این درس به درد کجای موبایل من می خوره

X و Z چیست ؟

Subject

Date

به نام خدا

میان ترم : پنجشنبه ۲۰ آبان ۱۲ ـ ۸

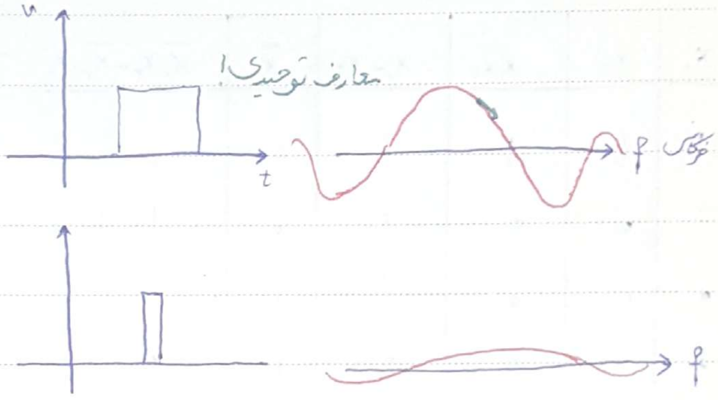
Digital signal processi (DSP)
- Coding / deCoding → محافظت از بلایای طبیعی
- Compressing
- encryption → محافظت از بلایای انسانی

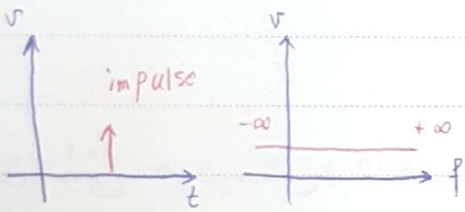
Digital Communication

نایکوئیست ویو   Nicoist wave

↕

تعداد سمپل گیری = ۲ × ماکسیمم فرکانس

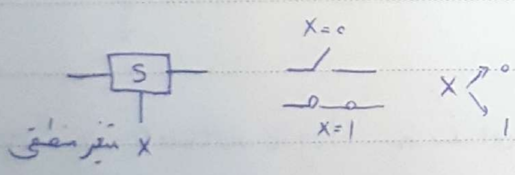


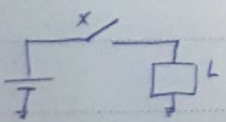


dis (گسسته)
- binary → X: 0, 1, Z → جبر بولی
- non binary

مدارها:
- ترکیبی (Combinational) : خروجی ها فقط به ورودی های حال حاضر بستگی دارد
- ترتیبی (sequential) : خروجی ها هم به ورودی های حال حاضر هم به مقادیر گذشته (حافظه تاریخی مدار) بستگی دارد

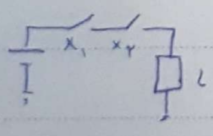
متغیر / Variable





$$L(X) = \begin{cases} 1 & x = 1 \\ 0 & x = 0 \end{cases}$$

1) AND



$$L(x_1, x_2) = \begin{cases} 1 & x_1 = x_2 = 1 \\ 0 & x_1 = 0 \text{ or } x_2 = 0 \end{cases}$$

$X_1 . X_2$

۲

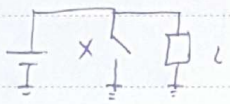
PAPCO

Subject
Date ۹۵/۷/۳

تعداد (Fan, out) و (Fan, in) چرا بیشتر ۴ باشد

$$L(x_1, x_2) = x_1 + x_2 = \begin{cases} 1 \\ 0 \end{cases} \quad x_1 = x_2 = 0$$

$$L(x) = \bar{x} = \begin{cases} 0 & x = 1 \\ 1 & x = 0 \end{cases}$$

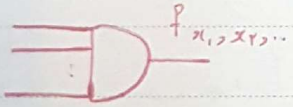


⇒ Not گیت

x_1	x_2	$x_1 x_2$	$x_1 + x_2$	$\bar{x}$	$\bar{x}_1 \cdot x_2 + \bar{x}_2 \cdot x_1$ → XOR
1	1	1	1	0	0
1	0	0	1	0	1
0	1	0	1	1	1
0	0	0	0	1	0

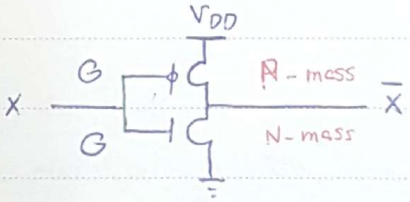
★★ گیت‌های منطقی: هر عمل منطقی (AND, OR, NOT, ...) توسط یک gate منطقی پیاده‌سازی می‌شود

AND



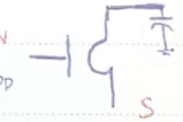
NOT

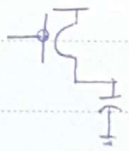
یک gate دارای تعدادی ورودی و یک خروجی است.



درین D

سورس S





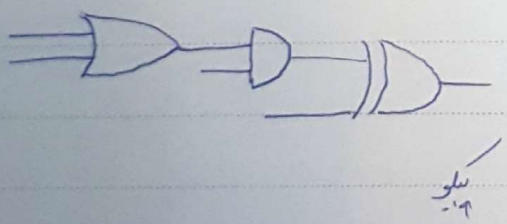
ماسفت

بیشتر ۴ = Fan, in → ماکسیمم تعداد ورودی‌ها

بیشتر ۴ = Fan, out → ماکسیمم تعداد خروجی‌ها

باید صفر باشد در بهترین حالت P

gate ← logic network ← شبکه‌ای از گیت‌ها



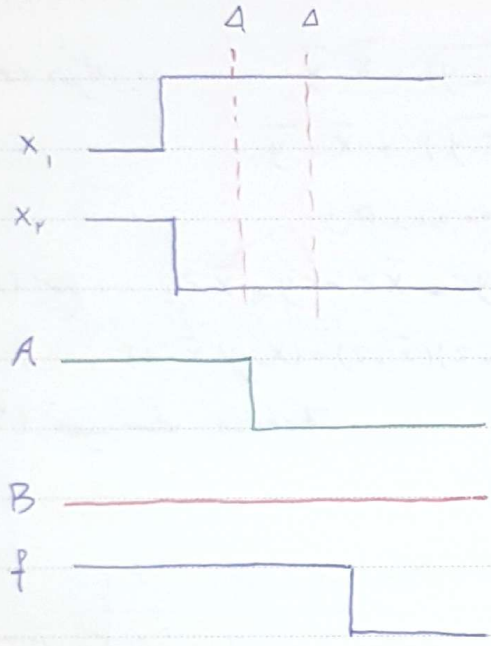
gate delay = 4 ps

X : UnKnown

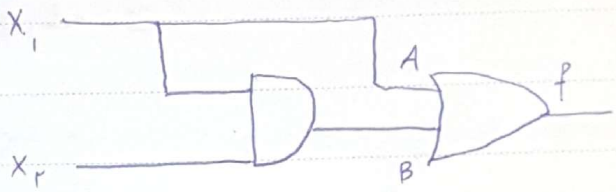
Z : High Impedance

Subject

Date



delay حداکثر (critical path) (CP)



جبر بولین :

I) $0.0=0$ $0.1=1.0=0$

$1+1=1$

$1.1=1$ $1+0=0+1=1$

$0+0=0$ $\bar{0}=1$ $\bar{1}=0$

قضایای X :

$X.1=X$ $X+0=X$

$X.0=0$ $X+1=1$

$X.X=X$ $X+X=X$

$X.\bar{X}=0$ $X+\bar{X}=1$

$\overline{(\bar{X})}=X$ $\bar{X}=X \rightarrow$ متمم

نمی‌شود روش حساب بزرگ کرد

قضایای چند متغیره :

۱) جابه جایی $X.Y=Y.X$ $X+Y=Y+X$

۲) انشراک پذیری $X.(Y.Z)=(X.Y).Z$ $X+(Y+Z)=X+Y+X.Z$

۳) توزیع پذیری $X.(Y+Z)=X.Y+X.Z$ $X+(Y.Z)=(X+Y).(X+Z)$

۴) جذب $X.(X+Y)=X$ $X+(X.Y)=X$ $X.1+X.Y=X.(1+Y)=X$

۵) ترکیب پذیری $X.Y+X.\bar{Y}=X$ $(X+Y)(X+\bar{Y})=X$

Subject
Date

Shannon limit شنون لیمیت

۶) دمرگان $\overline{(x+y)} = \bar{x}\cdot\bar{y}$ ۷) شبه جذب $x + \bar{x}y = x + y$

$\overline{(x\cdot y)} = \bar{x} + \bar{y}$ $x\cdot(\bar{x}+y) = x\cdot y$

این ۲ تا رو بلدن هرچی میرند رو اندکی فرق نداره ت

۸) اجماع $xy + yz + \bar{x}z = xy + \bar{x}z$

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

۹) قضیه بسط Shannon $f(x_1, \dots x_n) = x_1 f(1, x_2, \dots, x_n) + \bar{x}_1 f(0, x_2, \dots, x_n)$

ساده کردن = سفتز کردن

سفتز برلین :

مراحل : ۱) تشکیل جدول درستی از روی توصیف تابع

۲) تشکیل تابع f از روی جدول

۳) ساده سازی تابع f بوسیله قضایای فوق

۴) کشیدن ساختار مدار نهایی

تابع ۳ ورودی طراحی کنید که زمانیکه $x=1$ و یکی از دو ورودی دیگر هم "۱" هستند خروجی "۱" دهد

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

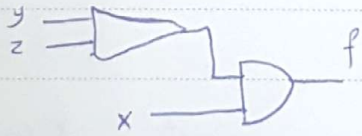
$x\bar{y}z$ ← 1 0 1

$xy\bar{z}$ ← 1 1 0

minterm: xyz

sop Some of product

$f = x\bar{y}z + xy\bar{z} + xyz = x(z+y)$



Maxterm: $(\bar{x}+y+z)(x+\bar{y}+z)(x+\bar{y}+\bar{z})(x+y+\bar{z})(x+y+z)$

Pos : Product of some

$f = \sum m(0,1,2,4) =$

Subject
Date

نوشتن تابع f از روی جدول درستی

↳ از روی سطرهایی که $f=1$

↳ از روی سطرهایی که $f=0$

minterm : برای یک تابع n متغیره یک minterm از حاصلضرب هر n متغیر چه بصورت x و یا $\bar{x}$ تشکیل می‌شود

Maxterm : " " " " maxterm از حاصل جمع "

برای یک تابع n متغیره : minterm 2^n ، maxterm 2^n دارد

x_1	x_2	x_3	minterm	maxterm
0	0	0	$m_0 = \bar{x}_1\bar{x}_2\bar{x}_3$	$M_0 = x_1 + x_2 + x_3$
0	0	1		
0	1	0		
0	1	1		
1	0	0	$m_4 = x\bar{y}\bar{z}$	$M_4 = (\bar{x} + y + z)$
1	0	1		
1	1	0		
1	1	1		

Sum of Product : عباراتی که از مجموع چند جمله که هر یک حاصلضرب چند متغیر هستند.
SoP

مثال : $f = x_1x_2 + x_2x_3 + \bar{x}_1x_2x_3$

Product of Sum : عباراتی که از حاصل ضرب چند جمله که هر یک حاصل جمع چند متغیر هستند
Pos

هر تابع f را می‌توان برحسب مجموع minterm هایی که در سطر متناظر آنها در جدول درستی $f=1$ است نوشت

" " $f=0$ " حاصلضرب Maxterm " " "

SoP هایی که همه عبارت ها مانند minterm ← Connonical SoP ← نوشتن این یکیست

" Pos " Maxterm " " Pos

افتضاح ترین از نظر ساده سازی

Subject
Date

$f = \sum m(1,4,5,7) = m_1 + m_4 + m_5 + m_7$

$\bar{f} = m_0 + m_2 + m_3 + m_6 \Rightarrow f = \overline{(m_0 + m_2 + m_3 + m_6)} = \bar{m}_0 \cdot \bar{m}_2 \cdot \bar{m}_3 \cdot \bar{m}_6 = M_0 M_2 M_3 M_6$

ساده کنید :

$f = x_2 \bar{x}_3 x_4 + x_1 x_2 x_4 + x_1 \bar{x}_2 x_4$

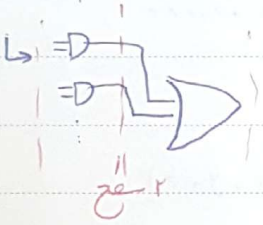
$= x_2 \bar{x}_3 x_4 + x_1 x_4$

$f = \sum m(2,3,4,5,6)$

$x_2 + x_1 \bar{x}_3$

Sop , Pos two level rep of func

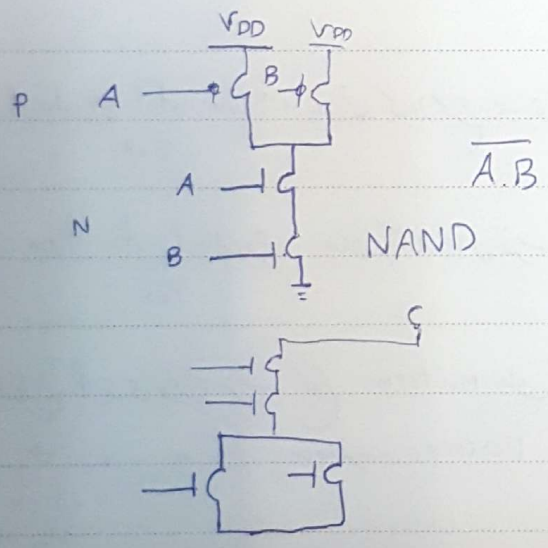
NAND ، NOR از نظر سرعت و راحت تر

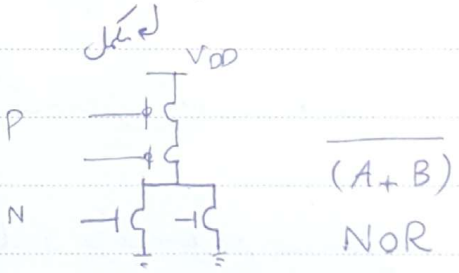


CMOS

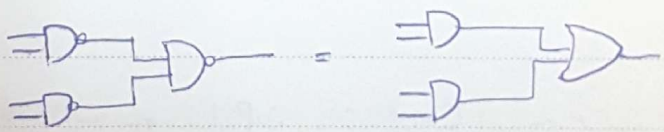
Compliment Mos

inverter دو ترانزیستور





AND را با NAND می سازند



Subject
Date

نمایش اعداد:

★ ۱۰ Decimal
★ ۲ Binary
۸ octal
۱۶ Hex

فیلی‌صم
MSB ↑ ↑ LSB

$B = b_{n-1} \dots b_1 b_0 \quad b_i \in \{0,1\}$

$V(B) = \sum_{i=0}^{n-1} b_i \times 2^i$

$B = 1101 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$

$(1101)_2 = (13)_{10}$

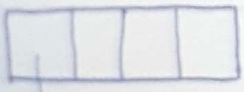
$(0,8125)_{10} = (0,1101001)_2$

واسافار: $b_{-1}\times 2^{-1} + b_{-2}\times 2^{-2} + \dots$

$0.b_{-1}b_{-2}b_{-3}\dots$

اعداد علامت دار (signed)

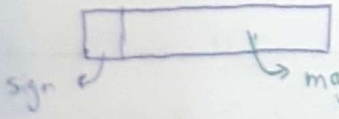
نمایش ← (علامت ـ اندازه)
① Sign-magnitude
② 1's compliment
★ ③ 2's compliment



Sign { ۰: مثبت / ۱: منفی

$[-7, \dots, 7]$

$[-8, \dots, 7]$ → 2's compliment



① SM (sign-magnitude)

محدوده نمایش $[-(2^{n-1}-1), \dots, 2^{n-1}-1]$

$B = b_{n-1} \dots b_1 b_0$

$$V(B) = \begin{cases} \sum_{i=0}^{n-2} b_i \times 2^i & b_{n-1} = 0 \\ -\sum & b_{n-1} = 1 \end{cases}$$

+۷: 00111

-۷: 10111

Subject
Date

۲) 1's Compliment

$N_{منفی} = L - |N|$

با 1's $2^n - 1$

نحوه نمایش: $[-(2^{n-1}-1), \dots, 2^{n-1}-1]$

$-5 = (2^4 - 1) - 5 = (1111) - (0101) = 1010$

۳) 2's Compliment

$N = 2^n - |N|$

محدوده نمایش: $[-2^{n-1}, \dots, 2^{n-1}-1]$

یکی بیشتر

$5 \rightarrow 0101 \xrightarrow{نات} 1010 + 1 = 1011$

هرگونه از سمت راست حرکت می‌کند به اولین 1 برسد سمت چپی ها را نات می‌کند

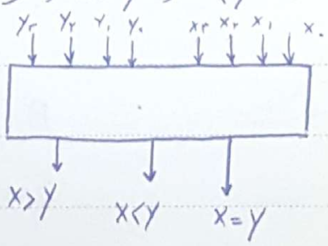
	2's	1's	sign-mag
3	011	3	3
2	010	2	2
1	001	1	1
0	000	+0	+0
-1	111	-0	-3
-2	110	-1	-2
-3	101	-2	-1
-4	100	-3	-0

sign-extention هیچ اتفاقی نمی‌افتد یعنی هی 1 قبل عدد را بگذار

$-3 = 101$

$-3 = 11111101$

مداری طراحی کنید که در عدد ۴ بیتی X و Y را مقایسه کرده و هرکدام از حالات X=Y و X<Y و X>Y را گزارش دهد



Subject
Date

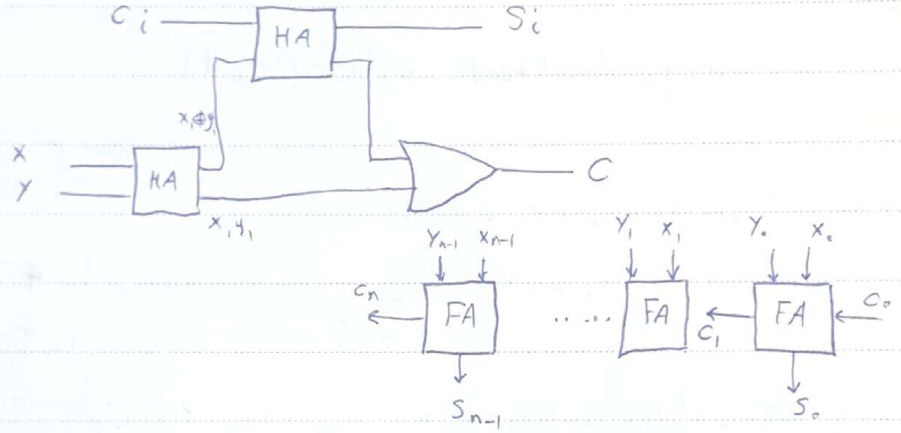
Hardware language

Verilog
VHDL
HDL
↳ discription

HDL → behavioral
HDL → RTL
HDL → structual (gate level)

assign ↔ wire
always ↔ reg

Critical Pass یا کمترین تأخیر



$f_{max} \propto \frac{1}{2n\tau_g}$ $\quad CP = n\Delta t = 2n\tau_g$

```
module FA(Ci, X, Y, S, Cout);
    input  Ci, X, Y;
    output S, Cout;
    assign S = X ^ Y ^ Ci;
    assign Cout = (X & Y) | (X & Ci) | (Y & Ci);
end module
```

Always ⇒

```
output reg S, Cout;
always @(x, y, Ci)
begin
    S = X ^ Y ^ Ci;
    Cout = ....
end
```

PAPCO

Subject
Date

```
Paramater  n=8;
input  [n-1:0]  x,y;
input  ci;
output  co;
output  [n-1:0] s;
assign  {co,s} = {1'b0,x} + {1'b0,y} + ci;
end
```

Carry_lookhead adder (CLA)

ایده: محاسبه C_i ها بصورت مستقل از هم (موازی)

$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$= \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} \underbrace{c_i}_{(g_{i-1} + p_{i-1} c_{i-1})}$$

CLA تأخیر: $4\tau_g$ p_{ix}

RCA: $2n\tau_g$

$$C_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \dots + p_i p_{i-1} \dots p_1 p_0 c_0$$

RCA	CLA
پایین : سرعت	بالا : سرعت
پایین : پیچیدگی (area)	زیاد : پیچیدگی

سلسله مراتبی

جمع و تفریق اعداد علامت دار:

S&M: وقتی دو عددی که قرار است جمع شوند هم علامت باشند هم علامت باشند ⇐ جمع اندازه‌ها و اضافه کردن علامت در انتها

```
  0101        1110
  0010        1001
 ------      ------
  0111        1111
```

وقتی علامت مختلف دارند عددی که کوچکتر از لحاظ کمتر است کم می‌کنیم و علامت بزرگتر را می‌گذاریم

Subject
Date

جمع اعداد 1's complement

+5 0101 -5 1010
+2 0010 -2 1101
+7 0111 -3 (1)0111 → 1
1000

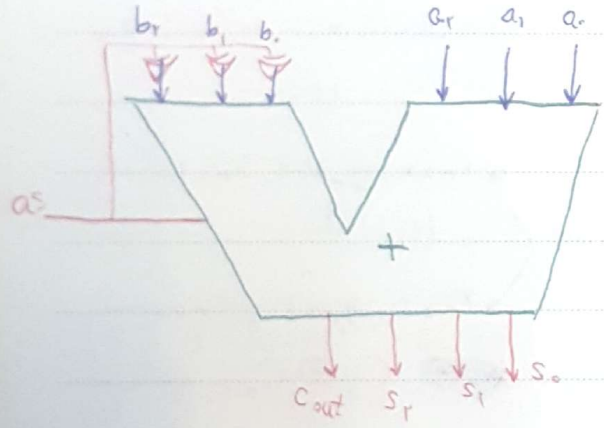
وقتی Carry out بعد این جمع صحیح نیست و باید تصحیح شود

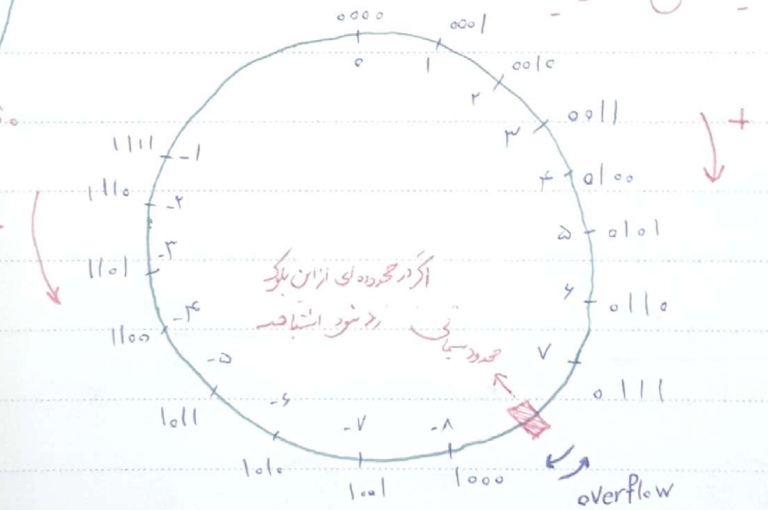
+5 0101 -5 1010
-2 1101 -2 1101
+3 (1)0010 → 1 -7 (1)0111 → 1
0011 1000

جمع اعداد در 2's comp

0101 5 1011 -5
0010 2 1110 -2
0111 (1)0011 (1)1001

نمایش گرافیکی جمع و تفریق اعداد 2's compl





+5 0101
+4 0100
+9 1001 → -7 ✗ غلط

unsigned دیبالت   signed : علامت دار
Subject
Date   complement 2's دیبالت:

sign extension

```
 0101            signext       0 0101   +5
 0100              =>          0 0100   +4
 ----                          ------
 1001  -> 7 X                  0 1001   (+9) ✓
```

```
wire [3:0] a, b;
wire [4:0] c;
assign c = { a[3], a } + { b[3], b };
```

```
-6    1010
-3    1101
     -----
   ✗ 0111   (+7) X

     1 1010
     1 1101
    -------
   ✗ 1 0111  (-9) ✓
```

برای جمع یک ۳بیتی با ۱۰بیتی باید هردو تا ۱۱بیتی sign extension شوند (۱- عدد منفی هم ! اضافه کن)

معیار تشخیص overflow:

```
C4  C3  C2  C1  C0
     x3  x2  x1  x0
     y3  y2  y1  y0
-------------------
C4  S3  S2  S1  S0
```

overflow → $x_3=0$, $y_3=0$, $S_3=1$
overflow → $x_3=1$, $y_3=1$, $S_3=0$

overflow = $C_n \oplus C_{n-1}$

```
(00)000            (01)100
    1001               0111
    0010               0010
    ----               ----
    1011   ok ✓        1001    X

(11)100            (10)000
   0111               1001
   1110               1110
   -----              -----
   10101  ok ✓        10111    X
```

Subject assign فرق assign با بدون

Date

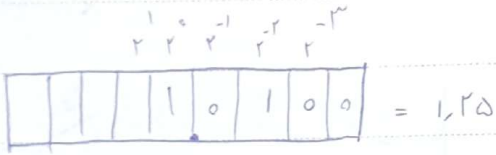
Binary coded decimal (BCD) به ۶ (۰۱۱۰) اضافه کن

۳۵

۰۰۱۱ ۰۱۰۱ ۰

Fixed-Point

Floating Point



B: $b_{n-1} \cdots b_1 b_0 \mid b_{-1} b_{-2} \cdots b_{-K}$ (n, K)

int ↵ ↳ float

$$V(b) = \sum_{i=-K}^{n-1} b_i \times 2^i$$

Floating Point نمایش

32 بیت:

Single percision

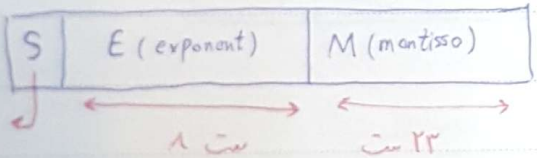
double percision 64 بیت

X = ۱۰۱۰۱ . ۰۱۰۱۱

Y = ۱۰۱۱۰ ۱۱۱۰۱۰

X+Y = ۰۰۰ ۱۰۱۰۱ . ۰۱۰۱۱

۱۰۱ ۱۰۱۱۱ . ۱۱۰۰۰



$V(B) = \pm 1, M \times 2^{(E-127)}$

$E = 0, \ldots, 255$

$exp = -127 \rightarrow 0$

$+128 \rightarrow \infty$

$2^{-127} \simeq 10^{-38}$

۳۸ ۳۸-

۱۰ → ۱۰

E = 11 bit

M = ۵۲ bit

$\pm 1.M \times 2^{(E-1023)}$

۳۰۸ ۳۰۸-

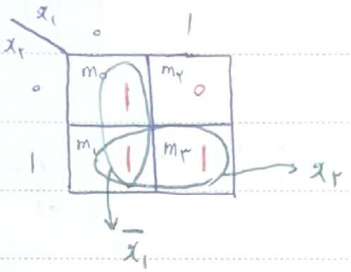
۱۰ → ۱۰

۶۴-bit

Subject
Date

روش های ساده سازی
- جبر بولین
- جدول کارنو
- روش Quine-Mccluskey
- استفاده از مفهیم اجماع
- روش های BDD

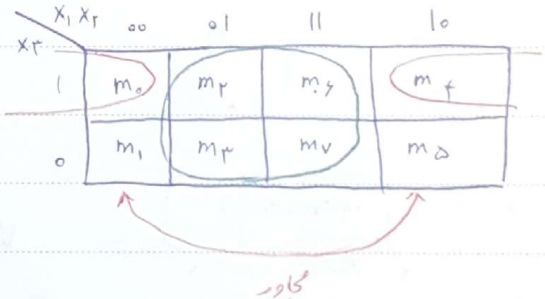
x_1	x_2	
0	0	m_0 1
0	1	m_1 1
1	0	m_2 0
0	0	m_3 1



$$f = \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + x_1 x_2$$
$$= \bar{x}_1 + x_2$$

0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

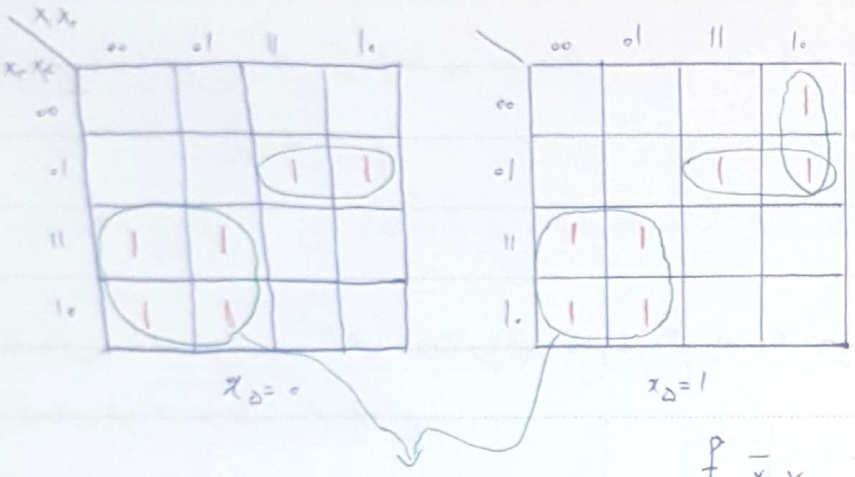
دسته های ۲ تایی یا ۴ تایی یا ۸ تایی ...



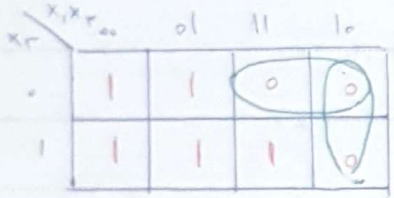
اگر کل اعداد رو پوشش داد دیگه لازم نیست چیزی اضافه کنی ★ جواب لزوما یکتا نیست

سوال کوییز

* تابع زیر را با کمترین هزینه پیاده سازی کنید $f(x_1,x_2,x_3,x_4) = \Sigma m(4,7,8,10,11,12,15) + d(3,5,9)$



$$f = \bar{x}_1 x_3 + x_1 \bar{x}_3 x_4 + x \bar{x}_2 \bar{x}_3 x_5$$



کارنو برای POS

$$f_{pos} = (\bar{x}_1 + x_3)(\bar{x}_1 + x_4) \rightarrow 9$$

$$f_{sop} = \bar{x}_1 + x_2 x_3 \rightarrow 6$$

هزینه Cost یک عبارت بولین : تعداد gate + تعداد ورودی

$$f = x_1 \bar{x}_2 x_3 + x_1 x_2 + \bar{x}_1 x_3 x_4 x_5 = 16$$

(۴ ، ۳ ، ۵ ، ۴)

Not را حساب نمی‌کنیم

اگر یک تابعی به ما دادند و گفتند با کارنو مینیمایز کنید باید هم Sop و هم pos را حساب کنید و کمترین عدد را حساب کنید

۱) f_{sop}

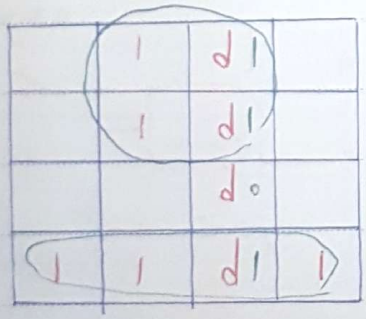
۲) f_{pos}

۳) Cos → min Cost

جدول ۶ تایی

do n't care : یعنی هم نیست و دست بازه

$$f = \sum m(2,4,5,6,10) + d(12,13,14,15)$$



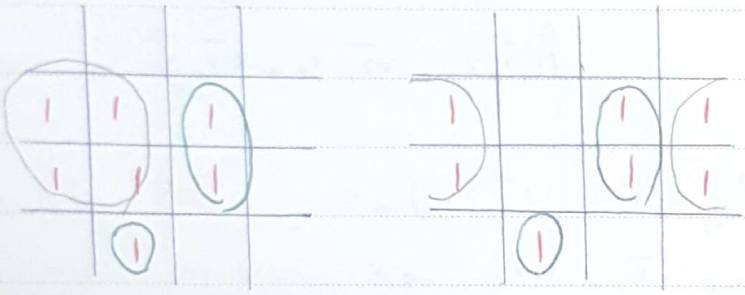
$$x_3 \bar{x}_4 + x_2 \bar{x}_3$$

Subject
Date

لزوما برای تعیین d در SOP و یا POS لازم نیست d ها یکی باشند چون برای SOP دنبال ۱ هستیم (می‌توانند یکسان باشند)
برای POS دنبال ۰ هستیم

$$f_{POS} \neq f_{SOP}$$

مدارهای با چند خروجی: گاهی اوقات دو یا چند تابع باید باهم پیاده‌سازی شوند. پیاده‌سازی مستقل آنها می‌تواند بسیار پرهزینه‌تر از حالتی شود که هر دو را باهم پیاده کنیم.



$f_1 = \bar{x}_1 x_4 + x_2 x_4 + \bar{x}_1 x_2 x_3 \rightarrow Cost = 14$

$f_2 = x_1 x_4 + \bar{x}_2 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 \rightarrow Cost = 10$

$\rightarrow 24$

سلول‌های مشترک

$$\Rightarrow \begin{cases} \bar{x}_1 x_4 + x_1 x_2 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 \rightarrow 14 \\ \bar{x}_2 x_4 + x_1 x_2 x_4 + \bar{x}_1 x_2 x_3 \bar{x}_4 \rightarrow 7 \end{cases} \quad \text{یا } 23$$

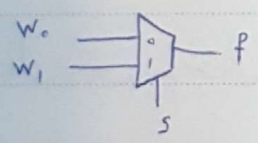
مدار ترکیبی

مالتی پلکر Multiplexer

یک MUX دارای تعدادی ورودی و تعدادی کنترل کننده و یک خروجی است. بسته به مقدار کنترل کننده یکی از ورودی‌ها به خروجی می‌رود.

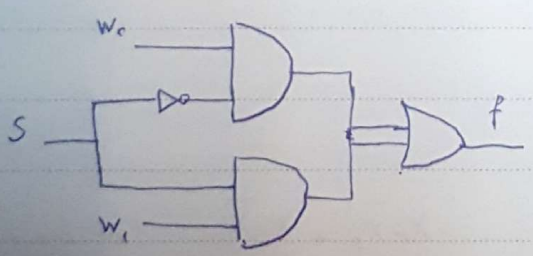
مثال:

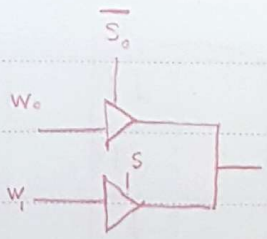
۲ to 1 MUX



s	f
0	w_0
1	w_1

$$f = \bar{s}_0 w_0 + s_0 w_1$$



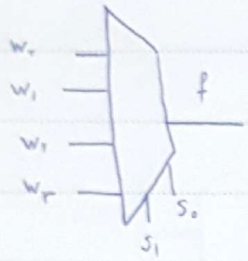


برای استیت بعدی

```
wire f
assign f = s ? w1 : w0;

reg f;
always @(w0, w1, s)
begin if (s==1)
    f = w1;
  else
    f = w0;
end
```

$$f = \begin{cases} x & s=1 \\ z & s=0 \end{cases}$$

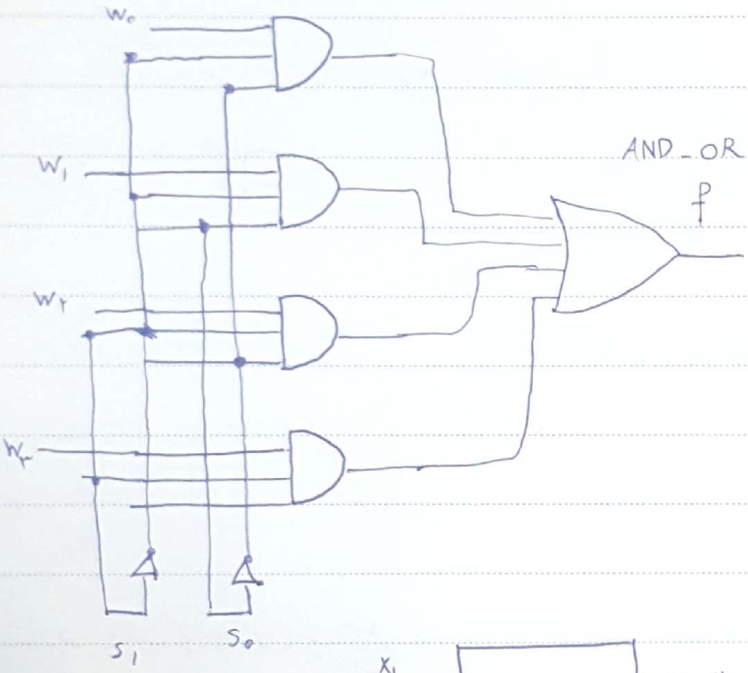


مثلا $f = \bar{S}w_0 + Sw_1$

$f = \bar{S}_0\bar{S}_1 w_0 + \bar{S}_1 S_0 w_1 + S_1\bar{S}_0 w_2 + S_1 S_0 w_3$

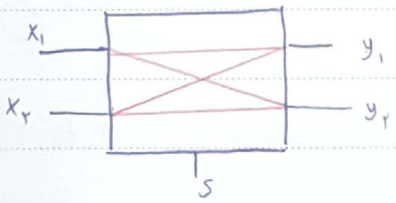
: mux

S_1	S_0	f
0	0	w_0
0	1	w_1
1	0	w_2
1	1	w_3



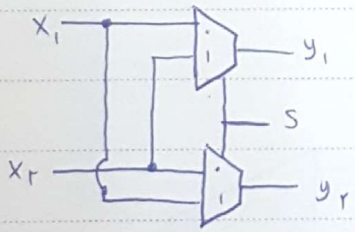
بنابراین MUX در حقیقت نوعی پیاده سازی AND-OR درصفحه است.

مثال: Crossbar, switch

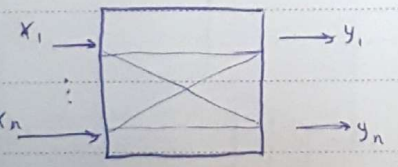


$S=0 \Rightarrow \begin{cases} x_1 = y_1 \\ x_2 = y_2 \end{cases}$

$S=1 \Rightarrow \begin{cases} x_1 = y_2 \\ x_2 = y_1 \end{cases}$



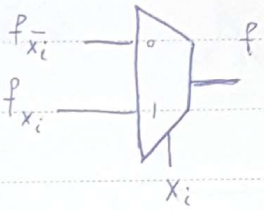
عکس ★



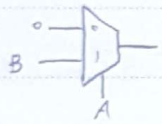
Subject
Date

پیاده سازی توابع با استفاده از mux :

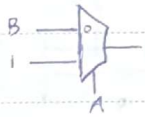
Shenon قضیه $f(x_1, \dots, x_n) = \bar{x}_i f_{\bar{x}_i} + x_i f_{x_i}$

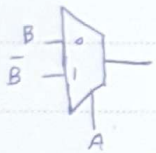


$f = AB = AB + A.0$



$f = A + B = A.1 + \bar{A}.B$



$f_{XOR} =$ 

$f = \bar{w}_1 w_2 + w_1 \bar{w}_2$

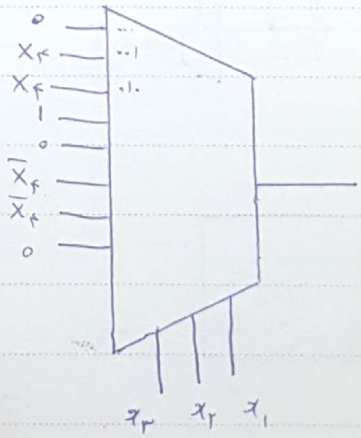
$f = ABC = A(BC) + \bar{A}(0) = A\left(B(C) + \bar{B}(0)\right) + \bar{A}(0)$ ← بهترین حالت = ساده سازی ← فرق می کند نسبت به کدام متغیر جا w_2

در حالت کلی می توان یک تابع n متغیره توسط یک mux 2^{n-1} to 1 پیاده سازی کرد.

مثال: تابع زیر را با یک mux 8 to 1 پیاده سازی کنید.

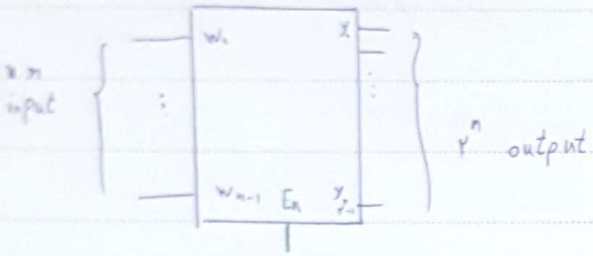
$f = (x_4, x_3, x_2, x_1) = \sum m(3, 5, 6, 9, 10, 11)$

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{x}_4$	0	1	2	(3)	4	(5)	(6)	7
x_4	8	(9)	(10)	(11)	12	13	14	15



Subject
Date

دکودر Decoder : یک دکودر برای Decode کردن اطلاعات کدشده استفاده می‌شود.



دارای n ورودی و 2^n خروجی و یک سیگنال Enable است ورودی دارای 2^n حالت مختلف است.
در حقیقت دکودر به ازای هر حالت ورودی، تنها یکی از خروجی ها را "۱" می‌کند و بقیه صفر می‌باشند

مثال: دکودر ۲ به ۴ (2 to 4)

E_n	w_1	w_0	
1	0	0	1 0 0 0
1	0	1	0 1 0 0
1	1	0	0 0 1 0
1	1	1	0 0 0 1
0	X	X	0 0 0 0

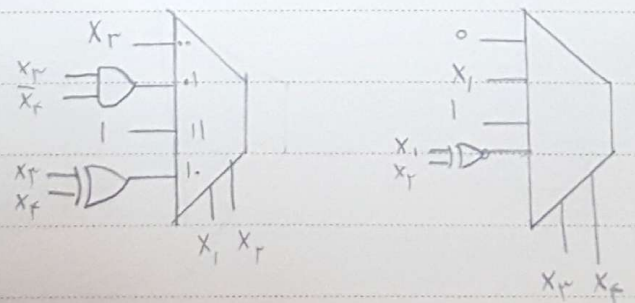
همچنین

وقتی $E_n = 0$ ، خروجی ها همه صفر هستند
خروجی one-hot است

سوال کوییز: با استفاده از یک MUX 4 to 1 و حداقل تعداد گیت های دیگر تابع زیر را بسازید

$$f(x_1, x_2, x_3, x_4) = \sum m(2, 3, 6, 9, 10, 13, 15) + \sum d(0, 12, 14)$$

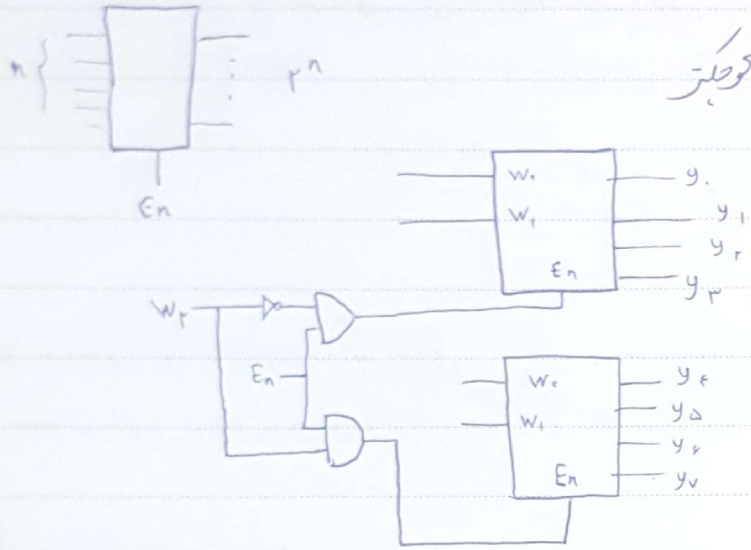
x_3x_4 / x_1x_2	00	01	11	10
00	d	0	1	1
01	0	0	0	1
11	d	1	1	d
10	0	1	0	1



Subject
Date

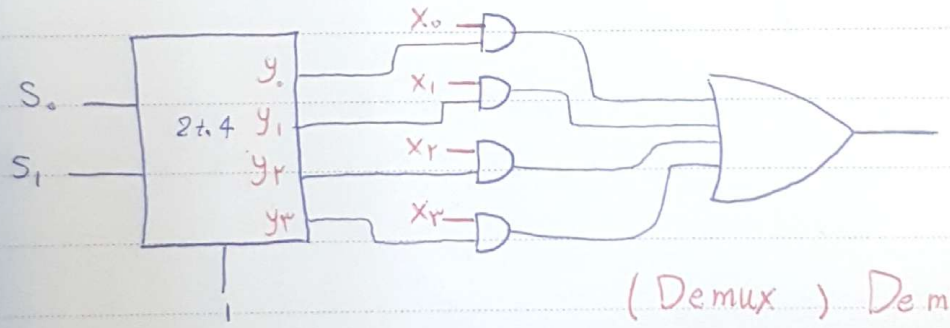
Decoders

ساخت دکودر بزرگتر، از روی دکودهای کوچکتر



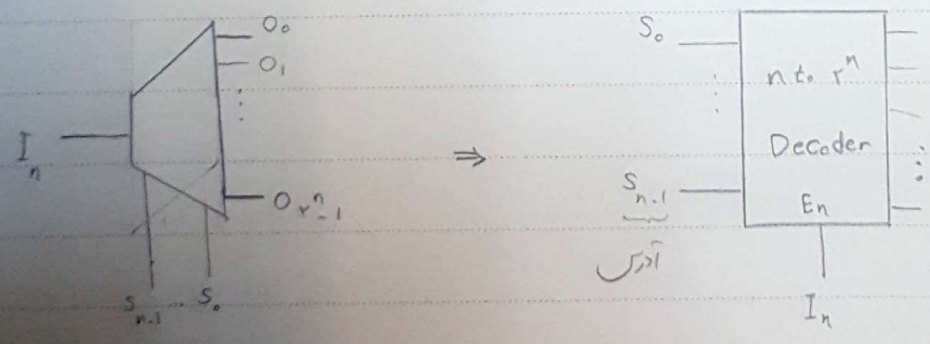
$f = \Sigma m(0,1,2,3,4,5,7)$

مثال: با استفاده از یک دکودر 2 to 4 یک mux 4 to 1 بسازید

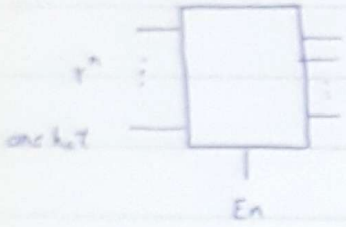


Demultiplexer (Demux)

عکس کار mux را انجام می دهد یعنی قرار دادن سیگنال ورودی بر روی یکی از 2^n خروجی ممکن برحسب سیگنال های کنترل کننده.



Encoder : عکس decoder



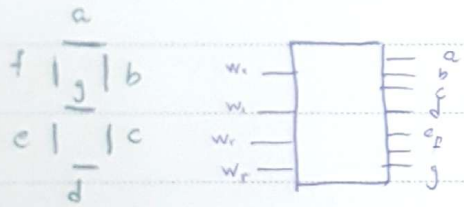
ورودی	خروجی
۰۰۰۱	۰۰
۰۰۱۰	۰۱
۰۱۰۰	۱۰
۱۰۰۰	۱۱

Priority Encoder : ورودی ها طرای اولویت هستند . priority

W_3 W_2 W_1 W_0	y_1 y_0 z
۰ ۰ ۰ ۰	d d ۰
۰ ۰ ۰ ۱	۰ ۰ ۱
۰ ۰ ۱ X	۰ ۱ ۱
۰ ۱ X X	۱ ۰ ۱
۱ X X X	۱ ۱ ۱

CODE CONVERTER

BCD_to_7seg



W_3 W_2 W_1 W_0	a b c d e f g
۰ ۰ ۰ ۰	۱ ۱ ۱ ۱ ۱ ۱ ۰
⋮	⋮
۱ ۰ ۰ ۱	۱ ۱ ۱ ۱ ۰ ۱ ۱

Binary to Gray

b_2 b_1 b_0	g_2 g_1 g_0
۰ ۰ ۰	۰ ۰ ۰
۰ ۰ ۱	۰ ۰ ۱
۰ ۱ ۰	۰ ۱ ۱
۰ ۱ ۱	۰ ۱ ۰
۱ ۰ ۰	۱ ۱ ۰
۱ ۰ ۱	۱ ۱ ۱
۱ ۱ ۰	۱ ۰ ۱
۱ ۱ ۱	۱ ۰ ۰

$g_2 = b_2$

$g_1 = b_1 \oplus b_2$

$g_0 = b_0 \oplus b_1$

مثال:

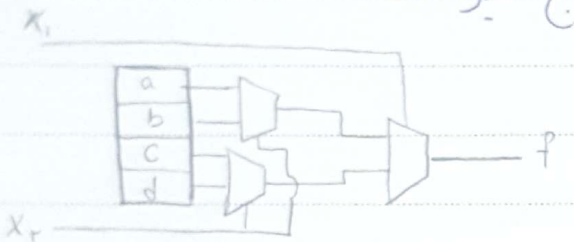
۰ ۰ ۰	
۰ ۰ ۱	
۰ ۱ ۱	
۰ ۱ ۰	
۱ ۱ ۰	
۱ ۱ ۱	
۱ ۰ ۱	
۱ ۰ ۰	

پیاده‌سازی براساس LUT Look up table

یک حافظه‌ی ۱ بیتی را در نظر بگیرید که مقدار "۰" یا "۱" را می‌تواند در خود حفظ کند.

یک LUT مجموعه‌ای از حافظه‌های ۱ بیتی که توسط آن توابع مختلف را پیاده‌سازی می‌کند.

مثال: یک 2-input LUT قادر به پیاده‌سازی هر تابع ۲ متغیره است.



X_1	X_2	
0	0	a
0	1	b
1	0	c
1	1	d

مثال:

1
0
0
1

$\Rightarrow f = \overline{X_1}\,\overline{X_2} + X_1 X_2$

1	0
0	1

با تغییر مقادیر ذخیره شده در LUT، تابع نیز تغییر می‌کند.

هر تابع K متغیره را با یک K-input LUT می‌توان پیاده‌سازی کرد.

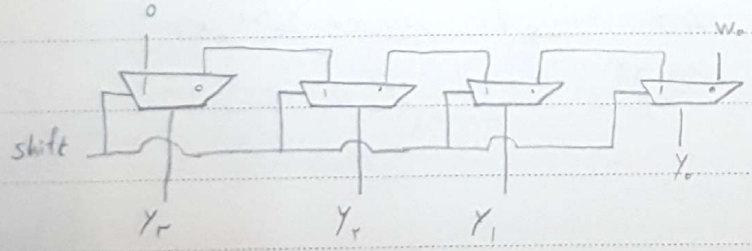
در اکثر FPGA های موجود در بازار $K = 4, 5, 6$

shift = 0 → No action

shift = 1 → right shift

مدار: شیفت به راست ورودی: W_3, W_2, W_1, W_0

خروجی: Y_3, Y_2, Y_1, Y_0



Subject:
Date:

barrel shift

حالت کلی shift که تعداد shift آن قابل برنامه‌ریزی و کنترل است.

S_1	S_0	Y_3 Y_2 Y_1 Y_0
0	0	W_3 W_2 W_1 W_0
0	1	W_0 W_3 W_2 W_1
1	0	W_1 W_0 W_3 W_2
1	1	W_2 W_1 W_0 W_3

assign $Y = \{\{X_1, X_0\}, \{X_3, X_2\}\}$

$Y = X >> 2;$

0 0 X_3 X_2

مدار در صفحه ۳۷۴ کتاب وا

پیاده‌سازی برایس: SOP (AND-OR)، POS (OR-AND)

۲ MUX

۳ LUT

۴ فقط NAND، فقط NOR

OAI, AOI

NAND ⇒ دلیل تایمینگ

۴) گیت کامل، گیتی که بتوان آن گیت‌های AND، OR، NOT ساخت.

گیت XOR، XNOR کامل نیستند

پیاده‌سازی برایس NAND، NOR

NAND → AND-OR + double noting

NOR → OR AND + double noting

$f = AOI \rightarrow \bar{f}$ پیاده‌سازی با AO (AND-OR SOP) $\rightarrow \bar{f}$ در کانر "۱" هایی پیاده‌سازی می‌کنیم

در انتها NOT می‌کنیم

OAI (OR_AND inverter)، AOI (AND_OR inverter)

PAPCO

$f = \sum m(4, 2, 7, 8, 14, 10) \rightarrow$ AoI

	1		1
	1	1	
	1	1	

f

1		1	
1	1	1	1
1			1
1			1

$\bar{f}$

$\bar{f} = x_4 x_3 + \bar{x}_2 x_1 + \bar{x}_3 x_2 + x_4 x_2 x_1$

$\Rightarrow f = \overline{(\quad)}$

Standard cell

AoI 321

$((a.b.c) + (d.e) + f)$

NAND_AND $\Rightarrow$ NOR

NAND_OR + NOT

$\Rightarrow$

تابع زیر با معادلات NAND_AND , NOR_OR پیاده سازی کنید.

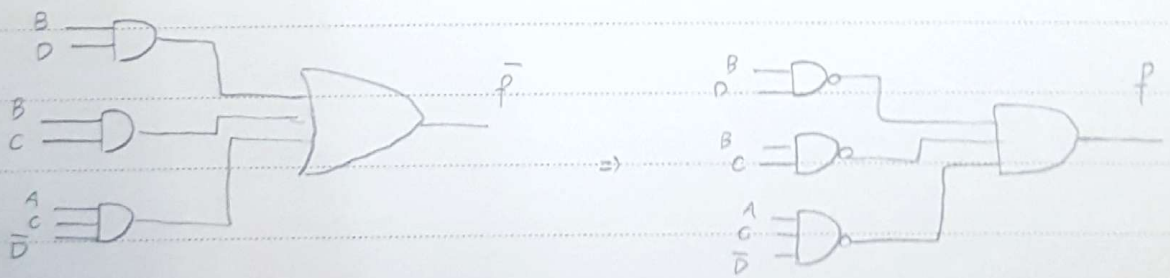
$f = \sum m(0, 1, 2, 3, 4, 8, 9, 12)$

A, B, C, D

MSB ↵

	1	1	
	1	1	
	1	1	1

$\bar{f} = BD + Bc + Ac\bar{D}$



Subject:
Date

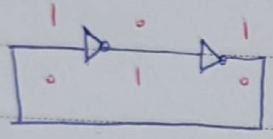
مدارهای ترتیبی : memory elements

المان های حافظه

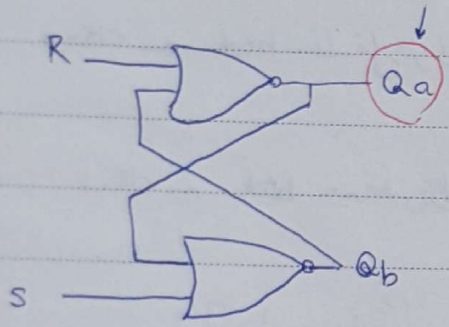
۱) ساده ترین المان

تعداد زوج المان NOT

مشکل اصلی : راحت نبودن تغییر مقدار حافظه



۲) SR latch



	S	R	Qa	Qb
حفظ	0	0	0/1	1/0
reset	0	1	0	1
set	1	0	1	0
ورودی	1	1	0	0
			نامعلوم	

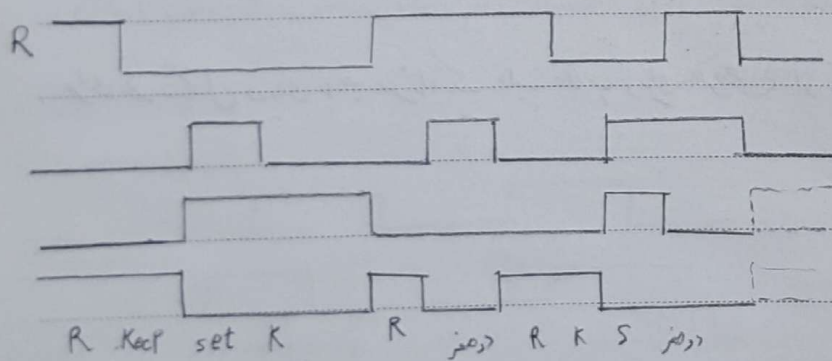
Active High

$$Q_a = \overline{(Q_b + R)} = \overline{Q}_b \cdot \overline{R}$$

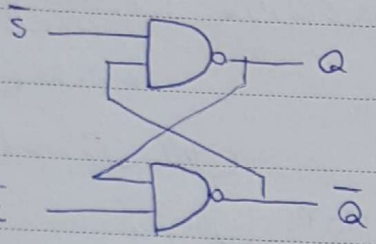
$$Q_b = \overline{(Q_a + S)} = \overline{Q}_a \cdot \overline{S}$$

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	نامعین

$R=0, S=1 \rightarrow Q_b=0 \Rightarrow Q_a=1$

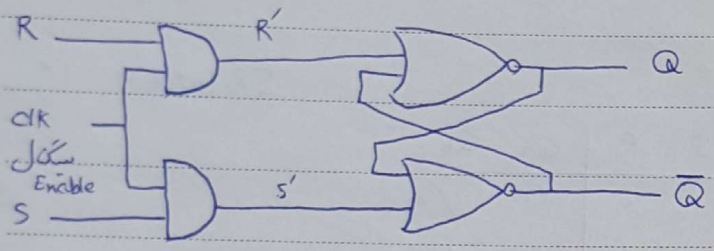


نامعین به دلیل تاخیر داخلی گیت NOR



$\bar{S}$	$\bar{R}$	$Q(t+1)$
0	0	نامعین
0	1	1
1	0	0
1	1	$Q(t)$

Active low

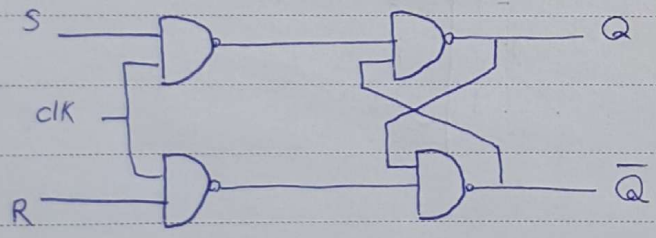


وقتی clk=0 ← latch مقدار قبل خود را حفظ می‌کند

وقتی clk=1 ← latch مانند حالت نرمال خود عمل می‌کند

level sensitive

clk	S	R	$Q(t+1)$
0	X	X	$Q(t)$
1	0	0	$Q(t)$
1	0	1	0
1	1	0	1
1	1	1	X



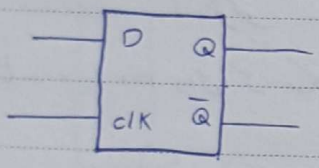
level Sensitive به سطح clk وابسته است

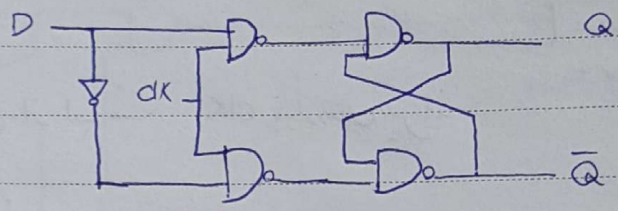
برای هر سیگنال ورودی باید سر راه یک فلیپ فلاپ و برای خروجی از یک فلیپ فلاپ باشد

Subject:
Date

Gated D latch

نوع پرکاربردتر latch ی باشد که دارای دو ورودی و دو خروجی ی باشد



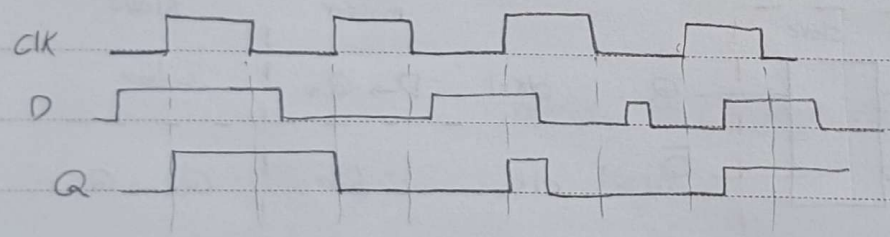


SR latch

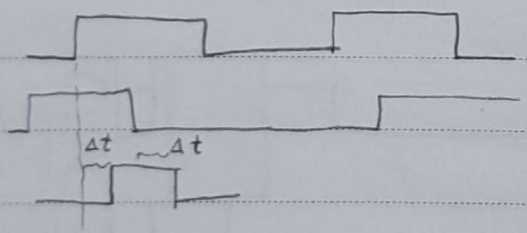
ورودی S, R مکمل هم هستند لذا حالت نامطلوب S=R=1 اتفاق نمی افتد

$$clk=1 \begin{cases} D=1 \Rightarrow \begin{cases} S=1 \\ R=0 \end{cases} \Rightarrow \begin{cases} Q=1 \\ \bar{Q}=0 \end{cases} \\ D=0 \Rightarrow \begin{cases} S=0 \\ R=1 \end{cases} \Rightarrow \begin{cases} Q=0 \\ \bar{Q}=1 \end{cases} \end{cases}$$

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

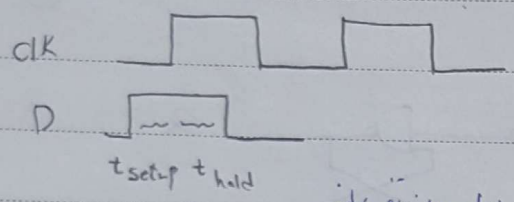


نکته: در دنیای واقع، گیت ها دارای یک تأخیر به نام Propagation dday هستند. این بدین معناست که وقتی clk=1 است خروجی Q با یک تأخیری، تغییرات ورودی را دنبال می کند.



Subject:
Date

Setup time



$$\tau_{clk} \geq \tau_{latch} + \tau_{logic} + \tau_{setup}$$

ترکیبی

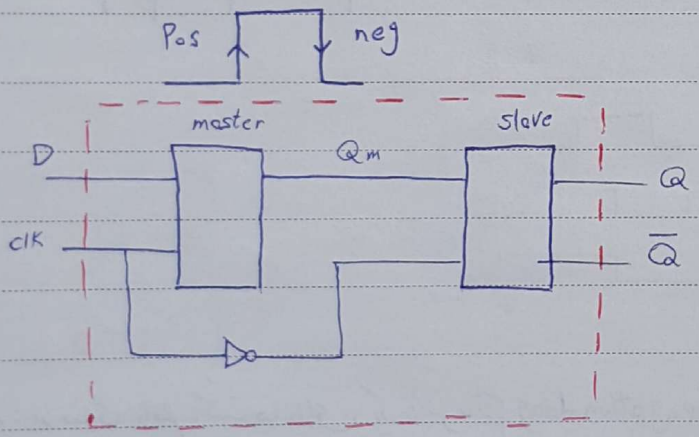
t_{setup} : min زمانی که سیگنال D باید قبل از لبه بیت clk پایدار و بدون تغییر باشد

t_{hold} : min زمانی که سیگنال D باید بعد از لبه بیت clk پایدار و بدون تغییر باشد

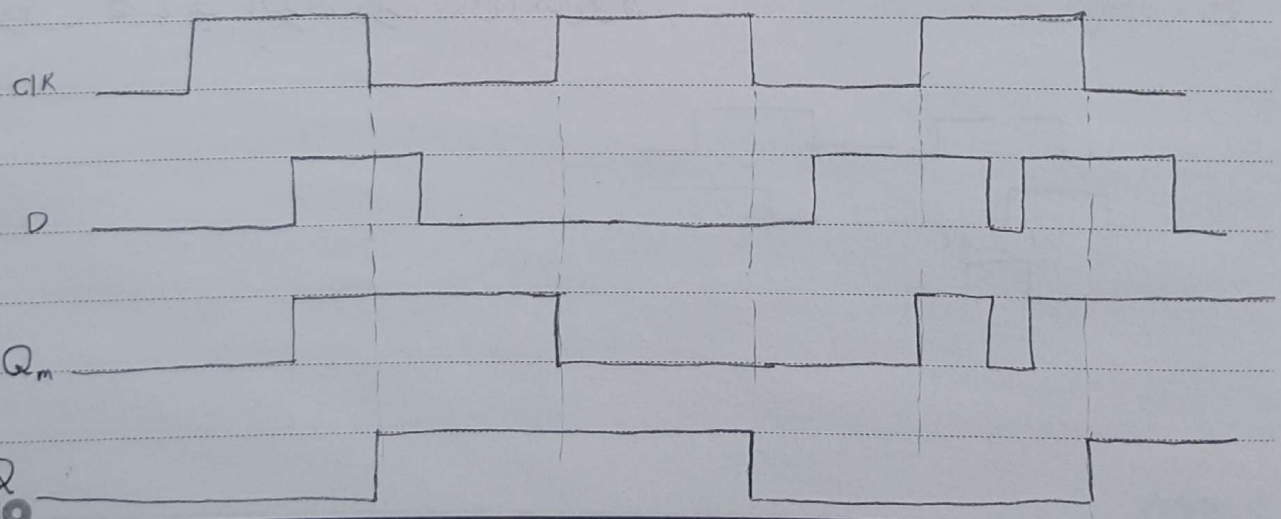
Master slave D_FlipFlop

دیدیم که latch ← level sensitive بود

خیلی موارد نیاز به رجیسترهایی داریم که به لبه‌ی بالارونده یا پایین‌رونده clk حساس باشند



	master	slave
clk=1	$D \rightarrow Q_m$	حفظ مقدار
clk=0	حفظ مقدار	$Q_m \rightarrow Q$

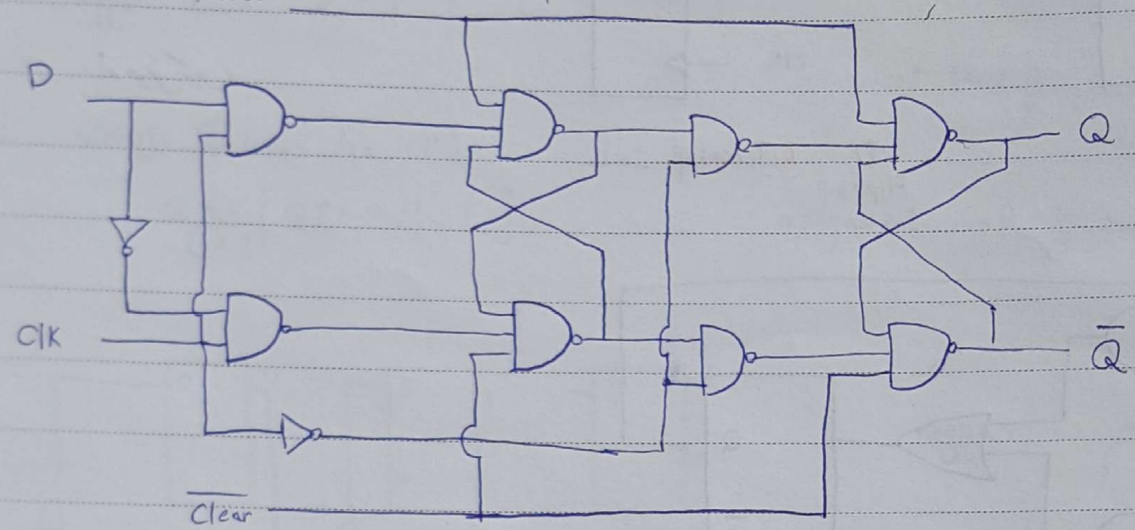


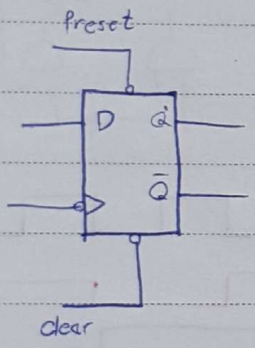
PAPCO

Subject :
Date

مطلوبست توانیم مقدار یک FF را به "0" یا "1" مقداردهی کنیم

Active low ← $\overline{\text{Preset}}$ (set)





$\overline{\text{clear}} = 0 \Rightarrow Q = 0$

$\overline{\text{Preset}} = 0 \Rightarrow Q = 1$

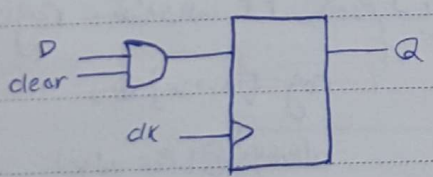
O.W ⇒ normal FF

وریلاگ :

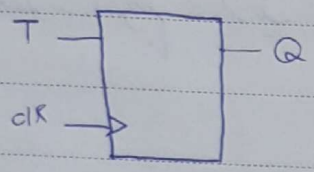
```
always @( posedge clk, negedge clear)
begin
    if( clear == 0)
        Q <= 0;
    else
        Q <= D;
end
```

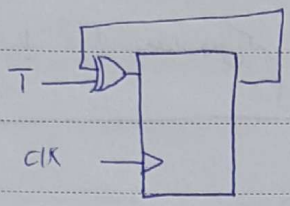
آسنکرون

برای سنکرون شدن negedg را حذف می‌کنیم



Subject:
Date



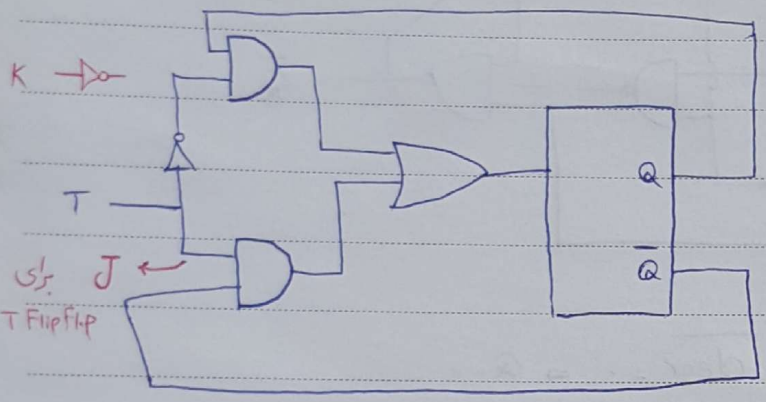


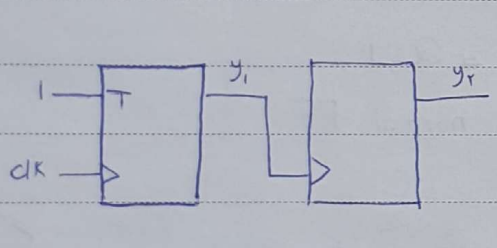
T Flip Flop

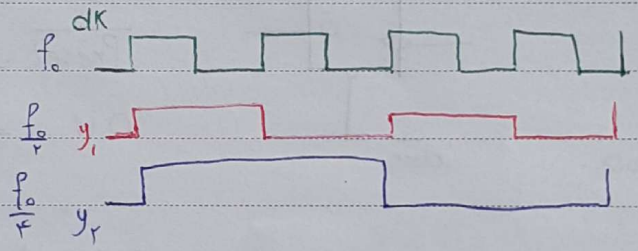
Toggle

لچ عوض کردن

ساخت T Flip Flop با D Flip Flop





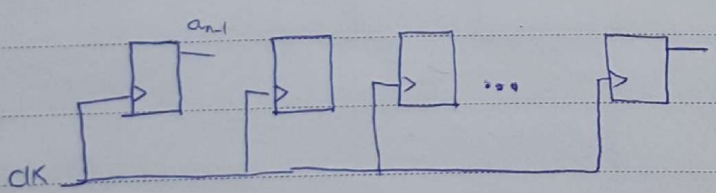


JK Flip Flop

مشابه T Flip Flop قابلیت set ، reset کردن را هم دارد

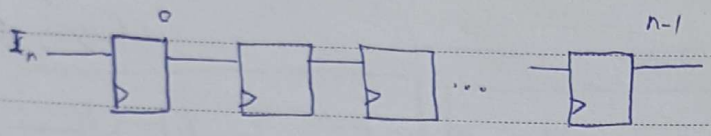
J	K	Q (t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	$\bar{Q}$(t)

register : تعدادی (n) FF که کنار هم کلاک یکسان می‌خورند و n-بیتی می‌سازند




```
reg [n-1:0] Q;
always @(Pos edge)
Q<=D;
```

Subject:
Date

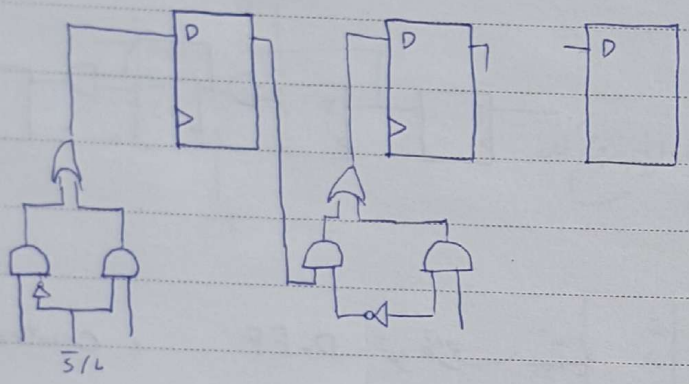


shift register

در مدارهای ترکیبی : assignment blocking

در مدارهای ترتیبی : assignment nonblocking

```
always @(Posedge clk)
    Q <= {Q[n-2:0], In};
   [n-1:0]
```

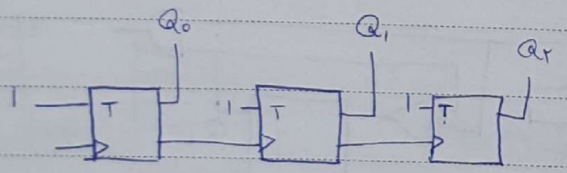


هم shift register هم register

Counter (شمارنده)

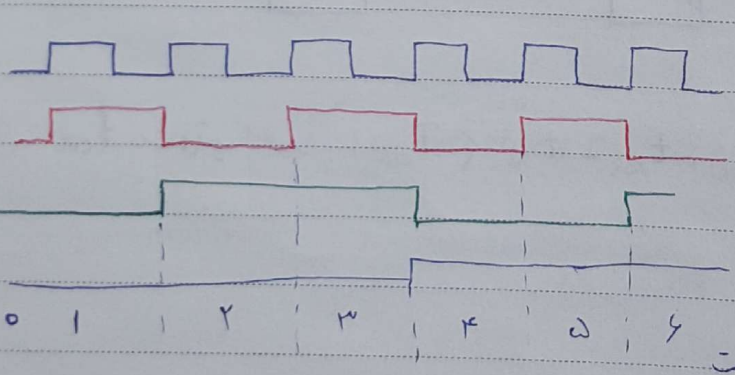
دو نوع Counter داریم :

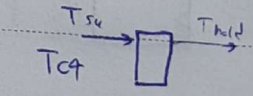
- سنکرون ← همه FF ها توسط clk مدیریت می‌شوند
- آسنکرون ← FF های میانی توسط خروجی های میانی کلاک می‌شوند



مثال :

3-bit up Counter آسنکرون





delay هم دارد به علت گیت های داخلی

delay زیادی دارد

خوب تجمیعی و برای n بزرگ اصلاً شمارنده خوبی نیست

Subject:
Date

clk	Q_2	Q_1	Q_0
۰	۰	۰	۰
۱	۰	۰	۱
۲	۰	۱	۰
۳	۰	۱	۱
۴	۱	۰	۰
۵	۱	۰	۱
۶	۱	۱	۰
۷	۱	۱	۱
۸	۰	۰	۰

Counter شمارنده :

مشاهدات :

۱) Q_0 در هر کلاک toggle می‌کند

۲) Q_1 وقتی $Q_0=1$ toggle می‌کند

۳) Q_2 وقتی $Q_1=1, Q_0=1$ toggle می‌کند

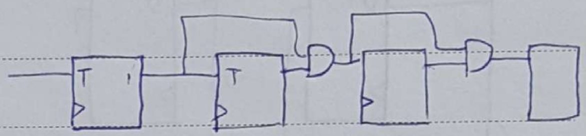
$T_0 = 1$

$T_1 = Q_0$

$T_2 = Q_1 Q_0$

$\vdots$

$T_n = Q_{n-1} \cdots Q_0$



Counter با D-FF : نحوه طراحی ← آینده   نتیجه DFF in

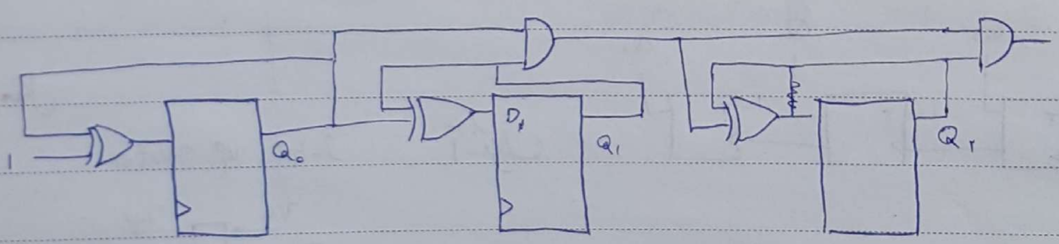
$D_0 = 1 \oplus Q_0$

$D_1 = Q_1 \oplus Q_0$

$D_2 = Q_2 \oplus Q_1 Q_0$

$\vdots$

$D_i = Q_i \oplus Q_{i-1} Q_{i-2} \cdots Q_1 Q_0$

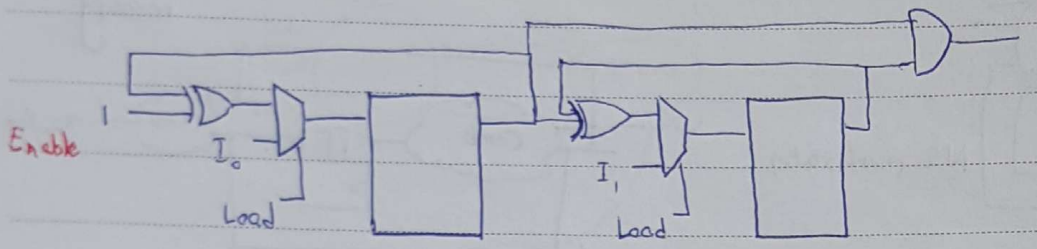


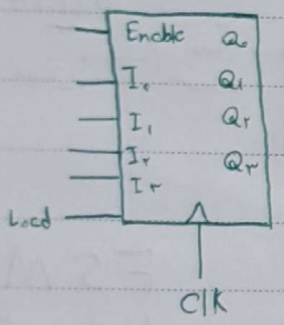
$T_{cp} = t_{cq} + 2t_{AND} + t_{XOR} + t_{su} < T_{clk} \Rightarrow$ f_{clk} بدست می‌آید

PAPCO

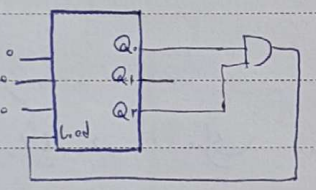
Subject :
Date

Counter با لود موازی :

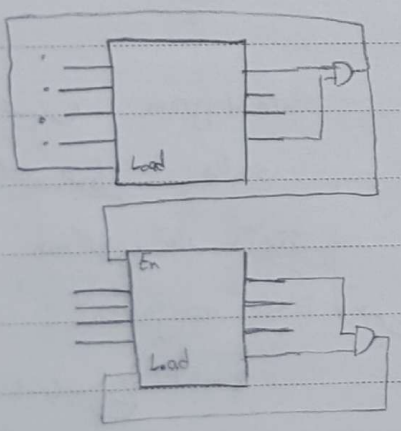




مداری طراحی کنید که شمارنده‌ای باشد که وقتی به ۵ می‌رسد از اول شروع کند

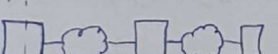


مدار شمارنده BCD از ۰ تا ۹۹ و اگر به ۹۹ رسید صفر شود.



طراحی مدارهای
- ترکیبی
- ترتیبی
  - سنکرون
  - آسنکرون
- (ترتیبی - ترکیبی)

طبقه FF



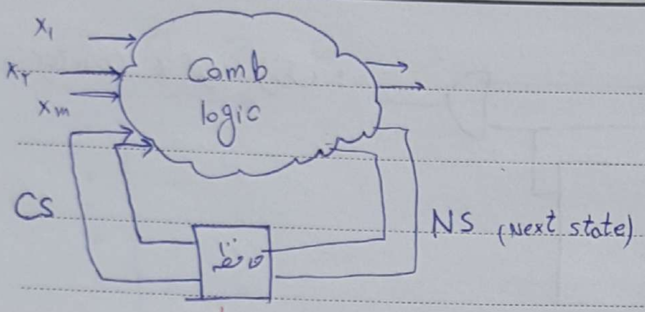
حافظه مدار

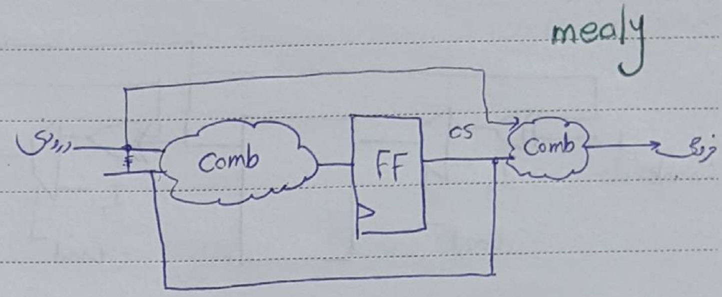
سنکرون ← mealy خروجی هم به ورودی هم به CS (current state) تنگی دارد

mealy ↓

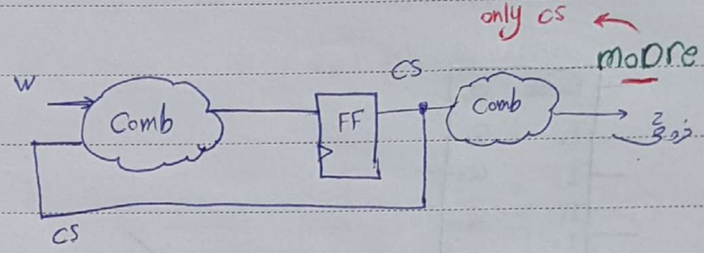
moore خروجی فقط به CS تنگی دارد

Subject:
Date



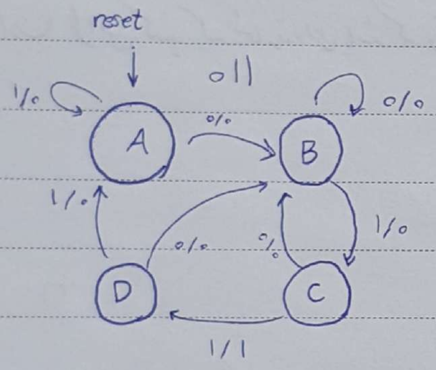


آسنکرون ⇐ مدار ترکیبی داریم
سنکرون ⇐ FF ها



FSM

Finite state machine



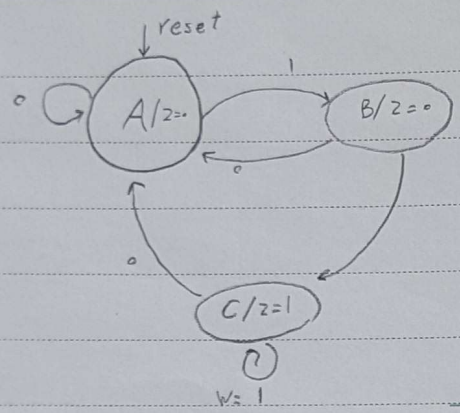
مثال: مداری طراحی کنید که یک ورودی دارد و زمانیکه دو "1" متوالی در ورودی ببیند خروجی را "1" کند.

	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
ورودی	0	1	0	1	1	0	1	1	1	0	1
خروجی	0	0	0	0	0	1	0	0	1	1	0

1) کشیدن state diagram
2) ایجاد جدول state
3) min کردن تعداد state
4) state assignment و تعیین مدارهای ترکیبی لازم برای خروجی و NS
5) انتخاب نوع FF
6) پیاده سازی

Subject:
Date

۱) کشیدن state diagram :

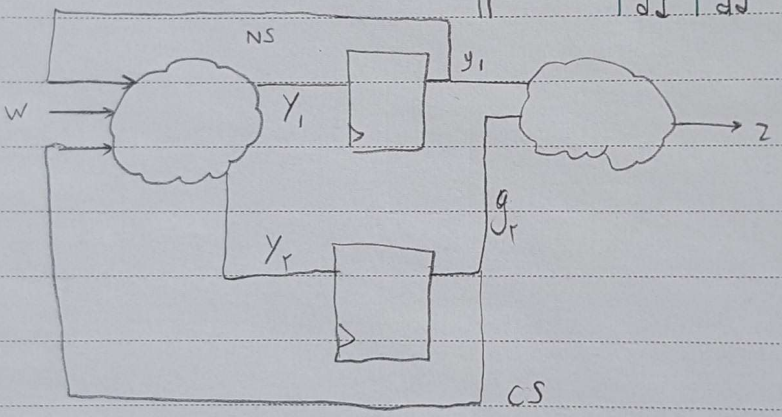


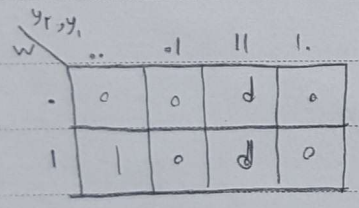
۲) state table

CS	NS w=0	NS w=1	z
00 ← A	00 A	01 B	0
01 ← B	00 A	10 C	0
10 ← C	00 A	10 C	1
11	dd	dd	d

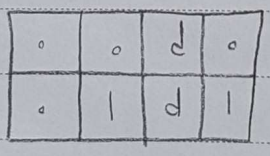
state تعداد n

$\lceil \log_2^n \rceil$ = FF تعداد

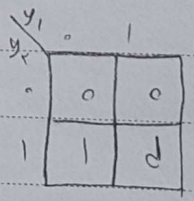




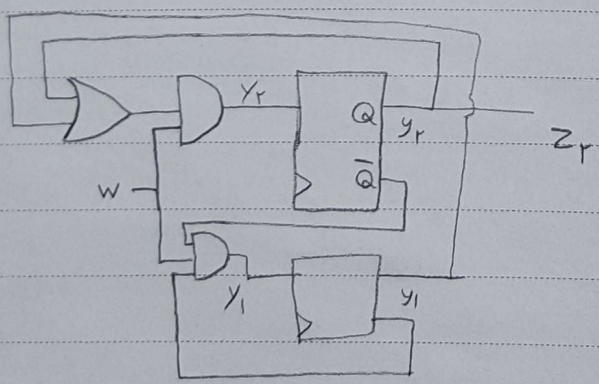
$Y_1 = w\bar{y}_1\bar{y}_2$



$Y_2 = w(y_1 + y_2)$



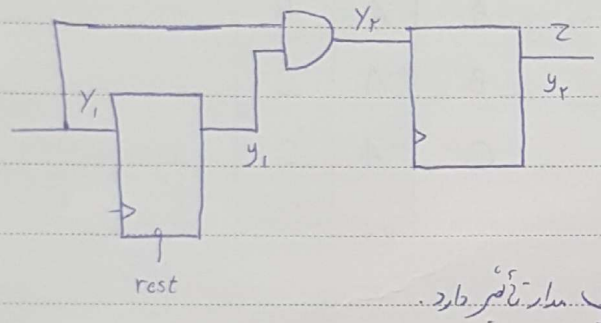
$z = y_2$



$y_1 y_2$	$w=0$	$w=1$	z
A ← 00	00 A	10 C	0
B ← 01	00 A	11 C	0
C ← 11	00 A	11 C	1

کاهش ⇐

state assignment



moore

$Y_1 = w$

$Y_2 = w y_1$

$z = y_2$

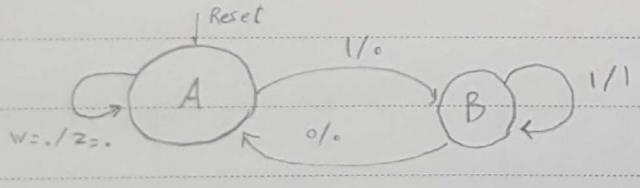
نتیجه ⇐ state assignment در هزینه پیاده‌سازی نهایی مدار تأثیر دارد.

خروجی در جایی سیگنال کلاکی که دومین "1" وارد می‌شود، "1" شود.

w_2: 0 1 0 1 1 0 1 1 1 0 1

z: 0 0 0 0 1 0 0 1 1 0 0

کمتر State ← طبیعتاً هزینه کمتر



$\lceil \log_2 n \rceil$ در کران بالا

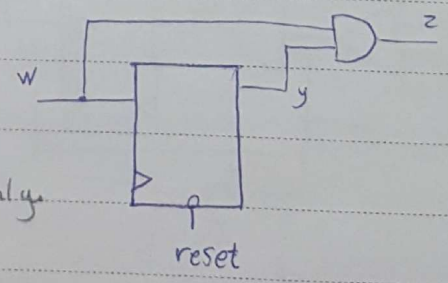
y	Y		z	
	0	1	0	1
A ← 0	A	B	0	0
B ← 1	A	B	0	1

$Y = w$

$z = wy$

۱- ورودی مستقیم به خروجی وصل شده

۲- تعداد طبیعتاً غالباً کمتری دارد.



mealy

در مدل mealy چون بیتر توصیف مدار روی Edge ها (یال ها) می باشد احتمالاً state کمتری دارد

تعداد state

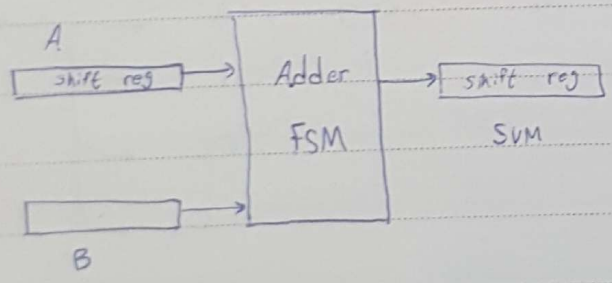
$$\left\lceil \log_2^n \right\rceil$$

moore → mealy
- یا تعداد خانه ها کمتری شود
- یا تعداد مدارهای ترکیبی کم می شود (به علت اینکه تعداد d ها بیتر می شود)

پیاده سازی جمع کننده سری : Serial Adder

در عدد بدون علامت $A = a_{n-1} \dots a_0$ ، $B = b_{n-1} \dots b_0$ را در نظر بگیرید با فرض اینکه جمع آنها $S = S_{n-1} \dots S_0$ شود می خواهیم مداری طراحی کنیم که جمع را بصورت سری انجام دهد.

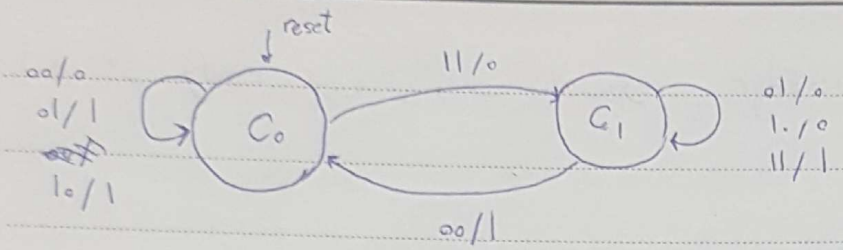
دو روش پیاده سازی ← moore
mealy



مدل Mealy :

در State در نظر می گیریم ← C_0 : Carry in از مرحله قبل صفر است

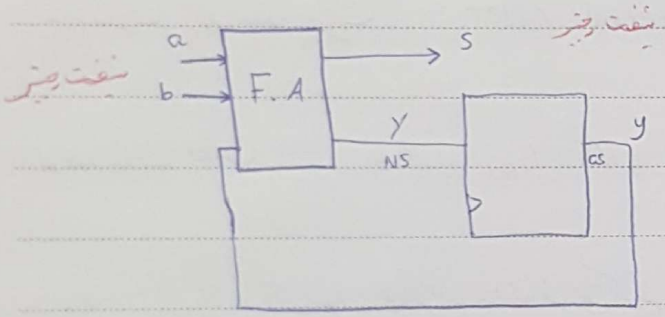
C_1 : Carry in از مرحله قبل یک است



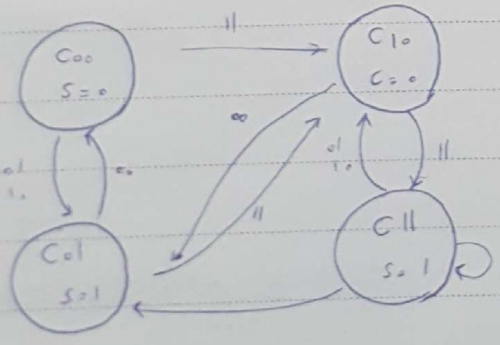
y	NS (Y) 00	01	10	11	S 00	01	10	11
C_0 ←	0	0	0	0	1	0	1	0
C_1 ←	1	0	1	1	1	0	0	1

$\Rightarrow Y = ab + ay + by$

$S = a \oplus b \oplus y$



Moore مدل



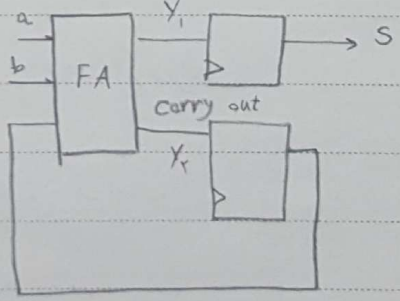
	00	01	10	11	S
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
01	01	10	10	11	1

Sam = 0 , Carry = 0 : C_{00} ← حالات

: C_{01} ←

: C_{10} ←

: C_{11} ←

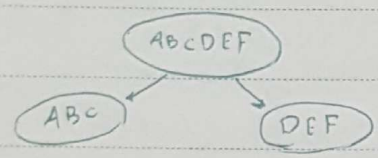


Subject:
Date

روش های کاهش تعداد state :

	NS		out	
	w=0	w=1	w=0	w=1
A	B	C	1	1
B	C	D	0	1
C	B	D	0	1
D	B	C	1	1

۱) partitioning
۲) implication graph
جدول استلزام



۱- روش Partitioning :

تعریف (K-Successor)

I) o-successor : در یک FSM، حالت S_i را در نظر بگیرید S_w را o-suc، حالت S_1 گوییم اگر به ازای ورودی w=0 از S_i به S_w رویم به همین ترتیب S_2 را 1-suc، حالت S_i گوییم اگر به ازای ورودی w=1 از S_i به S_2 رویم.

II) در یک FSM، دو حالت S_i و S_j را در حالت معادل (equivalent) گوییم اگر و تنها اگر به ازای تمام وضعیت های ورودی، دنباله خروجی آنها مستقل از اینکه از S_i یا S_j شروع کنیم یکسان باشد.

نتیجه: اگر دو حالت S_i و S_j معادل باشند آنگاه تمامی K-suc آنها نیز معادلند

روش Partitoning :

۱) ابتدا فرض می کنیم همه State ها معادل اند ← تشکیل P_1
۲) پس State ها را براساس یکسان بودن خروجی به ازای تمام حالت های ورودی از هم جدا می کنیم و آنهایی که دارای خروجی یکسان هستند را در یک بلوک قرار می دهیم ← تشکیل P_2
(یعنی State هایی که خروجی متفاوت دارند، نمی توانند با هم معادل باشند)

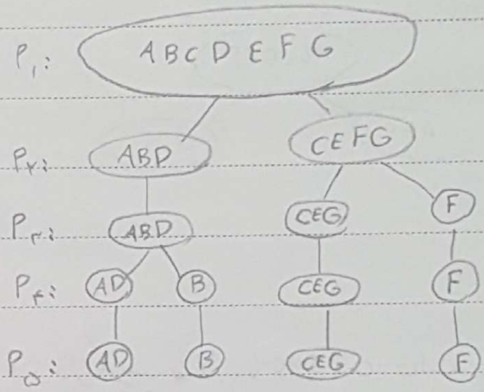
Subject:
Date

۳) پس از این Partition کردن را به این صورت ادامه می دهیم که در هر بلوک ، state های را که K_suc آنها در بلوک دیگری می باشند ، را از آن بلوک جدا کرده بلوک جدیدی تشکیل می دهیم

۴) مرحله سوم را تا زمانی که بلوک جدیدی ایجاد نشود ، ادامه می دهیم در پایان ، state هایی که داخل یک بلوک هستند معادلند

مثال:

CS	NS w=0	NS w=1	Z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	F	0
G	F	G	0



A و D معادلند
CEG معادلند

CS	NS 0	NS 1	Z
A	B	C	1
B	A	F	1
C	F	C	0
F	C	A	0

اگر FSM بصورت ناکامل توصیف شده باشد باید به ازای همه مقادیر ممکن مسأله حل شده و به ازای ورودی که کمترین مقدار نهایی را ایجاد کند مسأله پاسخ داده می شود.

۲) روش Imp graph

اگر یک FSM با n state را در نظر بگیریم (a_0 , ... , a_{n-1}) جدول زیر را تشکیل می دهیم

Subject:
Date

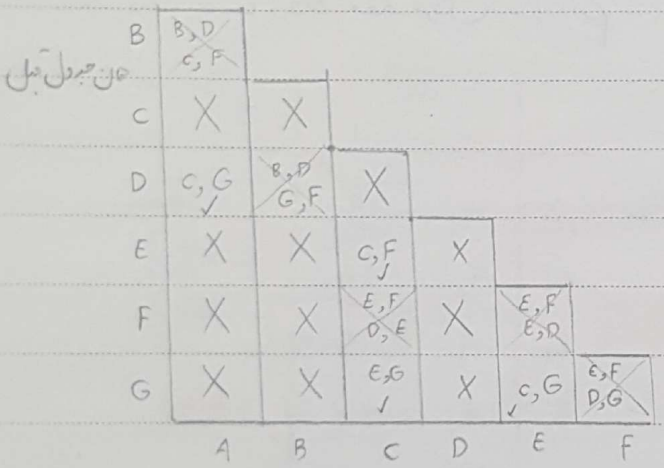
۱) در تمام خانه هایی که خروجی حالت های مربوطه شان متفاوت است، X می گذاریم

۲) برای بقیه حالت ها، اگر دو حالت به ازای تمام وضعیت های ورودی دارای حالت بعدی یکسان باشند، علامت ✓ می گذاریم

۳) اگر حالت های بعدی آنها یکسان نبودند، بررسی می کنیم در صورت معادل بودن چه حالت هایی این دو معادل می شوند و در جدول درج می کنیم.

۴) شرط ها را بررسی می کنیم چنانچه شرطی به زوج حلقی اشاره کند که در خانه متناظرشان X باشد در آن خانه نیز X می گذاریم بررسی شرط ها ادامه می دهیم تا جایی که تمام خانه ها دارای شرط ارضا شده یا X داشته باشند یا ✓ بخورند

۵) تمام خانه هایی که X ندارند معادلند

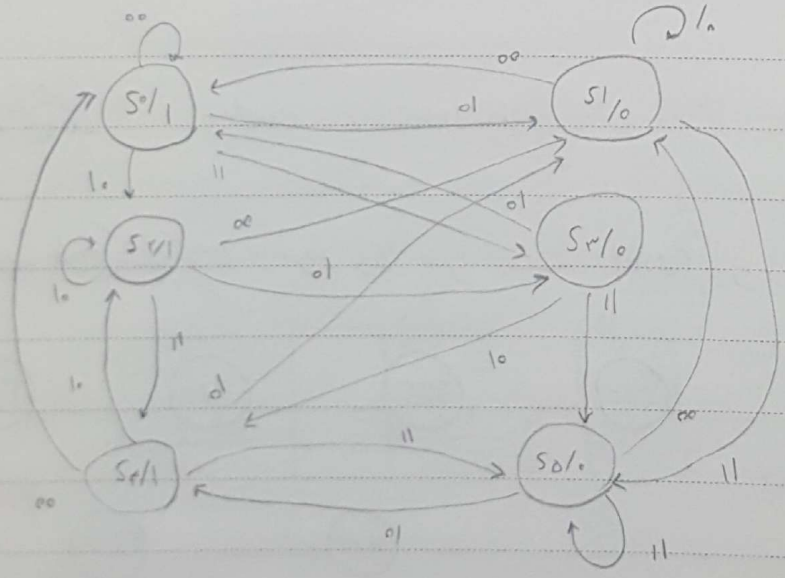


از جدول قبل

(C E G)

(A D

سوال کوئیز:



تبدیل مدارهای ملی به مور:

مثال:

برای تبدیل این مدار به مدار مور باید به ازای هر یک از حالت های A و B در حالت در نظر بگیریم

← هدف: استقلال حالت ها از ورودی ها

به ازای NS = A هردو خروجی ۰ و ۱ داریم.

NS = B هردو خروجی ۰ و ۱ را داریم

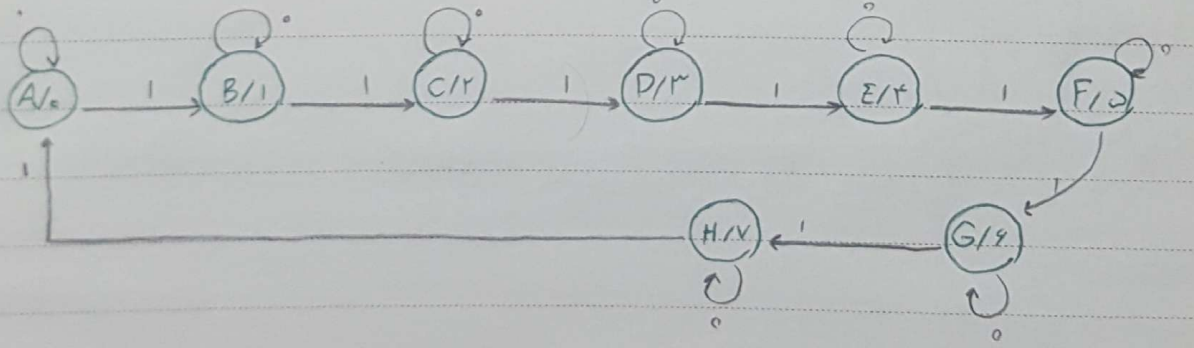
NS = C فقط خروجی ۰ را داریم.

CS	NS w=0	NS w=1	Z 0	Z 1
A	A	B	0	1
B	A	C	1	0
C	C	B	0	0

	NS 0	NS 1	Z
A_0	A_0	B_1	0
A_1	A_0	B_1	1
B_0	A_1	C	0
B_1	A_1	C	1
C	C	B_0	0

مثال: طراحی یک module - ۸ counter

شمارنده ای طراحی کنید که تا ۷ شمرده و به صفر ریست Reset شود و مجدداً بشمارد.



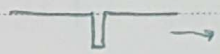
Subject:
Date

	CS	NS 0	NS 1	Z
000 ←	A	A	B	۰
001 ←	B	B	C	۱
	C	C	D	۲
	D	D	E	۳
	E	E	F	۴
	F	F	G	۵
	G	G	H	۶
111 ←	H	H	A	۷

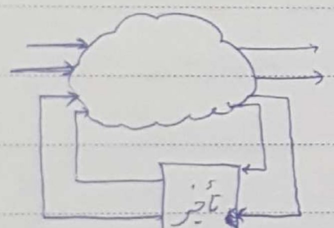
$Y_2 Y_1 Y_0$

جدول کارنو

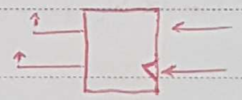
$$\begin{cases} y_0 = w \oplus y_0 \\ y_1 = w y_0 \oplus y_1 \\ y_2 = w y_0 y_1 \oplus y_2 \end{cases}$$



مدارهای آسنکرون: موجودی به نام کلاک نداریم، کسی معطل هیچ کس نیست



آسنکرون مثل latch



سنکرون

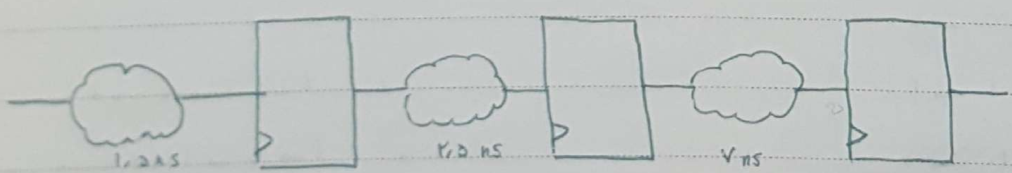
در آسنکرون مسابقه برای زودتر رسیدن داریم

۲ ورودی را همزمان نباید تغییر داد.

در مدارهای آسنکرون: ۱) تغییر در state توسط clock مدیریت نمی‌شود.

۲) تنها یکی از ورودی ها در آن واحد تغییر می کند

۱.۲) بعد از تغییر یک ورودی زمانی باید برای پایدار شدن state جدید بگذرد.

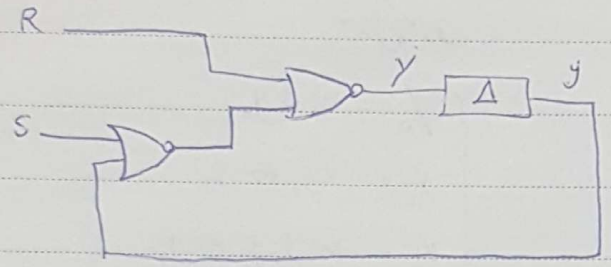


سنکرون: تأخیر: ۲۱ ns (۳×۷)

آسنکرون: تأخیر: ۱۱ ns (۱.۵+۲.۵+۷)

Subject:
Date

مثال: SR-latch



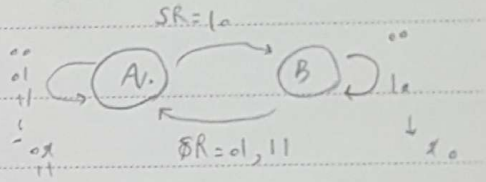
y: CS

Y: NS

$Y = S\bar{R} + \bar{R}y$

روش گام: برای استیت پایدار سطح هم نام را دورش خط می‌کشیم

CS y	SR= 00	01	10	11	Ns(Y)
A 0	(0)	(0)	1	(0)	پایدار
B 1	(1)	0	(1)	0	پایدار



$Y_1 = y_1\bar{y}_2 + w_1\bar{y}_2 + \bar{w}_1\bar{w}_2 y_1$

$Y_2 = y_1 y_2 + w_1 y_2 + w_2 + \bar{w}_1\bar{w}_2 y_1$

$z = \bar{y}_1 y_2$

مثال: transition table

CS y_2y_1	NS 00	01	10	11	z
00	(00)	01	10	-	0
01	11	(01)	-	11	0
10	00	(10)	(10)	(10)	1
11	(11)	10	10	10	0

Flow table جدول ← برد ... اگر A و B و

مقدمه ادل:

✓ 0 → X → 1

✓ 1 → X → 0

✓ 1 → 1 → 1

✓ 0 → 0 → 0

x 1 → 0 → 1 ⇒ glitch

x 0 → 1 → 0

✓ 1 → 1 → 0 → 0

x 1 → 0 → 1 → 0 → 0 → 0

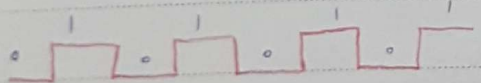
در A state تغییر مزدان در ورودی ۰۰ به ۱۱ غیر مجاز است
همینطور در B state تغییر مزدان از ۰۰ به ۱۰ غیر مجاز است
در C state ابتدا به نظر می رسد باید تغییر ۱۱ به ۰۰ منتقل را باشد اما چون در همان سطر حالت های پایدار دیگری هم وجود دارند که انتقال آنها به حالت ورودی ۰۰ مجاز است باید این حالت حفظ شود.
در D state تغییر ۰۰ به ۱۱ را حفظ می کنیم چون D پل ارتباطی انتقال بین B به C باشد ۱ → ۰ → ۰

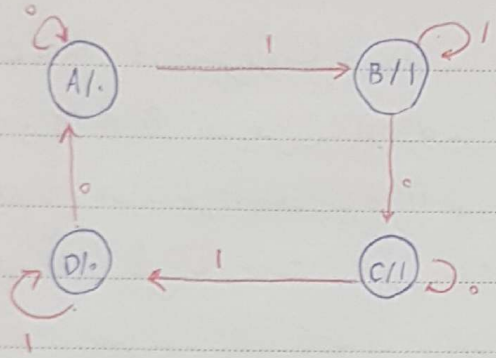
	۰۰	۰۱	۱۰	۱۱	
A	(A)	B	C	D	۰
B	D	(B)	D	D	۰
C	A	(C)	(C)	(C)	۱
D	(D)	C	C	C	۰

مراحل پیاده سازی مدارهای سنکرون:

۱) طراحی یک state diagram
۲) بست آوردن Flow table و کاهش تعداد state
۳) انتساب حالت ها state assignment و بست آوردن transition table
۴) بست آوردن عبارت بولین خروجی و Ns
۵) پیاده سازی مدار

مثال: serial parity generator
مداری طراحی کنید که با یک ورودی w و خروجی z به گونه ای که زمانیکه تعداد خروجی ۱ پس باید z=۰ و در غیر اینصورت z=۱





Flow table ای که در هر سطح یک حالت پایدار دارد ←

یعنی Primity flow table

جدول جریان

	w=0	w=1	z
A	(A)	B	0
B	C	(B)	1
C	(C)	D	1
D	A	(D)	0

State assignment ← باید دقت کنیم در تغییر حالت‌ها $y_2 y_1$ نداشته باشیم مثلاً

$y_2 y_1$	0	1	z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

$11 \rightarrow 00$ ✓

$11 \rightarrow 10 \rightarrow (10)$ ✗

race

assignment نامناسب مستلزم تغییر حالت دو متغیر $y_2 y_1$

assignment جدید:

y_2 y_1	Y_2	Y_1	z
0 0	(01)	01	0
0 1	11	(01)	1
1 1	(11)	10	1
1 0	00	(10)	0

جدول کارنو ⇐ $Y_1 = w\bar{y}_2 + \bar{w}y_1 + y_1\bar{y}_2$

مدار در کتاب

$Y_2 = wy_2 + \bar{w}y_1 + y_1y_2$

$z = y_1$

Subject:
Date

مثال: طراحی یک D_latch Gated ← ورودی ها: G, D
خروجی ← Q

state	inputs D	G	output Q	توضیحات
a	0	1	0	latch تعریف
b	0	0	0	بعد از a
c	1	0	0	بعد از b
d	1	1	1	
e	1	0	1	بعد از d, f
f	0	0	1	بعد از e

	00	01	11	10
a	b, –	ⓐ, 0	d, –	–, –
b	ⓑ, 0	a, –	–, –	c, –
c	b, –	–, –	d, –	ⓒ, 0
d	–, –	a, –	ⓓ, 1	e, –
e	f, –	–, –	d, –	ⓔ, 1
f	ⓕ, 1	a, –	–, –	e, –

← یعنی هم ست

رسم PFt

حالت های سازگار: خواهیم دید حالت a و b و c سازگارند ← a
f, e, d سازگارند ← d

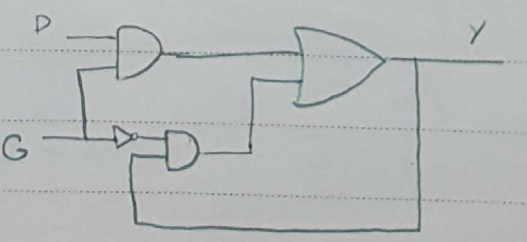
y \ D G	00	01	11	10
0 ← a, b, c	ⓑ, 0	ⓐ, 0	d, –	ⓒ, 0
1 ← d, e, f	ⓕ, 1	a, –	ⓓ, 1	ⓔ, 1

state assign

ⓐ, 0	ⓐ, 0	d, –	ⓐ, 0
ⓓ, 1	a, –	ⓓ, 1	ⓓ, 1

کانونی ⇒ $Y = DG + \bar{G}y$

$Q = y$



Subject:
Date

مراحل کاهش تعداد State :

PFT ← Priority Flow table

۱) استفاده از Partitioning برای حذف حالت های معادل در جدول

← دو حالت S_i و S_j می توانند معادل باشند اگر I) خروجی یکسان داشته باشند

II) کلیه حالت های بعدی نامشخص آنها در یک ستون حالت بعدی باشند

۲) استفاده از روش implication graph برای تعیین حالت های سازگار

۳) ایجاد دیاگرام ادغام merger diagram

۴) انتخاب زیر مجموعه‌ای از حالت های سازگار که قابل ادغام باشند یعنی در نهایت کمترین تعداد زیر مجموعه هایی انتخاب شود که کل حالت ها را پوشش می دهد.

۵) رسم FT جدید

۶) تکرار مراحل ۳ و ۵ تا زمانی که کاهش بیشتری ممکن نباشد

تعریف سازگاری : دو حالت S_i و S_j سازگار هستند اگر به ازای هر وضعیت ورودی آنها یکی از شرایط زیر صادق باشد

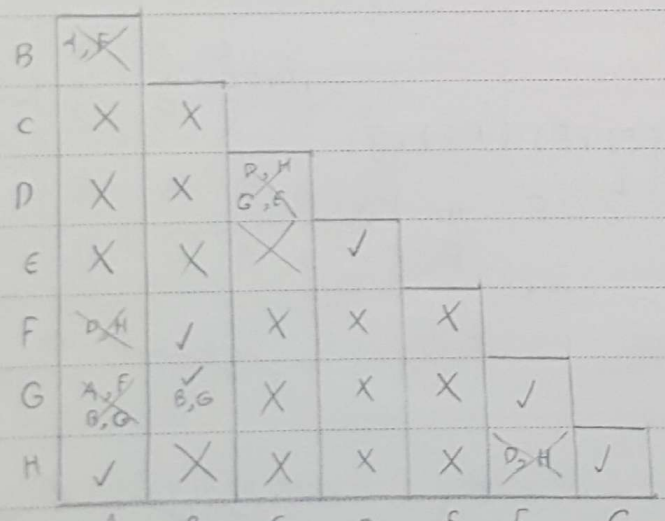
۱) یا حالت بعدی آنها یکسان باشد

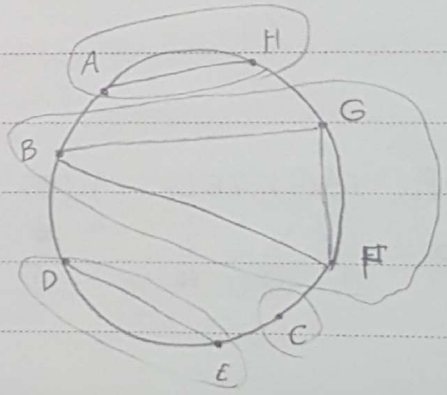
۲) یا هر دو S_i و S_j پایدار باشند

۳) یا حالت بعدی S_i و S_j یا هر دو نامشخص باشند

علاوه بر این خروجی S_i و S_j باید یکسان یا نامشخص باشد

	٠٠	٠١	١٠	١١	
A	(A)	H	B	–	٥
B	F	–	(B)	C	٥
C	–	H	–	(C)	١
D	A	(D)	–	E	١
E	–	D	G	(E)	١
F	(F)	D	–	–	٥
G	F	–	(G)	–	٥
H	–	(H)	–	E	٥





CS	٠٠	٠١	١٠	١١	Z
A	A	A	B	D	٥
B	(B)	D	(B)	C	٥
C	–	A	–	(C)	١
D	A	(D)	B	(D)	١

Subject:
Date

	00	01	10	11	
A	A	F	C	_	0
B	A	(B)	_	H	1
C	G	_	(C)	D	0
D	_	F	_	(D)	1
E	G	_	(E)	D	1
F	_	(F)	_	K	0
G	(G)	B	J	_	0
H	_	L	E	(H)	1
J	G	_	(J)	_	0
K	_	B	E	(K)	1
L	A	(L)	-	K	1

۱) پیدا کردن ← Partitioning

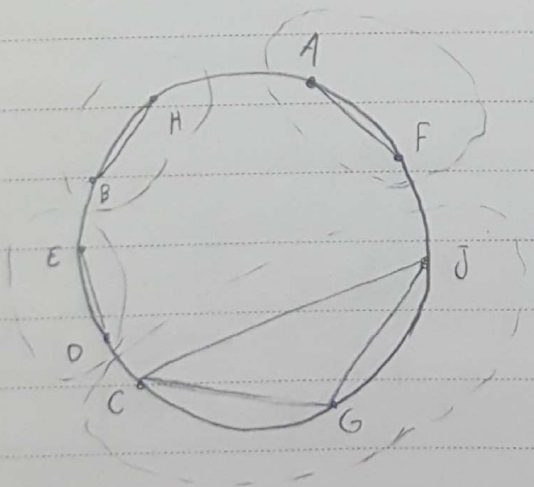
$P_1 = (AG)\ (BL)\ (C)\ (D)\ (E)\ (F)\ (HK)\ (J)$

P_2 : A G BL C C G F HK J

B, L معادلند

K, H معادلند ⇐ حذف دو سطر آخر

$P_3 = P_2$



۲) حالت های سازگار ← جدول استلزام (کتاب)

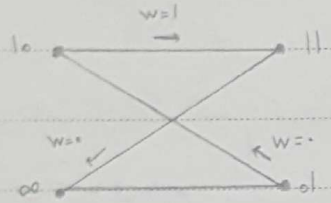
A	(A)	(A)	C	B	0
B	A	(B)	D	B	1
C	(C)	B	(C)	D	0
D	C	A	(D)	(D)	1

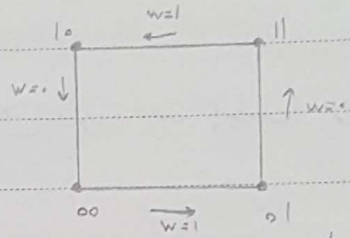
Subject:
Date

انتساب حالت State assignment

	w=0	w=1	Z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	11	0

	w=0	w=1	Z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0





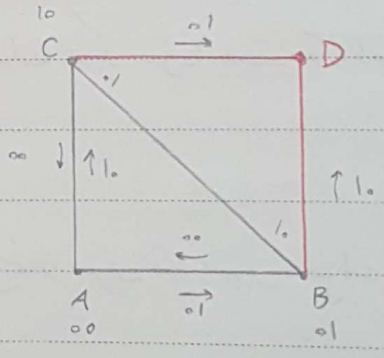
transition diagram

نامطلوب به دلیل وجود قطر

مطلوب

مثال: Flow table

	00	01	10	11	
A	(A)	B	C	–	00
B	A	(B)	D ~~C~~	B	01
C	A	D ~~B~~	(C)	(C)	10
D	–	B	C	–	dd



B → C ✓

01 → 10

{ B → D → C

{ 01 → ▬ → 10 ✓ no glitch

{ B C → D → B ✓

Subject :
Date

FT برای transition diagram

۱- از سطر اول flow table شروع کرده به هر حالت پایدار شماره‌ای نسبت می دهیم حتی اگر در یک سطر چند حالت پایدار به ازای ورودی های مختلف باشد به آنها شماره های متفاوتی نسبت می دهیم

۲- در هر ستون (یک ورودی ثابت) همه حالت هایی که همانم حالت پایدار آن ستون هستند (حالت های ناپایدار) شماره‌ی حالت پایدار متناظر با آن را می گیرند

۲-۱ اگر حالت ناپایداری دیدیم که همانم آن را در آن ستون حالت پایداری نبود، باید شماره حالت پایدار نهایی را به آن نسبت می دهیم.

۳- هر سطر flow table را با یک رأس نمایش می دهیم

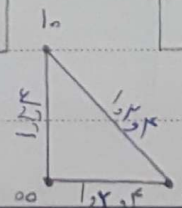
۴- دو رأس V_i و V_j را در نظر می گیریم اگر این دو دارای شماره یکسان در یک ستون هستند آنها را توسط یک edge بهم متصل می کنیم

۴-۱ اگر V_i و V_j هر دو در ستون مذکور ناپایدارند با رنگ منطبق شماره مشترک آنها را روی edge می نویسیم

۴-۲ اگر یکی از V_i و V_j در آن ستون پایدارند با رنگ آن شماره مشترک آنها را روی edge می نویسیم

	00	01	10	11	
A	(A)	B	C	—	00
B	A	(B)	C/A	(B)	01
C	A	B/A	(C)	(C)	10

A	(1)	2	4	—	00
B	1	(2)	4	(3)	01
C	1	2	(4)	(5)	10



B → A → C

01 → 00 → 10

mealy ← yes → glitch ← تغیر

moore ← No

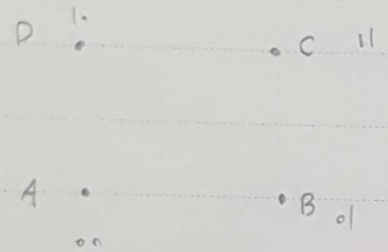
PAPCO

Subject
Date

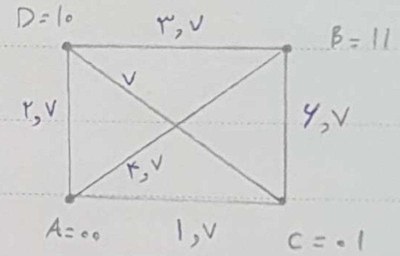
۲- بهره بردن از حالت های Unspecified

	00	01	10	11	
A	(A)	B	C	(A)	00
B	(B)	(B)	D	C	01
C	A	(C)	D	(C)	10
D	B	–	(D)	A	11

A	(۱)	۴	۷	۲	۰۰
B	(۳)	(۴)	۷	۶	۰۱
C	۱	۵	۷	(۶)	۱۰
D	۳	–	(۷)	۲	۱۱



⟹



A	(A)	D	D	(A)	00	00	11	00
B	(B)	(B)	D	C	01	01	1-	01
C	A	(C)	B	(C)	-0	10	10	10
D	B	B	(D)	A	-1	0-	11	00

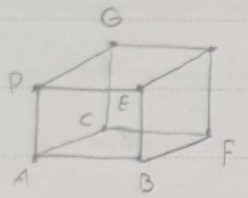
Subject
Date

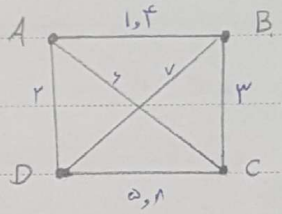
: State assignment

۳ـ اضافه کردن State های جدید ناپایدار

مثال:

	①	②	۶	۴	۰۰
A	۱	③	۷	④	۰۱
B	⑤	۳	⑥	۸	۱۰
C	۵	۲	⑦	⑧	۱۱
D					





روی مکعب HD فاصله همینگ بین State ها باید ۱ باشد

	۰۰	۰۱	۱۰	۱۱	
A	Ⓐ	Ⓐ	C	B	۰۰
B	A	Ⓑ	E	Ⓑ	۰۱
C	Ⓒ	F	Ⓒ	G	۱۰
D	G	A	Ⓓ	Ⓓ	۱۱
E	–	–	D	–	-۱
F	–	B	–	–	--
G	C	–	–	D	۱_

→ don't care

Subject Comb →
Date

data = assemble
جمع نتایج گذاری

۴- انتساب حالت به روش one-hot

همان مثال صفحه‌ی قبل

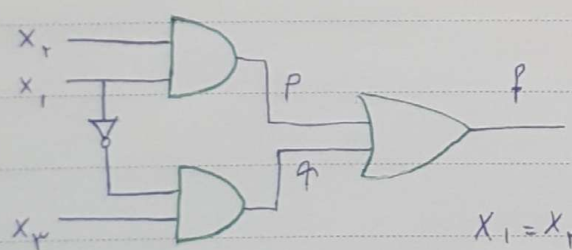
تعداد بیت = تعداد حالات اولیه

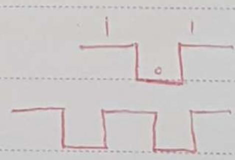
A	0001	(A)	(A)	F	E	00
B	0010	E	(B)	G	(B)	01
C	0100	(C)	H	(C)	J	10
D	1000	I	J	(D)	(D)	11
E	0011	A	–	–	B	0–
F	0101	–	–	C	–	–0
G	1010	–	–	D	–	–1
H	0110	–	B	–	–	01
I	1100	C	–	–	D	1–
J	1001	–	A	–	–	00

state reduction →

hazard پدیده هازارد

hazard ، ایجاد glitch بر روی سیگنال ها به دلیل وجود تأخیرهای ذاتی متفاوت در مسیرهای مختلف

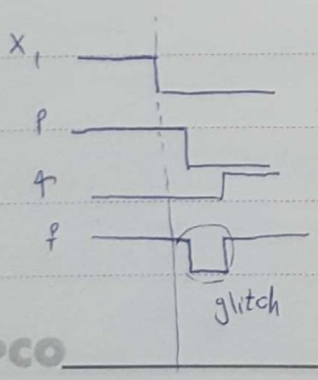


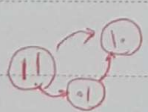


انواع hazard ← static, dynamic

$x_1 = x_2 = x_3 = 1 \to f = 1 \quad (p = 1, q = 0)$

$x_1 : 1 \to 0 \Rightarrow f = 1 \quad (p = 0, q = 1)$





x_3 \ x_1x_2	00	01	11	10
0			1	
1	1	1	1	

$f = x_1x_2 + \bar{x}_1x_3 + x_2x_3$ ← برای جلوگیری از هازارد → جمله‌ای اضافه کردن → برای ترکیبی حرف نیازی نیست