Switching Algebra

(1) Impodency
- $x \cdot x = x$
- $x + x = x$
- $x + 1 = x$ $\quad x + 0 = x$
- $x \cdot 0 = 0$ $\quad x \cdot 1 = x$

(2) Commutative
- $x + y = y + x$
- $x \cdot y = y \cdot x$

(3) Complement
- $x + x' = 1$
- $x \cdot x' = 0$

(4) Distributive
- $x(y+z) = xy + xz$
- $x + yz = (x+y)(x+z)$

(5) Associative
- $(x+y)+z = x+(y+z)$
- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$

Switching Expression

↳ finite no of combinations of switching variables and consts using (+, ., NOT)

ex: $x + y'z + xz$

Properties:

(1) Absorption
- $x + xy = x$
- $x + x'y = x + y$

(2) Dual
- $x \cdot (x'+y) = xy$
- $\Rightarrow xx' + xy = xy$

(3) Consensus
- $xy + x'z + yz = xy + x'z$
- $\Rightarrow xy + x'z + yz$
- $= xy + x'z + yz(x+x')$
- $= xy(1+z) + x'z(1+y)$
- $= xy + x'z$.

DeMorgan's Law

(1) $(xy)' = x' + y'$

(2) $(x+y)' = x'y'$

eg $(x+y)[x(y'+z')]' + x'y' + x'z'$ Simplify

$= (x+y)[x + yz] + x'y' + x'z'$

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

$= (x+x')(x+y)' + yz + x'z'$

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

$= x + z' + y' + z' = x + y + 1 = 1$ Ans.

Switching function

$f(a,b,c) = a + bc$

↳ draw truth table

This is one way of representing.

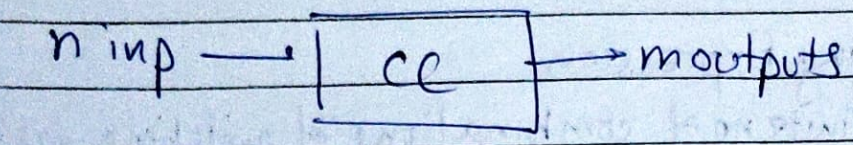
__/__/__

~~Logical Form~~

Circuits

- combinational → only input
- sequential → input and previous memory (o/p)

Combinational Circuit



cc → has logic gates in it

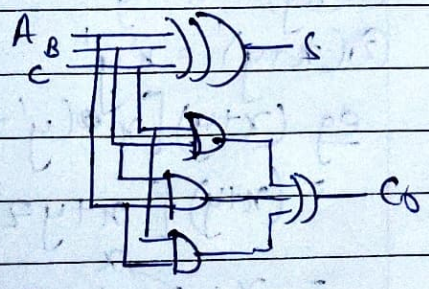
Arithmatic logic unit

- performs arithmetic and logical operations
- ~~adders & gates~~ → addition subtraction
- ~~1's complement ckt~~, half adder, full adder.

Full adder

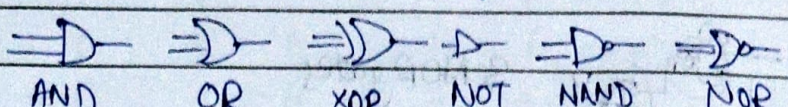
$\Sigma = A \oplus B \oplus C_{in}$

$C_0 = AB + AC_{in} + BC_{in}$



Logic Design

↳ ckts are designed using gates



Gates

< 10 gates → SSI

10-100 " → MSI

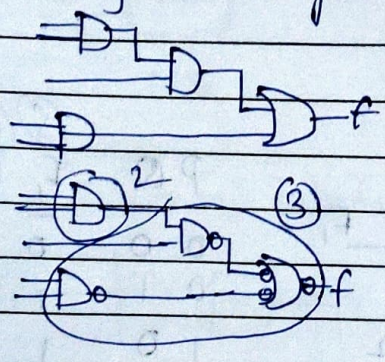
100-1000 " → LSI

>1000 " → VLSI

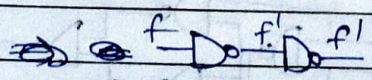
NAND-NAND ⇒ AND-OR

NOR-NOR ⇒ OR-AND

Q Identify min no of NAND gates

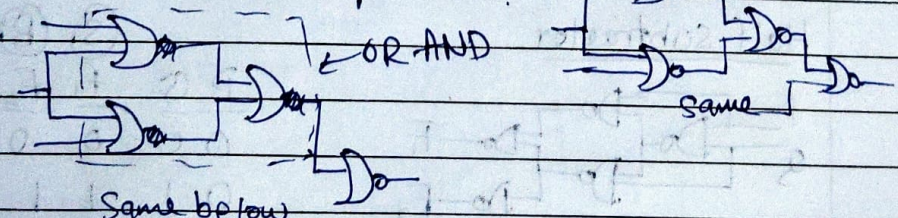


⇒



3+2 = 5

Q What does this example mean?

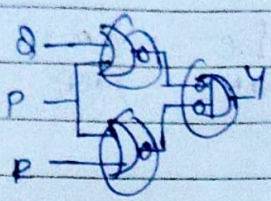


$\overline{(B+A\bar{C}).(C+\bar{A}B)}$

$= B(\bar{A}+\bar{C}).C(\bar{A}+\bar{B}) = B\bar{A}C + 0 = B\bar{A}C$

Q Find no of 2i/p NOR gates for $f = P + QR$

Ans $P + QR = (P+Q)(P+R)$

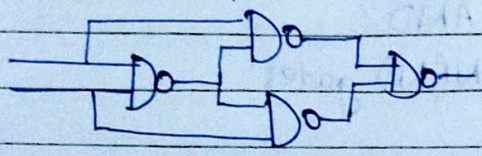


3 NOR gates

Q Find min NOR gates for $f = x + x\bar{y} + x\bar{y}z$

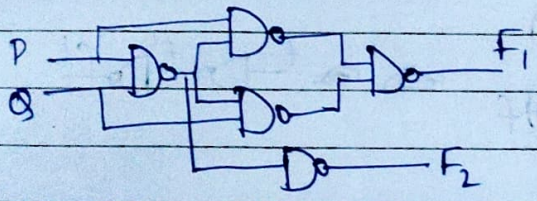
Ans $= x(1 + \bar{y} + \bar{y}z) = x$ 0 NOR gates

Q What is this?



Ans XOR gate

Half Adder

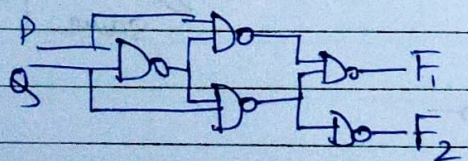


$F_1 = P \oplus Q$

$F_2 = PQ$

P	Q	F_1 (S)	F_2 (C)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half subtractor

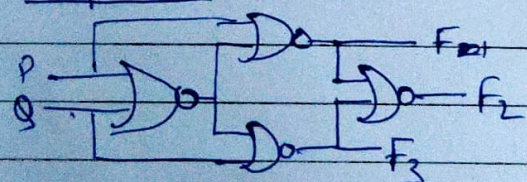


$F_1 = P \oplus Q$

$F_2 = \bar{P}Q$

P	Q	f_1 (S)	F_2 (B)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Comparator



$F_2 = P'Q' + PQ$

$F_1 = P'Q$

$F_3 = PQ'$

P	Q	F_2	F_1	F_3
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	1	0	0

$F_2 \Rightarrow P=Q$

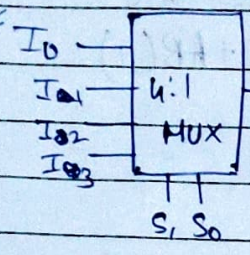
$F_1 \Rightarrow P<Q$

$F_3 \Rightarrow P>Q$

Multiplexer (MUX)

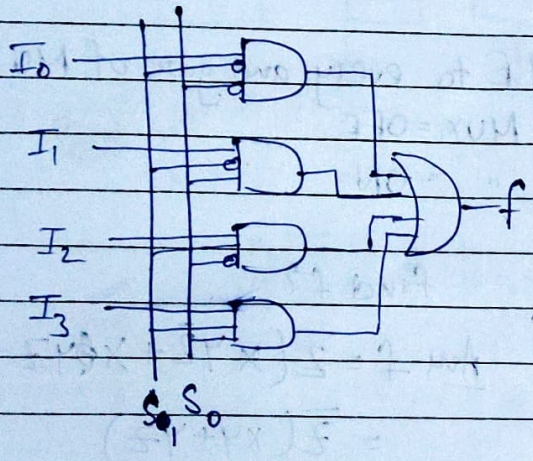
- Switch that connects one set of n inputs to output.
- it can't change logic, it only connects i/p & o/p.
- it is funtionally complete

4:1 MUX



S_1	S_0	o/p
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$$f = \bar{S_1}\bar{S_0}I_0 + \bar{S_1}S_0I_1 + S_1\bar{S_0}I_2 + S_1S_0I_3$$

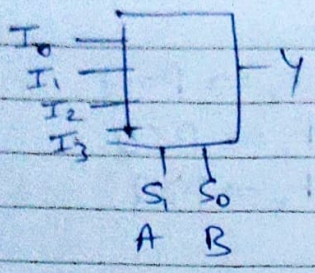


Ckt diagram

Q Prove MUX is fine complete (2:1)

Ans

AND	OR	NOT
$A_0 = B$	$A_0 = B$	$A_0 = A$
$I_0 = 0$	$I_1 = A$	$I_1 = 1$
$I_1 = A$	$I_2 = 1$	$I_2 = 0$



$f = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$

$g(A,B) = \bar{A}\bar{B} + AB$

$= A'B'(1) + A'B(0) + AB'(0) + AB(1)$

Keep (I_0, I_1, I_2, I_3) as $(1,0,0,1)$

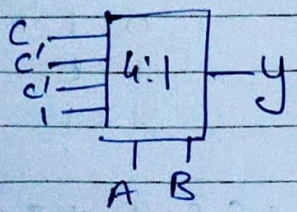
Q $g = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$

$= \bar{A}\bar{B}(C) + \bar{A}B(1) + A\bar{B}(0) + AB(C)$

Q $f(A,B,C) = \Sigma(1,2,4,6,7)$

Ans $= \bar{A}B\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + AB\bar{C} + ABC$

$= \bar{A}B(C) + \bar{A}B(\bar{C}) + A\bar{B}(\bar{C}) + AB(1)$



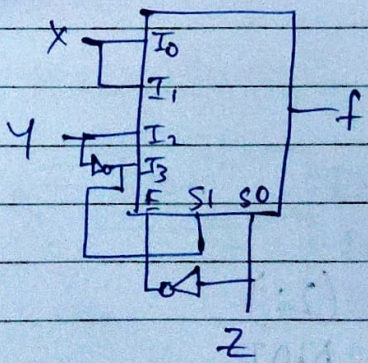
Enable

Add a wire i/p of E to every and gate of MUX

When E=0, MUX = OFF

E=1, " = ON

Q



find f?

Ans $f = \bar{Z}(XY\bar{Z} + X\bar{Y}Z + X\bar{Y} + \bar{Y}\bar{Y}Z)$

$= \bar{Z}(XY + \bar{Y}Z)$

$= XY\bar{Z}$

Q $F = \Sigma(2,3,5,6,7)$

a) $S_1 = B$, $S_0 = C$

b) $S_1 = C$, $S_0 = B$

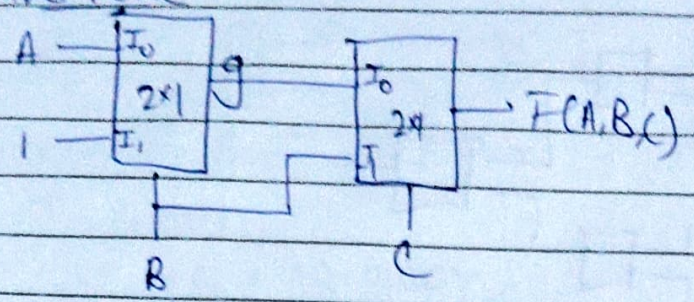
Ans $f_1 = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + AB\bar{C} + AB\bar{C}$

$= (0)\bar{B}\bar{C} + (A)\bar{B}C + (1)B\bar{C} + (A)BC$

f_2 = reverse (✓)

Q/ F(A,B,C) = Σ(2,3,6,7)

Q find func



$q = B'A + B \cdot A = A\bar{B} + AB$

$F = \bar{C} I_0 + C I_1$

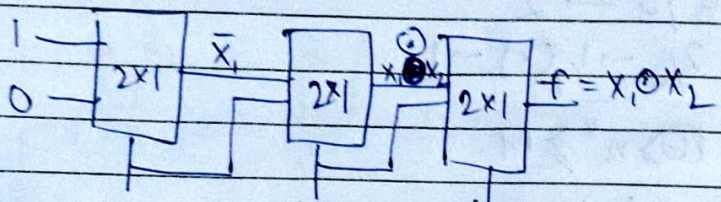
$= \bar{C}(A\bar{B} + B) + CB$

$= A\bar{B}\bar{C} + B\bar{C} + CB$

$= A\bar{B}\bar{C} + AB\bar{C} + \bar{A}BC + ABC + A\bar{B}C$

$= \Sigma(2,3,4,6,7)$

Q



$x_1 \cdot 0 + \bar{x}_1 \cdot 1$

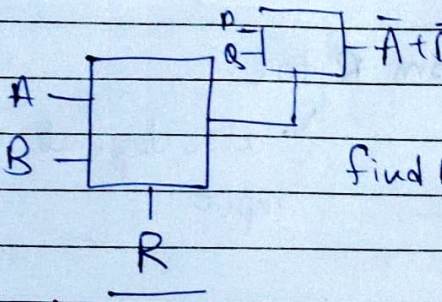
$\bar{x}_2 \bar{x}_1 + x_1 x_2 = x_1 \oplus x_2$

$\bar{x}_1(\bar{x}_2 \bar{x}_1 + x_2 x_1) + x_1 x_2$

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

$x_1 \odot x_2$

Q



find P, Q

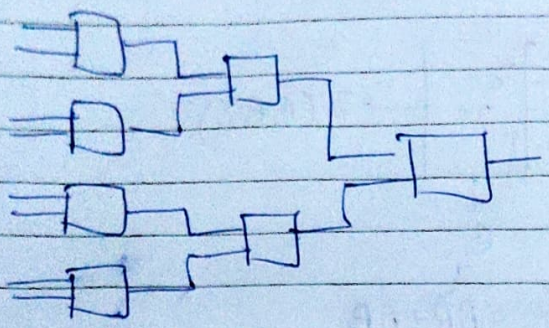
$(\bar{R}A + RB)P + (\bar{R}A + RB)Q$

$= (R + \bar{A})(\bar{R} + \bar{B})P + A\bar{R}Q + BRQ$

$= R\bar{B}P + \bar{A}\bar{R}P + \bar{A}\bar{B}P + A\bar{R}Q + BRQ$

Solve using options

2:1 → 8:1



for m×1 from k×1
keep on doing m/k till you get 1.

eg 32×1 using 4:1

Ans 32/4 → 8 8+2+1=11
8/4 → 2
2/4 → 1/2 (x) → 1

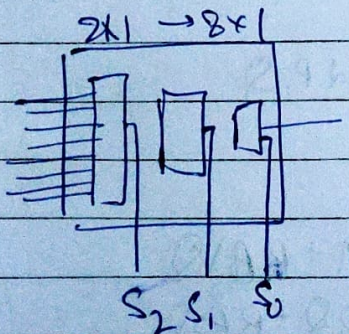
So, $n^k \geq M$

$k \geq \log_n M$

$D = \sum_{k=1}^{\log_n^m} \frac{M}{N_k}$

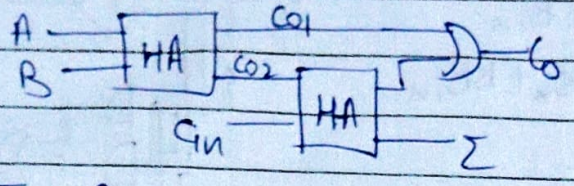
for m×1 using n×1

Use select lines from R to L.



↳ also depends on type of input

Full adder using half adder



$\Sigma = A \oplus B \oplus C_{in}$

$C_0 = C_{01} + C_{02} = AB + (A \oplus B) C_{in}$

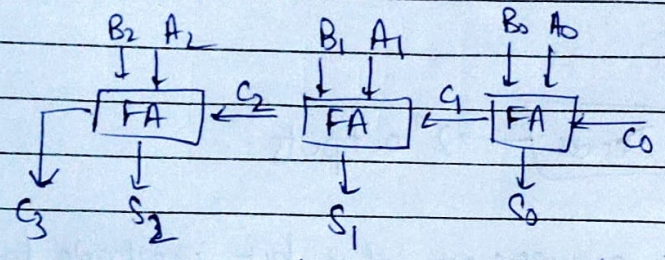
$= AB(1 + C_{in}) + A\bar{B}C_{in} + \bar{A}BC_{in}$

$= AB + AC_{in}(B + \bar{B}) + BC_{in}(A + \bar{A})$

$= AB + AC_{in} + BC_{in}$.

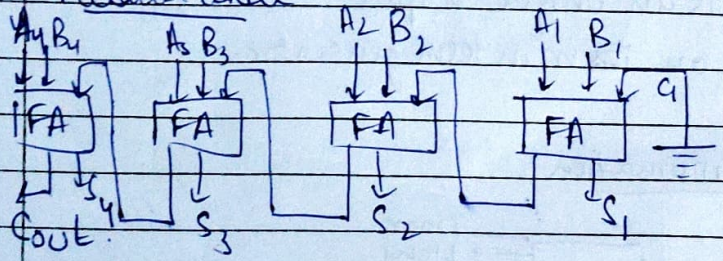
Binary parallel adder

- adds n bit
- has n full adders



ripple carry adder!

Parallel adder



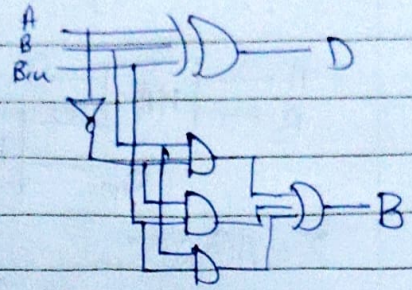
Subtraction (2's Complement)

- add a not gate to all B inputs.

Full Subtractor

$D = A \oplus B \oplus B_{in}$

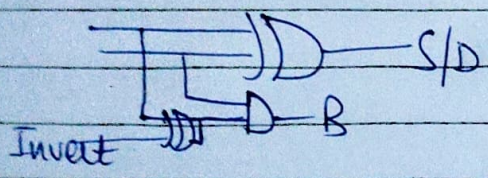
$B = A'B + A'B_{in} + BB_{in}$



Full Subtractor using Half Subtractor

~~Same as adder~~

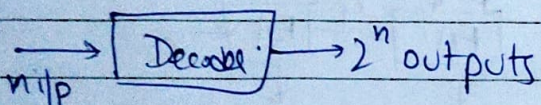
Add an invert as a switch from adder to sub.



if invert=0 ⇒ add

" " =1 ⇒ sub

Decoder

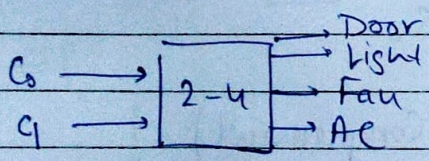


Decoding: conversion of n bit input code to m bit output code with $n \leq n \leq 2^n$ s.t each code is an unique output

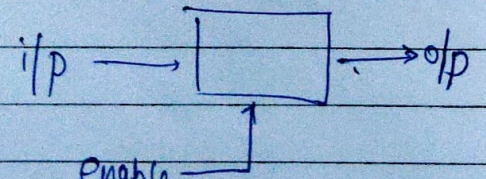
→ It is an ~~m~~ minterm generator.

Use

Home automation

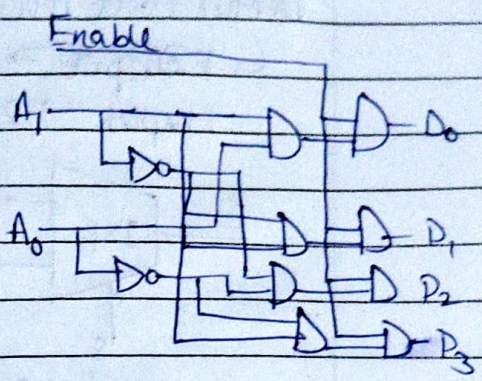


Decoder can have a enable



2-4 decoder (with enable)

↳ 2 inp (A_1, A_0)
↳ 2^2 o/p (D_0, - -, D_3)



Truth Table

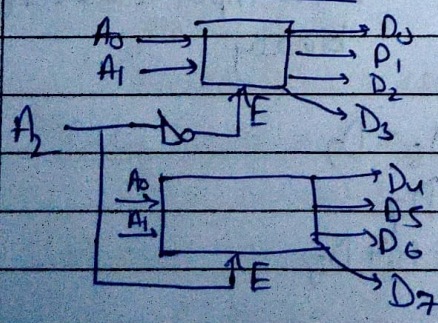
EN	A_1	A_0	D_0	D_1	D_2	D_3
0	x	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	1
1	1	0	0	0	1	0
1	1	1	0	0	0	1

3-8 Decoder

A_2	A_1	A_0	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	✓							
0	0	1		✓						
0	1	0			✓					
0	1	1				✓				
1	0	0					✓			
1	0	1						✓		
1	1	0							✓	
1	1	1								✓

Simple logic digram like earlier.

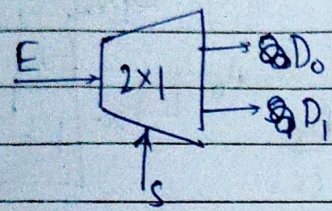
3-8 using 2-4



DeMux (does reverse of MUX)

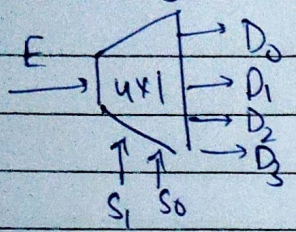
↳ ~~P~~output →

1 input → 2^n outputs



$D_0 = \bar{S}E$

$D_1 = SE$



$D_0 = S_1' S_0' E$

$D_1 = S_1' S_0 E$

$D_2 = S_1 S_0' E$

$D_3 = S_1 S_0 E$

Demux v/s Decoder

→ 1×4 demux is just a 2×4 decoder with enable

↳ S_1, S_0 ⇒ decoder input

↳ E ⇒ decoder enable

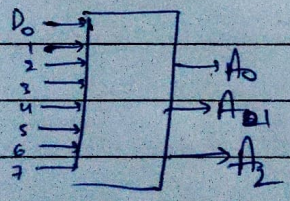
DEMUX ⇒ Decoder with Enable

Decoder generates all minterms

↳ so we can write funcs with it

Encoder

Encoding - opposite of decoding - conversion of m bit i/p to n-bit o/p code with $n \leq m \leq 2^n$ s.t each valid code word produces a unique output code.



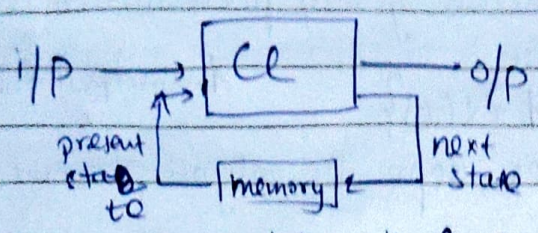
each input is a unique binary

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_4 + D_5 + D_6 + D_7$

$A_2 = D_2 + D_3 + D_6 + D_7$

Sequential Circuit



→ it is a comb. ckt with feedback through memory
↳ stored info at any time is state
→ o/p depends on i/p and previous o/p
→ next state depends on i/p & present state
→ also called sequential machines / state machines
→ info stored in memory elements at particular time defines the state of seq ckt.

Types

① Synchronous
↳ when change of states occurs at discrete instants of time, controlled by a clock

② Asynchronous
↳ when machine is not clocked and the transition is decided by a change in input, and the change might occur at any time interval.
↳ uses time delay devices

Synch
↳ achieved by clock signal of fixed amp
↳ timing device called master-clock generator gives CLK.
↳ clock signals are generated in a periodic train of clock pulses of 0 & 1
↳ also called clocked seq. ckt

Mealy Machine (seq ckt)
↳ a machine in which s(t+1) is determined by s(t) and present i/p x(t).

$$s(t+1) = \delta\{s(t), x(t)\}$$

δ → state transition function.

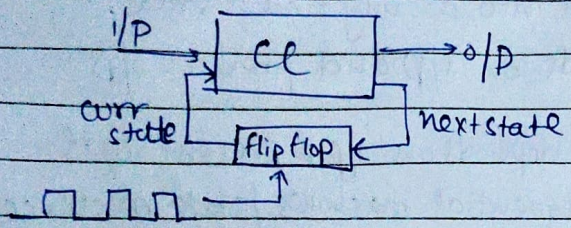
Moore Model

↳ o/p is a function of only present state and independent of external output.

$z(t) = \lambda\{s(t)\}$ → output func

↳ eg: counter, depends only on present count

Synch



→ deploys a clock, which determines when computational stuff occurs

→ storage elements used in clocked seq. ckts are called flipflops

↳ they can store 0/1

↳ a ckt may have many flipflops

↳ they only update with clock.

Storage Elements

↳ can have 0/1, changes when directed by i/p signal

↳ main diff

↳ no of inputs

↳ how many inputs affecting binary state.

↳ types

↳ latches

↳ flipflops

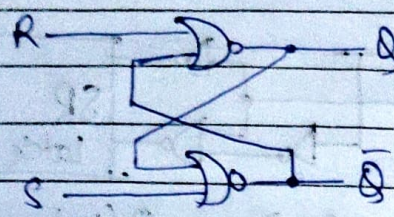
→ latches are used in ~~sync~~ asynch

→ Flipflops are latches with clocks.

Latches

Is built with nand, nor, not

SR Latch (Set n Reset)



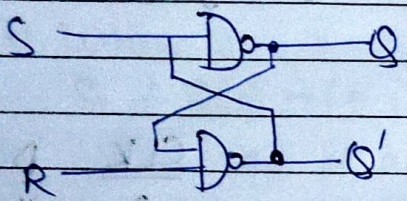
S	R	Q	$\bar{Q}$	
1	0	1	0	Set
0	0	1	0	Set
0	1	0	1	Reset
0	0	0	1	Reset
1	1	0	0	UD

$Q=1 \Rightarrow$ Set

$Q=0 \Rightarrow$ Reset

if $S=R=1$, $Q=0=Q'$, which is impossible

S'R' latch



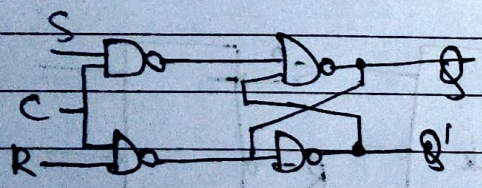
S'	R	Q	Q'	
0	1	1	0	Set
1	1	1	0	Set
1	0	0	1	Reset
1	1	0	1	Reset
0	0	1	1	UD

$Q=0 \Rightarrow$ set

$Q=1 \Rightarrow$ Reset

if $Q=R=0$, its undefined

SR Latch with Clock



C	S	R	Q	
0	x	x		No change
1	0	0		No change
1	0	1		Q=0 ⇒ reset
1	1	0		Q=1 ⇒ set
1	1	1		UD

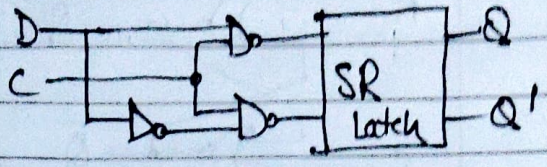
Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	x
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	x

So, $Q(t+1) = Q(t)R' + S$

To eliminate the UD state, we introduce D latches

D Latch

- ensure S and R are never together.
- add inverter
- one input
  - D → S
  - D' → R



C = 0 ⇒ same

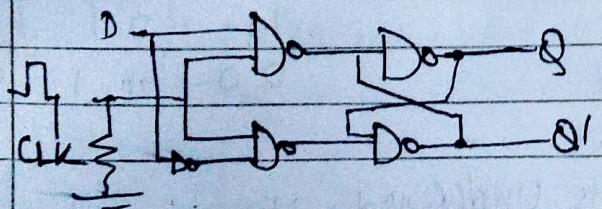
C = 1 ⇒ Transparent

C	D	Q(t+1)
0	X	No change
1	0	Q = 0 → Reset
1	1	Q = 1 ⇒ set

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

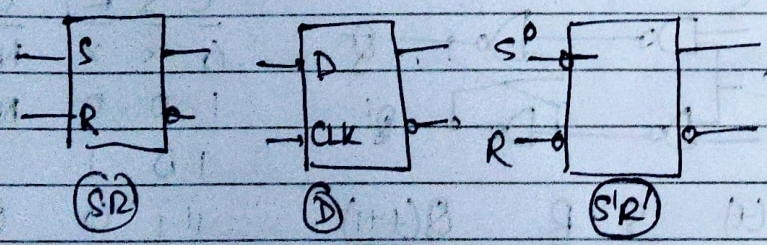
So, $Q(t+1) = D$

Edge triggered D latch



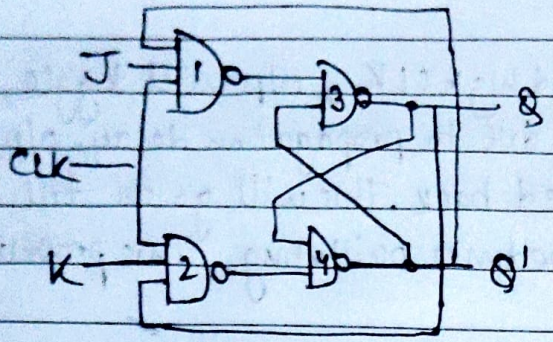
CLK	D	Q
0	X	NC
1	X	NC
↓	X	NC
↑	0	0
↑	1	1

Graphic Symbol



JK latch

- more versatile
- more functions
- universal flipflop
- storage of binary data
- J/K → control inputs, CLK → Clock input



CLK	J	K	Q	Q'
0	x	x	–	–
0	x	x	–	–
1	0	0	NC	NC
1	0	1	0	1 (R)
1	1	0	1	0 (S)
1	1	1	Q'(t)	Toggle

① When J=K=0, both NAND gates are deactivated, So ckt is inactive. Output is just the previous state

② When J=0, K=1, ① is active, ② is inactive hence Latch can't be set. It is in reset.
So, if Q(t) ⇒ (1,0), it'll reset to (0,1)
else if Q(t) ⇒ (0,1), it'll be same.

③ When J=1, K=0, ② is active, ① is inactive, it'll be set.
So if Q(t) → (0,1) ⇒ it'll set to (1,0)
else it'll be same at (1,0)

④ When J=1, K=1
if Q=0, ④ will be set and Q=0 ⇒ Q=1.
if Q=1, ③ will be reset and Q=1 ⇒ Q=1
Hence this case always complement the earlier state
This property is called toggling.

Q(t)	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

So,
$Q(t+1) = JQ(t)' + K'Q(t)$

Racing

↳ at J=K=1 and high CLK, o/p will toggle, as new o/p are fed. due to propagation delay, o/p toggles again and fed back. This will go on till CLK=0. The output becomes oscillating. This property is called racing.

→ can be fixed by edge triggering or using a clock with $t < t_p$.

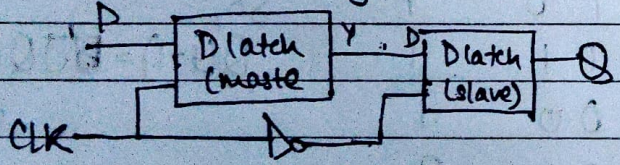
Edge triggered case [JK Master Slave]

CLK	J	K	Q
0	x	x	NC
1	x	x	NC
↓	x	x	NC
↑	0	0	NC
↑	0	1	0 (Reset)
↑	1	0	1 (Set)
↑	1	1	Toggle

Flip Flops

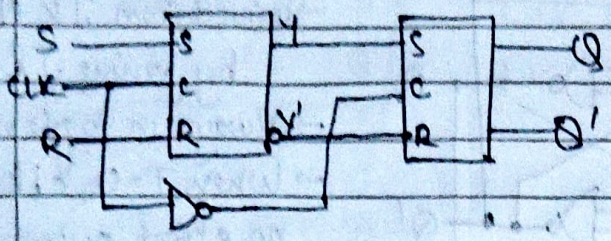
↳ it is a one-bit memory like latch
↳ solves latch transparency
↳ level sensitive [active when clock = 1]
↳ edge triggered (active only when 0→1 or 1→0)

A flipflop can be built using 2 latches in master slave config.



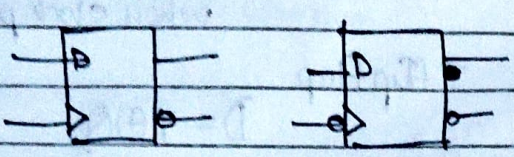
Master → gets external i/p
Slave → " i/p from master.
→ if clk=1, master enabled, and i/p latched
→ if clk=0, " disabled, and slave generates outputs.

SR Flip Flop



C=1, masteractive
C=0, slave active

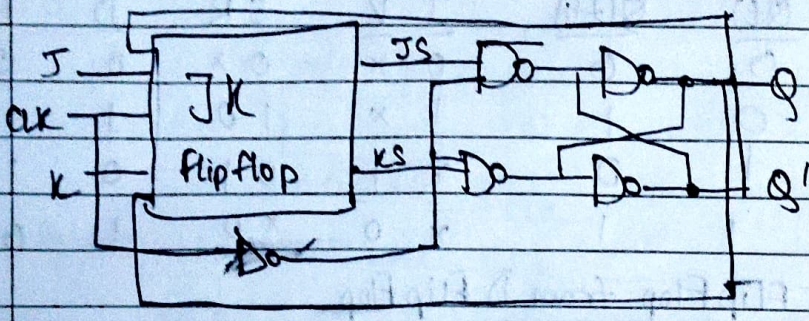
Graphic Symbols



(Positive) (negative)

Clock → allowhead

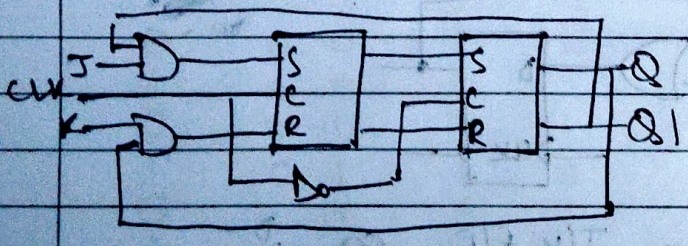
JK Master slave Flip Flop



(1) Master → edge triggered (rising)
Slave → " " (trailing)

(2) Master → positive clock edge Slave → negative clock edge.

JK MS with SR



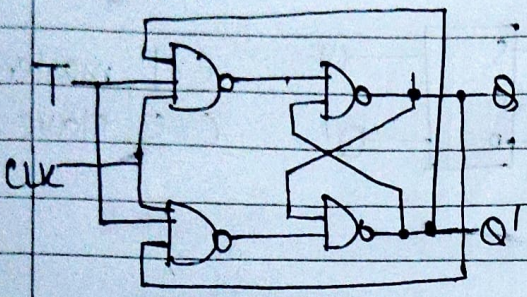
$D = JQ' + K'Q$

J → Sets Flip flop

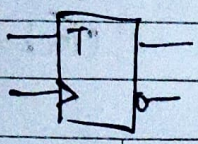
K → resets flip flop.

J-K=1 → toggle

T Flip flop



→ derived from JK flip flop by joining J & K together
→ always in toggle mode
→ when T=0, clk has no effect, output is same
→ when T=1, output toggles when clock pulse is applied



T flip flop

$D = T \oplus Q$

Excitation Table

↳ lists transition states and change.

Memorize

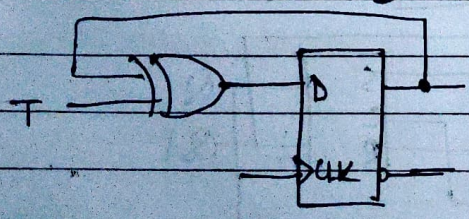
Q(t)	Q(t+1)	J	K	S	R	D	T
0	0	0	x	0	x	0	0
0	1	1	x	1	0	1	1
1	0	x	1	0	1	0	1
1	1	x	0	x	0	1	0

T Flip Flop from D Flip flop

Q(t)	(T)	(D)
0	0	0
0	1	1
1	0	1
1	1	0

Q(t)	T'	T
0	x	1
1	1	x

$D = T \oplus Q$



D F/F from J/K F/F

Q(t)	D	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

(J)

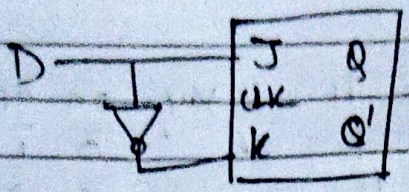
Q	D
0	1
1	x

$J = D$

(K)

Q	D
0	x
1	x

$K = D'$



Ckt Diagram

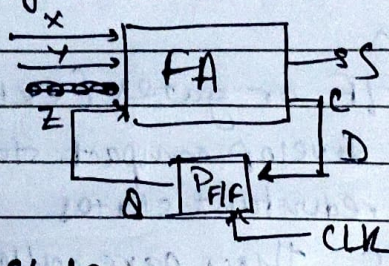
How to solve?

- Obtain state eqn
- Fill state table
- Draw state diagram

State equation

↳ Algebraic expression assigning the next state of F/F in terms of present state and i/p

Ques



Any Next state

$$Q(t+1) = D(t)$$

Sum

$$S(t) = Y(t) \oplus x(t) \oplus Q(t)$$

Carry $\Rightarrow C(t) = x(t)Y(t) + Y(t)Q(t) + x(t)Q(t)$

State Table

↳ Time sequence of i/p o/p and internal states

z(t)	x	y	z(t+1)	S
0	0	0	0	0
1	0	0	0	1
0	0	1	0	1
1	0	1	1	0
0	1	0	0	1
1	1	0	1	0
0	1	1	1	0
1	1	1	1	1

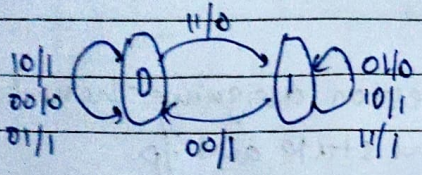
Transition Table

Z	XY = 00	01	10	11	XY = 00	01	10	11
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Notes all transitions from one state to next.

State diagram

↳ graphically directed graphs of the table.



State Reduction

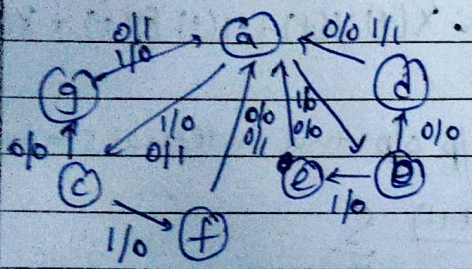
↳ reduce F/F or gates (ckt of min cost)

↳ process to develop compact state diagrams and avoid intro of redundant states.

↳ if $S_x = S_y$, they are called redundant states

↳ if S_x and S_y are equivalent, it means machine started in these states and identical o/p was received.

Eg

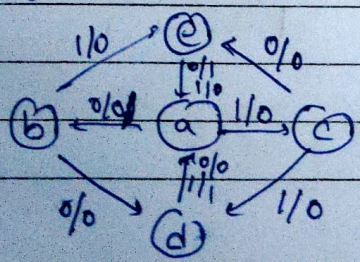


State table

Q(t)	Q(t+1) x=0	Q(t+1) x=1	o/p x=0	o/p x=1
a	b	c	0	0
b	d	e	0	0
c	g	f	0	0
d	a	a	0	1
e	a	a	1	0
f	a	a	0	1
g	a	a	1	0

New table

Q(t)	Q(t+1) x=0	Q(t+1) x=1	o/p x=0	o/p x=1
a	b	c	0	0
b	d	e	0	0
c	e	d	0	0
d	a	a	0	1
e	a	a	1	0

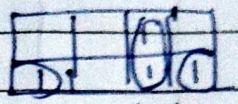


New dgm

Design of D F/F

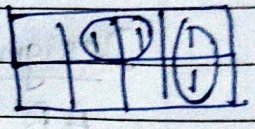
A	B	X	Q(t)	A(t+1)	B(t+1)	D_A	D_B
0	0	0		0	0	0	0
0	0	1		0	1	0	1
0	1	0		0	1	0	1
0	1	1		1	1	1	1
1	0	0		1	0	1	0
1	0	1		0	0	0	0
1	1	0		1	1	1	1
1	1	1		1	0	1	0

D_A

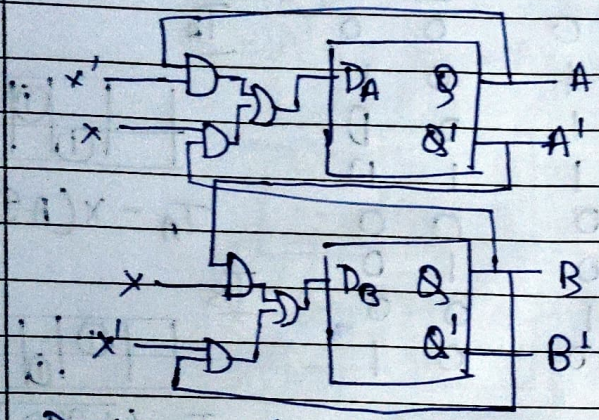


$D_A = Ax' + Bx$

D_B



$D_B = A'x + Bx'$



Ckt Diagram.

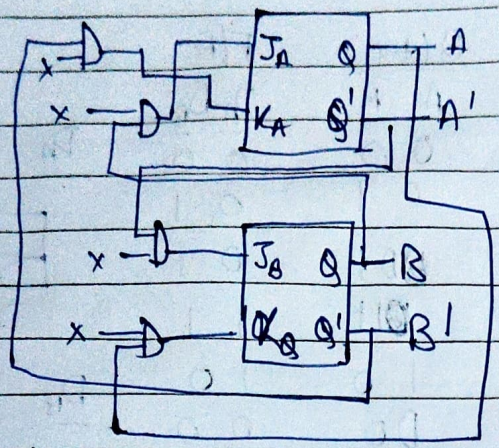
Design of J|K F/F

A	B	X	A(t+1)	B(t+1)	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	1	1	x	x	0
1	0	0	1	0	x	0	0	x
1	0	1	0	0	x	1	0	x
1	1	0	1	1	x	0	x	0
1	1	1	1	0	x	0	x	1

from k maps,

$J_A = Bx$ $\quad J_B = A'x$

$K_A = B'x$ $\quad K_B = Ax$

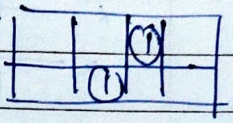


Ckt Diagram

Design of T F/F

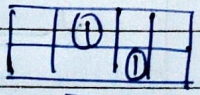
A	B	X	A	B	T_A	T_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	0
0	1	1	1	1	1	0
1	0	0	1	0	0	0
1	0	1	0	0	1	0
1	1	0	1	1	0	0
1	1	1	1	0	0	1

T_A

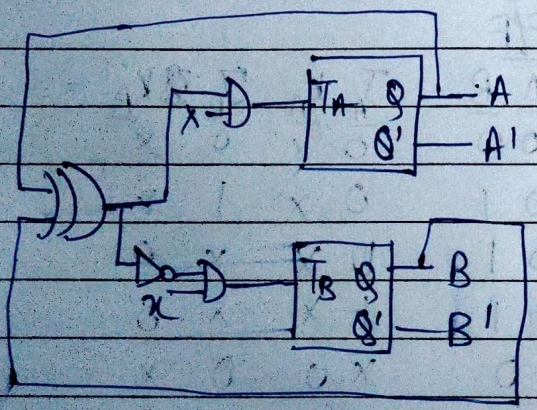


$T_A = x(A \oplus B)$

T_B



$T_B = \overline{(A \oplus B)}\,x$



Ckt Digram

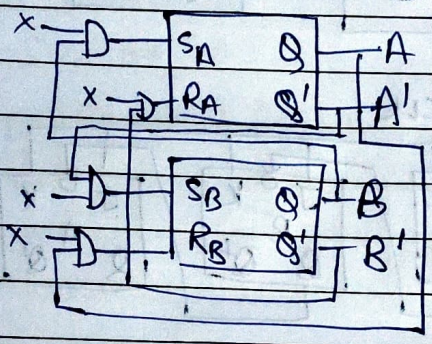
Design of SR F/F

A	B	X	A	B	S_A	R_A	S_B	R_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	0
0	1	0	0	1	0	x	x	0
0	1	1	1	1	1	0	x	0
1	0	0	1	0	x	0	0	x
1	0	1	0	0	0	1	0	x
1	1	0	1	1	x	0	x	0
1	1	1	1	0	x	0	0	1

from k maps,

$S_A = Bx$ $R_A = B'x$

$S_B = A'x$ $R_B = Ax$



circuit diagram

Digital Counter

- used to record number of occurances of input or to generate time sequences to control o/p
- ripple or synchronous
  - not clocked (slow)
  - clocked

→ Ripple counters use TF/F or JK F/F
→ F/F are cascaded in series and asynch
→ one o/p drives the next
→ abbrv. as MOD, it is the total counts/states the counter makes
→ MOD n counters