۱۰۰۰۱

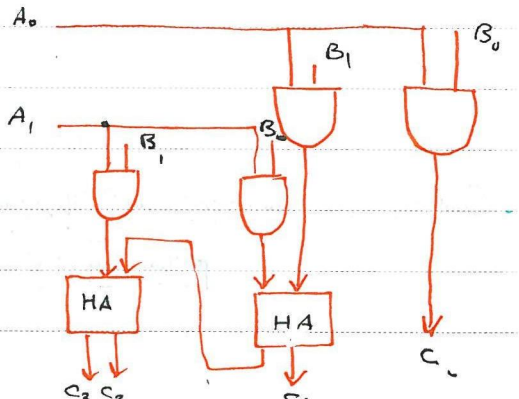
جلسه چهارم

ضرب کننده باینری :

می خواهیم مداری طراحی کنیم که کار ضرب کردن را انجام دهد. ابتدا برای سادگی کار یک ضرب کننده دو بیت در دو بیت را در نظر می گیریم -

$$\begin{array}{cc} B_1 & B_0 \\ A_1 & A_0 \\ \hline A_0B_1 & A_0B_0 \end{array}$$

$$\begin{array}{cc} A_1B_1 & A_1B_0 \\ \hline C_2 & C_1 \quad C_0 \end{array}$$



PqPCO

می‌خواهیم عدد سه بیتی A را در عدد چهار بیتی B ضرب کنیم. همانطور که از ضرب آن پیداست به Full-Adder های زیادی نیاز داریم بنابراین از آرایه‌های جمع‌کننده چهار بیتی باینری استفاده می‌کنیم.

$A_2\ A_1\ A_0$

$B_3\ B_2\ B_1\ B_0$

$B_0A_2 \quad B_0A_1 \quad B_0A_0$

$B_1A_2 \quad B_1A_1 \quad B_1A_0$

$B_2A_2 \quad B_2A_1 \quad B_2A_0$

$B_3A_2 \quad B_3A_1 \quad B_3A_0$

$C_6\ C_5 \quad C_4 \quad C_3 \quad C_2 \quad C_1 \quad C_0$

figure 4.16 – page 148

مدار مقایسه‌گر یا comparator :

کار این مدار این است که دو عدد چهار بیتی A و B را دریافت می‌کند آنها را با هم مقایسه کرده و می‌گوید که کدام یک بزرگتر است و یا این که با هم مساوی است. فرض کنید که دو عدد A و B به شکل زیر را داشته باشیم :

$$A = A_3A_2A_1A_0 \quad , \quad B = B_3B_2B_1B_0$$

مسلماً مقایسه بیت‌ها از با ارزش‌ترین بیت شروع می‌شود یعنی B_3 و A_3.

یک پارامتری به نام x_i را در نظر بگیرید : $x_i = A_iB_i + A_i'B_i' \qquad For\ i = 0,1,2,3$

اگر $x_i = 1$ شود مشابه آن چیزی که برای gate XNOR تعریف می‌کردیم، یعنی دو بیت ورودی آن با هم برابرند. در صورتی که دو عدد A و B با هم برابر باشند نتیجه می‌گیریم که همه‌ی خروجی‌های $x_3x_2x_1x_0$ باید 1 باشند.

$$(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

$$(A<B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1 + x_3x_2x_1A_0'B_0$$

برای اینکه عدد A از B بزرگتر باشد به ترتیب از با ارزش‌ترین بیت به کم ارزش‌ترین بیت آنها را چک می‌کنیم. اگر با ارزش‌ترین بیت A (A_3) بزرگتر از با ارزش‌ترین بیت B (B_3) باشد باید A_3B_3' برابر 1 شود. چرا که تنها حالتی که اتفاق می‌افتد و صحیح است که $A_3 = 1$ و $B_3 = 0$ باشد. برای مقایسه‌ی بیت بعدی (A_2, B_2) ابتدا شرط مساوی بودن $A_3 = B_3$ باید برقرار باشد

$x_3 = 1$ ⇐

پس از آن A_2 و B_2 همانند A_3 ، B_3 باهم جمع می‌شوند. چون شرط منفی بودن x_3 برقرار است، x_3 باید در A_2B_2 ضرب شود.

figure 4.17 page 150

Decoder ها یا مدارهای رمزگشا :

کار مدارهای رمزگشا این است که n بیت را در ورودی دریافت کند و آنرا به دو به توان n خروجی تبدیل کند. مثلاً 8 در قفل را در نظر بگیرید که از d_0 تا d_7 نامگذاری شده‌اند. سه سوئیچ رمزی در اختیار داریم که می‌تواند صفر یا یک باشد. کار decoder این است که اگر مثلاً این سه سوئیچ هر سه صفر باشند، در d_0 باز شود و اگر هر سه یک باشند در d_7 جدول صحت آن به شکل زیر است:

Table 4.6 page 152

figure 4.18 page 151

برخی اوقات مدارهای دیجیتال به ورودی موسوم به Enable مجهز می‌شوند. کار این ورودی این است که اگر 1 باشد مدار فعال شده و خروجی‌ها به درستی می‌آید ولی اگر صفر باشد مدار به ازای هر ورودی غیرفعال است.

گاهی اوقات مدار اصطلاحاً Active_low است. در این حالت خروجی‌ها با Enable ، fire می‌شود اگر 1 بیاید مدار غیرفعال است.

Example : یک Decoder دو در چهار با ورودی Enable و با گیت NAND طراحی کنید.

figure 4.19

Example : می‌خواهیم به کمک دو Decoder سه در هشت یک Decoder چهار در شانزده درست کنیم. جدول صحت آن به شکل زیر است:

w	x	y	z	D_0	D_1	...	D_8	D_9	----	D_{16}
0	0	0	0	1	0	0	0	0	.	0
0	0	0	1	0	1	0	0	0	0	0
:										
1	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	1	0	0
:										
1	1	1	1	-	-	-	-	-	-	1

P4PCO

ما دو Decoder سه درهشت داریم که هرکدام 8 خانه از آدرس‌های خروجی حافظه را می‌سازند. Decoder اول D_7 تا D_0 و Decoder دوم D_{15} تا D_8 را می‌سازند.

Example: می‌خواهیم یک Full-Adder را به کمک یک Decoder سه درهشت تولید کنیم.

$$S(x,y,z) = \Sigma(1,2,4,7)$$

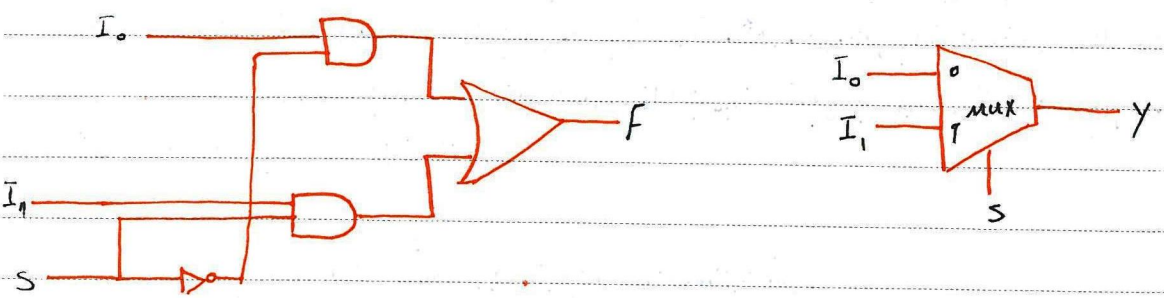
$$C(x,y,z) = \Sigma(3,5,6,7)$$

Figure 4.21

مالتی‌پلکسر

Multiplexer: این مدار 2^n ورودی، n خط آدرس و 1 خروجی دارد. تمام ورودی‌ها توسط درب‌های سه‌حالته به خروجی متصل‌اند. n خط آدرس روی هرکدام از این درب‌ها است. مثلاً برای یک Mux هشت‌ورودی، احتیاج به سه خط آدرس داریم. اگر آدرس ... بیاید درب اول باز شده و این ورودی به خروجی جاری می‌شود.

Example: یک Mux دو ورودی را در نظر بگیرید. سلول سمبلیک و مدار آن به شکل زیر است: Figure 4.24



Example: جدول صحت برای یک Mux چهار ورودی به صورت زیر است:

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

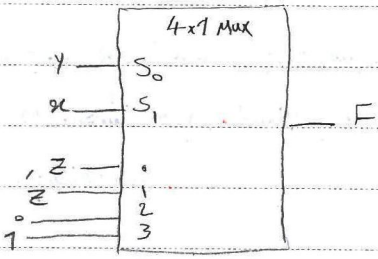
می خواهیم یک Mux طراحی کنیم که زمانی که ورودی E صفر باشد (ورودی E صفر باشد) برحسب آدرس هایی که دریافت می کند، یکی از 4 بیتی را در خروجی بنشاند. یعنی اگر آدرس صفر داشت کلمه A و اگر آدرس یک داشت کلمه B را در خروجی بنشاند.

page 160 → Figure 4.26

Example(imp): به کمک یک mux چهار دریک خروجی زیر را پیاده سازی کنید. $F(x,y,z) = \Sigma(1,2,6,7)$

page 161 → Figure 4.27

x	y	z	F	
0	0	0	0	F = z
0	0	1	1	
0	1	0	1	F = z'
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

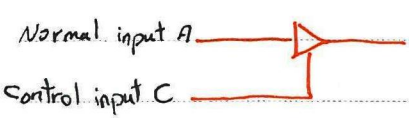


توضیح: از آن جایی که 8 بیتم در این مثال مورد نظر بوده است به یک Mux هشت دریک احتیاج داریم که ورودی هایی که این بیتم ها را می سازد خطوط آدرس این Mux باشد. ولی چون صورت مسئله ما را ملزم به استفاده از یک Mux چهار دریک کرده است، به جای آن که آنها را با صفر و یک نمایش دهیم آنها را با کم ارزش ترین بیت ورودی نمایش می دهیم و دو ورودی دیگر نقش آدرس دهی به این Mux را خواهند داشت.

Example: به کمک یک Mux هشت دریک، تابع زیر را پیاده سازی کنید.

page 162 → Figure 4.28

گیت های سه وضعیتی:



شکل صفحه قبل یک بافر سه وضعیتی است که یک ورودی و یک خروجی و یک پین کنترل دارد. زمانی که پین کنترل ما صفر باشد، بافر ما مانند یک مقاومت با امپدانس بالا عمل می‌کند ولی زمانی که C = 1 باشد بافر نقش عادی خود را خواهد داشت.

طراحی Multiplexer به کمک Decoder

می‌خواهیم یک Mux چهار در یک را به کمک بافرهای سه وضعیتی و یک Decoder دو در چهار طراحی کنیم (برای Decoder ورودی Enable را هم در نظر بگیرید)

page 163 → figure 4.30 (b)

Example: یک Mux دو در یک را به کمک بافر سه وضعیتی پیاده سازی کنید.

page 163 → figure 4.30 (a)

جلسه دهم

مدارات رمزگشا - Encoders

برخلاف Decoders، Encoders مدارهایی هستند که داده‌های ورودی را دریافت کرده و در خروجی رمز را تولید می‌کنند. با توجه به جدول صحت زیر، خروجی‌های یک مدار Encoder را می‌توان به شکل زیر بدست آورد.

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	1
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

$$z = D_1 + D_3 + D_5 + D_7$$

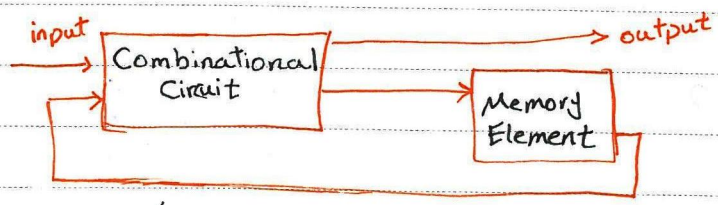
$$y = D_2 + D_3 + D_6 + D_7$$

$$x = D_4 + D_5 + D_6 + D_7$$

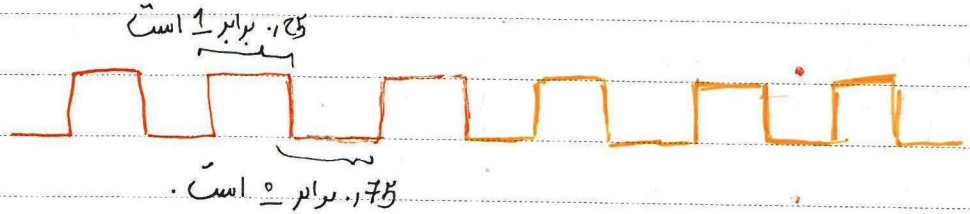
تمرین: خروجی یک Encoder ، 16x4 را طراحی کنید.

فصل 5 ، مدارات ترتیبی :

برخلاف مدارات ترکیبی که خروجی مستقیماً به ورودی ربط دارد، در مدارهای ترتیبی که از یک مدار ترکیبی و یک عنصر حافظه باید حداقل تشکیل شده باشد، خروجی می‌تواند به ورودی feedback شود.

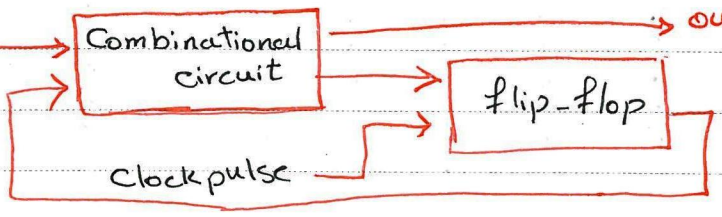


در واقع یک مدار ترتیبی از مجموعه‌ای از ورودی‌ها، خروجی‌های ترکیبی دارو حالت‌های داخلی تشکیل شده است.
در بسیاری از این مدارات برای همزمان سازی همه اعمال ها از clock pulse استفاده می‌شود.



از پالس ها به طور منظم از صفر و یک ها هستند که بر حسب این که در هر T از این پالس ها چقدر در صد یک باشد، Duty cycle را تشکیل می دهند.

یکی از عناصر حافظه که در مدارات ترتیبی استفاده می شود، عنصری دوسیم به flip-flop است. این عنصر که با لبه های بالا رونده کار می کند نقش مهمی دارد. نحوه اتصال آن به شکل زیر است:



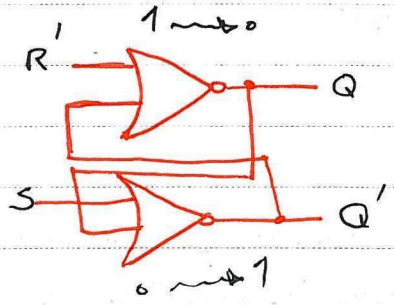
عناصر حافظه: الف) Latchs این حافظه ها مانند flip-flops عناصر حافظه هستند که با clockpulse کار می کنند اما در تمام مدت زمانی که clockpulse=0 (یا 1=) به ما [ ] تحویل می دهد. در غیر این صورت Deactive است.

SR Latch: این Latch دو ورودی R, S دارد. اگر S=1 ⇐ output(Q)=1 و اگر R=1 ⇐ output(Q')=0 و اگر S=0 و R=0 ⇐ output: بدون تغییر نسبت به حالت قبلی

R=1 ← Reset

اگر R=1 و S=1 ⇐ حالت ناخواسته (ممنوعه) زیرا Q=Q'=0

↳ مدار ترتیبی ↳ Impossible

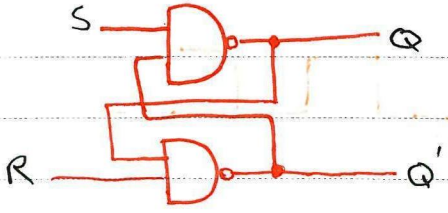


forbidden

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

طراحی SR Latch با گیت NAND:

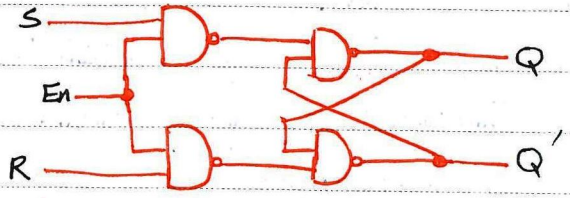
همه چیز برعکس!



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

PAPCO

طراحی مدار SR Latch همراه با Clock pulse :



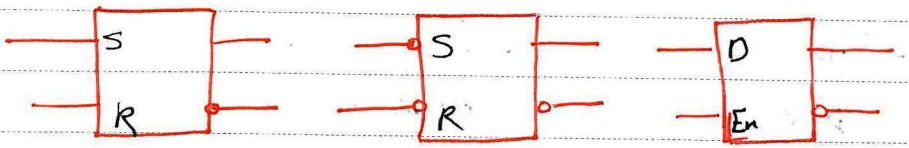
En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q=0 reset state
1	1	0	Q=1 set state
1	1	1	Indeterminate

Figure 5.6.

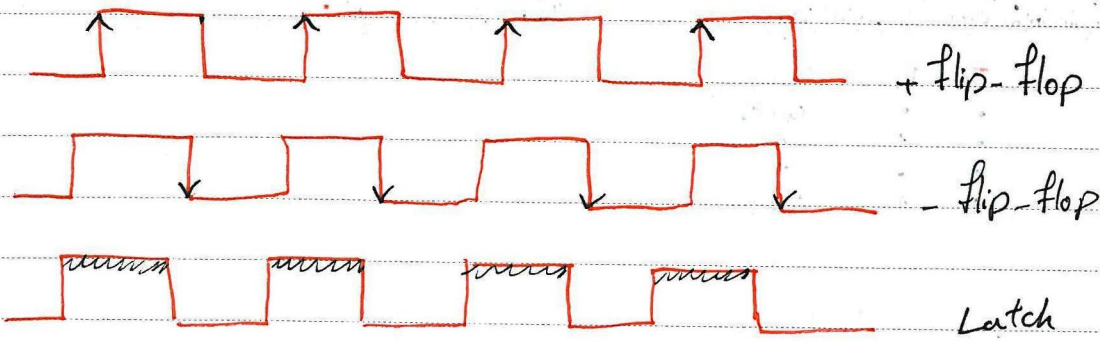
D_latch :

ورودی D تنها یک ورودی است و کار آن این است که هرچه در D بیاید، خروجی هم همان خواهد بود

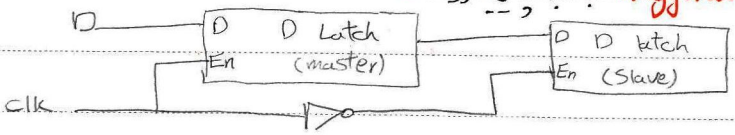
Graphics symbols of Latchs :



ب، flip-flops : مانند Latchs، flip-flops زمانی ON هستند که clock pulse در حد قرارداشته باشد (مثلاً لبه بالارونده یک روند لحظه‌ای که پالس از صفر به یک می رود)



Example : Triggered D flip-flop با لبه پالس رونده :



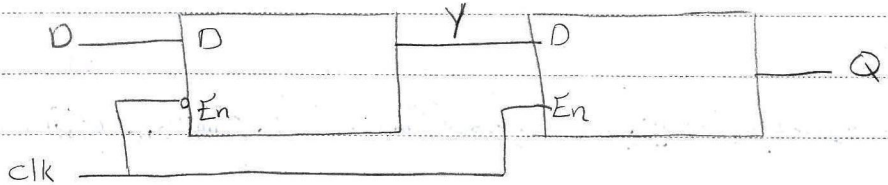
در این مدار زمانی که $Q = 0$ هم‌چنان D $latch_2$ ، $Y = 1$ ← D $Latch_1 = 1$ ← clock pulse = 1

به محض این‌که clock pulse = 0 شد ← $Q = 1$ ← D $latch_2 = 1$ ← NS

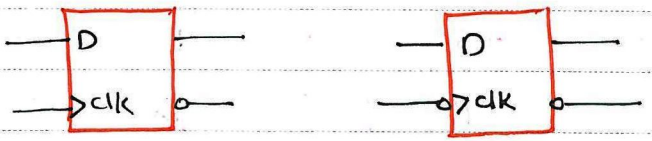
⇐ می‌توان نتیجه گرفت درست در لحظه‌ای که از 1 به 0 رفته است مدار فعال شده

⇐ مدار حساس به لبه پایین رونده این پالس است.

تمرین: D flip-flop با لبه بالا رونده:



Graphic symbol for edge-triggered D flip-flop:



چند flip-flop دیگر:

a. JK flip-flop: این flip-flop مشکل SR latch را برطرف کرده و در آن دیگر حالت ناخواسته نداریم:

J	K	Q(t+1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	set
1	1	Q'(t)	Complement

b. D flip-flop:

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

c. Toggle flip-flop (T-F):

T	Q(t+1)	
0	Q(t)	No change
1	Q'(t)	complement