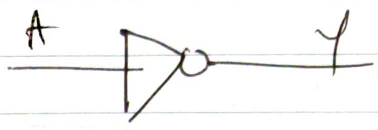
UNIT 3

DIGITAL ELECTRONICS

Logic Gates

1) NOT (inverter)
- one I/P, one O/P.
- complement
- $Y = \overline{A}$, $Y = A'$

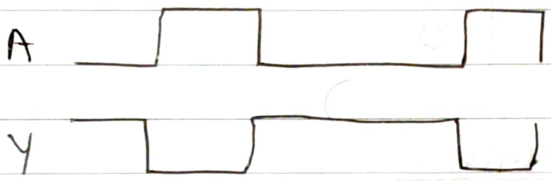


(7404)

Truth Table

A	Y
0	1
1	0

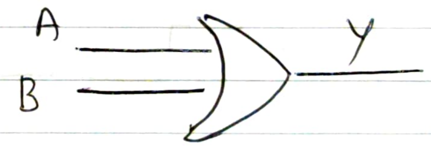
Timing diagram



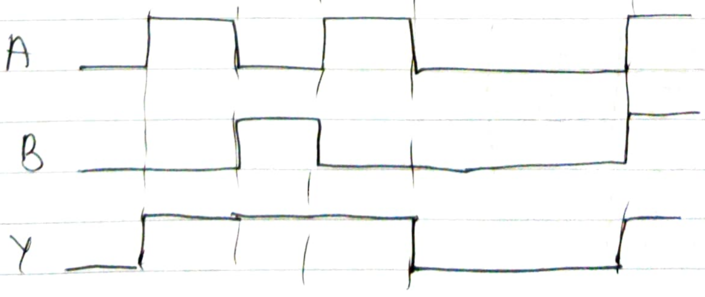
2) OR
- $Y = A + B$

Truth table.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



Timing diagram

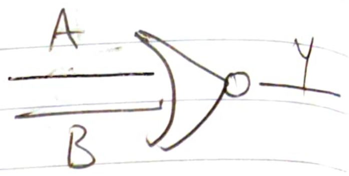


Universal Logic Gate

- Logic gate which can infer any other logic gates among basic logic gates

3) NOR

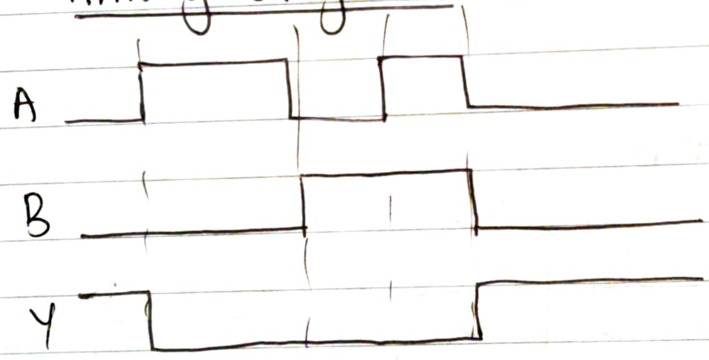
- $Y = \overline{A+B}$



Truth Table

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

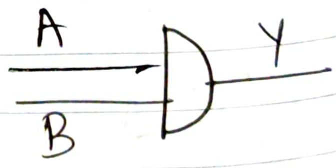
Timing Diagram



4) AND.

$Y = A \cdot B$

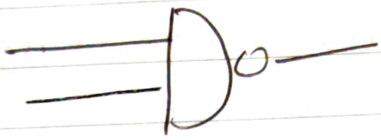
Truth Table



A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

5) NAND

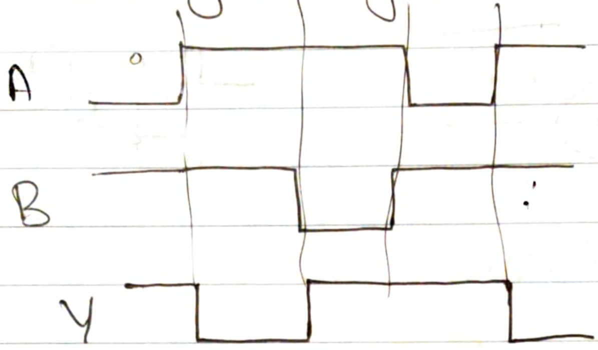
- $Y = \overline{AB}$



Truth Table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Timing Diagram

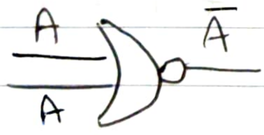


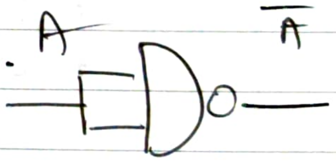
Universality of ~~NAND~~ NAND and NOR

NOR

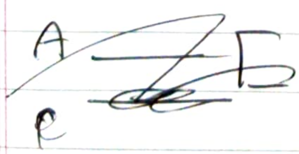
NAND

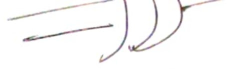
① NOT gate.

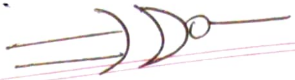




② ~~OR gate~~

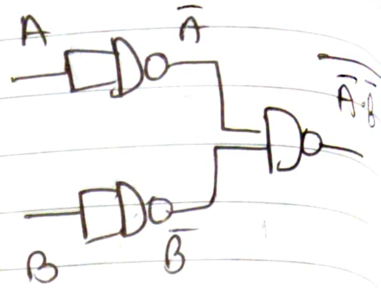


XOR : 

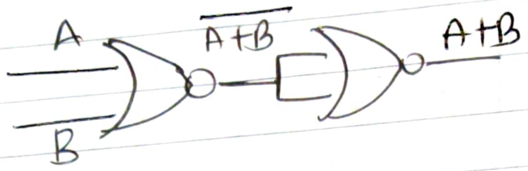
XNOR : 

NAND.

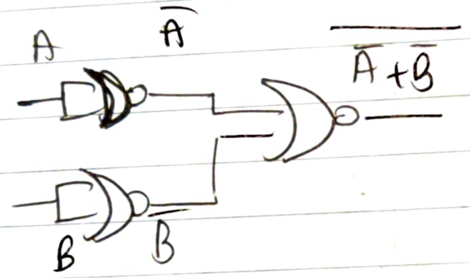
NOR

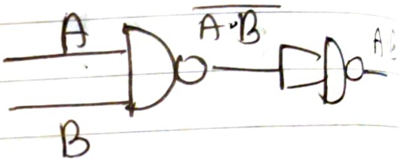


(2). OR gate



(3) AND gate





BOOLEAN LAWS & THEOREMS

(1) Identity Law

$A\cdot 1 = A$ $\quad$ $A+0 = A$

$A\cdot 0 = 0$ $\quad$ $A+1 = 1$

(2) Idempotent Law.

$A+A = A$

$A\cdot A = A$

(3) Complement Law

$A+\bar{A} = 1$

$A\cdot\bar{A} = 0$

④ Involution Law.

$$\overline{\overline{A}} = A$$

⑤ Associative Law

$$(A+B)+C = A+(B+C)$$
$$(A \circ B) \cdot C = A \circ (B \cdot C)$$

⑥ Commutative Law

$$A+B = B+A$$
$$A \cdot B = B \cdot A$$

⑦ Distributive Law

$$A \circ (B+C) = A \cdot B + A \cdot C$$
$$A + (BC) = (A+B)(A+C)$$

⑧ $A + AB = A$

$A + \overline{A}B = A + B$

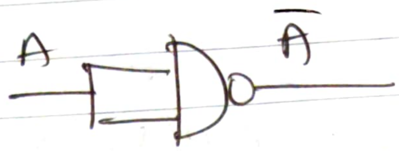
⑨ De Morgan's Law

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$
$$\overline{A + B} = \overline{A} \circ \overline{B}$$

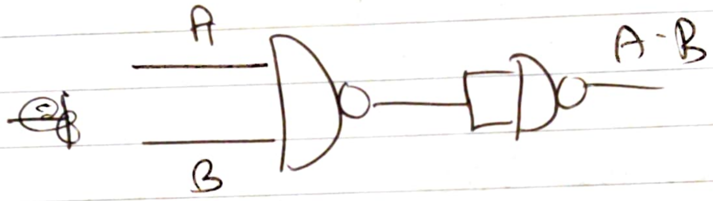
NAND Realisation

~~⇒ NAND as XOR~~

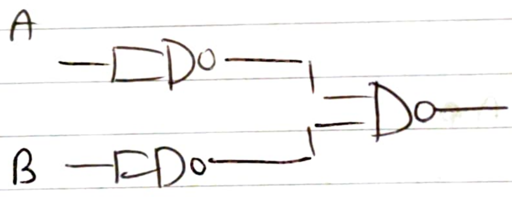
1) NAND as NOT



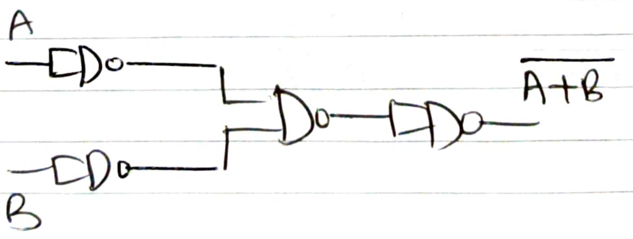
2) NAND as AND



3) NAND as OR

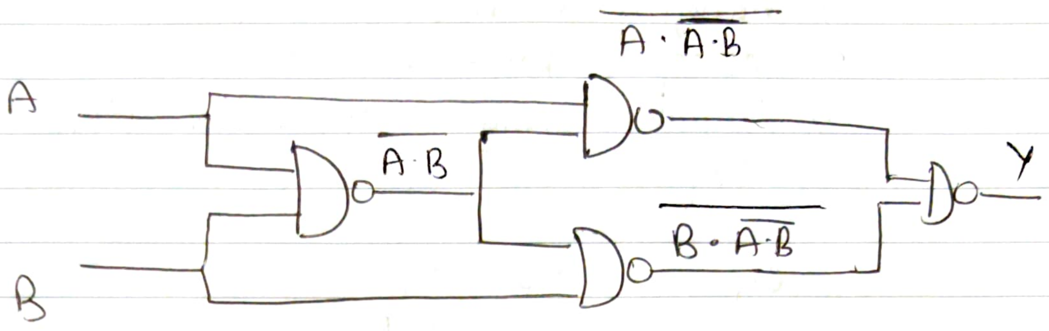


4) NAND as NOR



5) NAND as XOR

$$A \oplus B = A\overline{B} + \overline{A}B$$


A
B
$\overline{A\cdot B}$
$\overline{A\cdot\overline{A\cdot B}}$
$\overline{B\cdot\overline{A\cdot B}}$
Y

$$Y = \overline{\left(\overline{A\cdot\overline{A\cdot B}}\cdot\overline{B\cdot\overline{A\cdot B}}\right)}$$

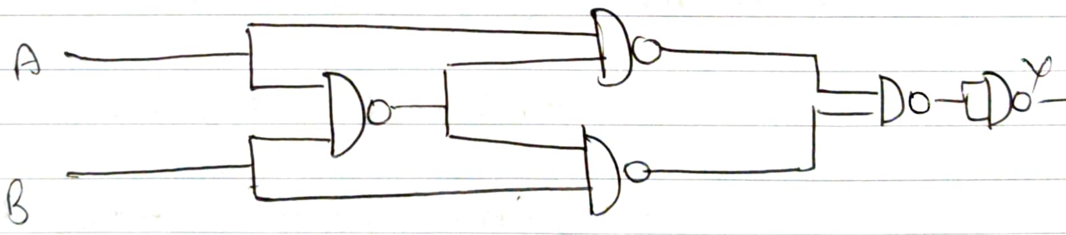
$$= A\cdot\overline{A\cdot B} + B\cdot\overline{A\cdot B}$$

$$= A\cdot(\overline{A}+\overline{B}) + B\cdot(\overline{A}+\overline{B})$$

$$= A\cdot\overline{B} + B\cdot\overline{A}$$

$$Y = A \oplus B$$

6) NAND as XNOR



$$A \odot B = AB + \overline{A}\overline{B}$$

Duality Theorem

Starting from a boolean relation, we can derive another boolean relation by
1) changing each OR to AND
2) changing each AND to OR
3) complementing any 0 or 1 in the expression

eg: $A + 0 = A$
$A \cdot 1 = A$

$A(B+C) = AB + AC$
$A + BC = (A+B)(A+C)$

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

Consensus Theorem

Finds a redundant term which is a consensus of two other terms.

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

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

In the first expression, BC is the consensus term
$\therefore$ if $BC = 1$, $B = 1$ & $C = 1$
$\therefore$ any of the two terms AB and $\bar{A}C$ will become 1 for $A = 0$ or 1.

Conversion of Boolean Expression to NAND circuit

Method 1

Step 1: Draw using basic gates

Step 2: Replace each gate by its NAND equivalent.

Step 3: Eliminate double inversions

Method 2

Step 1: Draw using basic gates

Step 2: Draw bubbles at o/p of each AND gate and add a not.
Draw not gates before OR gate i/ps and add bubbles before i/p.

Step 3: Eliminate double inversions.

Canonical Form / Standard Form

- Product of Sum or Sum of Products
- We have only SOP (minterms)

- A boolean variable can be expressed in either true form or complemented form.
- In standard form, the boolean function will contain all variables in either true form or complemented form

ADDERS

1) Half-Adder

- Circuit that adds 2 single bit numbers A and B.
- Maximum sum = $2_{10} = (10)_2$
  2 in decimal 10 in binary
- Output denoted by 2 bits: carry and sum.
- LSB of O/P : sum
  MSB of O/P : carry

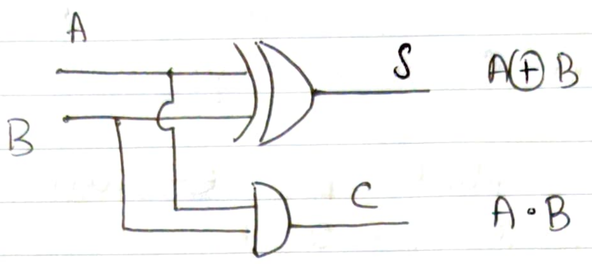
Truth Table

A	B	C	S	in decimal. sum =
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	2

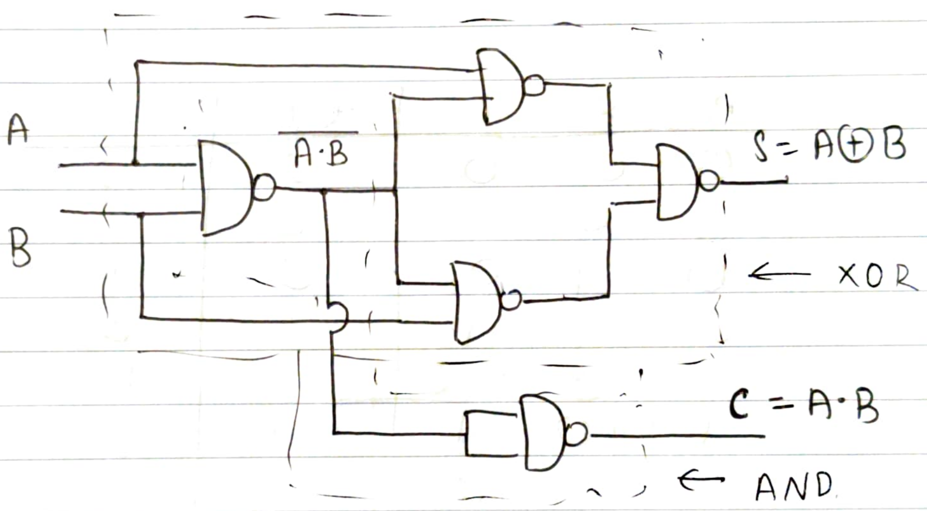
$C = A \cdot B$
$S = A \oplus B = A\bar{B} + \bar{A}B$.

O/P value for sum resembles XOR gate and for carry resembles AND gate.

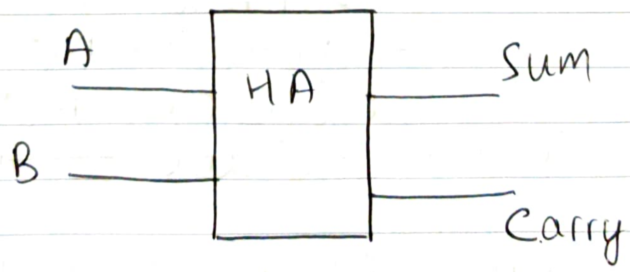
Circuit



Circuit using NAND gates



We can use the HA symbol to show the level of abstraction



2) Full Adder

- Adds 3 bits, where 3^{rd} digit is it carry bit from the previous stage.

eg:

$$\begin{array}{cccccl} & 1 & 1 & 1 & & \leftarrow \text{carry out}\\ & 0 & 1 & 0 & 1 & \\ + & 1 & 1 & 1 & 1 & \\ \hline 1 & 0 & 1 & 0 & 0 & \\ & \uparrow & \uparrow & \uparrow & \uparrow & \\ & & \text{sum o/p.} & & & \end{array}$$

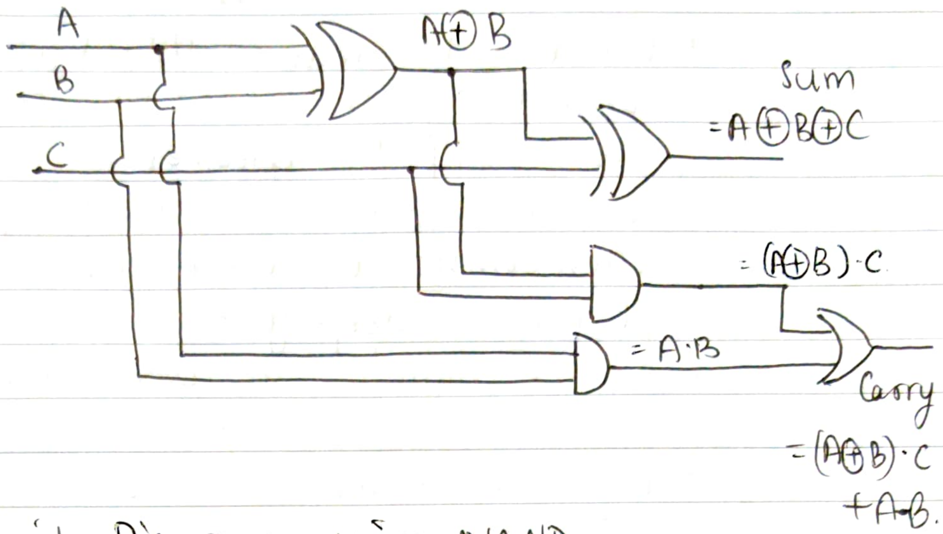
A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Using Sum of Products

$$\text{Sum} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$
$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$
$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C})$$
$$\text{sum} = A \oplus B \oplus C. \longrightarrow (1)$$

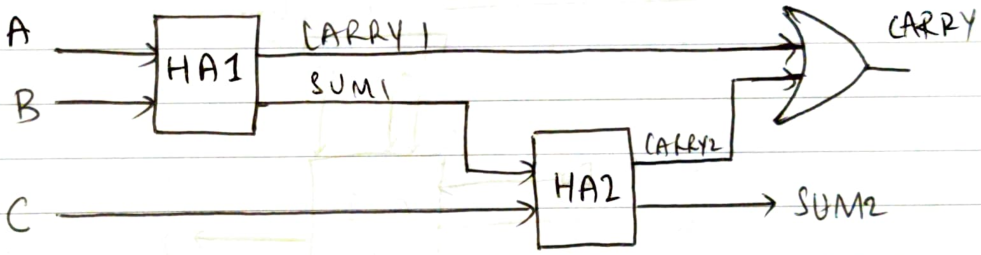
$$\text{Carry} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$
$$= (\bar{A}B + A\bar{B})C + AB$$
$$\text{carry} = (A \oplus B)C + AB \longrightarrow (2).$$

Circuit Diagram

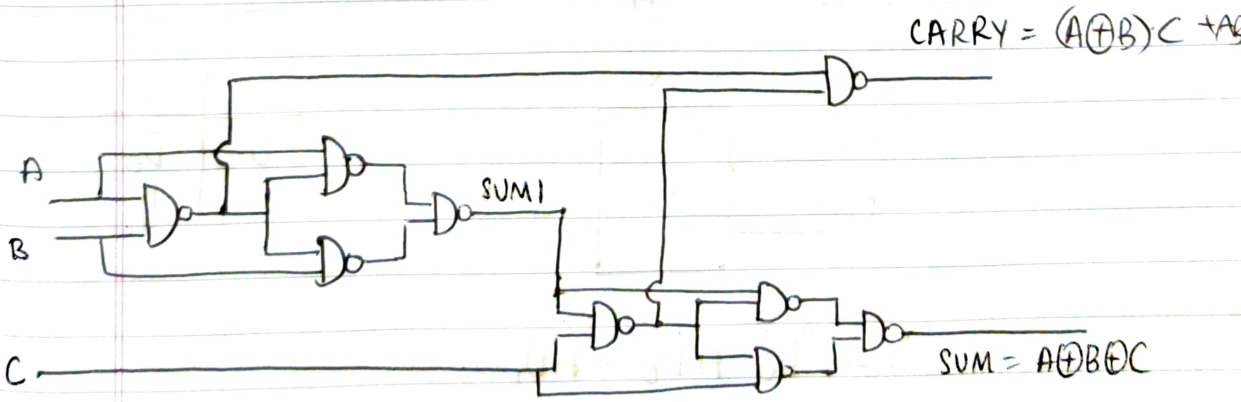


Circuit Diagram using NAND

1) Using HA blocks.



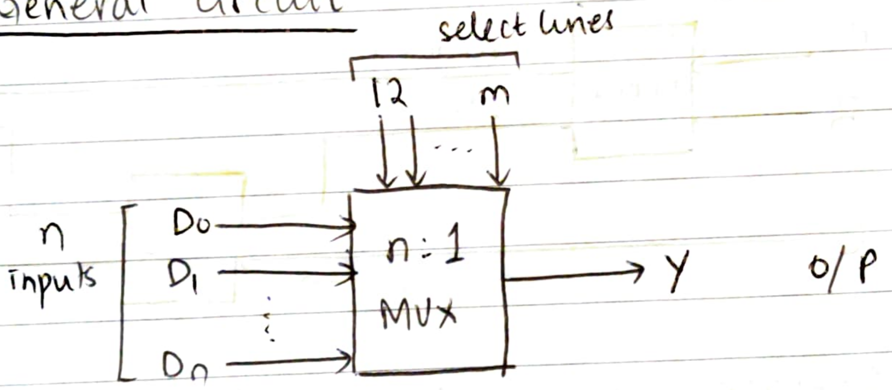
2) Replace with NAND and remove double inversions



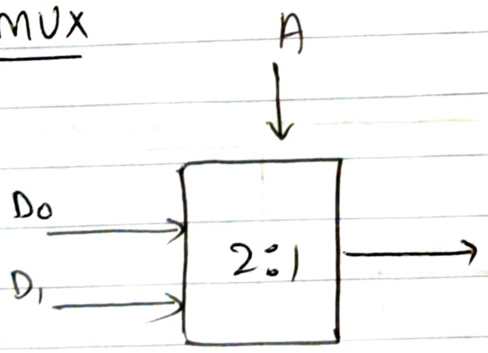
Multiplexer

- Many - to - one (MUX)
- circuit with many I/Ps but one o/p.
- Control signals used to connect o/p from one I/P to another.
- Control signals ⟶ select lines
- Each combo of select lines selects one I/P line to display.
- n I/P signals, m control lines and 1 o
- m select lines can select at most 2^m I/P signals
- $n \le 2^m$

General Circuit



2:1 MUX



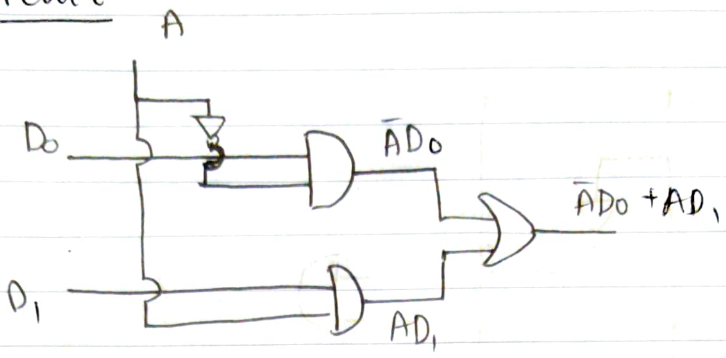
Truth Table

A	Y
0	D_0
1	D_1

$Y =$ ~~D_0~~ $\bar{A}D_0 + AD_1$

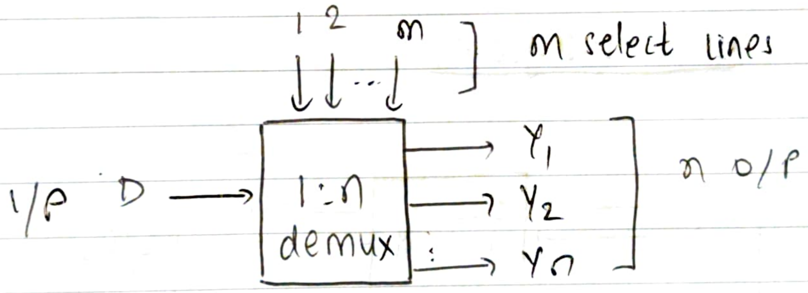
$$\boxed{Y = \bar{A}D_0 + AD_1}$$

Circuit

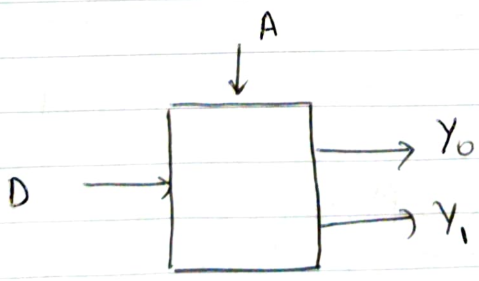


De-Multiplexer

- combinational circuit with 1 I/P and many O/Ps (DEMUX)
- select lines connect I/P to one O/P.
- 1 I/P, n O/P, m select lines
  $2^m \geq n$



1:2 DEMUX

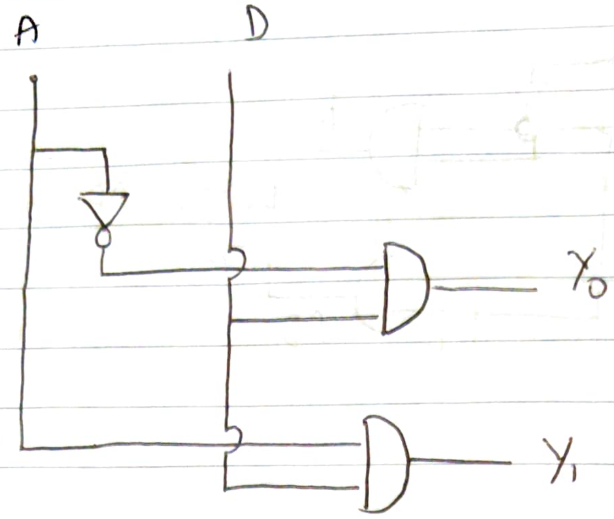


Truth Table

A	Y_0	Y_1
0	D	0
1	0	D

$Y_0 = \bar{A}D$
$Y_1 = AD$

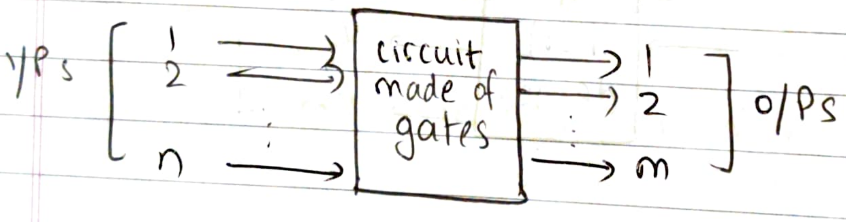
Circuit



Combinational Circuits

- Combines I/P and generates O/P
- Depends only on present value of I/P
- No memory/storage present to store previous outputs

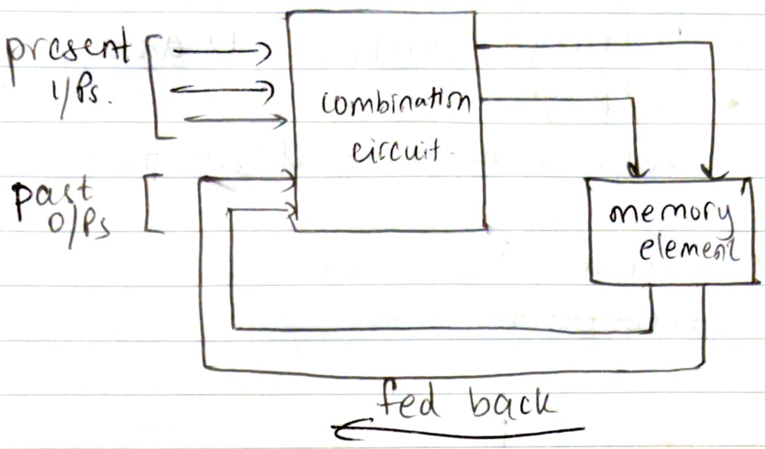
Block Diagram



Sequential Circuits

- System which has sequencing/pattern in O/P
- O/P depends on present I/P and past O/P
- Eg: counter, elevator, traffic light controller
- Storage/memory element required to store past O/P.

Block Diagram



There are 2 types of sequential circuits

1) Asynchronous
- circuit O/P can change at any time
- no clock
- Eg: Latch

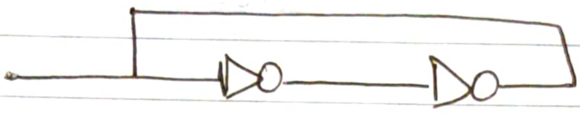
2) Synchronous
- circuit O/P changes only at discrete instants of time
- Acheives synchronisation by using timing signal (clock)
- Eg: flip flop.

- Latches and flipflops are sequential circuit elements which, with the help of Boolean logic, can create memory.
- Storage elements that store binary information (0 or 1)

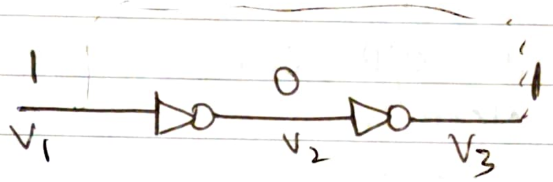
LATCH

- simplest form of memory element
- 2 stable states: 0 and 1
- O/P is set to a stable state until something is done to change it.

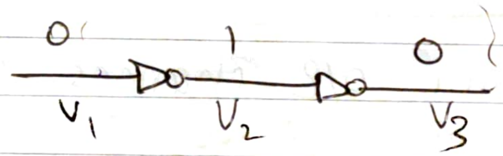
- simplest 2 inverters in series with a feed back.



To store a '1'



To store a '0'

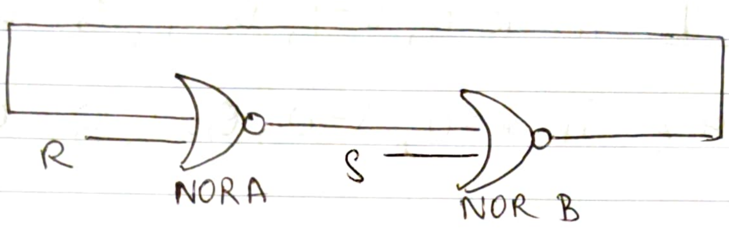


- To store a 1, V_1 is first connected to $V_{cc} = +5V$ and then the feed back line from V_3 is reconnected to V_1 and V_1 is disconnected from V_{cc}.
  This keeps V_3 stable at 1.

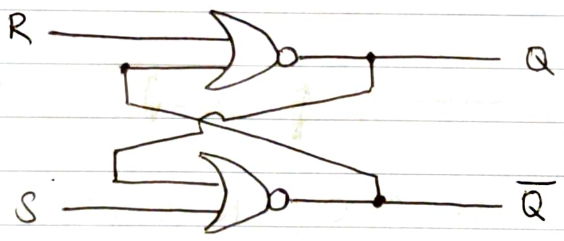
- To store a 0, V_1 is first connected to ground and then the feedback line from o/p is reconnected to V_1 and the ground connection is removed from V_1. This keeps V_3 stable at 0.

More effective circuit for latch

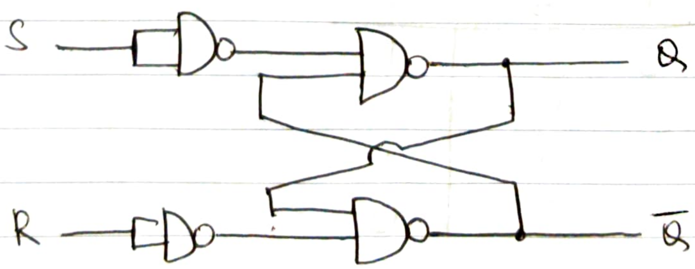
- Replacing inverters with NAND gates or NOR gates
- We have only NAND gates.
- The extra inputs to these gates can be used to switch latch from one stabe state to another.
- Not latch only one I/P.



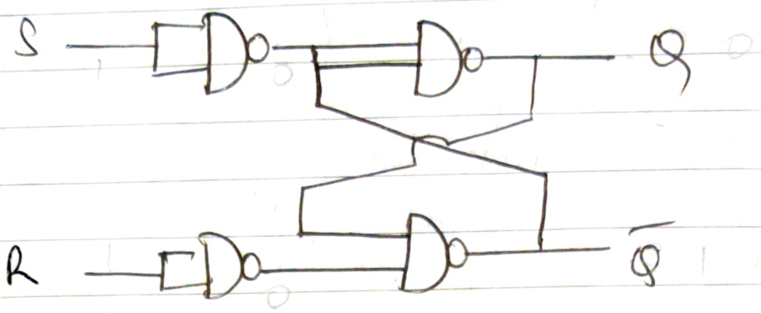
or



USING NAND



Truth Table



Case I if $R=0$, $S=1$, $Q=1$, $\bar{Q}=0$

Case 2 if $R=0$, $S=0$, $Q=1$, $\bar{Q}=0$] memory

Case 3 if $R=1$, $S=0$, $Q=0$, $\bar{Q}=1$

case 4 if $R=0$, $S=0$, $Q=0$, $\bar{Q}=1$] memory

Case 5 if $S=\emptyset$, $R=\emptyset$, $Q=\emptyset$, $\bar{Q}=\emptyset$

if $S=0$, $R=0$, either $Q=0$, $\bar{Q}=1$
$Q=1$, $\bar{Q}=0$
depending on whichever gate is faster.

NAND gate TT

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

Q_t	S	R	Q_{t+1}	$\bar{Q}_{t+1}$	Comments
0	0	0	0	1	no change
0	0	1	0	1	reset
0	1	0	1	0	set.
0	1	1	1	1	*
1	0	0	1	0	no change
1	0	1	0	1	reset
1	1	0	1	0	set
1	1	1	1	1	*

Truth Table

S	R	Q_{t+1}
0	0	Q_t
0	1	0
1	0	1
1	1	*

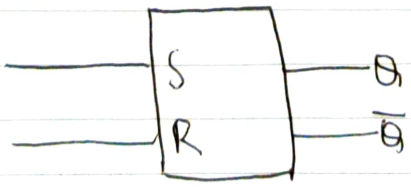
* - forbidden state

S = 1 and R = 1 is ~~not~~ forbidden.
as Q = 1 and $\bar{Q}$ = 1

⇓

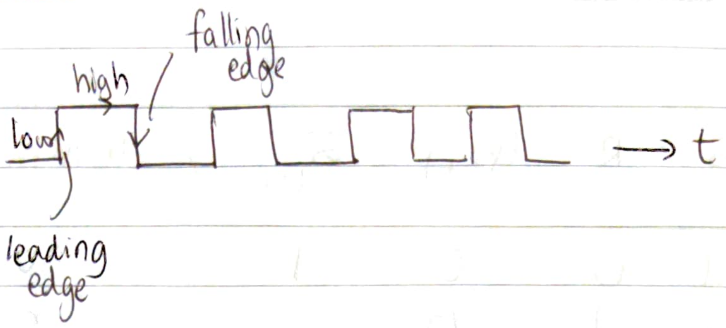
when S=0 and R=0 after this,
o/p is indeterminant

Standard Symbol



IEEE symbol

Clock



- to regulate input
- can be added to circuits to trigger at high, leading edge trigger or falling edge trigger.

Triggering

1) Level triggering
   - change in circuit happens when clock is high

2) Positive edge trigger
   - change in memory element when clock goes from low to high

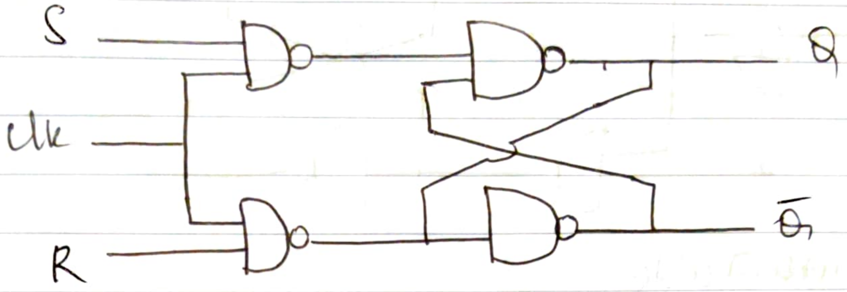
3) Negative edge trigger
   - Change in memory element when clock goes from high to low

To get edge triggering, differentiator circuit.

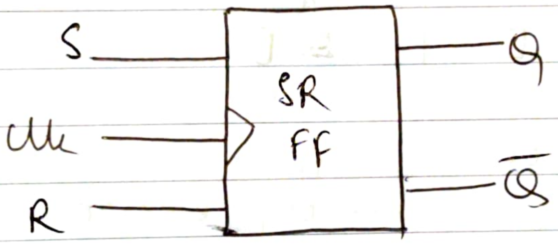
Flip Flop

- latches are level sensitive (level triggering - enable
- Flip flops are sensitive to edge triggering

1) SR Flip Flop



- Set-reset flip flop

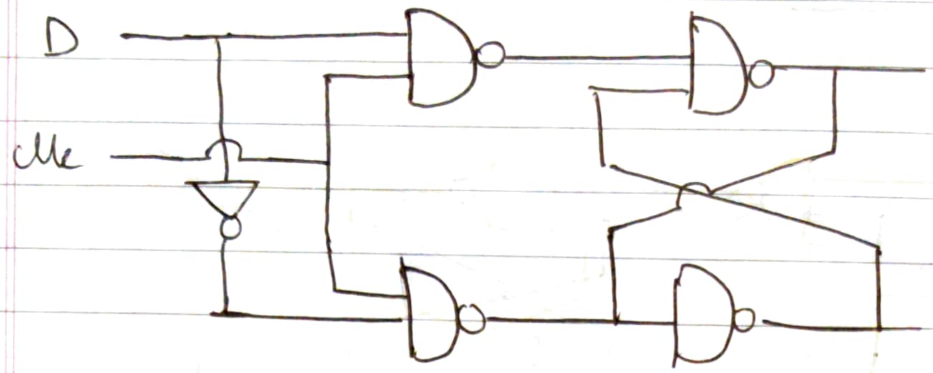


Truth Table

clk	S	R	Q	$\bar{Q}$
0	x	x	Memory	
1	0	0	~~memory~~	
1	0	1	0	1
1	1	0	1	0
1	1	1	Forbidden state.	

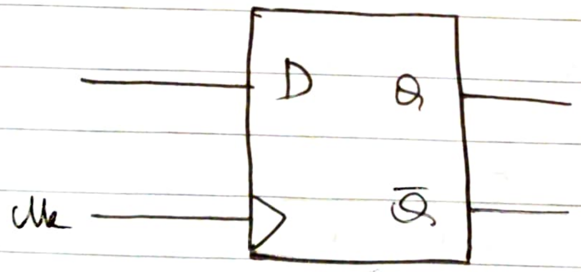
2) D Flip flop

- data flip flop
- can be used to store data
- if clock off, data stored



Truth Table

Clock	D	Q_{t+1}
0	X	Q_t
1	0	0
1	1	1



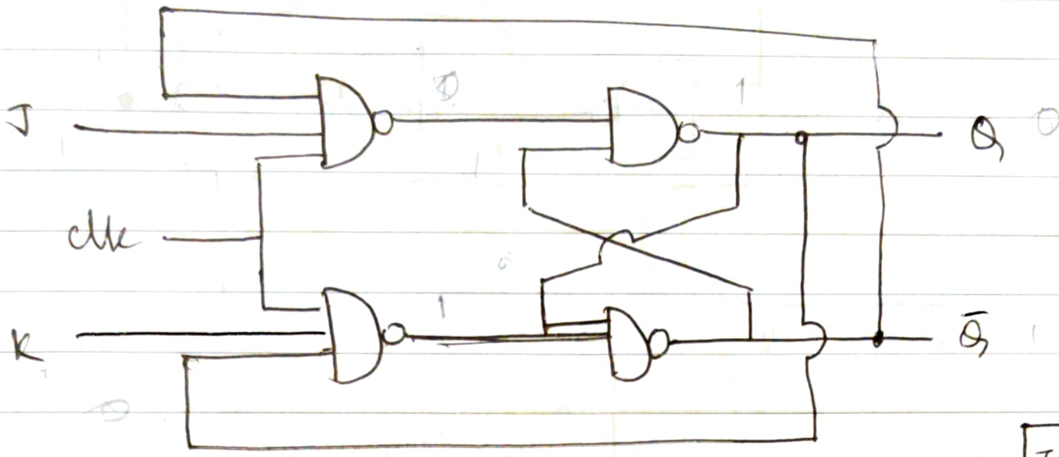
3) ~~JK Flip Flop~~ JK Flip Flop

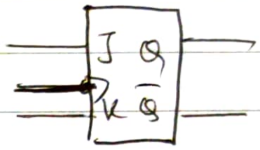
- ~~toggle flip flop~~

- to use 1-1 state
- JK Flip Flop

NAND TT.

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0





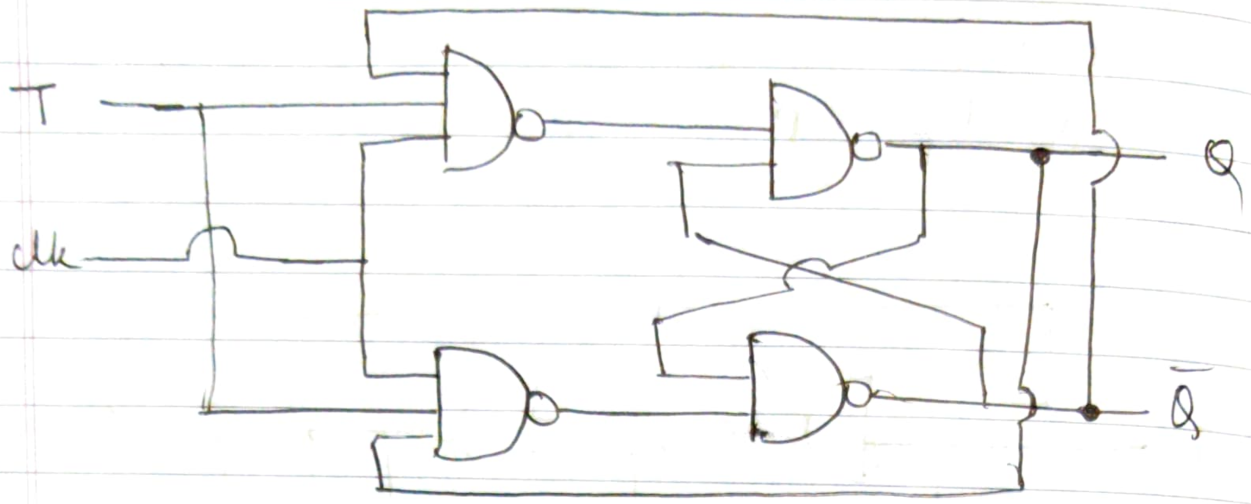
Truth Table

Clk	Q_t	J	K	Q_{t+1}	$\overline{Q_{t+1}}$	Comments
0	1	X	X	1	0	Memory
0	0	X	X	0	1	Memory
1	0	0	0	0	1	~~Memory~~
1	0	0	1	0	1	Reset
1	0	1	0	1	0	Set
1	0	1	1	1,0,1,0...	0,1,0,1...	toggle
1	1	0	0	1	0	Memory
1	1	0	1	0	1	Reset
1	1	1	0	1	0	Set
1	1	1	1	0,1,0,1,..	1,0,1,0	toggle

Racearound condition.

- Avoided using Master-slave flip flop

4) T Flip Flop
- toggle flip flop



Truth Table

clk	T	Q_{n+1}
0	X	Q_n (memory)
1	0	Q_n (Memory)
1	1	$\bar{Q}_n$ (toggling)

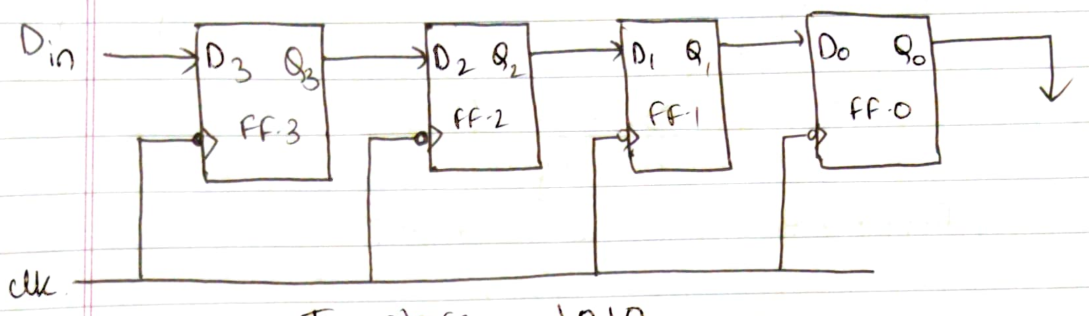
Registers

- n-bit register capable of storing n-bit word.
- Applications: shifters & counters.

- Types of registers
  - SISO
  - SIPO
  - PISO
  - PIPO

Shift Register (SISO)
(4 D FFs)

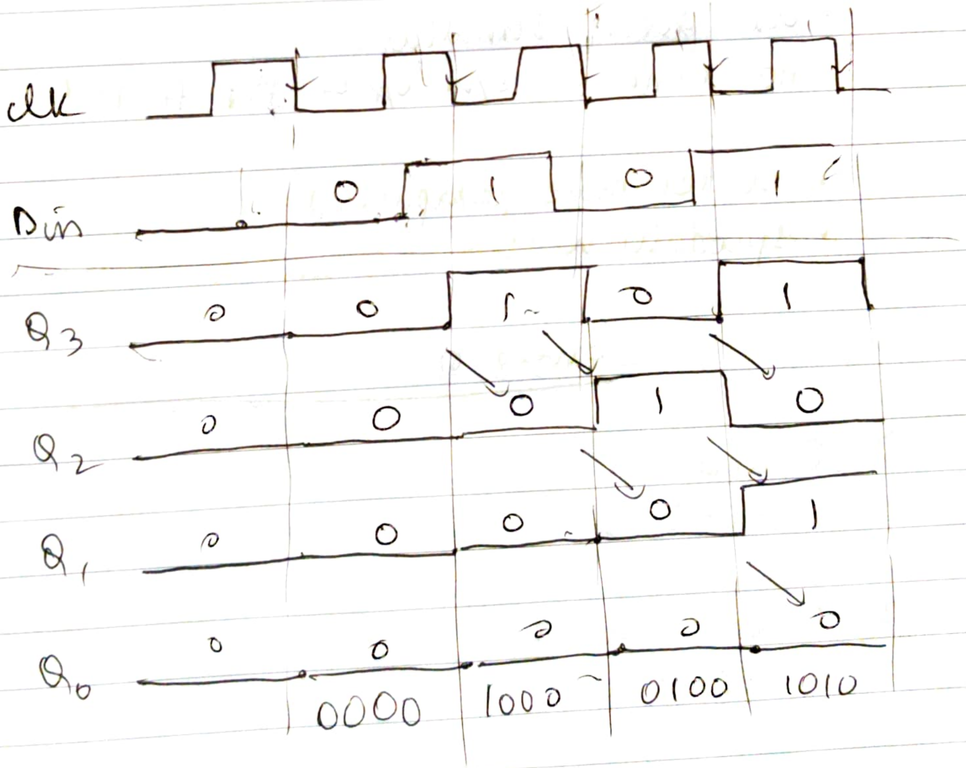
Shift-right mode.



To store 1010

clk	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
↓	0	0	0	0
↓	1	0	0	0
↓	0	1	0	0
↓	1	0	1	0

Timing Diagram



Counters

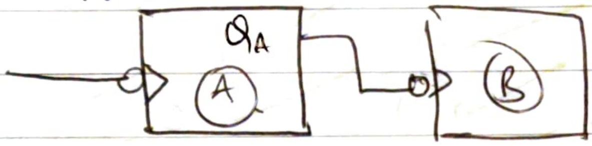
- Counts upto $2^n - 1$ where n = no. of flip flops

Types of Counters

Based on Clock

1) Ripple / Asynchronous counter

- clk applied to first ff and o/p of first ff connected to clk of next ff.
- circuit is more simple for more states
- slow



2) Synchronous counter

- clk given simultaneously
- no connection b/w o/p of first ff and clock of next ff.
- circuit more complicated
- speed high

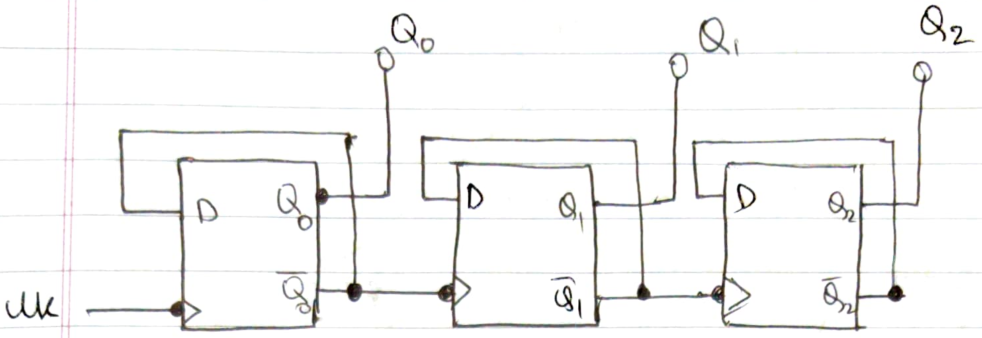
Based on Counting Progression

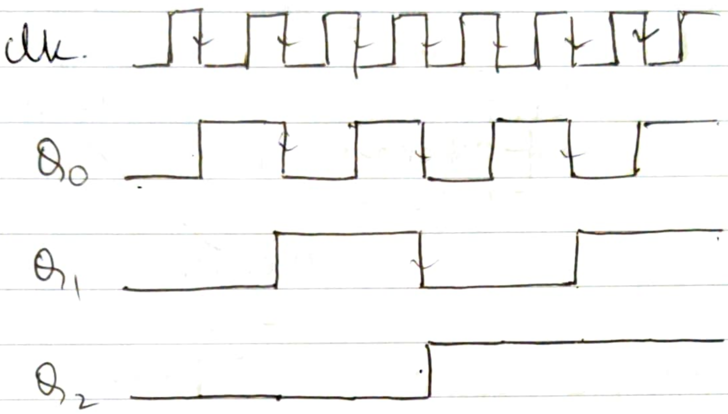
1) UP counters
   0 - 1 - 2 - 3 . . . -

2) DOWN counters
   7 - 6 - 5 - 4 . . .

3) UP/DOWN counters
   combination

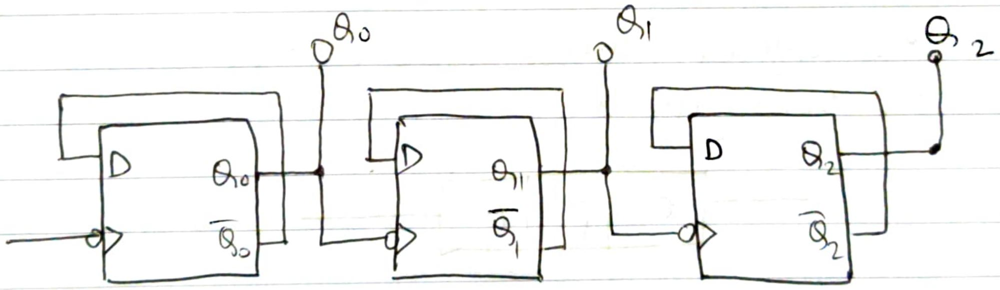
3-bit Asynchronous Up Counter (DFF)





Q2	Q1	Q0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

3-bit Asynchronous Down Counter (DFF)



~~Q~~	Q2	Q1	Q0
~~clk~~	1	1	1
	1	1	0
	1	0	1
	1	0	0
	0	1	1
	0	1	0
	0	0	1
	0	0	0