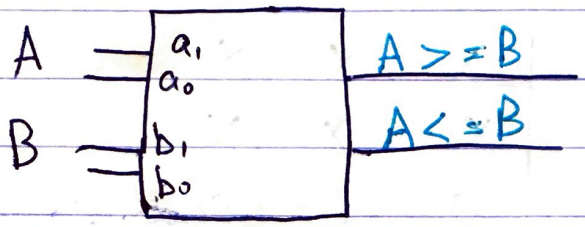
ENCS533
Advanced Digital Systems Design

Dr. Abdallatif Abuissa
Sec: 1
2018/2019
3rd year - 2nd semester
Sajeda Murra

Lec 1:

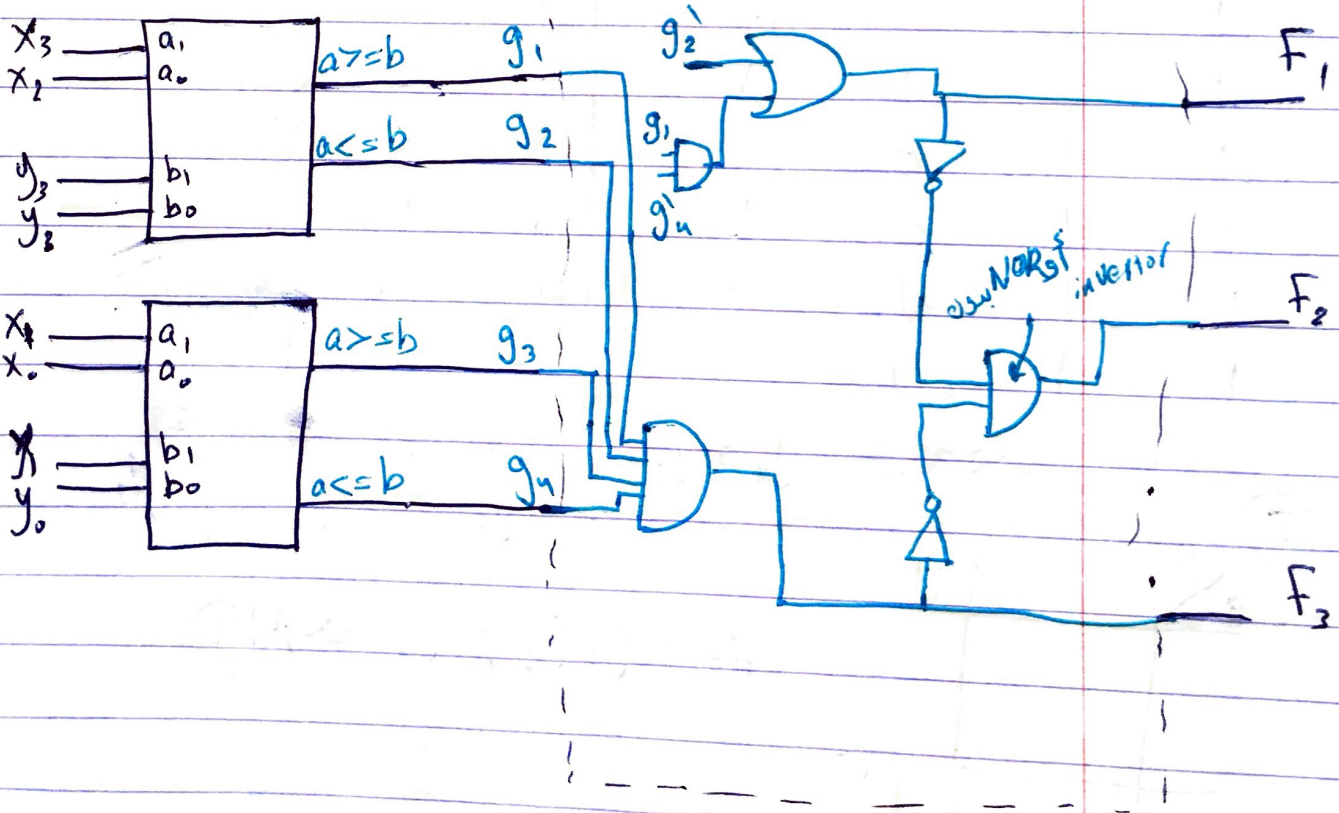
Review

Ex1: Design 4-bit comparator using the following 2-bit comparator.



Sol: $X = X_3\ X_2\ X_1\ X_0$, $Y = Y_3\ Y_2\ Y_1\ Y_0$

$F_1 (X>Y)$ $F_2 (X<Y)$ $F_3 (X=Y)$



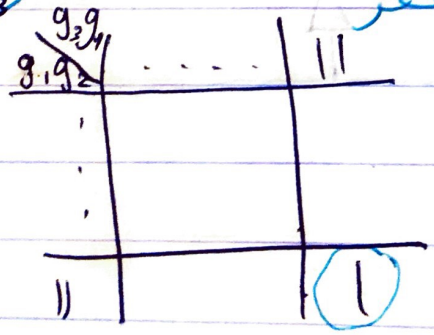
least. sig

most sig. ←

g_1	g_2	g_3	g_4	F_1 (>)	F_2 (<)	F_3 (=)
0	0	0	0	X	X	X
0	0	0	1	X	X	X
0	0	1	0	X	X	X
0	0	1	1	X	X	X
0	1	0	0	X	X	X
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	X	X	X
1	0	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	0	0
1	1	0	0	X	X	X
1	1	0	1	0	1	0
1	1	1	0	1	0	0
1	1	1	1	0	0	1

* مستحيل يكون g_3 و g_4 مع بعض 0 أو g_1 و g_2 مع بعض صفر

F_3



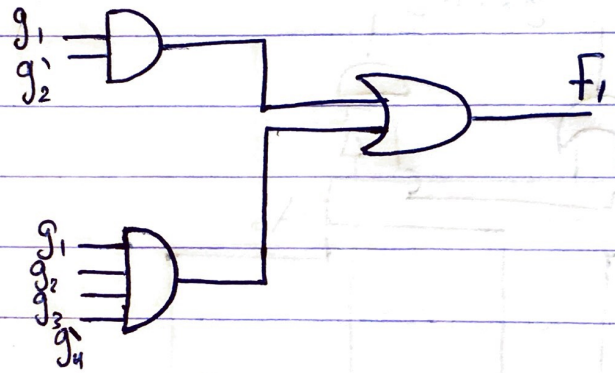
$F_3 = g_1 g_2 g_3 g_4 \Rightarrow$ AND ...

F_1

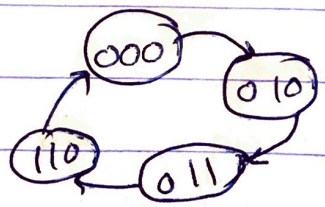
g_1g_2 \ g_3g_4	00	01	11	10
00	X	X	X	X
01	X			
11	X			1
10	X	1	1	1

$F_1 = g_2' + g_1 g_4'$

$F_1 = 1$
when $g_1 = 1$ && $g_2 = 0$ $(g_1 g_2')$
or if $g_1 = 1$ & $g_2 = 1$
$g_3 = 1$ & $g_4 = 0$



EX2: Design counter that counts 0,2,3,6,



001 → x xx
100 → x x x
101 → x x x
111 → x xx

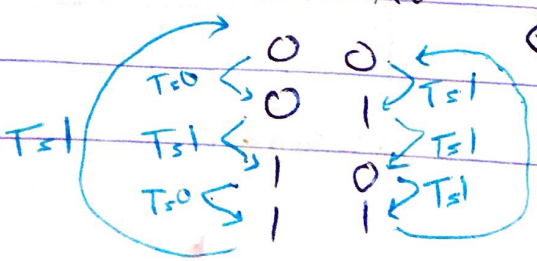
أكثر كوست بالديزاين
وممكن يدخل بستيت دونت
كيرو يعلق فيهم "يصير
فيه مشاكل"

Sol:

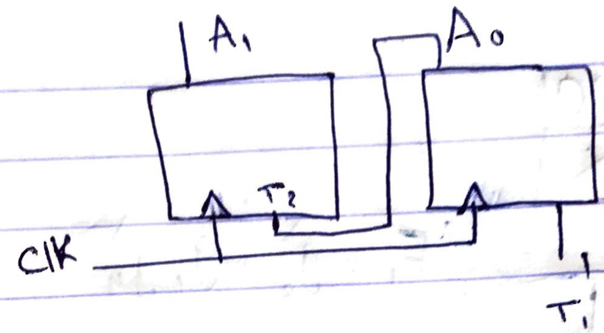
4 states ⇒ 2 Flip Flops

A_1 A_0

⇐ regular counter



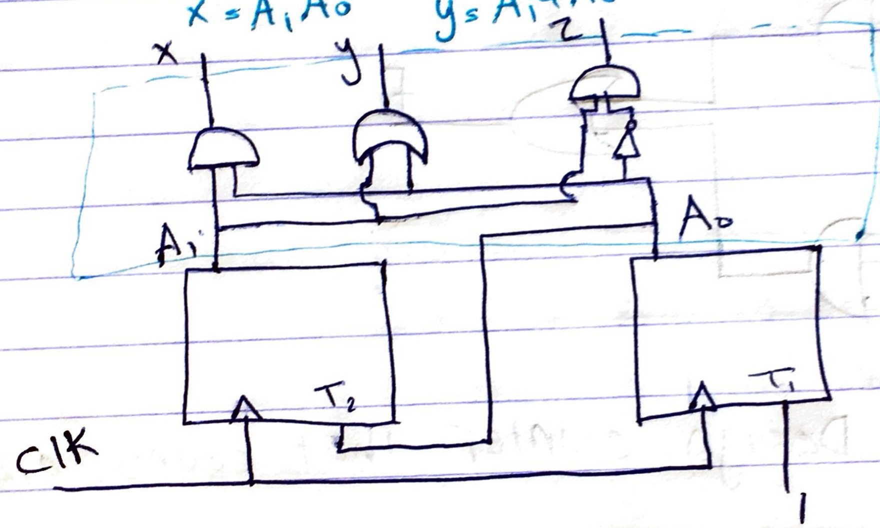
⇒ $T_1 = 1$ $T_2 = A_0$



A_1	A_0	x	y	z
0	0	0	0	0
0	1	0	1	0
1	0	0	1	1
1	1	1	1	0

ممكن يطلع ـ الديزاين أبسط كثير حسب الأرقام

$x = A_1 A_0$ $y = A_1 + A_0$ $Z = A_1 A_0'$



مثلاً لو 1,3,5,7

بثبت least sig على او بكمل ← 2 bit counter

الباقي زي قبل

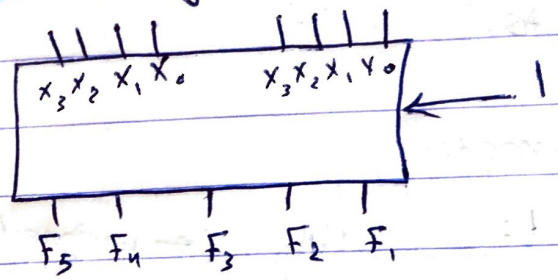
EX3: X is 4 bit number, design $F = 2X+1$

Sol1:

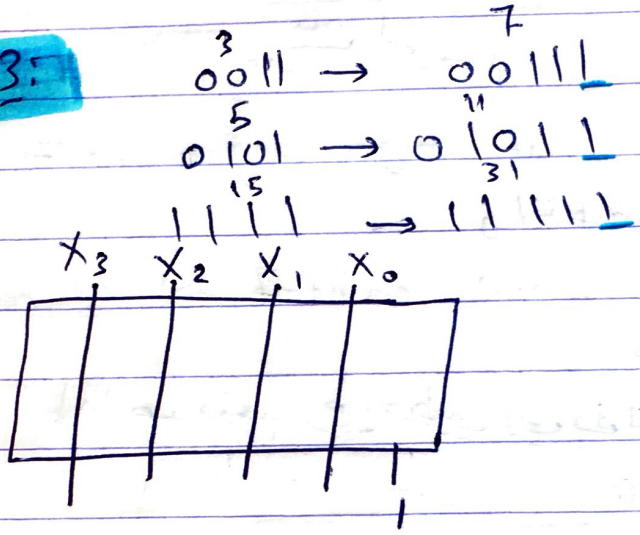
X_3	X_2	X_1	X_0	X_3	X_2	X_1	X_0	F_5	F_4	F_3	F_2	F_1

غلبة كثير و بدو وقت

sol 2: using 4bit adder



sol 3:



Busses:

① Std_Logic_vector

② as a number

declared as inout std_Logic_vector

①

count <= count + 1;

111 ← unsigned package
+ 1
1000
count

default value = U

②

declared as inout integer rang 0 to 7

count <= count + 1;

need to check when count = 7 ⇒ if yes count = 0

else add one

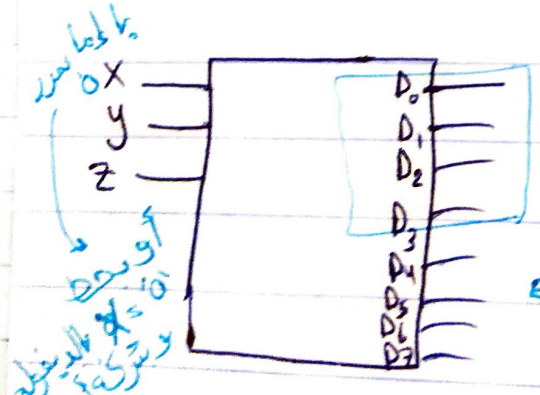
إذا حذفتهم رح يعطيني إيرور بطلع عم الرينج

default value = 0

on hardware: default value is a random number in both cases.

Open ports:

* لما في أوبن بورتس ما بدي أشبكهم وبدي أخذ الباقي.
* الانبوت ممنوع أتركه open إلا إذا أعطيته default value قبل بصير أعمله open



بدي أستخدمه كـ 2x4 decoder

bit → default value is 0.

universal gate

universal port map (a, b, open, c, open);

default يعطي ايرور لذا ما في أو بس '1' — nand — output y ما بلزمني

Generic:

delay : delay_length ← type

entity and 2 generic map (35 ns) port map(a,b,c)

default بصير أخير هاي أو بوخذ ال

Assert and report statement:

cin=0 — 8 bit + 8 bit = 16 bit — 2^16 case — test-bench is not useful

```
process
    loop x
        0 → 256
            loop y
                0 → 256
                wait until rising_edge(clk)
```

sr-flip flop — s = 1 r = 1 — unwanted state

```
assert (not (s='1' and r='1'))
report "set and reset in the same time"
severity note;
```

syntax ككل وحدة — severity العدة

حسب التولد
بطلع أسود
أزرق
أحمر

severity : 1. note
2. warning ↓ بالترتيب
3. error
4. failure → بوقف ال simulation

assert (myresult = actual_result)
report " unexpected result "
severity error ;

بقدر أحطها بال entity عشان ما أكرر
بكل arch ومع ال entity بيصير الها
begin

Generic :

adder sum <= x+y ;
generic { x n bit , y n bit , sum n+1 bit

2**n = 2^n

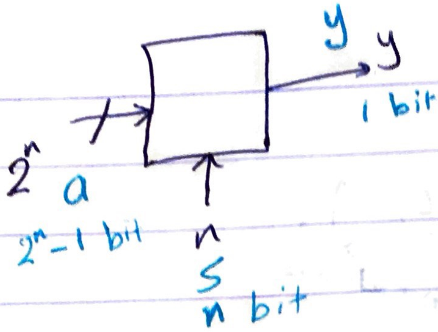
decoder $n \times 2^n$

d(con_integer(a)) <= 1 ;
0 → n

أو

constant d_cons : std_logic_vector (2**n -1 down to 0)
:= (0 => '1' , others => '0') 0000 0001

3×8
a = 0 d <= 0000 0001
d <= shl (d_constant, a); a = 5 d = 0010 0000
a = 7 d = 1000 0000



y <= a(conv_integer(s));

Loop statements

جوا بروسيس

— process

variable count : integer := 0;

begin

⋮

loop

⋮

count := count + 1;

⋮

infinit لوب → wait until rising_edge(clk)

end loop

implicit decleration for i

— for i in 0 to 8 loop جوا الprocess

⋮

end loop;

— while x>2 or y=5 loop جوا بروسيس

⋮

end loop

Exit statement

if (condition) then exit; → break زي

if (condition) then next; → continue زي

Lec 13:

CONV_STD_LOGIC_VECTOR (i, 4);
الرقم الانتجر ← عدد البتات bits

Expected <= con_stal_logic_vector (i+j, 5);
الناتج خمسة بت لما أجمع ٤

- الكلوك حسب الـadder اللي أتعامله
- بعد ما أجرب كل الكيسز بحط wait عشان تنهي الـprocess

Digital Integrated circuits :

Implementation of algorithi

1 General purpose HW "micro processor"
→ implementation by SW
adv ⇒ flexible

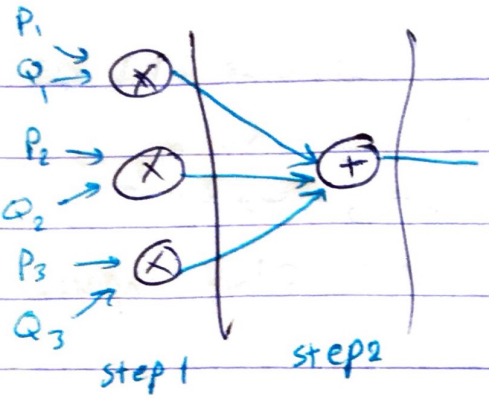
2 specific circuit Hw "specific purpose"
adv ⇒ speed
ASIC → Applicatio specific integrated circuits
disadv. ⇒ not flexible

parallel processing يسرع عن طريق

Ex:	price	Quantity
item1	P_1	Q_1
item2	P_2	Q_2
item3	P_3	Q_3

$$cost = P_1*Q_1 + P_2*Q_2 + P_3*Q_3$$

① ② ③ → step ④



لما بدي أعمل parallel

* FPGA → speed / flexible بجمع بين خصائص الفئتين

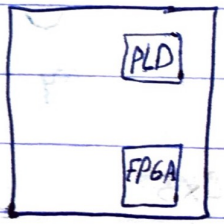
بس ما بتقدر توخذ سرعة الـ specific بتكون أقل و برضو مش flexible زي الـ general

field programmable gate array

* PLDs programmable logic device

Lec 14:

* PLD programmable logic device



① PROM programmable read only memory

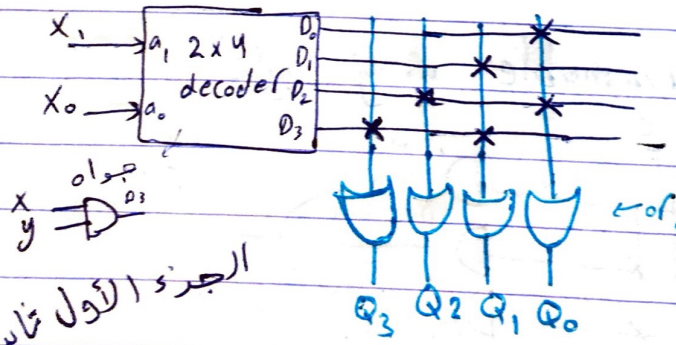
Ex: $F(X) = X^2 + 1$, X is 2-bit input.

$X = 11$ $X^2 = 1001$

$+1$

$= 1010$

X_1 X_0	Q_3 Q_2 Q_1 Q_0
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 1
1 1	1 0 1 0



الجزء الأول ثابت

الجزء الثاني programmable بعير التشبيك

4 x 4 ROM

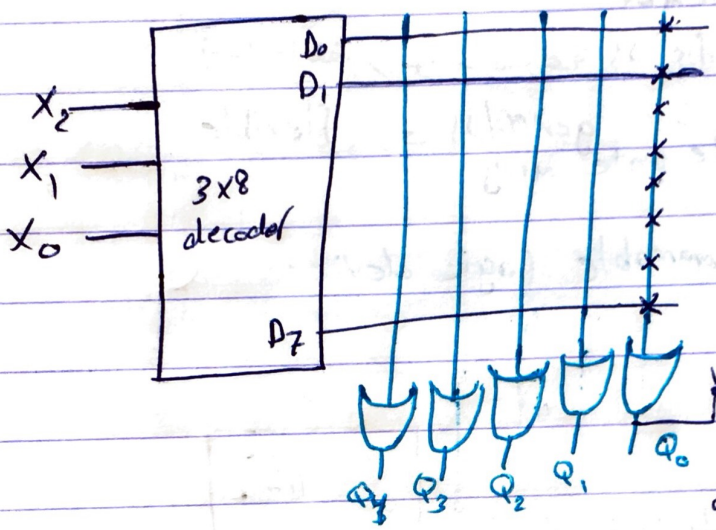
locations

data

4x3 ← X_0 في Q_1 بدك optimal هاذا بدهم يعملوا ②
و Q_0 هي $\grave{X_0}$ فبير 4x2 ROM
optimal صح سطلوب نعملو

Ex: $F(X) = 2X + 3$ X is 3 bit - input.

$2*7+3 = 17 \Rightarrow$ 5 outputs



X_2	X_1	X_0	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	1	1
0	0	1	0	0	1	0	1
0	1	0	0	0	1	1	1
⋮							
1	1	1	0	0	0	0	1

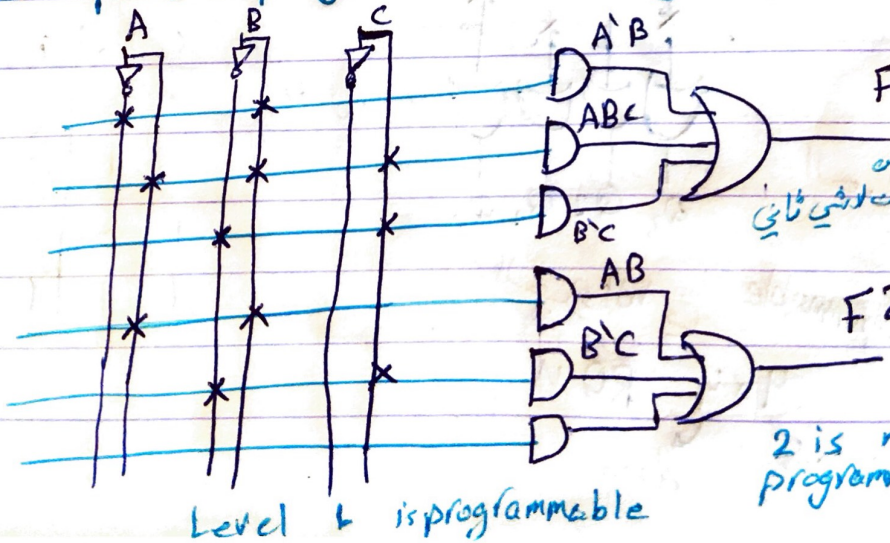
هاذا بيي أقلل كاوبيحمال

8x5

- ROM: non volatile, faster
- RAM: Volatile

عا بكتب عليها إما إذا مسحتها كاملة. وهي نوع مقسم sectors بمعنى sector

② PAL "programmable array Logic"



معكوس البوت لاشي ثاني

2 is not programmable

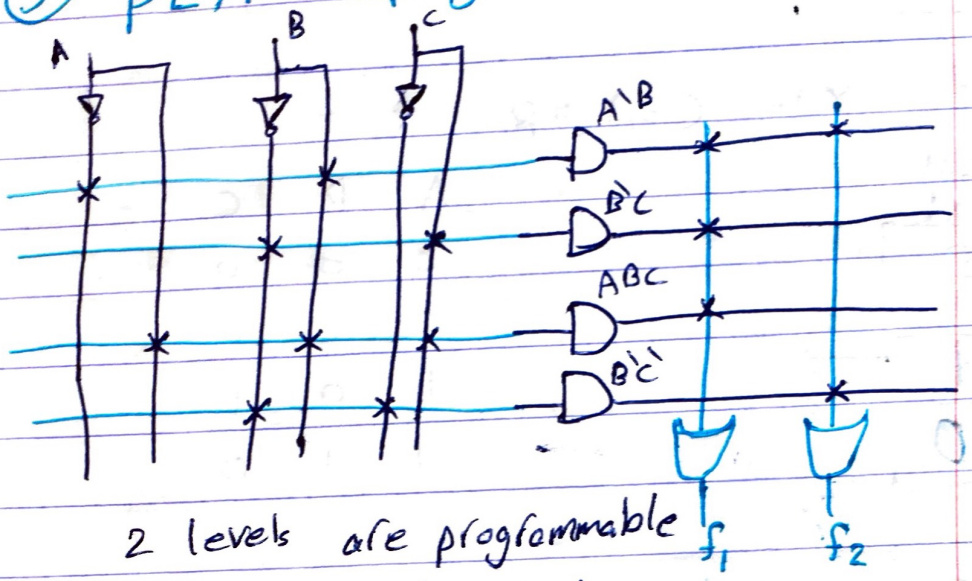
Level 1 is programmable

$f_1 = \Sigma$

$F_1 = A'B + ABC + B'C'$

$F_2 = AB + B'C$

③ PLA (programmable Logic Array)



2 levels are programmable

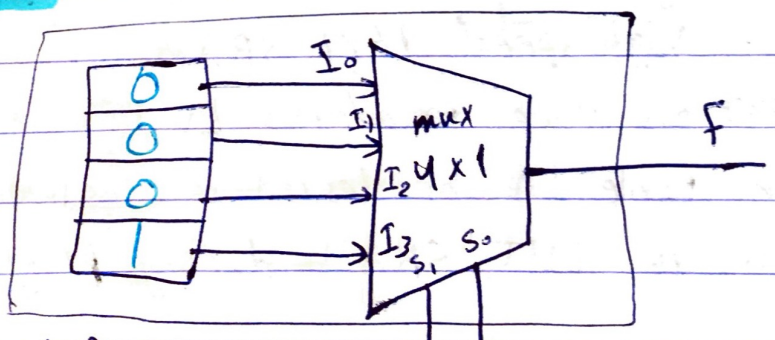
$f_1 = A'B + B'C + ABC$

$f_2 = B'C' + A'B$

⊛ FPGAs (Field programmable Gate Array)

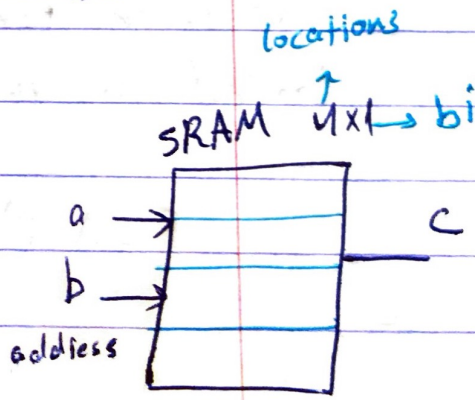
- FPGAs are usually based on look-up-table LUT approach.

Ex: 2 input LUT will look like this

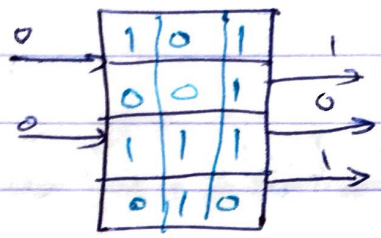


F = A and B

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



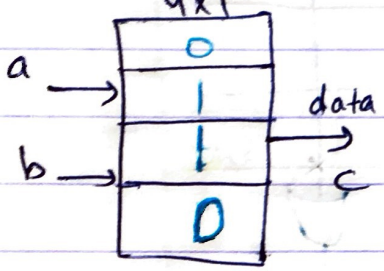
PROM 4x3 bits



decoder ترکیبی
وبعد و ...

Ex: XOR $C = A \oplus B$

SRAM 4x1

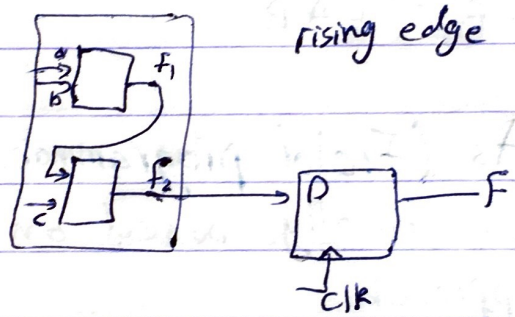


A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

F = a and b and c

2 LUT

rising edge



Ex: real example: in Altera FLEX10K

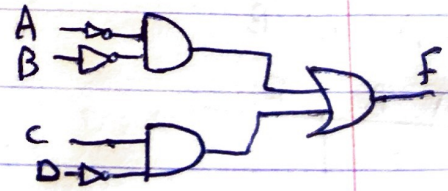
the gate logic is implemented using LUT.

The LUT is high speed 16x1 SRAM.

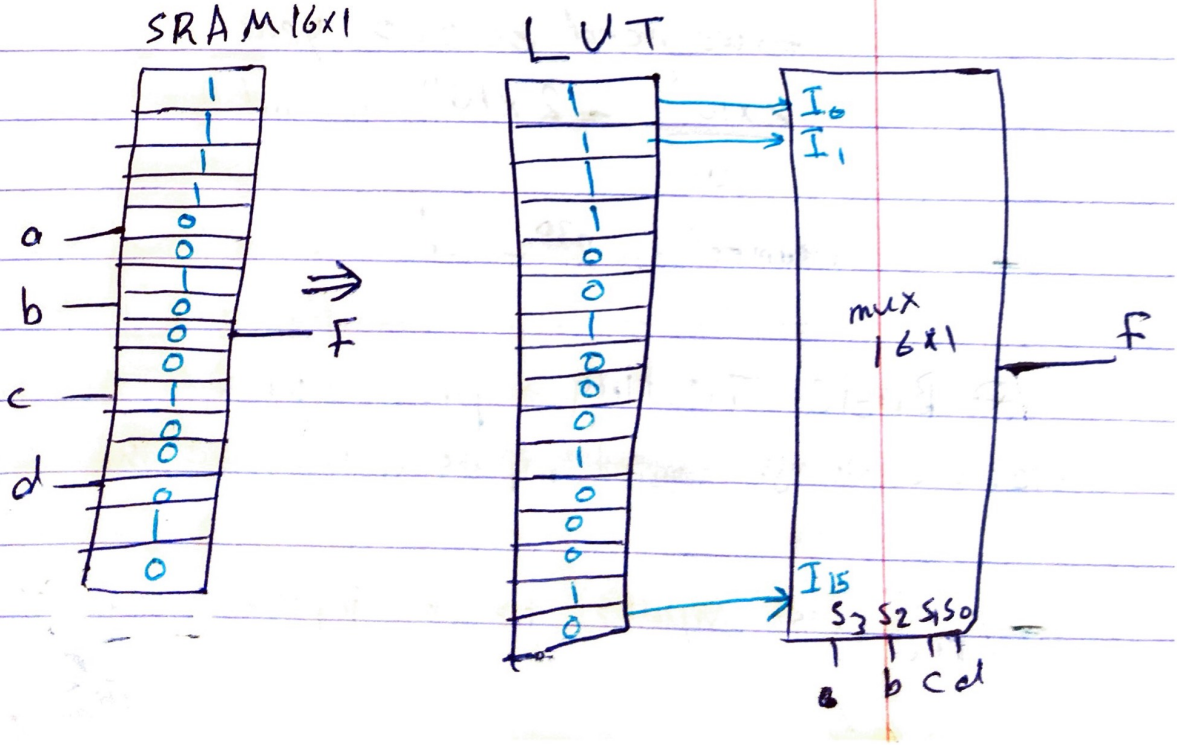
⇒ 4 inputs are used to address LUT memory.

The truth table for the desired gate network is loaded into the LUT SRAM during programming

$F = A'B' + CD'$



address				Data
a	b	c	d	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

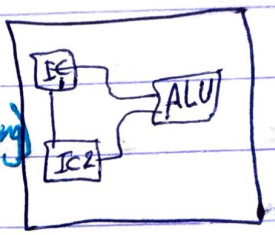


Lec 15:

✱ Testing of Digital circuits

2 approaches for testing:

① functional testing (exhaustive testing)



② structural testing (non-exhaustive testing)

Ex: in exhaustive testing with 1 GHz (10^9) speed of testing cpu.

✱ if the circuit has 32 inputs

$\Rightarrow 2^{32} \backsimeq \underbrace{4 \times 10^9}_{\text{test patterns, test vectors, combinations}} \Rightarrow$ then we need 4 seconds

✱ if the circuit has 64 inputs

$\Rightarrow 2^{64}$ test vectors $\backsimeq 2 \times 10^{19}$

using 1 GHz speed

$\Rightarrow$ we need $\backsimeq$ 585 year

$\frac{2 \times 10^{19}}{10^9} = 2 \times 10^{10}$ second

- we assumed $2^{30} = 10^9$

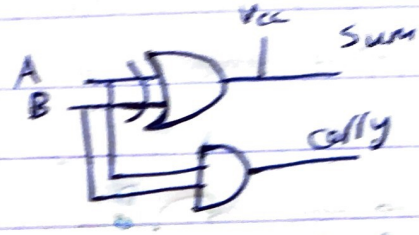
✱ Basic Testing procedure:

Note: we always **control inputs** & **observe the outputs**,

- apply test inputs to the inputs of the circuit,

- observe the outputs and compare them with expected values.

Ex: Half adder



A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Assume that sum is short-circuited with V_{cc}.

non-exhaustive ⊛ بزبط أجرب كلهم بها إذا عندي بختار اللي بين زي A B 0 0

⊛ لهذا بدي أشوف كل الاحتمالات

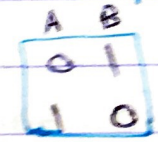
A B

① sum/Vcc 0 0 / 1 1

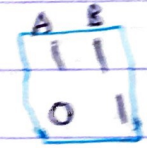
② sum/Gnd 0 1 / 1 0 — * إذا بدي أخصر ① و ② 1 1

③ carry/Vcc 0 0 / 0 1 / 1 0 — * إذا بدي أخصر ② و ③ 0 1

④ carry/Gnd 1 1



2/4 يغطي



4/4 يغطي

لو أخذت

⊛ Fault modelling

Most common is the single stuck at faults.

① Node is short circuited with Vcc → stuck at 1 (s-a-1, sa1)

② Node is short circuited with ground → stuck at 0 (s-a-0, sa0)

⊛ path sensitisation Method (2-Value logic)

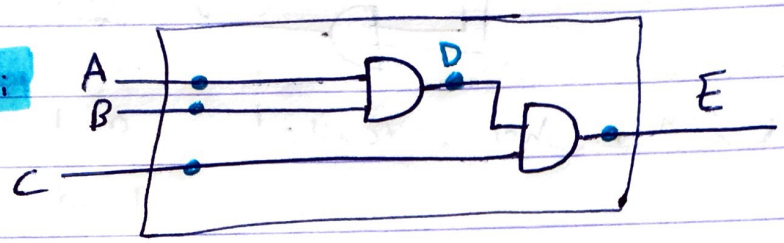
procedure:

For each node in the circuit:

① Backtrace phase: drive the node to non-fault condition.

② propagation phase: steer the content of the node at an output where we can observe & compare.

Ex:



5 nodes

To test node D:

ⓐ Assume D is saϕ

① Backtrace → put 1 on D

→ $\overline{AB}$ = 11

② propagation phase → C=1

as a vector ← $\overline{ABC}$ = 111 → if E=0 faulty

↳ if E=1 not faulty

ⓑ Assume D is sa1

① Backtrace phase → $\overline{AB}$ = 00 or 01 or 10.

② propagation phase → C=1 → $\overline{ABC}$ = 001 or 011 or 101

if E=1 faulty

if E=0 fault-free

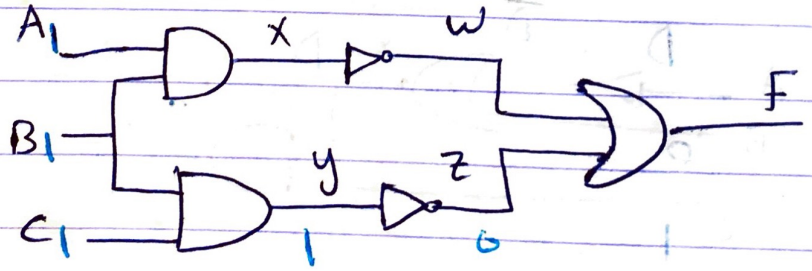
Fault	test vectors ABC	fault free E	faulty E
A sa1	011 * ②	0	1
B sa1	101 *	0	1
C sa1	110 *	0	1
D sa1	001, 011*, 101 ①	0	1
E sa1	000, 001, 010, 011*, 110, 101, 110	0	1
A sa0	111 *	1	0
B sa0	111 *	1	0
C sa0	111 *	1	0
D sa0	111 *	1	0
E sa0	111 *	1	0

4 test vectors allowed

① if vectors : 000, 010, 001, 100 } 20% fault coverage

② if vectors : 111, 011, 101, 110 } 100% fault coverage

Ex:



B sa1

- back trace put 0 on B A=1
- to propagate node w to output F z=0 y=1 $\overline{BC}$ = 11 B=1

contradiction.

D-Algorithm (5-value logic)

① 1: normal logic 1

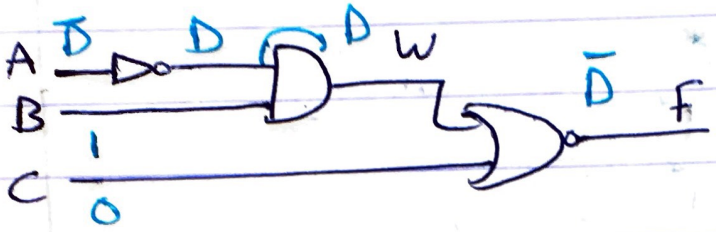
② 0: normal logic 0

③ X: unknown

④ D: represents logic 1 under fault-free condition, and 0 under faulty condition.

⑤ $\bar{D}$: 0 under fault-free, 1 under faulty.

Ex:



test A sa L

→ put 0 on A ($A = \bar{D}$)

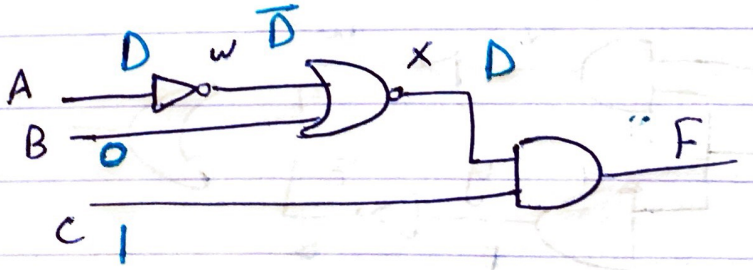
A	B	C
0	1	0

if f=0 no fault

if f=1 fault

Lec 16:

Ex:



A sa0

A	B	C
1	0	1

if F=1 not faulty, if f=0 faulty.

operation on 5-value logic:

1 invert "NOT" $Z = \overline{A}$

A	Z
0	1
1	0
X	X
D	$\overline{D}$
$\overline{D}$	D

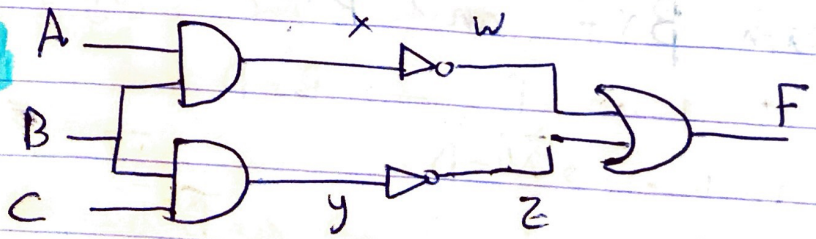
2 AND $Z = A \cdot B$

A \ B	0	1	X	D	$\overline{D}$
0	0	0	0	0	0
1	0	1	X	D	$\overline{D}$
X	0	X	X	X	X
D	0	D	X	D	0
$\overline{D}$	0	$\overline{D}$	X	0	$\overline{D}$

3 OR $Z = A + B$

A \ B	0	1	X	D	$\overline{D}$
0	0	1	X	D	$\overline{D}$
1	1	1	1	1	1
X	X	1	X	X	X
D	D	1	X	D	1
$\overline{D}$	$\overline{D}$	1	X	1	$\overline{D}$

Ex:



test B sa1:

put 0 on B ⟶ $B = \bar{D}$

use path BXWF

⇒ A = 1 $X = \bar{D}$ ⇒ $W = D$

to propagate from w to f $Z = 0$ $y = 1$

$\bar{B}C = 11$

contradiction on the value of B → path fails

path ByZF will fail "symmetry"

(*) try path BXWf & ByZf together

put 0 on B ⇒ $B = \bar{D}$

A = 1 → $X = \bar{D}$ ⟶ $W = D$ ⟶ $F = D$

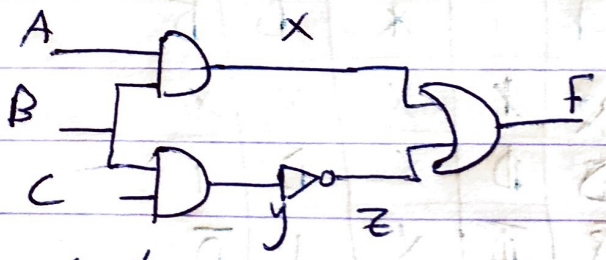
C = 1 → $y = \bar{D}$ ⟶ $Z = D$

test vector

A B C if f = 1 no fault

1 0 1 f = 0 faulty

Ex:



test B sa0

↳ path Bxf and ByZf together

put 1 on D ⟶ B = D

A = 1 ⟶ X = D

C = 1 ⟶ y = D ⟶ $Z = \bar{D}$ ⟶ F = 1 fail

2- try path BXf

put 1 on B → B=D

→ A=1 X=D

to propagate from node x to output f

→ z=0 y=1 → $\overline{BC}$=11

must be 1

contradiction path falls

هاي اللي بفحص فيما اذا صح او لا

ABC

1 1 1 if not faulty B=1

x=1

y=1 z=0 ⇒ f=1

no difference

ABC if faulty B=0

x=1

y=0 z=1 ⇒ F=1

3- path By Zf

put 1 on B → B=D

→ C=1 → y=D z=$\overline{D}$

to propagate from node z to output F

x=0 ⇒

A	B	
0	0*	
0	1	→ AB=01
1	0*	

$\overline{ABC}$

011 if not faulty

x=0

y=1 z=0 f=0

if faulty

x=0

y=0 z=1 F=1

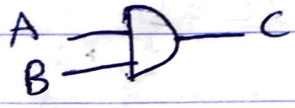
② إذا طلب all test vectors ممكن يلزمني كل الـ paths أما إذا كان بس المهم أطلع test vector واحد

⊛ Fault collapsing

2 concepts

① fault equivalance

الـ faults اللي نفس التست فيكتور.



A sa ∅ : AB = 11

B sa∅ : AB = 11

C sa∅ : A B = 1 1

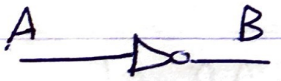
A sa∅ : $\overline{AB}$ = 11

B sa∅

C sa1

A sa1 ≡ B sa1 ≡ C sa1 AB = 00

A sa1 ≡ B sa1 ≡ C sa∅



A sa∅ ≡ B sa1

A sa1 ≡ B sa∅

② fault dominance

f1	f2
101	000
111	001
000	
001	

المسيطر هو اللي f1

f_1 is said to dominate f_2 if the

test vectors of f_2 are subset of the test vectors of f1.

بشطب الكبير لإنو أي واحد من الصغير رح يفحص الكبير معه.

A sa1

A	B
0	1

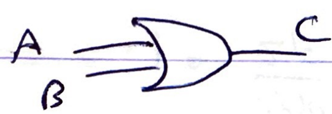
B sa1

A	B
1	0

~~C sa1~~

A	B
~~0~~	~~0~~
~~0~~	~~1~~
~~1~~	~~0~~

C sa1 dominates A sa1 and dominates B sa1



A sa ∅

A	B
1	0

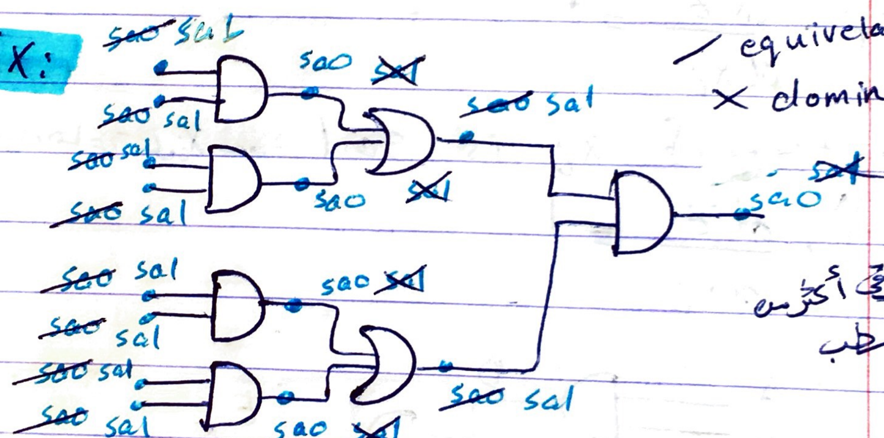
B sa ∅

A	B
0	1

~~C sa ∅~~

A	B
~~1~~	~~0~~
~~0~~	~~1~~
~~1~~	~~1~~

Ex:



/ equivelance
× dominance

فممكن أعالج أكثر من هيك شطب

① exhaustive testing 8 inputs → $2^8 = 256$ vectors

② using fault model 15 node $* 2 = 30$ (sa∅, sa1)

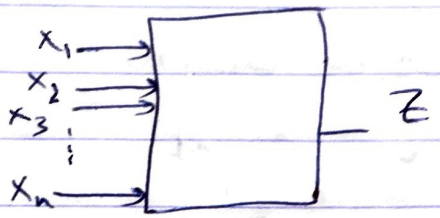
maximum 30 test vector

exP. سلبيات ① إذا ضفت input $2^9 = 512$ الضعف

Linear ② إذا ضفت input $16 \times 2 = 32$

③ fault collapsing → max 15 test vectors.

⊛ Boolean Difference Method



$$Z(x) = f(x_1, x_2, x_3 \dots x_n)$$

$$\frac{dZ}{dx_i} = f(x_i = 0) \oplus f(x_i = 1)$$

Z is sensitive to x_i when $\frac{dZ}{dx_i} = 1$

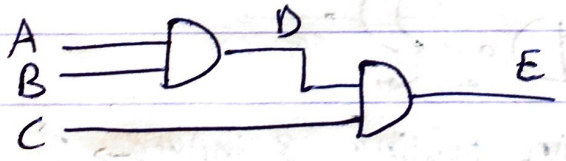
to test if x_i is sa ∅ يكون حالة على x_i ١-

$$x_i \cdot \frac{dZ}{dx_i} = 1$$

to test if x_i is sa 1 او x_i يكون حالة على

$$(x_i)' \cdot \frac{dZ}{dx_i} = 1$$

EX:



to test node A.

$$E = f(A, B, C) = A.B.C$$

$$\frac{dE}{dA} = f(A=0) \oplus f(A=1)$$

$$= 0 \oplus B.C = B.C$$

Z is sensitive to A when $\frac{dE}{dA} = 1$

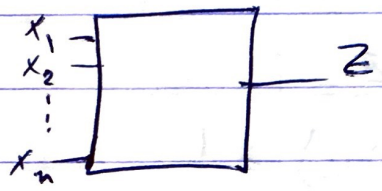
$$B.C = 1 \Rightarrow \overline{BC} = 11$$

A sa∅
$A \cdot (B \cdot C) = 1$ $\quad \overline{ABC} = 111$

A sa 1
$A' \cdot (B \cdot C) = 1$ $\quad \overline{ABC} = 011$

Lec 17:

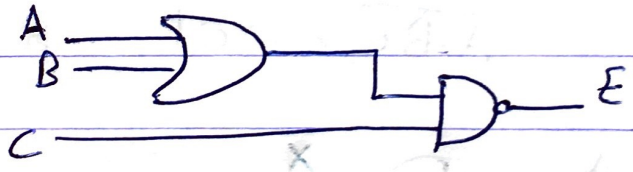
Boolean Difference:



$$\frac{dz}{dx_i} = f(x_i = 0) \oplus f(x_i = 1)$$

z is sensitive to x_i when $\frac{dz}{dx_i} = 1$

EX:



to test node B.

$$E = [(A+B) \cdot C]'$$

$$\frac{dE}{dB} = f(B=0) \oplus f(B=1)$$

$$= (A \cdot C)' \oplus C'$$

A	C	$(A \cdot C)'$	C'	$\oplus$ ← $\frac{dE}{dB}$
0	0	1	1	0
0	1	1	0	1
1	0	1	1	0
1	1	0	0	0

E is sensitive to B when $\frac{dE}{dB} = 1$ — AC: 01

$X \oplus Y = X Y' + X' Y$

$= (A \cdot C)' \cdot C + (A \cdot C) C'$

$= (A' + C') \cdot C$

$= A'C + CC'$

$= A'C \overset{??}{=} 1 \Rightarrow \overline{AC} = 01$

test B saϕ $\quad B \cdot \frac{dE}{dB} = 1$

$B \cdot (A'C) = 1$

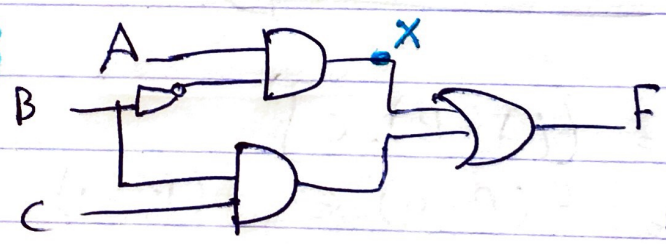
$\overline{ABC} = 011$

test B sa1 $\quad B' \cdot \frac{dE}{dB} = 1$

$B' \cdot A'C = 1$

$\overline{ABC} = 001$

EX:



to test node B.

$F = AB' + BC$

$\frac{dF}{dB} = f(B=0) \oplus f(B=1)$

$= A \oplus C$

F is sensitive to B when $\frac{dF}{dB} = 1$

$A \oplus C = 1$

A C
0 1
1 0

B sa 0

$B \cdot (A \oplus C) = 1$

A	B	C
0	1	1
1	1	0

B sa L

$B' \cdot (A \oplus C) = 1$

A	B	C
0	0	1
1	0	0

to test node X

$F = X + BC$

$\frac{dF}{dx} = f(x=0) \oplus f(x=1)$

$= BC \oplus 1 = (BC)'$

F is sensitive to X when $(BC)' = 1$

B	C
0	0
0	1
1	0

to test X sa 0

$X \cdot \frac{dF}{dx} = 1$

$(AB') \cdot (BC)' = 1$

A	B	C	① (AB')	② $(BC)'$	①·②
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	0	0

$\overline{ABC} = 100$
101

$(AB') \cdot (BC)' \stackrel{?}{=} 1$

$AB' \cdot (B' + C')$

$= AB' + AB'C'$

$= AB'(1 + C')$

$= AB' \stackrel{?}{=} 1$

A B C

1 0 X → 100, 101

* if $\frac{dF}{dx}$ is 0, then the node is untestable

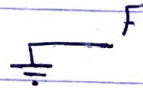
* if $\frac{dF}{dx} = 1$ then F is always sensitive to X.

ex①: 

$F = X'$

$\Rightarrow \frac{dF}{dx} = f(x=0) \oplus f(x=1) = 1 \oplus 0 = 1$

ex②: 



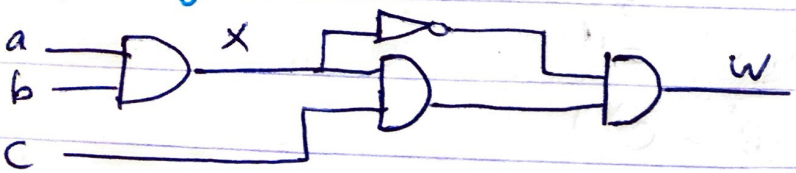
$F = X \cdot X' = 0$

$\Rightarrow \frac{dF}{dx} = f(x=0) \oplus f(x=1) = 0 \oplus 0 = 0$

Untestable Faults

- due to redundent hardware ex②
  * completely untestable nodes
  * partially untestable nodes

⊥ completely untestable faults



X is completely untestable

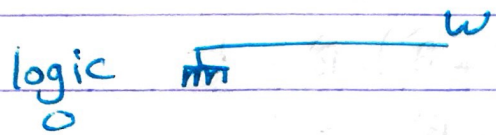
X sa0 untestable

X sa1 untestable

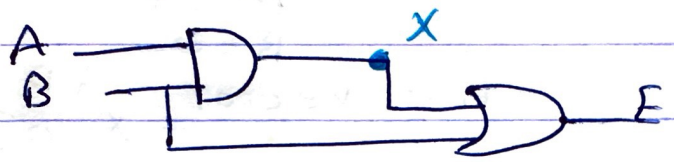
$w = X \cdot C \cdot \overline{X}$

$= 0$

$\frac{dw}{dx} = 0$



2 partially untestable faults



$E = AB + B$

X sa 1

Backtrace

A	B
0	0
0	1
1	0

$X = \overline{D}$

propagation $B = 0$

$\Rightarrow$ test vectors

A	B
0	0
1	0

X sa 0

Backtrace

A	B
1	1

$X = D$

propagate $B = 0$ ← untestable

no test vector, partially untestable. sa1 $\Leftrightarrow$ sa0 ✗

$E = AB + B$

$E = X + B$

$\frac{dE}{dx} = f(X=0) \oplus f(X=1)$

$= B \oplus 1 = B'$

E is sensitive to X when $B' = 1 \Rightarrow B = 0$

X sa 0

$X \cdot \frac{dE}{dx} = 1$

$(A.B).B' = 1$

$0 = 1$

untestable

X sa 1

$X' \cdot \frac{dE}{dx} = 1$

$(AB)'.B' = 1$

$(A'+B').B' = 1$

$A'B'+B' = 1$

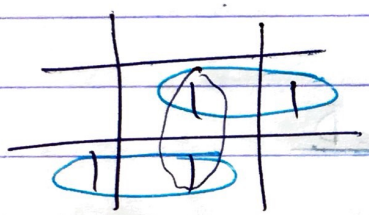
$(A'+1)B' = 1$

$B' = 1$

$B = 0$

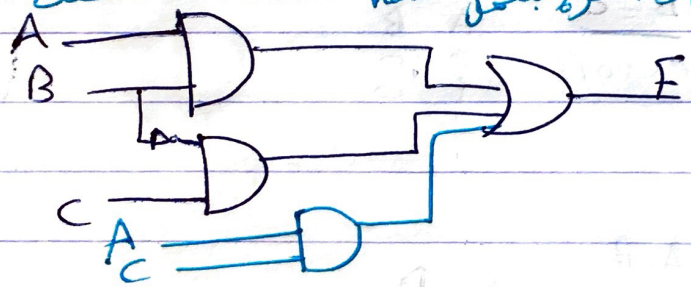
test vector AB → 00

X 0 → 10



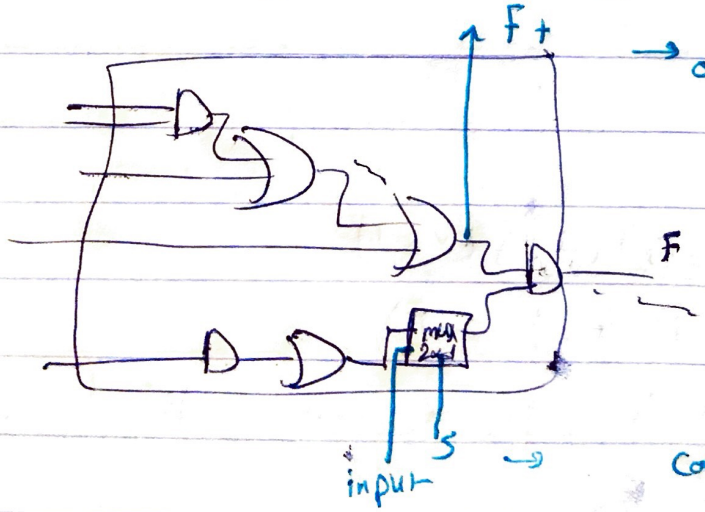
تكرار أو زيادة مقصودة في الهارد وير

في بعض الأحيان ممكن أخذ كمان واحدين عشان أحل hazard في بعض أسرع فبعمل



⊛ Design for testability (DFT)

التي بعمله أثناء التصنيع عشان أسهل التشخيص.



→ observability يزيد

controlability يزيد

① طريقة add hock

② في طريقة structured DFT

③ built in self test

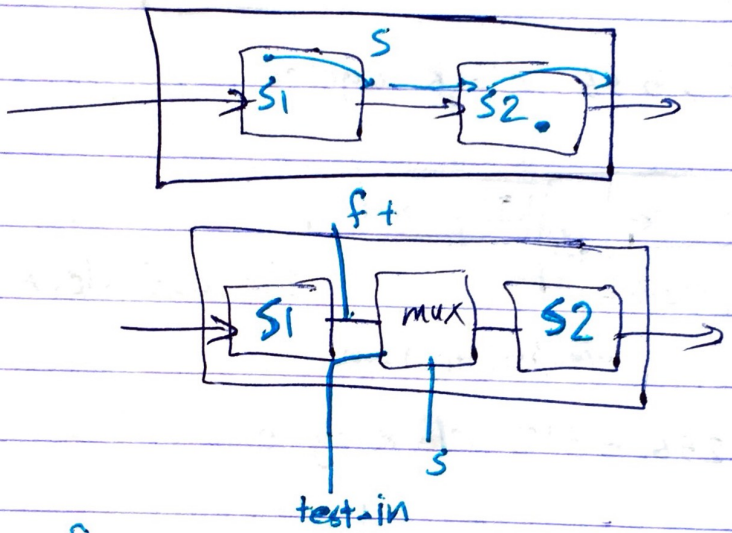
Lec 18:

✱ Design for testability (DFT)

1. ad Hoc DFT
2. structured DFT
3. Built-in self-test (BIST)

✱ Ad Hoc DFT

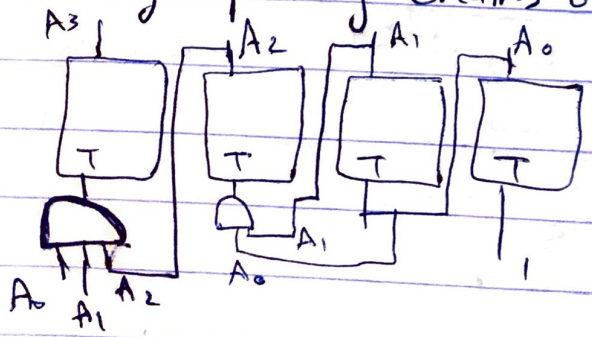
① partitioning of system into subsystems



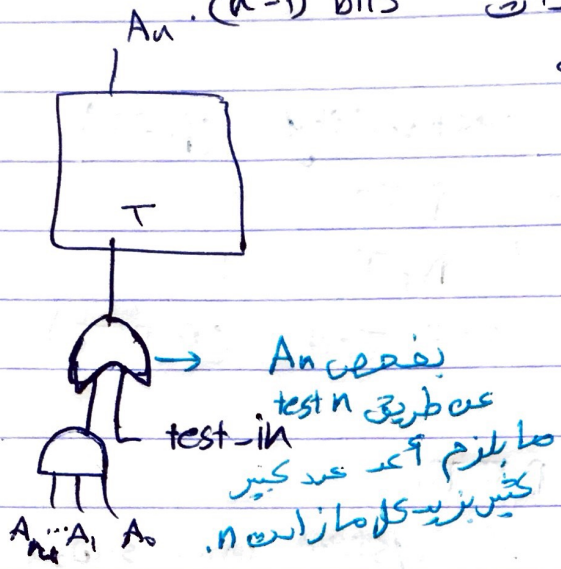
f_t : increase observability of S1

test_in: increase controlability of S2

② Breaking up long chains of sequential circuit.



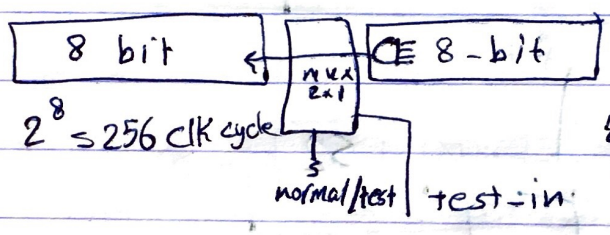
✱ إذا بدي أفحص bit في كاونتر إذا رقمها n لازم كل



16-bit

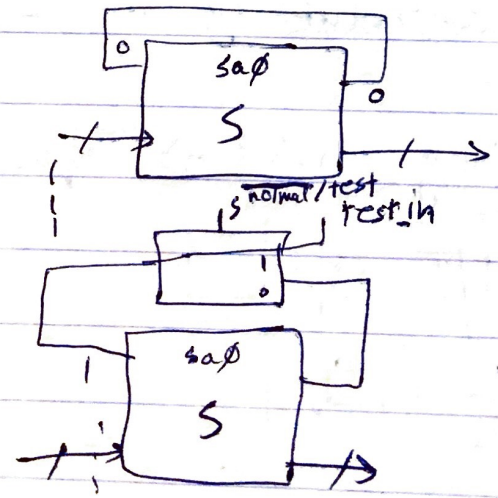
16-bit counter

$2^{16} = 65,536$ clock cycles



256 + 256 = 512 clock cycle

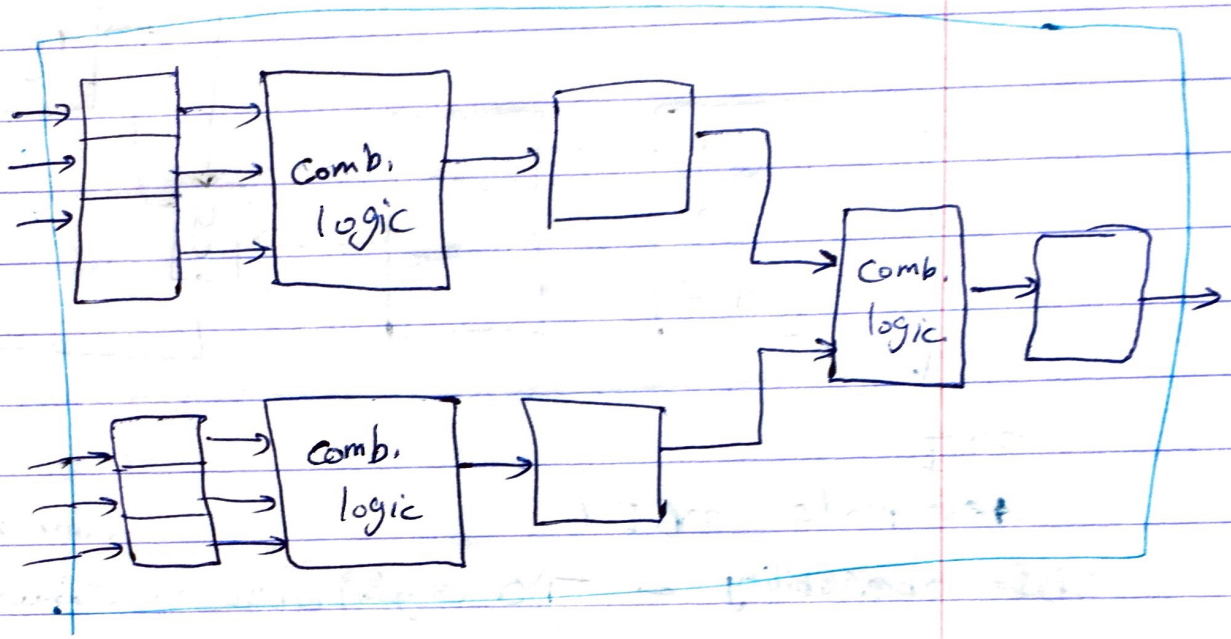
③ Breaking the feedback loop



⑨ Initializing sequential circuits

⊛ لما يزيدوا Pins يزيدوا بطريقة معقولة مش محد كبير كفين

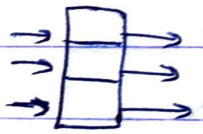
⊛ structured DFT



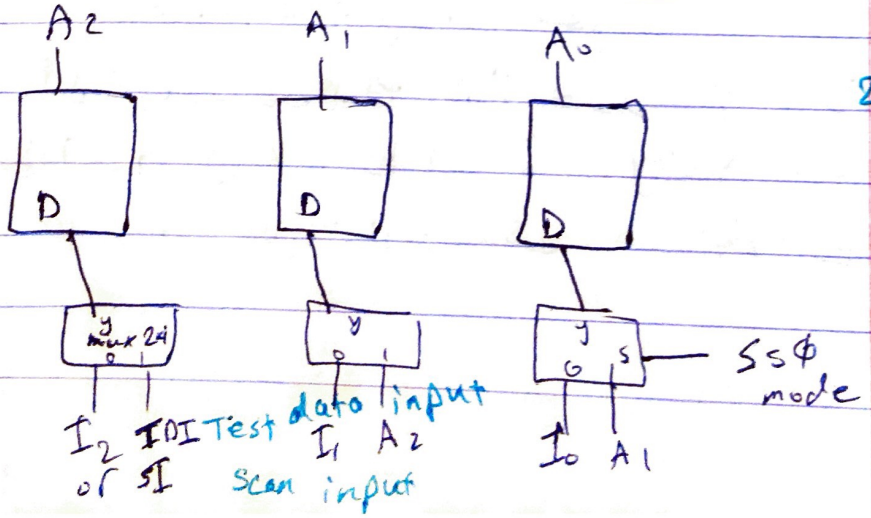
⊛ Scan path testing

⊛ we need scan register

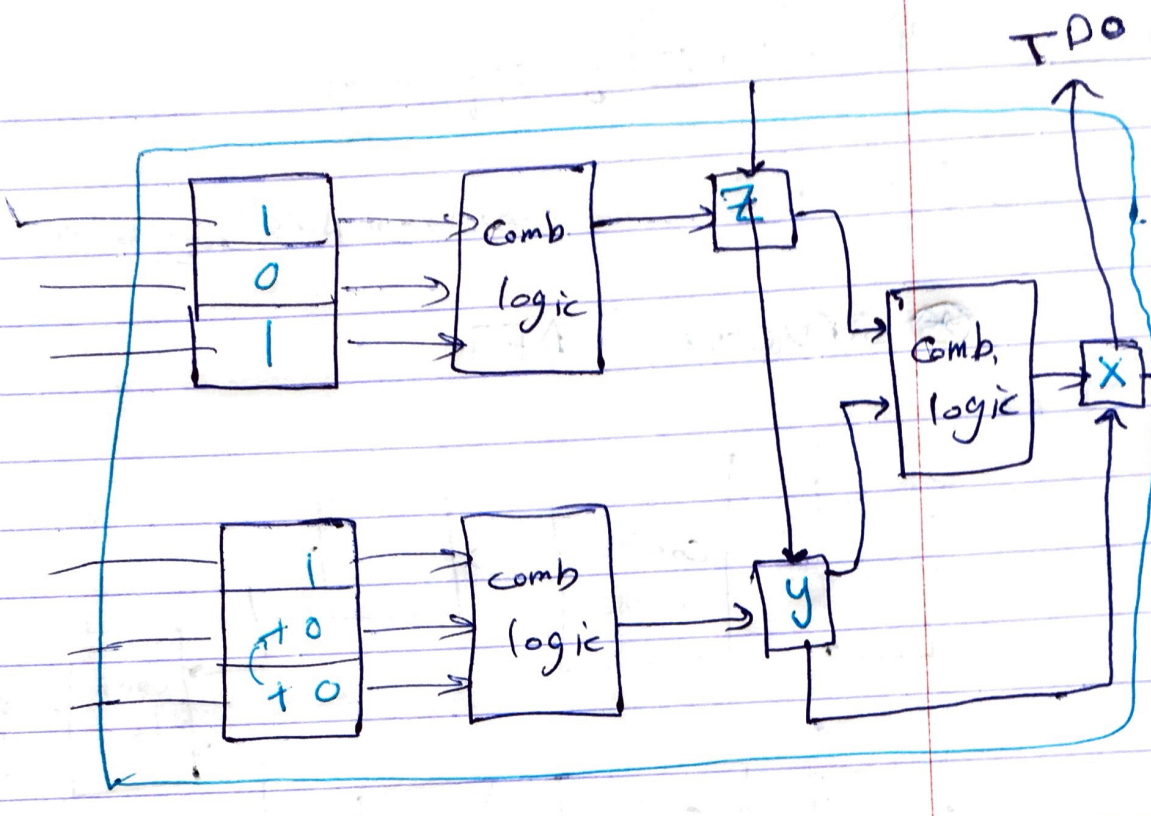
① normal register



② shift register



20% hardware area overhead



TDI

test mode TDI ← controlability

~~TDI~~ observability ← TDO عشان أشوف بغير التسبت مود

بسويها نورمال مود لكلوك وحدة خلالها بعمل ريـ.

بعمل تست مود للريجستر بمرقيم 101 001 6 clock

بوصل بعدين برجع نورمال مود عشان أشوف output

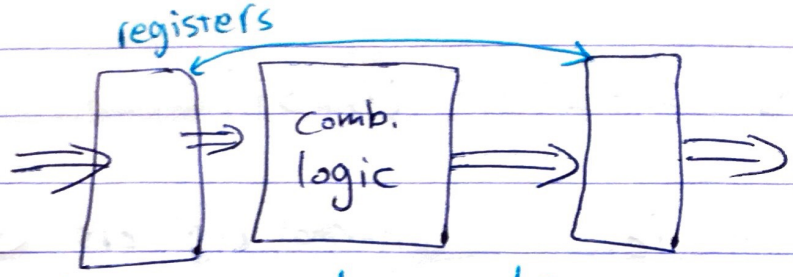
Ex: 1
1
0
1
0

بطلع X بعدين Y بعدين Z

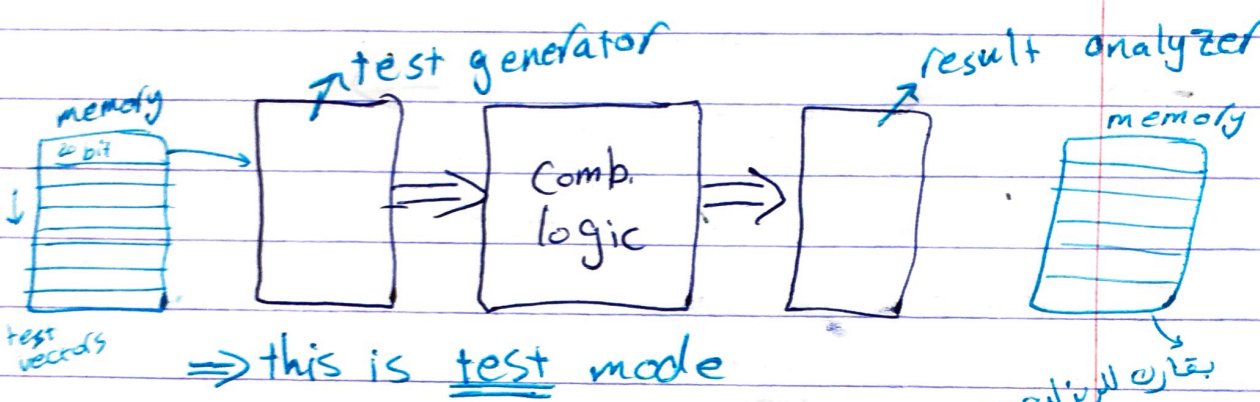
← حسب ترتيب الأسهم

* أول اشي بفحص الـ flipflops بمر ؛ لازم أشوفهم ؛
إذا ١ faulty ولذا ٥ faulty بس مش معروف
وين الغلط بالزرط يلو ممكن وحدة ضاربة بصفر بعدها ضارب بواحد.

⊛ Built_in Self_Test (BIST)



⇒ this is normal mode

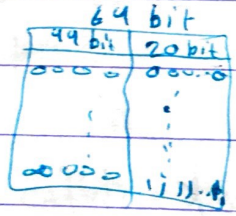


⇒ this is test mode

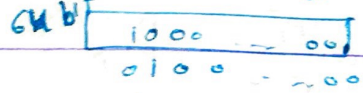
بقارن الريزلت من هاي ← اللي طلعت من comb.

problems:

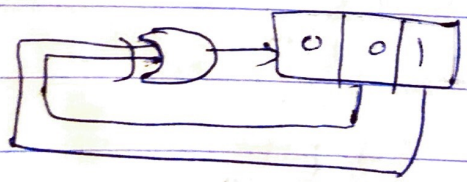
1. area overhead.
2. when n bits and n is large, if we want to test 1000,000 we need to convert to counter,



بدال كاونتر لو shift reg



maximum = 64 test vector



001
100
010
101
110
111
011
001

linear feed back register

١- ميزته سهل
٢- عشوائي

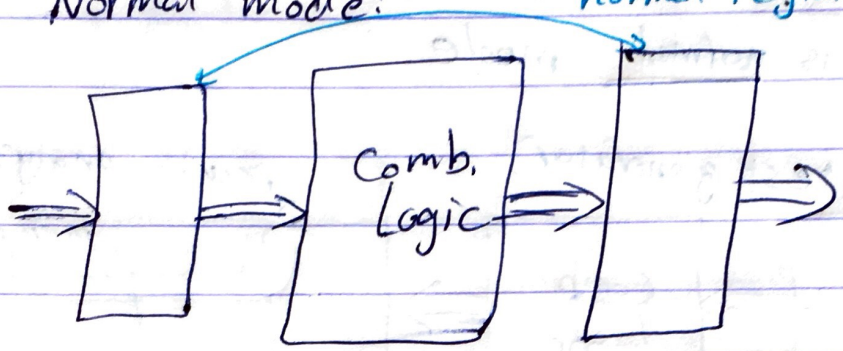
$2^n - 1$ بستثني الصفر
عشان ضايعلق

بعطي

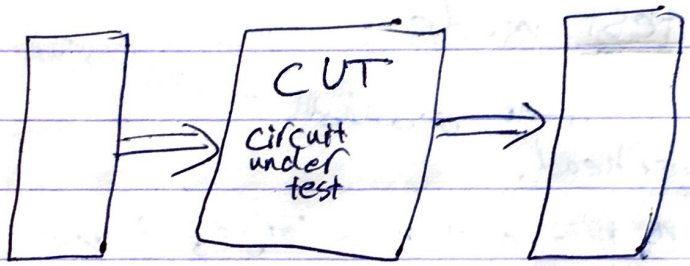
Lec19:

⊛ Built_In self_Test (BIST)

Normal mode: · normal registers



Test mode:

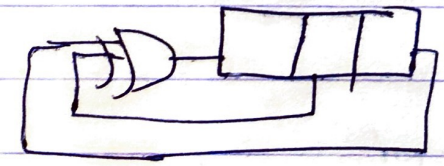


Test pattern Generator
TPG

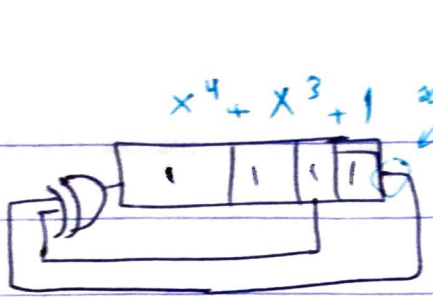
signeture register
Result Analyzer

⊛ Linear feedback shift register (LFSR) as TPG

$x^3 + x^2 + 1$



A	B	C	
0	0	1	A = B⊕C
1	0	0	shift[B = A
0	1	0	C = B
1	0	1	
1	1	0	
1	1	1	
0	1	1	
0	0	1	

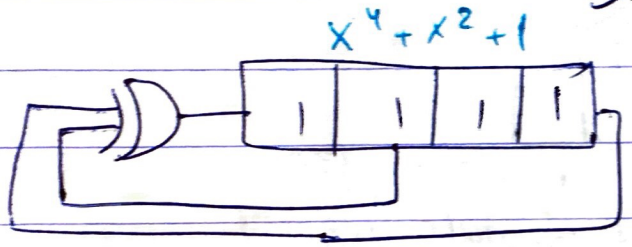


1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1

15 test vector

$2^4 - 1 = 15$
↙ zeros

⊛ LFSR ⇒ من طرف التشفير



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

6 من 15

Maximal length LFSR
Tap sequence of LFSR
يعطي أحسن اشي مواقع الـ xor حسب عدد الـ bits

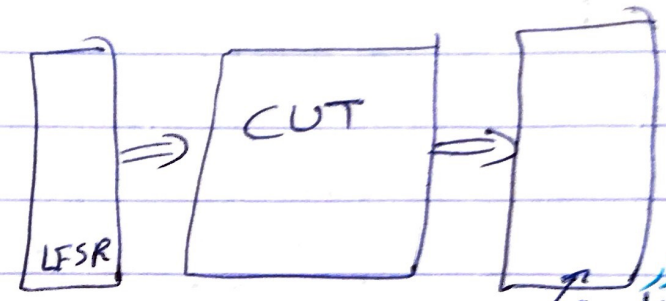
⊛ مبني على فكرة الـ prime polynomial لازم البرايم يعطي أقل من $2^n - 1$

أي متسم للبرايم يكون برايم

$x^3 + x^2 + 1$ ← برايم
$x^0 + x + x^3$
$x^3 + x + 1$ ← برايم

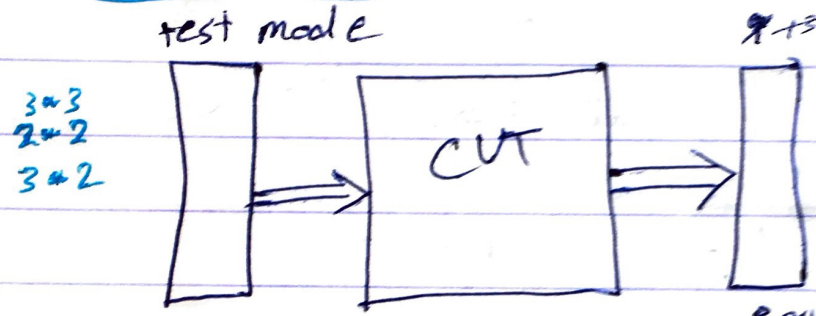
برايم $x^4 + x^3 + 1 \Rightarrow x^4 + x + 1$

المتهم بطلع نفس العدد لكن بترتيب مختلف



signature
بعتقد على مليون تست بيكتور
result analyzer
عشان ما أتأكد من كل نتيجة

Lec 20:



3 * 3
2 * 2
3 * 2

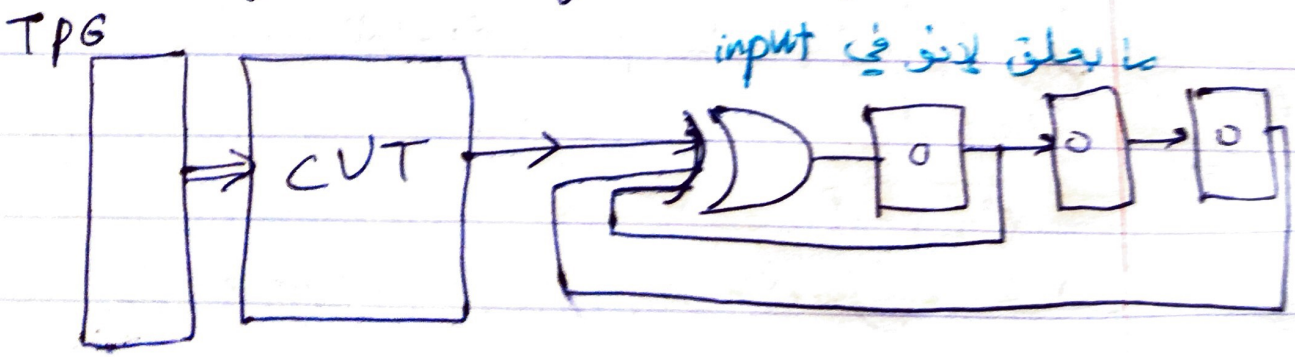
TPG
LFSR
PRPG

Result analyzer
"signature register"

LFSR : $2^n - 1$ case, almost randomly
بتبع عشوائي مش بالترتيب .

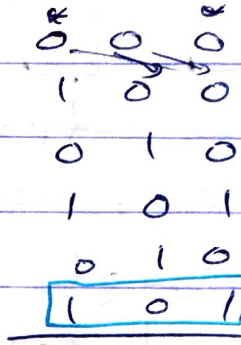
Signature Register

① single Input signature Register (SISR)



ما بعلق لانو في

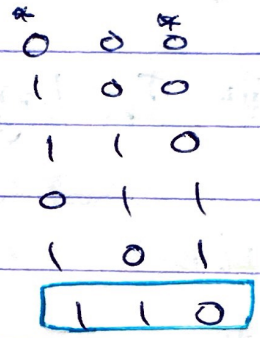
Ex: signature analyzer is initilized to all zero state, this signature register is fed with data stream 10111 (LSB arrive first). Find the good signature.



*	0	0	0
	1	0	0
	0	1	0
	1	0	1
	0	1	0
	1	0	1

good signature

assume in real test the data stream is 10101



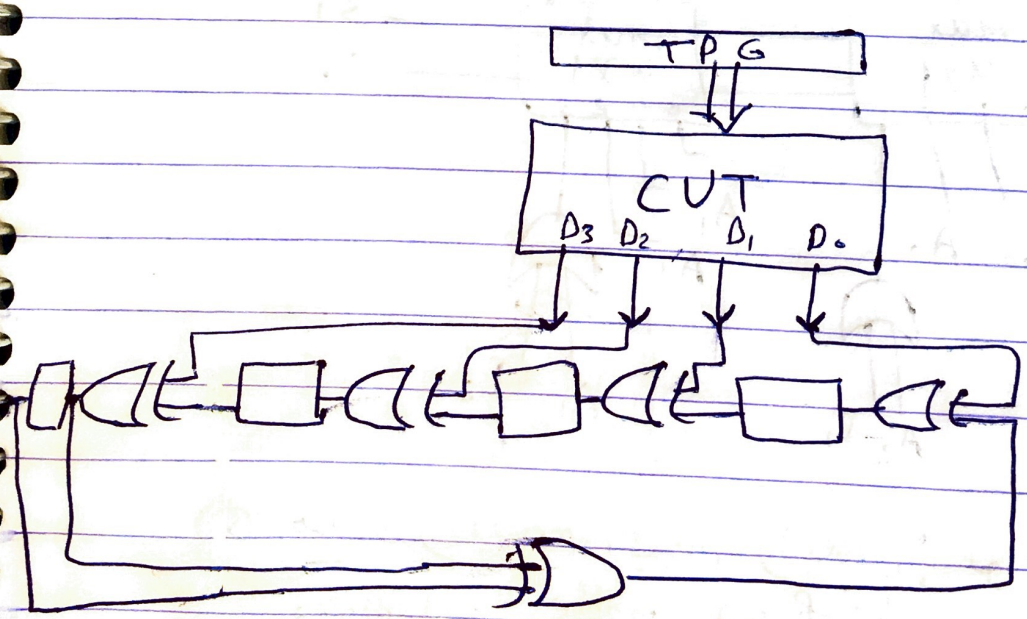
*	0	0	0
	1	0	0
	1	1	0
	0	1	1
	1	0	1
	1	1	0

bad signature (faulty CKT)

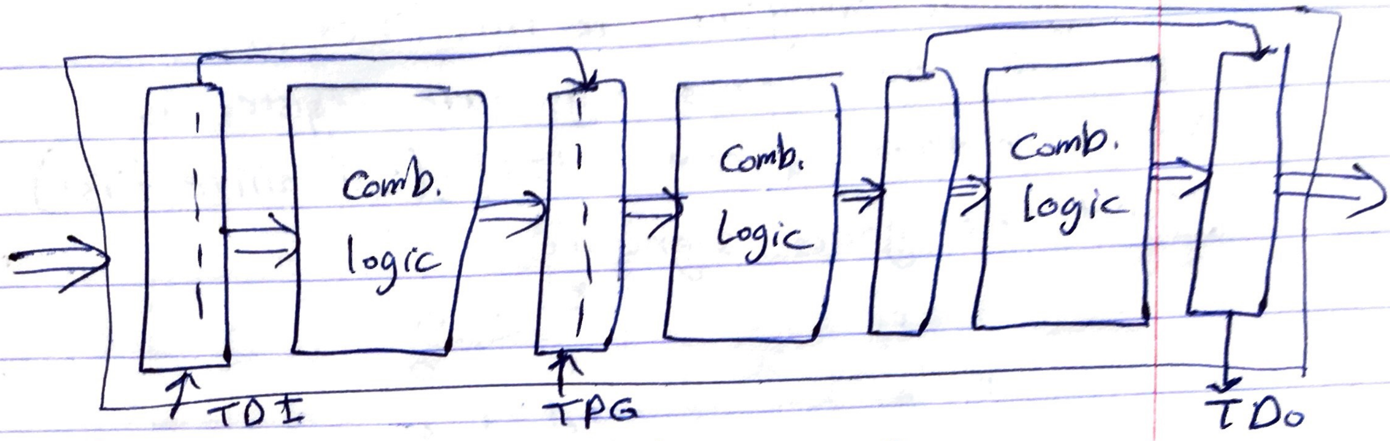
* احتمال الغلط يعطيني good signature

$\frac{1}{8} = \frac{1}{2^3}$ ← عدد الـ register

② parallel loading of signature register "Multiple input signature register" (MISR)



$\frac{1}{16} = \frac{1}{2^4}$ ← good * احتمال الغلط يعطيني



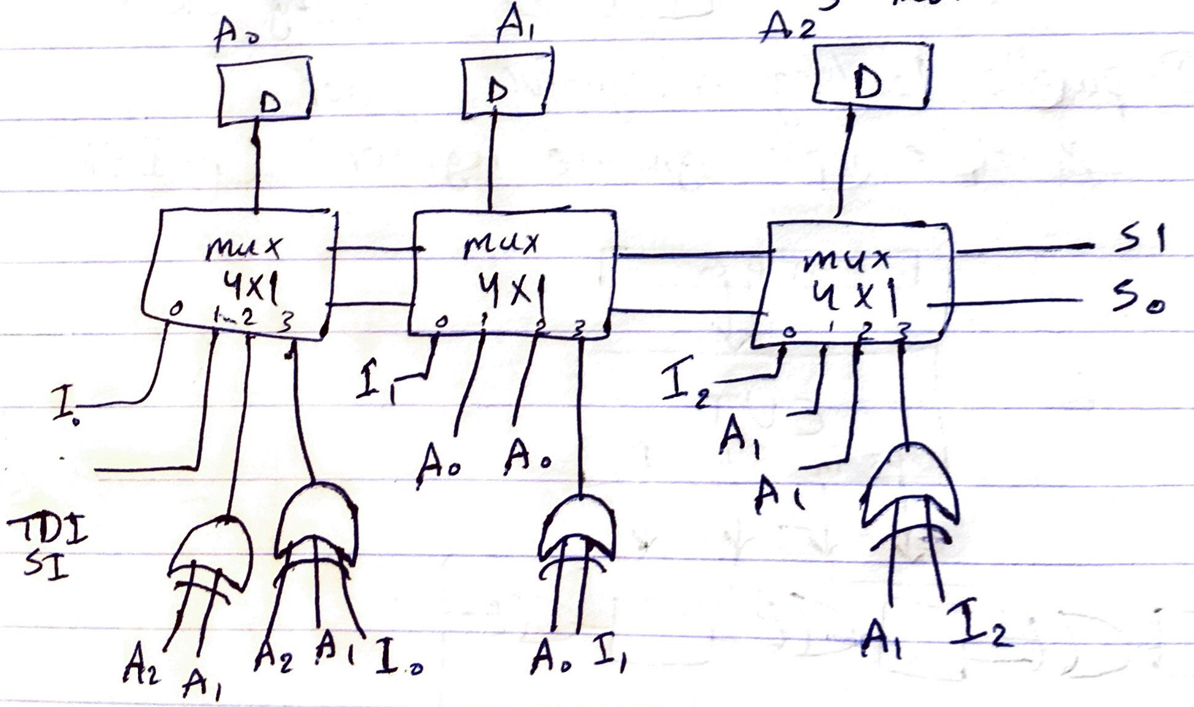
① Normal mode: normal register

Test mode: → ② LFSR "TPG"
→ ③ signature register

④ scan path register " بيدخل زي شفت

BILBO (Built In logic Block observation)

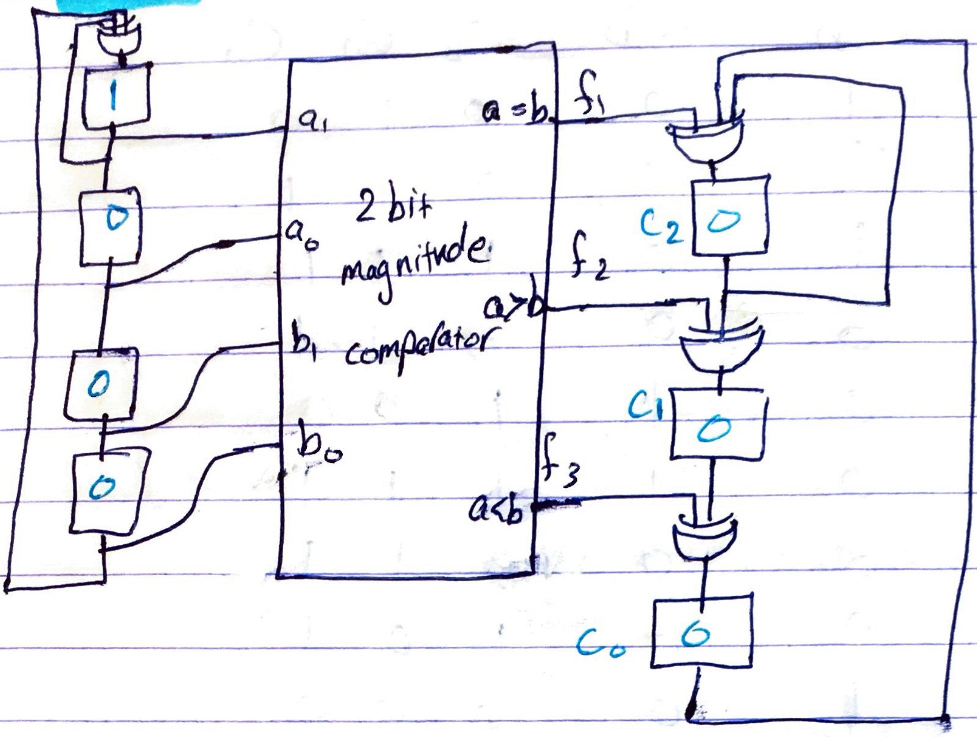
3-bit BILBO → 3 registers
3 mux-es 4x1



⊛ ما بقدر أفحص ثنين ورا بعض إلا لو بينهم ريجستر مشترك

بكون الـ mode → LFSR أو signature ما بزبطن مع بعض.

Ex: BIST Example



$C_2 = f_1 \oplus C_{2old} \oplus C_{0old}$

$C_1 = f_2 \oplus C_{2old}$

$C_0 = f_3 \oplus C_{1old}$

* LFSR ما بصير أبلش صفر
* signature بنحط أعتبرهم صفر بالبداية

* بحسب c من f وقيم c قبل

a_1	a_0	b_1	b_0	a=b (f_1)	a>b (f_2)	a<b (f_3)	c_2	c_1	c_0
1	0	0	0	0	1	0	0	1	0
1	1	0	0	0	1	0	0	1	1
1	1	1	0	0	1	0	1	1	1
1	1	1	1	1	0	0	1	1	1
0	1	1	1	0	0	1	0	1	0
1	0	1	1	0	0	1	0	0	0
0	1	0	1	1	0	0	1	0	0
1	0	1	0	1	0	0	0	1	0
1	1	0	1	0	1	0	0	1	1
0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	1	1	1	1
1	0	0	1	0	1	0	0	0	1
0	1	0	0	0	1	0	1	1	0
0	0	1	0	0	0	1	1	1	0
0	0	0	1	0	0	1	1	1	0

$C_2 = C_{20} \oplus C_{00} \oplus 1$

$C_1 = C_{20} \oplus f_2$

$C_0 = f_5 \oplus C_{10}$

cont. example assume $(a=b)$ saϕ

table

$a=b$	$a>b$	$a<b$	C_2	C_1	C_0	
0	1	0	0	1	0	
0	1	0	0	1	1	
0	1	0	1	1	1	
0	0	0	0	1	1	
0	0	1	1	0	0	
0	0	1	1	1	1	
0	0	0	0	1	1	
0	0	0	1	0	1	
0	1	0	0	0	0	
0	0	1	0	0	1	
0	0	1	1	0	1	
0	1	0	0	0	0	
0	1	0	0	1	0	
0	0	1	0	0	0	
0	0	1	0	0	1	bad signature

(faulty CKT)

⊕ كل ما زاد عدد الريجسترز بقل احتمال أوصل للصح من faulty CKT.

Lec 21:

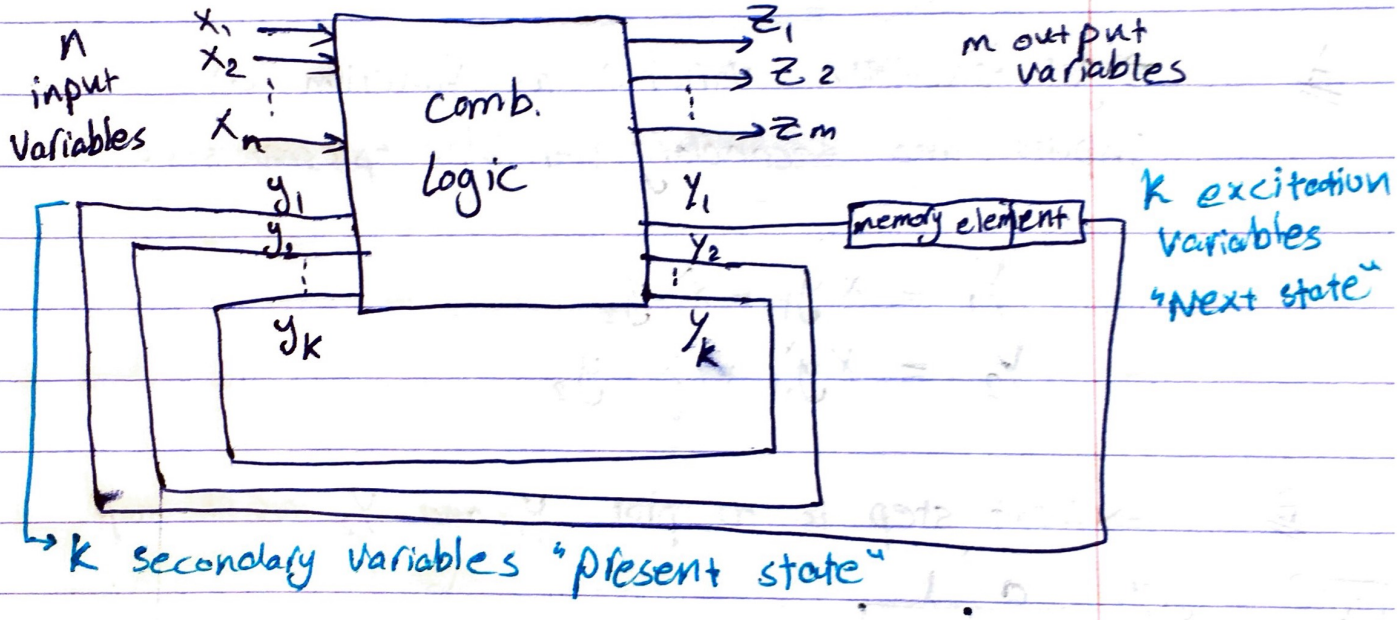
✱ Asynchronous sequential Logic

✱ نفس الـ clk ⇐ synchronous

why Asynchronous ?

1. Less limitation on speed.
2. power saving: الـ clk بتشغل كل القطع أما لما asynchronous بشغل القطعة اللي لازم تشتغل هيا

✱ بس بصير بيو شغل زيادة على توجيه الـ data صح.



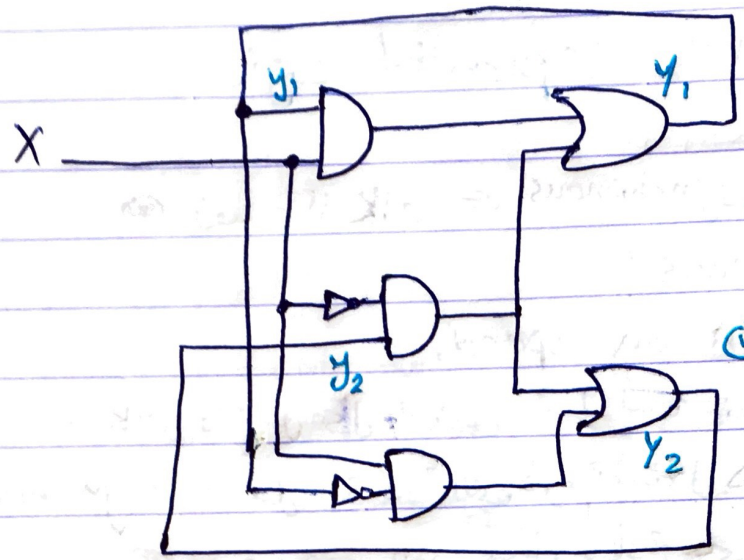
✱ circuit is stable when $y_i = Y_i$ for all $i = 1, 2, \ldots$

✱ Fundamental mode of operation:

Assume one input change at a time and only when the circuit is stable.

✱ هاد اللي رح نتعلمه.

Analysis Example:



① ← feed back so sequential

② no common clk so Asynchronous

③ 1 input

④ 2 excitation Y_1, Y_2
2 secondary y_1, y_2

A ⇒ get excitation variable as function of inputs and secondary variables "present state"

$$Y_1 = x y_1 + x' y_2$$
$$Y_2 = x y_1' + x' y_2$$

B ⇒ Next step is to plot Y_1 and Y_2 as a map.

Y_1

$y_1 y_2$ \ x	0	1
00	0	0
01	1	0
11	1	1
10	0	1

بفصل الانبوت عن ال cp.

$$Y_1 = x y_1 + x' y_2$$

Y_2

$y_1 y_2$ \ x	0	1
00	0	1
01	1	1
11	1	0
10	0	0

$$Y_2 = x y_1' + x' y_2$$

⇒ Next step is to find the transition table and show $Y = Y_1 Y_2$ inside each s

$y_1 y_2$ \ x	$x=0$ ($Y_1 Y_2$)	$x=1$ ($Y_1 Y_2$)
00	**00**	01
01	11	**01**
11	**11**	10
10	00	**10**

(bold = stable state)

□ system is stable $y_1 y_2 = Y_1 Y_2$

* لمشي ل state لحظياً بعدين بكمل لحد ما يوصل لـ stable state ، "بعدما إذا غيرت الانبوت بتغير وبكمل.."

* $Y = y$ circuit to represent stable state.

* total state is a combination of internal state and inputs

The previous circuit has 4 stable total states:

$y_1 y_2 x$ ($y_1 y_2$ x) = 000, 011, 110, 101

* Analysis with flow table

EX:

	$x=0$	$x=1$
a	**a**	b
b	c	**b**
c	**c**	d
d	a	**d**

EX:

	$x_1 x_2$ = 00	01	11	10
a	**a, 0**	**a, 0**	**a, 0**	b, 0
b	a, 0	a, 0	**b, 1**	**b, 0**

2 inputs

2 states ⇒ 1 state variable

1 output

to change this to transition table

⇒ state assignment

assign a=0 , b=1

- find the equations for next state & output.

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

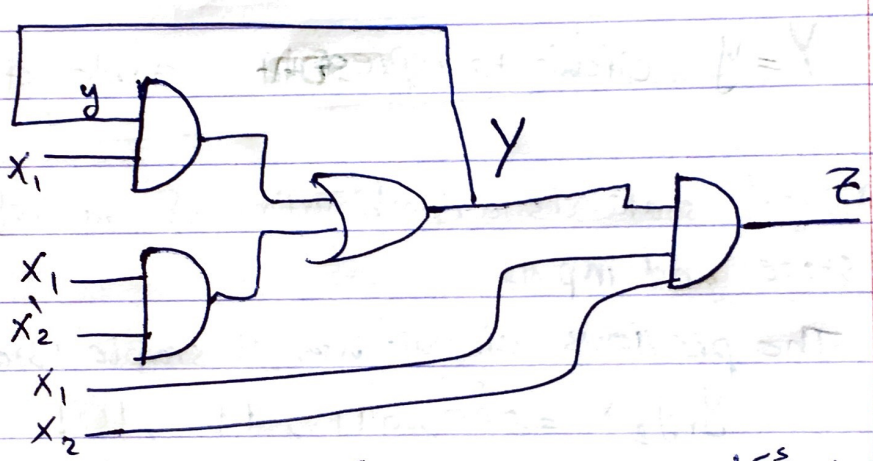
a , b

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

← output

$$Y = x_1 y + x_1 x_2'$$

$$Z = x_1 x_2 y$$



* لو أكثر من state Variable لازم أدمج قبل، عشان أعرف الـ stable state,

* Race condition:

* إذا غيرت أكثر من state وبدهن يتغيرن مع بعض.

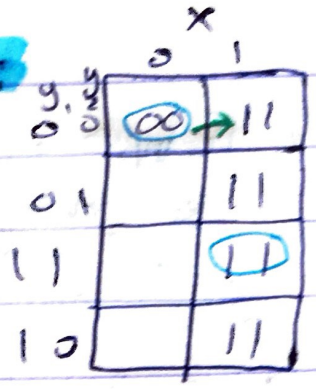
Ex:

x \	0	1
	00	01

when we change x from 0 to 1 while we are in total state $y_1 y_2 x$

0 0 0 ⇒ no race

Ex:



x
0 → 1

Race ? ⇒ Yes

① 00 → 11

total stable state $y_1 y_2 x = 111$

② 00 → 01 → 11

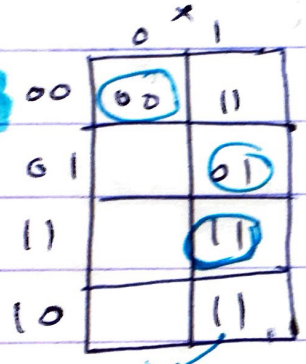
total stable state $y_1 y_2 x = 111$

③ 00 → 10 → 11

total stable state = 111

there is a race, but noncritical race.

Ex:



critical لو 10 بروح

① 00 → 11

total stable state = 111

② 00 → 01

total stable state = 011

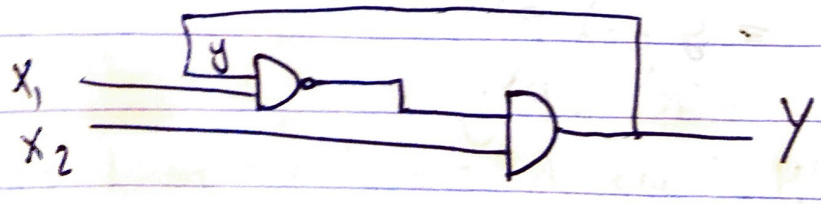
③ 00 → 10 → 11

total stable state = 111

⇒ critical Race.

Lec 22:

Ex: unstable circuit:



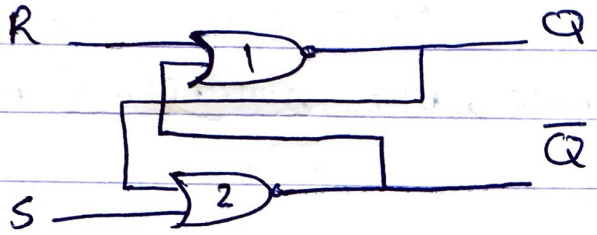
$$Y = (X_1 Y)' \cdot X_2$$

y \ X_1X_2	00	01	11	10
0	(0)	1	1 ↑	(0)
1	0	(1)	0 ↓	0

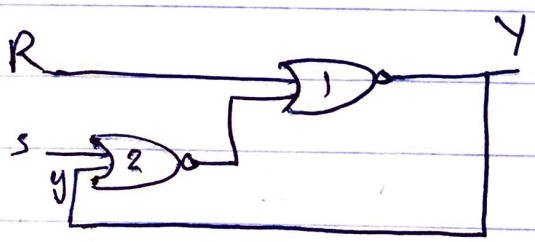
if input = 11
state changes from 0 to 1 and from 1 to 0 infinitely.

Circuits with Latches:

⊛ SR Latch



S	R	Q	$\overline{Q}$
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0 = undefined



y \ SR	00	01	11	10
0	(0)	(0)	0	1
1	(1)	0	0	(1)

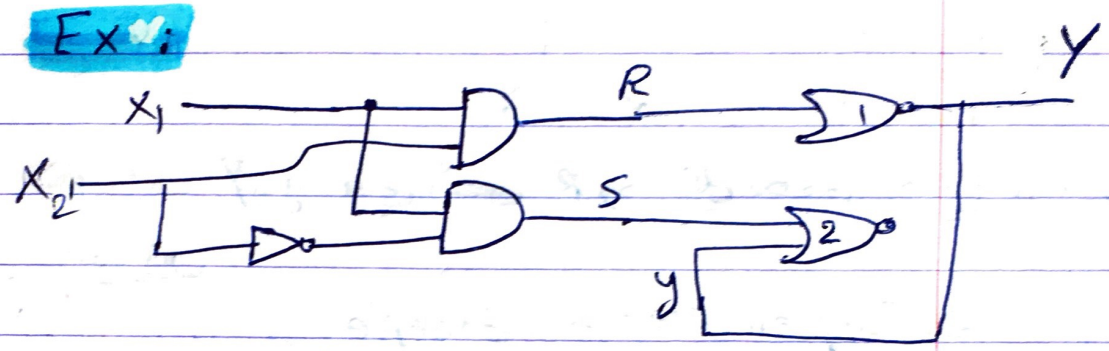
$$Y = ((y+S)' + R)'$$
$$= (y+S)\,R'$$
$$= SR' + R'y$$

Usually S and R should not be 1 on the same time ⇒ $SR = 0$

$$Y = SR' + R'y + 0$$
$$= SR' + S\bar{R} + R'y$$
$$= S(R' + R) + R'y$$
$$= S + R'y$$

Excitation function of SR (nor gate)

Ex:



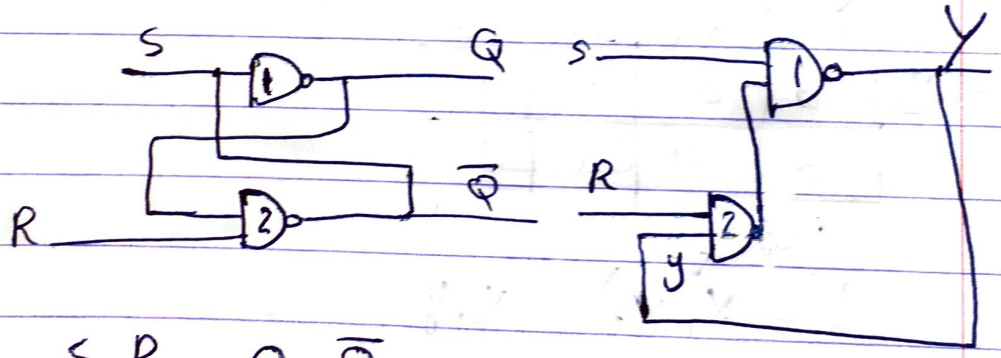
$S = X_1 \cdot X_2'$

$R = X_1 \cdot X_2$ $\quad SR = (X_1 X_2') \cdot (X_1 X_2)$

$= 0$

$Y = S + R'y = (X_1 X_2') + (X_1 \cdot X_2)' y$

SR using 2 NAND gates:



S	R	Q	$\bar{Q}$
0	1	1	0
1	1	1	0
1	0	0	1
1	1	0	1
0	0	1	1 → undefined

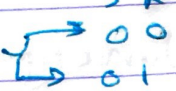
$Y = ((y \cdot R)' \cdot S)'$

$= yR + S'$

⊛ Latch Excitation Table

nor :

y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



⊛ بعرف Y و y بدي أعرف R و S اللي بحققن هاي الحالة بهدف البيزاين .

⊛ Implementation Example

flow table

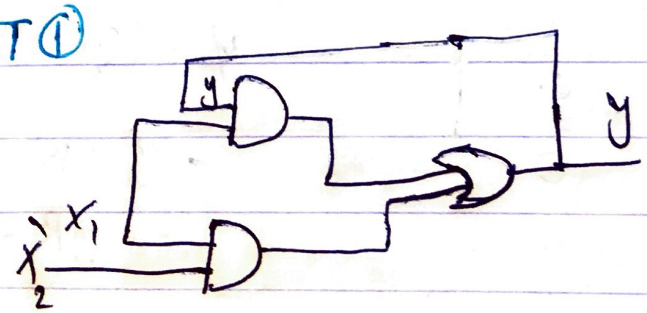
	x_1x_2 = 00	01	11	10
a	(a)	(a)	(a)	b
b	a	a	(b)	(b)

assign a = 0 b = 1

y	x_1x_2 = 00	01	11	10
0	0	0	0	(1)
1	0	0	(1)	(1)

$$Y = x_1 x_2' + x_1 y$$

CKT ①



to use SR latch

S

y \ x_1x_2	00	01	11	10
0	0	0	0	1
1	0	0	x	x

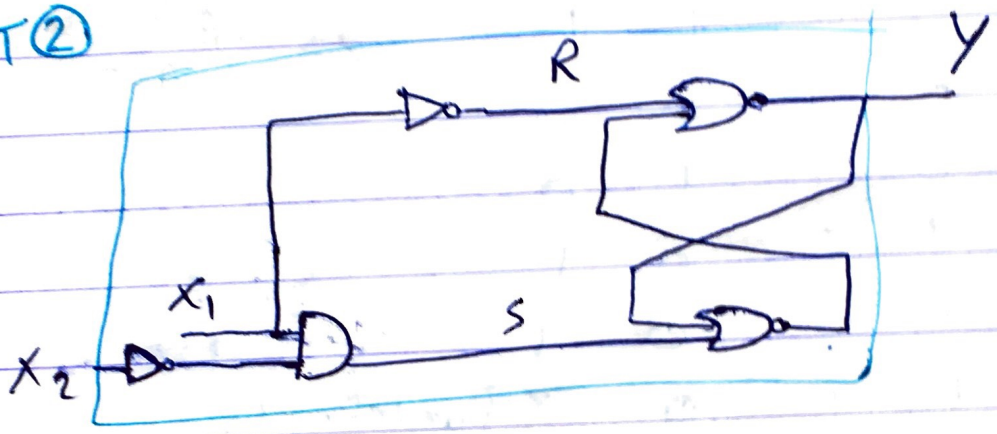
R

y \ x_1x_2	00	01	11	10
0	x	x	x	0
1	1	1	0	0

④ تعطي ثبات أكثر من feed back عادي.

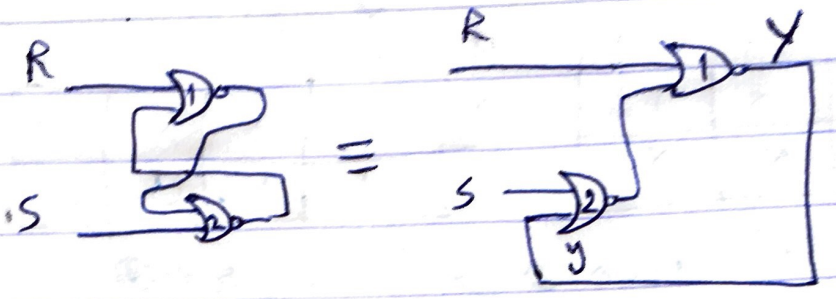
$S = x_1 x_2'$ $\quad R = x_1'$

CKT ②



CKT ① ≡ CKT ②
but using SR latch

Lec 23:



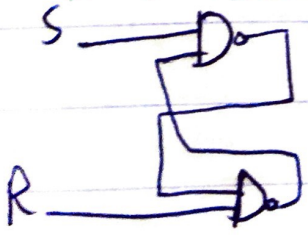
S	R	y	Y	
0	0	0	0	
0	0	1	1	← SR Latch 9-3
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	X	} undefined unwanted (S.R = 0)
1	1	1	X	

$$Y = S + R'y$$

Excitation table of SR latch

y	Y	S	R	
0	0	0	X	no change of reset
0	1	1	0	
1	0	0	1	
1	1	X	0	no change of set

في السلايدات معطوفة بالعلط



$$Y = S + S'Ry$$

$Y_{nor} = S + R' y$ $\Rightarrow$ $S_{nand} = S'_{nor}$

$Y_{nand} = S' + Ry$ $R_{nand} = R'_{nor}$

Analysis example:

$S_1 = x_1 y_2$ $R_1 = x_1' x_2'$ $\Rightarrow S_1 R_1 = 0$ ✓

$Y_1 = S_1 + R_1' y_1$ إذن يقرأ استخدم

$= x_1 y_2 + (x_1' x_2')' y_1$

$S_2 = x_1 x_2$ $R_2 = y_1 x_2'$ $\Rightarrow S_2 \cdot R_2 = 0$ ✓

$Y_2 = S_2 + R_2' y_2$ يقرأ استخدم

$= x_1 x_2 + (y_1 x_2')' y_2$

$y_1 y_2$ \ $x_1 x_2$	00	01	11	10
00	(00)	00	→ 01	00
01	01	(01)	↓ 11	11
11	00 ←	11	(11)	10
10	00	(10)	11	(10)

← التعويض

← transition table

لو صارت x_1 واحد لما أتبع السيركت يروح على 01 بعدين 11 10

Race: Y_1 & Y_2 changes together

$11 \rightarrow 00$ $(y_1 y_2 \; x_1 x_2 : 00 \; 00)$

$11 \rightarrow 10$ $(y_1 y_2 \; x_1 x_2 : 00 \; 00)$

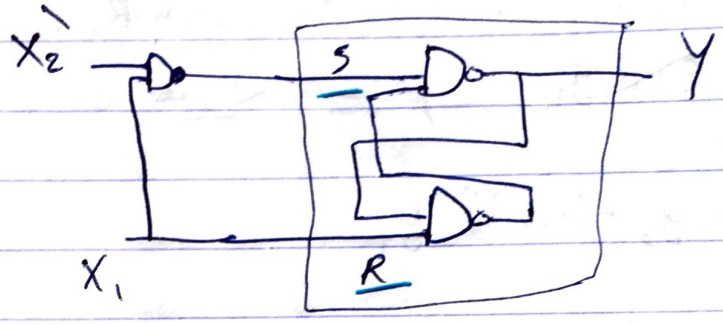
$11 \rightarrow 01$ $(y_1 y_2 \; x_1 x_2 : 01 \; 00)$ ← critical race

⇒ last ex on Lec 22:

using nand

$S_{nand} = S'_{nor} = (X_1 X_2')'$

$R_{nand} = R'_{nor} = X_1$



Design example:

if G=1 ⇒ Q=D

if G=0 ⇒ Q no change ← sequential memory element

Design procedure:

1 Get primitive flow table.

↳ only one stable state per line "one & only one"

D	G	Q	stable ??
0	0	0	stable
0	0	1	stable
0	1	0	stable
0	1	1	no
1	0	0	stable
1	0	1	stable
1	1	0	no
1	1	1	stable

stable states عدد ← عدد الأسطر

2^{inputs} عدد الأعمدة

مهمات ↙

state	inputs D G	output Q	comments
a	0 1	0	after b,c,f
b	1 1	1	after d,e,a
c	0 0	0	after a,d
d	1 0	0	after c
e	1 0	1	after b,f
f	0 0	1	after e

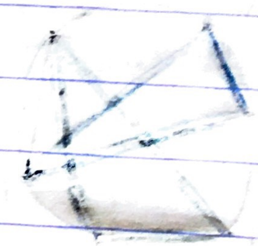
plimitive flow table

DG	00	01	11	10
a	c,-	(a),0	b,-	-,-
b	-,-	a,-	(b),1	e,-
c	(c),0	a,-	-,-	d,-
d	c,-	-,-	b,-	(d),0
e	f,-	-,-	b,-	(e),1
f	(f),1	a,-	-,-	e,-

-,- ممنوع أو على stable من

حسب ال comments

في المرحلة الأوتبوت ما بهمني -



Lec 24:

compatible states: متوافقة

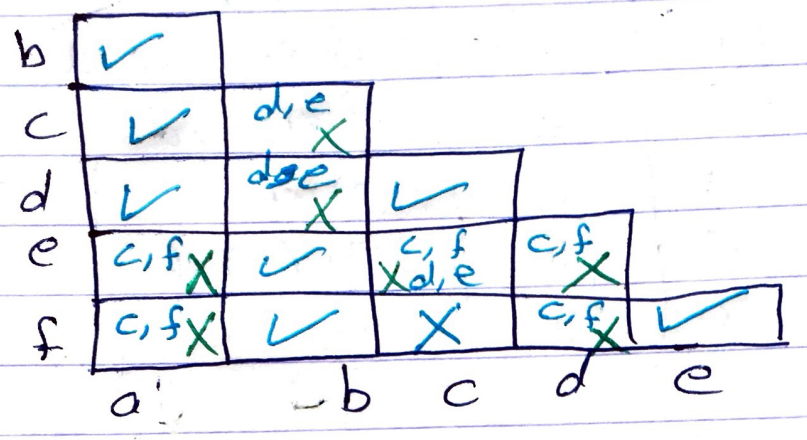
↲ لما الأوتبوت محدد لازم يكون متساوي ، لما مش محدد بشوف ال state.

equi. (a,b) (a,c) ⇒ (c,b) equiv

comp. (a,c) (b,c) ⇏ (a,b) → بوضع وحدة a

comp. a,b يكونوا مش شرط

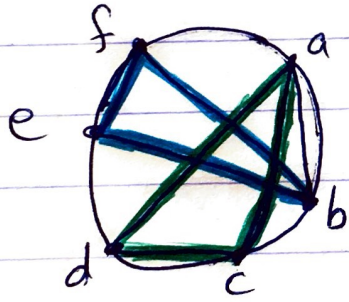
implication chart



Compatible pairs:

(a,b), (a,c), (a,d), (b,e), (b,f), (c,d), (e,f)

Maximal compatibles (merger diagram)



(a, c, d)
(b, e, f)
(a, b)

minimum set of maximal compatibles that covers all states and closed.

* لما كل ال ✓ ما في معهم شروط بطلع closed بدون ما أفصلها.

(a, c, d) ⟹ covers all states & closed
(b, e, f)

DG

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

أوتوماتة الحدة

DG

	00	01	11	10
A	A,0	A,0	B,—	A,0
B	B,1	A,—	B,1	B,1

من صفر لواحد — من صفر لواحد

state assignment A=0 B=1

DG

y	00	01	11	10
0	0	0	1	0
1	1	0	1	1

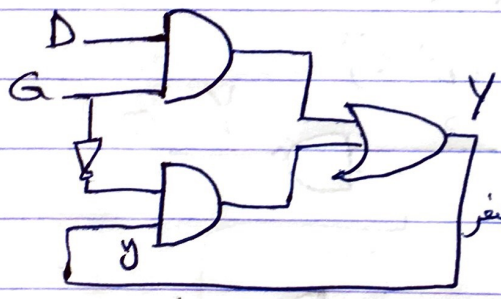
DG

y	00	01	11	10
0	0	0	X	0
1	1	X	1	1

Q = y

$Y = DG + G'y$

إما 0 أو 1 أو X



- بدي أنتقل بكل الحالات من صفر لصفر ← 0
- من واحد لواحد ← 1
- من صفر لواحد أو واحد لصفر ← X

لنشوف من كل الحالات إذا 1 والصفر أقوى من X

Implementation using SR latches:

Excitation table of SR

y	Y	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Y DG

y \ DG	00	01	11	10
0	0	0	1	0
1	1	0	1	1

DG

y \ DG	00	01	11	10
0	0	0	1	0
1	X	0	X	X

$S = DG$

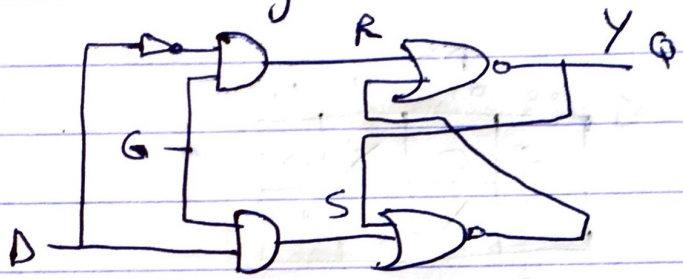
GG

y \ GG	00	01	11	10
0	X	X	0	X
1	0	1	0	0

$R = D'G$

لما أطلع SR بجدول عالطويل مش عالمعادلة لإنو ممكن يطلع غير.

using nor



* excitation قبل ما أجيب

$S = DG$

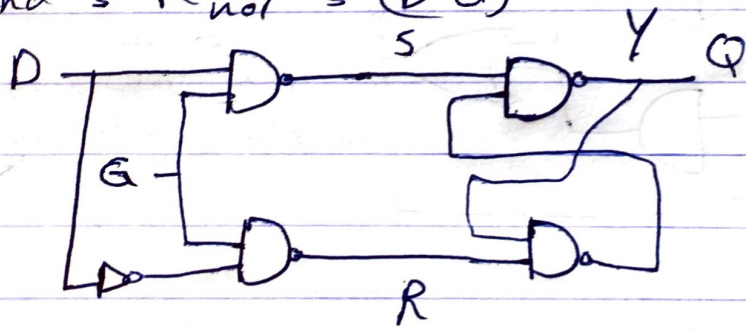
$R = G$ $SR \neq 0$

$Y = S + R'y$

using nand gates only

$S_{nand} = (S_{nor})' = (DG)'$

$R_{nand} = R'_{nor} = (D'G)'$



Ex: Assigning output to unstable states,

X \	0	1
a	(a),0	b,0
b	c,–	(b),0
c	(c),1	d,–
d	a,–	(d),1

$\Rightarrow$

0	0
X	0
1	1
X	1

⊛ Closed covering condition

Example with implied states

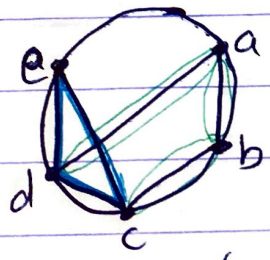
	a	b	c	d
b	b,c			
c	×	d,e		
d	b,c	×	a,d	
e	×	×	✓	b,c

(b, c) implies (d, e)
(d, e) implies (b, c)

compatible pairs

(a, b), (a, d), (b, c), (c, d), (c, e), (d, e)

Maximal comp (merger diagram)



لو بدي أوخذ a b cd لازم
a مع c في خط و b e d

(c, d, e)
(a, b)
(a, d)
(b, c)

closure table

compatible	a, b	a, d	b, c	c, d, e
implication	b, c	b, c	d, e	a, d b, c

شرطهم

Minimum set covers all states & closed

(c, d, e), (a, d), (b, c)

حل ثاني (a, b), (b, c), (d, e)

(a, d), (b, c), (d, e)

* اللي بوفيها بحقق شرطها مش شرط أحقق شروط الصدفة كلو.

Lec 25 :

* Race free state assignment.

Ex:

	0 (x)	1
a	(a)	d
B	c	(b)
c	(c)	b
d	(d)	a

a	00
b	01
c	10
d	11

آفي race كبير

a	00
b	01
c	11
d	10

ما في race

	x: 0	1
00	00	10
	11	01

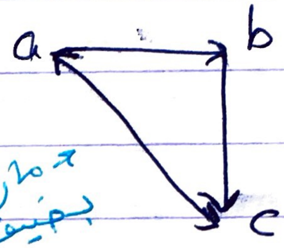
transition diagram:

Ex:

$x_1 x_2$

	00	01	11	10
a	(a)	b	c	(a)
b	a	(b)	(b)	c
c	a	(c)	(c)	(c)

transition diagram



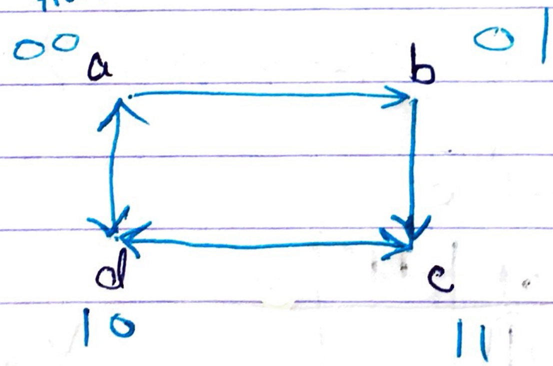
state assignment

a = 00

b = 01 ← race

c = 10

ما رح نربط بكل الاحتمالات
بنضيف state هاي بتكون transitions



$x_1 x_2$

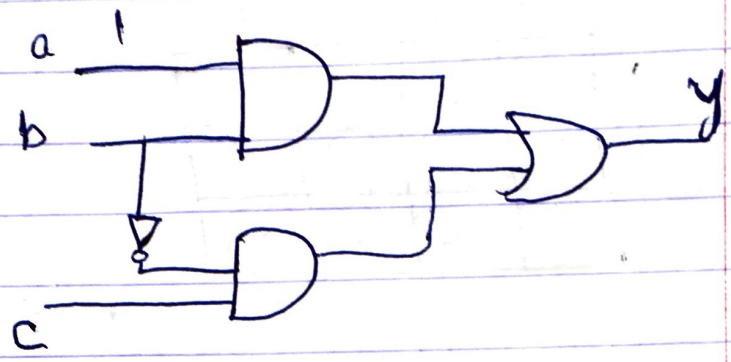
	00	01	11	10
a	(a)	b	d ←	(a)
b	a	(b)	(b)	c
c	d ←	(c)	(c)	(c)
d	a	—	c	—

— → don't care but not d

⊛ Hazards

- sequential CKT
- Combinational CKT

Ex:



1. a = c = b = 1 static output y = 1
2. 1 0 1 static output 1

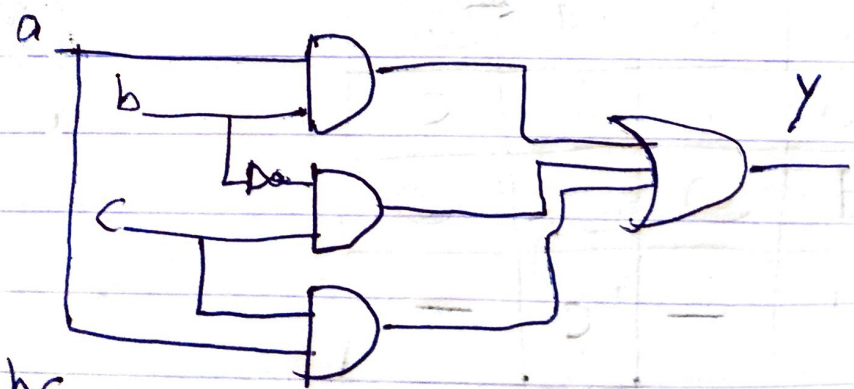
if gates have a delay ⇒ 10 ns between 1 & 2 0 appeared.

possible solutions; or with 20 ns delay
or add flip-flop after y

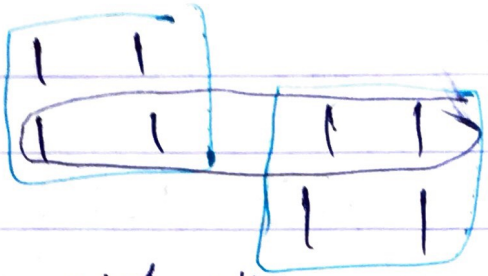
$y = ab + b'c$

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

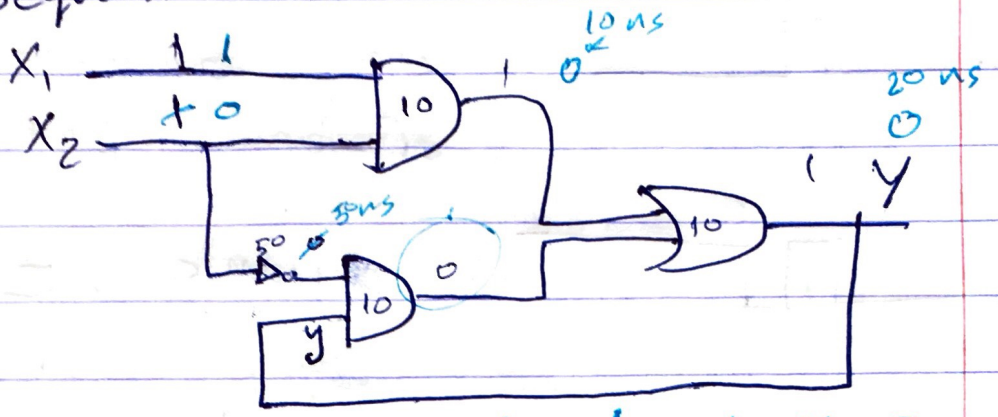
$y = ab + b'c + ac$



a bc
1 1 1 ⇒ y = 1
1 0 1 y = 1

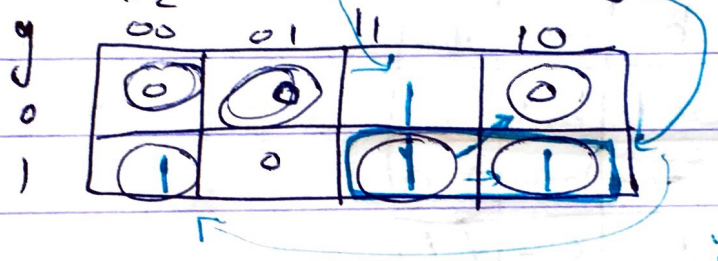


sequential CKT:



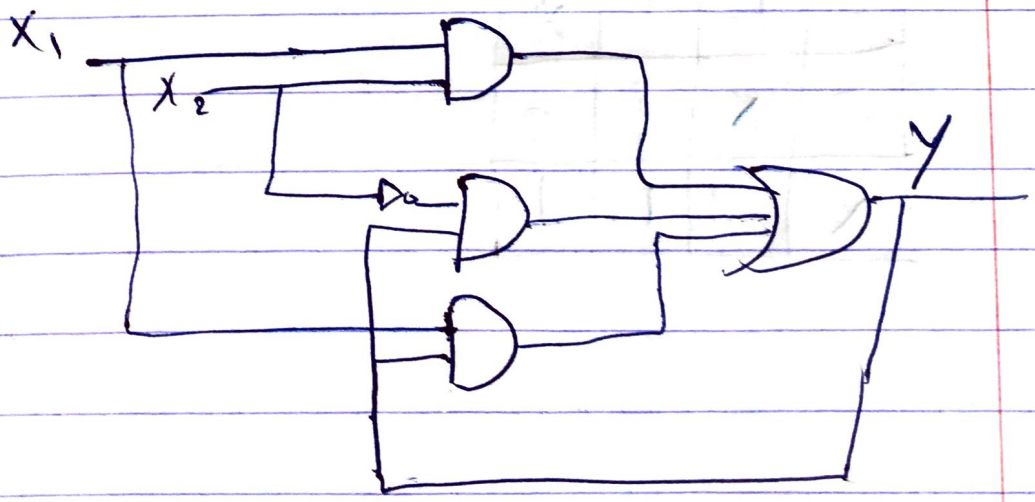
* يضل ثابت على صفر

$$Y = X_1X_2 + X_2'y$$



يتغير x_2 من 1 الصفر
يتطلع على 0 transition
بس 0 لهيك ثبت stable

$$Y = X_1X_2 + X_2'y + X_1y$$

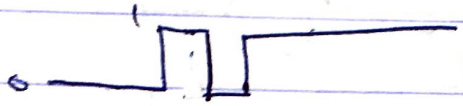


* SR latch: this hazard is avoidable.

Hazard types:

① static -1- hazard
and-or مجموعة

② static-0- hazard
or-and implementation مجموعة



③ dynamic hazard

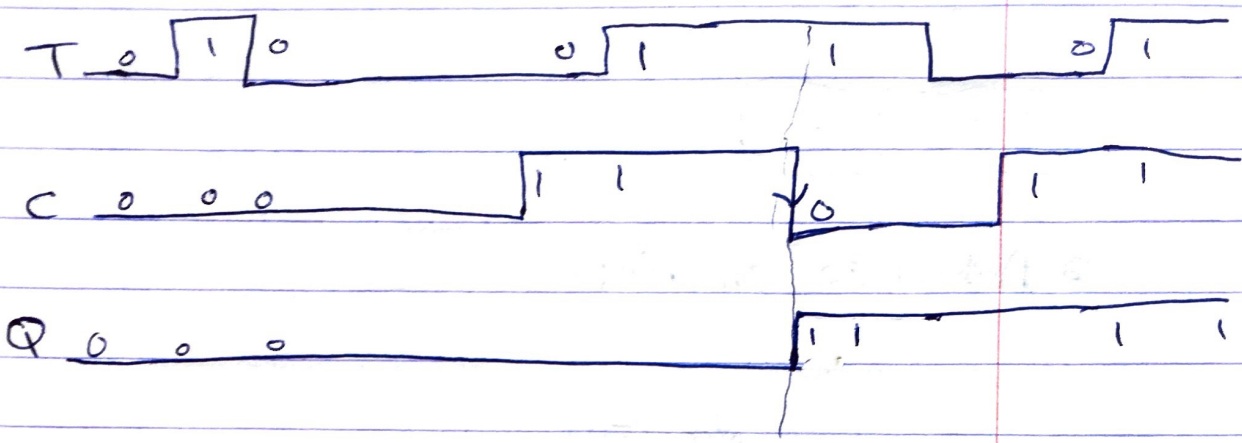
او 2 أهم الشيء.

Ex:

	x_1x_2 = 00	01	11	10
a	(a), 0	b, -	-, -	d, -
b	a, -	(b), 1	(b), 1	c, -
c	b, -	-, -	b, -	(c), 0
d	c, -	(d), 1	c, -	(d), 1

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

Lec 26:

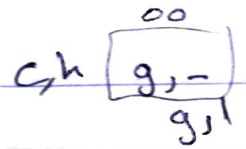


On negative edge all states are stable.

9.8 Design Example

	T	C	Q	comments
a	1	1	0	after d, f
b	1	0	1	a, g
c	1	1	1	b, h
d	1	0	0	c, e
e	0	0	0	d, f
f	0	1	0	e, a
g	0	0	1	b, h
h	0	1	1	g, c

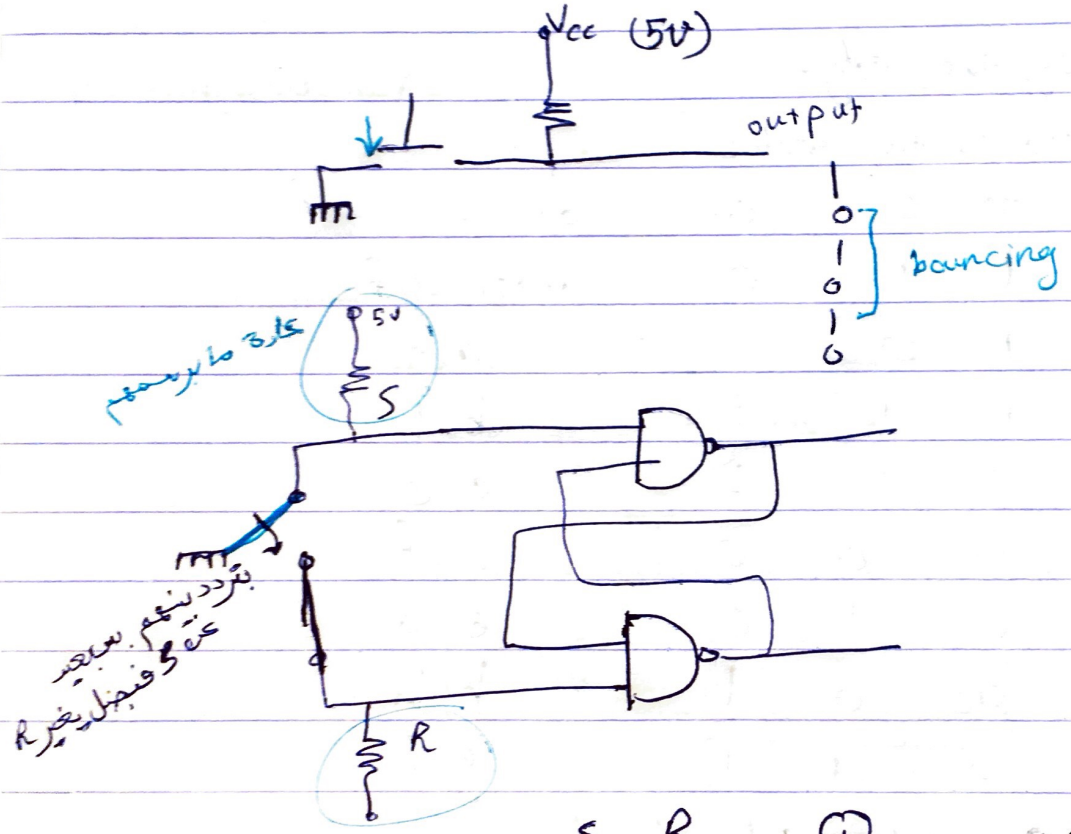
	00	01	11	10
a	-,-	f,-	(a),0	b,-
b	g,-	-,-	c,-	(b),1
c	-,-	b,-	(c),1	d,-
d	e,-	-,-	a,-	(d),0
e	(e),0	f,-	-,-	d,-
f	e,-	(f),0	a,-	-,-
g	(g),1	h,-	-,-	b,-
h	g,-	(h),1	c,-	-,-



في السلايدات

Lec 27:

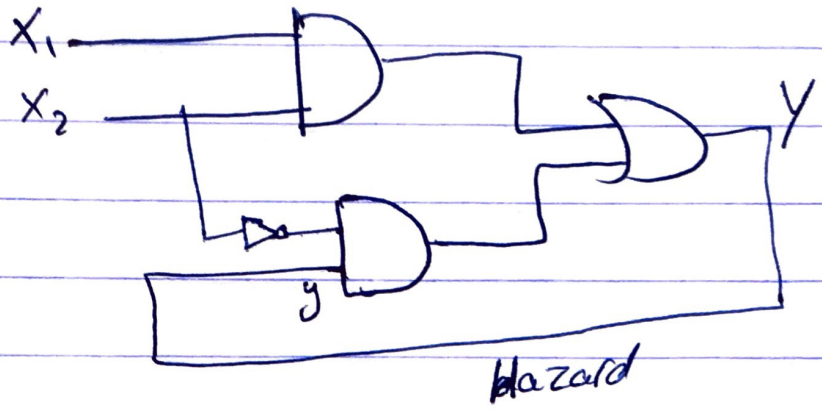
✱ Debounce circuit:



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

bouncing

الجواب مع وجود bouncing



Hazard

VHDL: coding ⇒ only state diagram

لكن مهم فهم كل اشي والفروق بين الأنواع زي الـ buffer والـ variable و الـ signal.