HDL

1)
a) Draw the HDL design flow.
b) State the verilog HDL supported levels of abstraction for desiging digital circuit.
c) Write HDL code to implement a 4-to-2 line priority encoder.

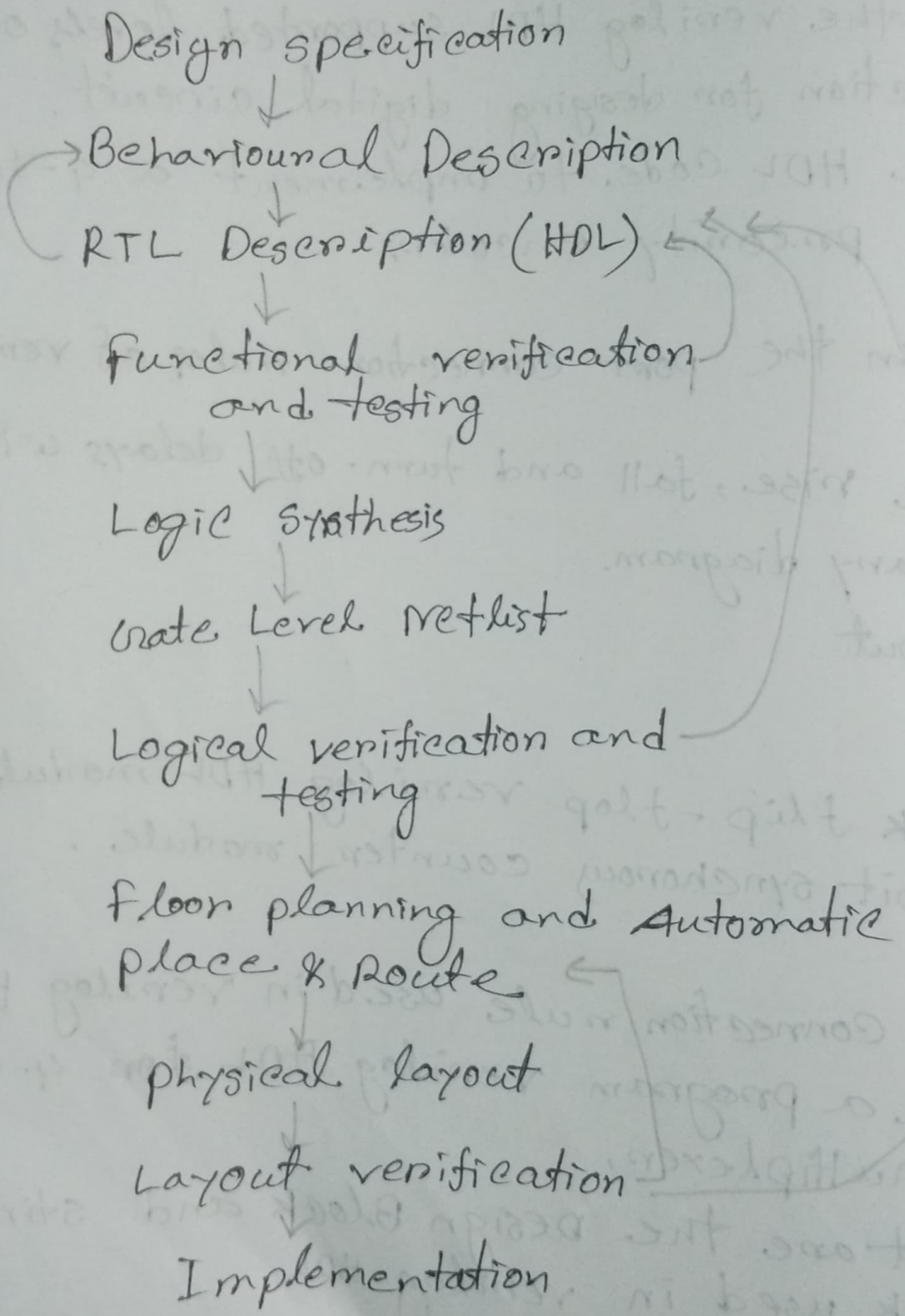
2. a) Explain the port connection rules of verilog HDL.
b) Define rise, fall and turn-off delays with necessary diagram.
c) output

3. a) JK flip-flop verilog HDL module } Code
b) 4 bit synchronous counter module. } for

* port connection rule used in verilog HDL.
*) write a program verilog HDL for 4-to-1 multiplexer.
*) what are the design Block and stimulus Block used in verilog HDL.

[C++] Code and diagram

HDL Design flow:



Port connection rules:

i) Inputs: Internally must always be of type net, externally the inputs can be connected to a variable of type reg or net.

ii) Output: Internally can be of type net or reg, externally the outputs must be connected to a variable of type net.

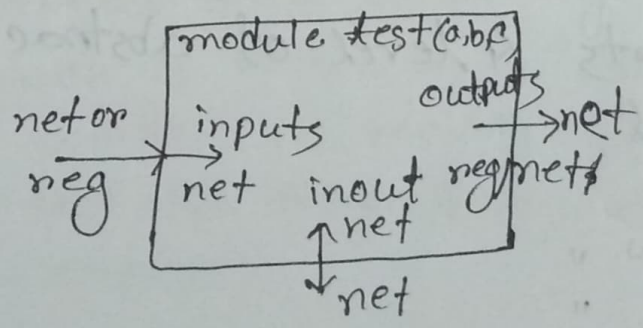
iii) Inouts: internally or externally must always be type net, can only be connected to a variable net type.

iv) Unconnected ports: Unconnected ports are allowed by using a ",".

v) The net data types are used to connected structure.

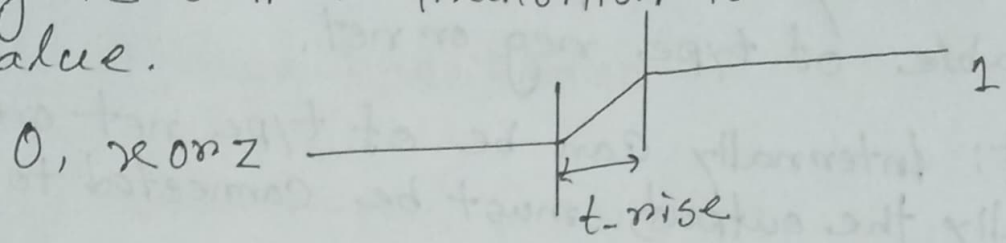
vi) A net data type is required if a signal can be driven a structural connection.

vii) Width matching: It is legal to connect internal and external ports of different sizes.

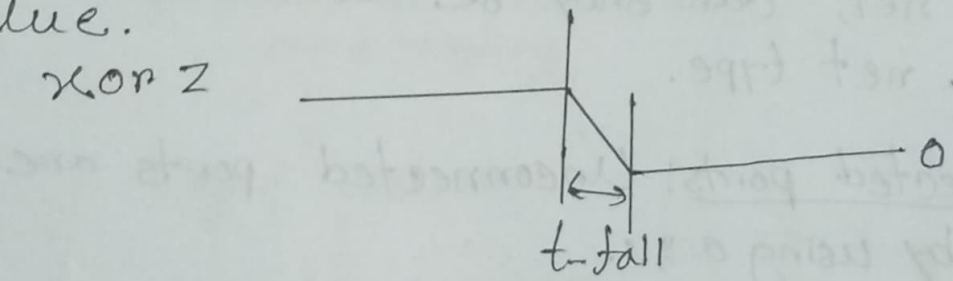


b) Define Rise Delay, Fall and Turn-off delay.

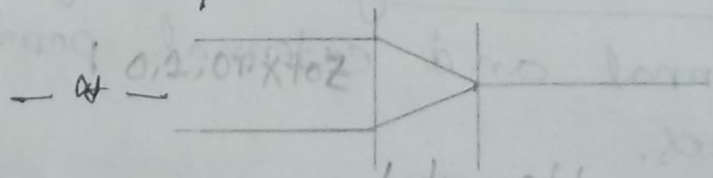
Rise delay: The rise delay is associated with a gate output transition to a 1 from another value.



Fall delay: The fall delay is associated with a gate output transition to a 0 from another value.



Turn-off delay: The turn off delay is associated with a gate output transition to the high impedance value (z) from another value. If the value changes to x, the minimum of the three delays is considered.



1.b)

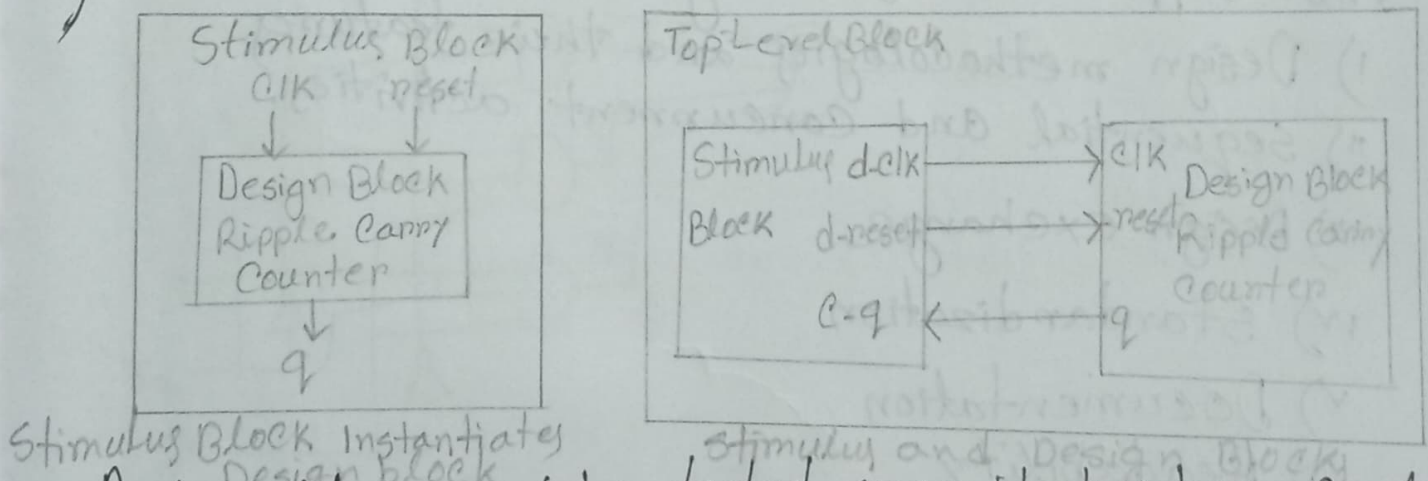
Verilog supports 4 level of abstraction namely,

i) Switch level
ii) Gate "
iii) Data flow "
iv) Behavioral "

At the switch level, the implementation is done with the help of switches. At the gate level, we talk about gate interconnections. In data flow level design is done based on equations, which can be said that it is based on data flow, how the data is flowing in the circuit, based on that the equations have written. The behavioral level is based on the behavior of the circuit.

— ✱ —

a)



Stimulus Block Instantiates Design block

Stimulus and Design Blocks

A design block must be tested once it has been completed. The design block's functionality can be tested by applying stimuli and observing the results. This type of block is known as a stimulus block, and it can be written in verilog as a test bench. To completely test the design block, many test benches might be employed.

— ✱ —

(A) VHDL stands for every High-speed Integrated Circuit hardware description language.

HDL stands for Hardware Description Language. It is a programming language that is used to describe, simulate, and create hardware like digital circuit(ics). HDL is mainly used to discover the faults in the design before implementing it in the hardware.

VHDL supports the following features:

i) Design methodologies and their features
ii) Sequential and concurrent activities.
iii) Design exchange
iv) Standardization
v) Documentation
vi) Readability
vii) Large-scale design
viii) A wide range of descriptive capability

Verilog: Verilog is also HDL for describing electronic circuits and systems.

```
and ( y0, i0, s1n, s0n);
and ( y1, i1, s1n, s0);
and ( y2, i2, s1, s0n);
and ( y3, i3, s1, s0);
or ( out, y0, y1, y2, y3)
end module
```

— * —

(4) Min value: The min value is the minimum delay value that the designer expects the gate to have.

Typ val: The typ value is the typical delay value that the designer expects the gate to have.

Max value: The max value is the maximum delay value that the designer expects the gate to have.

min, typ, or max values can be chosen at veril run time

i2 i1 -d-

↑ ↑ → i0

[7:0] i = 10101010

8.b) 7.b) এটার কি আগে K-map করে circuit আসবে কিভাবে

0000 → 0101 → 1010 → 0110 → 1001 → 0011 → 1100 → 1111 → 0000

Q_4	Q_3	Q_2	Q_1	$\bar{Q}_4$	$\bar{Q}_3$	$\bar{Q}_2$	$\bar{Q}_1$	T_4	T_3	T_2	T_1
0	0	0	0	0	1	0	1	0	1	0	1
0	0	0	1	X	X	X	X	X	X	X	X
0	0	1	0	X	X	X	X	X	X	X	X
0	0	1	1	1	1	0	0	1	1	1	1
0	1	0	0	X	X	X	X	X	X	X	X
0	1	0	1	1	0	1	0	1	1	1	1
0	1	1	0	1	0	0	1	1	1	1	1
0	1	1	1	X	X	X	X	X	X	X	X
1	0	0	0	X	X	X	X	X	X	X	X
1	0	0	1	0	0	1	1	1	0	1	0
1	0	1	0	0	1	1	0	1	1	0	0
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	1	1	1	1	0	0	1	1
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	0	0	0	0	1	1	1	1

T_4 (rows Q_4Q_3, columns Q_2Q_1)

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	0	X	1	X
01	X	1	X	1
11	0	X	1	X
10	X	1	X	1

$T_4 = Q_1 + Q_2$

T_3 (rows Q_4Q_3, columns Q_2Q_1)

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	1	X	1	X
01	X	1	X	1
11	0	X	1	X
10	X	0	X	1

$T_3 = \bar{Q}_4 + Q_2$

T_2 (rows Q_4Q_3, columns Q_2Q_1)

Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	0	X	1	X
01	X	1	X	1
11	1	X	1	X
10	X	1	X	0

$T_2 = Q_3 + Q_1$

T_1 (rows Q_4Q_3, columns Q_2Q_1)

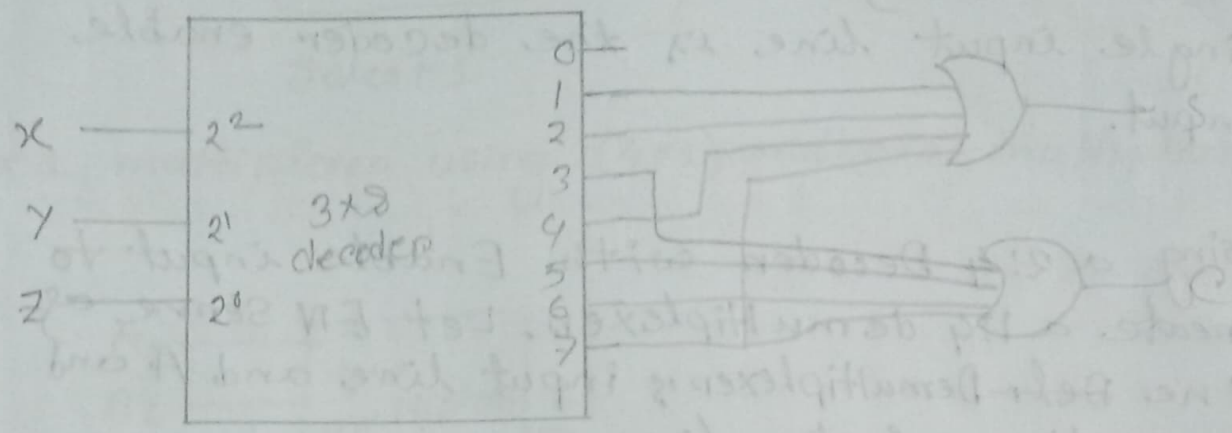
Q_4Q_3 \ Q_2Q_1	00	01	11	10
00	1	X	1	X
01	X	1	X	1
11	1	X	1	X
10	X	0	X	0

$T_1 = \bar{Q}_4 + Q_3$

Circuit diagram:

Section-B

2020, 5.a) implement a full adder circuit using a 3-to-8 decoder.



2020. 5.b) The decoder can be used as a demultiplexer by using the input lines for data selection and an enable line for data input.

A decoder as Demultiplexer:

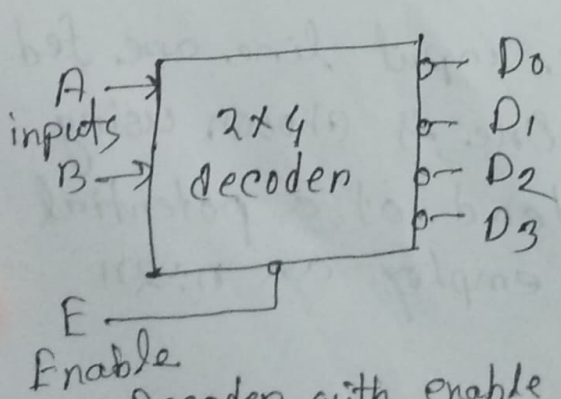
As a multiplexer, a Decoder with Enable input might be used. A circuit known as a demultiplexer takes data from a single line and sends it to one of 2n different output lines.

N selection lines and one input line are fed into a 2n demultiplexer. One is chosen using these selection lines instead of a potential 2n output line. we employ an n:2n

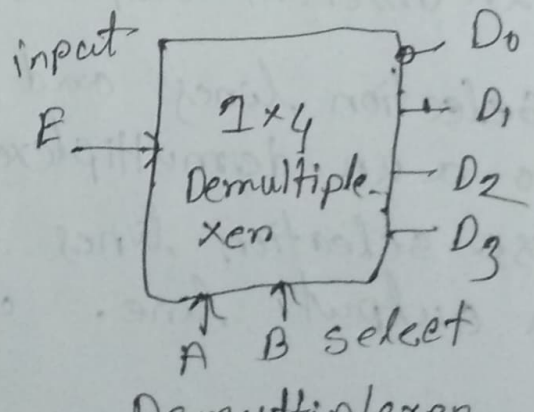
decoder with Enable input to create a 2n demultiplexer. The n selection lines of the Demultiplexer are the n input lines that the decoder receives, and the Demultiplexer's single input line is the decoder enable input.

Using a 2:4 Decoder with Enable input to Create a 1:4 demultiplexer. Let EN serve as the ~~Data~~ Demultiplexer's input line and A and B as the selection lines.

When A and B are used as selection inputs, and EN is used as the input line, the decoder depicted below performs as a 2:4 demultiplexer. Although the single input variable E has a path to all four outputs, the binary combination of the two selection lines A and B specifies that the input data is directed to only one of the output lines. ~~The~~



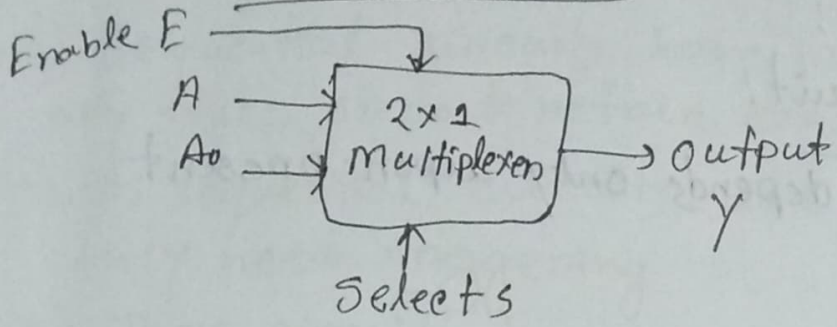
Decoder with enable



Demultiplexer

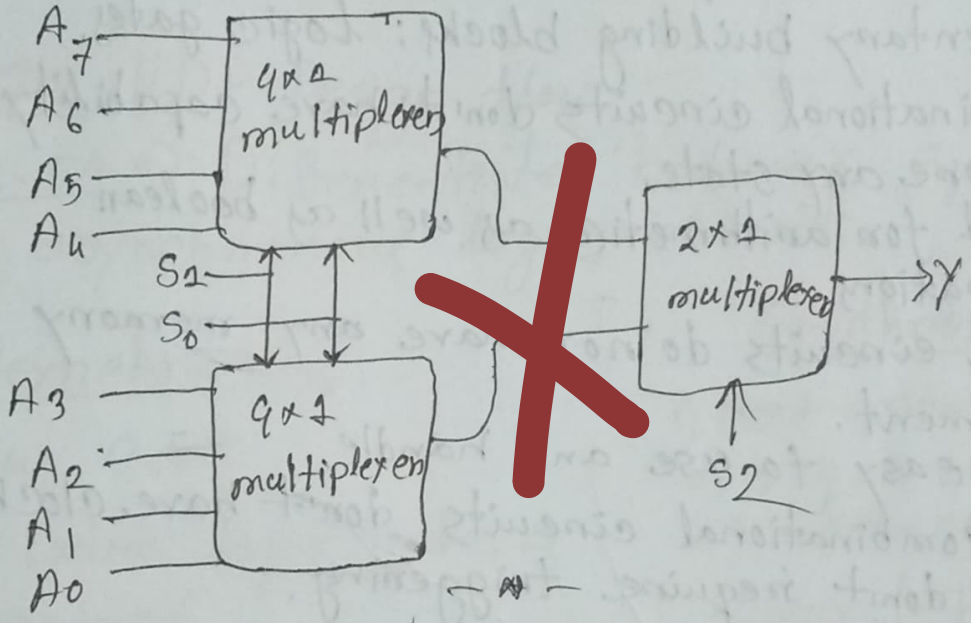
2020 5.c)

2x1 multiplexer

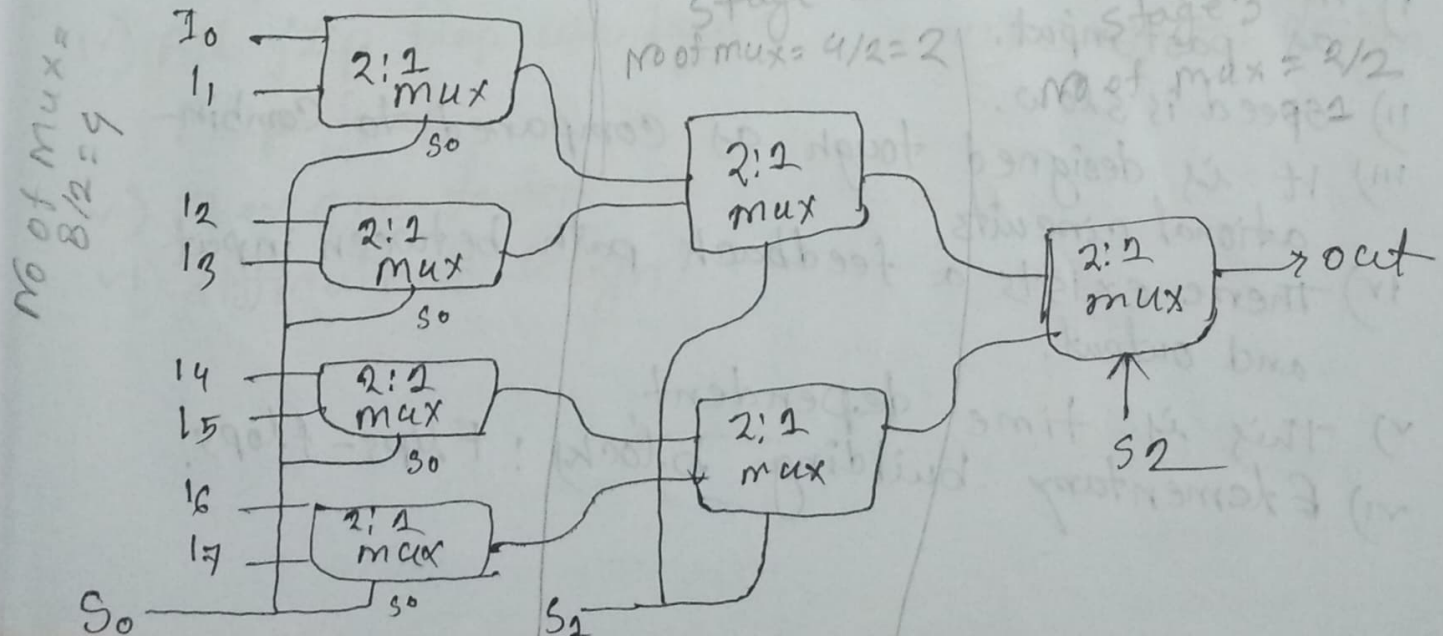


input	output
S0	Y
0	A0
1	A1

8x1 multiplexer using (4x1) and (2x1) multiplexer:



8:1 multiplexing using 2:1 multiplexers:



2020-5.a
2021-5.a Difference between Combinational and Sequential circuit;

Combinational circuit:

i) In this output depends only upon present input,
ii) Speed is fast.
iii) It is designed easy.
iv) There is no feedback between input and output.
v) This is time independent.
vi) Elementary building blocks: Logic gates.
vii) Combinational circuits don't have capability to store any state.
viii) used for arithmetic as well as boolean operations.
ix) These circuits do not have any memory element.
x) It is easy to use and handle.
xi) As combinational circuits don't have clock, they don't require triggering.

Sequential circuit:

i) In this output depends upon preset as well as past input.
ii) speed is slow.
iii) It is designed tough as compared to Combinational circuits.
iv) There exists a feedback path between input and output.
v) This is time dependent.
vi) Elementary building blocks: Flips-flops.

vii) Mainly used for storing data.
viii) Sequential circuits have capability to store any state. or to retain earlier state.
ix) As sequential circuits are clock dependent they need triggering.
x) These circuits have memory element.
xi) It is not easy to use and handle

— ∝ —

2021 7b) HDL Design flow

2020 1.5a.ii) Differentiate between synchronous and asynchronous logic circuit.

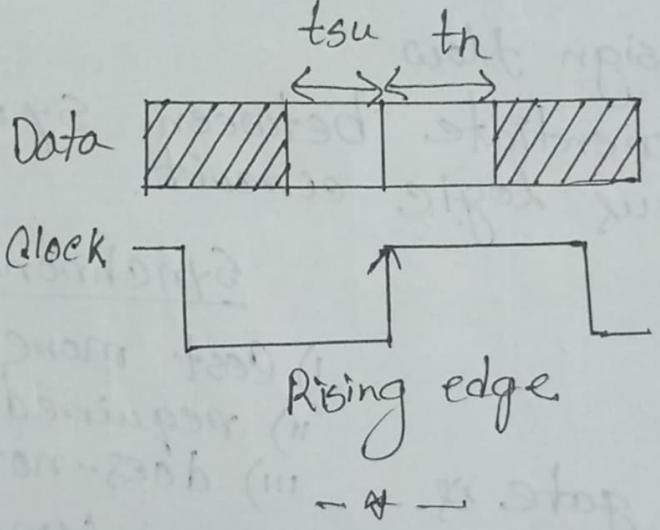
Asynchronous	Synchronous
i) Low cost	i) Cost more
ii) No Extra logic gate is required	ii) required
iii) speed depend on the flip-flop.	iii) does-not depend.
iv) All flip flop use toggle Flip-flop.	iv) Any flip-flop can be use.
v) They are faster	v) slower
vi) difficult to design.	vi) easy to design.

— ∝ —

2021. 5.b) Define setup time and hold time with necessary diagram.

⇒ **Setup time:** Setup time is the amount of time required for the input to a Flip-flo to be stable before a clock edge.

Hold time: Hold time is the minimum amount of time required for the input to a Flip-flop to be stable after a clock edge.

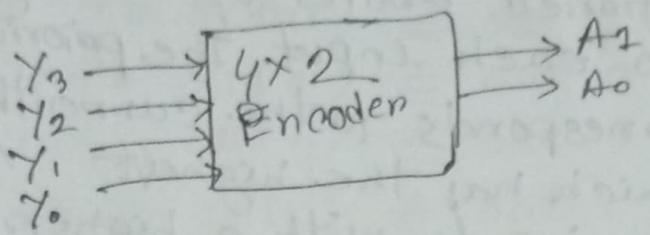


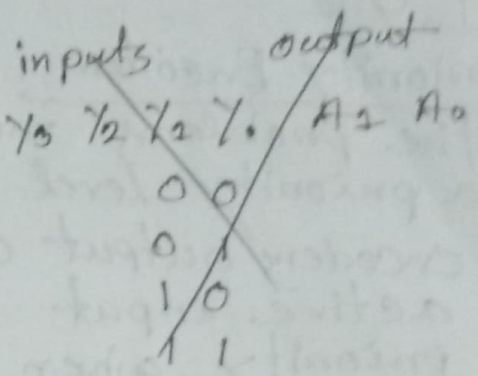
2021.5.c) D flip-flop with, preset, clear, diagram, truth table, timing diagram.

2021.7.c)

Encoder: An encoder is a digital circuit that converts a set of binary inputs into a unique binary code. An encoder is a combinational circuit that performs the reverse operation of a Decoder. It has a maximum of 2^n input lines and n output lines.

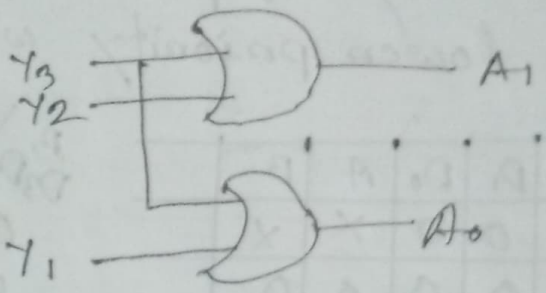
4-to-2 encoder





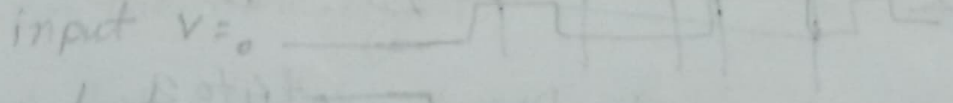
Inputs outputs

Y_3	Y_2	Y_1	Y_0	A_1	A_0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1



2.

```
V= 4'b 0000; out= 1
V= 4'b 0110; " = 0
V= 4'b 0011; " = 0
V= 4'b 1010; " = 0
```



2021 Q.c

Priority Encoder: The priority Encoder solves the problems mentioned above by allocating a priority level to each input. The priority encoders output corresponds to the currently active input which has the highest priority. when an input with a higher priority is present, all other inputs with a lower priority will be ignored.

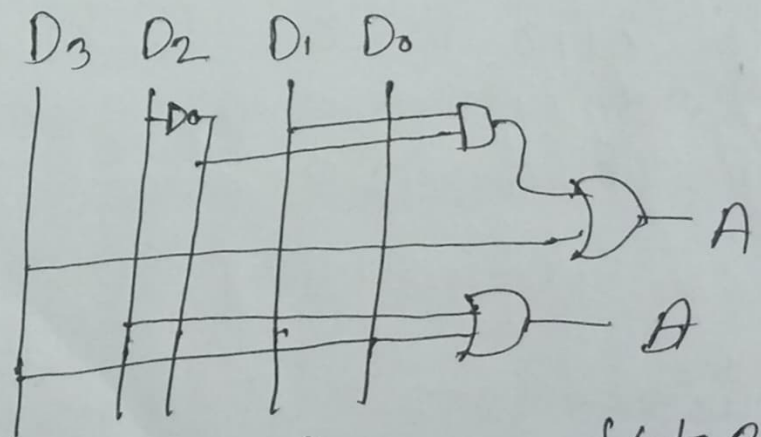
D_3	D_2	D_1	D_0	A	B
0	0	0	0	X	X
1	0	0	0	0	0
X	1	0	0	0	1
X	X	1	0	1	0
X	X	X	1	1	1

B / D_3D_2 \ D_1D_0	00	01	10	10
00	X	0	0	0
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$B = D_2 + D_3$

A / D_3D_2 \ D_1D_0	00	01	11	10
00	X	0	1	1
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$A = D_3 + D_1\bar{D}_2$



Circuit Diagram of 4 to 2 priority encoder

```
module priorityencoder4to2(A,B,D);
input [3:0]D;
output A,B;
wire D[2]b, a;
not (D[2]b, D[2]);
and (a, D[2], D[2]b);
or(A, a, D[3]);
or(B, D[3], D[2]);
endmodule.
```

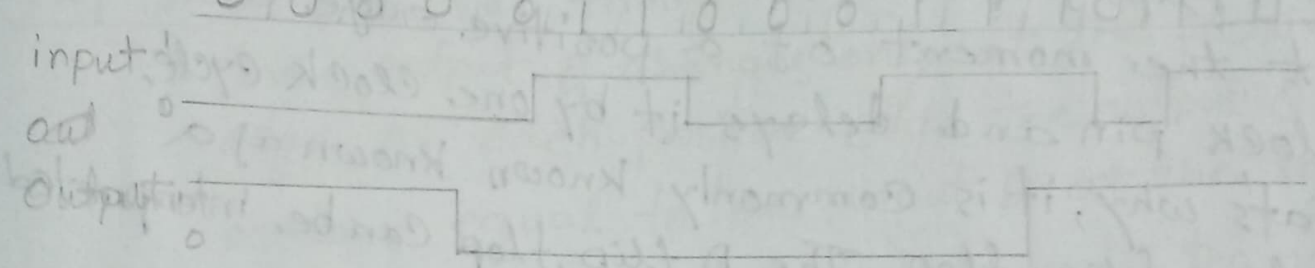
2021. 8. c)

```
nor(n, v[3], v[2]);
and(a, v[1], v[0]);
xor(x, n, a);
```

Output

```
v=4'b 0000; ⇐ out 1
#25 v=4'b 0110; ⇐ out 0
#25 v=4'b 0011; ⇐ out 0
#25 v=4'b 1011; ⇐ out 1
```

Timing diagram:



—x—

2020 6.a Compare between D latch and D flip-flop with timing diagram:

A flip-flop is synchronous. It works based on the clock signal.

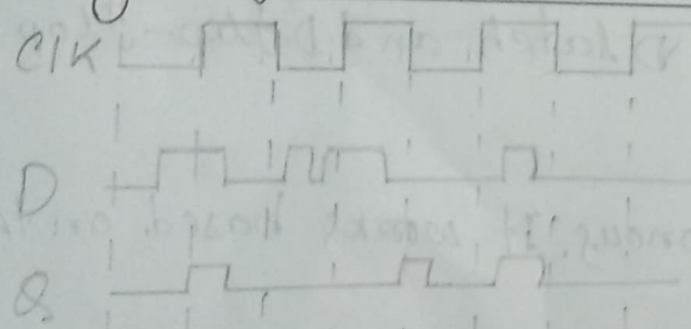
A latch is asynchronous. It does not work based on the time signal.

D Latch: Latch is an electronic device that can be used to store one bit of information. The D latch is used to compare or latch the logic level which is present on the Data line when the clock input is high. If the Data on the D line changes state while the clock pulse is

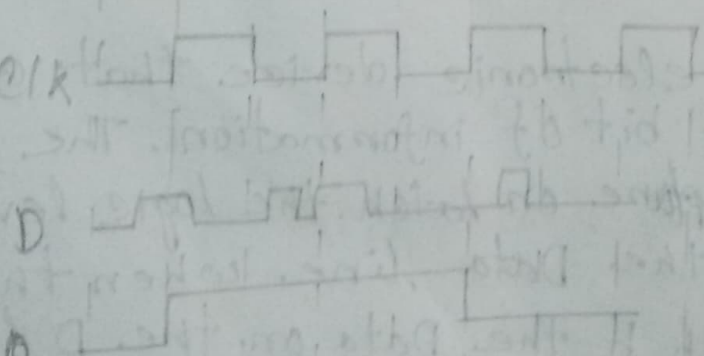
high, then the output Q, follows the input D. When the Clk input fails to logic 0, the last state of the D input is trapped and held in the latch.

D Flip flop: The working of D flip-flop is similar to the D latch except that the output of D Flip-flop takes the state of the D input at the moment of a positive edge at the clock pin and delays it by one clock cycle. That's why, it is commonly known known as a delay flip-flop. The D flip flop can be interpreted as a delay line zero order hold.

⇒ Timing diagram of D Latch:



Timing diagram of D flip flop:



Latch: From the timing diagram it is clear that the output Q's waveform resembles that of input D's waveform when the clock is high whereas when the clock is low Q retains the previous value of D (the value before clock dropped down to 0).

Flip flop: From the timing diagram it it clear that the output Q changes only at the positive edge. At each positive edge the output Q becomes equal to the input D at the instant and this value of Q is held until the next positive edge.

— ✱ —

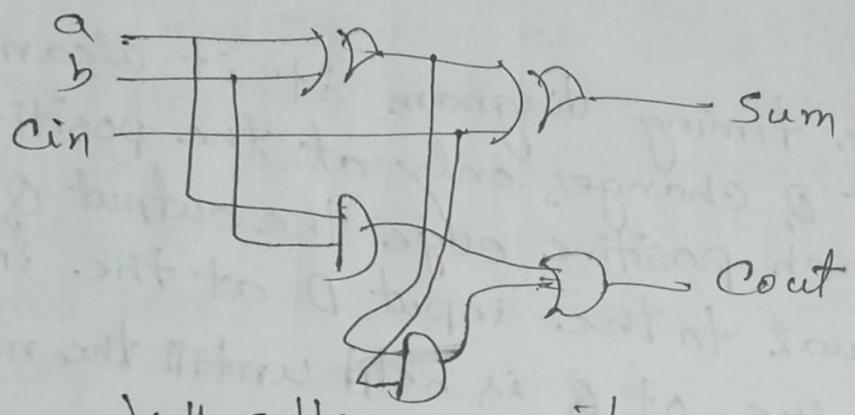
2020

7.a) Advantages of Synchronous Counter:

i) It operates at the same time.
ii) It has no associated propagation delay.
iii) Operation is faster than with an asynchronous counter.
iv) It's easier to design than the asynchronous counter.
v) Because the count sequence is regulated by ~~count~~ logic gates, the possibility of an error is reduced.

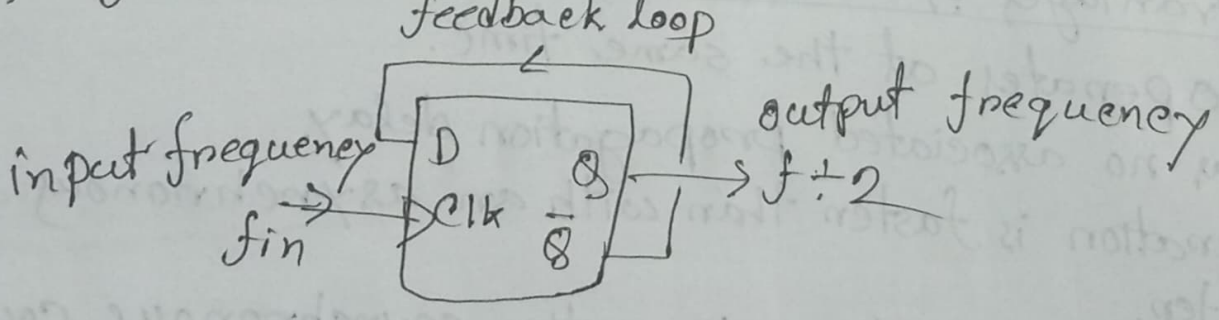
— 9 —

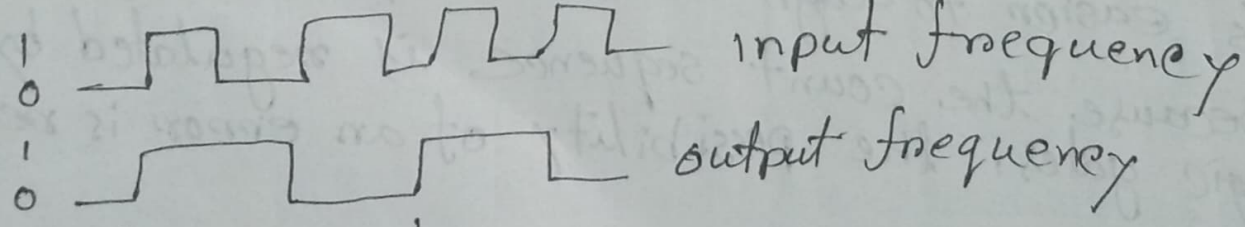
2020
8.c) A full adder is a digital circuit that performs addition. Full adders are implemented with logic gates in hardware.



full adder circuit.

2020
8.a) A binary counter can be used to generate a frequency diviiden by taking the output of any flip-flop and using it as a ~~clock~~ clock signal for another circuit.





frequency = f/2

2020

8.b output frequency = 512 kHz / 2^8

= 512/256

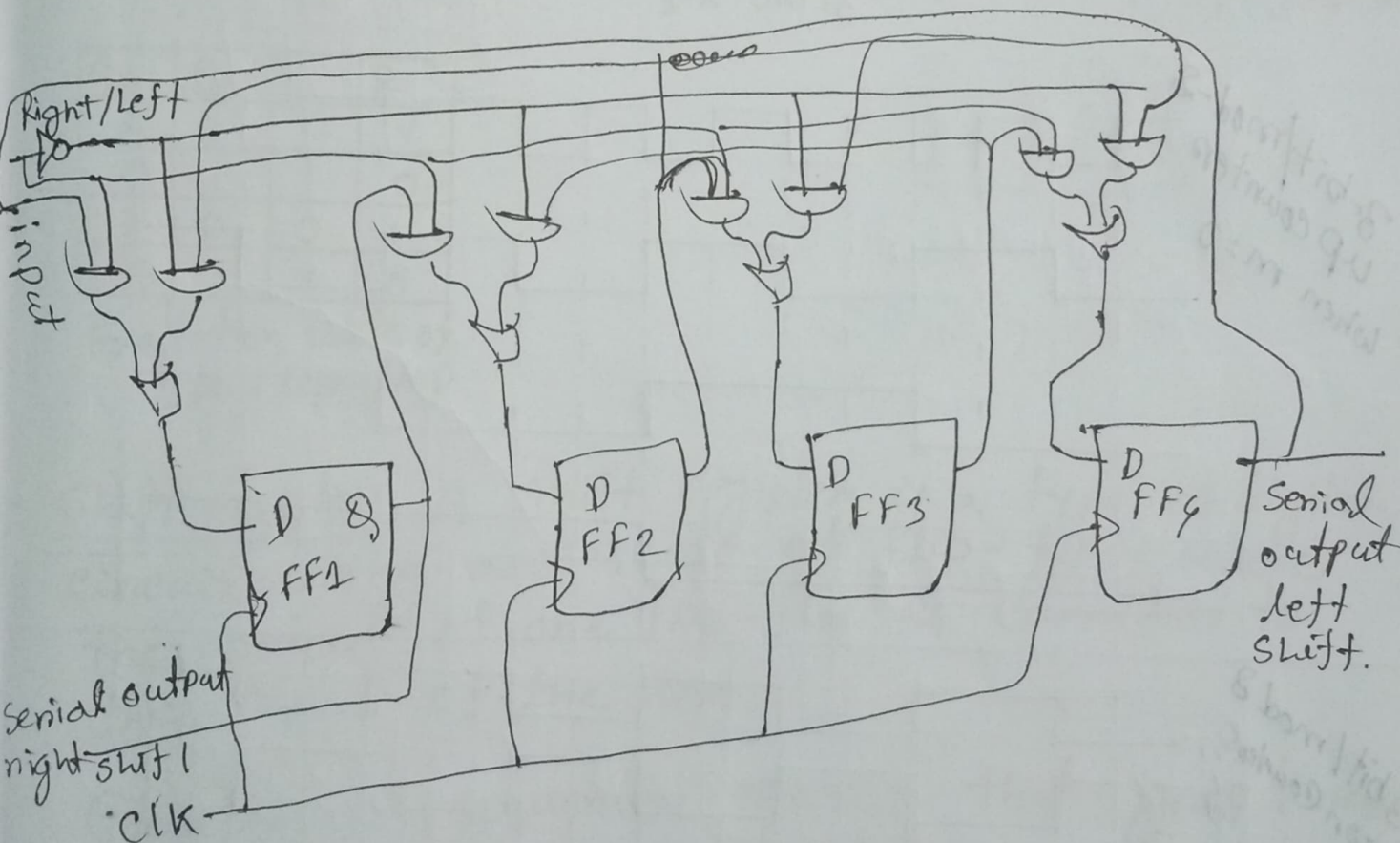
= 2 kHz ~~After 520 pulses~~

(Ans)

2020

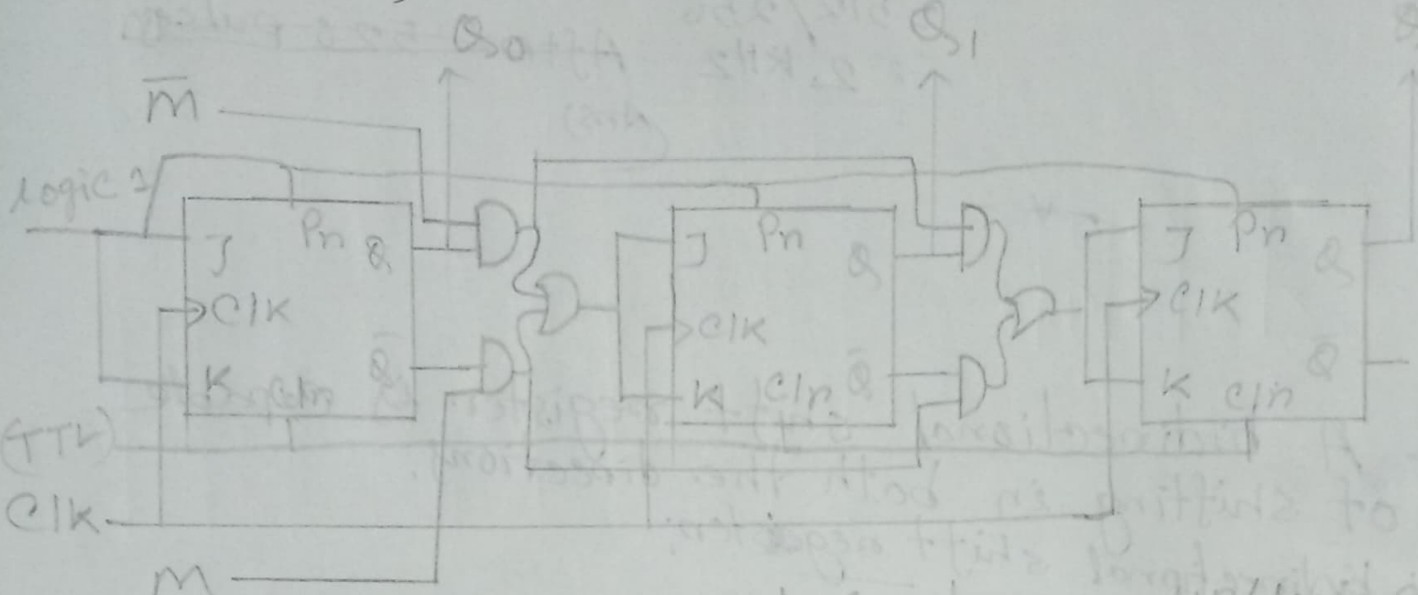
6.d A bidirectional shift register is capable of shifting in both the directions.

4 bit bidirectional shift register;

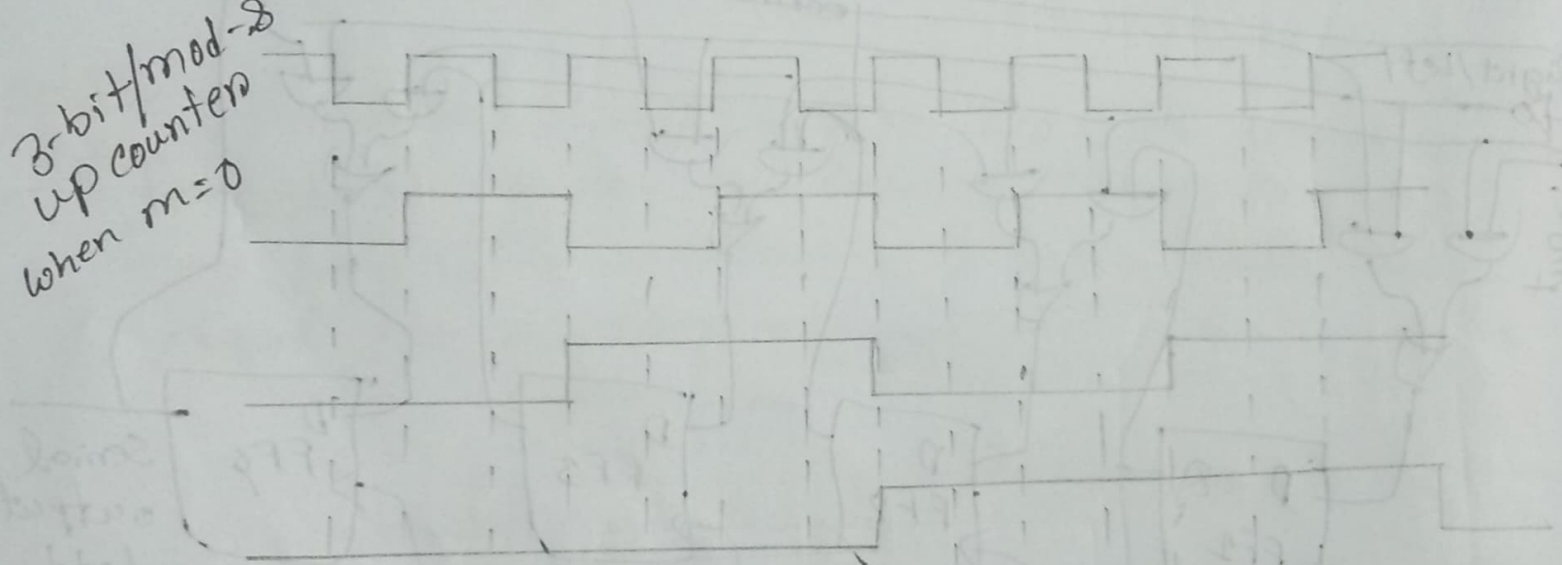


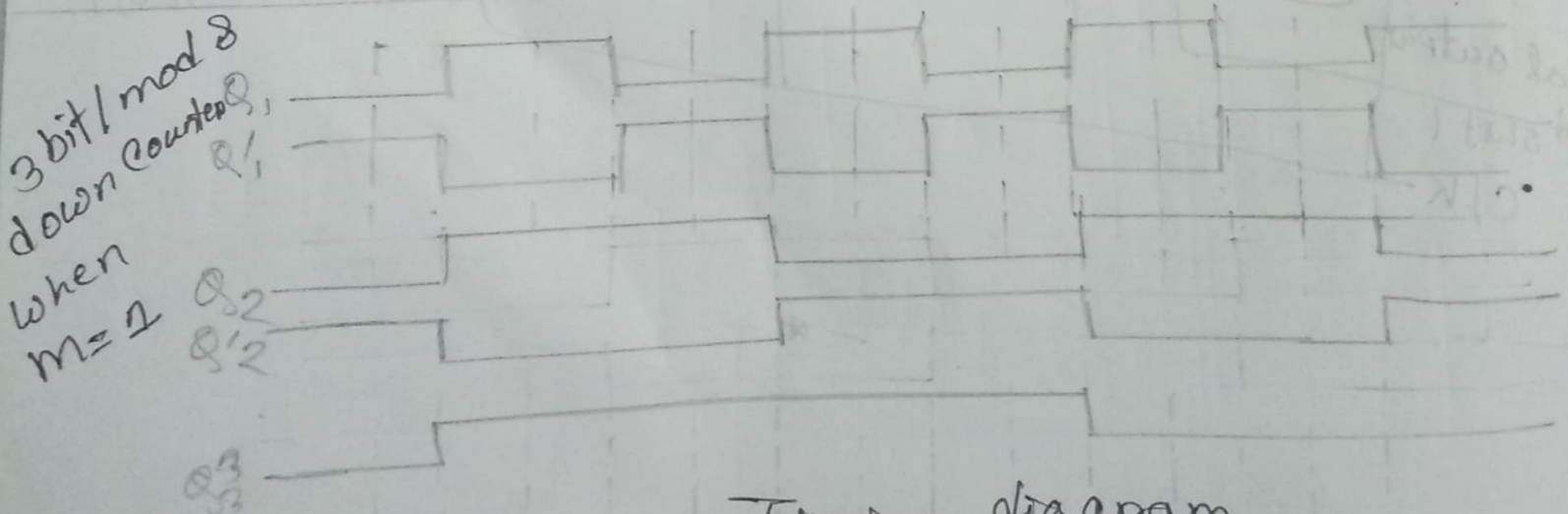
2018

5.c mod-8 Synchronous up/down counter that means 3-bit



3-bit Synchronous up/down counter





Timing diagram

2018

6.a) **Excitation table:** The excitation tables are used to determine the inputs of the flip-flop when the present state and the next state to which the flip-flop goes after the occurrence of the clock pulse are known.

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Excitation table of D-flip-flop

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Excitation table of JK flip-flop

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Excitation table of T flip-flop

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Excitation table of SR flip-flop

Shift register: A shift register is a type of digital circuit using a cascade of flip-flops where the output of one flip-flop is connected to the input of the next.

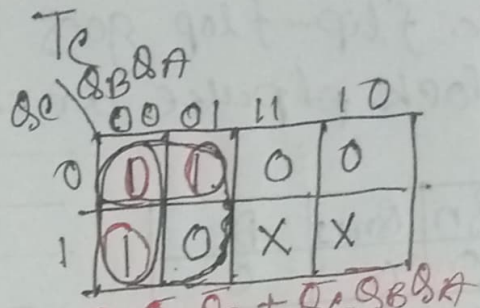
Counter: A sequential circuit that goes through a prescribed sequence of states upon the application of input pulses is called a counter.

2018
6.b

Count the sequence 0, 1, 5, 4, 3, 2, 0

$000 \rightarrow 001 \rightarrow 101 \rightarrow 100 \rightarrow 011 \rightarrow 010 \rightarrow 000$

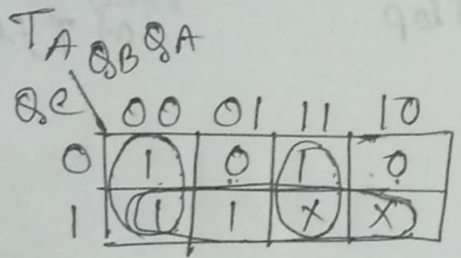
Q_C	Q_B	Q_A	$\bar{Q}_C$	$\bar{Q}_B$	$\bar{Q}_A$	T_C	T_B	T_A
0	0	0	0	0	1	0	0	1
0	0	1	1	0	1	1	0	1
0	1	0	0	0	0	0	1	0
0	1	1	0	1	0	0	0	1
1	0	0	0	1	1	1	1	1
1	0	1	1	0	0	0	0	1
1	1	0	X	X	X	X	X	X
1	1	1	X	X	X	X	X	X



$T_C = Q_C\bar{Q}_A\bar{Q}_B + Q_C\bar{Q}_B\bar{Q}_A$

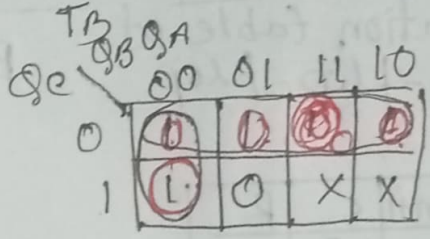
$T_C = Q_C\bar{Q}_B + \bar{Q}_B\bar{Q}_A$

$= \bar{Q}_B(Q_C + \bar{Q}_A)$



$T_A = Q_C + \bar{Q}_B\bar{Q}_A + Q_BQ_A$

$= Q_C + (\overline{Q_B \oplus Q_A})$



$T_B = \bar{Q}_C + \bar{Q}_B\bar{Q}_A$

$\bar{Q}_C\bar{Q}_B\bar{Q}_A + Q_C\bar{Q}_A\bar{Q}_B$