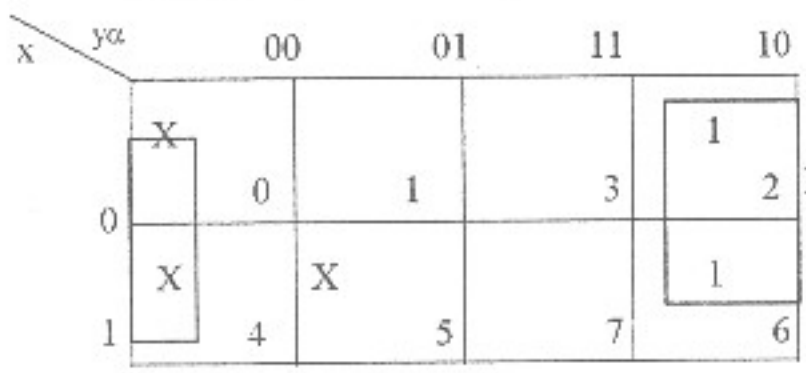
$S_x = x'y$



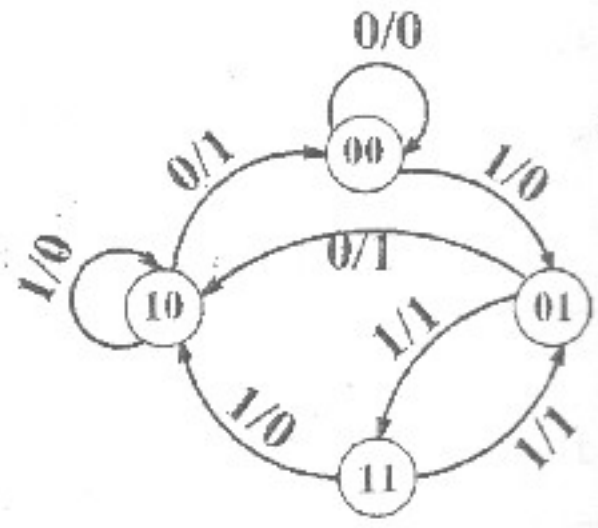
$R_x = y'\alpha' + xy$

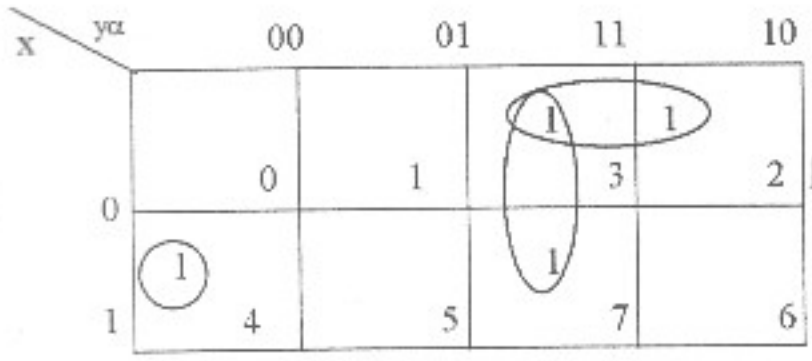


$S_y = x'\alpha$

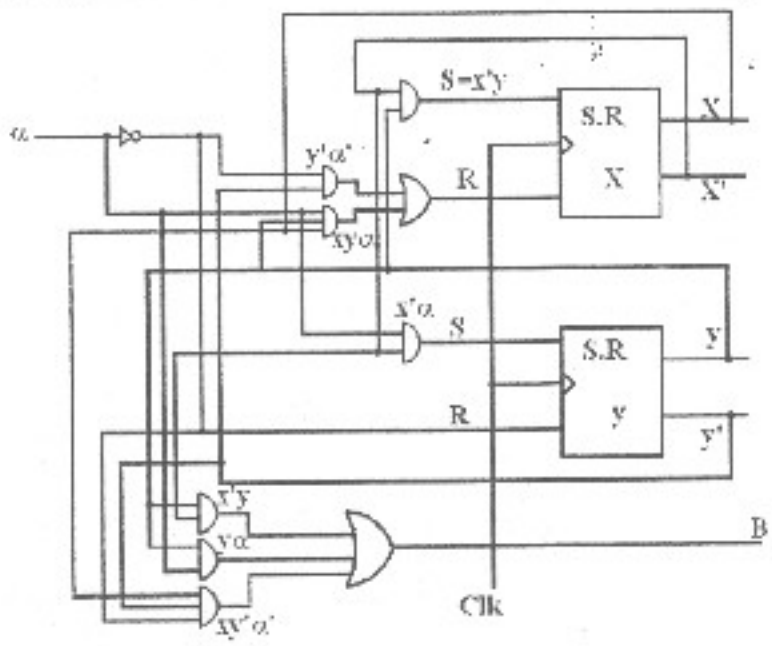


$R_y = \alpha$





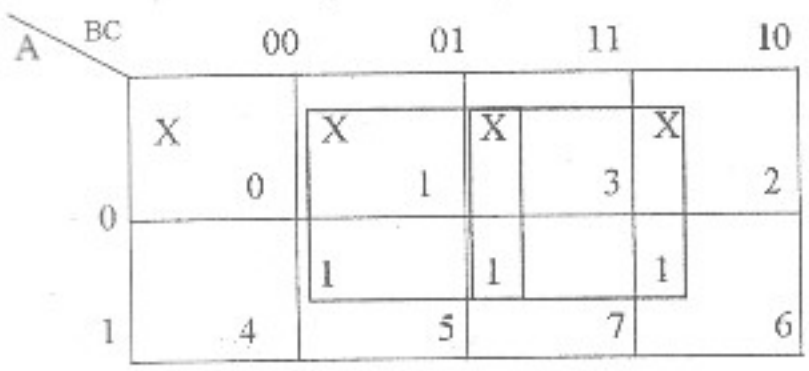
$\beta = x'y + y\alpha + xy'\alpha'$



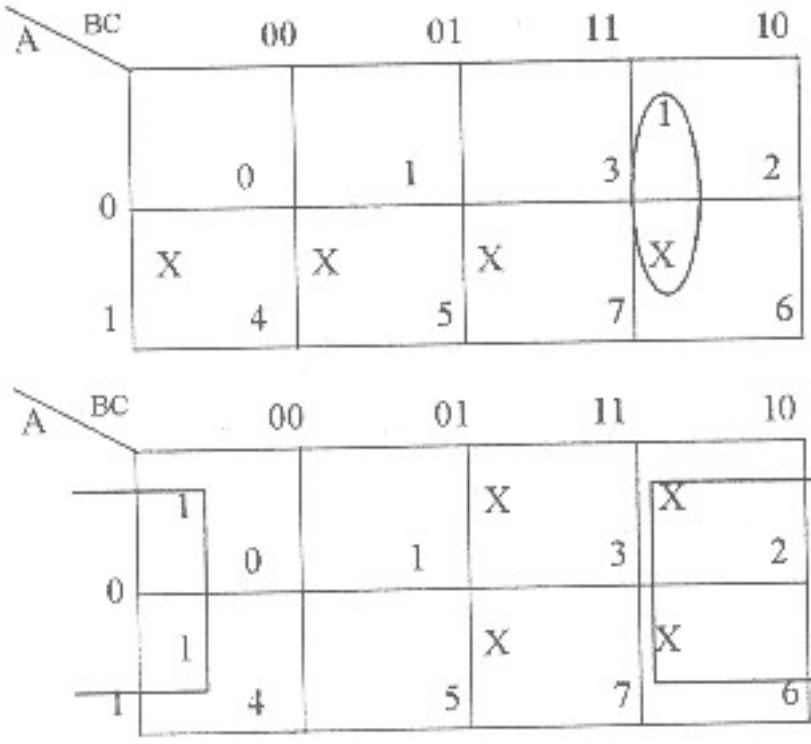
۲- به کمک J.K-F.F شمارنده‌ای که دیاگرام زیر را پوشش می‌دهد را طراحی کنید. (حالات ناخواسته را صفر قرار دهید).

$0 \rightarrow 2 \rightarrow 4 \rightarrow 6$

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

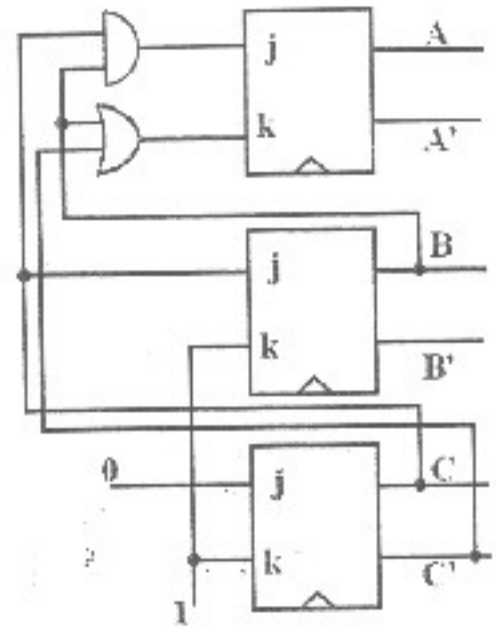


$K_A = C + B$



$J_A = BC'$

$J_B = C'$

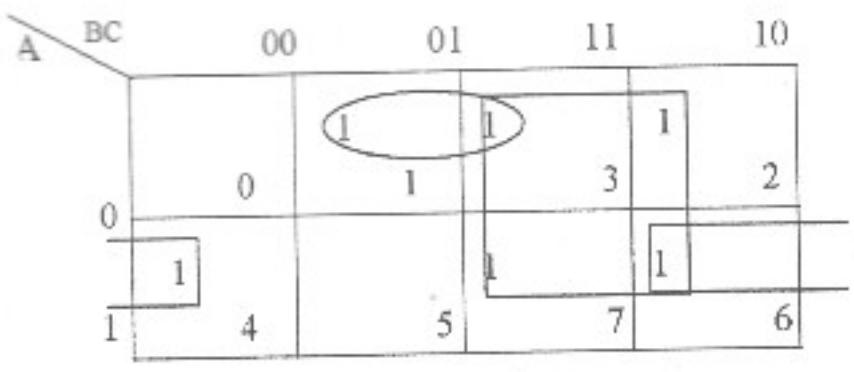


۳- شمارنده‌ای طراحی کنید به کمک T-F.F که دیاگرام زیر را بشمارد؟

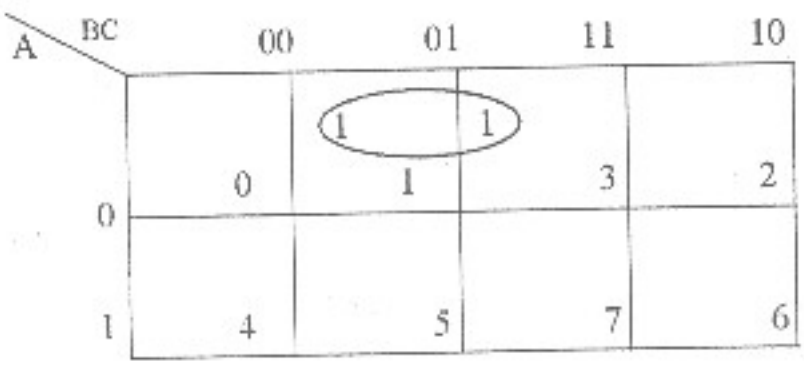
$0 \rightarrow 1 \rightarrow 7 \rightarrow 2 \rightarrow 6$
$\uparrow \quad 4 \leftarrow \leftarrow 5 \quad 3 \quad \downarrow$

بیتی $\left[\log_2^{2'}\right] = 3$

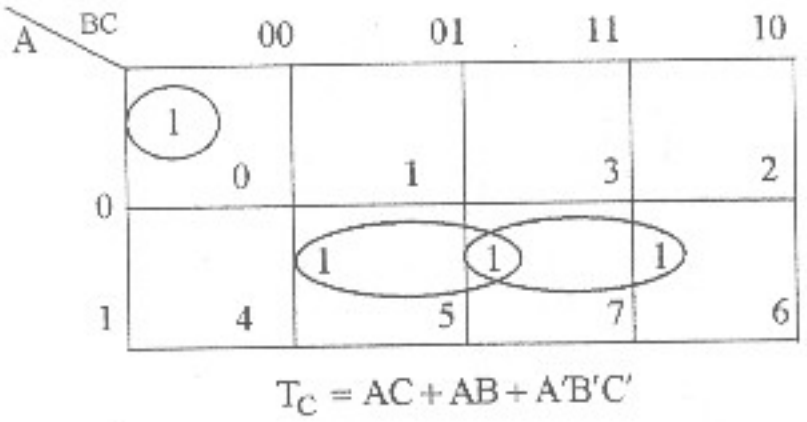
A	B	C	A	B	C	T_A	T_B	T_C
0	0	0	0	0	1	0	0	1
0	0	1	1	1	1	1	1	0
0	1	0	1	1	0	1	0	0
0	1	1	1	0	1	1	1	0
1	0	1	1	0	0	0	0	1
1	1	0	0	1	1	1	0	1
1	1	1	0	1	0	1	0	1



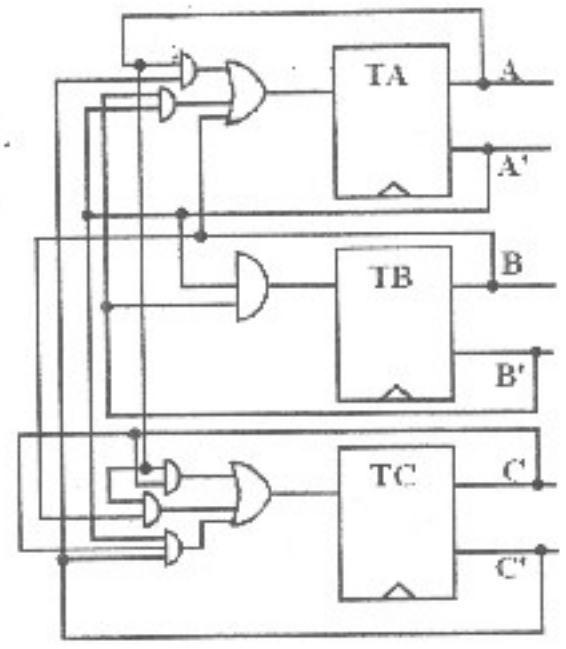
$T_A = A'C + AC' + B$



$T_B = CA'$



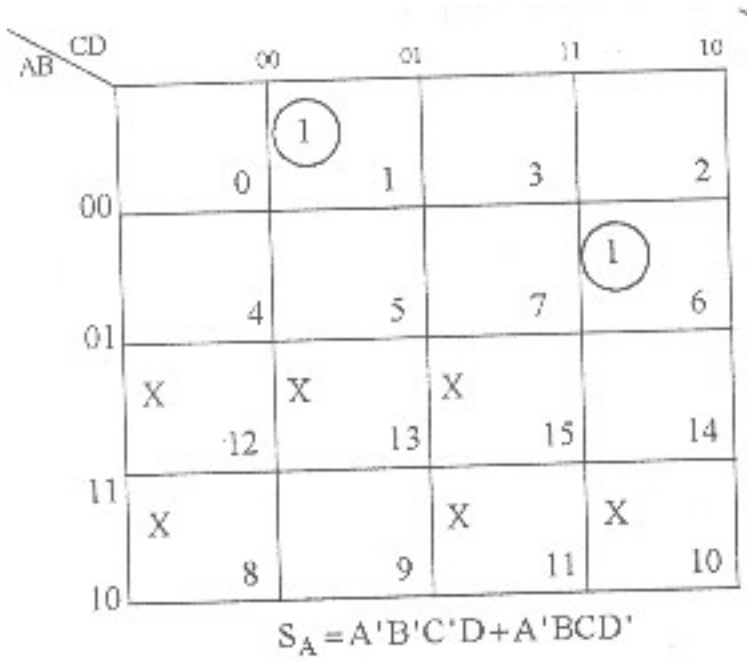
$$T_C = AC + AB + A'B'C'$$



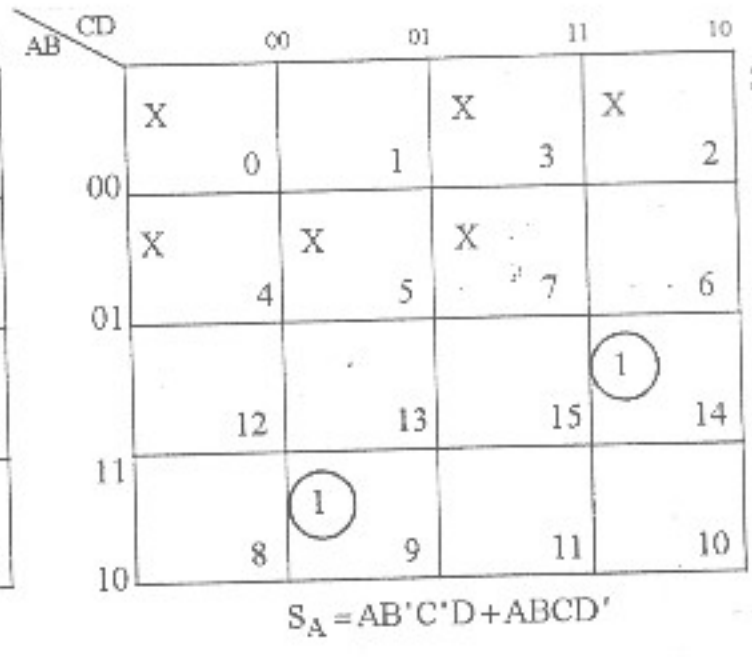
A B C
1 0 1
1 0 0 1
1 0 1 2
1 0 1 6

۴- به کمک R.S – F.F یک شمارندهٔ چهاربیتی طراحی کنید که برای ورودی زوج، حالت بعدی به اضافه 3 حالت فعلی باشد و برای ورودی فرد از حالت فعلی 2 تا کم کنیم تا حالت بعدی بدست آید. (طراحی انجام شود)

A	B	C	D	A	B	C	D	S_A	R_A	S_B	R_B	S_C	R_C	S_D	R_D
0	0	0	0	0	0	1	1	0	X	0	X	1	0	1	0
0	0	0	1	1	1	1	1	1	0	1	0	1	0	X	0
0	0	1	0	0	1	0	1	0	X	1	0	0	1	1	0
0	0	1	1	0	0	0	1	0	X	0	X	0	1	X	0
0	1	0	0	0	1	1	1	0	X	X	0	1	0	1	0
0	1	0	1	0	0	1	1	0	X	0	1	1	0	X	0
0	1	1	0	1	0	0	1	1	0	0	1	0	1	1	0
0	1	1	1	0	1	0	1	0	X	X	0	0	1	X	0
1	0	0	0	1	0	1	1	X	0	0	X	1	0	1	0
1	0	0	1	0	1	1	1	0	1	1	0	1	0	x	0
1	0	1	0	1	1	0	1	X	0	1	0	0	1	1	0
1	0	1	1	1	0	0	1	X	0	0	X	0	1	X	0
1	1	0	0	1	1	1	1	X	0	X	0	1	0	1	0
1	1	0	1	1	0	1	1	X	0	0	1	1	0	X	0
1	1	1	0	0	0	0	1	0	1	0	1	0	1	1	0
1	1	1	1	1	1	0	1	X	0	X	0	0	1	X	0
														1	0



$S_A = A'B'C'D + A'BCD'$



$S_A = AB'C'D + ABCD'$

$S_B = B'C'D + B'CD'$

$R_B = BC'D + BCD'$

$S_C = C'$

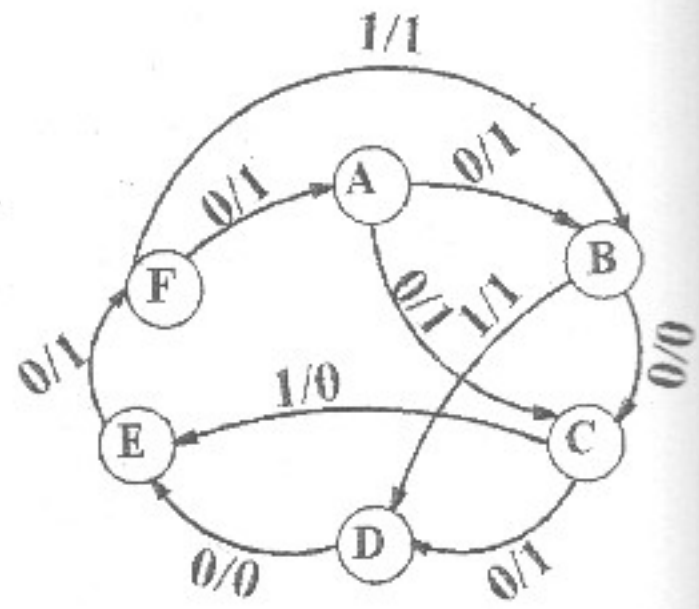
$R_C = C$

$S_D = 1$

$R_D = 0$

۵ـ مدار مربوط به دیاگرام حالات زیر را به کمک F.F – T طراحی کنید.

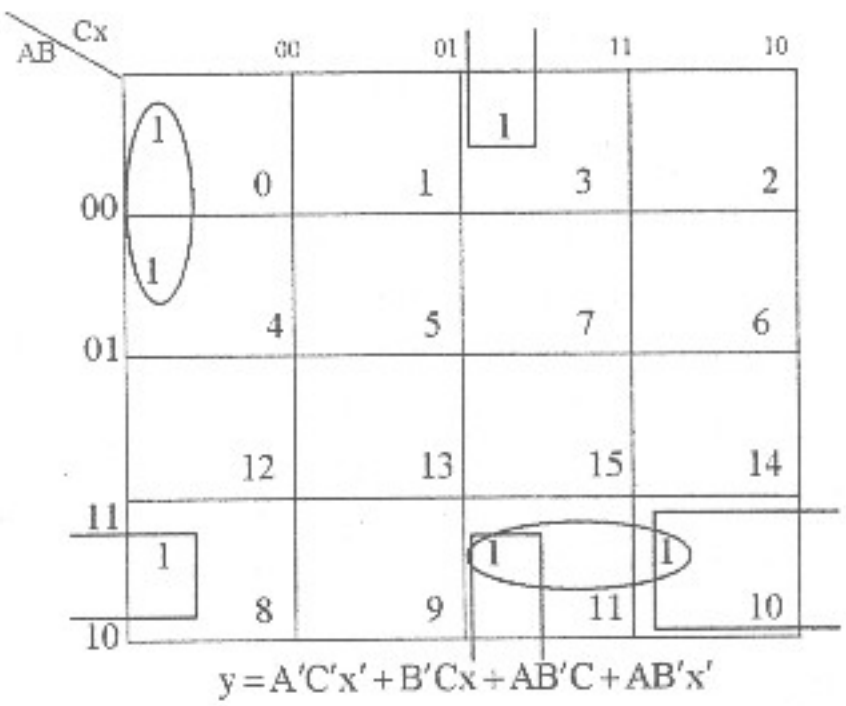
(حالت‌هایی را که نداریم صفر می گیریم.)



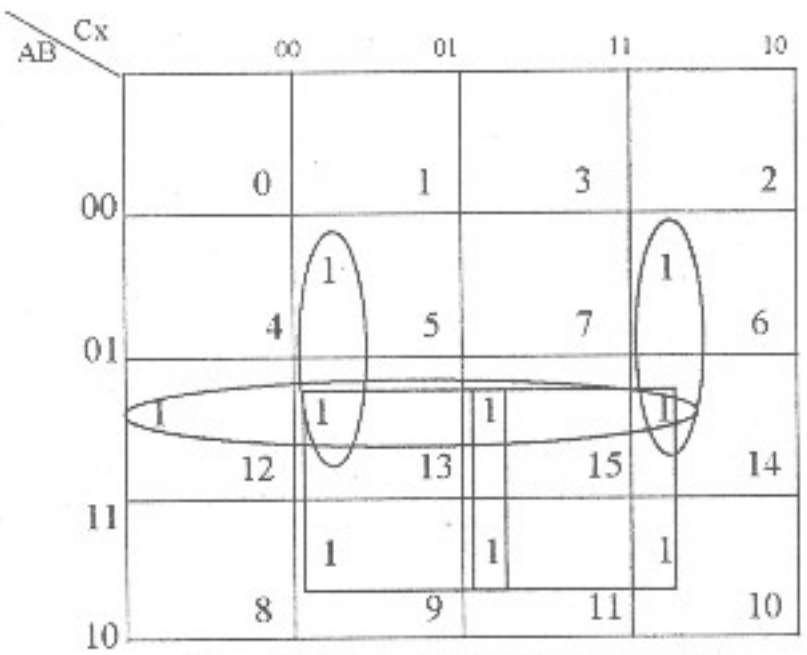
A	B	C	x	A	B	C	y	T_A	T_B	T_C
0	0	0	0	0	0	1	1	0	0	1
0	0	0	1	0	1	0	0	0	1	0
0	0	1	0	0	1	0	0	0	1	1
0	0	1	1	0	1	1	1	0	1	0
0	1	0	0	0	1	1	1	0	0	1
0	1	0	1	1	0	0	0	1	1	0
0	1	1	0	1	0	0	0	1	1	1
0	1	1	1	0	0	0	0	0	1	1
1	0	0	0	1	0	1	1	0	0	1
1	0	0	1	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	1	0	1
1	0	1	1	0	0	1	1	1	0	0
1	1	0	0	0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

A 000
B 001
C 010
D 011
E 100
F 101

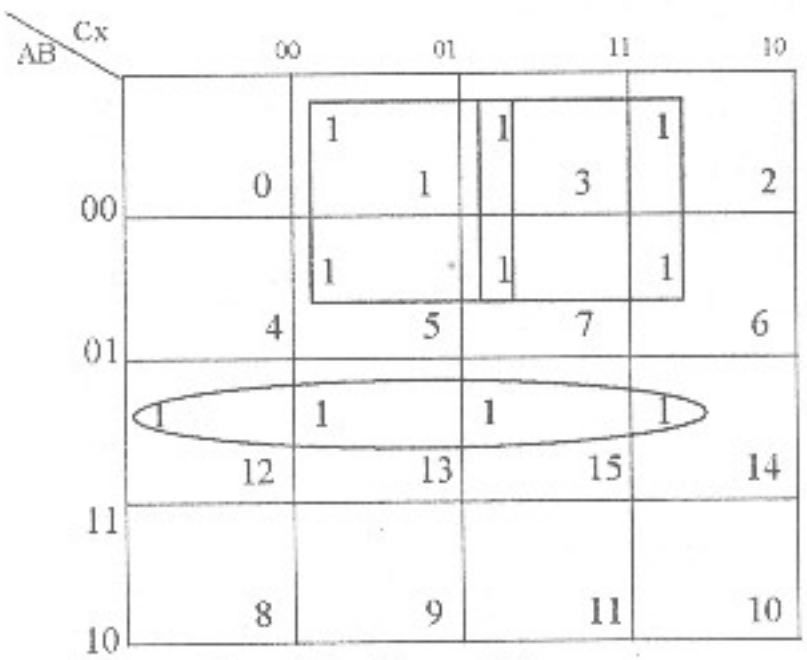
$\left[\log_2^6\right] = 3$



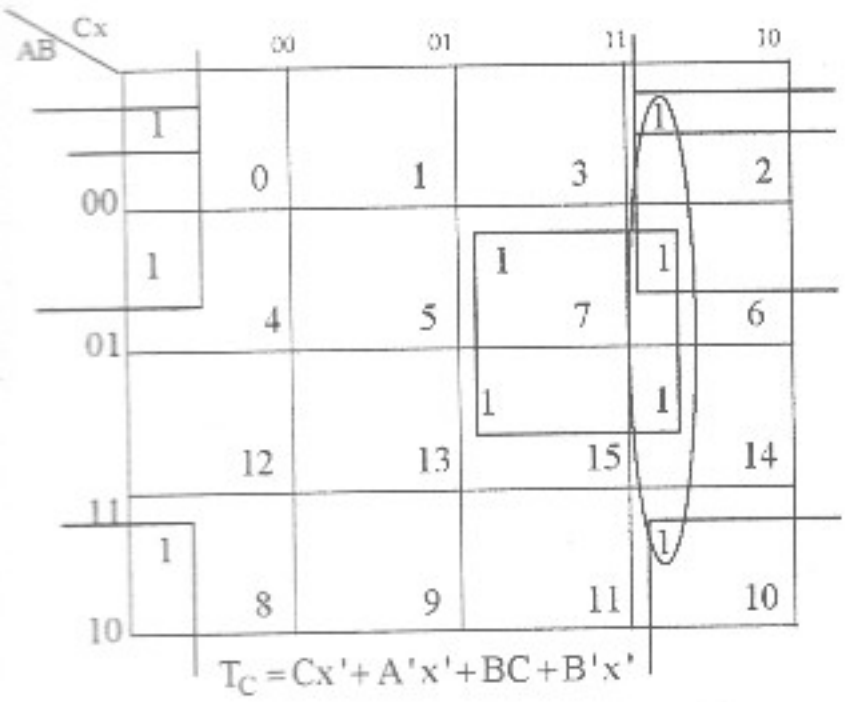
$$y = A'C'x' + B'Cx + AB'C + AB'x'$$



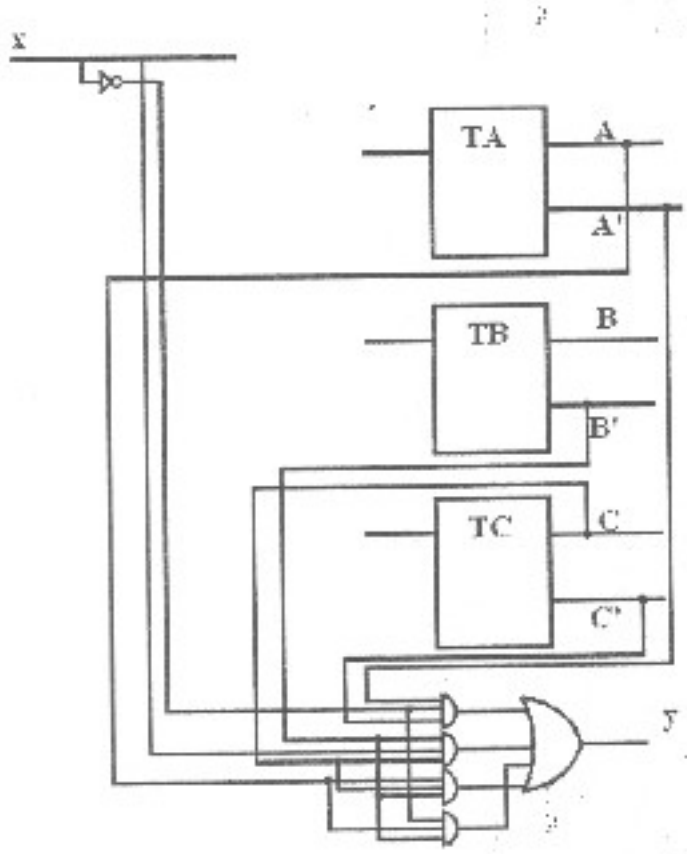
$$T_A = AB + Ax + AC + BC'x + BCx'$$



$$T_B = AB + A'x + A'C$$



$T_C = Cx' + A'x' + BC + B'x'$



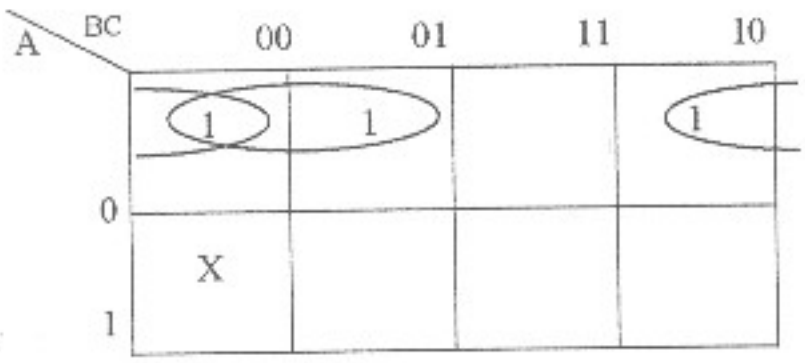
مابقی شکل با توجه به روابط بدست آمده از جدول کارنوها رسم شود.

۶- به کمک RS – F.F شمارنده‌ای طراحی کنید که دنباله زیر را بشمارد.

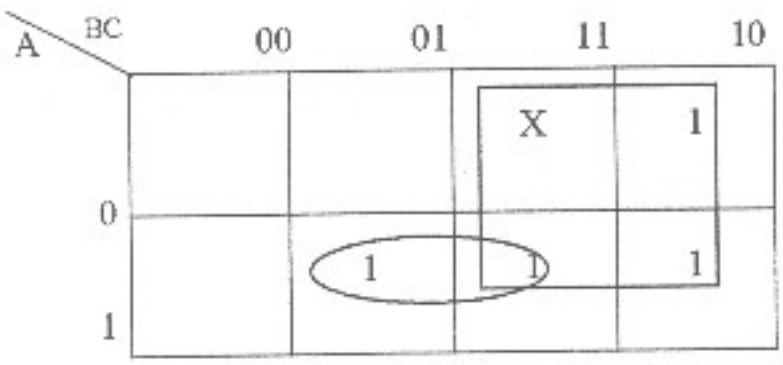
$0 \rightarrow 6 \rightarrow 2 \rightarrow 4$
$\uparrow$
$3 \leftarrow 5 \leftarrow 1 \leftarrow 7$

$\begin{bmatrix} 8 \\ 20g_2 \end{bmatrix} = 3$ تعداد بیتها

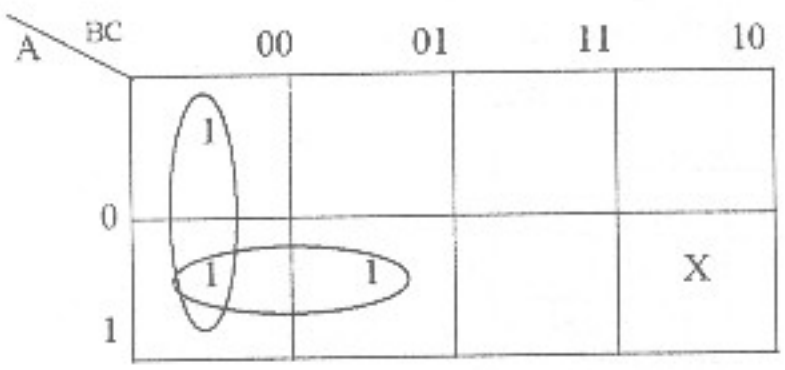
A	B	C	A	B	C	S_A	R_A	S_B	R_B	S_C	R_C
0	0	0	1	1	0	1	0	1	0	0	X
0	0	1	1	0	1	1	0	0	X	X	0
0	1	0	1	0	0	1	0	0	1	0	X
0	1	1	0	0	0	0	X	0	1	0	1
1	0	0	1	1	1	X	0	1	0	1	0
1	0	1	0	1	1	0	1	1	0	X	0
1	1	0	0	1	0	0	1	X	0	0	X
1	1	1	0	0	1	0	1	0	1	X	0



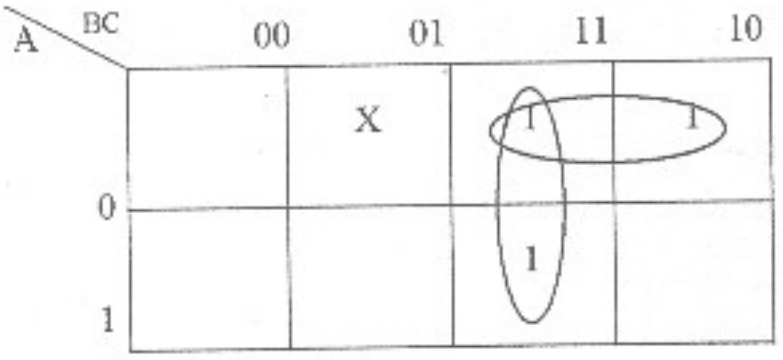
$$S_A = A'B' + A'C'$$



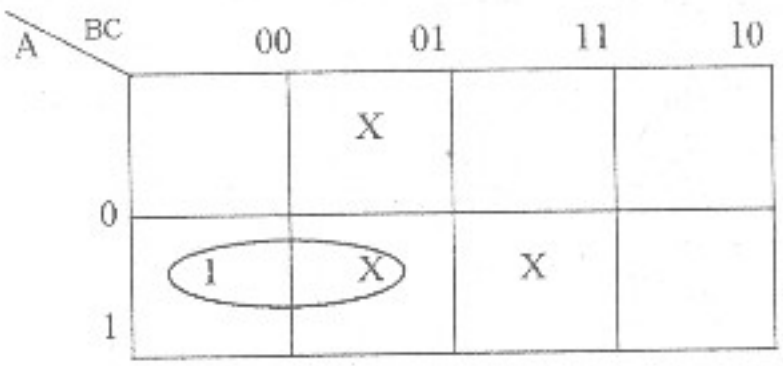
$$R_A = AC + B$$



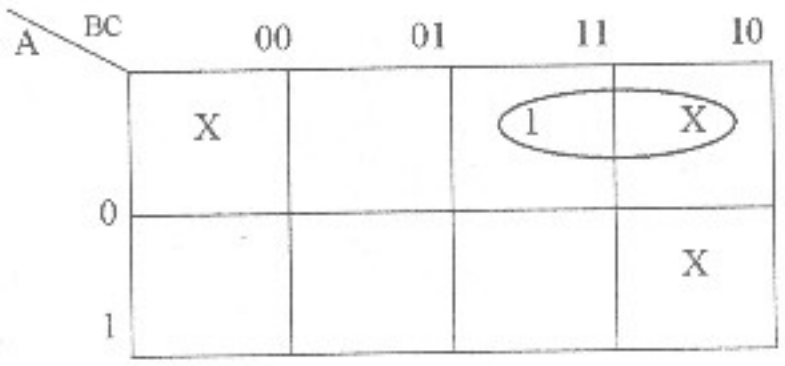
$$S_B = AB' + B'C'$$



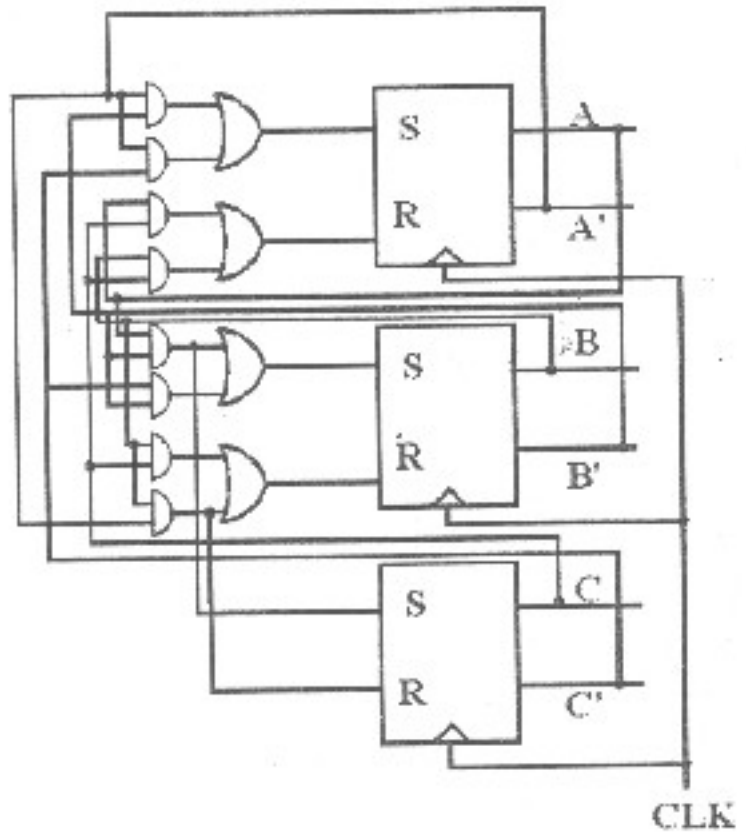
$$R_B = BC + A'B$$



$$S_C = AB'$$

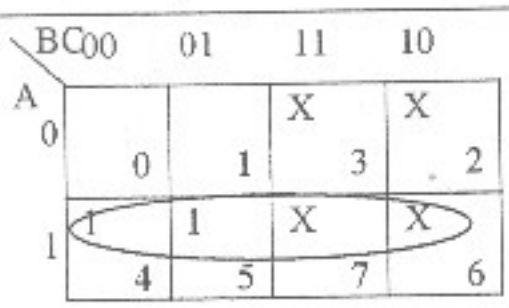


$$R_C = A'B$$

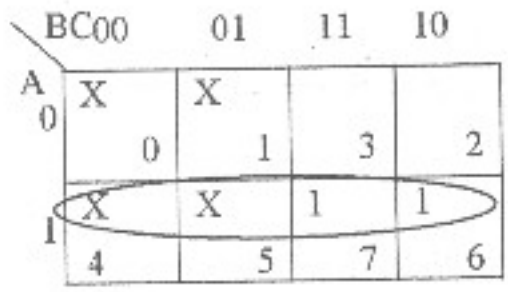


۷- به کمک jk .F.F مداری طراحی کنید که ورودی آن یک عدد سه بیتی باشد و خروجی‌اش معادل Gray ورودی.

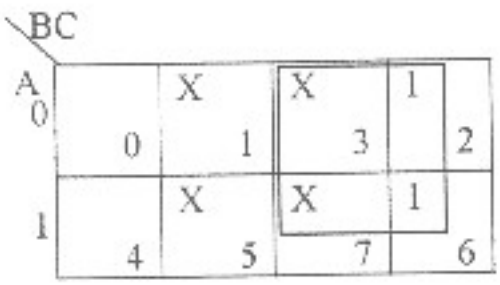
حالت فعلی			حالت بعدی			ورودی‌های F.F					
A_n	B	C	A_{n+1}	B	C	j_A	k_A	j_B	k_B	j_C	k_C
0	0	0	0	0	0	0	X	0	X	0	X
0	0	1	0	0	1	0	X	0	X	X	0
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	1	0	0	X	X	0	X	1
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	1	1	X	0	1	X	X	0
1	1	0	1	0	1	X	0	X	1	1	X
1	1	1	1	0	0	X	0	X	1	X	1



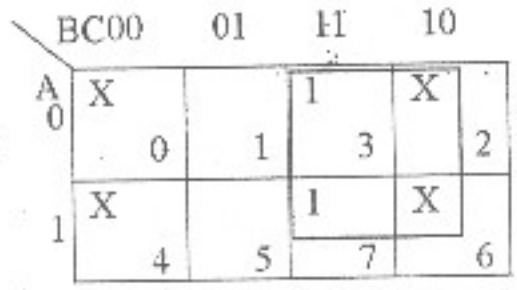
$j_B = A$



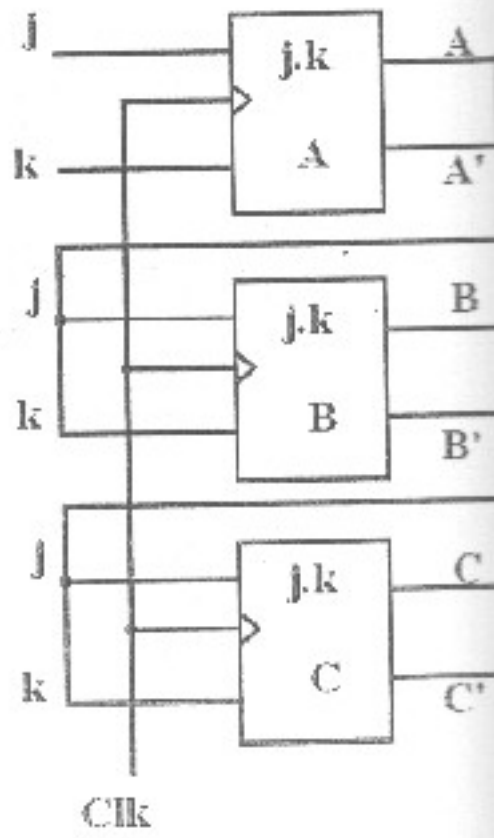
$k_B = A$



$j_C = B$

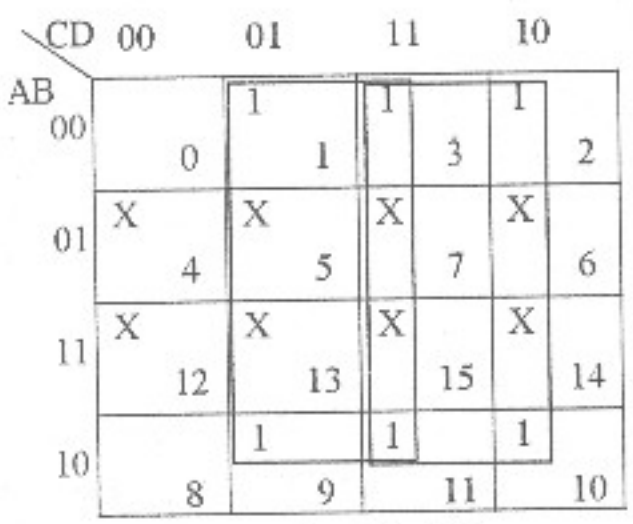


$k_C = B$

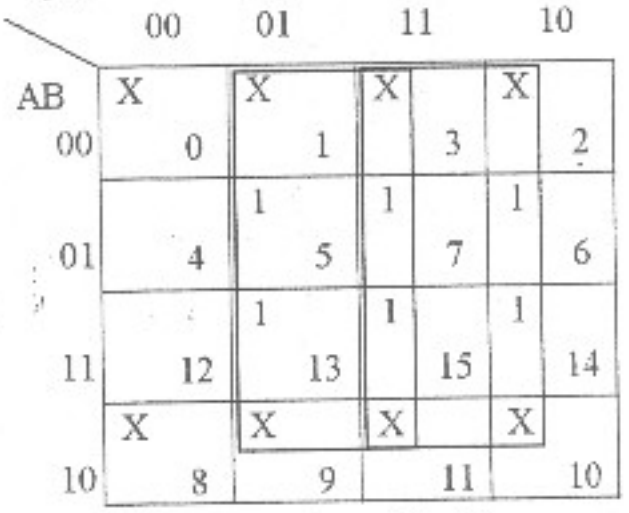


۸- به کمک jk . F.F مداری طراحی کنید که ورودی آن یک عدد 4 بیتی باشد و خروجی‌اش مکمل 2 ورودی باشد.

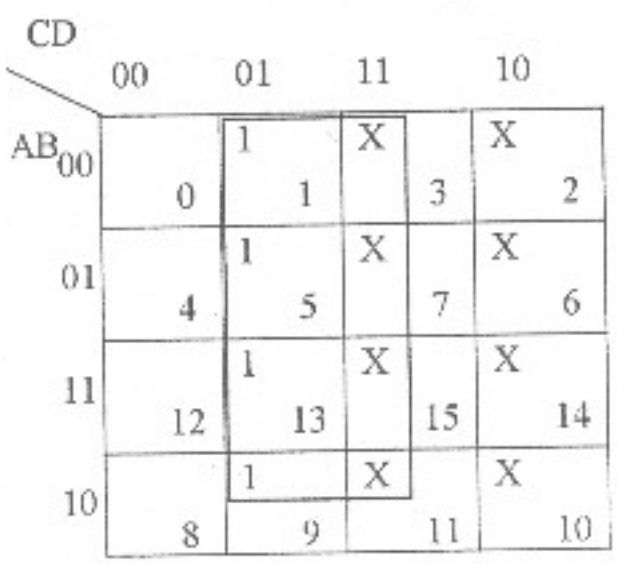
A_t	B	C	D	A	B	C	D	j_A	k_A	j_B	k_B	j_C	k_C	j_D	k_D
0	0	0	0	0	0	0	0	0	X	0	X	0	X	0	X
0	0	0	1	1	1	1	1	1	X	1	X	1	X	X	0
0	0	1	0	1	1	1	0	1	X	1	X	X	0	0	X
0	0	1	1	1	1	0	1	1	X	1	X	X	1	X	0
0	1	0	0	1	1	0	0	1	X	X	0	0	X	0	X
0	1	0	1	1	0	1	1	1	X	X	1	1	X	X	0
0	1	1	0	1	0	1	0	1	X	X	1	X	0	0	X
0	1	1	1	1	0	0	1	1	X	X	1	X	1	X	0
1	0	0	0	1	0	0	0	X	0	0	X	0	X	0	X
1	0	0	1	0	1	1	1	X	1	1	X	1	X	X	0
1	0	1	0	0	1	1	0	X	1	1	X	X	0	0	X
1	0	1	1	0	1	0	1	X	1	1	X	X	1	X	0
1	1	0	0	0	1	0	0	X	1	X	0	0	X	0	X
1	1	0	1	0	0	1	1	X	1	X	1	1	X	X	0
1	1	1	0	0	0	1	0	X	1	X	1	X	0	0	X
1	1	1	1	0	0	0	1	X	1	X	1	X	1	X	0



$j_B = C + D$

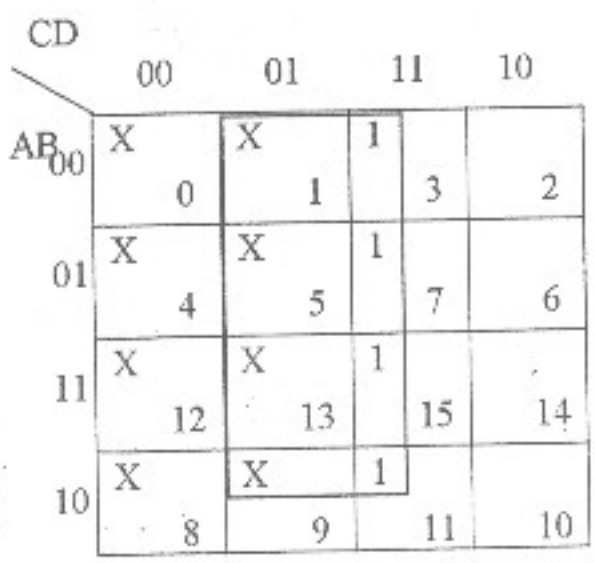


$k_B = C + D$



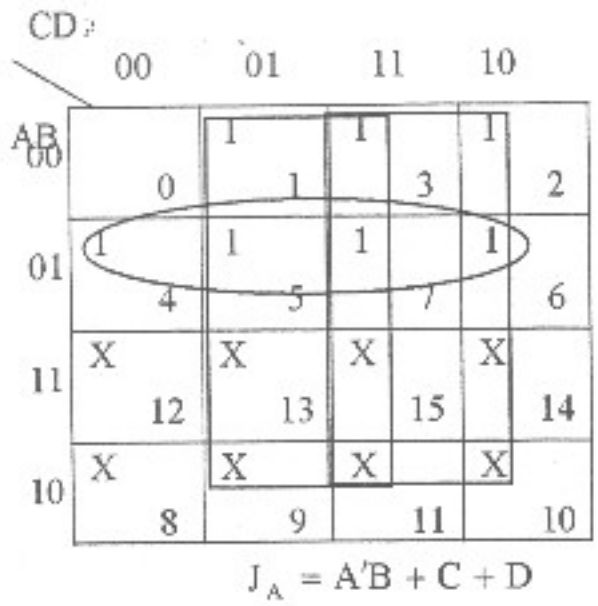
$j_C = D$

$j_D = 0$

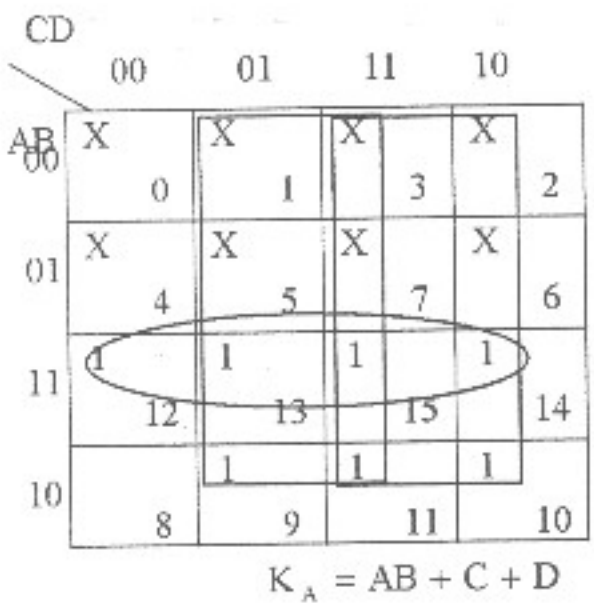


$k_C = D$

$k_D = 0$



$J_A = A'B + C + D$



$K_A = AB + C + D$

۹- با توجه به روابط زیر به کمک R.S.F.F مدار را طراحی کنید.(x ورودی و y خروجی)

$y = x'(A+B)$

$A_{t+1} = Ax + Bx$

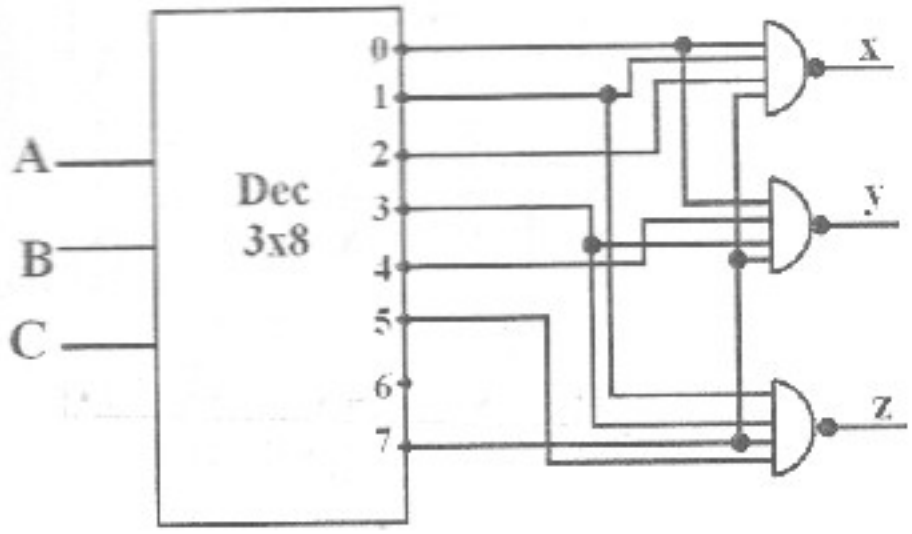
$B_{t+1} = A'x$

۱۰- دیاگرام حالات مربوط به مثال فوق را رسم کنید.

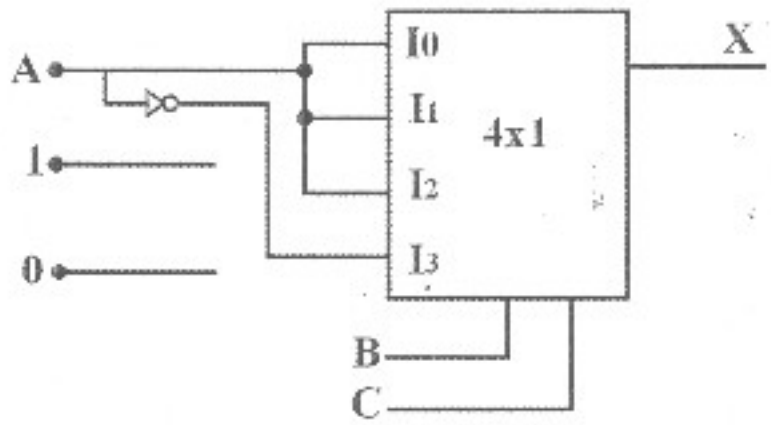
تعداد فلیپ فلاپ‌ها = n

2^n = تعداد حالات

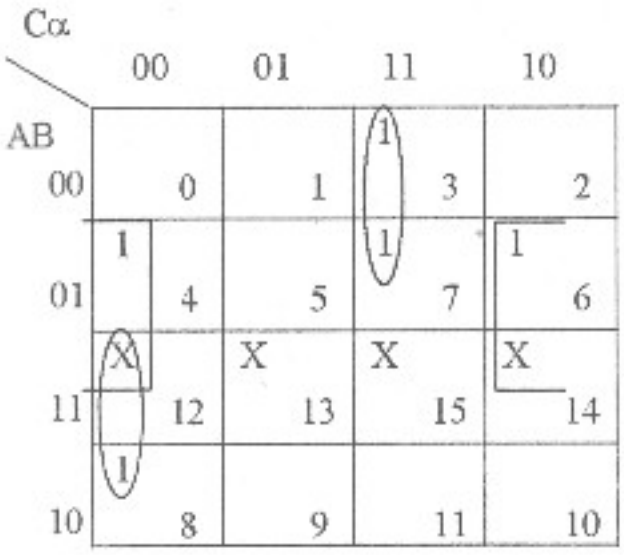
تعداد فلیپ فلاپ‌ها = $\lceil \log_2 \quad \rceil$



۱۱ـ با توجه به دیاگرام حالات زیر مدار را به کمک jK. F.F طراحی کنید.



A	B	C	α	A	B	C	B	j_A	k_A	j_B	k_B	j_C	k_C
0	0	0	0	0	0	1	0	0	X	0	X	1	X
0	0	0	1	1	0	1	0	1	X	0	X	1	X
0	0	1	0	0	1	0	0	0	X	1	X	X	1
0	0	1	1	1	0	0	1	1	X	0	X	X	1
0	1	0	0	0	1	1	1	0	X	X	0	1	X
0	1	0	1	0	0	0	0	0	X	X	1	0	X
0	1	1	0	1	0	0	1	1	X	X	1	X	1
0	1	1	1	0	0	1	1	0	X	X	1	X	0
1	0	0	0	1	0	1	1	X	0	0	X	1	X
1	0	0	1	0	1	1	0	X	1	1	X	1	X
1	0	1	0	0	0	0	0	X	1	0	X	X	1
1	0	1	1	1	0	0	0	X	0	0	X	X	1
1	1	0	0	X	X	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X



$$B = B\alpha' + A'C\alpha + AC'\alpha'$$

Cα / AB	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	X 12	X 13	X 15	X 14
10	X 8	X 9	X 11	X 10

$$j_A = B'\alpha + BC\alpha'$$

Cα / AB	00	01	11	10
00	X 0	X 1	X 3	X 2
01	X 4	X 5	X 7	X 6
11	X 12	X 13	X 15	X 14
10	8	1 9	11	1 10

$$k_A = C'\alpha + C\alpha'$$

Cα / AB	00	01	11	10
00	0	1	3	2
01	X 4	X 5	X 7	X 6
11	X 12	X 13	X 15	X 14
10	8	1 9	11	10

$$j_B = A'C\alpha' + AC'\alpha$$

$$j_C = B'C' + C'\alpha'$$

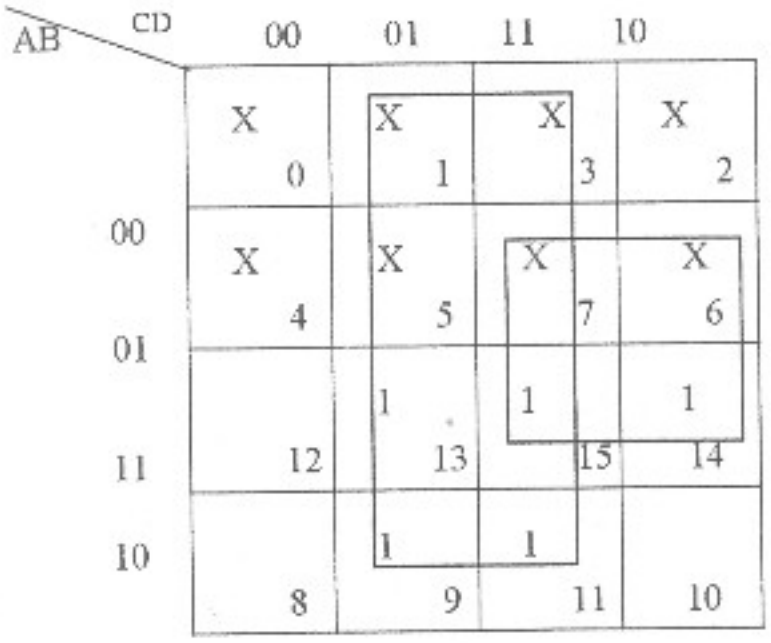
Cα / AB	00	01	11	10
00	X 0	X 1	X 3	X 2
01	1 4	1 5	1 7	1 6
11	X 12	X 13	X 15	X 14
10	X 8	X 9	X 12	X 11

$$k_B = C + \alpha$$

$$k_C = B'C + C\alpha'$$

۱۲ - یک شمارندهٔ 4 بیتی به کمک jk . F . F طراحی کنید که تنها اعداد زوج را به شمارد (راهنمایی: اگر حالت فرد بودند صفر را بشمارند) (شمارنده‌ای که مرتب است.)

A	B	C	D	A	B	C	D	j_A	k_A	j_B	k_B	j_C	k_C	j_D	k_D
0	0	0	0	0	0	1	0	0	X	0	X	1	X	0	X
0	0	0	1	0	0	0	0	0	X	0	X	0	X	X	1
0	0	1	0	0	1	0	0	0	X	1	X	X	1	0	X
0	0	1	1	0	0	0	0	0	X	0	X	X	1	X	1
0	1	0	0	0	1	1	0	0	X	X	0	1	X	0	X
0	1	0	1	0	0	0	0	0	X	X	1	0	X	X	1
0	1	1	0	1	0	0	0	1	X	X	1	X	1	0	X
0	1	1	1	0	0	0	0	0	X	X	1	X	1	X	1
1	0	0	0	1	0	1	0	X	0	0	X	1	X	0	X
1	0	0	1	0	0	0	0	X	1	0	X	0	X	X	1
1	0	1	0	1	1	0	0	X	0	1	X	X	1	0	X
1	0	1	1	0	0	0	0	X	1	0	X	X	1	X	1
1	1	0	0	1	1	1	0	X	0	X	0	1	X	0	X
1	1	0	1	0	0	0	0	X	1	X	1	0	X	X	1
1	1	1	0	0	0	0	0	X	1	X	1	X	1	0	X
1	1	1	1	0	0	0	0	X	1	X	1	X	1	X	1



$k_A = D + BC$

AB \ CD	00	01	11	10
	0	1	3	2
00				1
	4	5	7	6
01	X	X	X	X
	12	13	15	14
11	X	X	X	X
	8	9	11	10
10				

$$J_A = BCD'$$

AB \ CD	00	01	11	10
	0	1	3	1 2
00	X	X	X	X
	4	5	7	6
01	X	X	X	X
	12	13	15	14
11				1
	8	9	11	10
10				

$$k_A = D + BC$$

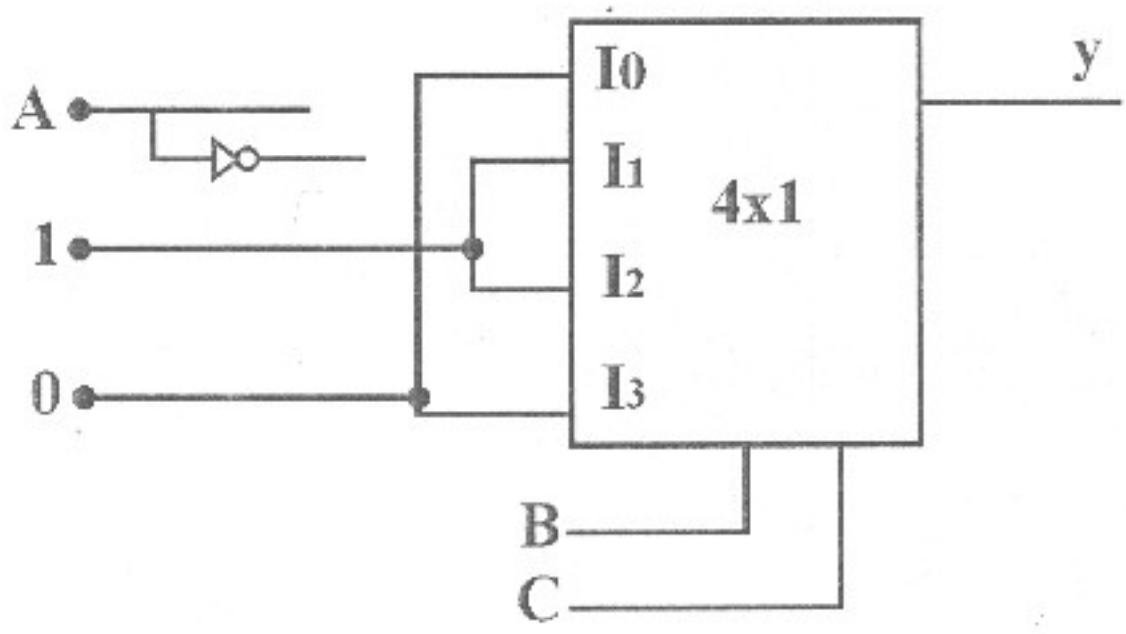
AB \ CD	00	01	11	10
	X 0	X 1	X 3	X 2
00		1	1	1
	4	5	7	6
01		1	1	1
	12	13	15	14
11	X	X	X	X
	8	9	11	10
10				

$$k_B = D + C$$

AB \ CD	00	01	11	10
00	1 (0)	(1)	X (3)	X (2)
01	1 (4)	(5)	X (7)	X (6)
11	1 (12)	(13)	X (15)	X (14)
10	1 (8)	(9)	X (11)	X (10)

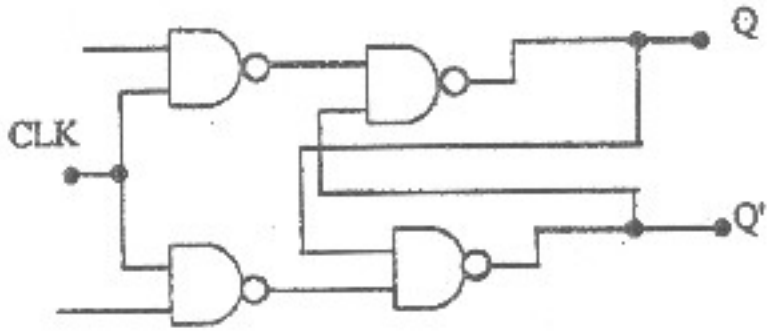
$J_C = D'$

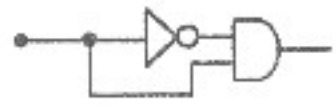
$K_C = 1 \quad J_D = 0 \quad K_D = 1$

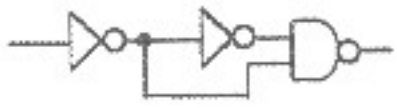


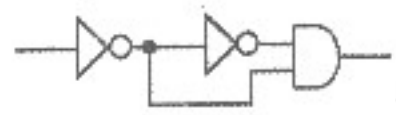
مجموعه تست‌های فصل چهارم:

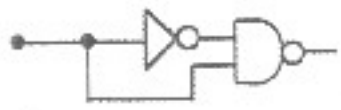
۱ـ در شکل مقابل یک فلیپ فلاپ RS ساعت‌دار ملاحظه می‌کنید. برای آن‌که این فلیپ فلاپ روی لبه پایین رونده پالس ساعت عمل کند، کدام یک از مدارهای زیر باید در مسیر سیگنال ساعت، در ورودی فلیپ فلاپ قرار گیرد؟



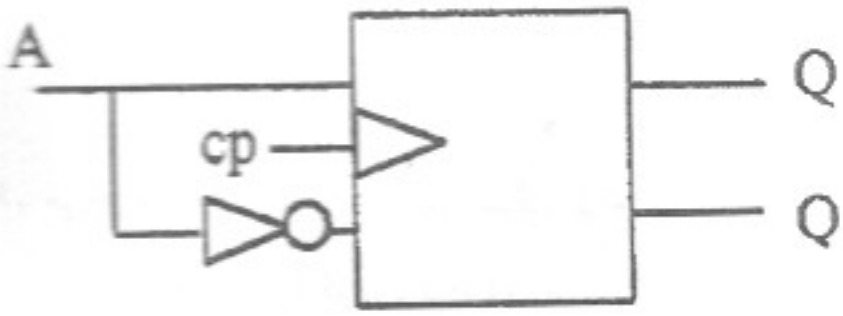
۱) 

۲) 

۳) 

۴) 

۲ـ فلیپ فلاپ شکل مقابل با کدام لبه پالس فعال شده و از کدام نوع است؟



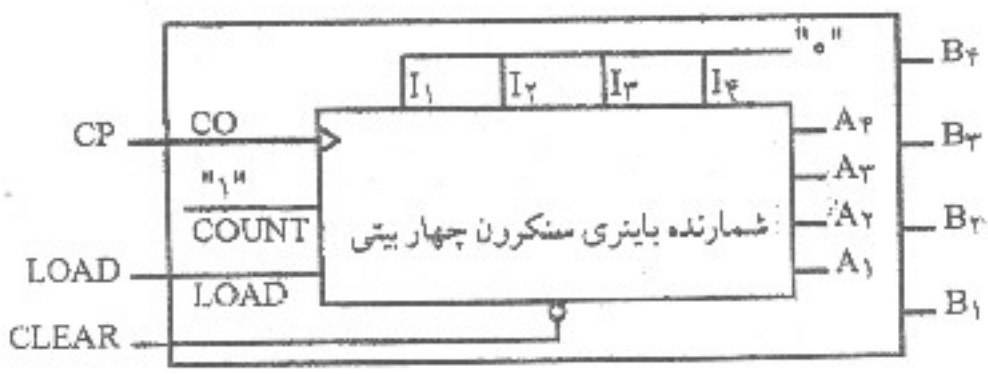
۱) پایین رونده ـ D

۲) پایین رونده ـ T

۳) بالا رونده ـ D

۴) بالا رونده ـ T

۳ـ برای آن‌که خروجی $B_4B_3B_2B_1$ مدار زیر اعداد زوج بین صفر و پانزده را مکرراً بشمارد لازم است که:



۱) $B_1=A_1, B_2=A_2, B_3=A_3, B_4=A_4, CLEAR=\bar{A}_4, LOAD="0"$ قرار داده شود.

۲) $B_1=0, B_2=A_1, B_3=A_2, B_4=A_3, CLEAR=A_4, LOAD="0"$ قرار داده شود

۳) $B_1=0, B_2=A_1, B_3=A_2, B_4=A_3, clear=\bar{A}_4, load="0"$ قرار داده شود.

۴) $B_1=0, B_2=A_1, B_3=A_2, B_4=A_3, clear=\bar{A}_4, load="1"$ قرار داده شود.

۴- معادلات ورودی فلیپ‌فلاپ T از یک شمارنده نزولی باینری سنکرون 4 بیتی را بنویسید. (کوچکترین بیت LSB شمارنده Q_A و بیت‌های دیگر به ترتیب Q_B و Q_C و Q_D می‌باشند).

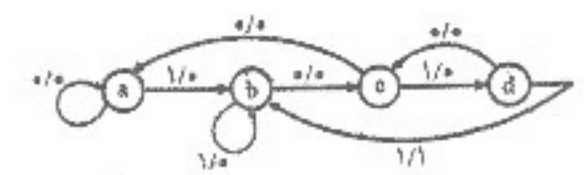
۱) $T_A = 1 \quad T_B = \bar{Q}_A \quad T_C = \bar{Q}_A.\bar{Q}_B \quad T_D = \bar{Q}_A.\bar{Q}_B.\bar{Q}_C$

۲) $T_A = 1 \quad T_B = Q_A \quad T_C = Q_A.Q_B \quad T_D = Q_A.Q_B.Q_C$

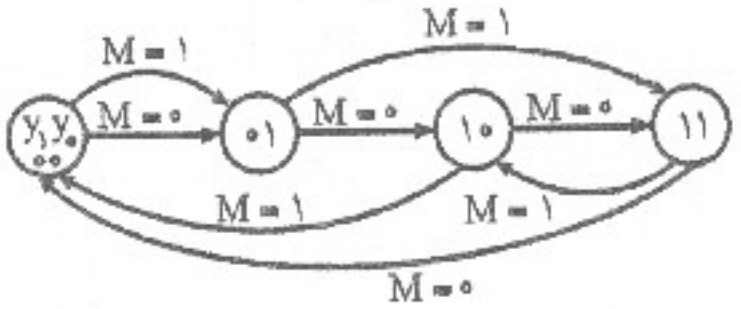
۳) $T_A = 1 \quad T_B = \bar{Q}_A \quad T_C = Q_A.Q_B \quad T_D = \bar{Q}_A.Q_B.\bar{Q}_C$

۴) $T_A = T_B = 1 \; , \; T_C = \bar{Q}_A \quad , \quad T_D = \bar{Q}_A.\bar{Q}_C$

۵- دز دیاگرام (State Diagram) زیر کدام خروجی درست است؟ وضعیت اولیه سیستم حالت a می‌باشد.



۶- می‌خواهیم یک شمارنده دو بیتی همگام طرح کنیم که با توجه به مقدار ورودی کنترلی M در کد باینری و یا گری (Gary) بشمارد. نمودار حالت در شکل زیر داده شده است. اگر از دو فلیپ فلاپ J . K استفاده کنیم، مقادیر ورودی فلیپ‌فلاپ‌ها را بدست آورید. (J_1 و K_1 ورودی‌های فلیپ‌فلاپ با ارزش بیشتر هستند)



۱) $J_1 = y_0 \; , \; K_1 = M \oplus y_0 \; , \; J_0 = M' + y_1' \quad , \quad K_0 = M' + y_1$

۲) $J_1 = M' + y_1' \; , \; K_1 = M' + y_1 \; , \; J_0 = y_0 \quad , \quad K_0 = M \oplus y$

۳) $J_1 = y_0' \; , \; K_1 = M\,y_0' + M'y \quad , \quad J_0 = M + y_1 \quad , \quad K_0 = M \oplus y_0$

۴) $J_1 = y_0 \; , \; K_1 = 1 \; , \; J_0 = M'y_0' + My_1' \quad K_0 = M' \oplus y_1$

۷- اگر با استفاده از یک فلیپ‌فلاپ T بخواهیم مداری طرح کنیم که رفتار یک فلیپ‌فلاپ J – K را داشته باشد ورودی فلیپ‌فلاپ توسط کدام یک از توابع زیر قابل بیان است؟

۱) $T = J = K$

۲) $T = J\bar{Q} + \bar{K}Q$

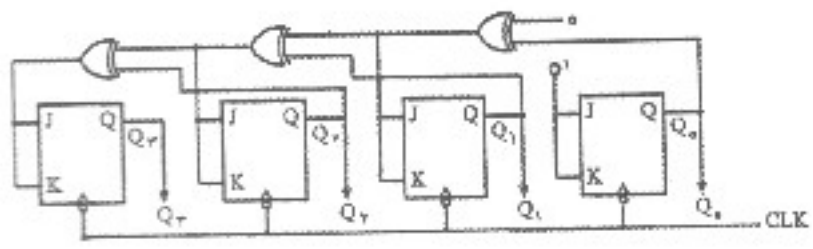
۳) $T = J\bar{Q} + KQ$

۴) با فلیپ‌فلاپ T نمی‌توان یک فلیپ‌فلاپ J – K طرح کرد

۸- جدول حالت مقابل را با دو فلیپ‌فلاپ J – K طراحی می‌کنیم. ورودی فلیپ‌فلاپ‌ها را تعیین کنید. (حالت بعدی به ازای دو مقدار ممکن ورودی x داده شده است و $y_2\, y_1$ خروجی فلیپ‌فلاپ‌ها هستند.)

$(y_2^+\, y_1^+)$ حالت بعدی حالت فعلی

$y_2 y_1$	x = 0	x = 1
00	00	01
01	10	01
10	10	11
11	11	00



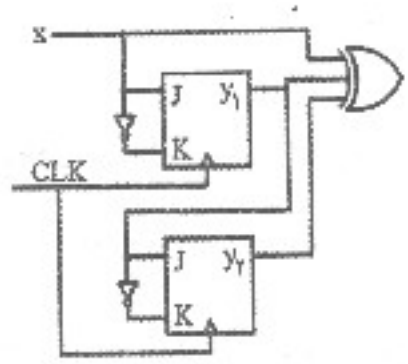
۱) $J_2 = \bar{x}K_2 = y_1 x_1\ ,\ J_1 = 1\ ,\ K_1 = y_2 \oplus x$

۲) $J_2 = 1\ ,\ K_2 = y_1\ ,\ J_1 = y_1\ ,\ K_1 = 1$

۳) $J_2 = y_1\bar{x}\ ,\ K_2 = y_1 x\ ,\ J_1 = x\ ,\ K_1 = \overline{y_2 \oplus x}$

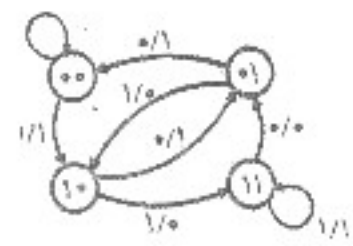
۴) $J_2 = y_1\bar{x}\ ,\ K_2 = y_2 y_1\ ,\ J_1 = x\ ,\ K_1 = y_2\bar{x}$

۹- کدام گزینه دیاگرام حالات مدار زیر را بیان می‌کند؟

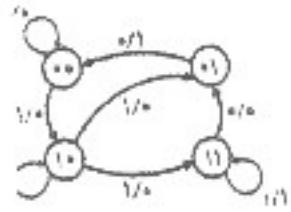


۱)

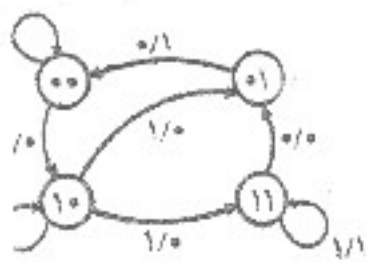
۲)



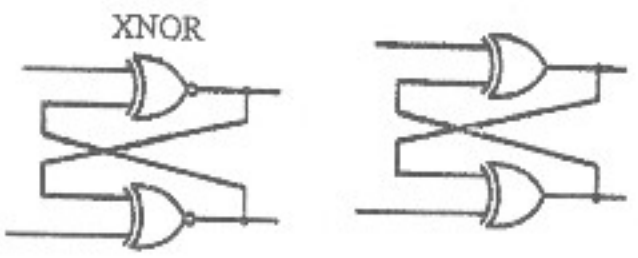
۳)



۴)



۱۰ ـ از دو شکل زیر بگویید کدام‌یک را می‌توان به‌جای SR – Latch استفاده نمود.



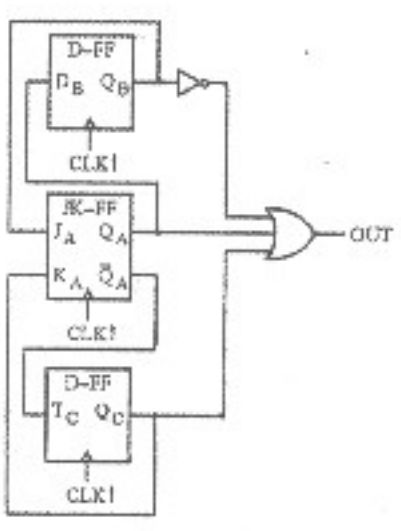
۱) مدار شکل الف

۲) مدار شکل ب

۳) هر دو مدار شکل الف و ب

۴) هیچ‌کدام از مدارها الف و ب

۱۱ ـ مدار مقابل از سه فلیپ‌فلاپ D ، JK و T تشکیل شده است اگر داشته باشیم در آغاز شروع کار $Q_A=1$ و $Q_B=0$ و $Q_c=1$ پس از دو پالس ساعت وضعیت خروجی فلیپ‌فلاپ‌ها و خروجی نهایی را معین سازید.



۱) $Y=1 \ , \ Q_c=1 \ , \ Q_B=1 \ , \ Q_A=1$

۲) $Y=1 \ , \ Q_c=1 \ , \ Q_B=1 \ , \ Q_A=0$

۳) $Y=1 \ , \ Q_c=0 \ , \ Q_B=0 \ , \ Q_A=1$

۴) $Y=1 \ , \ Q_c=1 \ , \ Q_B=0 \ , \ Q_A=1$

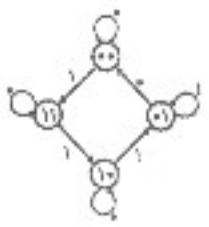
۱۲ ـ یک مدار ترتیبی سنکرون شامل دو فلیپ‌فلاپ j-k به نام‌های A و B است و معادلات ورودی فلیپ‌فلاپ‌ها در زیر نوشته شده است

$$\begin{cases} j_A = B\overline{x} \\ k_A = Bx \end{cases}$$

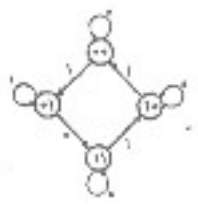
$$\begin{cases} j_B = x \\ k_A = A \odot x \end{cases}$$

دیاگرام حالت این مدار ترتیبی چیست؟

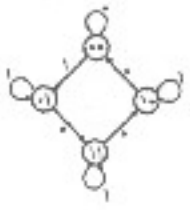
۱)



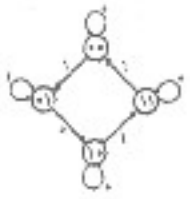
۲)



۳)



۴)



۱۳ ـ با یک فلیپ فلاپ D می‌خواهیم رفتار یک فلیپ‌فلاپ T را پیاده‌سازی کنیم. در این صورت:

۱) $D=T\oplus Q$ ۲) $D=T\odot Q$ ۳) $D=T\bar{Q}$ ۴) $D=\bar{T}Q$

۱۴ ـ پس از کاهش حالت جدول مقابل چند حالت خواهد داشت.

حالت فعلی	x=0	x=1
S_1	$S_1/0$	$S_6/0$
S_2	$S_1/1$	$S_6/1$
S_3	$S_4/0$	$S_5/1$
S_4	$S_1/0$	$S_7/0$
S_5	$S_2/0$	$S_3/0$
S_6	$S_4/0$	$S_5/0$
S_7	$S_2/0$	$S_3/0$

۱) 3 ۲) 5
۳) 7 ۴) 6

۱۵ ـ کدام جمله در مورد فلیپ فلاپ‌های Master – slave صحیح است؟

۱) حساس به لبه ۲) دو فیلیپ فلاپ آن به‌طور همزمان عمل می‌کنند.
۳) در زمان 1 بودن clock عمل می‌کند. ۴) هر سه گزینه صحیح است.

۱۶ ـ مزیت فلیپ فلاپ jk بر SR کدام است؟

۱) تأخیر آن کمتر است.
۲) از تعداد گیت‌های کمتری استفاده می‌کند.
۳) برخلاف فلیپ فلاپ SR حالت نامعلوم ندارد.
۴) مزیت خاصی ندارد.

۱۷ ـ اگر خروجی فلیپ فلاپ jk صفر باشد، با اعمال یک پالس ساعت نیز به صفر برسد. ورودی‌های آن کدام است؟ (d حالت بی‌اهمیت است)

۱) $k=1\ ,\ j=0$ ۲) $k=1\ ,\ j=d$ ۳) $k=d\ ,\ j=1$ ۴) $k=d\ ,\ j=0$

۱۸ ـ کدام‌یک از روابط زیر صحیح نیست؟

۱) $Q(t+1)=jQ'+k'Q$
۲) $Q(t+1)=TQ'+T'Q$
۳) $Q(t+1)=D$
۴) $Q(t+1)=S+RQ'$

۱۹ ـ علت حساس کردن فیلیپ فلاپ‌ها به لبه کدام است؟

۱) بالابردن سرعت انتقال و کاهش تأخیر گیت‌ها

۲) کاهش تعداد ترانزیستورهای مصرفی برای ساخت مدار

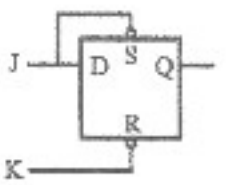
۳) جلوگیری از تأثیر تغییرات ورودی‌ها بر روی خروجی در هنگام 1 بودن پالس سرعت

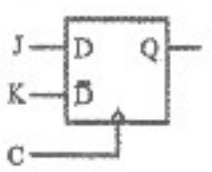
۴) امکان استفاده از آن‌ها در مدارات آسنکرون

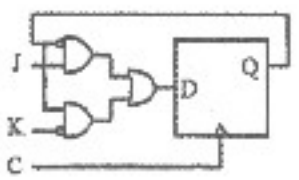
۲۰ ـ اگر ورودی یک فلیپ فلاپ T برابر 1 باشد. خروجی آن کدام است؟

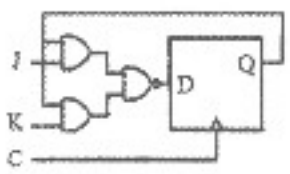
۱) خروجی قبلی ۲) ورودی قبلی ۳) مکمل خروجی قبلی ۴) مکمل ورودی قبلی

۲۱ ـ با استفاده از یک D – FF یک JK – FF بسازید.

۱)


۲)


۳)


۴)


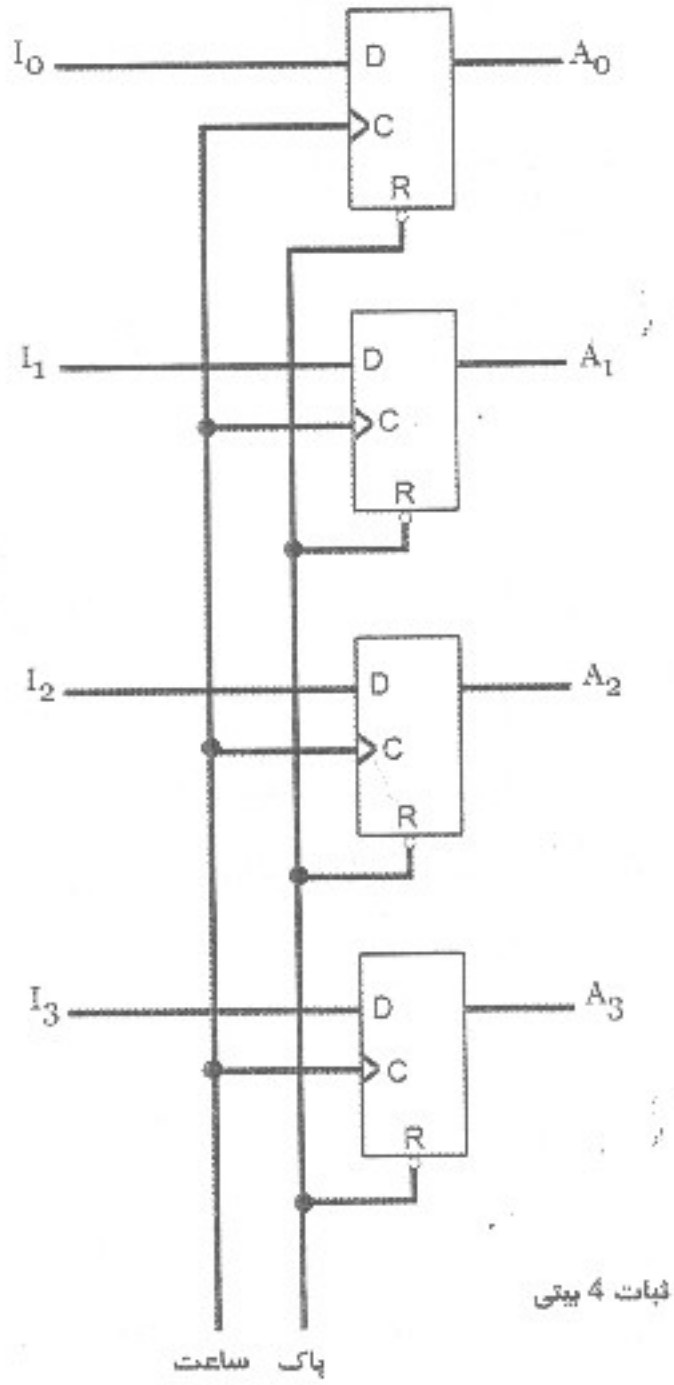
فصل پنجم

ثبات‌ها و شمارنده‌ها

ثبات‌ها یا رجیسترها:

ثبات‌ها مجموعه‌ای از خانه‌های حافظه هستند که هر خانه از یک فلیپ فلاپ تشکیل شده و توانایی این را دارد که یک بیت اطلاعات دودویی را درون خود نگه دارد (درون یک فلیپ فلاپ) یعنی یک ثبات n بیتی از n فلیپ فلاپ تشکیل شده است که در مجموع توانایی ذخیره‌سازی n بیت را دارا می‌باشد ساده‌ترین ثبات فقط از تعدادی فلیپ فلاپ تشکیل شده و در ساختمان آن هیچ گونه گیت خارجی وجود ندارد کلاک پالس توانایی فعال‌سازی فلیپ فلاپ‌ها و در مجموع ثبات ساده را به عهده دارد.

معمول‌ترین فلیپ فلاپی که در ساختن رجیستر مورد استفاده قرار می‌گیرد D فلیپ فلاپ است یعنی Data فلیپ فلاپ که به خود ی خود کار یک بافر را انجام می‌دهد.



ثبات 4 بیتی

I_0 تا I_3 ورودی‌ها هستند clk و clear یا Reset هم خطوط مشترک هستند. از خط clear برای Reset کردن فلیپ فلاپ‌ها یا پاک کردن مقادیر قبلی آن‌ها استفاده می‌شود. با صفر کردن خط Reset فلیپ فلاپ‌ها پاک می‌شوند

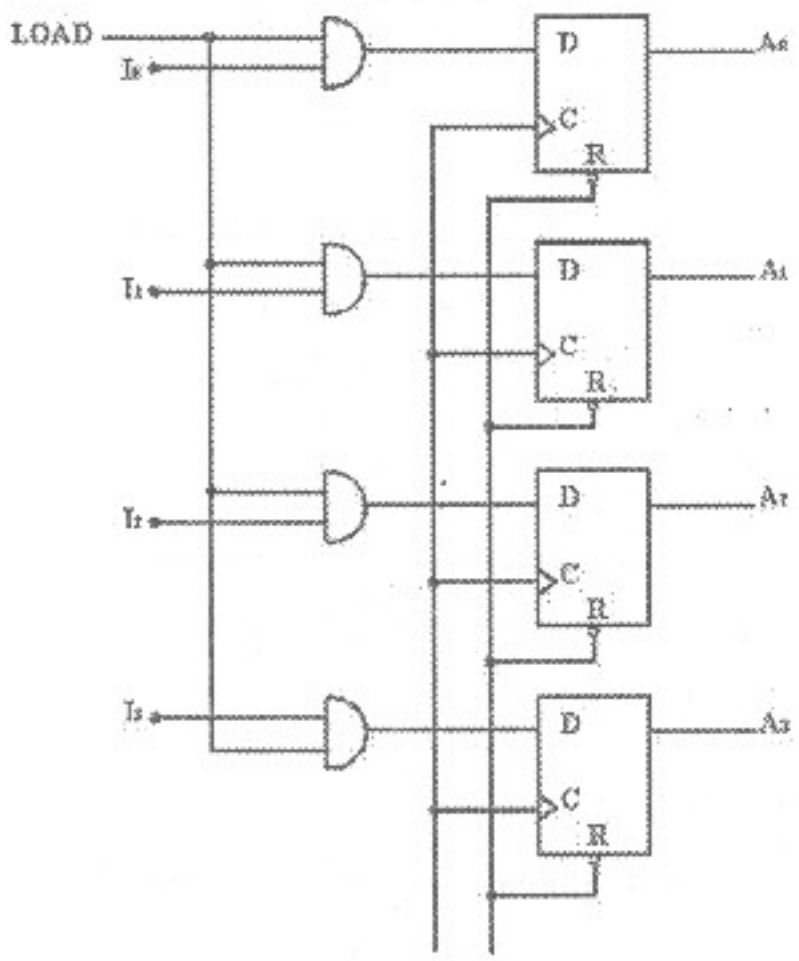
با یک شدن clk مقادیر ورودی I_0 تا I_3 وارد فلیپ فلاپ‌ها می‌شوند و در خروجی قابل مشاهده می‌باشند در زمانی که clk صفر است مقادیر خروجی مرحله قبل داخل فلیپ فلاپ‌ها باقی می‌مانند و می‌توان از آن‌ها استفاده کرد اگر تعداد فلیپ فلاپ‌ها مورد استفاده n تا بود رجیستر به دست آمده n بیتی می‌باشد.

ثبات با امکان رشد موازی loading

تعاریف:

۱ـ **بار شدن:** انتقال اطلاعات جدید به داخل یک ثبات را بار شدن می‌گویند.

۲ـ **بار شدن موازی:** زمانی که همه بیت‌های یک ثبات همزمان به کمک یک clk مشترک بار شوند بار شدن موازی می‌گویند. شکل زیر یک نمونه بار شدن موازی است. که با اعمال یک به clk مقادیر I_0 تا I_3 به طور همزمان داخل فلیپ فلاپ Load می‌شوند. اما با یک اشکال به این صورت که به علت تاخیری که ممکن است در هر کدام از فلیپ فلاپ‌ها به وجود آید خروجی‌ها به صورت موازی بار نشوند لذا سیستم جدید را پیشنهاد می‌کنیم.



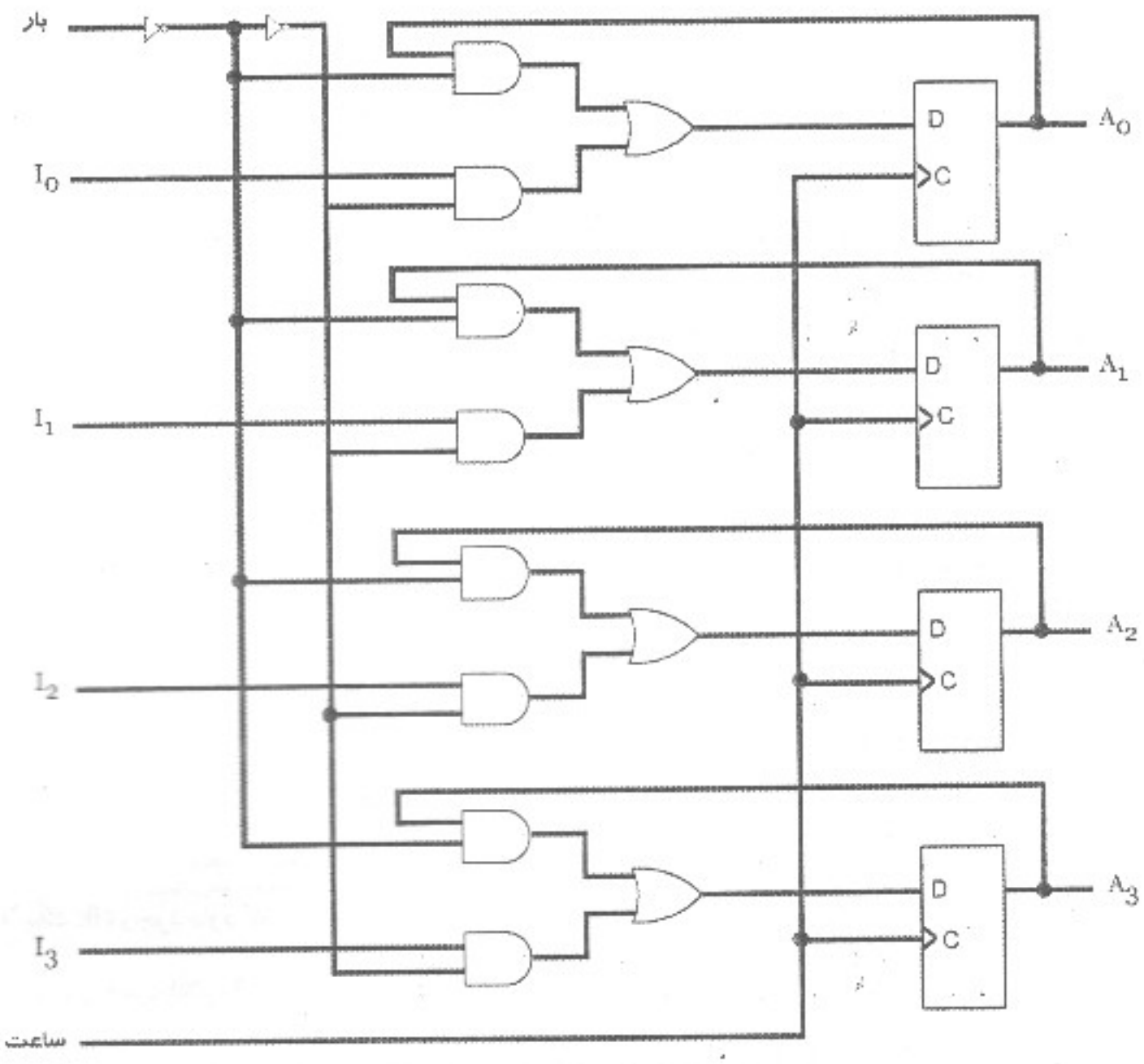
ورودی کنترل بار:

در واقع می‌خواهیم سیستم بار شدن موازی را پیاده سازی کنیم (بایک clk مشترک)

اما برای اعمال ورودی به شکل همزمان به فلیپ فلاپ‌ها از یک گیت AND (یا هر گیت کنترلی دیگرNOR, NAND, OR) برای کنترل بار استفاده می‌کنیم.

در خروجی همه فلیپ فلاپ‌های رجیستر خروجی یک AND وارد شده که ورودی AND یکی clk است و دیگری I_0 تا I_3 است.

شکل زیر بار شدن موازی با استفاده از گیت‌های کنترلی ورودی D فلیپ فلاپ‌ها خروجی OR است ورودی OR خروجی دو AND است. ورودی ANDها یک بار از خط load و هم یک بار از متمم خط load به همراه خروجی مرحله قبل فلیپ فلاپ با ورودی‌ها I_0 تا I_3 استفاده می‌کند.



ثبات ٤ بیتی با بار شدن موازی

$$\begin{cases} load = 0 \rightarrow \text{خروجی قبل} \rightarrow A_3A_2A_1A_0 \\ load = 1 \rightarrow \text{ورودی جدید} \rightarrow I_3I_2I_1I_0 \end{cases}$$

شیفت رجیستر

ثبات یا رجیستری که بتواند اطلاعات داخل خود را به سمت راست یا چپ شیفت (انتقال) دهد شیفت رجیستر نامیده می‌شود.

ثباتی که قابلیت انتقال اطلاعات درونش را به سمت چپ یا راست را داشته باشد شیفت رجیستر گفته می‌شود. Right shift Register اطلاعات درونش را به شیفت رجیستر معمولی این امکان را دارد که با هر بار عوض شدن clock یک شیفت انجام دهد. در این نوع شیفت رجیستر به ازای هر یک بیت اطلاعات که شیفت انجام می‌شود از یک سمت Register سمت راست ترین بیت خارج شده و از سمت چپ آن یک صفحه وارد می‌شود اما در شیفت رجیستر دورانی بیت خارج شده از سمت مخالف دوباره به داخل ثبات بر می‌گردد به این ترتیب در یک

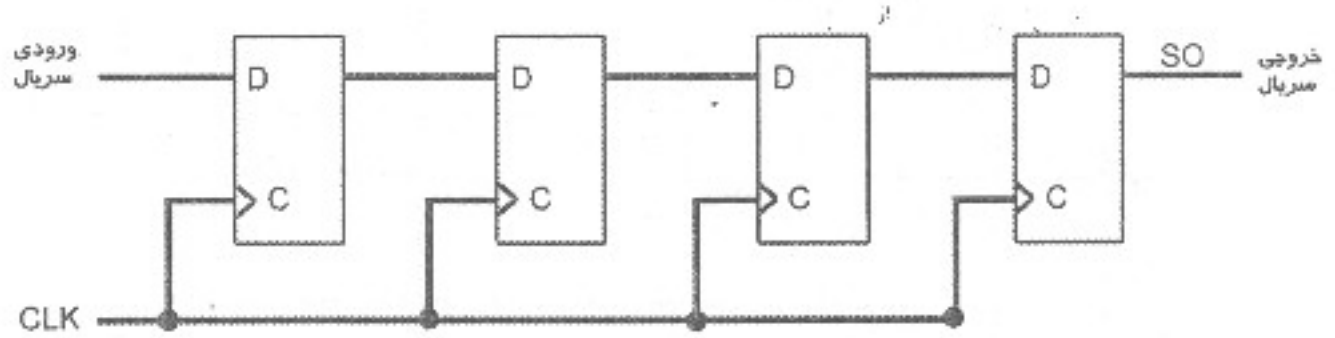
شیفت رجیستر n بیتی معمولی با n بار clock اطلاعات اصلی ثبات پاک شده و به جای آن صفر می‌نشیند اما در یک شیفت رجیستر دورانی n بیتی با n تا clock اطلاعات اولیه ثبات دوباره Load می‌شود در یک شیفت رجیستر معمولی یک بیت شیفت به چپ (یعنی وارد شدن یک بیت صفر از سمت راست) معادل با ضرب کردن عدد داخل شیفت رجیستر در عدد 2 می‌باشد و در یک Right shift یک عمل شیفت دادن معادل تقسیم عدد شیفت رجیستر بر 2 می‌شود.

یازده | 1 | 0 | 1 | 1 |

sh − R → 0	0 1 0 1	1	سمت راست	5
sh − R → 1	1 1 0 1	1	سمت راست	13
sh − L → 0	0 1 1 0	1	سمت چپ	6
sh − L → 1	0 1 1 1	1	سمت چپ	7

ساختار یک شیفت رجیستر

زنجیره‌ای از فلیپ فلاپ‌ها که خروجی یکی ورودی دیگری (بعدی) است



شیفت رجیستر 4 بیتی

در شکل بالا یک clk وجود دارد که برای همه فلیپ فلاپ‌ها مشترک است در فلیپ فلاپ اول یک ورودی SI وارد شده یا ورودی سریال وارد شده است. ورودی فلیپ فلاپ اول به Q_1 تبدیل شده که همان ورودی فلیپ فلاپ دوم استرف به همین ترتیب خروجی هر کدام ورودی بعدی می‌شود و خروجی آخر SO یا خروجی سریال می‌شود.

اگر یک SI به شکل 1101 داشته باشیم و وارد شیفت رجیستر شود و فلیپ فلاپ‌ها را هم D_4, D_3, D_2, D_1 نام‌گذاری کنیم و در مرحله اول فلیپ فلاپ‌ها مقادیر دلخواه قبلی را دارد اگر Clear داشتیم یک‌بار Reset می‌کنیم که همه مقادیر صفر شود.

وقتی clk اول وارد شود به درون ثبات انتقال پیدا می‌کند یعنی 1 وارد D_1 می‌شود و بقیه همان مقادیر قبلی را دارند وقتی clk دوم وارد شود صفر وارد D_1 می‌شود و 1 به D_2 می‌رود و بقیه همان مقادیر قبل را دارند وقتی clk سوم وارد شود 1 به D_1 صفر به D_2 و 1 به D_3 می‌رود و D_4 مقدار قبلی خود را دارد و وقتی clk چهارم را می‌زنیم کل مقادیر SI که 1101 است به ترتیب از چپ به راست در فلیپ فلاپ‌های D_1 تا D_4 وارد می‌شوند.

بدین ترتیب بعد از 4 clk اطلاعات وارد رجیستر شد حال عمل شیفت انجام می‌شود یک clk که زده شود 1 از D_4 خارج می‌شود خانه‌های D_1, D_2, D_3 اطلاعات خود را به خانه‌های بعدی می دهند و D_1 صفر می‌شود عدد 1 که از D_4 خارج می‌شود جایی ذخیره نمی‌شود اگر اطلاعات n بیتی را به وسیله یک شیفت رجیستر معمولی n بار شیفت دهیم مقادیر رجیسترها تماماً صفر می‌شود.

شیفت رجیستر دورانی

اگر اطلاعات 4 بیتی 0011 را داشته باشیم با استفاده از یک شیفت رجیستر دورانی بخواهیم آن را به راست شیفت دهیم به صورت زیر عمل می‌کنیم در یک شیفت رجیستر دورانی n بیتی اگر n بار شیفت دهیم به حالت اولیه می‌رسیم.

اگر بخواهیم شیفت رجیستر دورانی تولید شود از خروجی سریال به ورودی سریال وصل می‌کنیم.

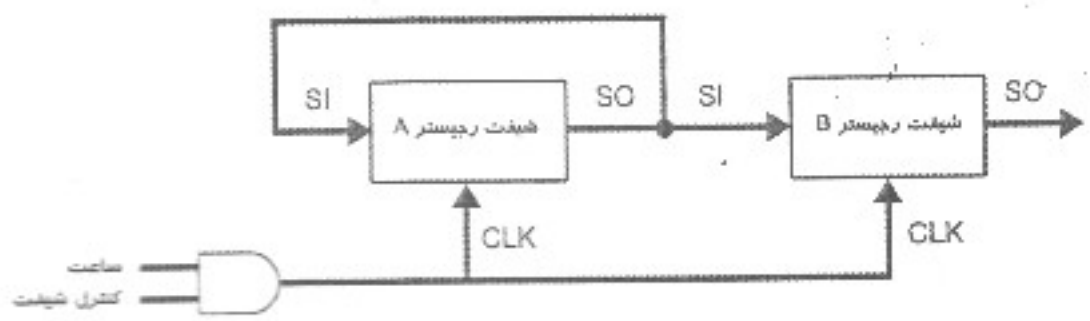
0 0 1 1
1 0 0 1

انتقال سریال

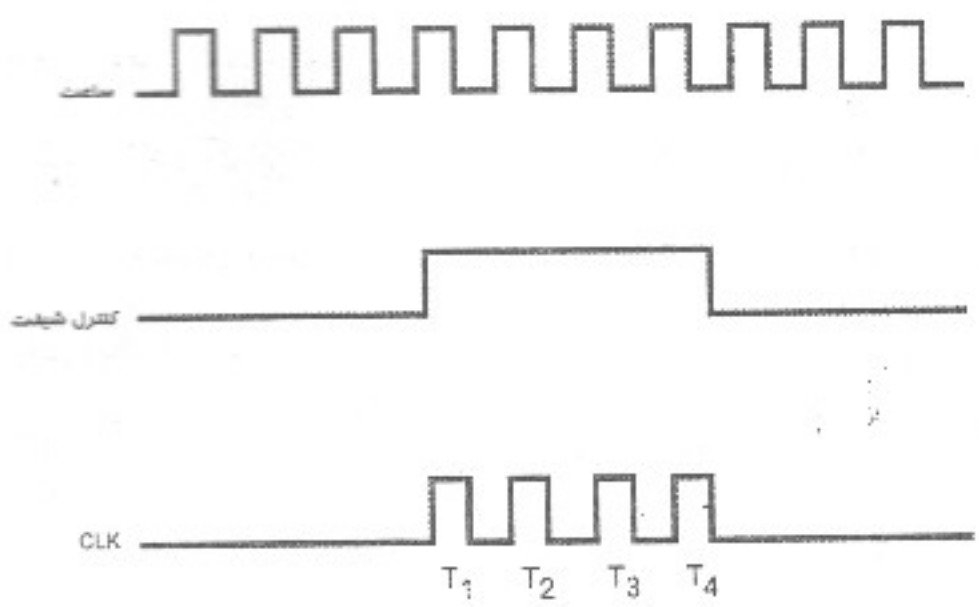
در صورتی که اگر اطلاعات به صورت سریال نباشد و موازی باشد به جای یک ورودی و یک خروجی 4 ورودی و 4 خروجی داریم.

برای اطلاعات 4 بیتی یک شیفت رجیستر اگر انتقال سریال انجام شود باید 4 بار کلاک زده شود ولی اگر انتقال موازی انجام شود با یک کلاک می‌توان اطلاعات را منتقل کرد.

عملیات در سیستم‌های دیجیتال معمولاً به صورت موازی انجام می‌شود اما هم اکنون نیز در بعضی موارد و در بعضی سیستم‌ها از انتقال سریال استفاده می‌شود. سرعت عملیات موازی به مراتب بیشتر از سرعت عملیات سریال است اما مشکل عملیات موازی در پیچیده بودن سخت افزار و همچنین حجم بالای سخت افزار مورد نیاز است. در صورتی که در انتقال سریال از سخت افزار ساده‌تر و کم‌تر استفاده می‌شود.



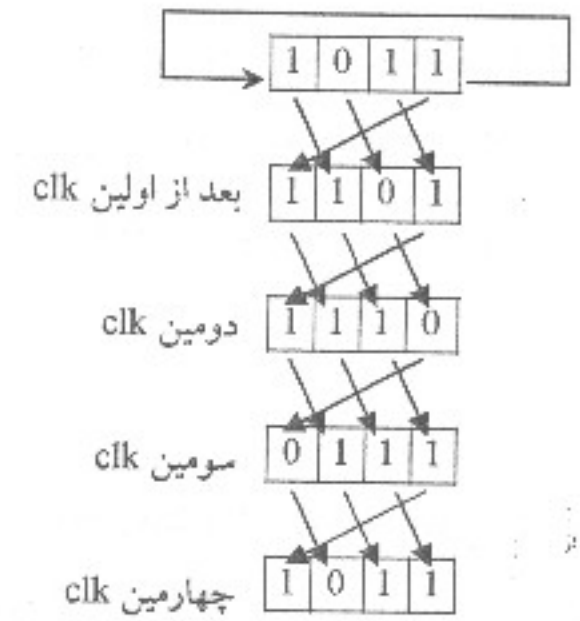
نمودار بلوکی



نمودار زمانی

انتقال سریال از ثبات A به ثبات B

شیفت رجیستر راست (دورانی)



در شکل بالا دو شیفت رجیستر B,A داریم که می‌خواهیم به کمک انتقال سریال محتویات A را وارد B می‌کنیم که هر کدام از شیفت رجیسترها یک SI (ورودی سریال) و یک SO (سریال خروجی) دارند.یک کلاک مشترک هم دارند A مقدار 1011 را وارد B که مقدارش برای ما مهم نیست (B = - - - -) می‌کند.

نحوه کار: با اولین کلاک چون B,A راست (Right) شیفت رجیستر هستند. A سمت راست‌ترین بیت اطلاعاتش را از طریق SO خود به B,SI انتقال می‌دهد و وارد سمت چپ‌ترین بیت B می‌شود بعد از یک کلاک A برابر 0101 خواهد بود و B هم مقدار - - - 1 را خواهد داشت.

$A = 1011 \rightarrow A = 0 = 0101$

$B = - - - - \rightarrow B = 1 - - -$

بعد از 4 کلاک اطلاعات A را به B منتقل می‌کنیم مقدار A=0000 و B=1011 خواهد بود و این حالت انتقال سریال انجام شده ولی محتویات A را از دست داده‌ایم برای این‌که مقدار A را از دست ندهیم می‌توانیم از یک شیفت رجیستر دورانی استفاده کنیم، یعنی از SO شیفت رجیستر A به A,SI وصل کنیم به این ترتیب بعد از 4 کلاک علاوه بر این‌که محتویات A را به B منتقل کرده‌ایم A نیز مقدار قبلی خود را دارد.

اما این شیفت رجیستر یک مشکل دارد و آن این است که دائم درحال عملیات شیفت دادن هستیم زیرا کلاک پالس دائماً فعال است. برای حل این مشکل از یک خط کنترل برای شیفت رجیستر استفاده می‌کنیم که شیفت رجیستر را کنترل کند. از یک AND استفاه می‌کنیم که یک ورودی آن خط clk است و ورودی دیگر خط کنترل شیفت است.

اگر کنترل شیفت 1 باشد، خط clk فعال است. اگر کنترل شیفت 0 باشد خط clk غیرفعال است و عمل شیفت انجام نمی‌شود

چند اصطلاح:

word time **(زمان کلمه)**: مدت زمانی که لازم است تا محتوی یک شیفت رجیستر به صورت کامل شیفت داده شود.

(bit time): مدت زمان بین دو کلاک پالس

انواع شیفت رجیسترها:

۱ـ SISO: ورودی و خروجی هر دو به صورت سریال می‌باشند.

(Serial Input , serial output)

۲ـ SIPO: ورودی به صورت سریال و خروجی به صورت موازی است.

(Serial Input , parallel output)

۳ـ PISO: ورودی به صورت موازی و خروجی به صورت سریال می‌باشد.

۴ـ PIPO: ورودی و خروجی هر دو به صورت موازی می‌باشند. (بهترین نوع شیفت رجیسترها)

شیفت رجیسترهای دوجهته با امکان بار شدن موازی (شیفت رجیسترهای یونیورسال)

از یک شیفت رجیستر می‌توانیم برای تبدیل داده سریال به موازی یا داده موازی به سریال استفاده کنیم فقط در این نوع شیفت رجیسترها باید به تمامی خروجی‌های فلیپ فلاپ دسترسی داشته باشیم.

کامل‌ترین شیفت رجیستر دارای خصوصیات زیر است:

۱ـ کنترل پاک برای پاک کردن ثبات به (0)

۲ـ ورودی ساعت برای همزمانی اعمال

۳ـ کنترل همه‌جایی به راست (شیفت به راست) برای فعال کردن عمل‌جابه‌جایی به راست و خطوط ورودی و خروجی سریال مربوط به جابه‌جایی راست

۴ـ کنترل جابه‌جایی به چپ (شیفت به چپ) برای فعال کردن عمل جابه‌جایی به چپ و خطوط ورودی و خروجی سریال مربوط به جابه‌جایی به چپ

۵ـ یک کنترل باز کردن موازی برای فعال کردن انتقال موازی و n خط ورودی مربوط به انتقال موازی

۶ـ n خط خروجی

۷ـ حالت کنترلی که علی‌رغم وجود پالس ساعت اطلاعات را در ثبات بدون تغییر نگه می‌دارد.

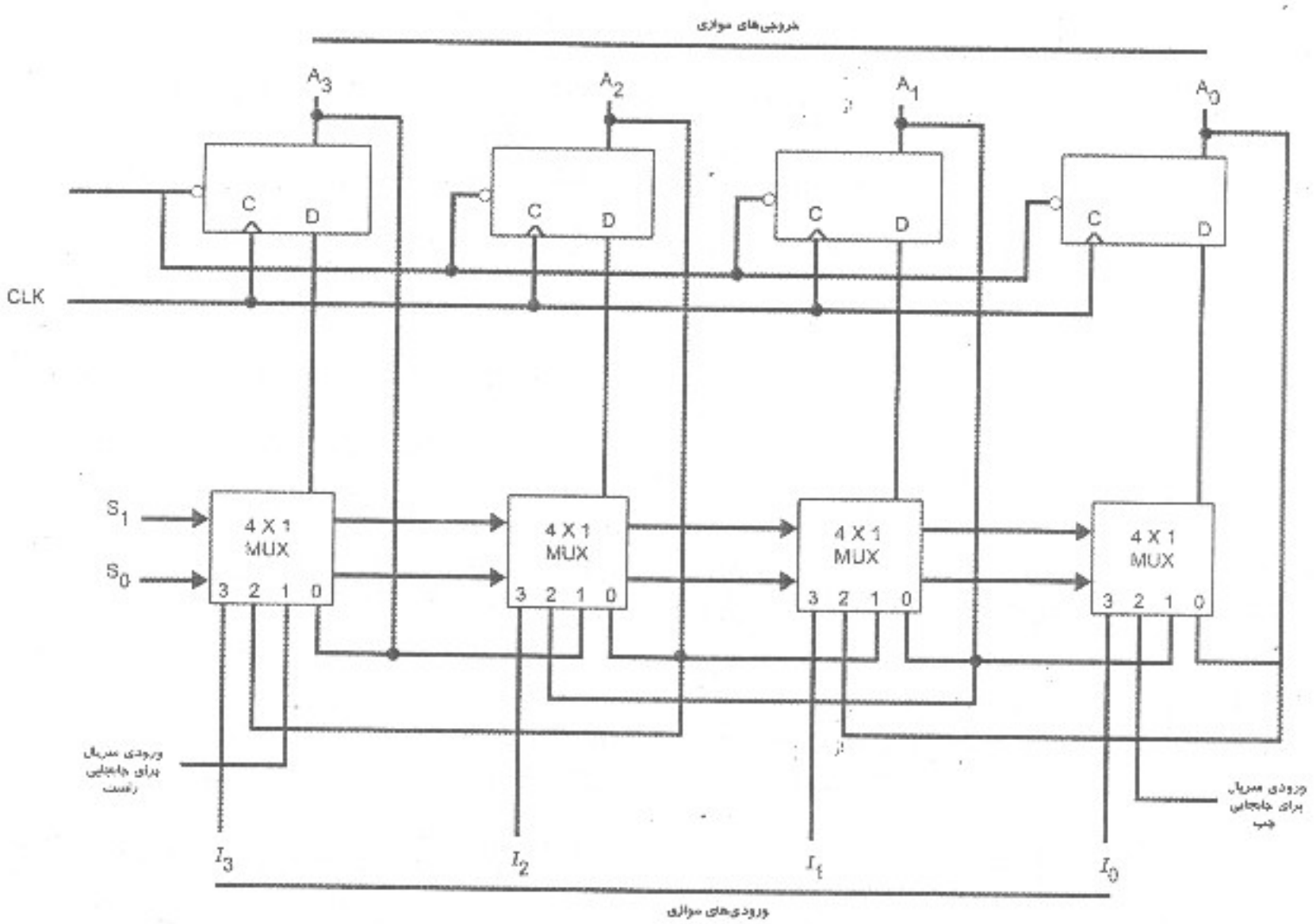
در شکل (۵-۵) 4 مولتی پلکسر داریم (1×4) و 2 خط select ,4 ورودی و خروجی خط‌های S_1,S_0,select 4 مولتی پلکسر را به هم وصل می‌کند، یعنی اگر یک select انجام دهیم و (0,0) به ورودی بدهیم خط صفر برای همه فعال می‌شود.

همچنین در شکل بالا 4 فلیپ فلاپ داریم که از نوع D فلیپ فلاپ هستند که یک پایه clear دارند و یک پایه clk دارند که در همه مشترک هستند.

خروجی مولتی پلکسر ورودی فلیپ فلاپ است.

اگر S_1,S_0 هر دو صفر باشد (خط select (0,0) است) خط صفر فعال می‌شود (برای تمام مولتی پلکسرها فعال می‌شود) در این‌حالت خروجی صفر وارد D فلیپ فلاپ می‌شود. یعنی یک بافر ساختیم که اطلاعات را از مولتی پلکسر به خروجی می‌برد و بعد باز به مولتی پلکسر باز می‌گرداند. (A_0 خروجی فلیپ فلاپ که وارد خط صفر مولتی پلکسر شده)

اگر (0,1) را در خط select قرار دهیم، خط 1 فعال می‌شود شیفت به راست داده می‌شود حال اگر مقدار (1,0) به خط select بدهیم شیفت به چپ داده می‌شود و خط 2 فعال می‌شود اگر A_1 به پایه 2 مولتی پلکسر 3 وصل می‌شود شیفت رجیستر دورانی خواهد شد اگر (1,1) را به خطوط select بدهیم خط 3 فعال می‌شود در این حالت ورودی‌های موازی وارد می‌شود خروجی‌های موازی خارج می‌شود در این حالت شیفت رجیستر با بار شدن موازی می گویند.



شیفت رجیستر یونیورسال 4 بیتی

جدول عملکرد شیفت رجیستر یونیورسال 4 بیتی با توجه به خطوط Select

عملکرد	S_0	S_1
بلاتغییر	0	0
شیفت راست	1	0
شیفت چپ	0	1
بار کردن موازی	1	1

مثال: محتویات شیفت رجیستر 16 بیتی A برابر A∅B2 می‌باشد پس از 3 شیفت به راست با ورودی یک و دو شیفت به چپ حاصل آن را در شیفت رجیستر B قرار می‌دهیم محتویات B چیست؟

12 بیتی 1010 , 0000 , 1011 , 0010

3 راست 1111 , 0100 , 0001 , 0110

2 چپ 1101 , 0000 , 0101 , 1000 → D∅58

جمع سریال

یک شیفت رجیستر به نام A داریم که عدد $A_{n-1}A_{n-2}....A_1A_0$ در آن وجود دارد یک شیفت رجیستر به نام B داریم که عدد $B_{n-1}B_{n-2}.....B_0$ در آن وجود دارد می‌خواهیم این دو عدد را با هم جمع کنیم برای این کار یک مدار طراحی می‌کنیم که یک انتقال سریال انجام می‌دهد که SO_B, SO_A به f وصل می‌شوند و f یک خروجی به نام S (حاصل جمع) و یک خروجی به نام C (carry) دارد. علاوه بر این‌ها یک C_{in} وارد f می‌شود. از طریق SI مقدار B.A در شیفت رجیسترها Load می‌شود حال می‌خواهیم عمل جمع را انجام دهیم باید یک clk مشترک برای شیفت رجیسترها قرار دهیم که وقتی کلاک یک باشد دو شیفت رجیستر یک بیت، شیفت به سمت راست می‌دهند و یک بیت اطلاعات آن‌ها از طریق SO وارد f می‌شوند یعنی B_0, A_0 وارد f می‌شوند و عمل جمع انجام می‌شود یک مقدار carry دارد و حاصل جمع S_0 است که با carry جمع می‌شود و S_0 حقیقی را با یک Cout تشکیل می‌دهد.

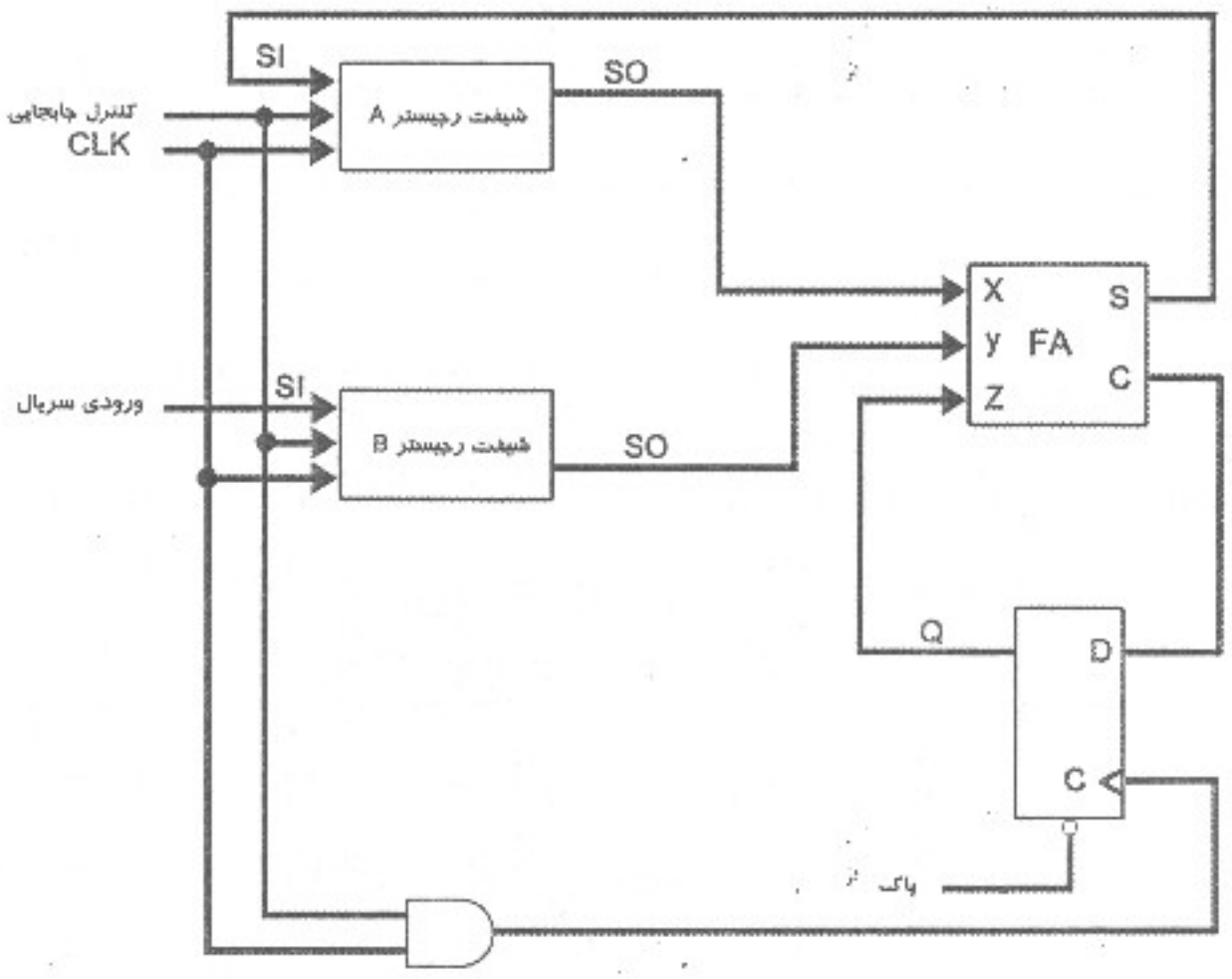
وقتی n بار کلاک بزنیم و کل جمع‌های دو به دو انجام شود یک شیفت به راست انجام می‌دهد. در انتها یک C_{out} ,S داریم.

تابع f می‌تواند یک Full Adder باشد که y,x ورودی آن است به اضافه یک ورودی به نام z که همان carry ورودی است و یک خروجی به اسم S و یک خروجی به نام C_{out} دارد.

برای این‌که carry را وارد کنیم از یک بافر استفاده می‌کنیم. برای بافر از یک D فلیپ فلاپ استفاده می‌کنیم این شکل که C_{out} (کری خروجی) وارد ورودی D فلیپ فلاپ می‌شود و از خروجی فلیپ فلاپ وارد تابع f می‌شود که همان Full Adder است که عمل جمع را انجام می‌دهد.

در شکل (۶-۵) دو شیفت رجیستر B,A داریم که به جای تابع f از یک Full Adder استفاده کردیم و مقادیر y,x از طریق رجیسترهای B,A وارد Full Adder می‌شوند.

Full Adder یک S خروجی دارد که وارد رجیستر A شده که با این کار اطلاعات A بعد از عمل جمع از بین می‌رود. خروجی C وارد D فلیپ فلاپ می‌شود .



جمع کننده سریال

بعد از کلاک خوردن از طریق خروجی فلیپ فلاپ وارد Z که همان Full Adder, C_{out} است می‌شود clk فلیپ فلاپ از AND خط کنترل شیفت و clk وجود آمده است.

مثال: یک جمع کننده سریال طراحی کنید که برای محاسبه رقم نقلی از R.S.F.F استفاده کند؟

C_{in}	x	y	C_{out}	S	S_c	R_c
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	0	X	0

$S_c = xy \qquad R_c = x'y' \qquad S = C_{in} \oplus x \oplus y$

تفاوت جمع کننده سریال با جمع کننده موازی در چیست؟

۱ـ جمع کننده موازی باید از ثبات‌هایی استفاده کند که امکان Load موازی داشته باشد که این به نوعی سخت افزار پیچیده‌تر است.

۲ـ در جمع‌کننده موازی باید به تعداد بیت‌ها شیفت رجیستر داشته باشیم ولی در جمع‌کننده سری تماماً با دو شیفت رجیستر می‌توانیم عمل جمع را انجام دهیم.

مثال: مداری طراحی کنید که به کمک jk فلیپ فلاپ جمع کننده سریال را به وجود آورد.

حالت فعلی	ورودی‌ها		حالت بعدی	خروجی	ورودی فلیپ فلاپ	
C_{in}	x	y	C_{out}	S	J_C	K_C
0	0	0	0	0	0	X
0	0	1	0	1	0	0
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	x	1
1	0	1	1	0	x	0
1	1	0	1	0	x	0
1	1	1	1	0	x	0

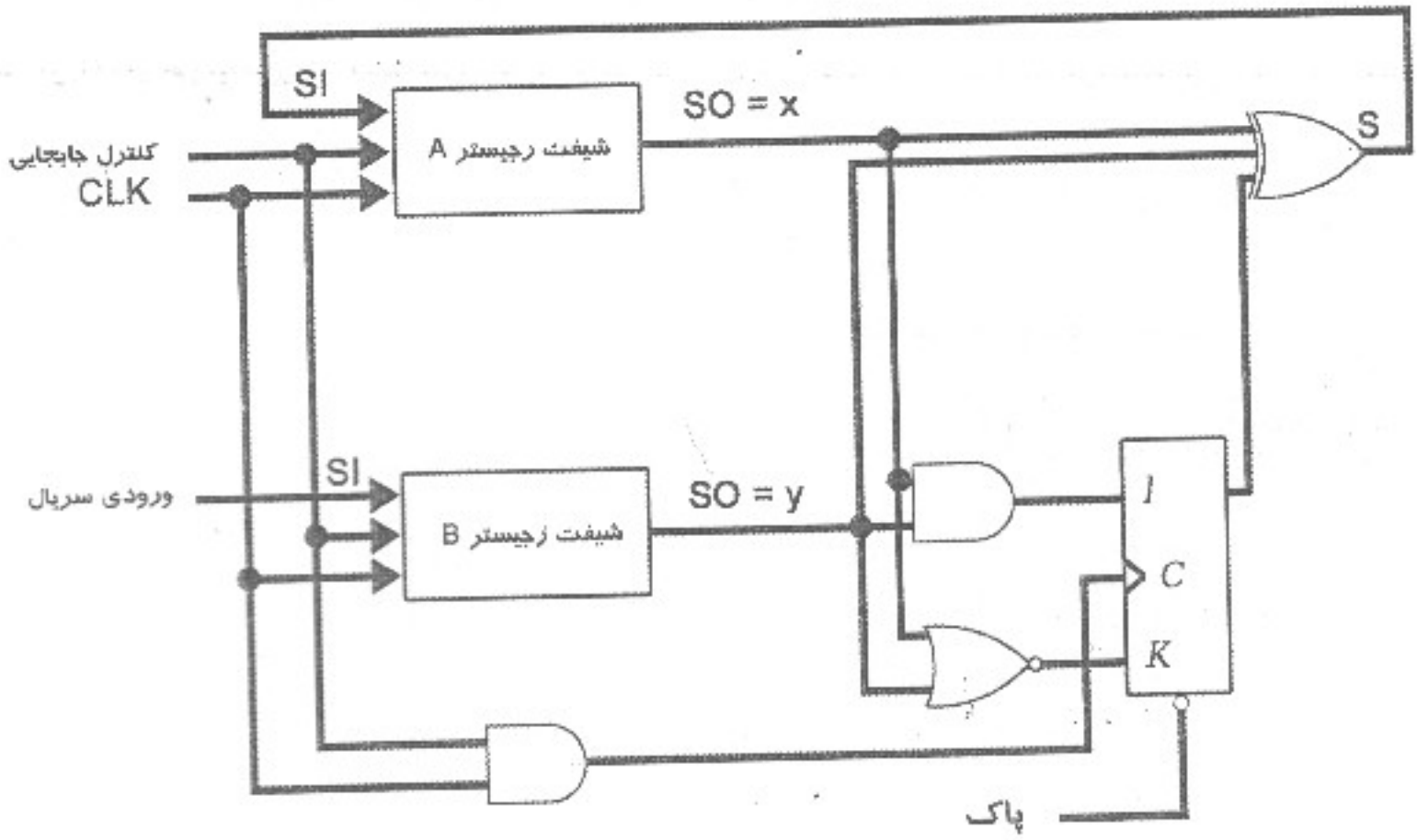
y,x را با هم جمع می‌کنیم حاصل را با C_{in} جمع می‌کنیم.

$j_c = xy$

حاصل جمع S و کری در Cout قرار می‌دهیم.

$k_c = x'y'$

$S = C_{in} \oplus x \oplus y$



فرم دوم یک جمع کننده سریال

در شکل بالا که فرم دوم یک جمع‌کننده سریال است ورودی j فلیپ فلاپ خروجی AND است که ورودی‌های آن x و y هستند که x خروجی سریال (SO) شیفت رجیستر A است و y خروجی سریال (SO) شیفت رجیستر B است. ورودی K فلیپ فلاپ خروجی یک NOR است که ورودی‌های آن x و y هستند که x خروجی سریال (SO) شیفت رجیستر A است و y خروجی سریال (SO) شیفت رجیستر B می‌باشد.

شمارنده موج گونه دودویی: (آسنکرون یا غیرهم زمان)

معمولا اولین فلیپ فلاپ یک کلاک پالس دارد در طراحی فلیپ فلاپ‌ها ورودی کلاک از خروجی فلیپ فلاپ قبل به دست می‌آید. به این معنی که خروجی Q فلیپ فلاپ وارد ورودی کلاک فلیپ فلاپ بعدی می‌شود.

در شمارنده آسنکرون حالاتی به نام حالات گذار است که بین دو حالت اصلی قرار می‌گیرند و در واقع مسیر بین یک حالت اصلی با حالت اصلی دیگر هستند.

مثلا وقتی می‌خواهیم از 1010 تا 1011 بشماریم حالات ناخواسته‌ای در این بین وجود دارند که راه رسیدن به عدد بعدی را به وجود می‌آورند به این حالات که حالت اصلی نیستند حالات گذار گفته می‌شود.

در شمارنده آسنکرون Active low یا Active High بودن کلاک پالس بسیار مهم است اگر ورودی کلاک Active Low باشد شمارش به شکل صعودی است و اگر کلاک به صورت Active High باشد شمارش آن به شکل نزولی است.

برای بوجود آوردن شمارنده آسنکرون صعودی یا نزولی یکی از راه‌ها تغییر دادن شکل کلاک است.

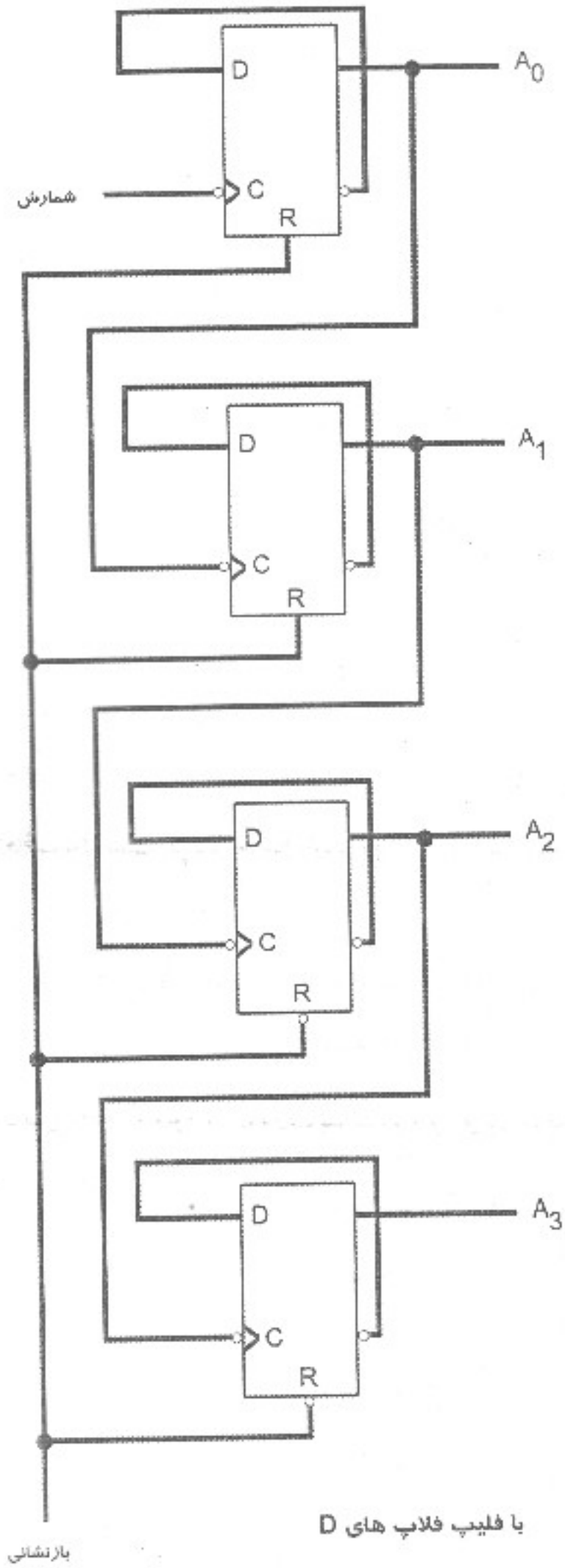
در T فلیپ فلاپ هنگامی که ورودی صفر باشد خروجی همان خروجی مرحله قبل است.

ورودی همه فلیپ فلاپ‌ها یک است هر زمان فلیپ فلاپ فعال شود باعث می‌شود خروجی مرحله قبل متمم شود ولی خروجی مرحله قبل که فعال شود کلاک پالس همه Active Low هستند و با هر گذرا باعث فعال شدن می‌شود.

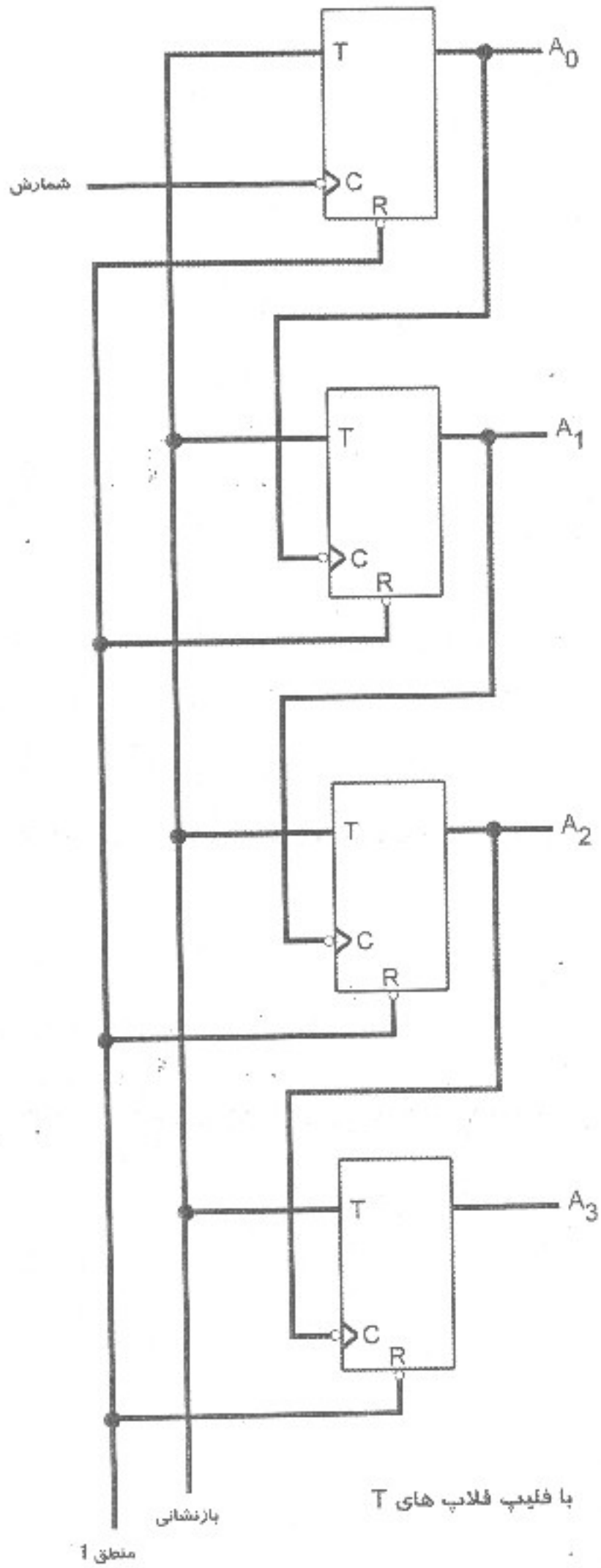
تنها فلیپ فلاپ اول است که clk حاصل است وقتی clk=0 باشد خروجی A_0 متمم می‌شود یعنی 1 اعمال می‌شود به دومین فلیپ فلاپ که فعال نمی‌شود پس $A_1 = A_2 = A_3$ می‌ماند. در کلاک بعدی T=1 است خروجی مرحله قبل متمم می‌شود $A_0 = 0$ بعدی 1 می‌شود پس دومی فعال نمی‌شود و بقیه هم صفر است.

با کلاک بعدی $A_0 = 1$ می‌شود وارد بعدی می‌شود و بدون تغییر باقی می‌ماند بعدی هم $A_1 = 1$ است و $A_2 = A_3$ خواهد بود.

با کلاک بعدی $A_0 = 0$ وارد فلیپ فلاپ بعدی می‌شود و آن را فعال می‌کند $A_1 = 0$ وارد می‌شود $A_1 = 1$ وارد بعدی می‌شود و بقیه مانند حالت فعلی می‌ماند.



با فلیپ فلاپ های D



با فلیپ فلاپ های T

شمارنده موج گونه دودویی 4 بیتی

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
⋮	⋮	⋮	⋮

شمارنده BCD موج گونه: (سنکرون)

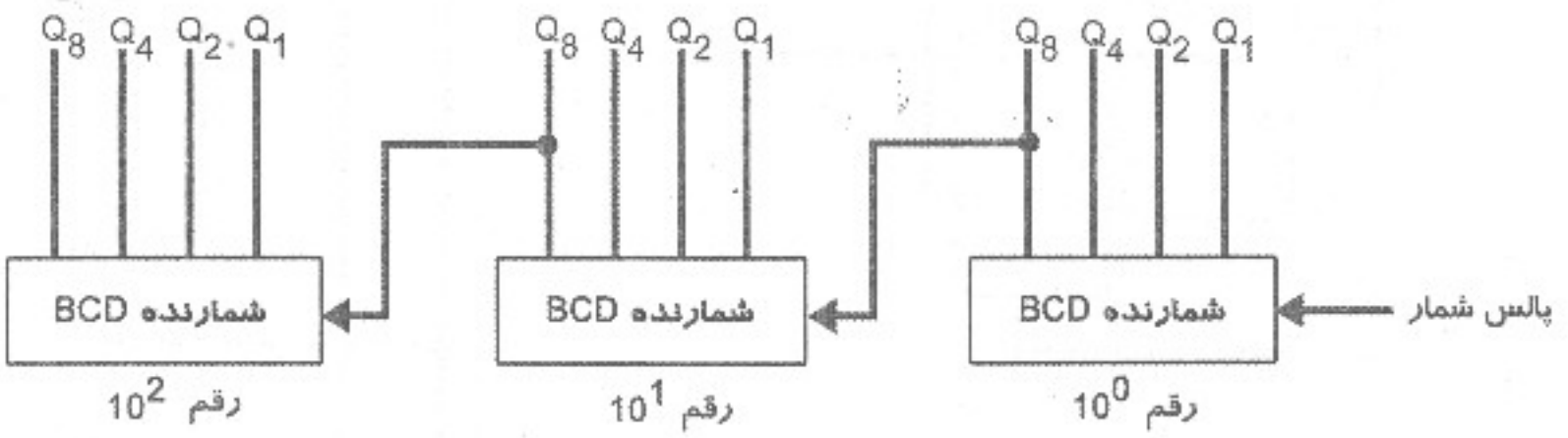
دیاگرام حالت یک شمارنده BCDکه از (0000) شروع شده تا به (1001) رسیده را در شکل زیر ملاحظه می‌کنید، در این‌جا فقط حالات اصلی نوشته شده یعنی حالات گذرا مشخص نشده اند.

در شکل صفحه بعد یک شمارنده موج گونه BCD را با jk فلیپ فلاپ طراحی کردیم. یعنی در شمارنده آسنکرون به clk را به فلیپ فلاپ اول می‌دهیم یعنی کم ارزش‌ترین است.

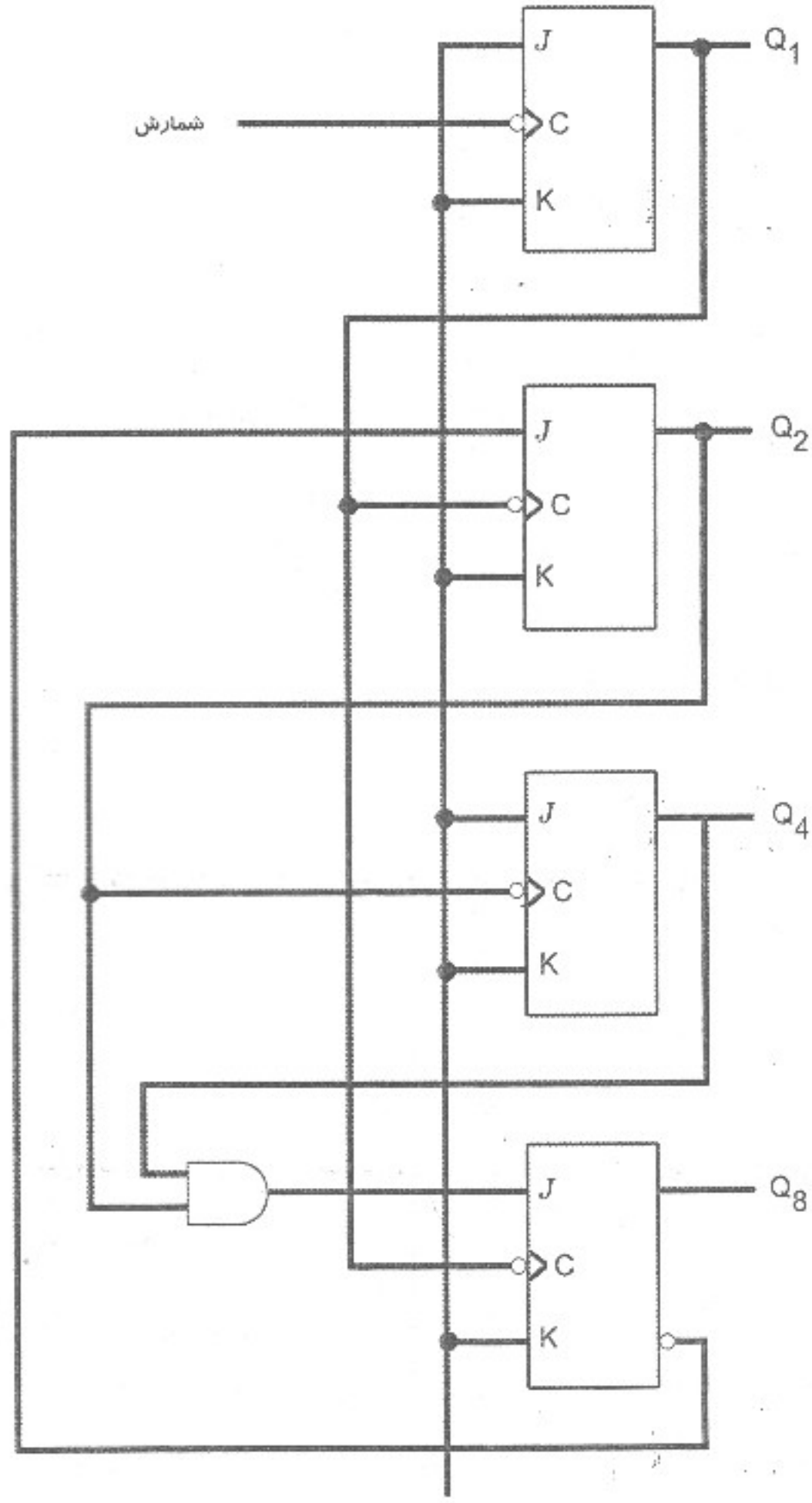
کلاک اول تنها کلاک مربوط به اولین فلیپ فلاپ را فعال می‌کند. کلاک دوم از خروجی اولی کلاک سوم از خروجی دوم و کلاک چهارم از خروجی سوم در فلیپ فلاپ اول و سوم j=k=1 است در دومی. y متمم خروجی فلیپ فلاپ چهارم (Q_8') است و k=1 در فلیپ فلاپ چهارم And , j , k=1 خروجی سومی (Q_4) و خروجی دومی (Q_2) است از 0000 شروع می‌کند یک کلاک در اولی باعث می‌شود jk فعال شوند چون j,k=1 است متمم می‌کنند با کلاک بعدی که از خروجی Q_1 مقدار 1 آمده فلیپ فلاپ‌های بعدی هم غیر فعال می‌مانند پس داریم 0001.

با کلاک بعدی $Q_1 = 0$ می‌شود مقدار صفر وارد فلیپ فلاپ بعدی می‌شود

فلیپ فلاپ دوم فعال می‌شود $Q_8' = 1$ وارد y می‌شود و k=1 پس متمم می‌کند یعنی $Q_2 = 1$ مقدار 1 وارد فلیپ فلاپ بعدی می‌شود پس سومی و چهارمی غیرفعال خواهد بود پس داریم 0010 و به همین ترتیب ادامه می‌دهد تا به 1001 برسد.



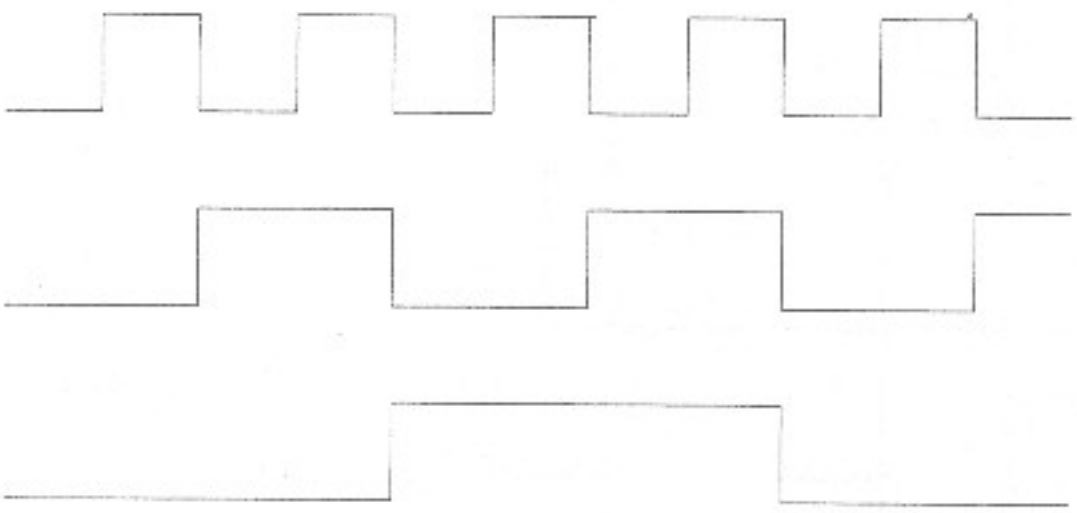
نمودار بلوکی یک شمارنده دهدهی BCD با سه دهه



شمارنده موج گونه BCD

Q_4	Q_3	Q_2	Q_1
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
⋮			⋮
1	0	0	0
1	0	0	1
0	0	0	0

طراحی خروجی فلیپ فلاپ یا شمارنده به کمک شکل کلاک پالس:



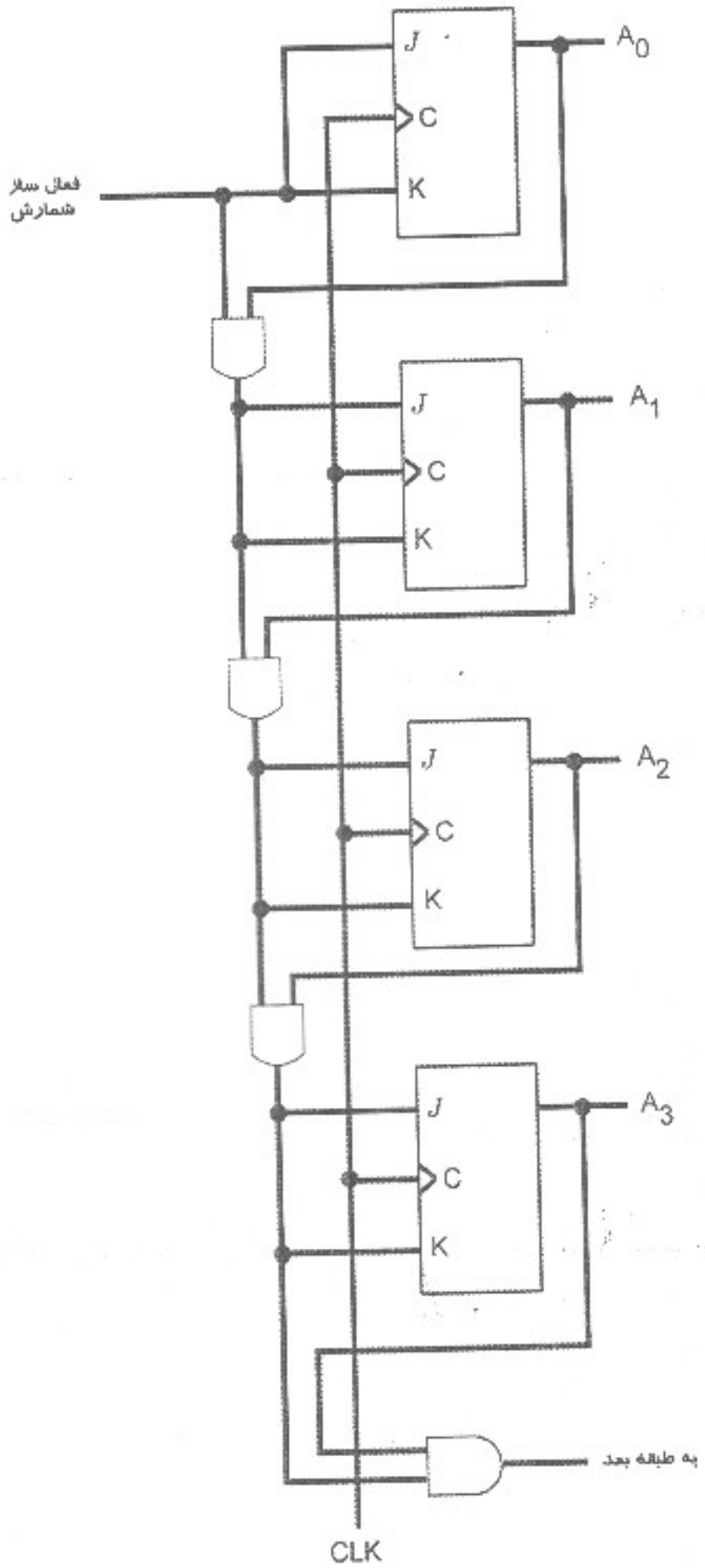
شمارنده همزمان سنکرون

تفاوت شمارنده همزمان با غیرهمزمان در این است که در شمارنده‌های همزمان یک کلاک مشترک وجود دارد که از کلاک مرکزی تامین می‌شود در صورتی که در شمارنده‌های غیر همزمان کلاک مرکزی تنها وارد اولین فلیپ فلاپ می‌شود و پایه کلاک بقیه فلیپ فلاپ‌ها از خروجی فلیپ فلاپ‌های قبلی تامین می‌شود.

شمارنده دودویی سنکرون

در یک شمارنده دودویی کلاک همه مشترک است اولین نکته در شمارنده همزمان این است که Active Low یا Active high بودن کلاک پالس تفاوتی در نحوه شمارش نمی‌کند. ورودی jk فلیپ فلاپ اول به خط فعال ساز شمارش وصل است یعنی وقتی فعال ساز شمارش 1 باشد در فلیپ فلاپ اول j=k=1 است و خروجی A_0 را متمم می‌کند.

اگر خروجی t=0 یک باشد باعث می‌شود که ورودی فلیپ فلاپ بعدی j=k=1 باشد و باعث متمم شدن خروجی A_1 می‌شود و در صورت صفر بودن هیچ تغییری در مقدار A_1 به وجود نمی‌آورد و به همین ترتیب تا آخر ادامه پیدا می‌کند.



شمارنده دودویی همزمان 4 بیتی

	A_3	A_2	A_1	A_0
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
⋮				

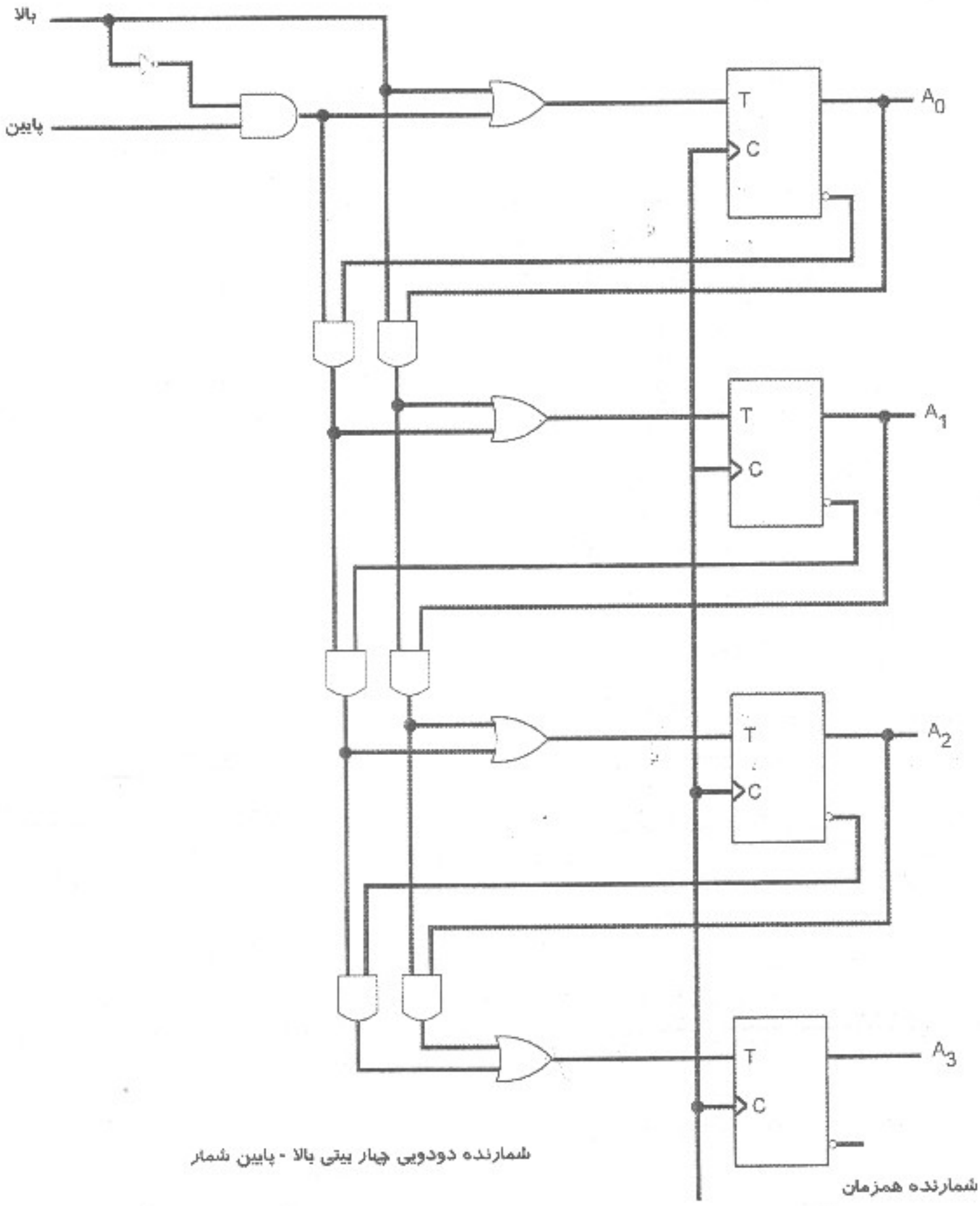
بالا، پایین شمار دودویی:

منظور از بالا و پایین شمار دودویی یک مداری است که توانایی دارد صعودی یا نزولی شمارش کند دو خط کنترل دارد که این خطوط تعیین می‌کنند که شمارنده صعودی شمارد یا نزولی یا شمارش را قطع کند. شکل(۱۳-۵) یک شمارنده دودویی چهار بیتی بالا ـ پایین شمار است که اگر خط بالا 1 باشد شمارنده صعودی می‌شمارد خط پائین می‌تواند صفر یا یک باشد و اگر خط بالا صفر باشد و خط پایین مقدارش یک باشد شمارنده به صورت نزولی می‌شمارد.

خط بالا صفر و خط پائین صفر باشد شمارش غیرفعال خواهد بود در طراحی شکل موردبحث از T فلیپ فلاپ استفاده شده در صورتی که بخواهیم از jk فلیپ فلاپ استفاده کنیم. تنها باید ورودی j و k را به هم وصل کنیم.

	پائین	بالا
نزولی	1	0
صعودی	x	1
شمارش غیر فعال	0	0

باید از OR که خروجی‌اش وارد فلیپ فلاپ می‌شود به صورت زیر استفاده کنیم



شمارنده دودویی چهار بیتی بالا - پایین شمار

شمارنده همزمان

	A_3	A_2	A_1	A_0
0	0	0	0	0
15	1	1	1	1
⋮				
5	0	1	0	1
4	0	1	0	0

شمارنده BCD

در فصل قبل طراحی شده

حالت فعلی				حالت بعدی				خروجی	ورودی فلیپ فلاپ			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

$TQ1 = 1$ $\quad TQ4 = Q2\,Q1$

$TQ2 = Q8\,Q1$ $\quad TQ8 = Q8\,Q1 + Q4\,Q2\,Q1$

$Y = Q_8Q1$

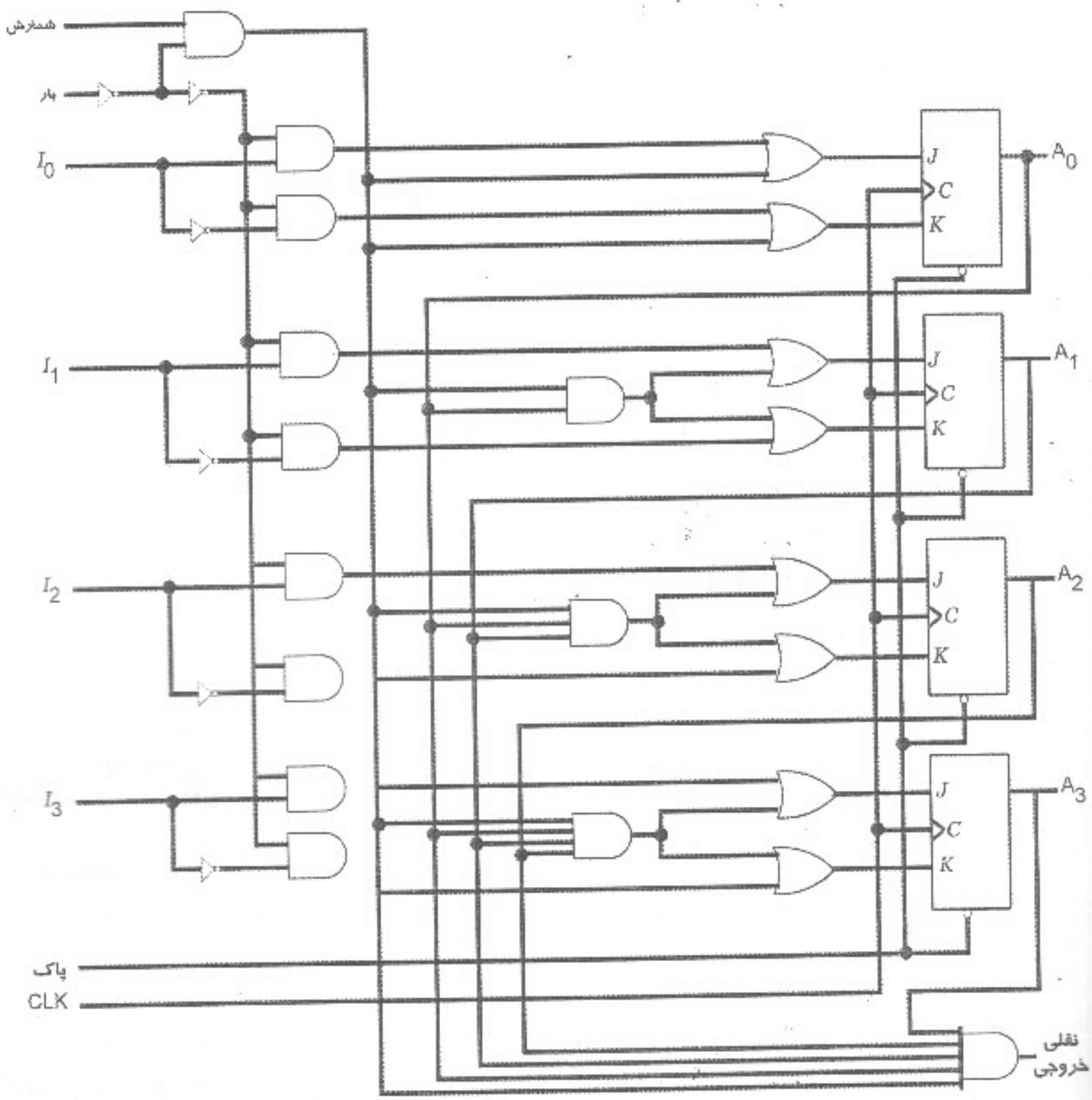
شمارنده دودویی با امکان بار شدن موازی:

در واقع مثل رجیستر که بار شدن موازی را انجام می‌داد عمل می‌کند. با این تفاوت که یک امکان دیگر به آن اضافه شده که کاملتر بودن است. یعنی علاوه بر این‌که شمارش را انجام می‌دهد این امکان را دارد که n بیت را درخود Load کند.

با Load موازی این امکان را داریم که یک عدد را برای شروع شمارش به آن بدهیم در شکل(۱۴-۵) از ورودی‌های I_0 تا I_3 به صورت I و I' استفاده می‌کنیم به این صورت که در AND اول خود I به همراه خط Load ورودی می‌شود و در AND هم I' به همراه خط Load وارد می‌شوند. خط Load برابر یک است محل مقدار I از And عبور می‌کند وارد OR می‌شود ورودی دیگری OR خط شمارش است در شمارنده یا Load موازی ورودی داریم یکی خط Load و دیگری کنترل شمارش است.

اگر خط Load برابر یک باشد عمل Load موازی انجام می‌شود اگر خط شمارش یک باشد آن‌گاه j=k=1 می‌شود خروجی And یک می‌شود وقتی خط شمارش یک و بار صفر باشد خروجی And صفر می‌شود.

تابع	شمارش	بار	CLK	پاک‌کننده
به O پاک می‌شود	x	x	x	0
ورودی بار شونده	x	1	↑	1
شمارش حالت دودویی بعدی	1	0	↑	1
بلاتغییر	0	0	↑	1

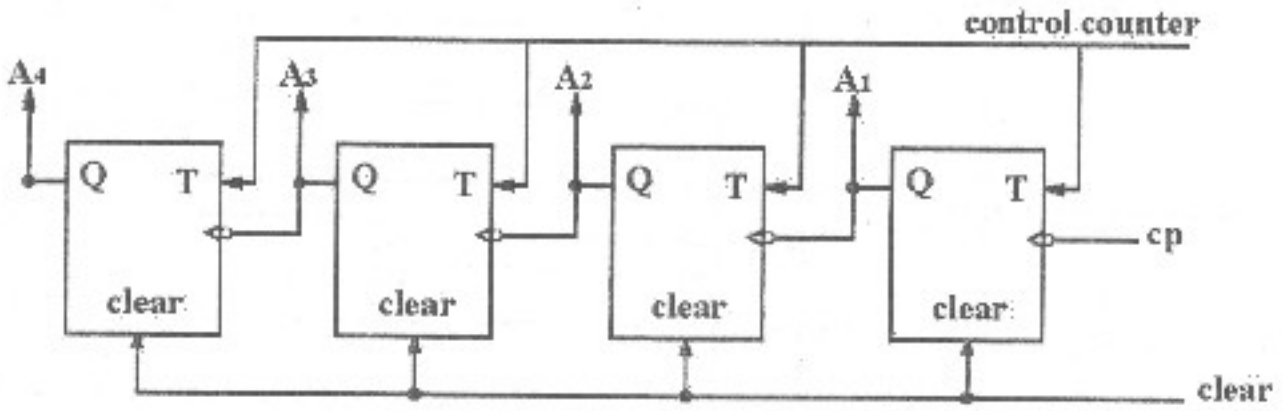


شمارنده دودویی 4 بیتی با امکان شدن موازی

شمارندهٔ سنکرون ماژول ـ (MOD – N):

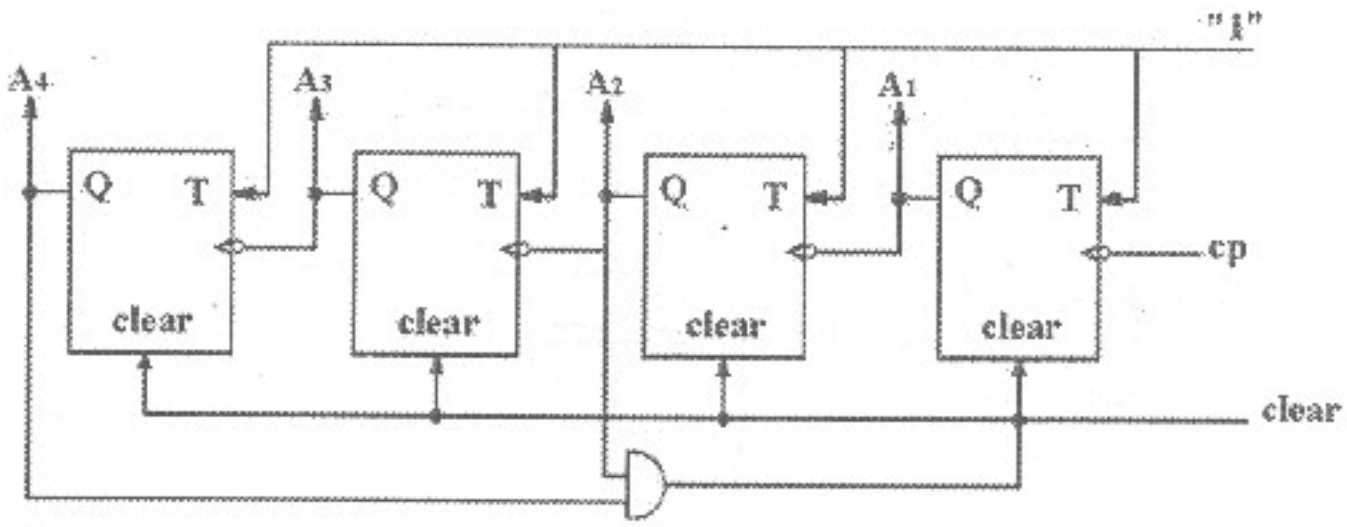
اگر یک شمارنده بتواند از صفر تا N – 1 را بشمارد آن شمارنده را ماژول N می‌نامیم. به عنوان مثال اگر شمارنده ماژول 8 داشته باشیم این شمارنده از صفر تا 7 را می‌تواند بشمارد.

شماتردهٔ زیر یک شمارندهٔ MOD – 15 می‌باشد.



15 – MOD

و در ضمن مدار زیر نیز یک شمارندهٔ سنکرون MOD – 10 می‌باشد یعنی از صفر تا 9 را می‌شمارد و دوباره بارگذاری از نو می‌شود.



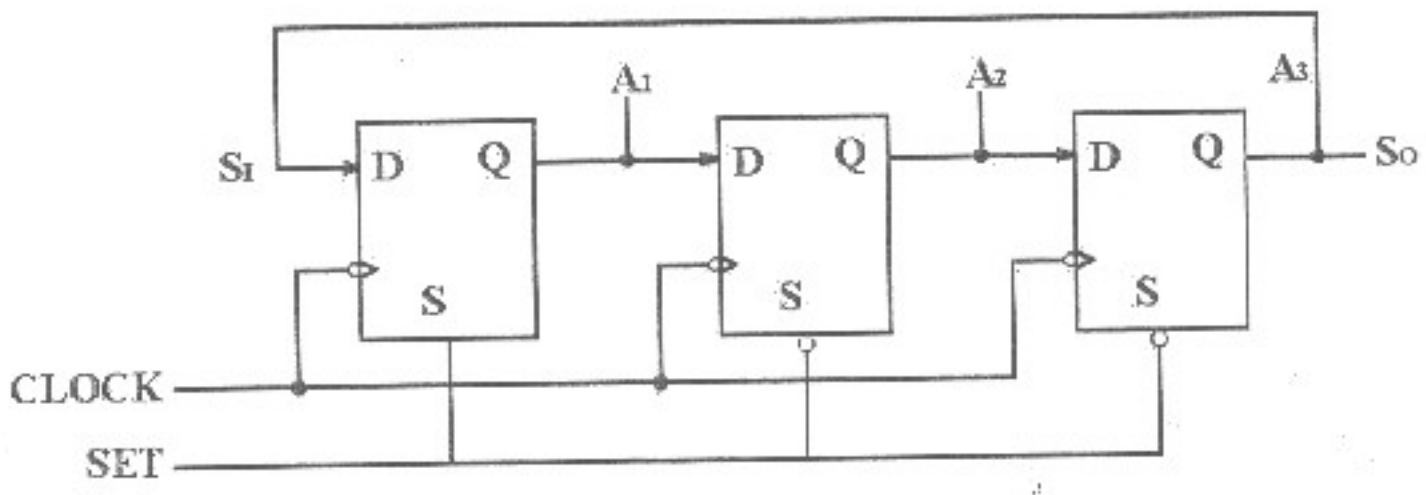
N –
10 – MOD

شمارندهٔ حلقوی: (Ring Counter)

اگر یک شمارندهٔ n بیتی داشته باشیم که هر کدام از F.Fهای موجود عملکردشان مانند فلیپ فلاپ باشد و خروجی آخرین فلیپ فلاپ به ورودی اولین فلیپ فلاپ وصل شود می‌توان یک شمارنده حلقوی ایجاد کرد. در یک شمارنده حلقوی می‌توان تمامی توان‌های صحیح z را ایجاد کرد. که این شمارش تا 2^{n-1} ادامه پیدا می‌کند.

شمارندهٔ n بیتی $1, 2, 4, 8, 16, \ldots, 2^{n-1}$

(با n فلیپ فلاپ)



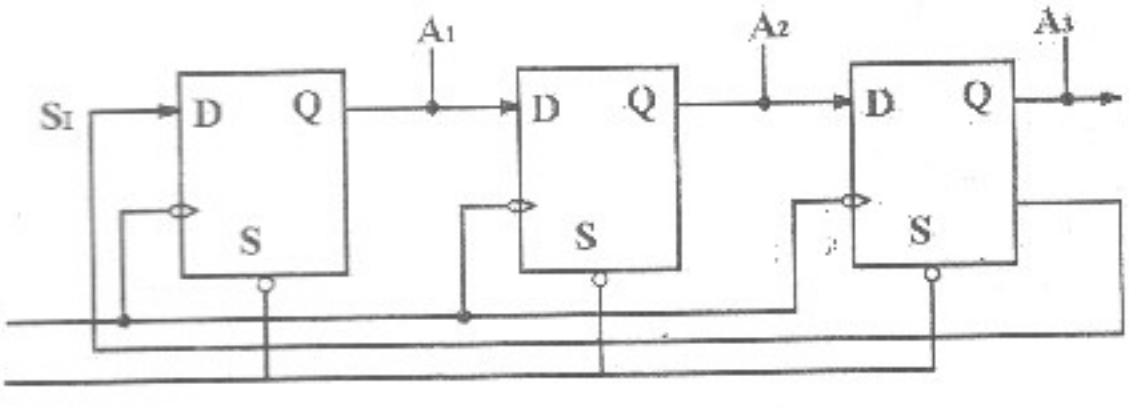
شمارندهٔ زیر یک شمارنده حلقوی سه بیتی می‌باشد که دنباله زیر ر تولید می کند:

1 → 2 → 4

تذکر: برای ایجاد مقدار اولیه باید به اولین فلیپ فلاپ خود s را وصل کنیم و سایر فلیپ فلاپ‌ها به Reset یا نقیض Set وصل شود.

شمارندهٔ جانسون:

مانند شمارندهٔ حلقوی عمل می کند با این تفاوت که ورودی S_I از Q' آخرین فلیپ فلاپ وارد می‌شود. و در ضمن برخلاف شمارندهٔ حلقوی اولین فلیپ فلاپ نیز از طریق نقیش S مقداردهی اولیه می‌شود.



برای تحلیل این شمارنده مقدار SET را صفر در نظر گرفته و در این حالت دنباله 6 → 4 → 0 → 1 → 3 → 7 تولید می‌شود.

جانسون

0	0	0	0	1	1	1
1	0	0	1	1	1	0
3	0	1	1	1	0	0
7	1	1	1	0	0	0
6	1	1	0	0	0	1
4	1	0	0	0	1	1
0	0	0	0	1	1	1

فرکانس خروجی شمارنده‌ها:

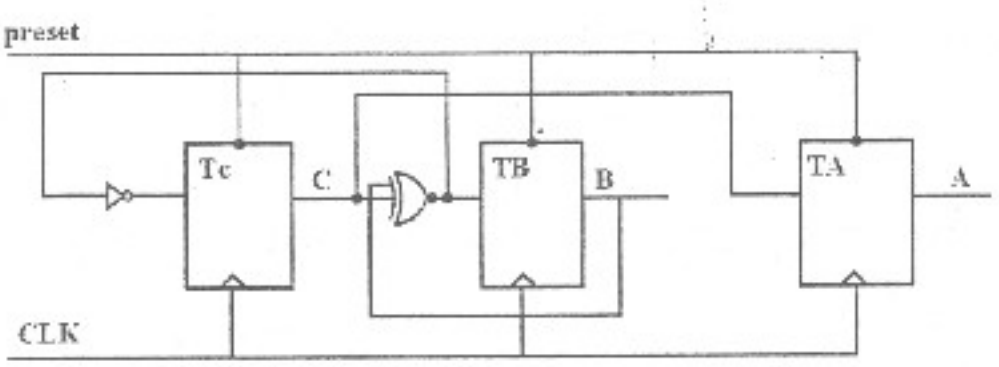
اگر فرکانس ورودی یک شمارنده MHZ باشد و این شمارنده توانایی شمارش n عدد را داشته باشد می‌توان فرکانس خروجی را از رابطه زیر بدست آورد:

$$\text{فرکانس خروجی} = \frac{\text{فرکانس ورودی}}{\text{تعداد حالات قابل شمارش توسط شمارنده}} = \frac{MHZ}{n}$$

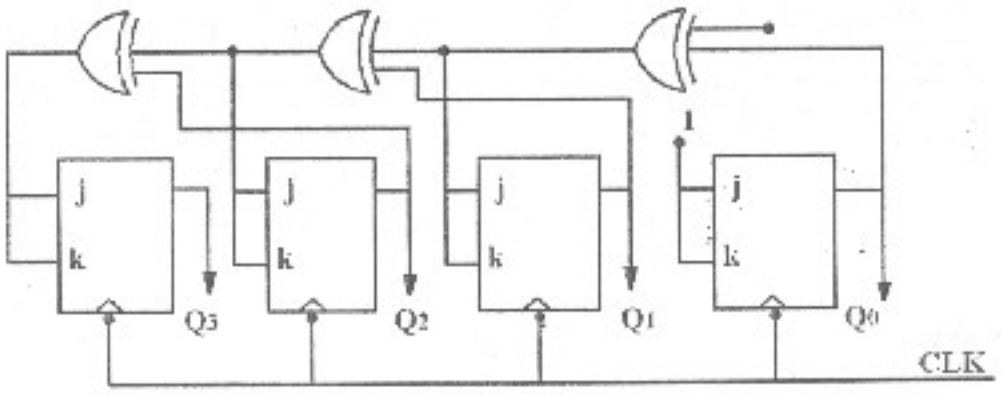
مجموعه مسائل فصل۵:

۱ـ در شمارنده زیر قبل از اعمال پالس‌های ساعت یا CIK شمارنده preset شده با آمدن پالس‌های ساعت سیکل شمارش عبارت است از :

	C	B	A		
preset	1	1	1	→	7
	1	0	0		4
	0	0	1		1
	0	1	1		3
	1	1	1		7

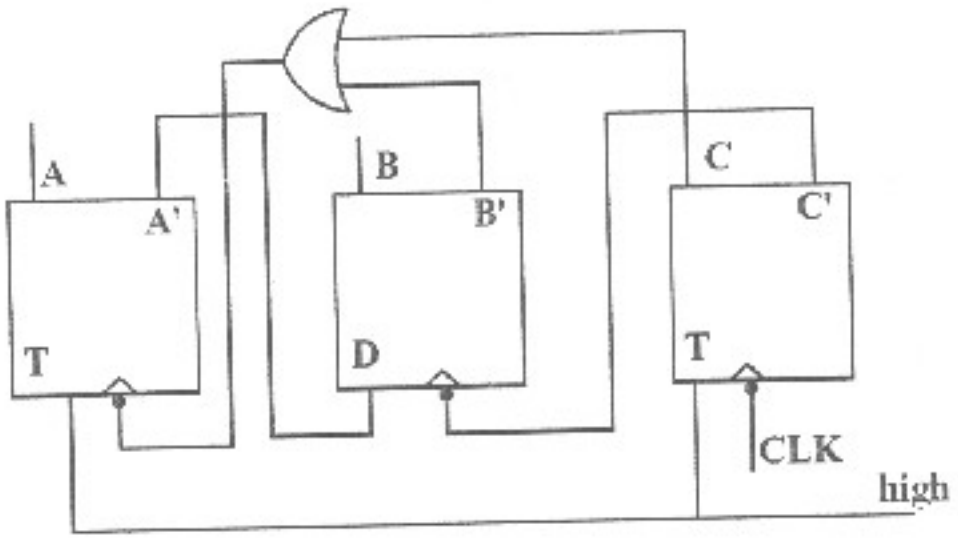


۲ـ در حالت زیر بیان کنید پس از 5 پالس ساعت چه رشته‌ای تولید می‌شود؟

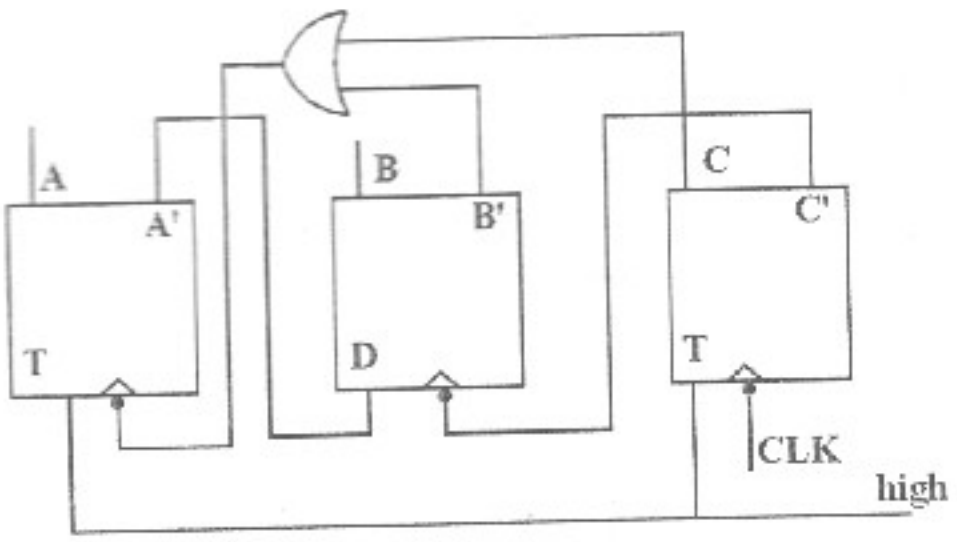


	Q_3	Q_2	Q_1	Q_0		
	0	0	1	0		
1	1	1	1	1	→	15
2	0	1	0	0	→	4
3	1	1	0	1	→	13
4	1	0	1	0	→	10
5	0	1	1	1	→	7

۳ـ اگر ABC = 000 باشد تحلیل کنید پس از 6 کلاک پالس ABC برابر چند خواهند بود؟

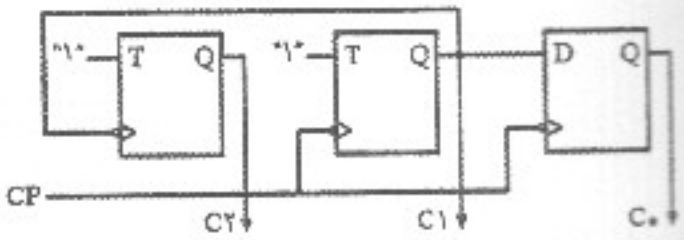


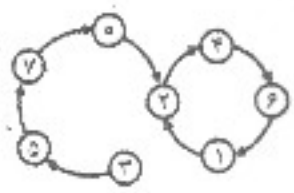
A	B	C	D
0	0	0	
0	0	1	①
0	0	0	②
0	0	1	③
0	0	0	④
0	0	1	⑤
0	0	0	⑥

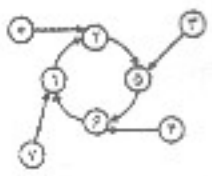


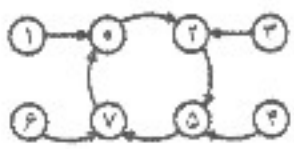
A	B	C	
1	0	1	
1	0	0	1
1	0	1	2
			⋮
1	0	1	6

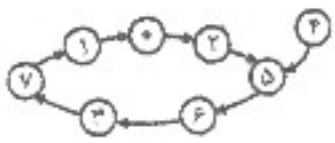
۵- دنباله شمارش مدار مقابل کدام است $(C_2, C_1 C_0)$ ؟



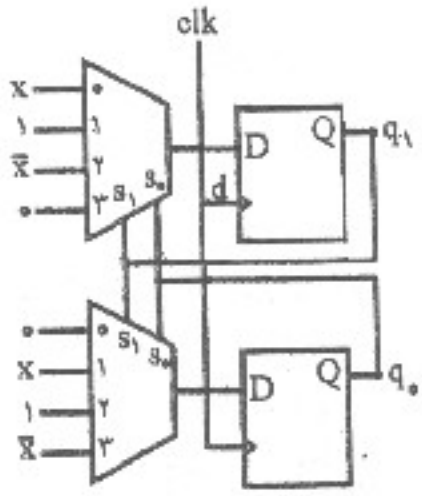
۱)


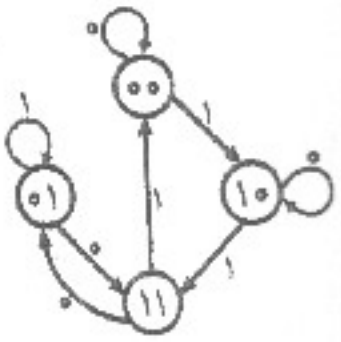
۲)


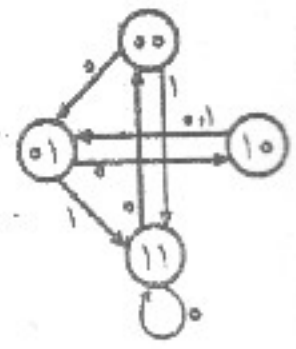
۳)


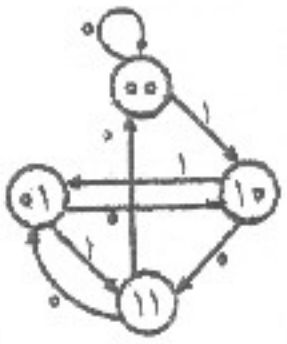
۴)


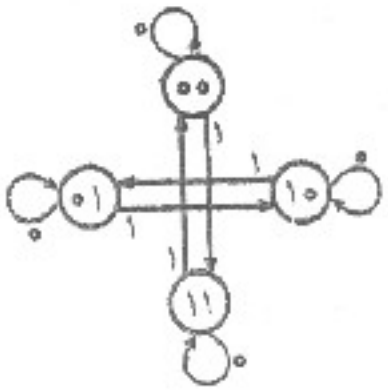
۶- نمودار حالت برای مدار زیر کدام است؟



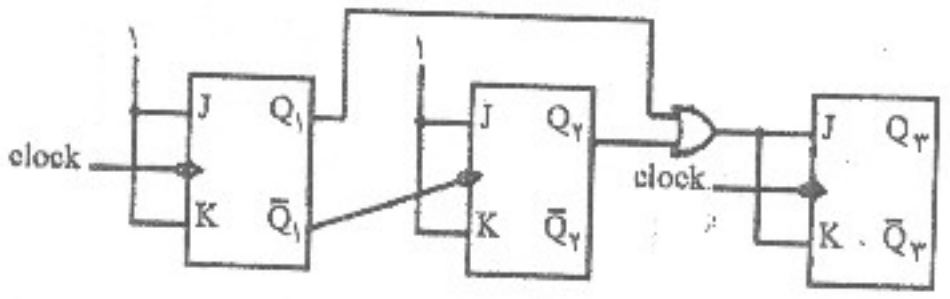
۱)


۲)


۳)


۴)


۷ـ شمارنده زیر چه رشته‌ای را به‌صورت تکراری می‌شمارد؟



۱) ⓪→③→④→⑥→⑦→①→②→⑤→

۲) ⓪→③→⑤→⑦→②→①→④→⑥→

۳) ⓪→③→⑥→⓪→③→⑥→⓪→①→⑥→

۴) ⓪→③→⑥→①→④→⑦→②→⑤→

۸ـ کدام‌یک از جملات زیر این ماشین را بهتر توصیف می‌کند؟

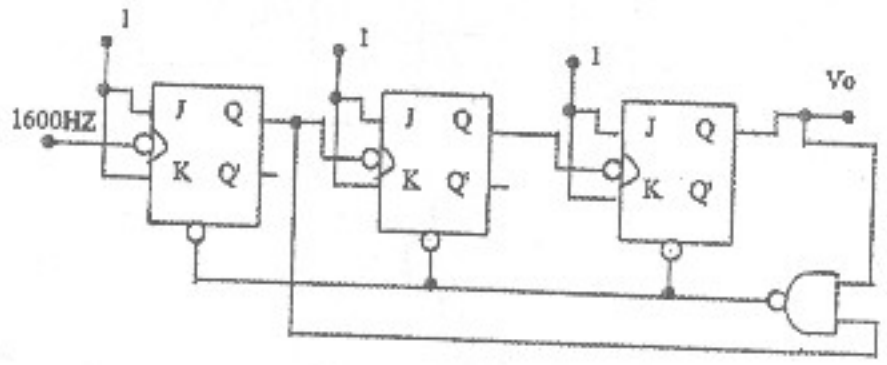
۱) دارای مسابقه غیر بحرانی است و نیازی به رفع آن نمی‌باشد.

۲) دارای مسابقه بحرانی (critical race) است و رفع آن به صورت زیر است:

۳) دارای مسابقه بحرانی است و رفع آن به صورت زیر است:

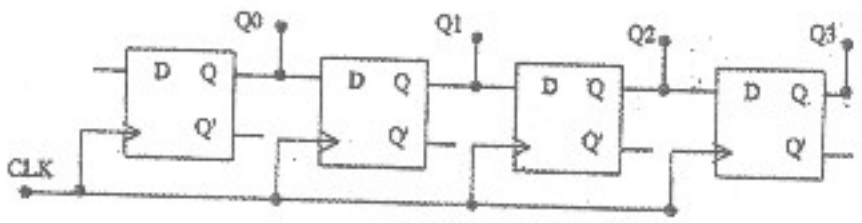
۴) هر دو جواب ۲ و ۳ صحیح هستند.

۹ـ فرکانس موج خروجی در مدار شکل زیر کدام است؟



۱) 320 HZ ۲) 200 HZ ۳) 160 HZ ۴) 400 HZ

۱۰ـ در شکل زیر یک ثبات 4 بیتی مشاهده می‌شود. کدامیک از موارد زیر غیرممکن است؟



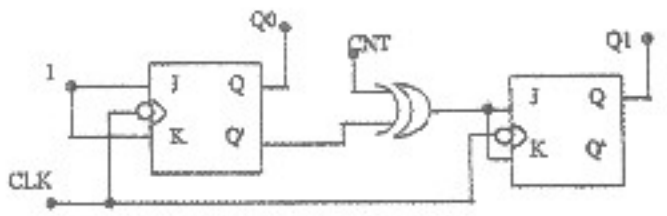
۱) خارج کردن داده‌ها به صورت سری از ثبات

۲) وارد کردن داده‌ها به صورت سری به ثبات

۳) وارد کردن داده‌هابه صورت موازی به ثبات

۴) خارج کردن داده‌ها به صورت موازی از ثبات

۱۱۱ـ مدار شکل زیر:



۱) به شرط CNT = 1 یک شمارنده پایین شمار (نزولی) 4 بیتی است.

۲) به شرط CNT = 0 یک شمارنده بالا شمار (صعودی) 2 بیتی است.

۳) به شرط CNT = 0 یک شمارنده پایین شمار 2 بیتی است.

۴) به شرط CNT = 1 یک شمارنده بالا شمار 4 بیتی است.

۱۱۲ـ یک Shift Register دارای m ورودی و n خروجی است نام این شیفت رجیستر چیست؟ $(m \neq n \neq 1)$

۱) SIPO ۲) PIPO

۳) بستگی دارد m بزرگتر است یا نه ۴) چنین چیزی امکان‌پذیر نیست.

۱۱۳ـ عدد $(1A)_{Hex}$ پس از دوبار شیفت منطقی به راست و سپس سه بار شیفت منطقی به چپ به چه عددی تبدیل می‌شود؟

۱) 30 ۲) 08 ۳) 1B ۴) AB