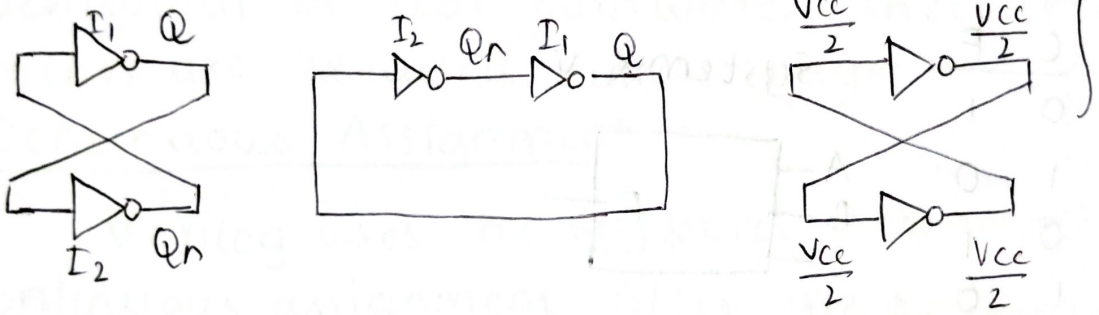
16/9

MODULE: 4

Sequential Logic Storage Devices

Cross coupled inverter pair



- when you give Q=0, it gives the input of inverter I_2 producing an o/p Qn=1 which inturn become the input for I_1 resulting in Q=0. Such a state is called a stable state.
- If you give Q=1, again it will return a state where Q=1, making a system stable. Such a system where there are two stable states ie 0 and 1 is called as a bistable

Meta Stability

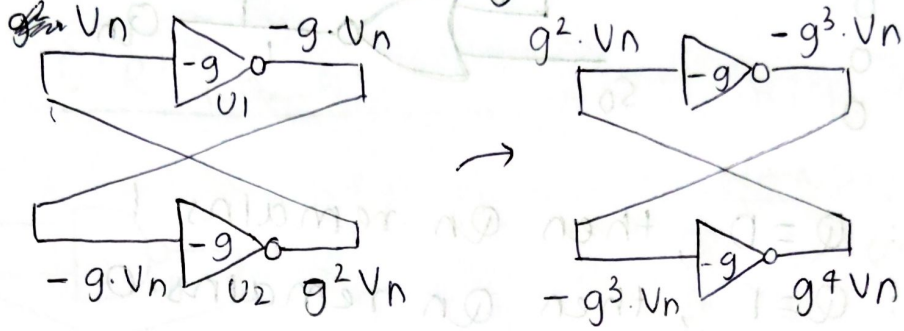
Sometimes a system will be having a voltage range in b/w 0 and 1. Then the system is said to be in metastable state.

Consider a system in metastable state where $Q = \frac{V_{cc}}{2}$

Examining metastability - moving towards 0

Let, $\frac{V_{cc}}{2} = V_n$. We are ~~ad~~ going to add a multiplying factor called ~~gam~~ gain represented by g inorder to push the voltage V_n towards 0.

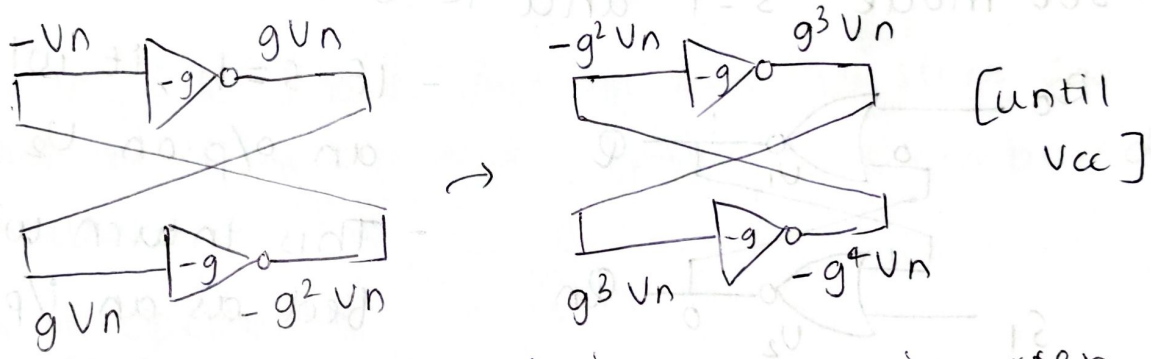
a)



[until GND]

Here the noise is amplified by the inverter with a -ve gain, pushing it slightly towards a logic '0'

b)



[until Vcc]

Here the noise is amplified by the inverter with a ~~re~~ -ve gain, pushing it slightly towards '1'.

18/9

S-R Latch

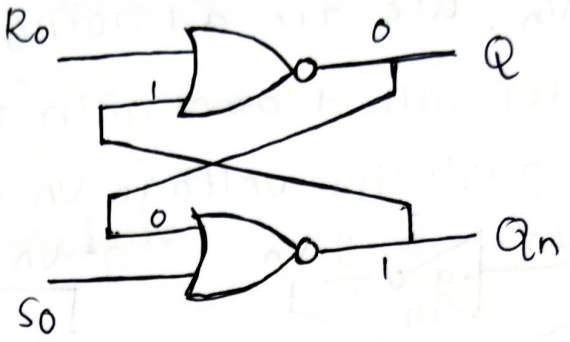
- Here S stands for set and R stands for ~~reason~~ reset.
- Set indicates that the o/p is forced to a logic 1 ie, Q=1 and reset indicates that the o/p is forced into a logic 0 ie, Q=0.

Modes of SR Latch

1. Store : This mode is enable when S=0 and R=0

NOR

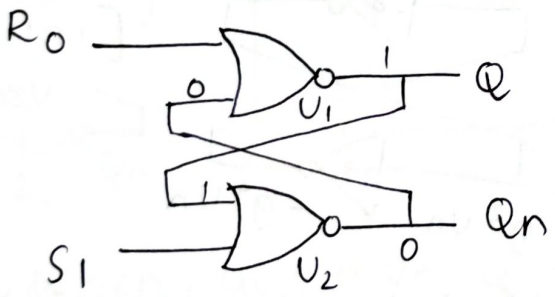
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



case 1 : Q=0 , then Qn remains 1
case 2 : Q=1 , then Qn remains 0

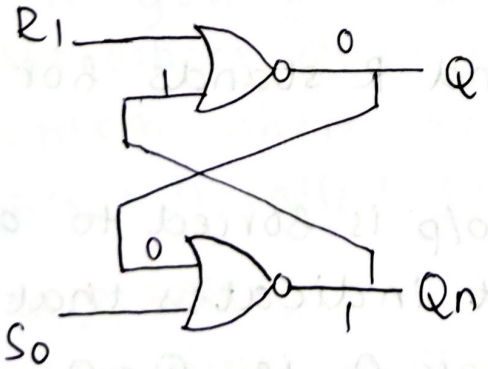
Hence, store mode is bistable.

2. Set mode : S=1 and R=0



- If S=1 , it will force an o/p on U_2 , Qn=0
- This inturn will be fed as an i/p to U_1 where R=0 , resulting in Q=1

3. Resetting : Q=0 , S=0 , R=1

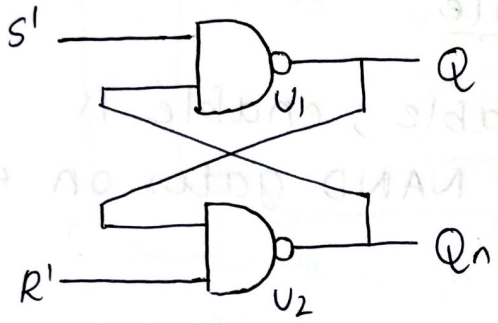


- when R=1, U_1 gives an o/p Q=0 which in turn is fed into U_2 along with S=0 forcing U_2 give an o/p Qn=1 which will again be fed into U_1 along with R=1 forcing U_1 to give o/p Q=0

4. Don't use mode : S=1 and R=1

When S=1 and R=1, we get both Q=0 and Qn=0. This might cause unpredictable outputs when we switch the S-R latch into store mode. Also there is a chance of the latch going into metastable state.

S'R' Latch



NAND

A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

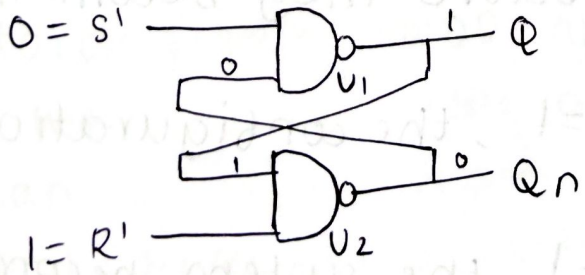
- In S'R' latch, the input codes store, ~~bar~~ bor set and reset are complemented and compared to SR latch.

Modes of operation

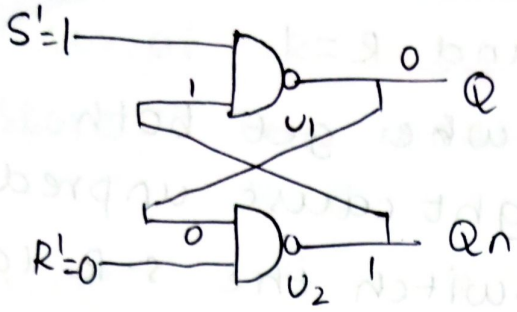
1. Store : S'=1, R'=1
2. Set mode

a) - For storing Q=1, S'=0 and R'=1

- If S'=0, it will force an o/p ~~when S'=0~~ on U_1 of Q=1



~~b~~b) Reset : S'=1 and R'=0



- If $R'=0$, o/p of U_2 ie $Q_n=1$ which will be the i/p of U_1 alongside $S'=1$ making the o/p of U_1, $Q=0$

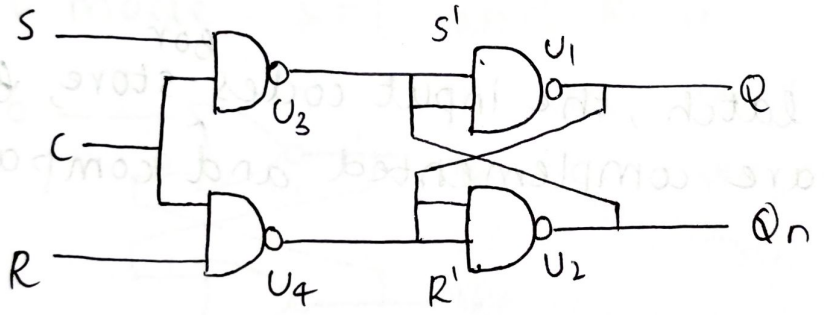
3. Don't use mode

when both S' and R' are 0, the outputs of both U_1 and U_2 ie, $Q_n=1$. This will cause the outputs to go metastable when S' and $R' = 1$

20/9

SR Latch with Enable

- In SR Latch with Enable, enable is implementing by adding two NAND gates on the i/p stage of an $S'R'$ Latch



NAND

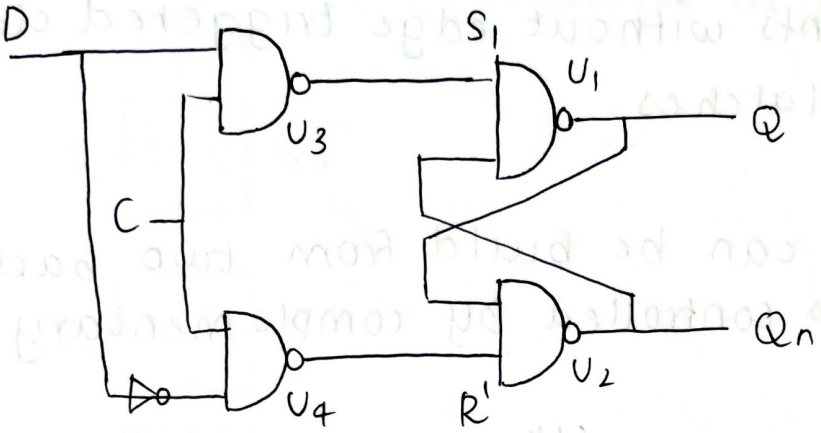
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

- In this configuration, any time $C=0$ the o/p of U_3 and U_4 will be 1. This will be fed into U_1 and U_2 respectively. Then the configuration becomes store state.

- when $C=1$, it has the effect of inverting the values of S and R before they become the inputs of U_1 and U_2.
- when $S=1$, $R=0$ and $C=1$, the configuration goes into Set mode.
- when $S=0$, $R=1$ and $C=1$, the system becomes Reset mode

D Latch

- In D-Latch, the SR Latch with enable is modified in such a way that instead of having two seperate i/p lines to control the o/ps ~~of~~, we are going to use a single i/p for U_3 and U_4 where that i/p is inverted while being fed into U_4. That i/p is known as D (data).



- This configuration removes the don't use states.

- This new circuit when C=1, D=1, Q=1 ie the set state and when C=1, D=0 then Q=0 ie, the reset state.

- This behaviour of o/p when C=1 is called ~~trapping~~ tracking the o/p.

Q.1 Explain the working of a multiplexer using truth table and gate diagram.

Q.2 Form a truth table for a 7 segment display starting from 0000 and ending at 1111

Q.3 What is bistable and metastable state? How can a circuit reach stability, ie Q=0 and Q=1 from metastable state? Explain with the help of a cross coupled inverted pair.

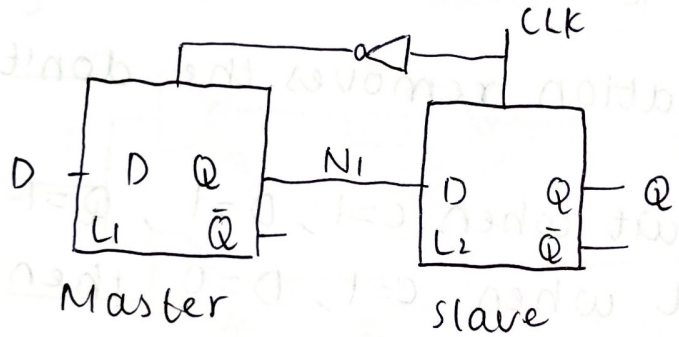
23/9

D Flip Flop

Difference b/w a Latch and a Flip flop.

- Flip flop is edge triggered ie, a bistable element with clock input
- The state of a flip flop changes only during the transition of the clock pulse from 0 to 1 (rising edge) or 1 to 0 (falling edge).
- Bistable elements without edge triggered clock are known as latches.

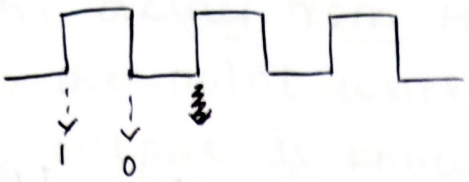
- A D flip flop can be build from two back to back D latches controlled by complementary clocks.



- The latch L_1 is called master and latch L_2 is called slave.
- The node b/w two latches is known as N_1

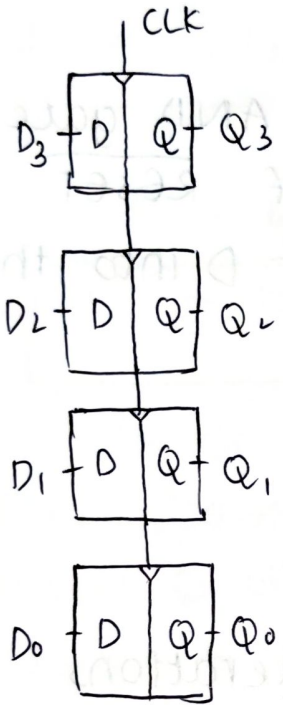
- when clk = 0, ~~not~~ L_1 become transparent and L_2 becomes opaque. Therefore whatever value is at D, propogates through N_1
- when clk = 1, L_2 becomes transparent and L_1 becomes opaque. The value at N_1 is transfered to the Q at L_2 and N_1 is cut off from D.

- A D flip flop copy is D to Q on the rising edge of the clock and remembers it's state at all other times.



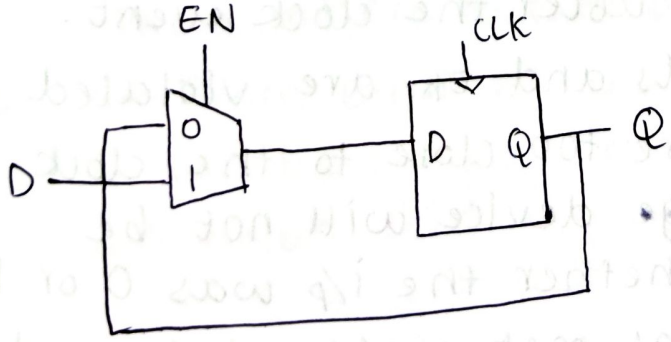
Register

- An n bit register is a bank of n flip flop that share a common clock i/p so that all bits of the register are updated at the same time.



Enabled Flip Flop

An enabled flip flop adds another i/p called EN to determine whether data is loaded on the clock edge

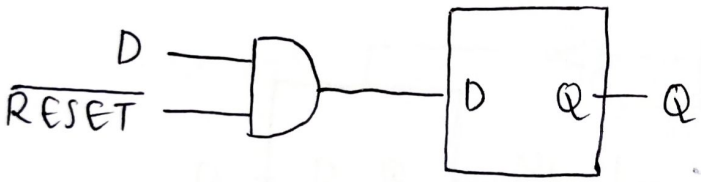


- An i/p multiplexer passes the value of D if EN is 1 (true) or recycles old state from Q

if EN is 0 (false)

Resettable Flip Flop

- A resettable flip flop adds another i/p called RESET.
- If RESET is false, the flip flop behaves like an ordinary D flip flop
- If RESET is true, the flip flop ignores D and resets the o/p to 0.
- When the $\overline{\text{RESET}}$ is false, the AND gate forces a 0 into the flip flop and if $\overline{\text{RESET}}$ is true, AND gate passes a value of D into the flip flop.

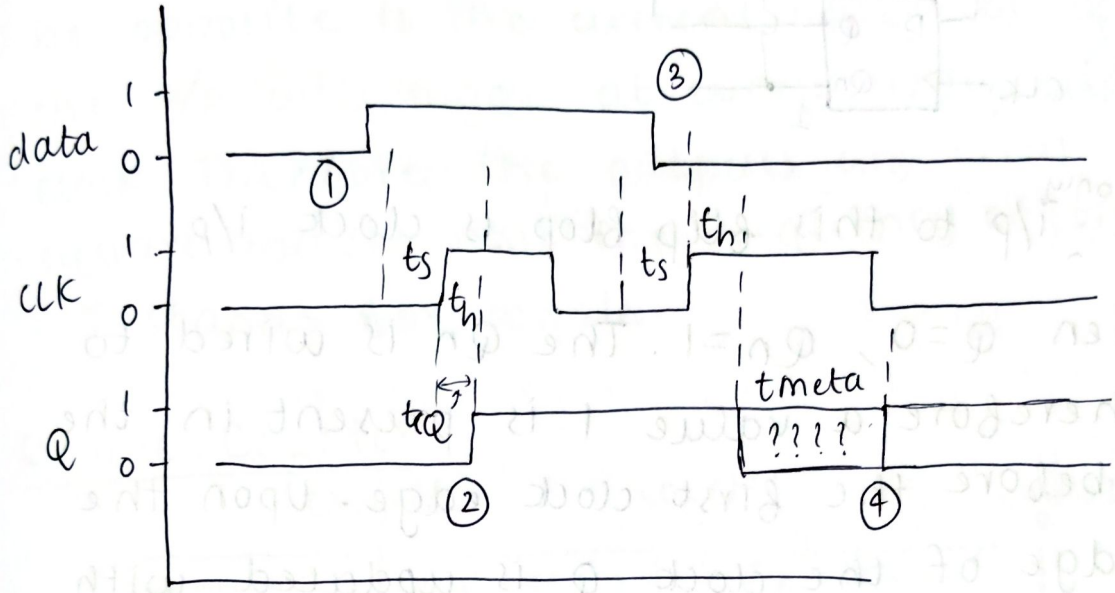


23/9

Sequential Logic Timing Considerations

1. Set up time (t_{setup} / t_s)
   - Specifies how long the data i/p needs to be at a steady state before the clock event
2. Hold time (t_{hold} / t_h)
   - Specifies how long the data i/p needs to be at a steady state after the clock event.
3. If the specifications t_s and t_h are violated (the i/p transitions are too close to the clock transition the storage device will not be able to determine whether the i/p was 0 or 1. This state is known as metastable state and the time for which the device is in meta-stable state is known as t_{meta}.

4. The delay from the time of ~~the~~ a clock event to the point where the data is present on the Q output is known as clock to Q delay ie, t_{cQ}.



① : First transition on data from 0 to 1 ~~need~~ meets ts or th specification allowing the device to latch successfully to the value.

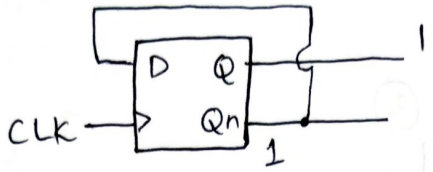
② : The value of D will show up at Q after the delay t_{cQ}

③ : The second transition of data from 1 to 0 violates the t_s or t_h specification and the device goes metastable for a time period t_{meta} where the o/p is ~~in~~ unknown.

④ : ~~Co~~ From coming out of metastable state, Q can be either 1 or 0 ie, random or unknown

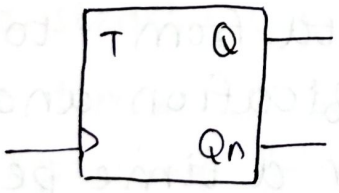
25/9 ## Toggle Flop Clock privel

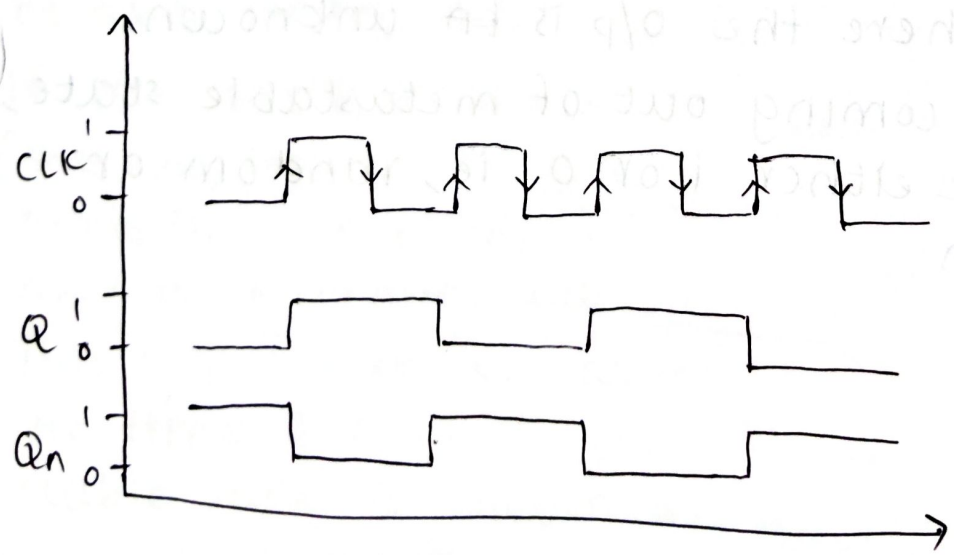
In T Flip-flop circuit, it contains a D flip flop configured in such a way that the $\bar{Q}_n$ o/p is wired back to the D i/p



- The only i/p to this flip flop is clock i/p
- when $Q=0$, $\bar{Q}_n=1$. The $\bar{Q}_n$ is wired to D i/p therefore a value 1 is present in the D i/p before the first clock edge. Upon the rising edge of the clock Q is updated with value of D ie, $Q=1$. Therefore $\bar{Q}_n=0$

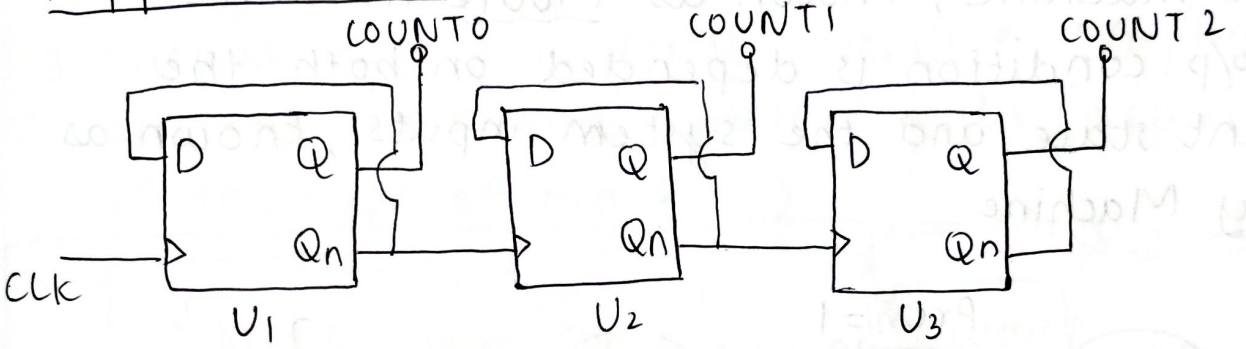
This configuration produce the o/p in the form of square waves with exact half the frequency of the incoming clock signals. So T flip flop is also known as 'Clock Divider'.

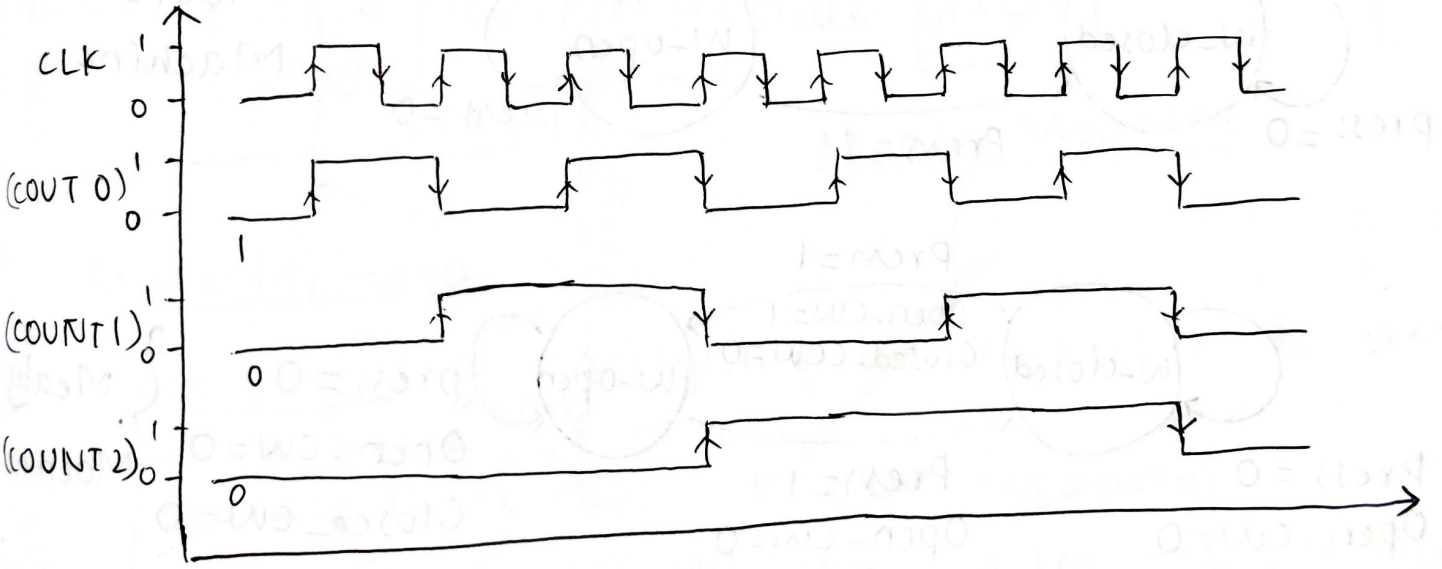




- Q and Qn will always be opposite logic values.
- when rising edge of a clock occurs, Q will be updated with the value in D.
- In this configuration, value of D will always be opposite to the current value of Q. Therbore the o/p will toggle at every rising edge of the clock. Therbore the outputs are created as square waves half the freauency of the clock.
- T stands bor toggle.

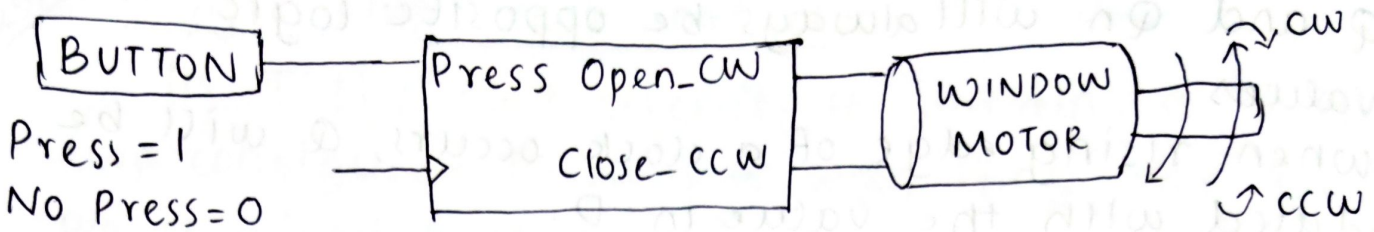
Ripple Counter





1/10 Finite State Machine (FSM)

- Also known as state machine.
- Circuit that contains pre defined finite no. of states. It can only be at one state at a time. Every state transition is based on a triggering event (edge of a clock)

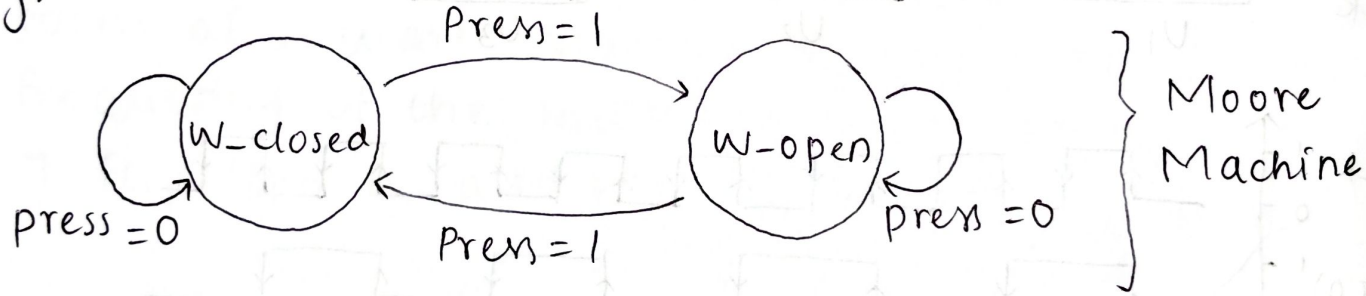


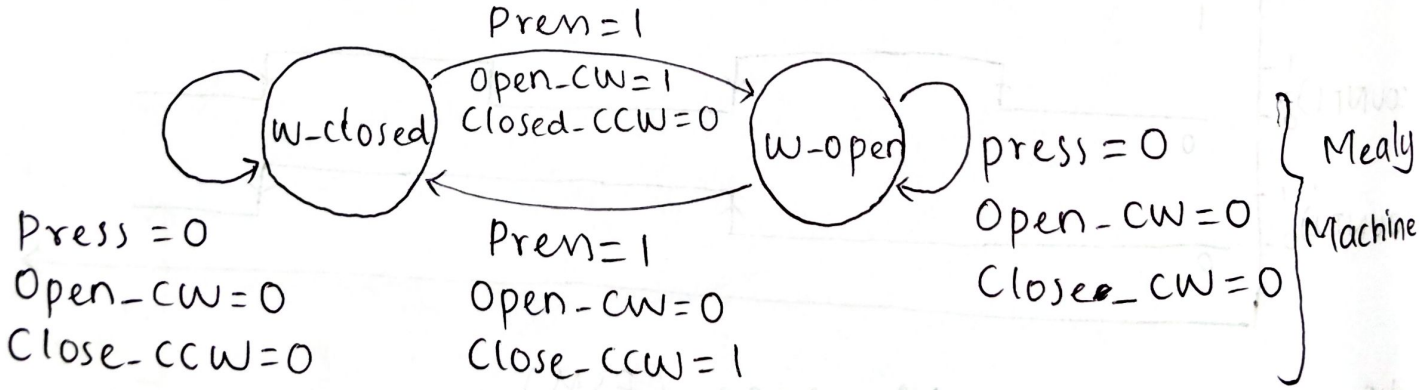
State diagram

- It is the graphical description of functionality of FSM.
- There are two different kinds of o/p conditions for a state machine:

1. The o/p is only depended on the current state of the machine, known as Moore machine.
2. The o/p condition is depended on both the current state and the system inputs, known as Mealy Machine

Eg:-





State Transition Table

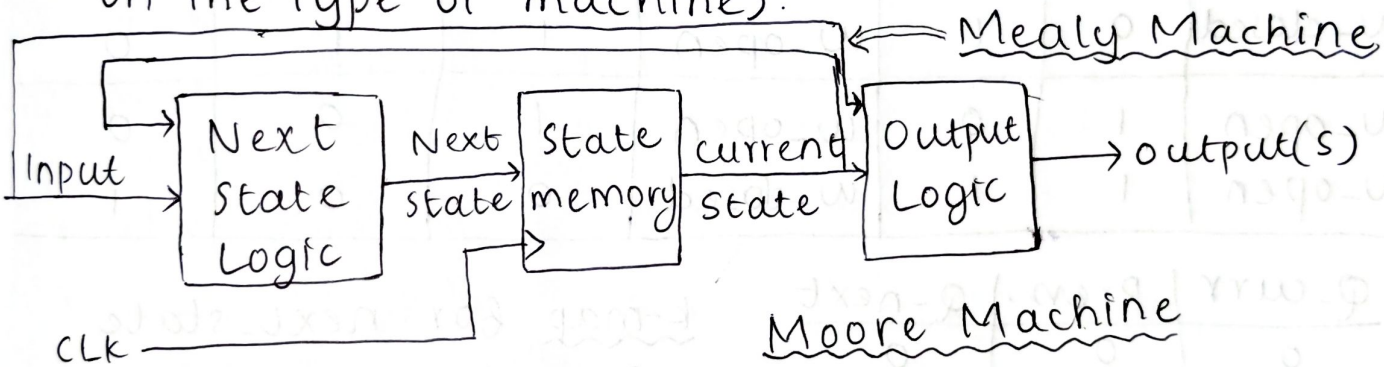
Current State	Input: Press	Next State	outputs: Open CW	outputs: Close CW
W_closed	0	W_closed	0	0
W_closed	1	W_open	1	0

w_open	0	w_open	0	0
w_open	1	w_closed	0	1

Logic Synthesis of FSM

- There are 3 components of a state machine:

1. Next state logic : It creates the signal next state based on current state and depending on ~~the type of machine~~ the system inputs.
2. State memory : It holds the current state. The current state is updated to next state on the rising edge of the clock (implemented using D flip flop)
3. Output logic : It creates system o/p based on current state and system inputs (depending on the type of machine).



State Memory

State Encoding : Assigning binary values to descriptive ~~words~~ names of states.

There are three types of state encoding:

1. Binary Encoding : The state codes are simply a set of binary codes ie, 00, 01, 10, 11
2. Gray code Encoding : In this, the value of the code differs only one with its neighbours ie, 00, 01, 11, 10
3. One hot Encoding : In this, the one hot state

codes will be 001, 010, 100 ...

	Binary	Graycode	One hot
S_0	000	000	00000001
S_1	001	001	00000010
S_2	010	011	00000100
S_3	011	010	00001000
S_4	100	110	00010000
S_5	101	111	00100000
S_6	110	101	01000000
S_7	111	100	10000000

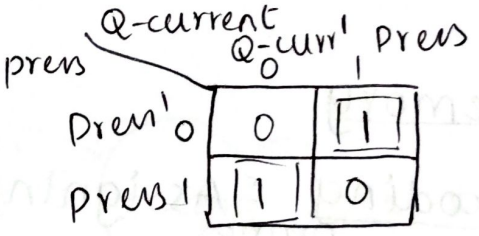
8/10 IMP

Current state	Input	Next state	Outputs	
Qcurr	Press		Q_next	open_cw	close_ccw
w_closed	0	0	w_closed	0	0	0
w_closed	0	1	w_open	1	1	0
w_open	1	0	w_open	1	0	0
w_open	1	1	w_closed	0	0	1

⇒

Q_curr	Press	Q_next
0	0	0
0	1	1
1	0	1
1	1	0

k-map for next_state



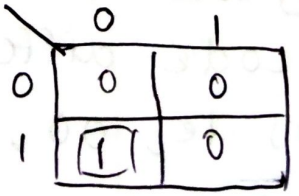
∴ Boolean expression for next state logic will be

$= (Q\_curr' \cdot Press) + (Q\_curr \cdot Press')$

[or $(Q\_curr \oplus Press)$]

⇒ output

Q_curr	Press	Open_cw
0	0	0
0	1	1
1	0	0
1	1	0

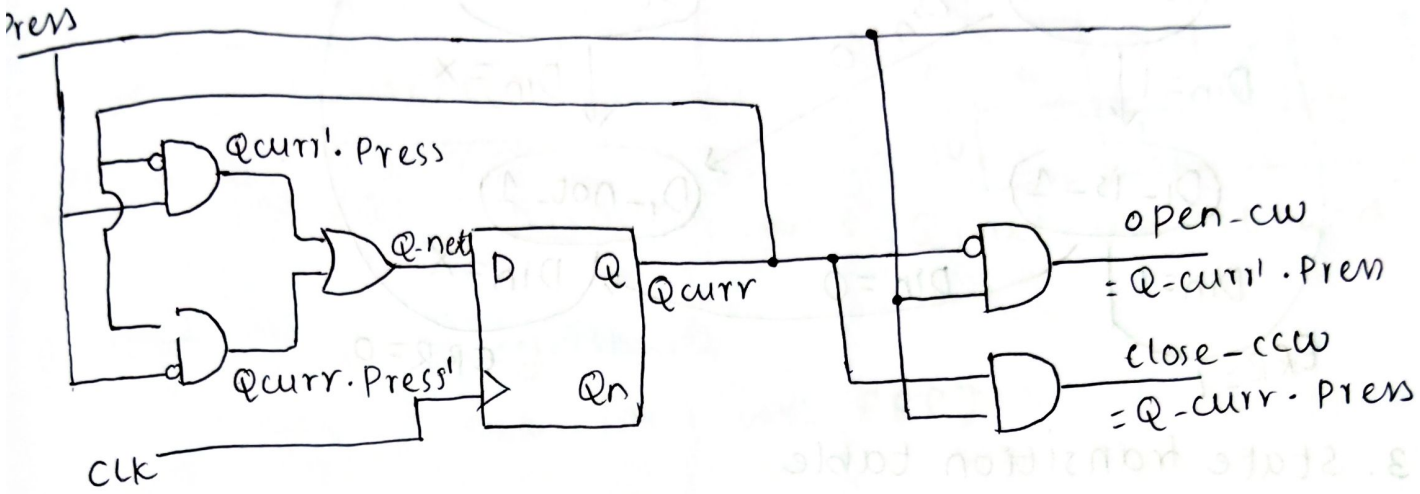


∴ $Qcurr' \cdot Press$

Q-curr	Press	Close-cw
0	0	0
0	1	0
1	0	0
1	1	1

	0	1
0	0	0
1	0	1

(Q-curr · Press)



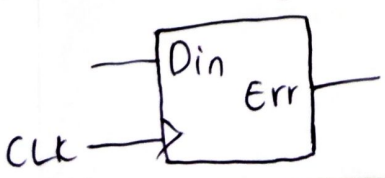
9/10 Finite State Machine Design Flow

1. word description
2. State diagram
3. State transition table
4. state memory synthesis
5. Next state logic "
6. Output logic "
7. Final logic diagram

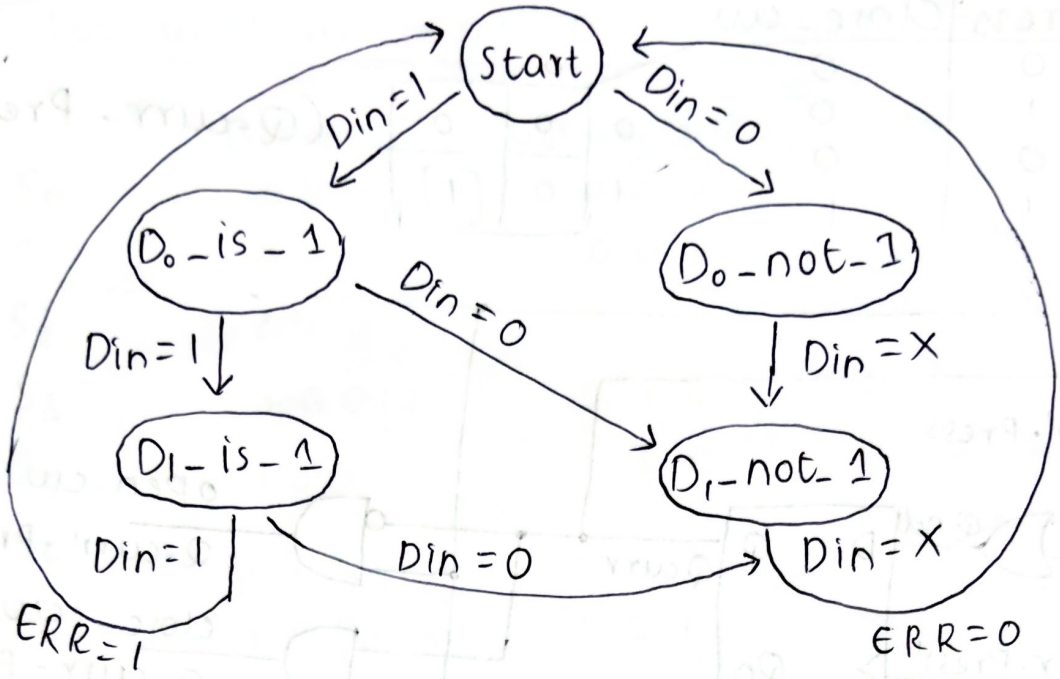
Eg: Serial bit sequence detector

1. word description : Design a circuit that will monitor an incoming serial bit stream. The stream consists of data in groups of 3 bits D_0, D_1, D_2. If the sequence occurs in the form of 111 an error is detected, ie ERR = 1 else for all other representation, ERR = 0.

2. state diagram



D_0	D_1	D_2
0	x	x
1	0	x
⋮		



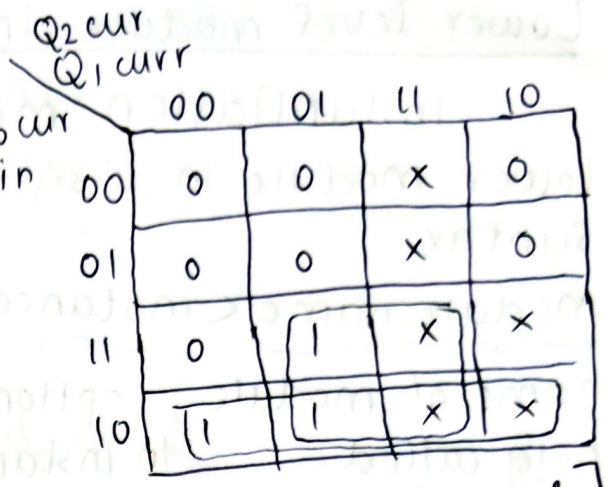
3. State transition table

Current state				Input	Next state				Output
	Q_2curr	Q_1curr	Q_0curr	Din		Q_2nxt	Q_1nxt	Q_0nxt	ERR
Start	0	0	0	0	D_0 not 1	0	1	1	0
Start	0	0	0	1	D_0 is 1	0	0	1	0
D_0 is 1	0	0	1	0	D_1 not 1	1	0	0	0
D_0 is 1	0	0	1	1	D_1 is 1	0	1	0	0
D_1 is 1	0	1	0	0	Start	0	0	0	0
D_1 is 1	0	1	0	1	Start	0	0	0	1
D_0 not 1	0	1	1	0	D_1 not 1	1	0	0	0
D_0 not 1	0	1	1	1	D_1 not 1	1	0	0	0
D_1 not 1	1	0	0	0	Start	0	0	0	0
D_1 not 1	1	0	0	1	Start	0	0	0	0

State	Code
• Start	0 0 0
• D_0 is 1	0 0 1
• D_1 is 1	0 1 0
• D_0 not 1	0 1 1
• D_1 not 1	1 0 0

5. Next state logic

Q_2curr	Q_1curr	Q_0curr	Din	Q_2nxt
0	0	1	0	1
0	1	1	0	1
0	1	1	1	1



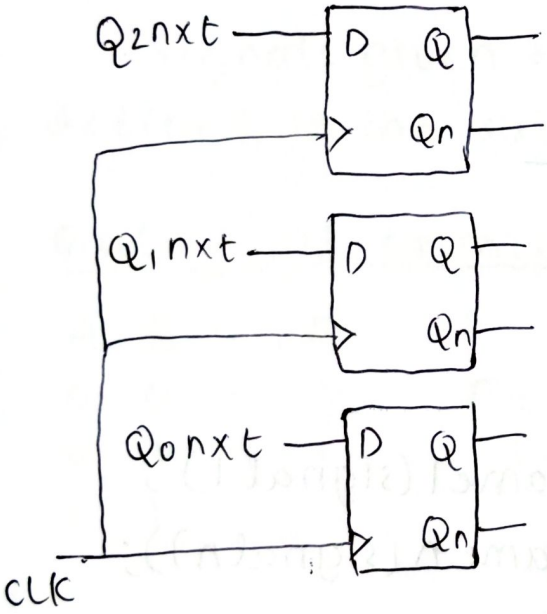
ie, $(Q_1 cur \cdot Q_0 cur) + (Q_0 curr \cdot Din')$

[Find Boolean expression w.r.t Q_1nxt, Q_0nxt and]

6. Output logic synthesis

[Find boolean expression wrt ERR]

7. Final logic diagram



Structural Design

- In verilog structural design refers to including lower level subsystems within a higher level module to produce desired functionality. This is called hierarchy and enables desired partitioning.

- A subsystem in verilog is simply another module called by a higher level module

Lower level module instantiation

- Instantiation means using or including a lower module in a system.

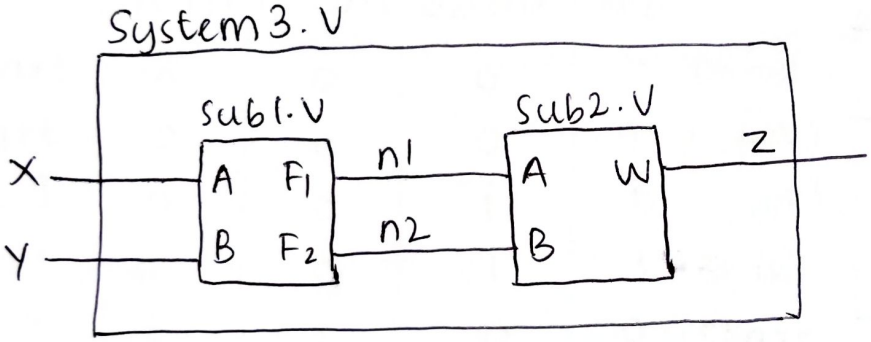
Syntax:

module_name <instance identifier> (port mapping)

name of module being called — optional identifier to instantiate multiple instances of the same sub module. — to map i/p and o/p codes of lower level module.

- There are two techniques to connect signals to the ports of the lower level module

1. Explicit port mapping



Syntax:

```
module_name <inst id> (.portname1(signal1),
                       .portnamen(signaln));
```

```
module sub1 (output wire F1, F2,
             input wire A, B)
  // behaviour
end module
module sub2 (output wire W,
             input wire A, B)
  // behaviour
end module.
module systems (output wire Z,
                input wire X, Y)
```

```
wire n1, n2;
Sub1 U0 (.F1(n1), .F2(n2), .A(X), .B(Y));
Sub2 U1 (.W(Z), .A(n1), .B(n2));
end module
```

2. Positional port mapping

Eg: // same module sub1 and sub2

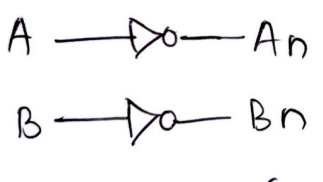
```
module systems (output wire Z,
                input wire X, Y)
wire n1, n2;
Sub1 U0 (n1, n2, X, Y);
Sub2 U1 (Z, n1, n2);
```

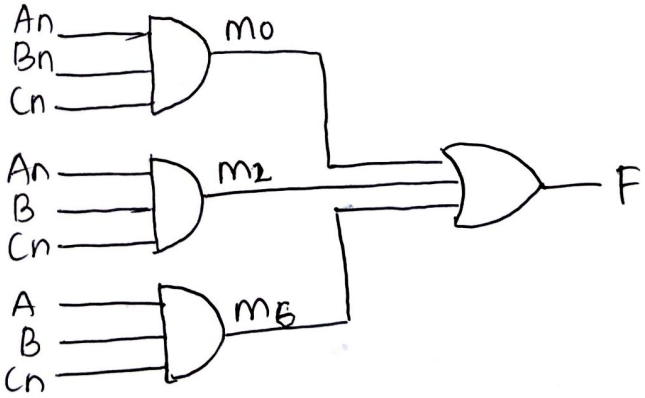
- Signals given the correct order of how it is defined in the sub modules.

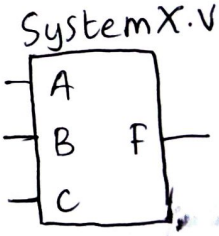
Gate level primitives

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

$F = \sum_{(0,2,6)} A'B'C' + A'BC' + ABC'$








```
module SystemX (output wire F,
                input wire A, B, C)
wire An, Bn, Cn;
wire m0, m2, m6;
not U0(An, A);
```

```
not U1(Bn, B);
not U2(Cn, C);
and U3(m0, An, Bn, Cn);
and U4(m2, An, B, Cn);
and U5(m6, A, B, Cn);
or U6(F, m0, m2, m6);
```