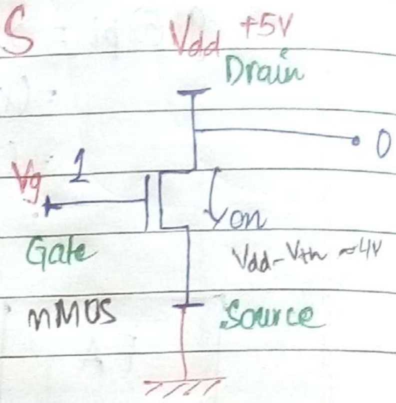
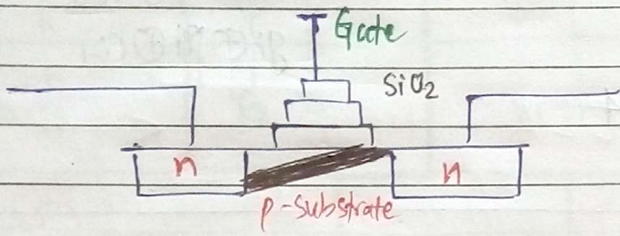
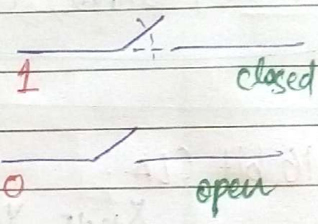
Orion
PAGE:
DATE: 22/02/17

REALIZING LOGICS WITH CMOS



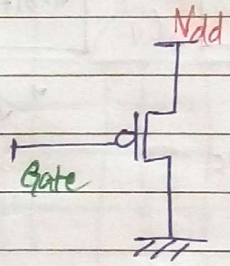
pMOS
nMOS } can act as a switch

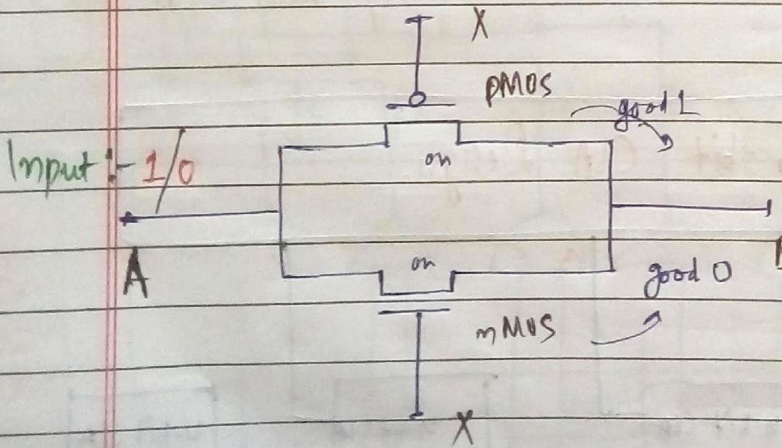


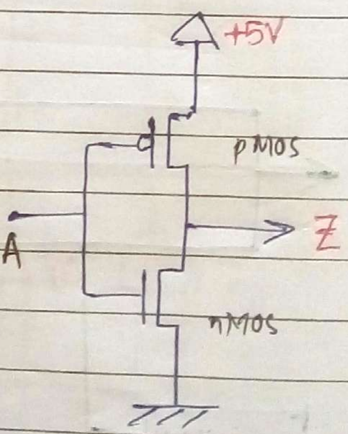


nMOS passes a good 0
nMOS passes a poor 1.

pMOS passes a good 1
pMOS passes a poor 0.

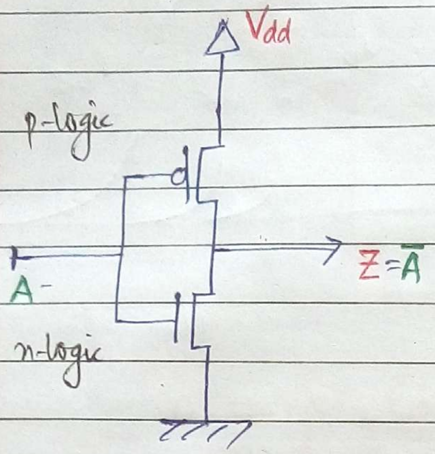






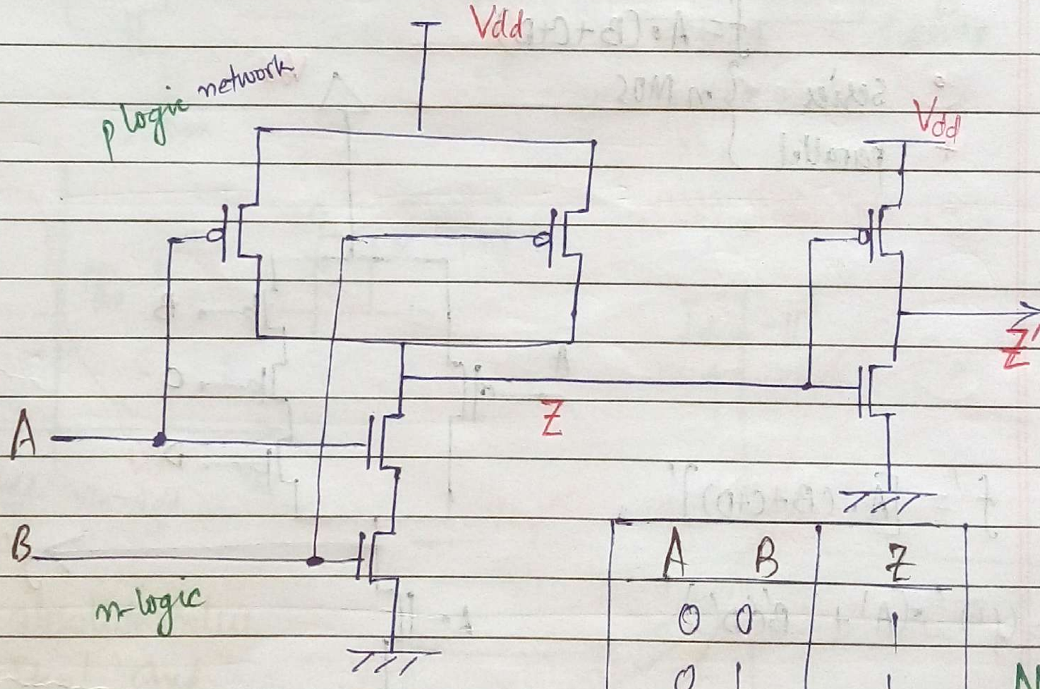
$\boxed{\text{CMOS}}$ inverter

			nMOS	pMOS
Complemented Switch	A=0	Z=1	off	on → passes good 1
	A=1	Z=0	on → passes good 0	off



2 - input NAND Gate

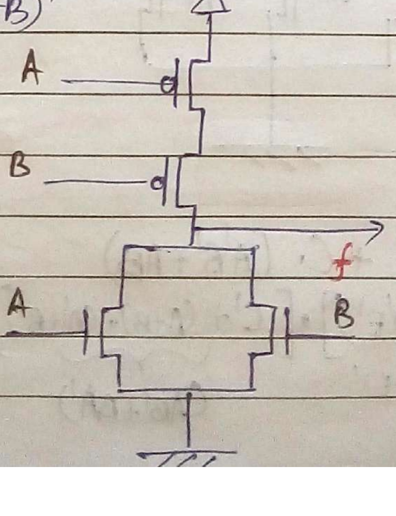
And - $((A \cdot B)')'$

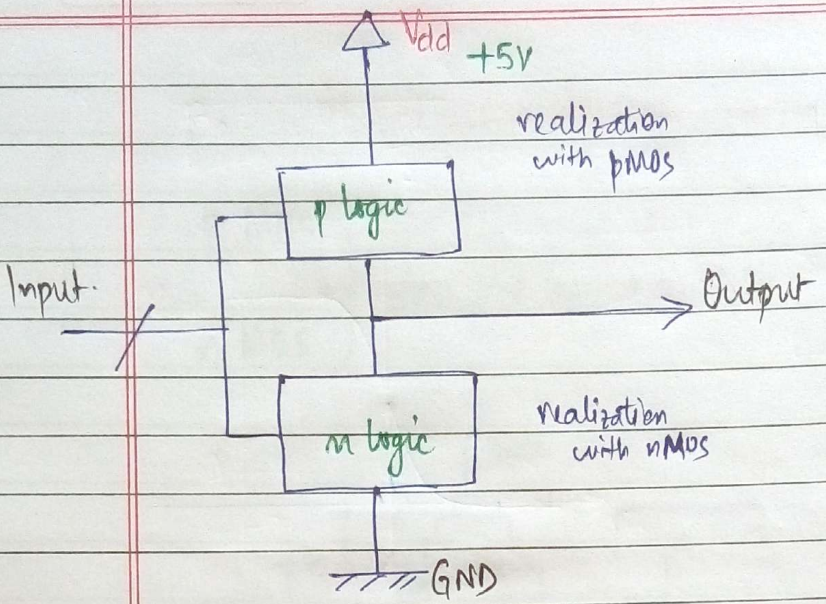


A	B	Z
0	0	1
0	1	1
1	0	1
1	1	0

NAND

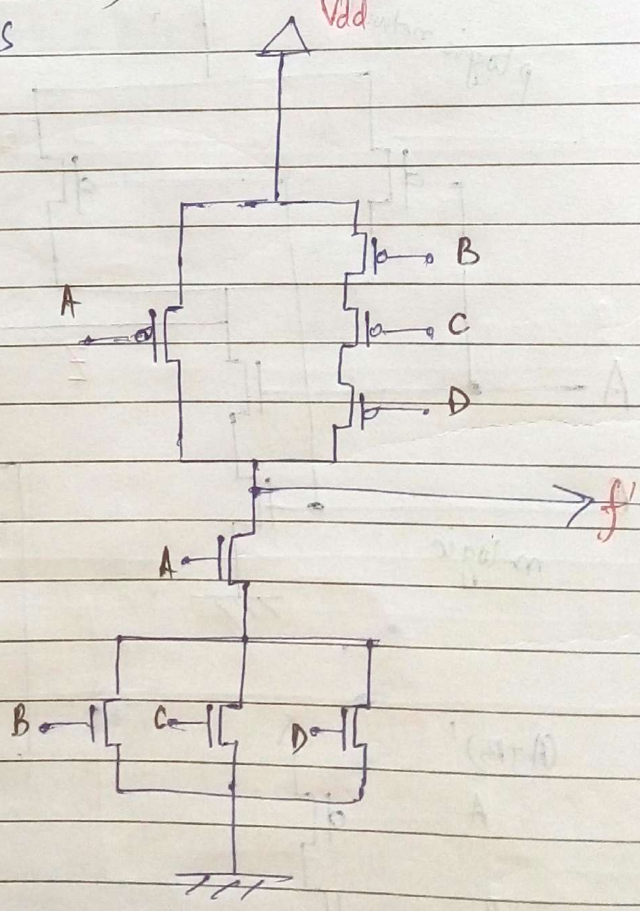
NOR $(A+B)'$





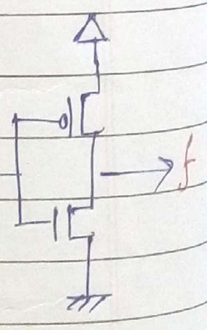
$f = A \cdot (B + C + D)$

$\cdot$ series } nMOS
$+$ parallel



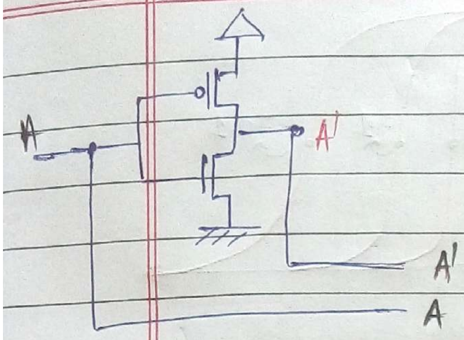
$f' = [A \cdot (B+C+D)]'$

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



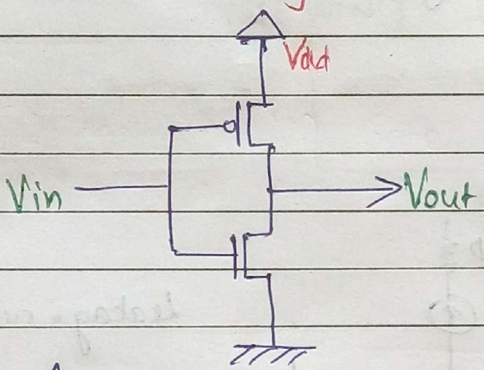
$f = (A+B) \cdot (C+D+E) + C \cdot (A'B' + AB)$

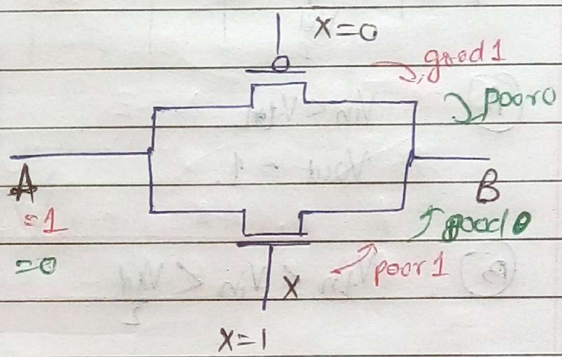
$f = (f')' = \left\{ \left[\underset{\downarrow \atop (A+B)'}{(A'B')} + \underset{\downarrow \atop (C+D+E)'}{(C'D'E')} \right] \cdot \left[C' + \underbrace{(A+B)\cdot(A'+B')}_{(AB'+BA')} \right] \right\}'$



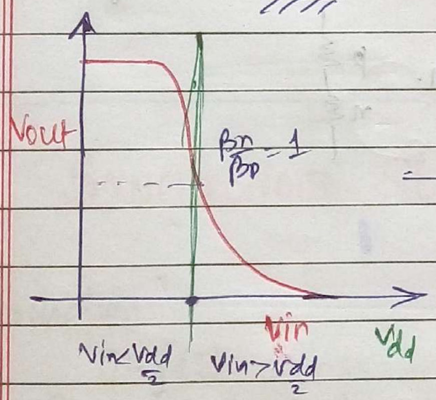
for each complemented input, 2 transistors are required.

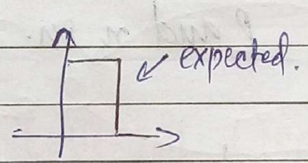
Transfer Characteristics of CMOS Inverter

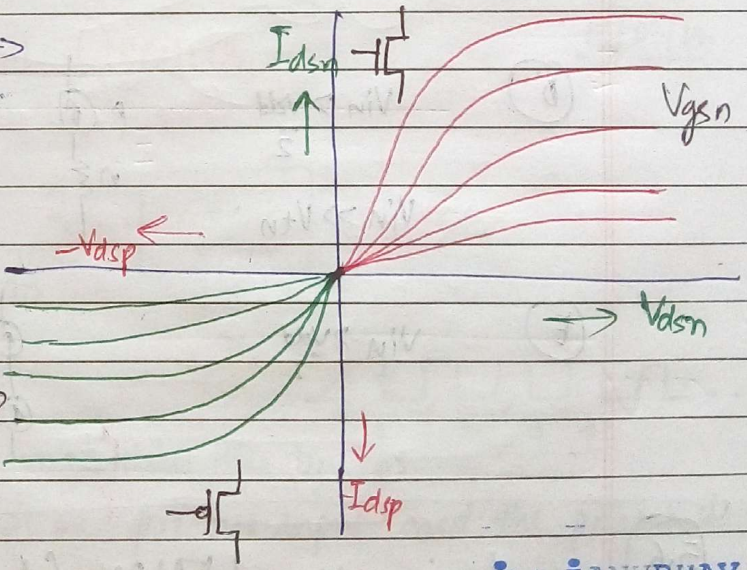




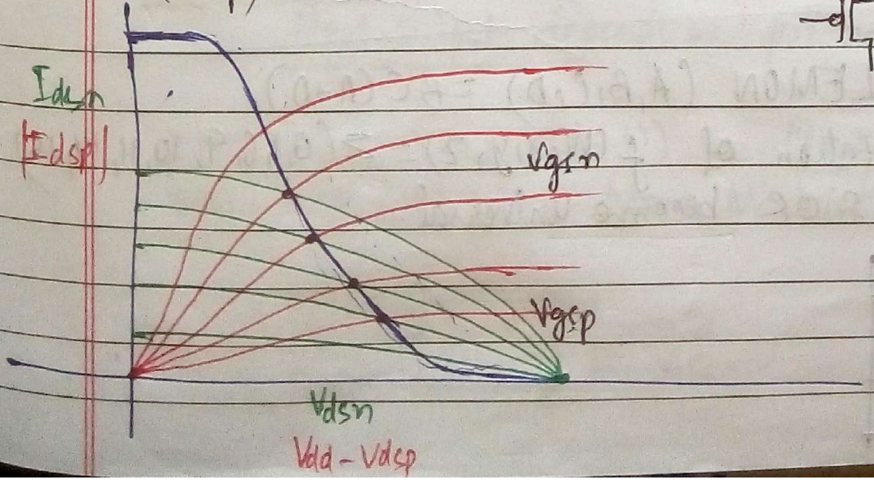
pass transistor





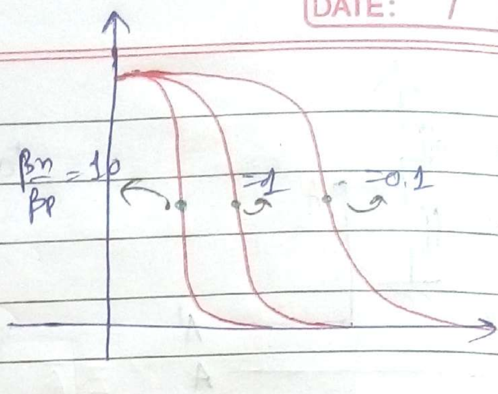


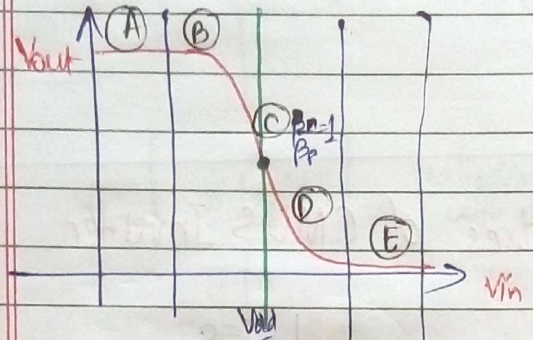
Take absolute value of |Idsp| and |Vdsp)



Some same value for Vgsp and Vgsn which gives you Vin of a inverter.

$\beta_n = \left(\frac{\mu \epsilon}{t}\right)\frac{W}{L}$



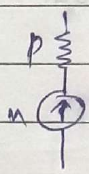


Five regions in the curve space.

(A) $V_{in} < V_{tn}$

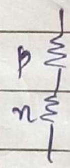
$V_{out} = 1.$

(B) $V_{tn} \le V_{in} < \frac{V_{dd}}{2}$



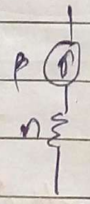
Leakage current.

(C) $V_{in} = \frac{V_{dd}}{2}$ p and n on.



(D) $V_{in} > \frac{V_{dd}}{2}$

$V_{in} >> V_{tn}$



(E) $V_{in} > \frac{V_{dd}}{2}$

5.6 4-in gate LEMON $(A,B,C,D) = BC(A+D)$

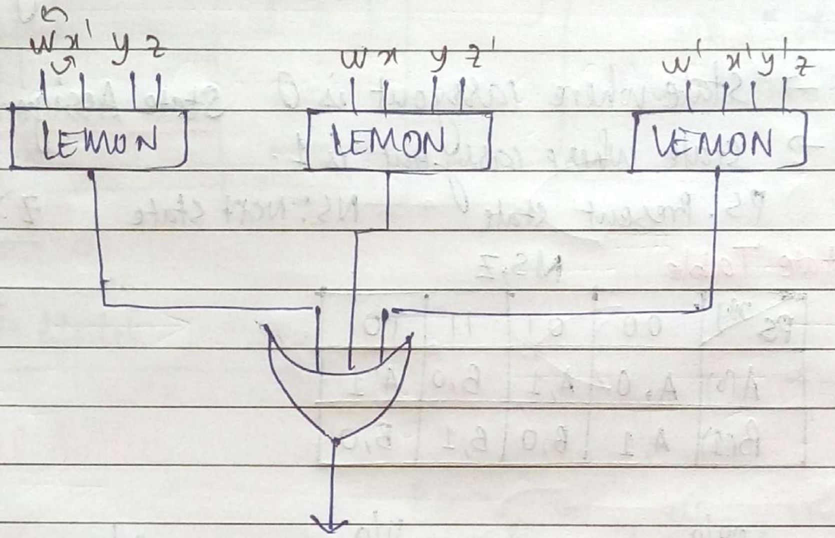
(a) Show a realitation of $f(W,X,Y,Z) = \Sigma(0,1,6,9,10,11,14,15)$

(b) Can LEMON & OR become universal.

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

yz \ wx	00	01	11	10
00	1	0	0	0
01	1	0	0	1
11	0	0	1	1
10	0	1	1	1

① LEMON($wx'yz$)
② LEMON($wxyz'$)
③ LEMON($w'x'y'z$)



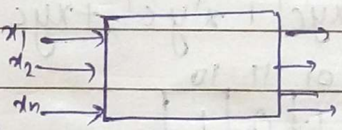
Synchronous Sequential Circuits

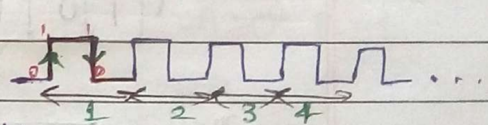
01/03/17

Combination

$O = f(I)$

Present output depends only on present input.



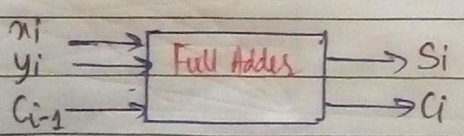


Sequential – Clock maintains the timings

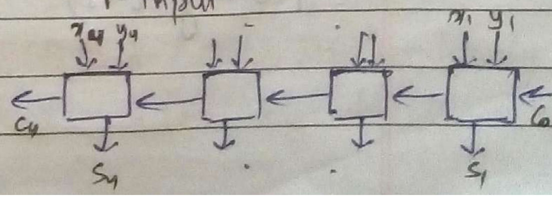
Next output depends on present input and the present state.

$Z(t) = \lambda(S(t-1), I(t-1))$ – Mealy Machine

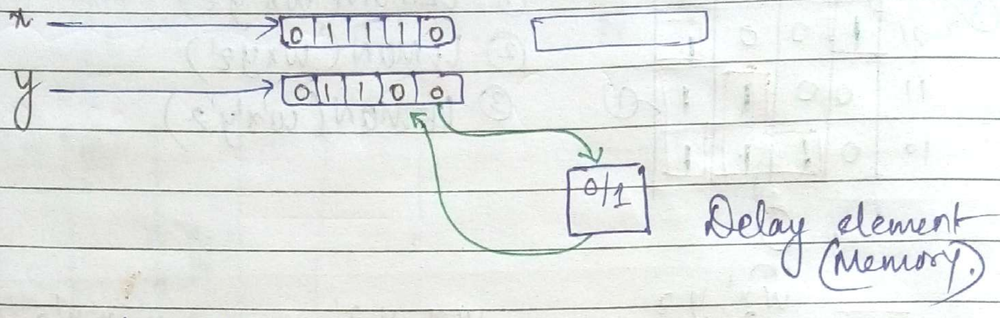
$Z(t) = \lambda(S(t-1))$ – Moore Machine



4-input



Serial Adder



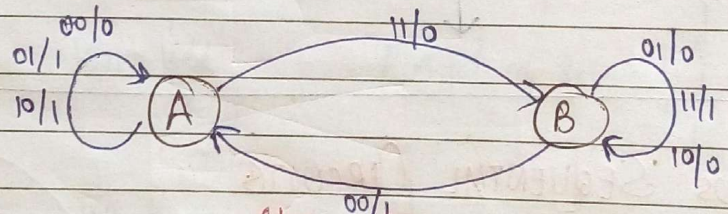
A → State where carryout is 0
B → State where carryout is 1.

State Assignment

PS: Present state   NS: Next state   Z: Output

State Table   NS, Z

PS \ xy	00	01	11	10
A(0)	A, 0	A, 1	B, 0	A, 1
B(1)	A, 1	B, 0	B, 1	B, 0

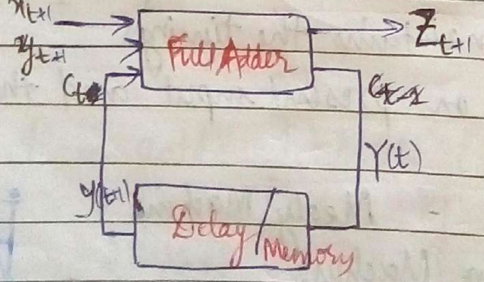


State Diagram

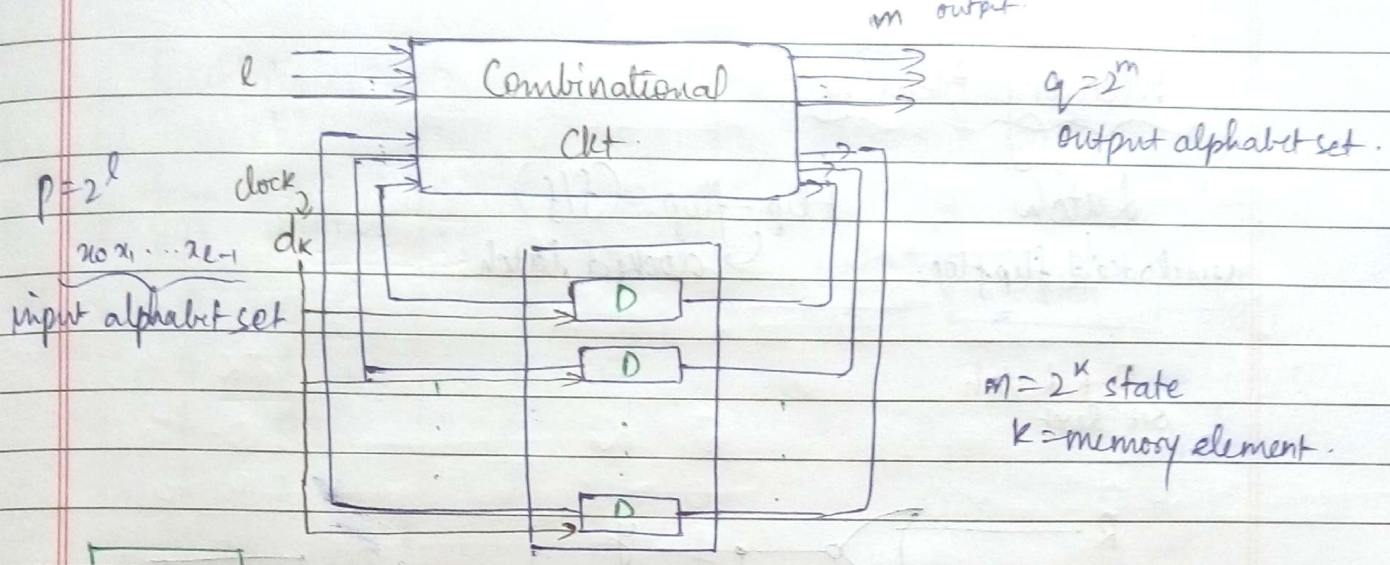
$Z = x'y'c + xy'c' + x'yc' + xyc$.

c \ xy	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$C = xy + xc + yc$



$Y(t) = y(t+1)$



FSM

Finite State Model

Serial Adder

02/03/17

PS \ x_1x_2	00	01	11	10
A (0)	A, 0	A, 1	B, 0	A, 1
B (1)	A, 1	B, 0	B, 1	B, 0

A: state with carry 0

B: state with carry 1

Z

PS = y \ x_1x_2		00	01	11	10
A	0	0	1	0	1
B	1	1	0	1	0

$$Z = x_1'x_2y' + x_1x_2'y' + x_1'x_2'y + x_1x_2y$$

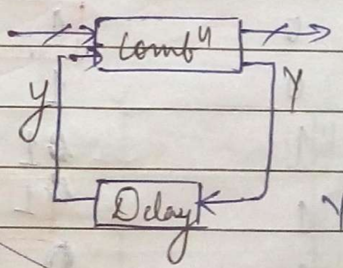
NS

PS = y \ x_1x_2		00	01	11	10
A	0	0	0	1	0
B	1	0	1	1	1

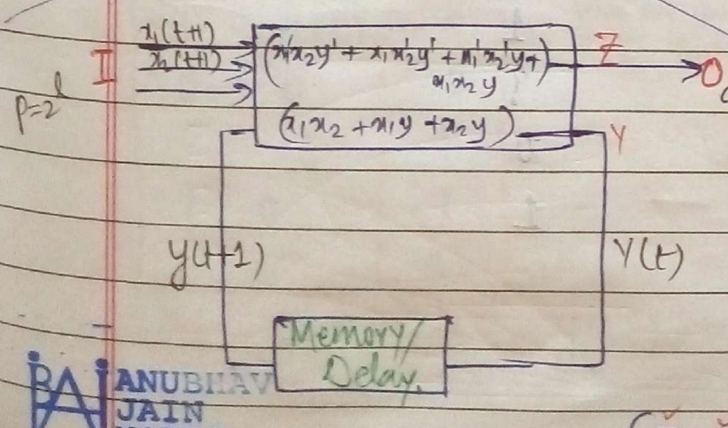
$$NS - Y = x_1x_2 + x_1y + x_2y$$

State Assignment

A = 0, B = 1.



$Y(t) = y(t+1)$



$$S(t+1) = \delta(S(t), I(t))$$

$$Z(t) = \lambda(S(t), I(t)) \text{ - Mealy}$$

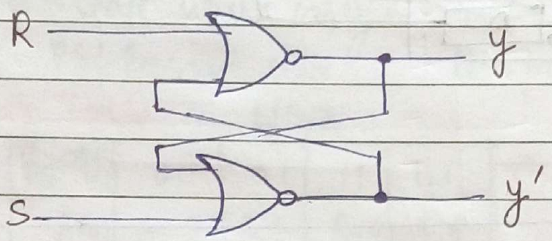
$$Z(t) = \lambda(S(t)) \text{ - Moore}$$

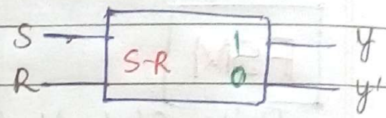
$$(I, O, \delta, \lambda, S)$$

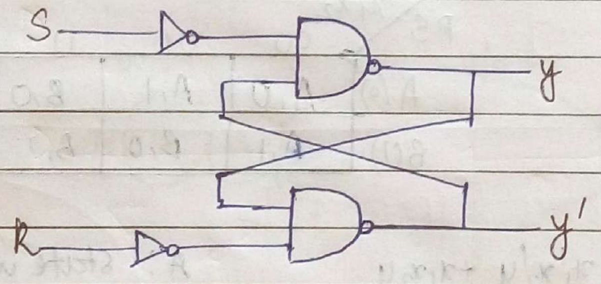
Memory Element

Latch , Flip-flop (f/f)
unclocked flip-flop. ↳ clocked latch.

S-R Latch
Set Reset





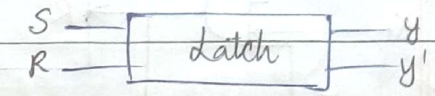


y(t)	S(t)	R(t)	y(t+1)
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

(Not allowed state) ↳ y = y' = 0.

Excitation Table

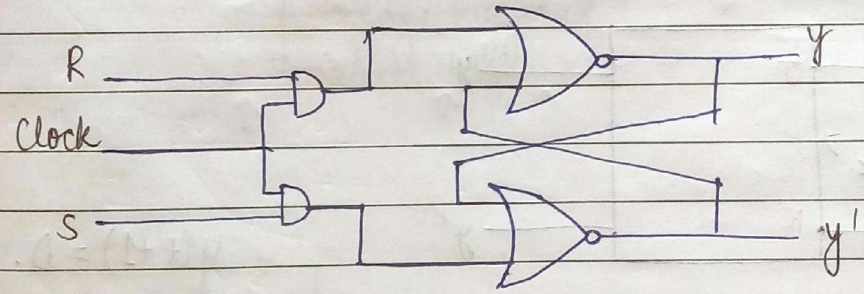
Transition		Input	
from $y(t)$	to $y(t+1)$	S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

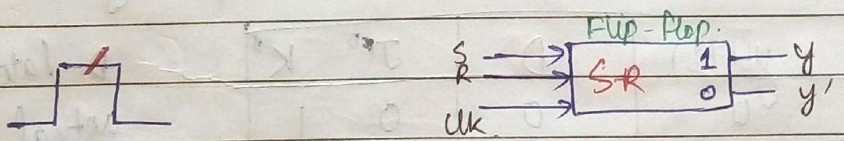


$y(t)$ \ SR	00	01	11	10
0	0	0	x	1
1	1	0	x	1

$y(t+1)$

$$y(t+1) = S + R'y(t)$$



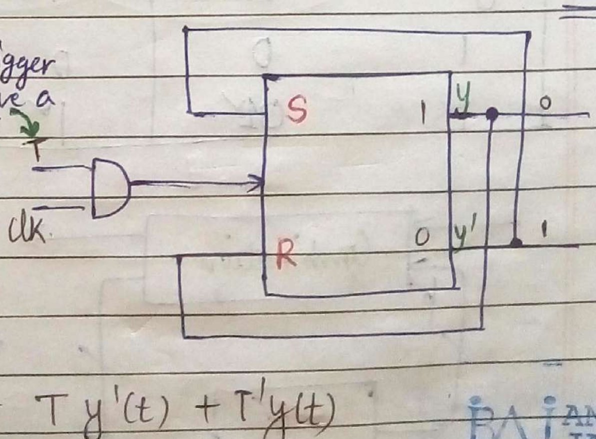


03/03/17

Toggle f-f

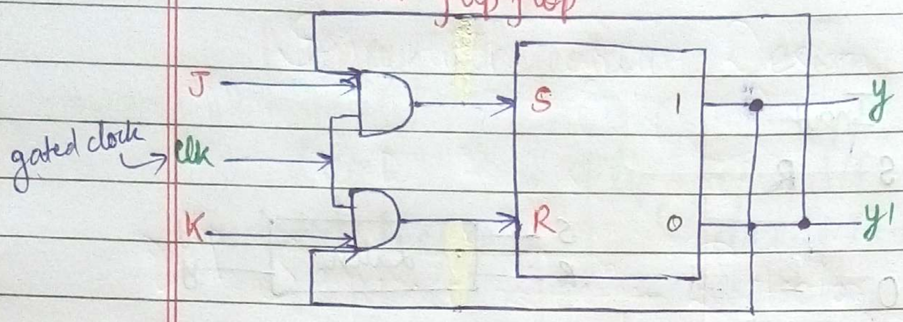
$y(t)$	$y(t+1)$	T
0	0	0
0	1 ✓	1
1	0	1
1	1 ✓	0

it will trigger when we give a input to T



$$y(t+1) = Ty'(t) + T'y(t)$$

J-K flip-flop



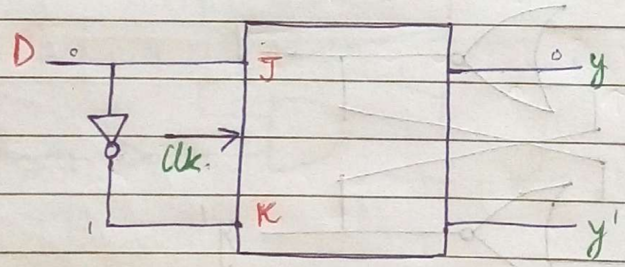
y(t)	y(t+1)	J	K
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

JK = 11

0 → 1

1 → 0

D flip-flop

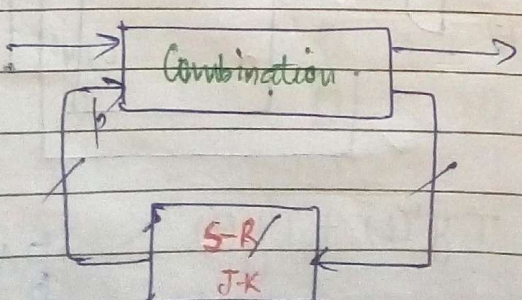


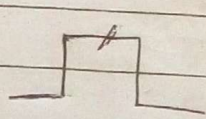
$y(t+1) = D$.

y(t)	y(t+1)	D	$J^{=D}$	$K^{=\bar{D}}$
0	0	0	0	1
0	1✓	1✓	1	0
1	0	0	0	1
1	1✓	1✓	1	0

D latch is actual memory element.

T = Toggle the prev. state (o/p).

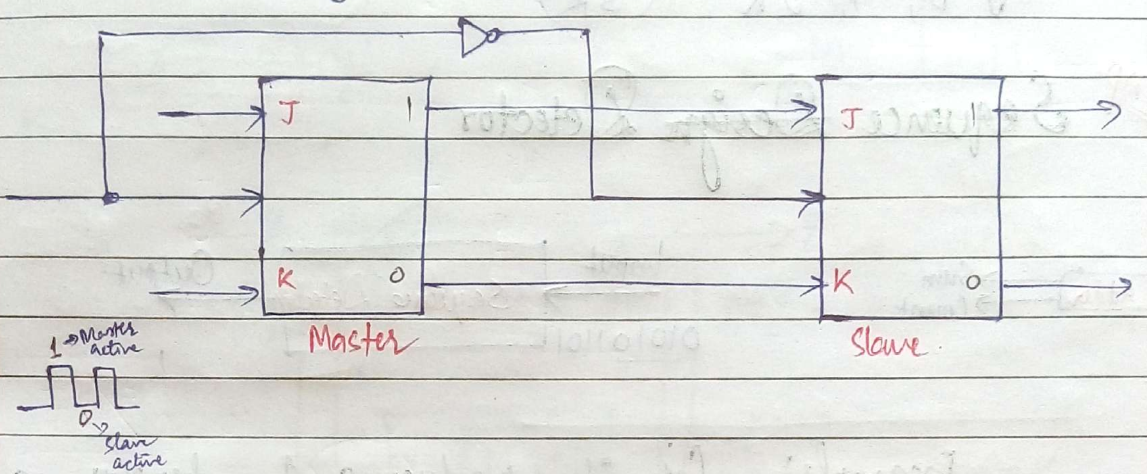




* clock should be long enough so that memory element should store the value (latch value).

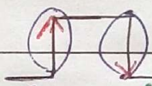
Propagation through the combinational logic.

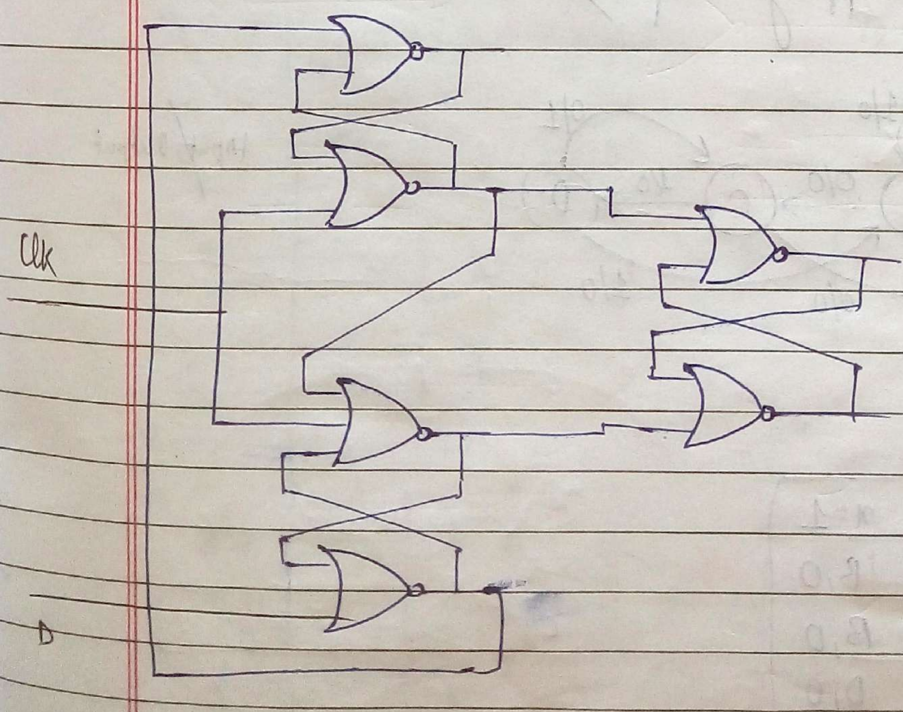
* clock should be short enough so that it is only stable. Propagation through the combinational logic.



Synchronous Sequential Ckt

synchronized with clock
- → edge triggered
- → level triggered





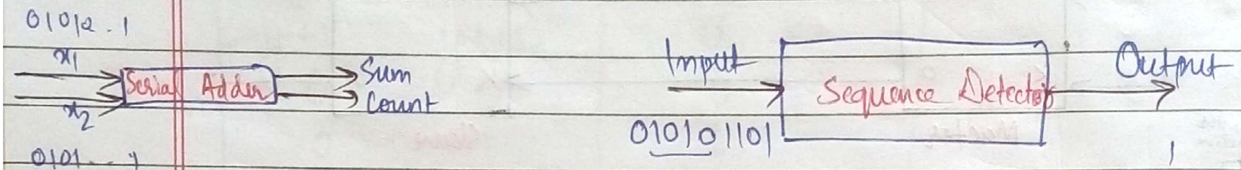
Negative edge triggered D f/f

PAGE:
DATE: 08/03/17

Sequential Circuit Design

Memory elements (Flip-flop)
D, T, JK (SR)

Imp. Sequence ~~Design~~ Detector



Example: Circuit produces a 1 whenever a sequence 1010 is detected.

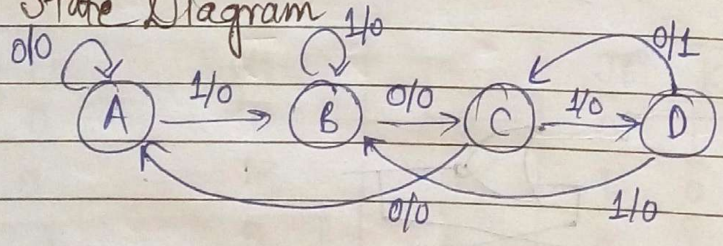
0 1 0 1 0 1 0 0 1 0 1 0

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

Overlapping input

(Consider the overlapping input.)

* State Diagram



Input/Output

* State Table

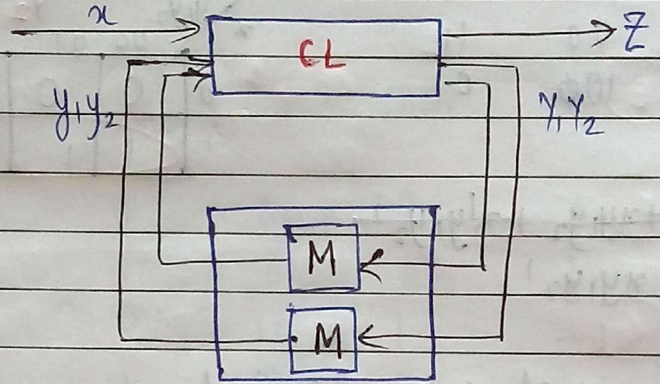
State Assignment $y_1 y_2$	Input / PS	NS, Z: $x=0$	NS, Z: $x=1$
00	A	A,0	B,0
01	B	C,0	B,0
11	C	A,0	D,0
10	D	C,1	B,0

Z

$x \backslash y_1y_2$	00	01	11	10
0	0	0	0	1
1	0	0	0	0

$Z = x'y_1y_2'$

	NS Y_1Y_2	Y_1Y_2
y_1y_2	$x=0$	$x=1$
00	00	01
01	11	01
11	00	10
10	11	01



Y_1

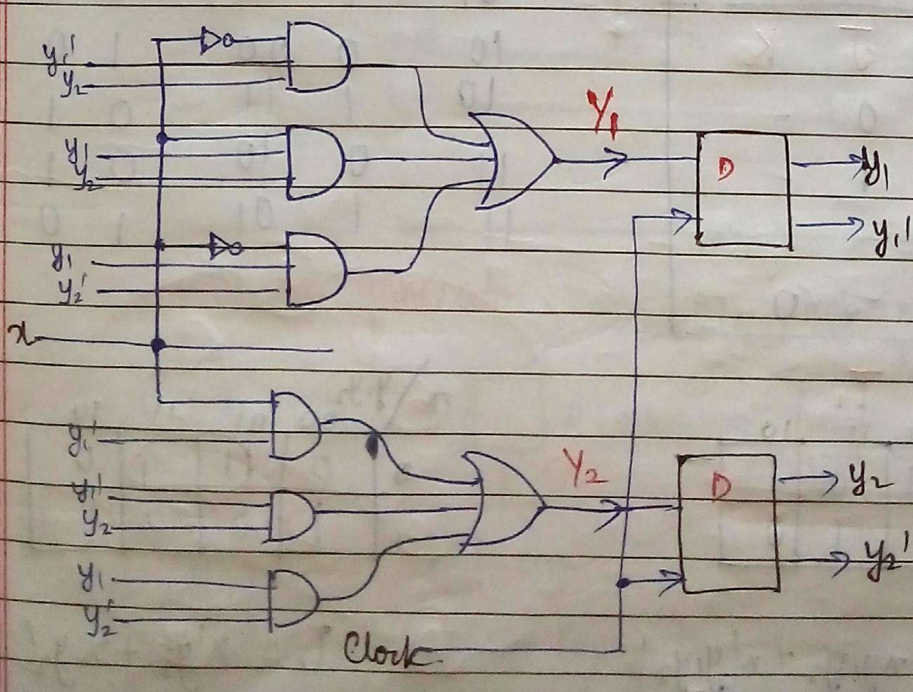
$x \backslash y_1y_2$	00	01	11	10
0	0	1	0	1
1	0	0	1	0

Y_2

$x \backslash y_1y_2$	00	01	11	10
0	0	1	0	1
1	1	1	0	1

$Y_1 = x'y_1'y_2 + xy_1y_2 + x'y_1y_2'$

$Y_2 = xy_1' + y_1'y_2 + y_1y_2'$



PAGE:
DATE: / /

Change State Assignments :-

A - 00 B - 01 C - 10 D - 11

$y_1 y_2$	NS $Y_1 Y_2$ $x=0$	$Y_1 Y_2$ $x=1$
00	00	01
01	10	01
10	00	11
11	10	01

Y_1

$x \backslash y_1y_2$	00	01	11	10
0	0	1	01	0
1	0	0	0	01

Y_2

$x \backslash y_1y_2$	00	01	11	10
0	0	0	0	0
1	1	1	1	1

~~$Y_1 = x'y_1y_2' + xy_1y_2 + x'y_1y_2'$~~

$Y_1 = x'y_2 + xy_1y_2'$

$Y_2 = x$ → Less no. of gates required ↳ More efficient design.

y(t) y(t+1)	T
00	0
01	1
10	1
11	0

y(t) y(t+1)	J	K
00	0	–
01	1	–
10	–	1
11	–	0

PS	x	NS $T_1 T_2$	T_1	T_2
00	0	00	0	0
00	1	01	0	1
01	0	10	1	1
01	1	01	0	0
10	0	00	1	0
10	1	11	0	1
11	0	10	0	1
11	1	01	1	0

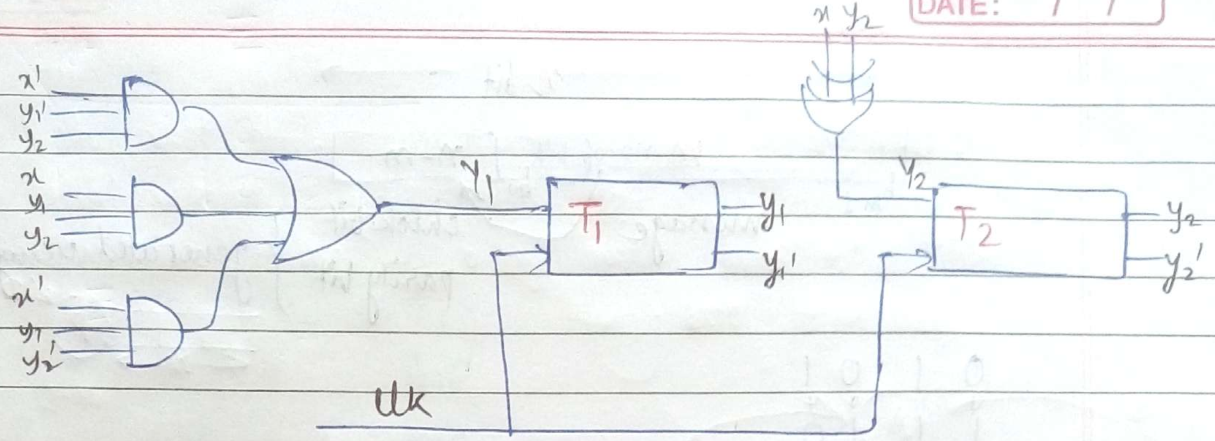
T_1

$x \backslash y_1y_2$	00	01	11	10
0	0	01	0	1
1	0	0	1	0

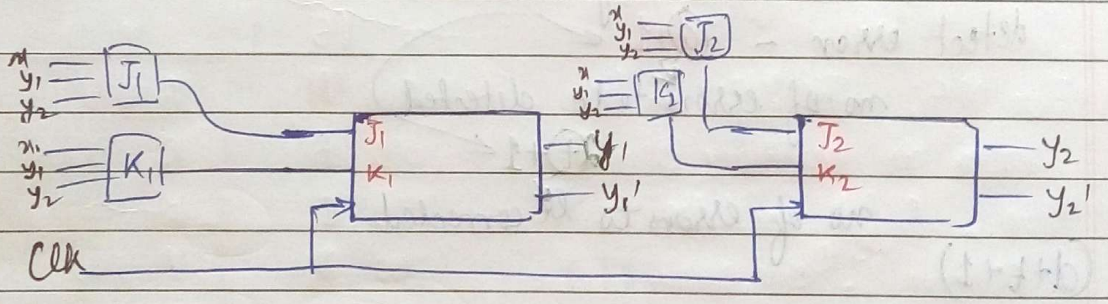
$x \backslash y_1y_2$	00	01	11	10
0	0	1	1	0
1	1	0	0	1

$T_1 = x'y_1'y_2 + xy_1y_2 + x'y_1y_2'$

$T_2 = x'y_2 + xy_2'$



PS $y_1 y_2$	Input x	NS	Excitation J_1K_1	J_2K_2
00	0	00	0 –	0 –
00	1	01	0 –	1 –
01	0	10	1 –	– 1
01	1	01	0 –	– 0
10	0	00	– 1	0 –
10	1	11	– 0	1 –
11	0	10	– 0	– 1
11	1	01	– 1	– 0



Parity Bit generator.

09/03/17.

Sender 0 1 0 1 1 0 . . . [ ] 0 1 0 1 . . . [ ] . . . Receiver

m p p

$p_2\ p_3\ p_1\ p_0$ (instead of 1 only, more no. of parity bits can be used)

p: odd even

even parity

– The no. of 1's in m bits are odd. ⇒ Parity is 1 ∴ so that together there are even no. of ones, else put 0.

PAGE:
DATE: / /

← n bit →

| b_{m-1} ... m no. of bit ... b_0 | n-m |

message → check-bit / parity bit } generated using m.

0 1 0 1
↓ ↓ ↓ ↓
1 1 1 0

1   1 1

Hamming distance → 3

Hamming weight = no. of 1's.

2^4 { 0 0 0 0 / 0 0 0 1 / ⋮ / 1 1 1 1 }

$2^5 = 32$

distance between any pair ≥ 2

$d+1$

1 (det)

Hamming distance

detect error – $d+1$

no. of errors to be detected.

$2t+1$

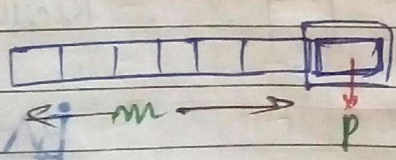
no. of errors to be corrected.

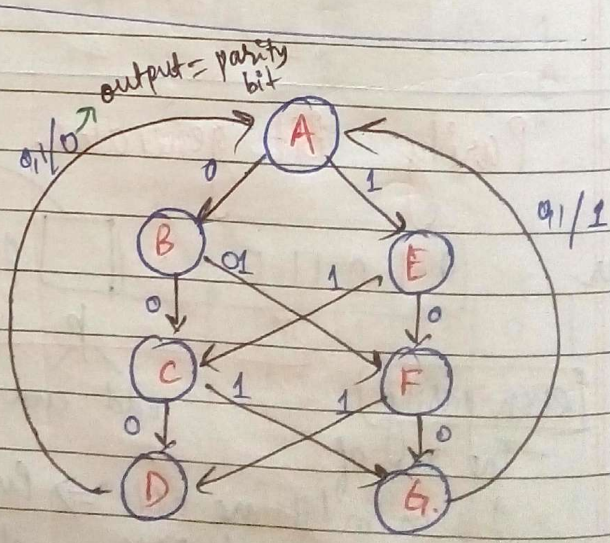
$(d+t+1)$

State Diagram

Even Parity bit generator

$M = 3$





NS, Z

State Assign. $y_1 y_2 y_3$	PS	x=0	x=1
0 000	A	B, X	E, X
2 010	B	C, X	F, X
3 011	C	D, X	G, X
6 110	D	A, 0	A, 0
7 111	E	F, X	C, X
4 100	F	G, X	D, X
5 101	G	A, 1	A, 1

Z

y_1	y_2	y_3	x=0	x=1
0	0	0	X	X
0	1	0	X	X
0	1	1	X	X
1	1	0	0	0
1	1	1	X	X
1	0	0	X	X
1	0	1	1	1

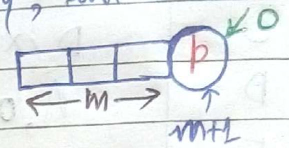
y_1y_2 \ y_3x	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	X	X
10	X	X	1	1

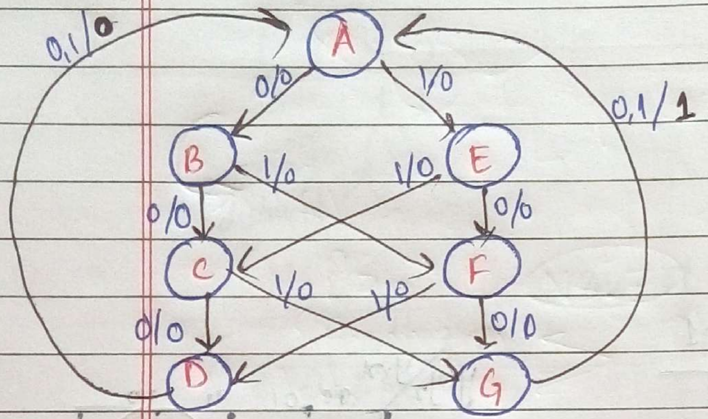
$Z = y_3.$

NS

	y_1	y_2	y_3	x=0	x=1
A	0	0	0	B 010	E 111
B	0	1	0	C 011	F 100
C	0	1	1	D 110	G 101
D	1	1	0	A 000	A 000
E	1	1	1	F 100	C 011
F	1	0	0	G 101	D 110
G	1	0	1	A 000	A 000

Yesterday, for intermediate output, we took don't care. But conventionally, for even parity, intermediate bits are set to be zero and for odd parity, intermediate bits are kept one.





$y_3y_2y_1$ State Assignment	PS	NS, Z: $x=0$	NS, Z: $x=1$
000	A	B, 0	E, 0
010	B	C, 0	F, 0
110	C	D, 0	G, 0
100	D	A, 0	A, 0
011	E	F, 0	C, 0
111	F	G, 0	D, 0
101	G	A, 1	A, 1

Clearly, $z = y_1 y_2' y_3$

For NS, JK f/f can be chosen as memory element. Here, 3 f/f will be required, one each for y_1, y_2 & y_3.

PS		NS: $x=0$	NS: $x=1$
000	A	B, 010	E, 011
010	B	C, 110	F, 111
110	C	D, 100	G, 101
100	D	A, 000	A, 000
011	E	F, 111	C, 110
111	F	G, 101	D, 100
101	G	A, 000	A, 000

	$x=0$: J_3K_3	$x=0$: J_2K_2	$x=0$: J_1K_1	$x=1$: J_3K_3	$x=1$: J_2K_2	$x=1$: J_1K_1
A	0–	1–	0–	0–	1–	1–
B	1–	–0	0–	1–	–0	1–
C	–0	–1	0–	–0	–1	1–
D	–1	0–	0–	–1	0–	0–
E	1–	–0	–0	1–	–0	–1
F	–0	–1	–0	–0	–1	–1
G	–1	0–	–1	–1	0–	–1

(we have an extra state → 001 taking it as don't case)

J_1

$y_3y_2 \backslash y_1x$	00	01	11	10
00	0	1	X	X
01	0	1	X	X
11	0	1	X	X
10	0	0	X	X

K_1

$y_3y_2 \backslash y_1x$	00	01	11	10
00	X	X	X	X
01	X	X	1	0
11	X	X	1	0
10	X	X	1	1

$J_1 = y_3'x + y_2x$ $\qquad$ $K_1 = x + y_2'$

J_2

$y_3y_2 \backslash y_1x$	00	01	11	10
00	1	1	X	X
01	X	X	X	X
11	X	X	X	X
10	0	0	0	0

K_2

$y_3y_2 \backslash y_1x$	00	01	11	10
00	X	X	X	X
01	0	0	0	0
11	1	1	1	1
10	X	X	X	X

$J_2 = y_3'$ $\qquad$ $K_2 = y_3$

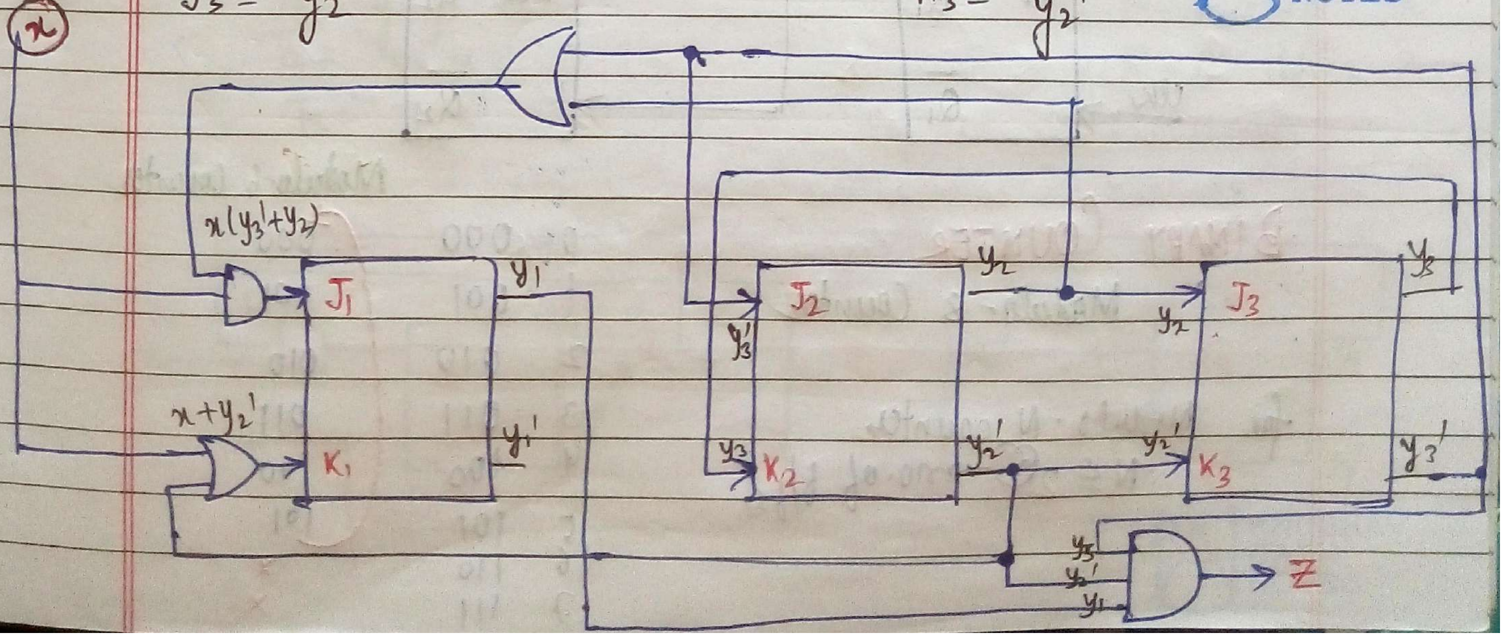
J_3

$y_3y_2 \backslash y_1x$	00	01	11	10
00	0	0	X	X
01	1	1	1	1
11	X	X	X	X
10	X	X	X	X

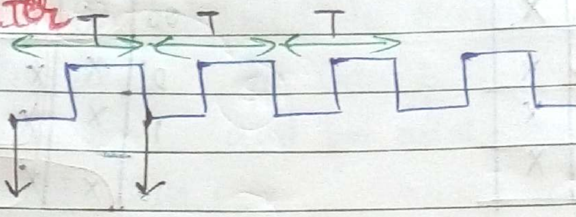
K_3

$y_3y_2 \backslash y_1x$	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	0	0
10	1	1	1	1

$J_3 = y_2$ $\qquad$ $K_3 = y_2'$



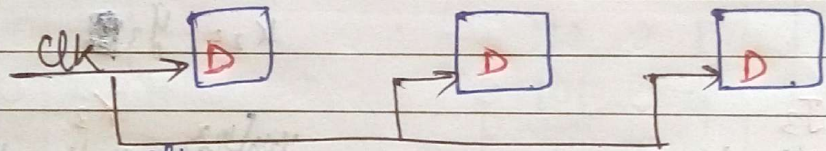
Counter



Combination of f/fs and the no. of state (of f/f) changes gives the count of clock pulse.

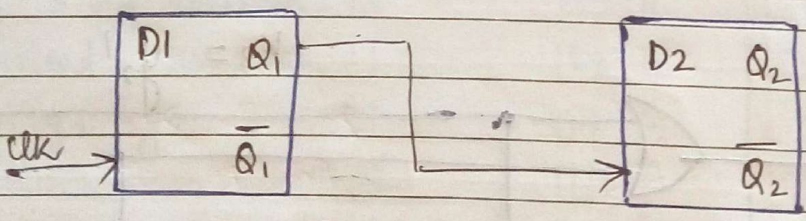
→ State of the f/f s will directly give you the count value.

Synchronous ⟷ Counters
Asynchronous



All f/fs are triggered with same clock / time

clk	
0	000
1	001
⋮	⋮
7	111



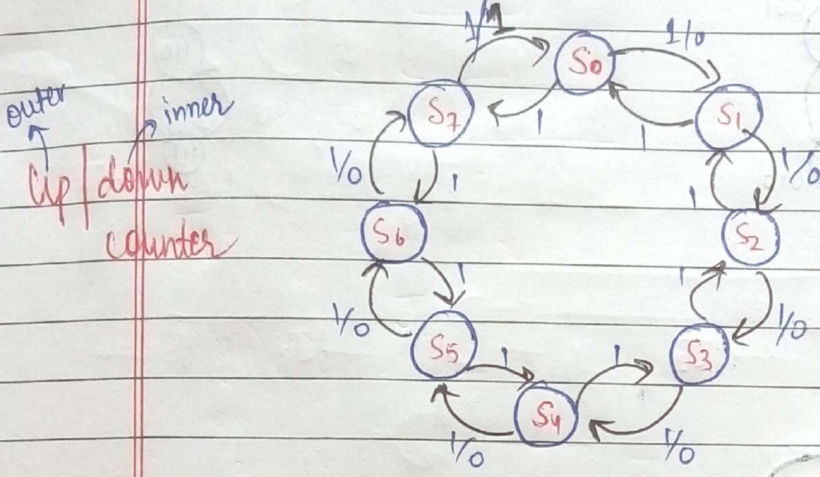
BINARY COUNTER

Modulo-8 Counter

	Modulo-8	Modulo-6 counter
0	000	000
1	001	001
2	010	010
3	011	011
4	100	100
5	101	101
6	110	×
7	111	×

for Modulo-N counter
$N \leq 2^{n}$ → n = no. of f/fs.

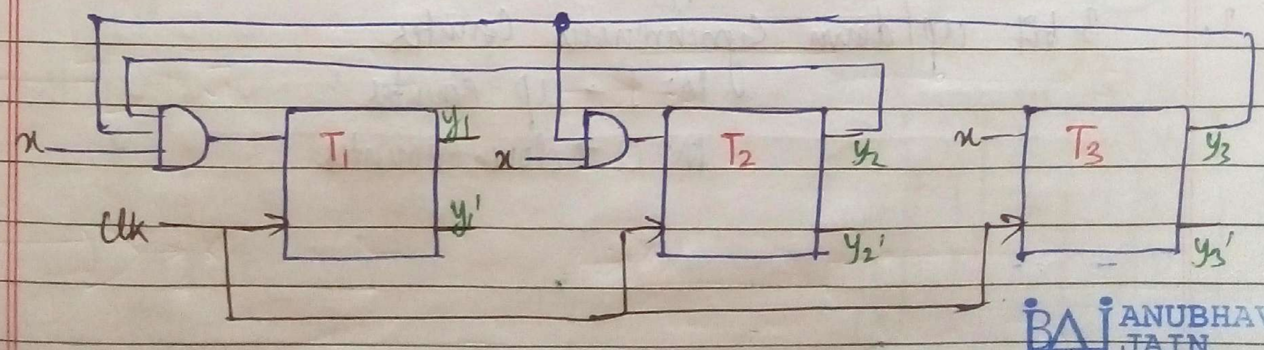
State Diagram



PS $y_1\, y_2\, y_3$	NS, Z	T (T_1 T_2 T_3)
000	001, 0	0 0 1
001	010, 0	0 1 1
010	011, 0	0 0 1
011	100, 0	1 1 1
100	101, 0	0 0 1
101	110, 0	0 1 1
110	111, 0	0 0 1
111	000, 1	1 1 1

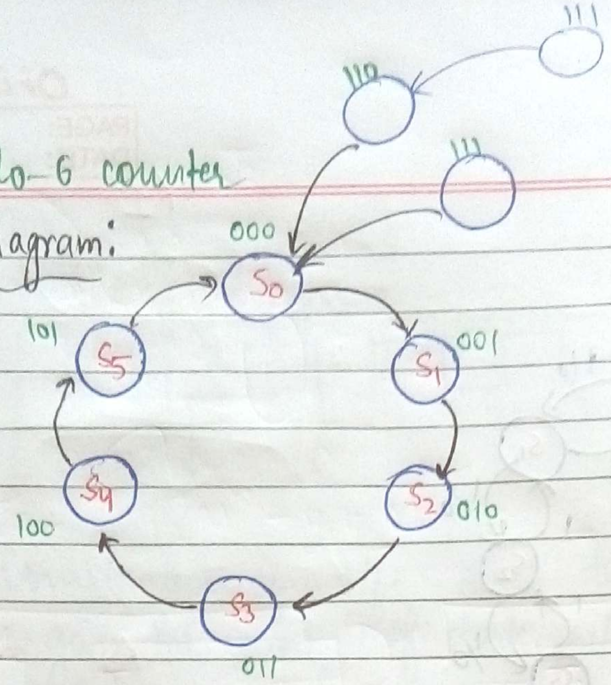
$T_1 = y_2 y_3 \cdot x$
$T_2 = y_3 \cdot x$
$T_3 = 1 \cdot x$

M = 1	up	x
M = 0	down	$\bar{x}$

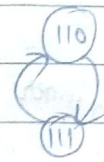


Modulo-6 counter

State Diagram:



110
111



* Lock-out state.

PS $y_1\ y_2\ y_3$	NS, Z M=1 (up)	M=0 (down)
0 0 0	001, 0	101, 0
0 0 1	010, 0	000, 1
0 1 0	011, 0	001, 0
0 1 1	100, 0	010, 0
1 0 0	101, 0	011, 0
1 0 1	000, 1	100, 0
1 1 0	000, 0	000, 0
1 1 1	000, 0	000, 0

J-K.

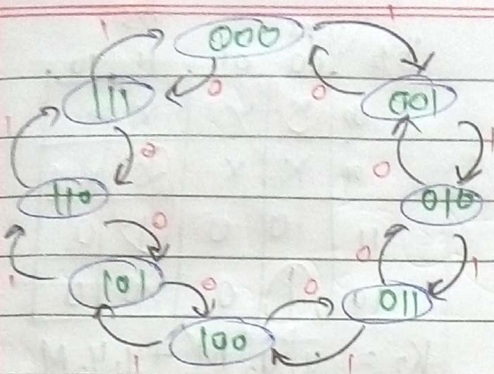
15/03/17.

Synchronous Counter

1. Modulo-8 counter (up)
2. Modulo-6 counter (up/down)
3. 3-bit up/down synchronous counter

M=1 → up counter

M=0 → down counter



PS			Mode	NS.	NS-implementation		
y_3	y_2	y_1	M		J_3K_3	J_2K_2	J_1K_1
0	0	0	0	111	1 –	1 –	1 –
0	0	0	1	001	0 –	0 –	1 –
0	0	1	0	000	0 –	0 –	– 1
0	0	1	1	010	0 –	1 –	– 1
0	1	0	0	001	0 –	– 1	1 –
0	1	0	1	011	0 –	– 0	1 –
0	1	1	0	010	0 –	– 0	– 1
0	1	1	1	100	1 –	– 1	– 1
1	0	0	0	011	– 1	1 –	1 –
1	0	0	1	101	– 0	0 –	1 –
1	0	1	0	100	– 0	0 –	– 1
1	0	1	1	110	– 0	1 –	– 1
1	1	0	0	101	– 0	– 1	1 –
1	1	0	1	111	– 0	– 0	1 –
1	1	1	0	110	– 0	– 0	– 1
1	1	1	1	000	– 1	– 1	– 1

Clearly, $J_1 = 1$, $K_1 = 1$ (No need to draw K-map).

J_2

y_3y_2 \ y_1M	00	01	11	10
00	1	0	1	0
01	X	X	X	X
11	X	X	X	X
10	1	0	1	0

$J_2 = y_1'M' + y_1M$

K_2

y_3y_2 \ y_1M	00	01	11	10
00	X	X	X	X
01	1	0	1	0
11	X	0	1	0
10	X	X	X	X

$K_2 = y_1'M' + y_1M. = J_2.$

J_3

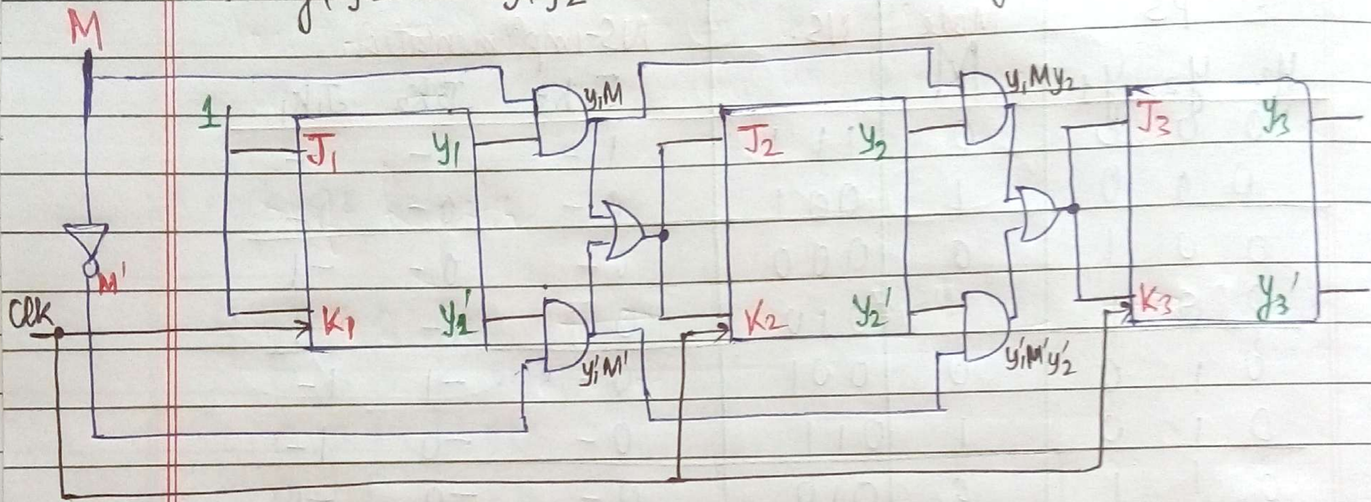
y_3y_2 \ y_1M	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	X	X	X	X
10	X	X	X	X

$J_3 = y_1'y_2'M' + y_1y_2M$

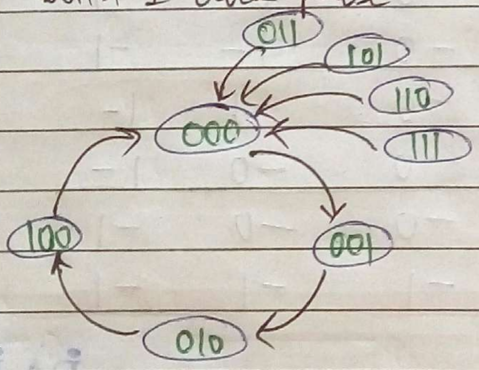
K_3

y_3y_2 \ y_1M	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	1	0	0	0

$K_3 = y_1'y_2'M' + y_1y_2M = J_3$



1. Design a synchronous ckt with D f/fs that goes through 0, 1, 2, 4, 0. All undesired states must go to 0-state with 1 clock pulse.



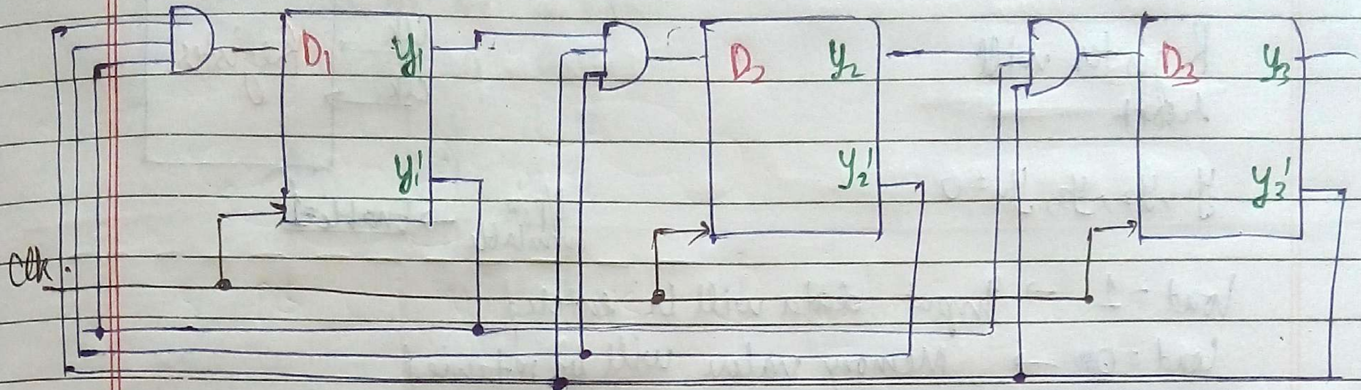
PS $y_3y_2y_1$	NS	D_3	D_2	D_1
000	001	0	0	1
001	010	0	1	0
010	100	1	0	0
011	000	0	0	0
100	000	0	0	0
101	000	0	0	0
110	000	0	0	0
111	000	0	0	0

(D-excitation: D_3, D_2, D_1)

$D_1 = y_1'y_2'y_3'$

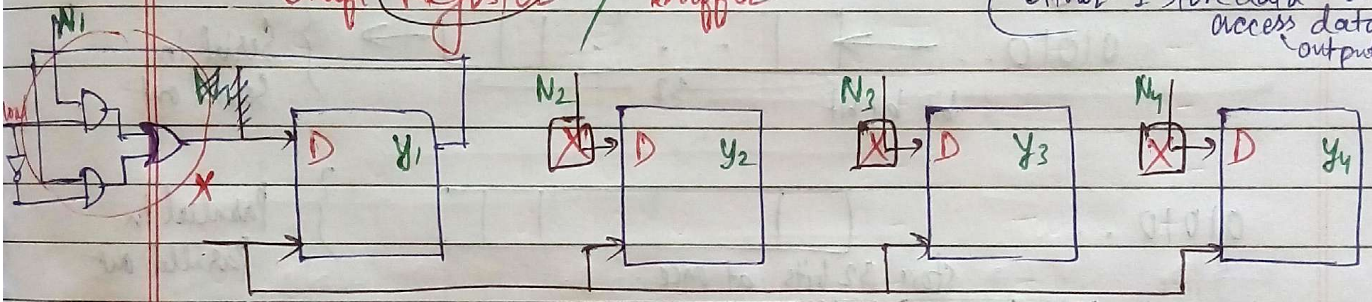
$D_2 = y_1y_2'y_3'$

$D_3 = y_1'y_2y_3'$

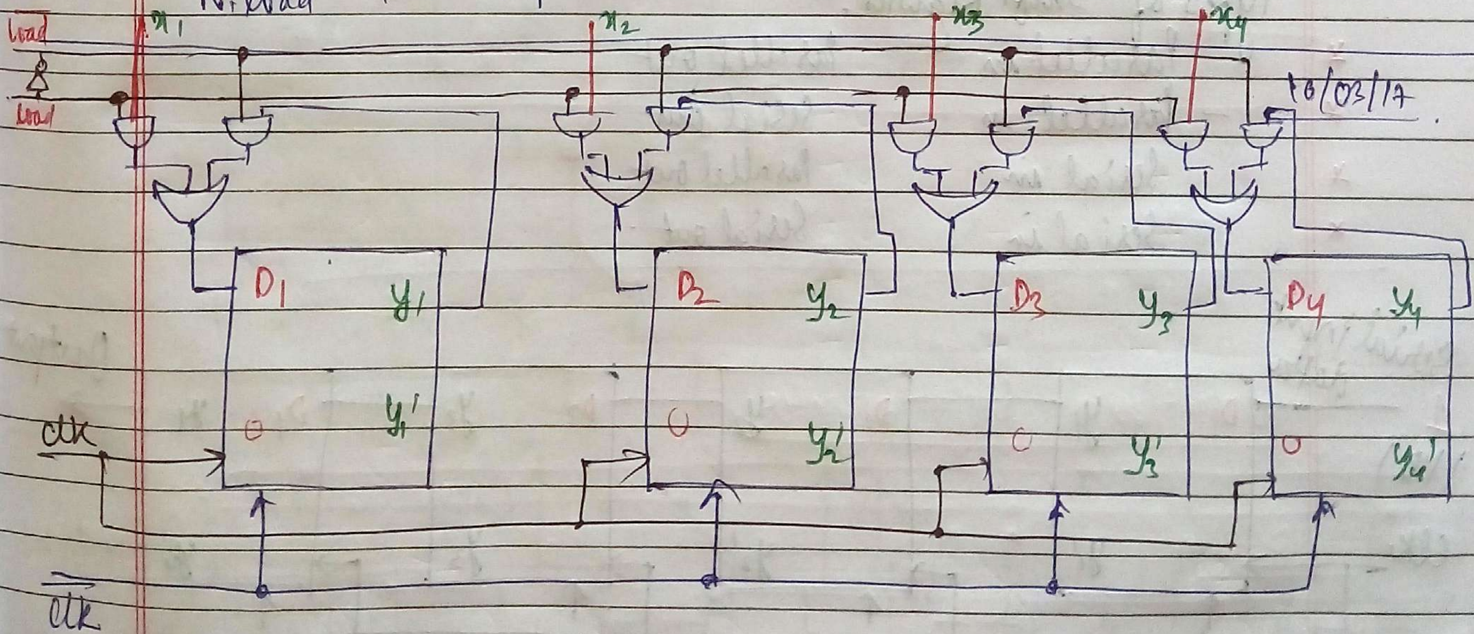


Shift Register / Buffer

(Either I store data or access data. output)



N. load + Y. $\overline{\text{Load}}$



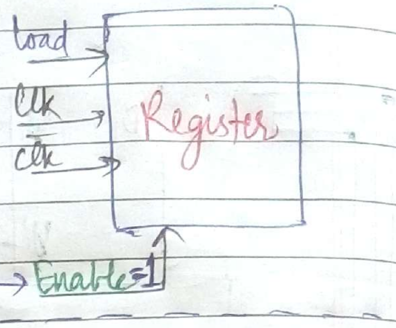
$$D_i = \boxed{x_i \cdot \text{Load} + y_i \cdot \overline{\text{Load}}}$$

4 bit Register (buffer)

if CLR = 0

Register will reset

$y_1, y_2, y_3, y_4 = 0$

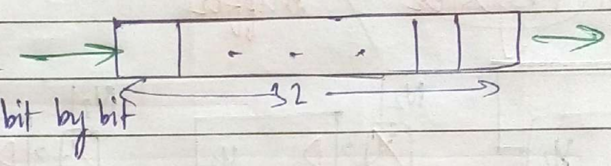


Load = 1 → Input data will be entered

Load = 0 → Memory value will be retained

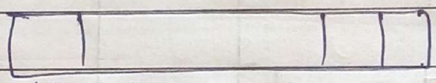
Shift Register

01010 . .



Serial in Serial out

01010 . . .



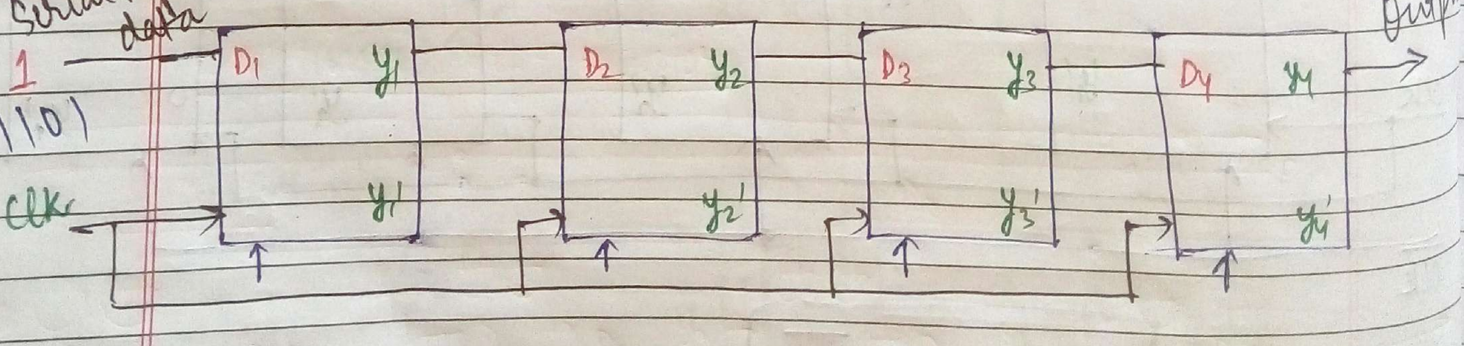
Parallel in Parallel out

store 32 bits at once.

32	32	32

Types of Shift Register :-

* Parallel in — Parallel out
* Parallel in — Serial out
* Serial in — Parallel out
* Serial in — Serial out.

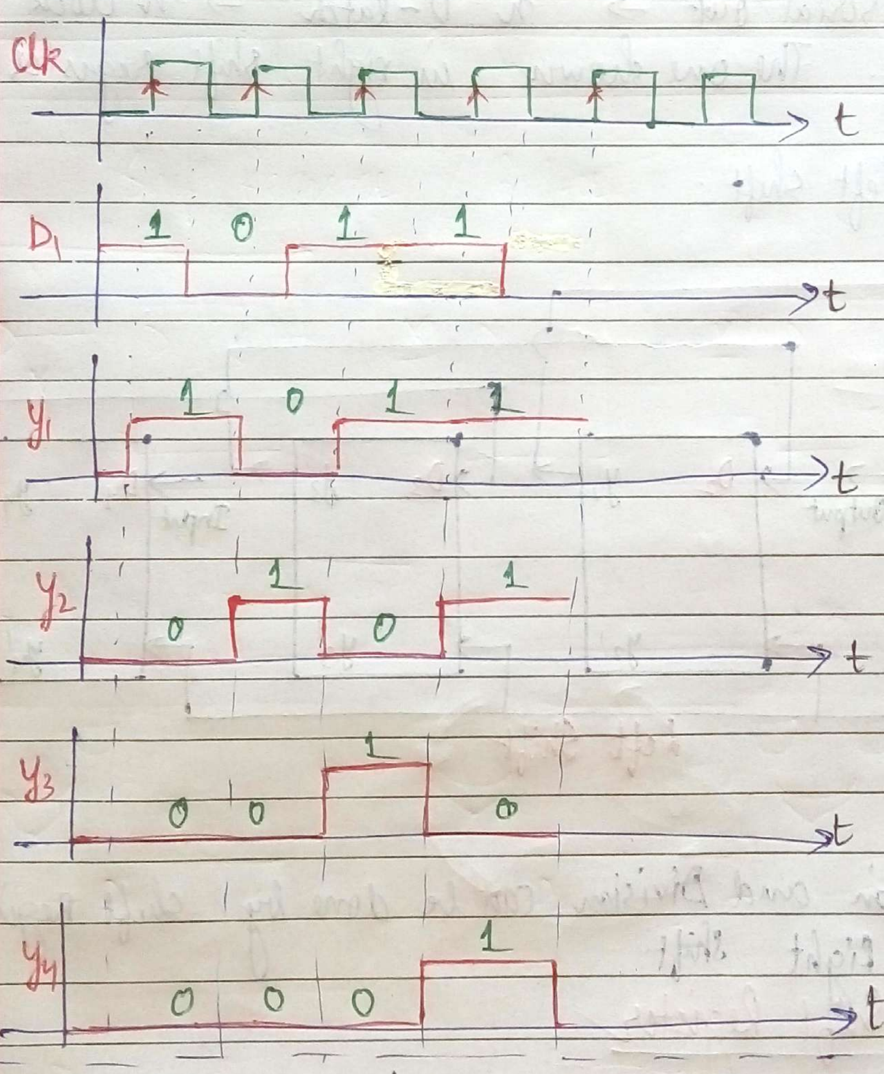


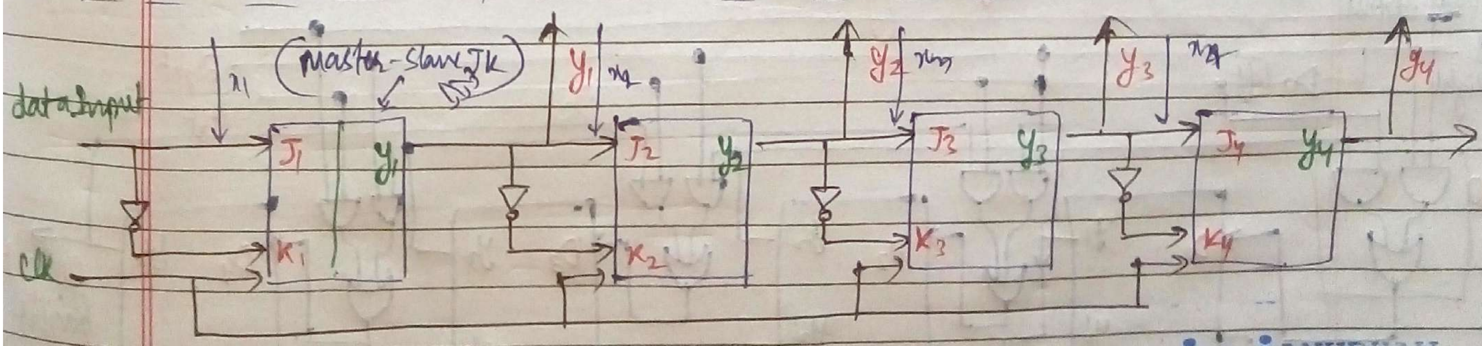
Serial in - Serial Out

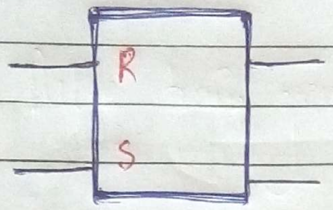
data-1101

data. (register value)

Clock.	D_1	D_2	D_3	D_4
0	0	0	0	0
1	1	0	0	0
2	0	1	0	0
3	1	0	1	0
4	1	1	0	1







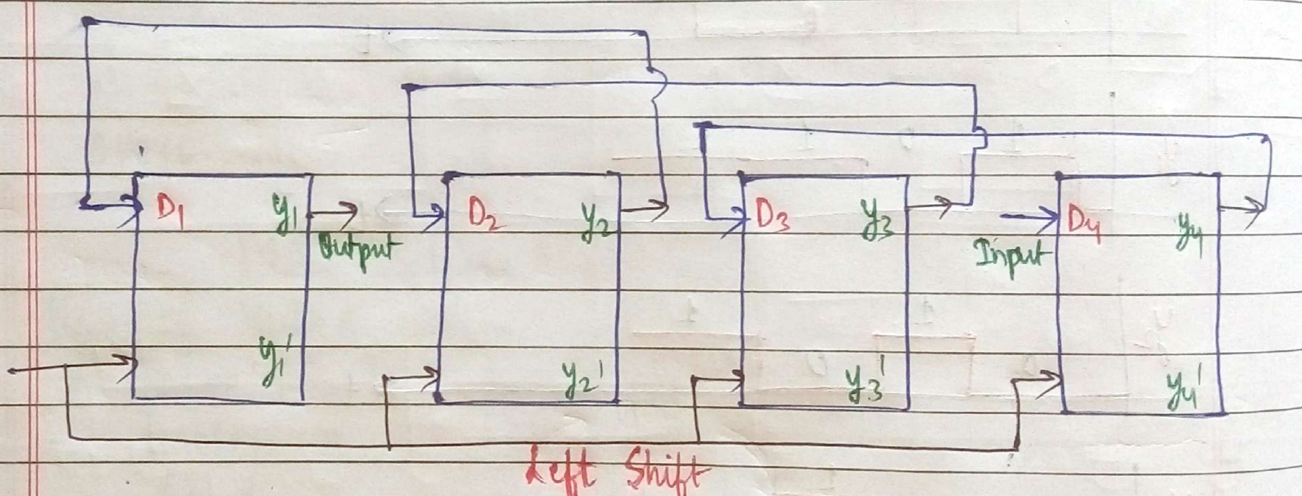
Serial in parallel out
parallel in parallel out

// Input will never appear

17/3/17.

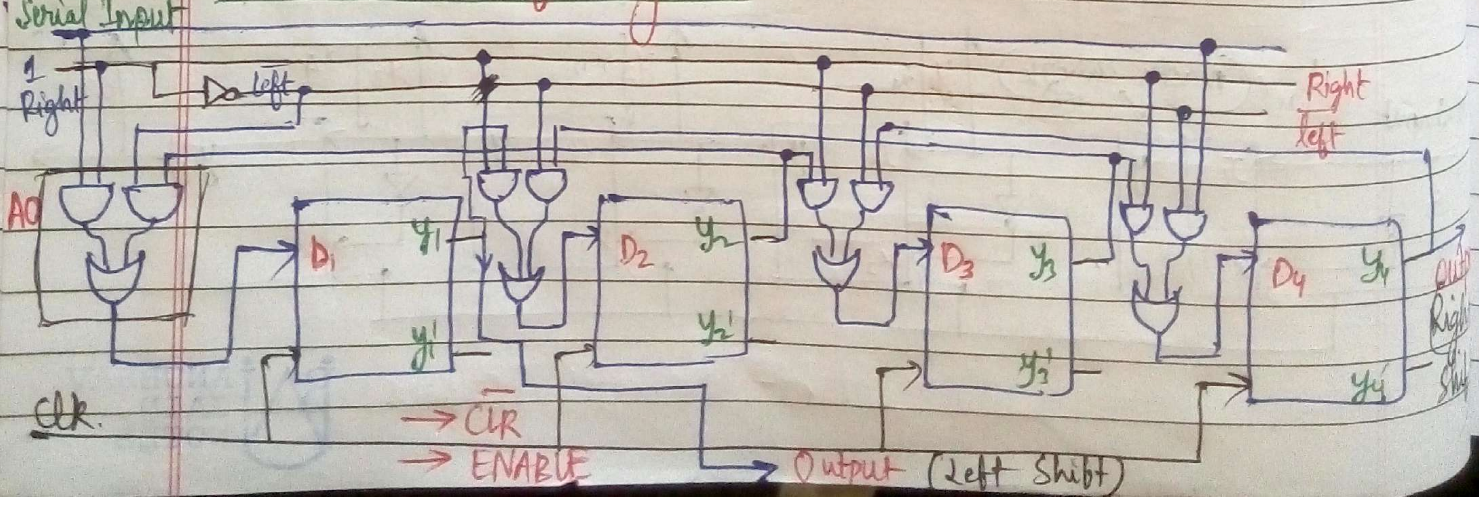
Serial in Serial out → n D-latch → n clock pulses required. The one shown is right shift register.

Make it left shift.



1010
Multiplication and Division can be done by Shift Register.
Left and Right shift.

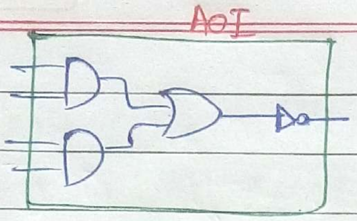
Birectional Shift Register



Orion

PAGE:
DATE: / /

AOI



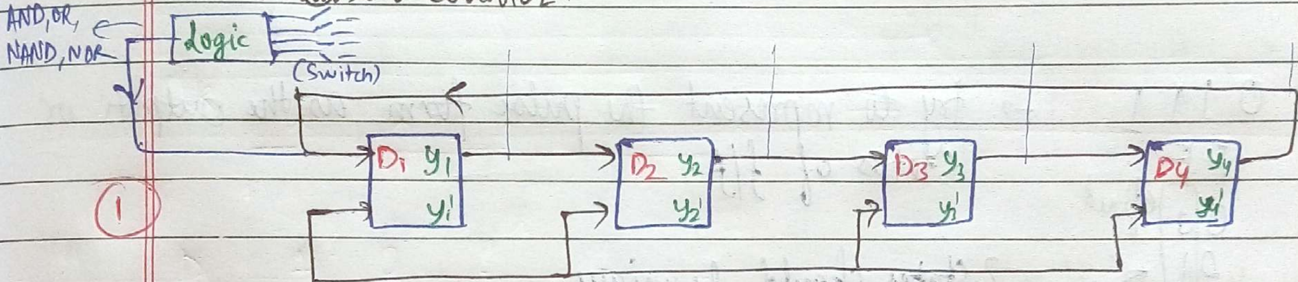
AOI And OR invertor

OAI Or and invertor.

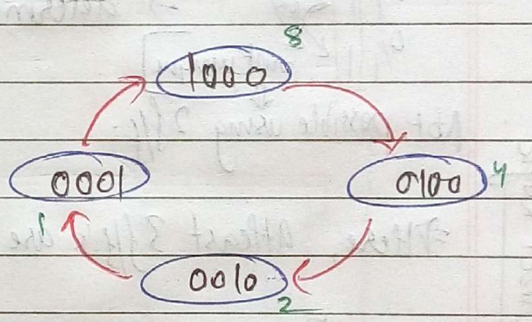
Can directly be used.

Counters

1. Ring counter
2. Johnson Counter

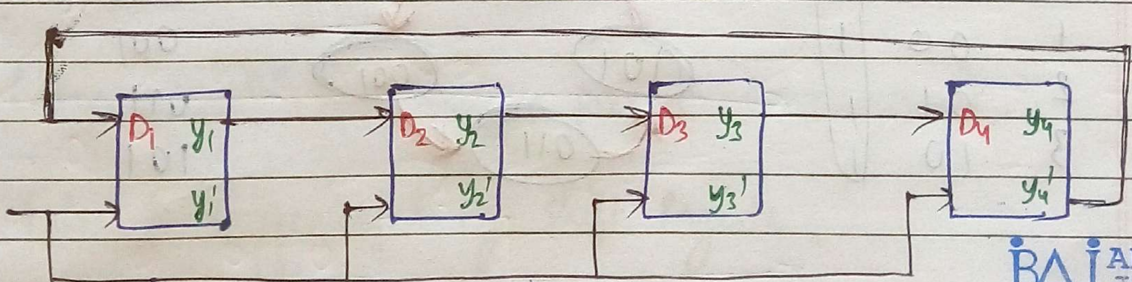


Clk	D_1	D_2	D_3	D_4
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1

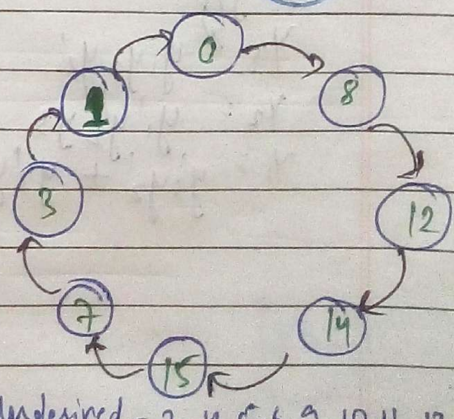


desired states → 1, 2, 4, 8

(3, 5, 6, 7) → undesired states.

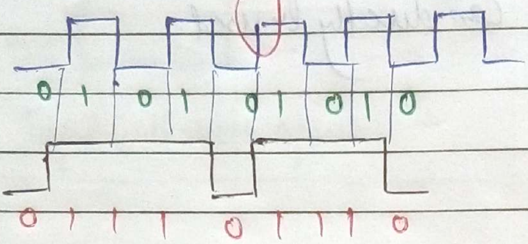


Clk.	D_1	D_2	D_3	D_4	
0	0	0	0	0	0
1	1	0	0	0	8
2	1	1	0	0	12
3	1	1	1	0	14
4	1	1	1	1	15
5	0	1	1	1	7
6	0	0	1	1	3
7	0	0	0	1	
8	0	0	0	0	6



Undesired - 2, 4, 5, 6, 9, 10, 11, 13.

Pulse- train generator



0 1 1 1 → try to represent the pulse form as the outputs or states of f/f.

FF	
0	0 ← fixed
0	1
1	1
0/1	1

→ States should be unique.

→ determine the no. of f/f's required.

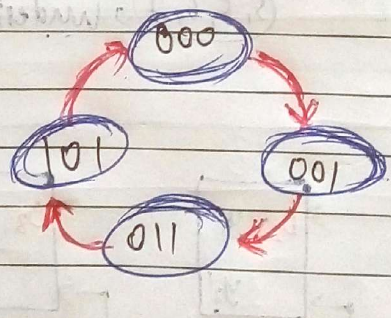
not unique

Not possible using 2 f/f.

Direct Logic

⇒ Here, atleast 3 f/f's are required.

Clk	F	F	F
0	0	0	0
1	0	0	1
2	0	1	1
3	1	0	1



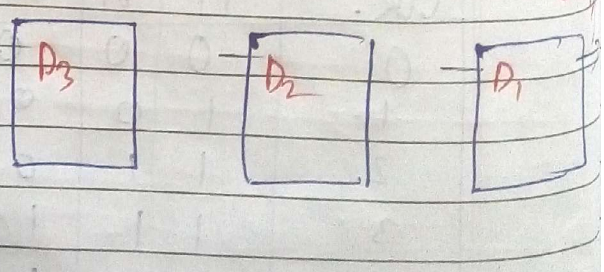
PS $y_3\ y_2\ y_1$	NS $D_3\ D_2\ D_1$
000	001
001	011
011	101
101	000

D f/f

$Y_3 = y_1 y_2 y_3'$

$Y_2 = y_1' y_2' y_3$

$Y_1 = y_3' y_2' + y_3' y_1$



Indirect logic

Use counter (Modulo)

$N=5 \Rightarrow n=3.$

$N < 2^n$

Modulo (N)- counter.

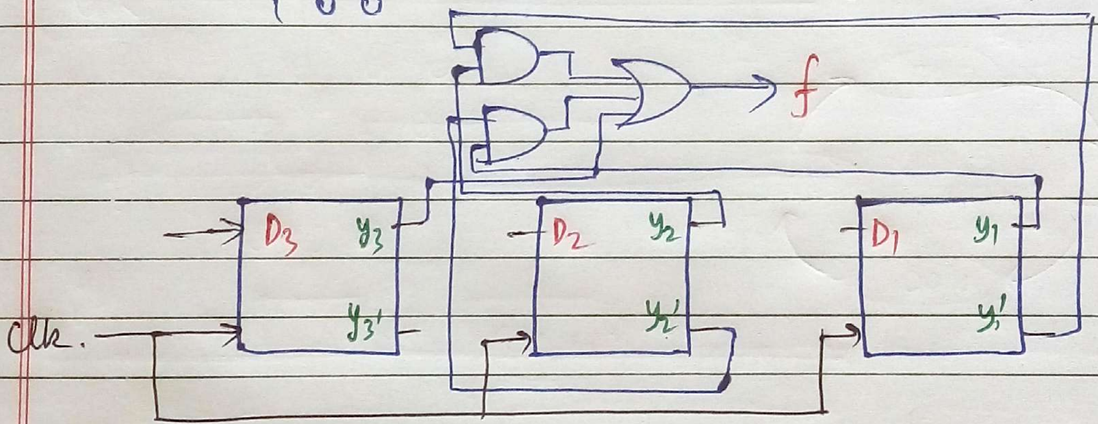
pulse train - 01101 01101

Modulo-5 Counter

y_3	y_2	y_1	f
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1

$y_3 \backslash y_2y_1$	00	01	11	10
0	0	1	0	1
1	1	X	X	X

$$f = y_3 + y_2'y_1 + y_2y_1'$$



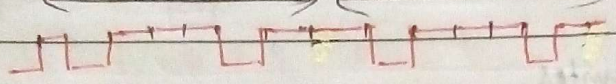
(Assuming Modulo-5 counter is present somewhere).

Pulse train generated by shift Register.

PAGE:
DATE: 22/03/17.

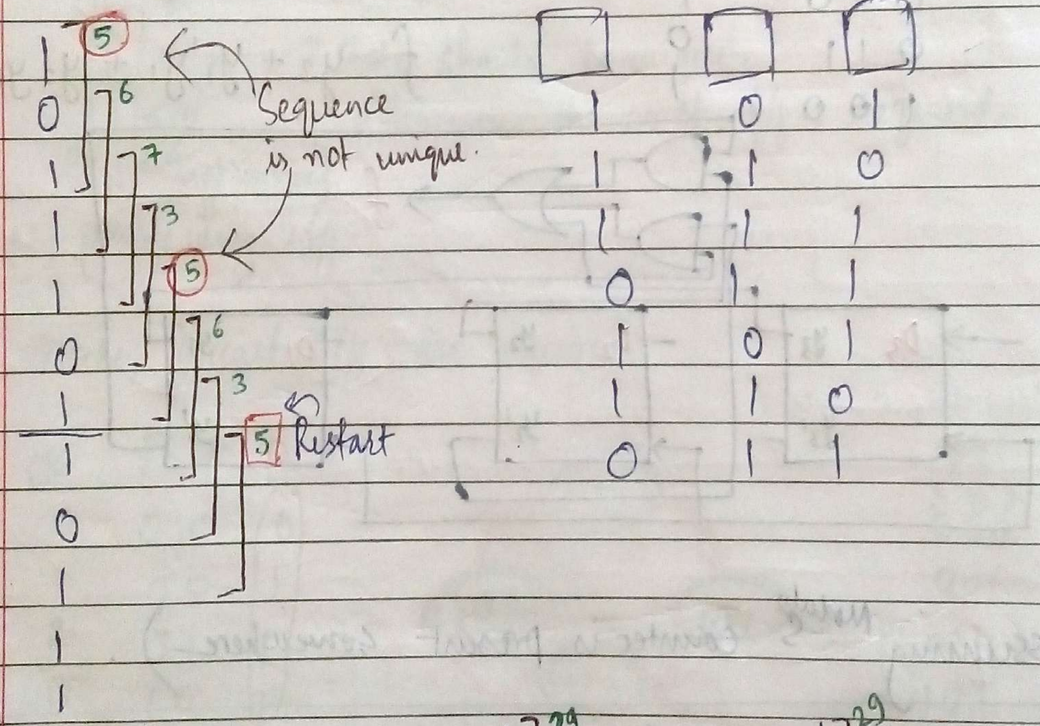
Pulse train generator using Shift Register

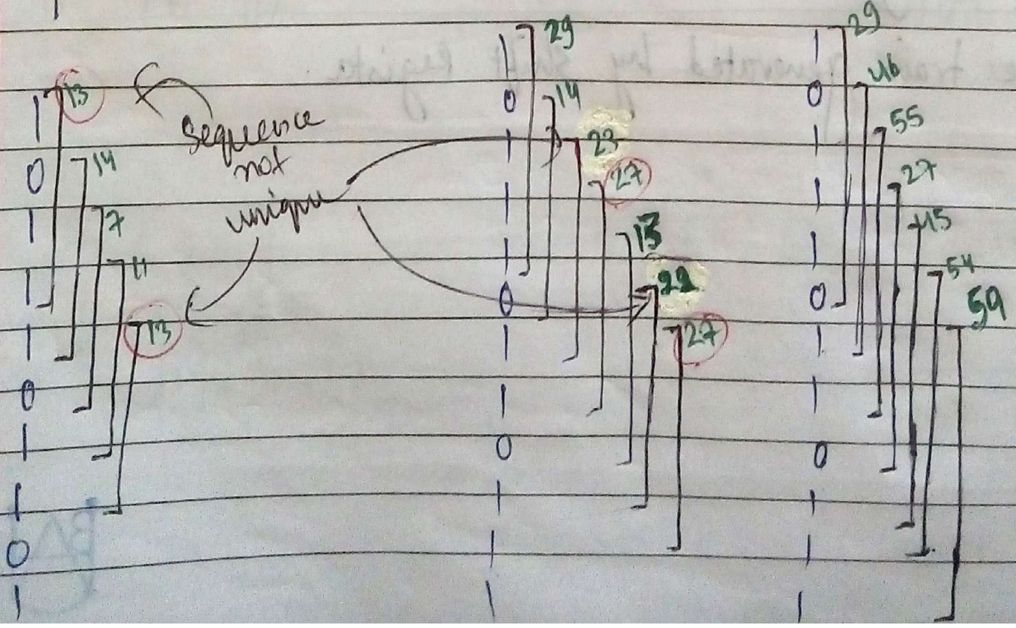
1 0 1 1 1 0 1   1 0 1 1 1 0 1

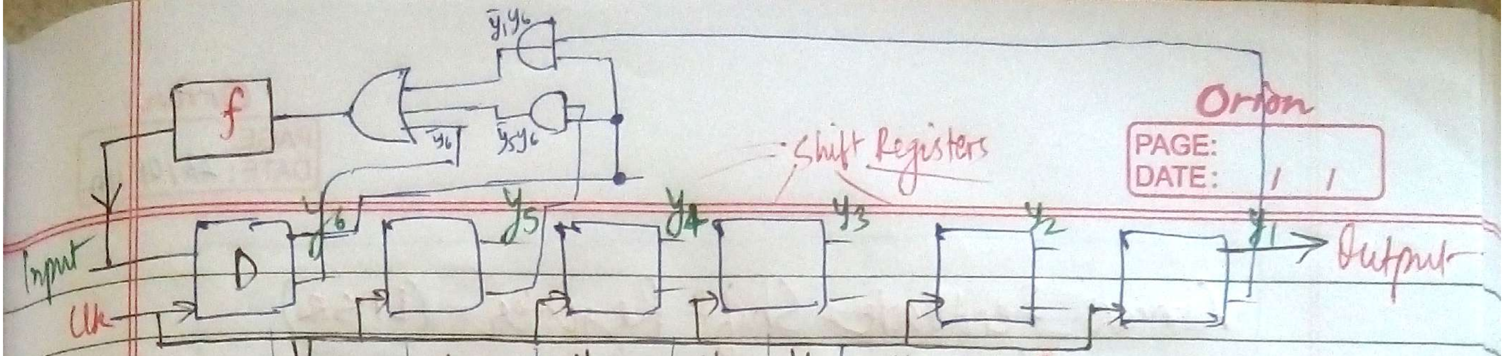


(1) How many shift Registers?

1. direct logic – synthesis of sequential ckts.
2. indirect logic – counter + addition logic.





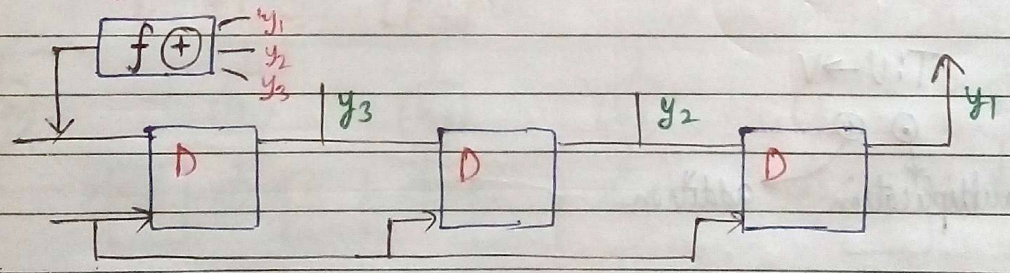


y_6	y_5	y_4	y_3	y_2	y_1	f
0	1	1	1	0	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	0
0	1	1	0	1	1	1
1	0	1	1	0	1	1
1	1	0	1	1	0	1
1	1	1	0	1	1	0

f
$y_4y_5y_6$
$y_1y_2y_3$
K-map.

$$\boxed{f = \bar{y}_6 + \bar{y}_5 y_6 + y_6 \bar{y}_1}$$

Linear Sequence Generator



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

C y_3	B y_2	A y_1		$A \oplus B$
1	1	1	7	0
0	1	1	3	0
0	0	1	1	1
1	0	0	4	0
0	1	0	2	1
1	0	1	5	1
1	1	0	6	1
1	1	1	7	0

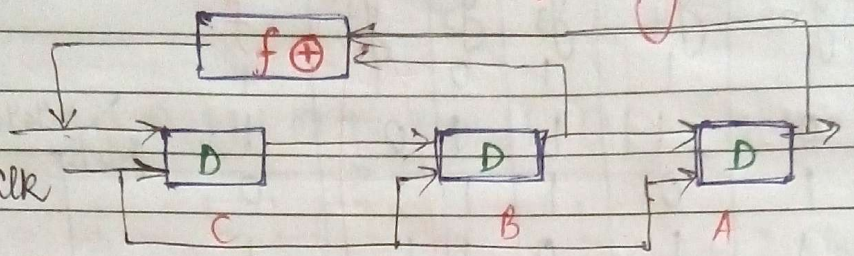
Repeat.

C y_3	B y_2	A y_1		$A \oplus C$
1	1	1	7	0
0	1	1	3	1
1	0	1	5	0
0	1	0	2	0
0	0	1	1	1
1	0	0	4	1
1	1	0	6	1
1	1	1	7	

Repeat

Linear Feedback Shift Register (LFSR)

Linear



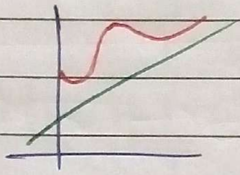
1 1 1

$f =$ A B C , $A \oplus B$, $A \oplus C$, $B \oplus C$, $A \oplus B \oplus C$

$2^3 - 1 = 7$

It generates all possible sequences except all 0.

$y = ax' + b$
affine function



F , GF(P) , GF(2) , 0 , 1

$T: U \to V$

$\odot$ $\oplus$

multiplication → addition

$$A \odot U_1 \oplus B \odot U_2 = A \odot T(U_1) \oplus B \odot T(U_2)$$

A, B elements of F
A, B {0,1}

f → $\oplus$ → Linear function

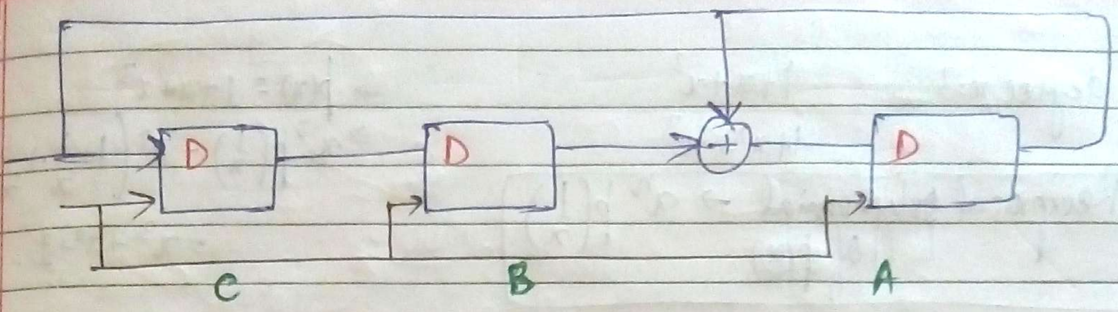
$\oplus$ – modulo 2 addition $\boxed{\text{XOR}}$

$\oplus$ – binary addition (with carry)

f – SOP, POS

AND, OR, NOT

XOR – {AND, OR, NOT}

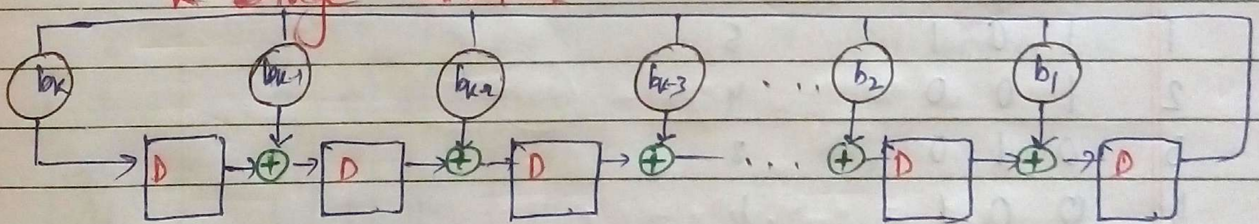


CLK	C	B	A	
0	1	1	1	7 (Initial seed)
1	1	1	0	6
2	0	1	1	3
3.	1	0	0	4
4	0	1	0	2
5	0	0	1	1
6	1	0	1	5
7	1	1	1	7

feedback polynomial
$f(x) = 1 + x^2 + x^3$.

---

k-Stage LFSR



$b_j = 1$, presence of feedback
$b_j = 0$, absence of feedback.

feedback polynomial of LFSR $p(x) = x^k + b_1 x^{k-1} + b_2 x^{k-2} + \dots + b_{k-1} x + b_k$.

$p(x) = \underline{1} + \dots + \underline{x^k}$ $(b_k = 1)$
k-stage LFSR

Primitive polynomial → irreducible, if the polynomial $f(x)$ of degree m divides $(x^{p^m - 1} - 1)$
↳ generate all elements.

degree, m=3 $\quad 1+x+x^3$

$1+x^2+x^3$

$$\boxed{\text{Reciprocal polynomial of } p(x) \rightarrow x^n p\left(\frac{1}{x}\right)}$$

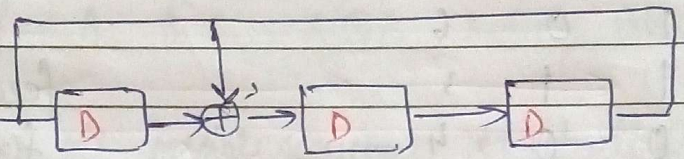
$\rightarrow p(x) = 1+x+x^3$

$\Rightarrow x^3 p\left(\frac{1}{x}\right) = x^3\left(1+\frac{1}{x}+\frac{1}{x^3}\right)$

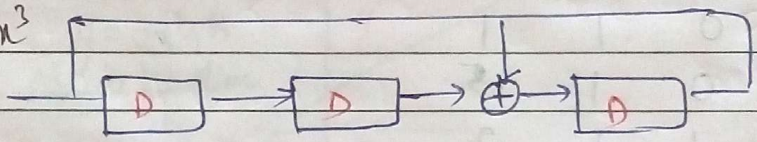
$= x^3+x^2+1.$

If $p(x)$ is primitive then reciprocal of $p(x)$ is also primitive.

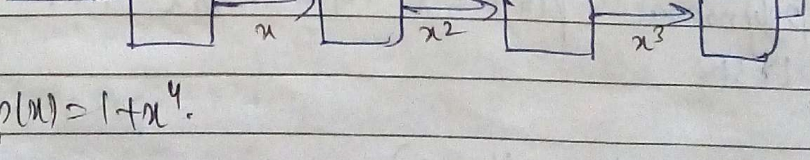
$1+x+x^3$



$1+x^2+x^3$



Clk.	C	B	A	
0	1	1	1	(IS) 7
1	1	0	1	5
2	1	0	0	4
3	0	1	0	2
4	0	0	1	1
5	1	1	0	6
6	0	1	1	3
7	1	1	1	

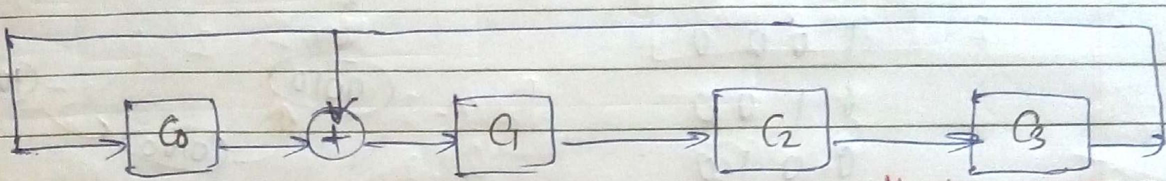


$p(x) = 1+x^4.$

Randomness → Knuth – Vol 2
Art of Computing
NIST. Randomness

PAGE:
DATE: 24/03/17

LFSR LINEAR FEEDBACK SHIFT REGISTER.



4-stage LFSR

feedback polynomial $p(x) = 1 + x + x^4$.
(degree 4 polynomial)

$2^4 - 1 = 15$

Clk	C_0	C_1	C_2	C_3	
0	1	0	1	0	10
1	0	1	0	1	5
2	1	1	1	0	14
3	0	1	1	1	7
4	1	1	1	1	15
5	1	0	1	1	11
6	1	0	0	1	9
7	1	0	0	0	8 ✓
8	0	1	0	0	4
9	0	0	1	0	2
10	0	0	0	1	1
11	1	1	0	0	12
12	0	1	1	0	6
13	0	0	1	1	3
14	1	1	0	1	13
15	1	0	1	0	✓ 10

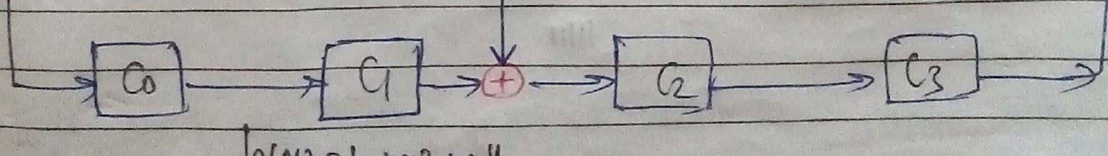
0000 ?

Pseudo Random Pattern generator.
(Deterministic)

Sequence generated satisfies $\underbrace{\text{randomness properties.}}$

1. Balanced
2. Co-relation is less

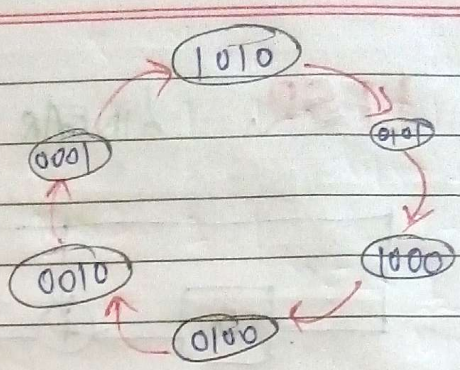
n stage LFSR with primitive polynomial will generate $2^n - 1$ sequences. (max. length sequence).

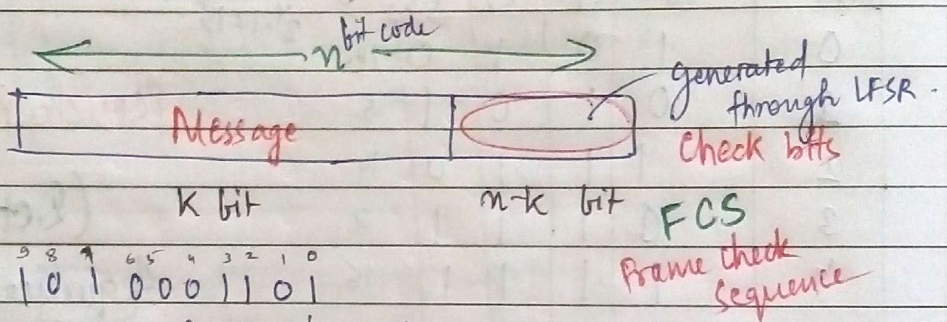


$p(x) = 1 + x^2 + x^4$.

Orion
PAGE:
DATE: / /

Clock	C_0	C_1	C_2	C_3
0	1	0	1	0
1	0	1	0	1
2	1	0	0	0
⋮	0	1	0	0
	0	0	1	0
	0	0	0	1
	1	0	1	0





Message - $\overset{9}{1}\,\overset{8}{0}\,\overset{7}{1}\,\overset{6}{0}\,\overset{5}{0}\,\overset{4}{0}\,\overset{3}{1}\,\overset{2}{1}\,\overset{1}{0}\,\overset{0}{1}$

$m(x) - 1.x^0 + 0.x^1 + 1.x^2 + \ldots$

$1 + x^2 + x^3 + x^7 + x^9.$

$p(x) = 1 + x^2 + x^4 + x^5 \; (110101)$

$$\frac{m(x)}{p(x)} = q(x) + \frac{r(x)}{p(x)}$$

Code $= x^5 . m(x) + r(x)$

$(x^{14} + x^{12} + x^8 + x^7 + x^5) + $ ~~xxxx~~ FCS

$$\begin{array}{l}
x^5+x^4+x^2+1 \,\big)\, x^9+x^7+x^3+x^2+x^0 \,\big(\, x^4+x^3+x \\
\quad x^9+x^8+x^6+x^4 \\
\hline
\quad +x^8+x^7+x^6+x^4+x^3+x^2+1 \\
\quad x^8+x^7+x^5+x^3 \\
\hline
\quad x^6+x^5+x^4+x^2+1 \\
\quad x^6+x^5+x^3+x \\
\hline
\quad x^4+x^3+x^2+x+1
\end{array}$$

Modulo 2 addition ≡ Subtraction

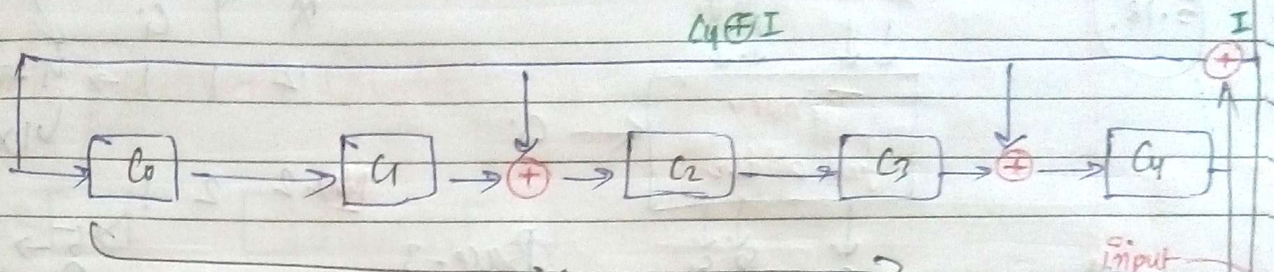
11111

$m(x)$ = dividend

$p(x)$ = divisor.

CRC → Cyclic Redundancy Code

PRP → Pseudo Random Pattern generator.



$1+x^2+x^4+x^5$ ← Remainder

	C_{4_t}	C_{3_t}	C_{2_t}	C_{1_t}	C_{0_t}	$C_{4_{t+1}}$: $C_4 \oplus C_3 \oplus I$	$C_{2_{t+1}}$: $C_4 \oplus C_1 \oplus I$	$C_{0_{t+1}}$: $C_4 \oplus I$	I (input)
0	0	0	0	0	0	1	1	1	1
1	1	0	1	0	1	1	0	1	0
2	1	1	1	1	1	1	1	0	1
3	1	1	1	1	0	0	0	1	0
4	0	1	0	0	1	1	0	0	0
5	1	0	0	1	0	1	0	1	0
6	1	0	0	0	1	0	0	0	1
7	0	0	0	1	0	1	0	1	1
8	1	0	0	0	1	1	1	1	0
9	1	0	1	1	1	0	1	0	1

$x^k \cdot m(x) + r(x)$

$x^5+x^4+x^2+1 \,\big)\, x^{14}+x^{12}+x^8+x^7+x^5 \,\big(\, x^9+x^8+x^6+x^4+x^2+x$

$\cancel{x^{14}}+x^{13}+x^{11}+x^9$

$x^{13}+x^{12}+x^{11}+x^9+x^8+x^7+x^5$

$\cancel{x^{13}}+\cancel{x^{12}}+x^{10}+\cancel{x^8}$

$x^{11}+x^{10}+x^9+x^7+x^5$

$\cancel{x^{11}}+\cancel{x^{10}}+x^8+x^6$

$x^9+x^8+x^7+x^6+x^5$

$\cancel{x^9}+\cancel{x^8}+\cancel{x^6}+x^4$

$x^7+x^5+x^4$

$\cancel{x^7}+x^6+\cancel{x^4}+x^2$

$x^6+x^5+x^2$

$\cancel{x^6}+\cancel{x^5}+x^3+1$

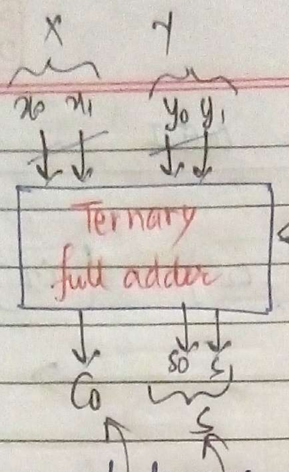
x^3+x^2+1.

110101010110

Orion

PAGE:
DATE: / /

5.18.



$\frac{X+Y}{3}$

Base-3
0, 1, 2

X	Y	C_i	divide by 3	C_o	S
2	1	1 →	4 →	1	1

$\hookrightarrow 1\times3^1 + 1\times3^0$
$3+1=4$

Max. 2 2 1 → 5 → 1 2
Max. Max. Max.

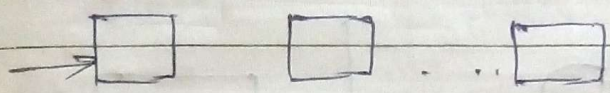
$(2)_3 + (2)_3 + (1)_3 = (12)_3$

* Carry cannot become 2.
Quotient
Reminder

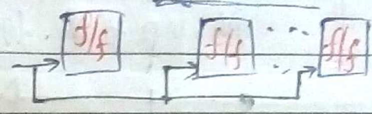
C_i	X (x_1 x_0)	Y (y_1 y_0)		C_o	S (s_1 s_0)
0	0 0	0 0	0	0	0 0
0	0 0	0 1	1	0	0 1
0	0 0	1 0	2	0	1 0
0	0 1	0 0	1	0	0 1
0	0 1	0 1	2	0	1 0
0	0 1	1 0	3	1	0 0
0	1 0	0 0	2	0	1 0
0	1 0	0 1	3	1	0 0
0	1 0	1 0	4	1	0 1
1	0 0	0 0	1	0	0 1
1	0 0	0 1	2	0	1 0
1	0 0	1 0	3	1	0 0
1	0 1	0 0	2	0	1 0
1	0 1	0 1	3	1	0 0
1	0 1	1 0	4	1	0 1
1	1 0	0 0	3	1	0 0
1	1 0	0 1	4	1	0 1
1	1 0	1 0	5	1	1 0

Class Test: Wed, Apr 5 (5-6pm)
Syllabus: Combination & Sequential ckt design

Asynchronous Counter

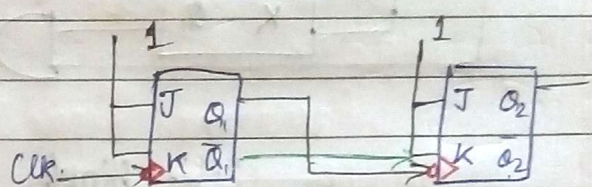


Synchronous:

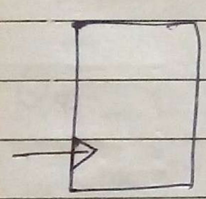


Single clock only.

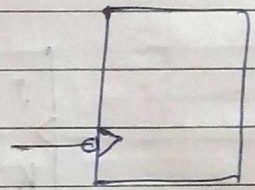
0	00
1	01
2	10
3	11
	00



Ripple counter
(Asynchronous counter, Previous output feeds the current input).

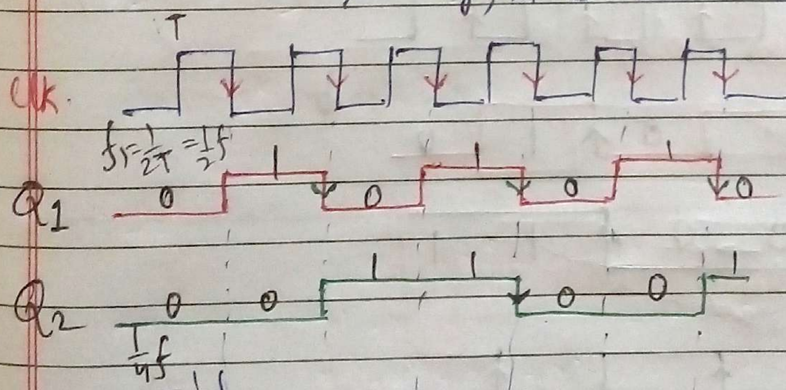


Positive edge triggered.



Negative edge of the clock.

initially, it is reset $Q_1=0$, $Q_2=0$



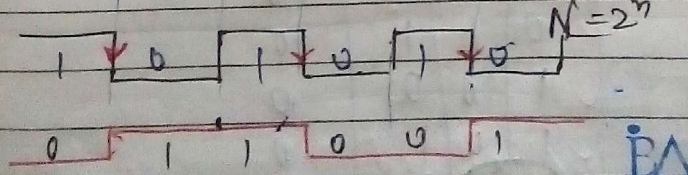
Up counter

Q_2	Q_1
0	0
0	1
1	0
1	1

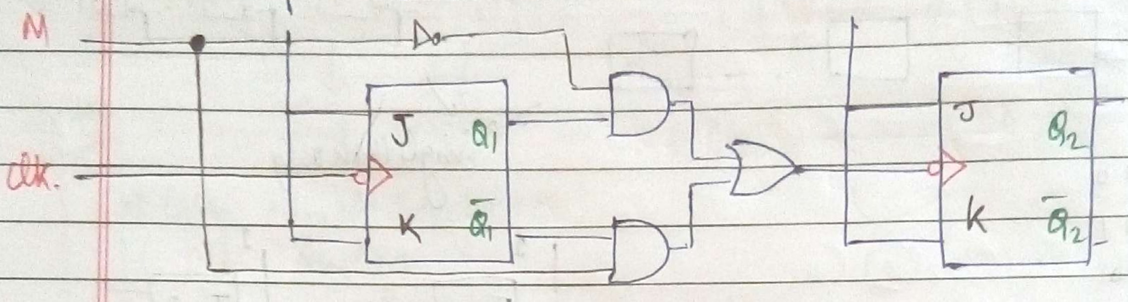
Asynchronous design is usually simpler than synchronous design.
But: Problems:-
* Propagation delay. (If $\bar{Q}_1$ is connected to K_2) green pen

Down counter:-

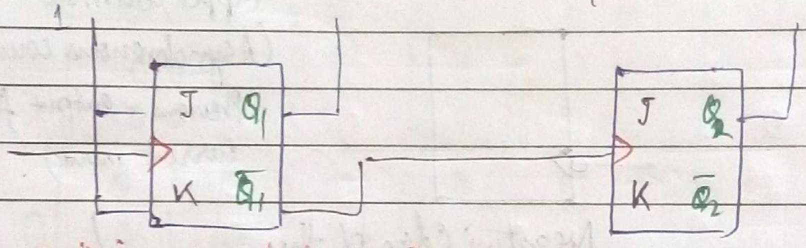
Q_2	Q_1
0	0
1	1
1	0
0	1
0	0



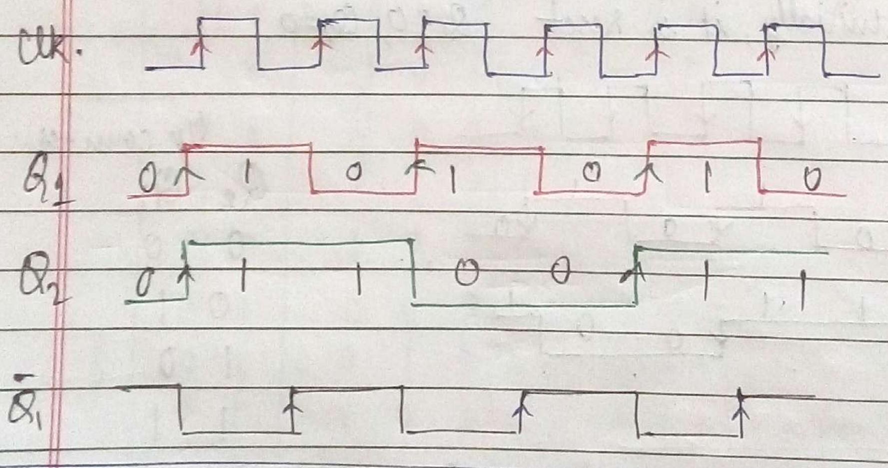
Mode $M=0$ (up)
$M=1$ (down)
$\bar{M}.up + M.down$



2-bit Ripple up/down counter.



positive-edge clock - 2 bit up counter

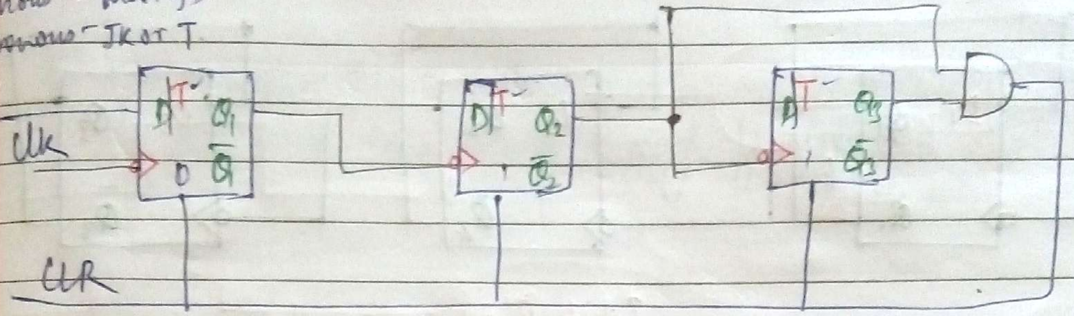


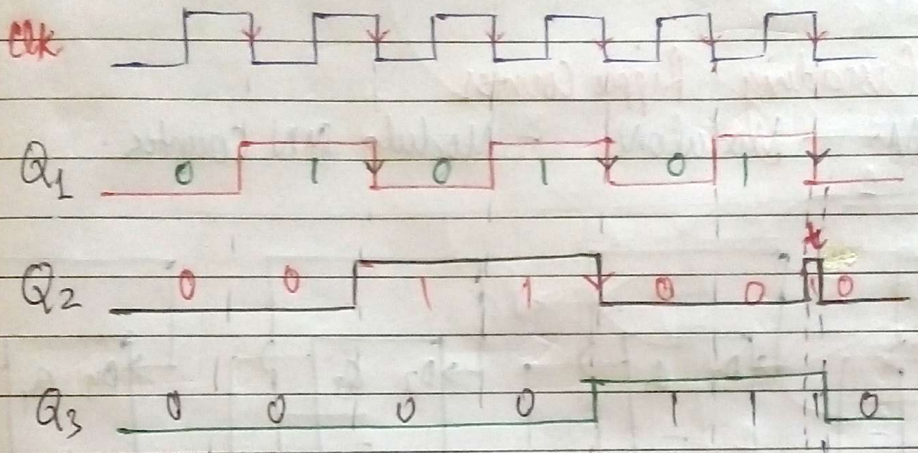
Modulo-6 counter:

Q_3	Q_2	Q_1	CLR
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
0	0	0	111 - X

$Q_3 Q_2 \bar{Q}_1 + Q_3 Q_2 Q_1$

Synchronous - Mostly D
Asynchronous - JK or T



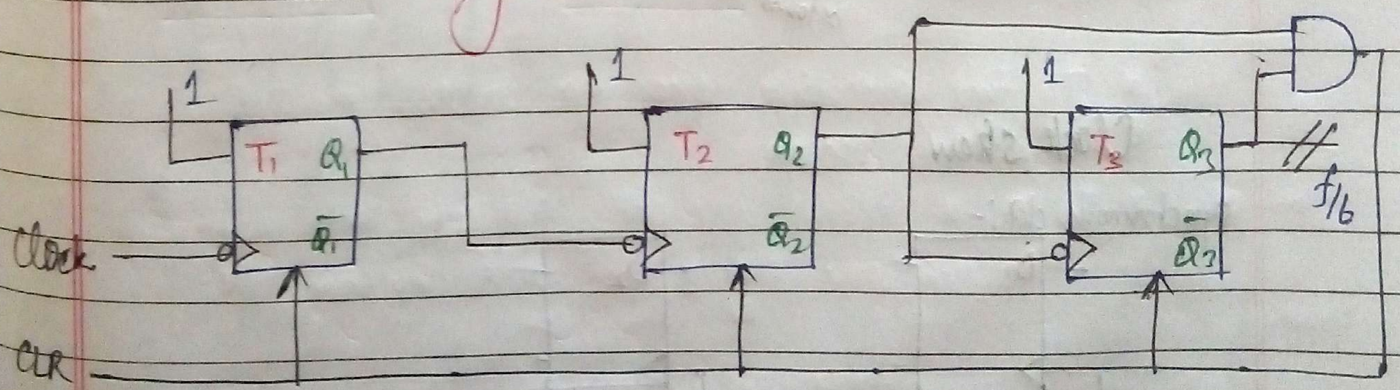


$Q_1 = Q_2 = Q_3 = 0$ $\qquad$ t = propogation delay of 2-input AND.

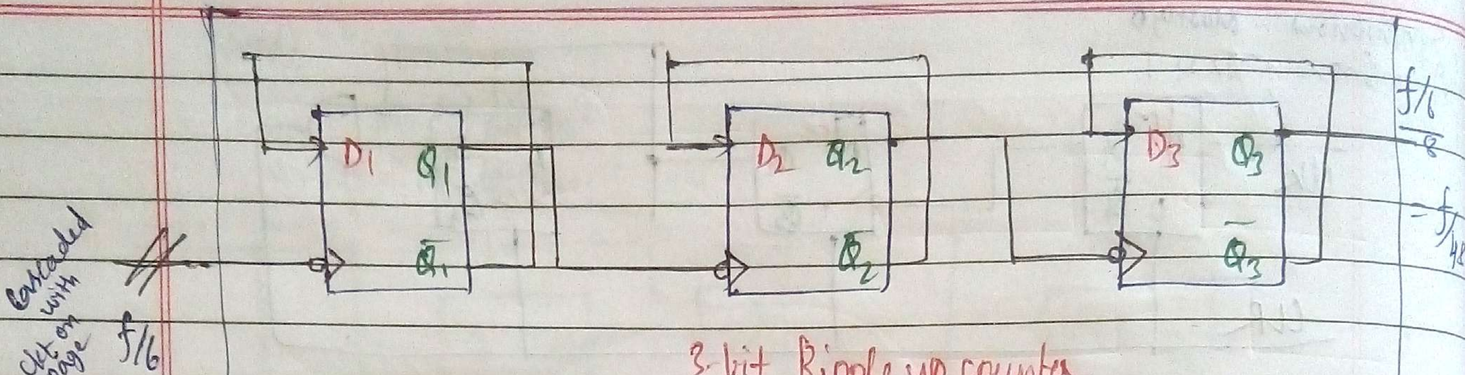
$f \rightarrow f/6$.

30/03/17.

Asynchronous Counter



Modulo-6 (T f/f)

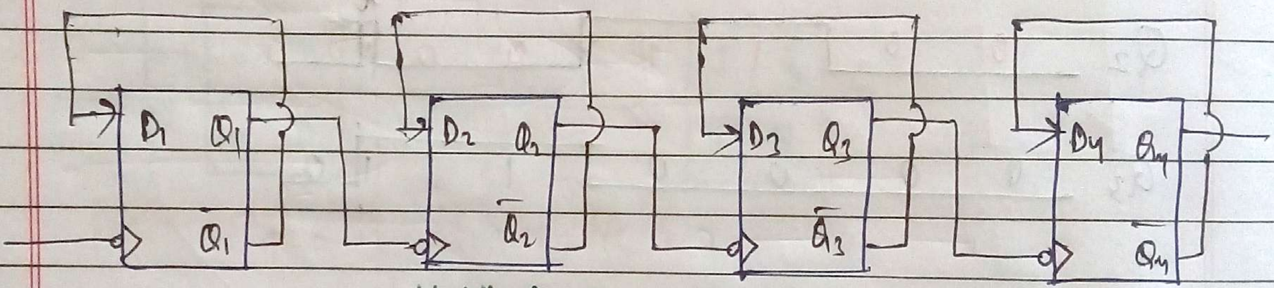


3-bit Ripple up counter

Mod-8 counter.

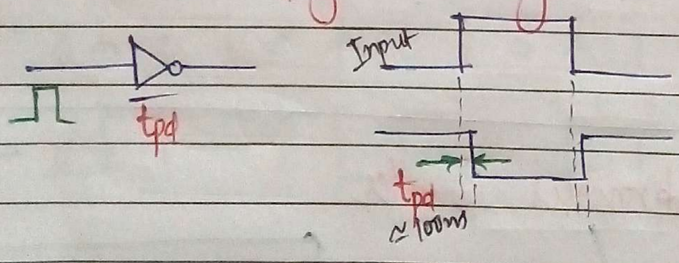
Cascading Ripple Counter

Modulo M — Modulo N = Modulo MN Counter.



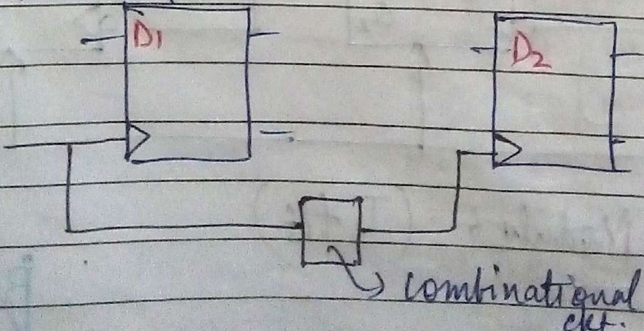
4-bit Ripple up Counter

Effect of Propagation Delay



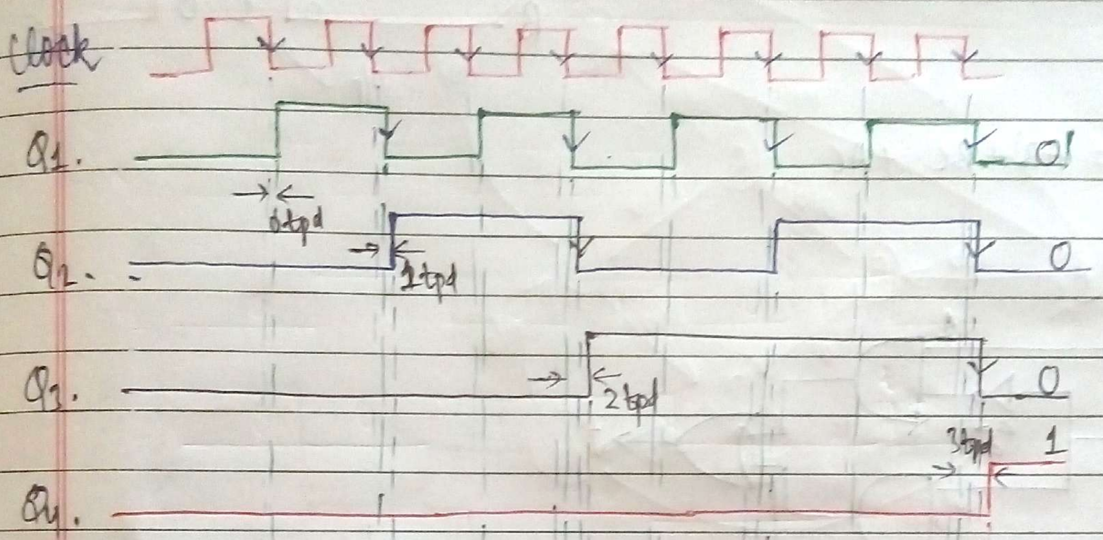
Clock skew

synchronous ckt:-



→ combinational ckt. → Propagation delay

In asynchronous ckts, even if there is no combinational ckt, there is a process time delay through flipflops.

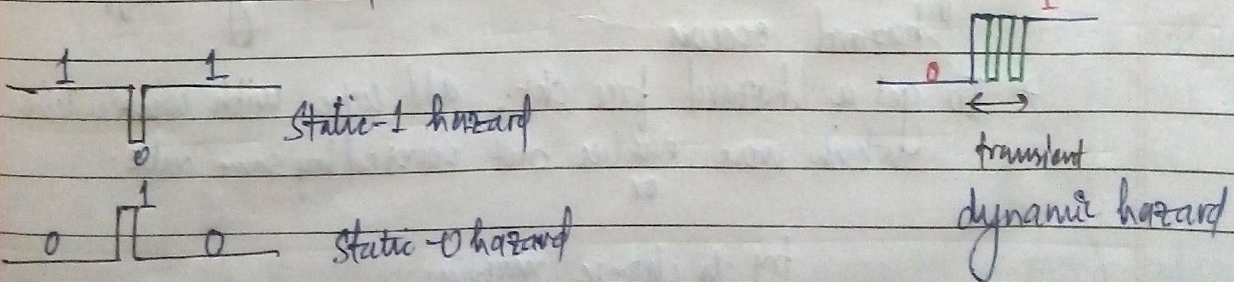


0 1 1 1
1 0 0 0
1 0 0 1

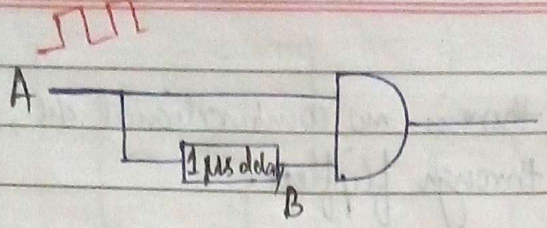
if $\left(3t_{pd} > \frac{T_{clk}}{2}\right)$

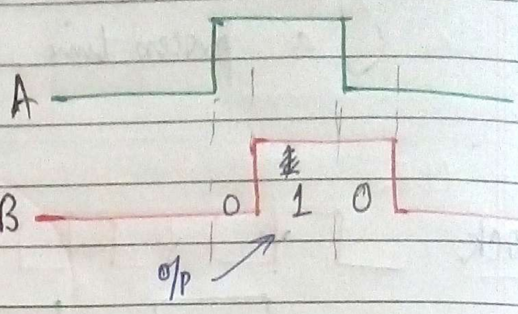
Hazard.

- 0 hazard
- 1 hazard

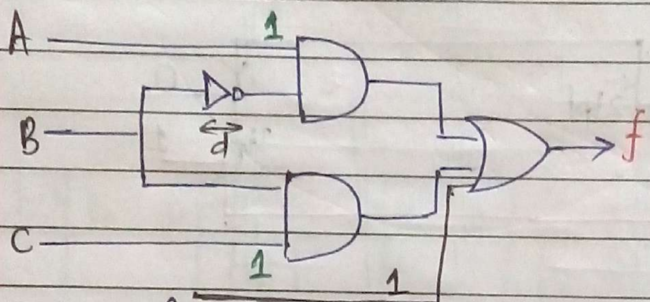


Combinational ckt. Sequential ckt.



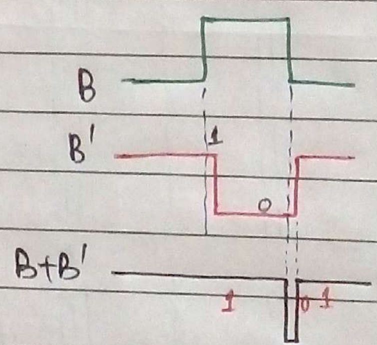


$f = AB' + BC + AC$



$f = B + B' = 1$., if $A=1, C=1$

↳ if static 1 is added, no hazard.



Static-1 hazard.

BC \ A	0	1
00		1
01		1
11	1	1
10		

→ AC

* If two consecutive ones are covered by 2 diff cubes then hazard occurs.
* To get a hazard free ckt, add cube with consecutive 1's which were earlier not covered by same cube.

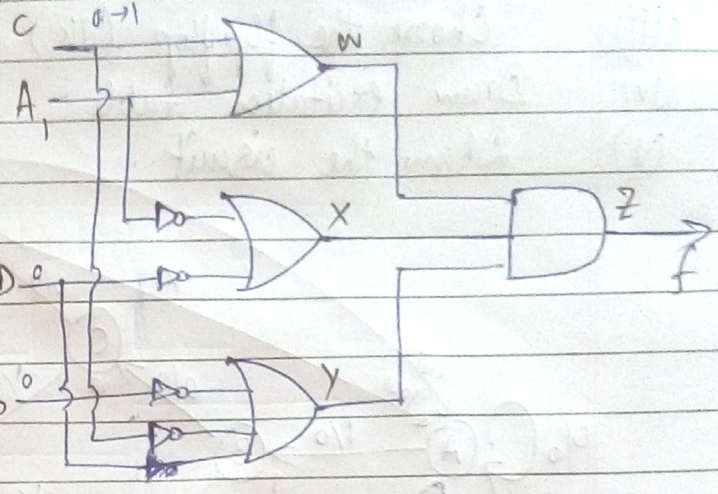
or

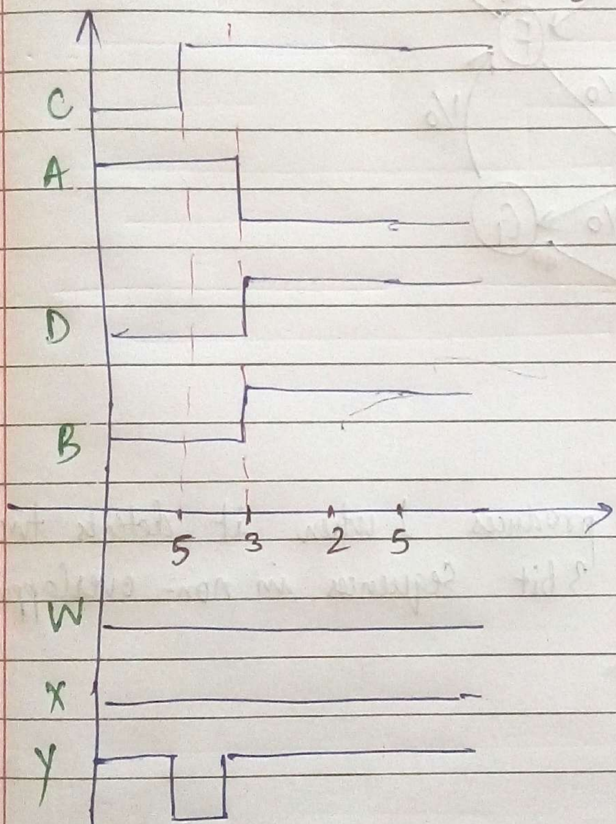
try to change mapping.

Product of Sums

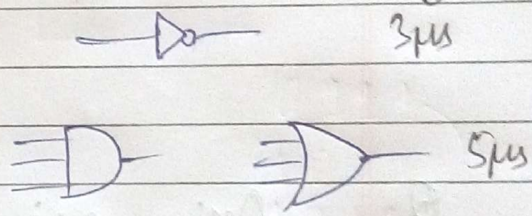
Ex. (A+C) (A'+D') (B'+C'+D)

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





Assume propogation delay (t_{pd})



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

Hazard free Ckt

Q Design a (2input, 2output) circuit that produces a output 1 wherever it detects the sequence 1100 or 1010 or 1001. The circuit goes back to its initial state after giving an output 1.

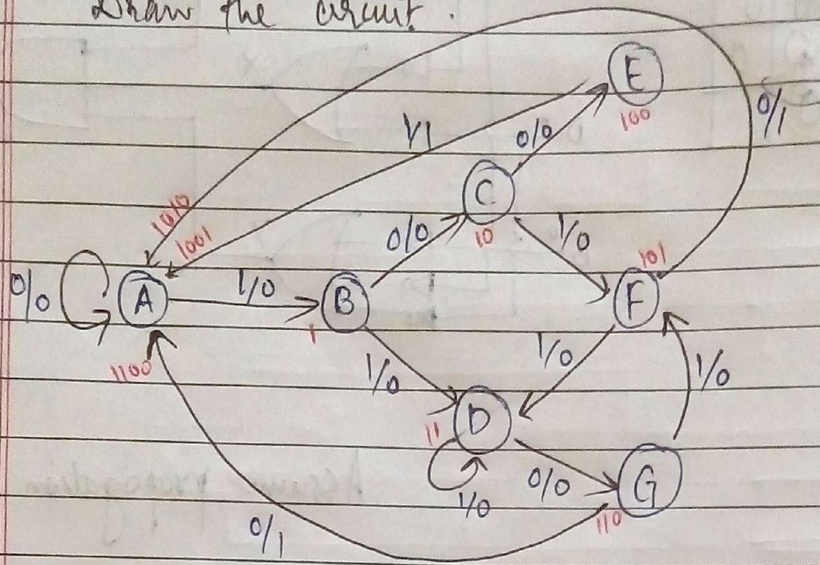
(n no. of 1's (0's) in preceding m sequences m=4) Kohavi

Input: 0 1 0 1 0 1 1 1 0 1

Output: 0 0 0 0 0 0 1 1 1 1

Orion
PAGE:
DATE: / /

Steps:- (i) Draw the state transition diagram
(ii) Draw state table
(iii) Choose the flip flop (JK)
(iv) Draw excitation table
(v) Draw the circuit.



Q. Design a circuit which produces 1 when it detects two 1's or three 1's in the preceeding 3 bit sequences in non- overlapping condition.

Q. Asynchronous
Counter design → Calculation of safe-frequency.

Q. For what value of propogation delay a 10 bit counter misses a count with the clock frequency of 10MHZ. (10 F/F)

$$f_c < \frac{1}{n t_{pd}}$$

$$t_{pd} = \frac{1}{10 f_c} = \frac{1}{10 \times 10^7} = 10 \text{ ns}$$

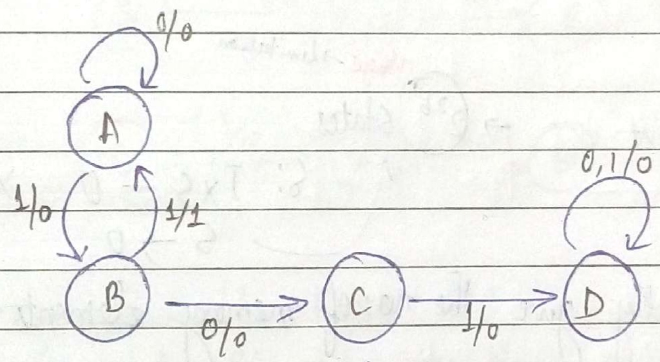
Capabilities and Limitations of Sequential Machines

Def^n

Sequential Machines: It is a quintuple $\langle I, O, S, \delta, \lambda \rangle$

I : Input
O : Output
S : State
δ : Output function
λ : State-transition function

λ: $S(t+1) = \lambda\{S(t), x(t)\}$
$I \times S \to S$, X-cartesian product
$Z(t) = \delta(S(t))$, Moore M/c
$= \delta(S(t), x(t))$, Mealy M/c



Sink
D: terminal/ graveyard state

Source, no other state exists for where the M/c can enter into that state

Input : 100, 101

Input	Output
00	00
01	00
10	00
11	01
000	000
001	000
010	000
011	001
100	...
110	010
111	010

N states → $\lceil \log_2 N \rceil$ Memory elements.

Input: 1 1 1 1 1 1 1 1 1 1 ...

$1 \quad \frac{1\times2}{2}$
$3 \quad \frac{2\times3}{2}$
$6 \quad \frac{3\times4}{2}$
10
⋮

$\boxed{K(K+1)/2}^{th}$ 1 at I/P ⇒ 1 at O/P

Output: 1 0 1 0 0 1 0 0 0 1 ...

It is difficult to design M/C.

Serial Adder
Carry – 0/1

Serial Multiplier
b - input
b - input
2b - bit output → 2^{2b} states (Huge → Limitation)

$\delta: I \times S \to O$ ✗
$S \to O$

no. of state directly give the no. of memory elements.

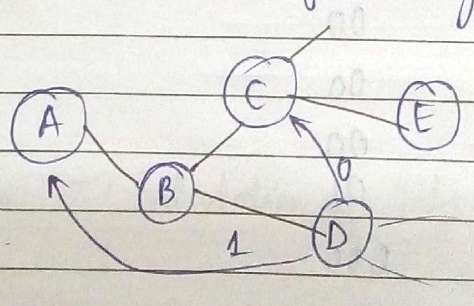
Minimization.

Limitation is due to the no. of memory elements.
→ Whether we can reduce the no. of memory elements (f/fs).

Observation:

There exist redundancy which eventually results the repeating behaviour of the m/c.

b-input → 2^b



Testing & verification of ckts.
Redundancy is not wanted.
Minimization of Sequential ckt.

State Minimization

S_i, S_j are two states of M
If we apply the same input sequence X to S_i and S_j, then their behaviour is same i.e they produce same output sequences.

$S_i = S_j$, $S_j = S_k$
$\Rightarrow S_i = S_k$

$M_{redundant}$ — M_1

M/c with redundant states $\longrightarrow M_1$ without redundant states.

X = 101

$$S_i \xrightarrow{1/0} A \xrightarrow{0/1} B \xrightarrow{1/1} C$$

$$S_j \xrightarrow{1/0} A' \xrightarrow{0/1} B' \xrightarrow{1/1} C'$$

k – distinguishable states

If S_i and S_j are the state of a machine M, if for a minimum value of K applying the k-sequences show a different output sequence then S_i and S_j are k-distinguishable.

$$S_i \xrightarrow{1/0} A \xrightarrow{0/1} B \xrightarrow{1/1} C$$

$$S_j \xrightarrow{1/0} A \xrightarrow{0/1} B \xrightarrow{1/0} C$$

K=3 ; 101

Outputs { S_i : 011
{ S_j : 010

i/p 1010
1011

K=3

M – n states ; max^m input sequence $\Rightarrow$ n-1
(to check)

PS	NS, Z	
	x=0	x=1
A	E, 0	D, 1
B	F, 0	D, 0
C	E, 0	B, 1
D	F, 0	B, 0
E	C, 0	F, 1
F	B, 0	C, 0

1-sequence 0, 1

A, B are 1 distinguishable

A, C $\xrightarrow{0,1}$ 1 equivalent

$P_0 = \{A, B, C, D, E, F\}$

1 equivalent states sets → $P_1 = \{A, C, E\}, \{B, D, F\}$

$P_2 = \{A, C, E\}, \{B, D\}, \{F\}$

$P_3 = \{A, C\}, \{E\}, \{B, D\}, \{F\}$.

$P_4 = \{A, C\}, \{E\}, \{B, D\}, \{F\}$

$P_5 = \{A, C\}, \{E\}, \{B, D\}, \{F\}$.

If $P_K = P_{K+1}$, It is K equivalent, no need to check for K+2 or further.

7/4/17

Theorem 1 : The equivalence partition is unique.

$P_0 = \{A, B, C, D, E, F\}$

$\{A, C\}, \{E\}, \{B, D\}, \{F\}$.

Let P_a and P_b be two partition.
There exist atleast two states S_i & S_j whose position in a block is different in P_a & P_b.

$M \rightarrow M_1$

M_1 - minimized

$P_a = \{S_i, S_j, \ldots\}\{\ldots\}$

$P_b = \{S_i, \ldots\}\{S_j, \ldots\}$

Now, say S_i and S_j are in two different blocks in partition P_b whereas they are in same block in partition P_a.

Since S_i and S_j are in 2 diff. blocks in P_b. So, there exists atleast one input sequence which produces diff^n output when applied to S_i and S_j. So, S_i and S_j are not equivalent states.
So, S_i and S_j cannot be in same block of P_a.
Hence, our assumption was wrong. S_i & S_j cannot ~~be~~ have diff positions in partitions ⇒ Eq. Partition ~~are~~ is unique.

Theorem 2: If the two states S_i and S_j of machine M are distinguishable then they are distinguishable by a sequence of length $(n-1)$ or less where n is the no. of states in M.

$P_0 = \{A, B, C, D, E, F\}$
$P_1 = \{A\} \; \underbrace{\{B, C, D, E, F\}}_{n-1}$

$P_i < P_j$, if each block of P_i is contained in P_j.

$P_{K+1} < P_K$ — P_K is the partition when input sequence of length K is applied.

K is restricted by $(n-1)$.
that means increasing length of input sequence by 1, results another partition.
Partitioning will be stopped when $P_{K+1} = P_K$

$P_3 < P_2$.

$P_3 = \{A, C\}, \{E\}, \{B, D\}, \{F\}$
$\alpha \quad \beta \quad \gamma \quad \delta$

$\alpha = A$; $\alpha = C$;
$\beta \quad \alpha \quad \delta \quad \gamma$ (Isomorphism)

PS	NS, Z: x=0	NS, Z: x=1
A	E,0	D,1
B	F,0	D,0
C	E,0	B,1
D	F,0	B,0
E	C,0	F,1
F	B,0	C,0

→

PS	NS, Z: x=0	NS, Z: x=1
α	β,0	γ,1
β	α,0	δ,1
γ	δ,0	γ,0
δ	γ,0	α,0

6 states → 4 states.

Complete Machine

Transitions (next states) and the behaviour (output sequence) of machine M is completely specified.

Incomplete Specified

→ Transitions and/or behaviour of machine M is not specified (defined).

PS	NS, Z: x=0	NS, Z: x=1
A	B,1	–,0
B	–,0	C,0
C	A,1	B,0

Incompletely specified

→

PS	NS, Z: x=0	NS, Z: x=1
A	B,1	T,0
B	T,0	C,0
C	A,1	B,0
T	T,–	T,–

We don't care about behaviour of state T as it is not defined in our machine.
⇒ It is completely specified

PS	NS, Z: x=0	NS, Z: x=1
A	C,1	E,– (α')
B	C,– (β')	E,1
C	B,0	A,1
D	D,0	E,1
E	D,1	A,0

→

PS	NS, Z: x=0	NS, Z: x=1
α	β,1	α,0
β	β,0	α,1

PS	NS, Z: x=0	NS, Z: x=1
α	β,1	γ,1
β	α,0	α,1
γ	β,1	α,0

Output=0 $P_1 = \{A, E\}_{\alpha} \{B, C, D\}_{\beta}$

Output=1 $P_1 = \{A, B\}_{\alpha} \{C, D\}_{\beta} \{E\}_{\gamma}$

PS	NS: x=0	NS: x=1
A	A,0	C,0
B	B,0	B,–
C	B,0	A,1

PS	NS: x=0	NS: x=1
A	A,0	C,0
B'	B',0	B'',–
B''	B',0	B',–
C	B^+,0	A,1

State Splitting

$B \rightarrow \{B', B''\}$

$B^+ = B'$
$\quad = B''$

PAGE:
DATE: / /

$\{A, B'\}, \{B'' C\}$
$\alpha \quad \beta$

α	$\alpha, 0$	$\beta, 0$
β	$\alpha, 0$	$\alpha, 1$

$\{A, B'\}, \{B'' C\}$
$\alpha \quad \beta$

α	$\alpha, 0$	$\beta, 0$
β	$\beta, 0$	$\alpha, 1$

Example.

PS	NS $x=0$	$x=1$
A	E, 0	C, 0
B	C, 0	A, 0
C	B, 0	G, 0
D	G, 0	A, 0
E	F, 1	B, 0
F	E, 0	D, 0
G	D, 0	G, 0

Find the partitions and minimize the machine.

$P_1 = \{A, B, C, D, F, G\}\{E\}$
$P_2 = \{A, F\}, \{B, C, D, G\}\{E\}$
$P_3 = \{A, F\}, \{B, D\}, \{C, G\}, \{E\}$
$P_4 = \{A\}, \{F\}, \{B, D\}, \{C, G\}, \{E\}$
$P_5 = \{A\}, \{F\}, \{B, D\}, \{C, G\}, \{E\} = P_6$
$\beta \quad \gamma \quad \delta \quad \epsilon \quad \alpha$

PS	NS $x=0$	$x=1$
α	$\gamma, 1$	$\delta, 0$
β	$\alpha, 0$	$\epsilon, 0$
γ	$\alpha, 0$	$\delta, 0$
δ	$\epsilon, 0$	$\beta, 0$
ϵ	$\delta, 0$	$\epsilon, 0$

12/4/17

Def^n

Covers / Contain - State S_i of m/c M_1 is said to cover or contain state S_j of M/c M_2, iff the input sequence which is applicable to S_j is also applicable to S_i and when then input sequence is applied to both M_1 and M_2, with the initial state S_i & S_j respectively they produce same output sequences provided output sequences are specified.

ANUBHAV JAIN NOTES

Compatible - Two states S_i & S_j of a m/c M are compatible iff every input sequence applicable to S_i & S_j, the same

Output sequence will be produced whenever, both output symbol are specified.

$\{S_i, S_j\}$, $\{S_j, S_k\}$, $\{S_i, S_l\}$ $\rightarrow$ $\{S_i, S_j, S_k, S_l\}$ $\downarrow$ α

M

P.S	NS, Z: I_1	I_2	I_3	I_4
A	—	C,1	E,1	B,1
B	E,0	—	—	—
C	F,0	F,1	—	—
D	—	—	B,1	—
E	—	F,0	A,0	D,1
F	C,0	—	B,0	C,1

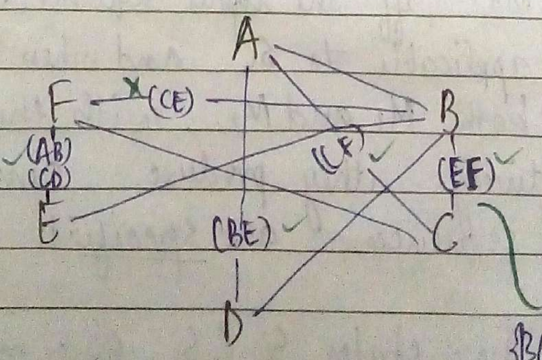
— input sequences

Merger Graph

It is a graph with n vertices, where n is the no. of states of m/c M

If the states S_i & S_j are not conflicting both next state as well as output then put an undirected arc between S_i and S_j.

If S_p and S_q are not conflicting for output but conflict for next state, put an undirected interrupt arc between S_p & S_q.



{BC} is implied by {EF}

AB, CD, BD, CF, BE, EF
AD, AC, BC.

To get a maximum compatible state (to get a complete polygon)

{ABC} {ACD} — {ABCD} {CF} {BE} {EF}

$\{ABCD\} \to \alpha$, $\{CF\} \to \beta$, $\{BE\} \to \gamma$, $\{EF\} \to \delta$

Complete polygon (ABCD) (each node is connected to other node).

$\{AB\} \to \alpha$, $\{CD\} \to \beta$, $\{EF\} \to \gamma$

Maximal compatible states give the bound of minimization.

	PS	NS, Z			
		I_1	I_2	I_3	I_4
{AB}	α	γ,0	β,1	γ,1	α,1
{CD}	β	γ,0	γ,1	α,1	—
{EF}	γ	β,0	γ,0	α,0	β,1

MERGER TABLE

13/4/17.

Ex.

PS	NS, Z	
	I_1	I_2
A	E,0	B,0
B	F,0	A,0
C	E,–	G,0
D	F,1	D,0
E	C,1	G,0
F	D,–	B,0

{Si Sj}

Si — Sj

I_k sequence to both Si & Sj

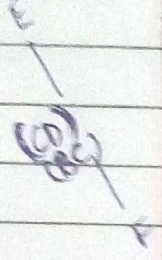
Sp — Sq

{Sp, Sq}

✓ : Compatible
X : Not compatible

B	AB EF				
C	BC	AC EF			
D	X	X	CD EF		
E	X	X	✓	CF CD	
F	DE	AB DF	BC DE	DF AB	CD BC
	A	B	C	D	E

Merger table to compatibility graph.

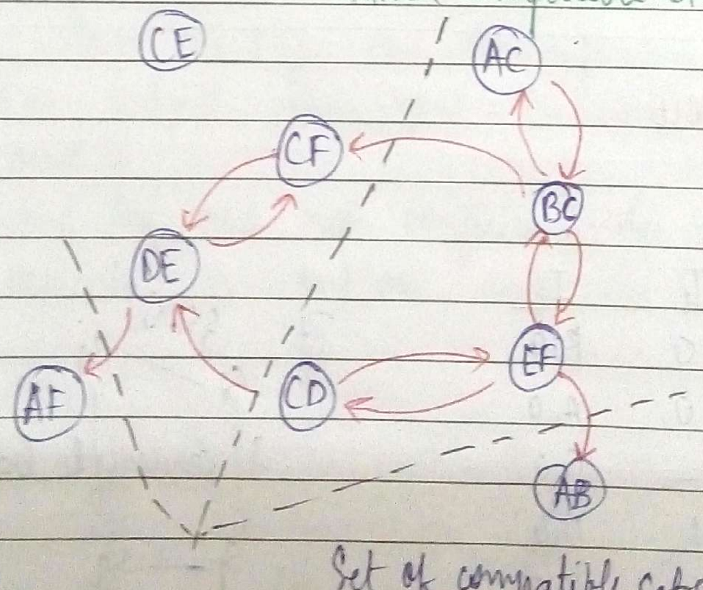


{EF} {DE} {CF} {CE} {CD} {BC} {AF} {AC} {AB}

Column	Compatible sets
E	{EF}
D	{EF} {DE}
C	{CEF} {CDE}
B	{CEF} {CDE} {BC}
A	{CEF} {CDE} {ABC} {ACF}

α β γ δ

Maximal Compatible sets.



Set of compatible sets of two nodes.
{AC} {BC} {EF} {CD} + {AB}
{ABC} {CD} {EF} → 3

PAGE:
DATE: / /

	PS	NS, Z I_1	I_2
{ABC}	α	γ,0	α,0
{CD}	β	γ,1	β,0
{EF}	γ	β,1	α,0

A	000
B	001
C	010
D	011
E	100

B, C: } 2

D, E: } 3 Hamming distance.

⇒ State Assignment can be changed.