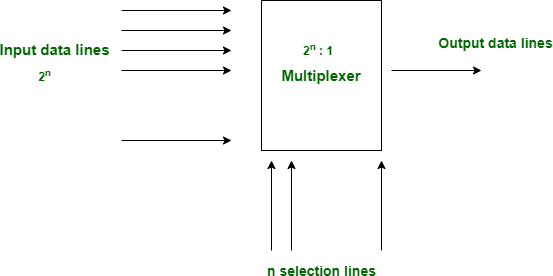
# NAME: DHVANI PATEL

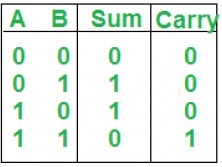
# ROLL NO: 21BCP116

# Assignment 7 : Application of Decoder, Recap of Logisim

* **Description:**

# Multiplexer:

* Multiplexer is a data selector which takes several inputs and gives a single output. In multiplexer we have 2n input lines and 1 output lines where n is the number of selection lines.



*Figure 1: The Block Diagram of Multiplexer*

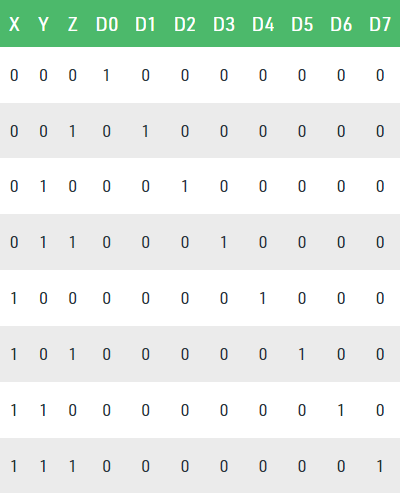
# Demultiplexer:

* Demultiplexer is a data distributor which takes a single input and gives several outputs. In demultiplexer we have 1 input and 2n output lines where n is the selection line.

# Lightbox

*Figure 2: The Block Diagram of Multiplexer*

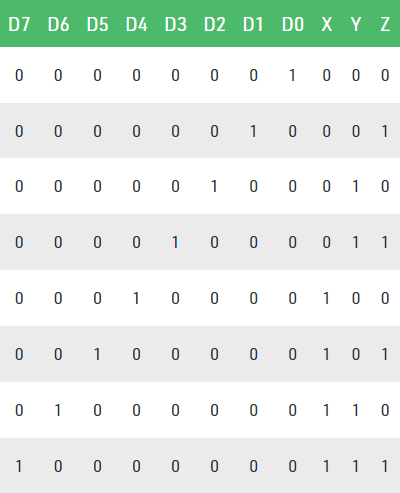
# Decoder:

* A decoder does the opposite job of an encoder. It is a combinational circuit that converts n lines of input into 2n lines of output.

*Figure 3: Truth Table of Decoder.*

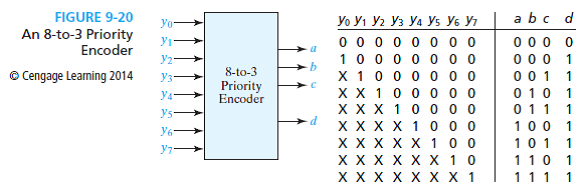
# Encoder:

* An encoder is a combinational circuit that converts binary information in the form of a 2N input lines into N output lines, which represent N bit code for the input. For simple encoders, it is assumed that only one input line is active at a time.



*Figure 3: Truth Table of Encoder.*

# 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. So when an input with a higher priority is present, all other inputs with a lower priority will be ignored.
* ****The priority encoder comes in many different forms with an example of an 8-input priority encoder along with its truth table shown below.

*Figure 2: The Block Diagram and Truth table of Priority encoder*

**Q 1. Write a Verilog code to implement BCD to seven segment display Decoder. Prepare the Truth Table and circuits and verify the same using Test Bench.**

* Code :

// Code your testbench here

module tb\_segment7 ;

reg [3:0] bcd;

wire [6:0] seg;

integer i;

// Instantiate the Unit Under Test (UUT)

segment7 uut (

.bcd(bcd),

.seg(seg)

);

//Apply inputs

initial begin

for(i = 0;i < 16;i = i+1) //run loop for 0 to 15.

begin

bcd = i;

#10; //wait for 10 ns3

end

end

initial

begin

$dumpfile ("dump.vcd") ;

$dumpvars (1) ;

end

endmodule

// Code your design here

module segment7(

bcd,

seg

);

//Declare inputs,outputs and internal variables.

input [3:0] bcd;

output [6:0] seg;

reg [6:0] seg;

//always block for converting bcd digit into 7 segment format

always @(bcd)

begin

case (bcd) //case statement

0 : seg = 7'b0000001;

1 : seg = 7'b1001111;

2 : seg = 7'b0010010;

3 : seg = 7'b0000110;

4 : seg = 7'b1001100;

5 : seg = 7'b0100100;

6 : seg = 7'b0100000;

7 : seg = 7'b0001111;

8 : seg = 7'b0000000;

9 : seg = 7'b0000100;

//switeh off 7 segment character when the bed digit is not a decimal number.

default : seg = 7'b1111111;

endcase

end

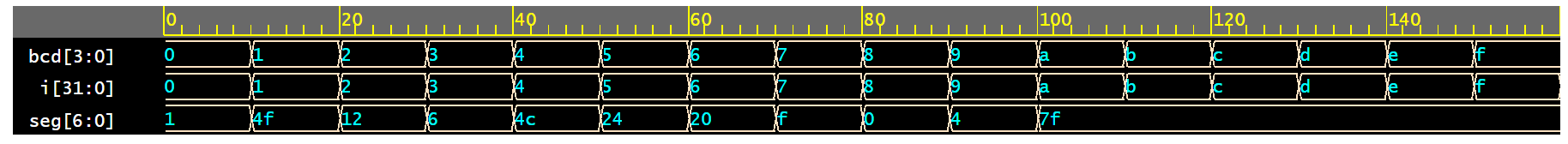
endmodule

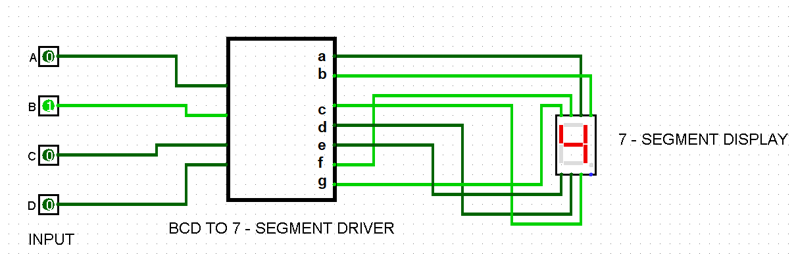
* Output :

VCD info: dumpfile dump.vcd opened for output.

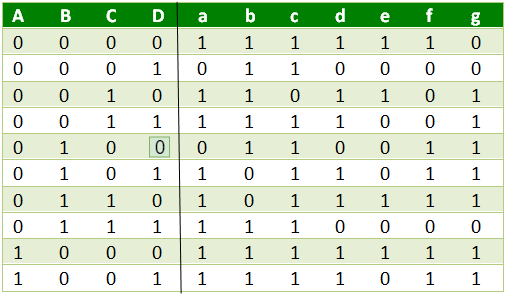
Finding VCD file...  
./dump.vcd

* Waveform :

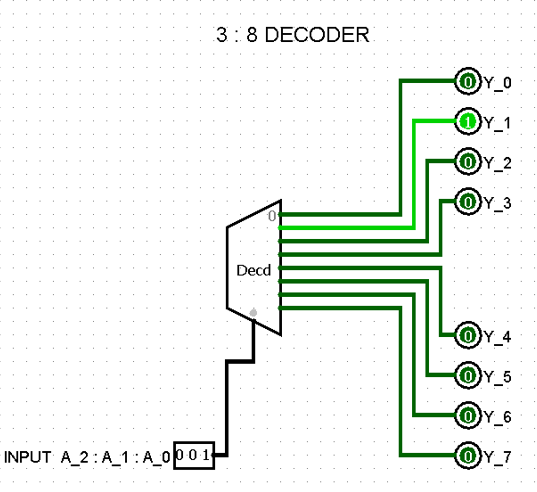




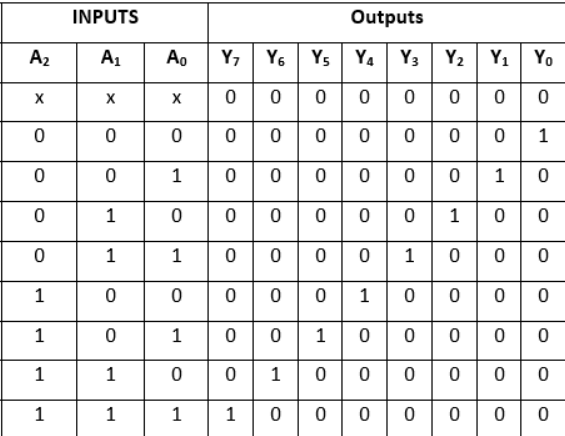
* **Truth Table :**



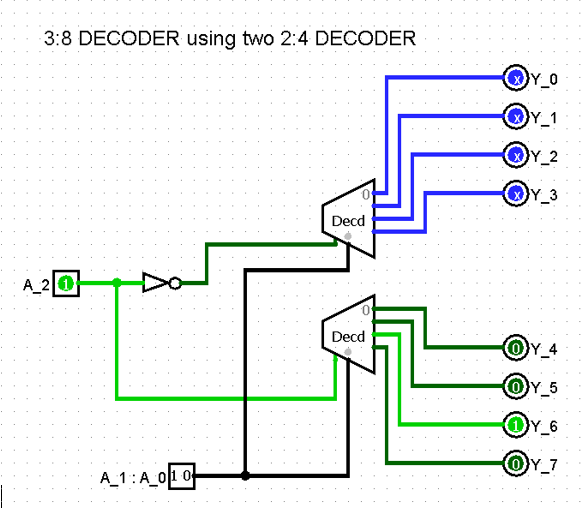
**Q 2 : Design a 3:8 Decoder using Logisim.**



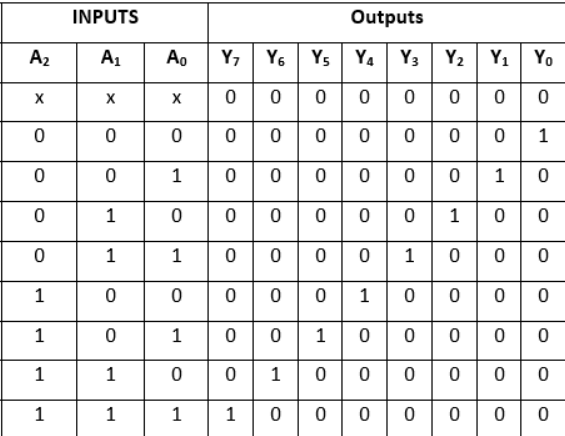
* **Truth Table :**



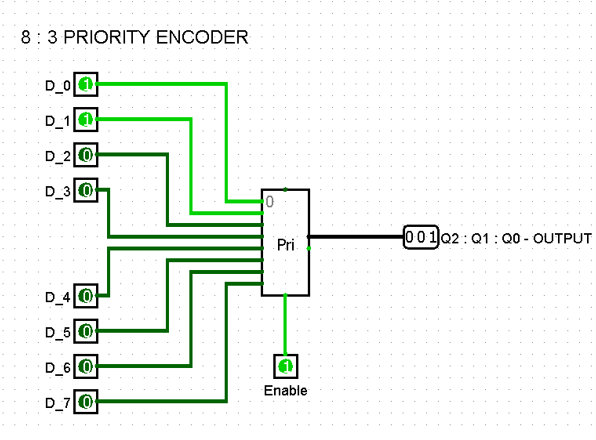
**Q 3 : Design a 3:8 Decoder using two 2:4 Decoder using Logisim.**



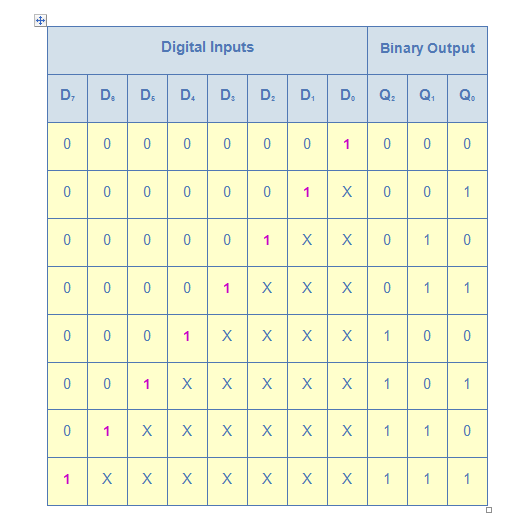
* **Truth Table :**



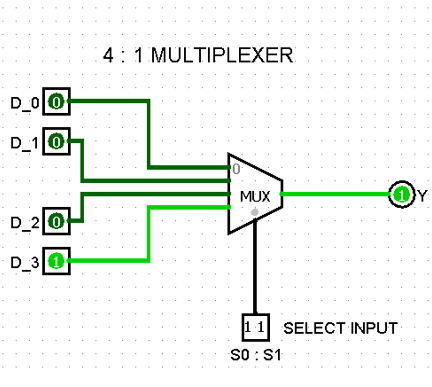
**Q 4 : Design an 8:3 Priority Encoder using Logisim.**



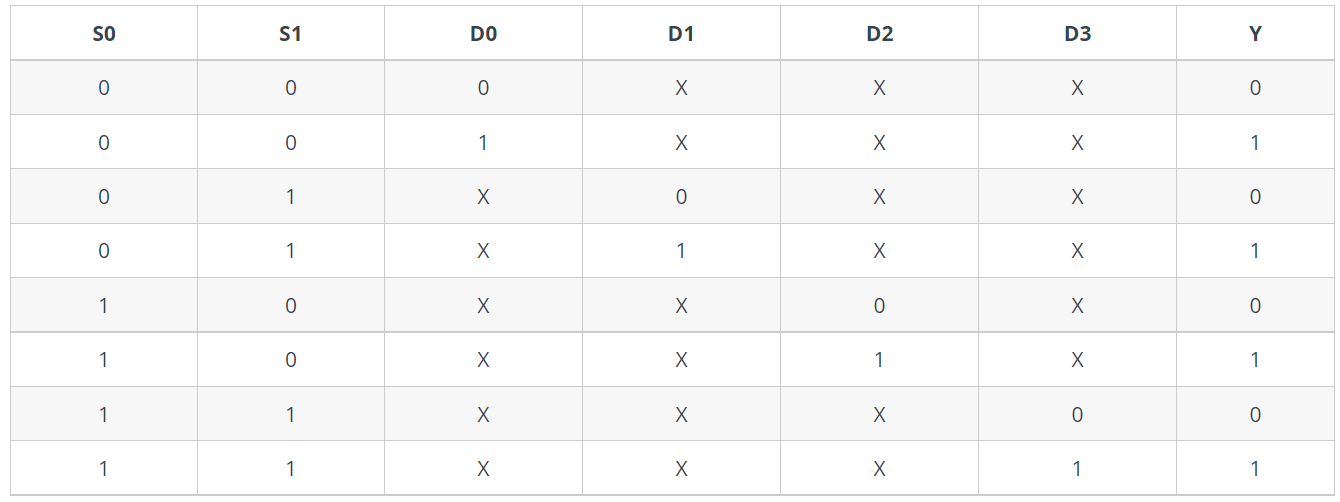
* **Truth Table :**



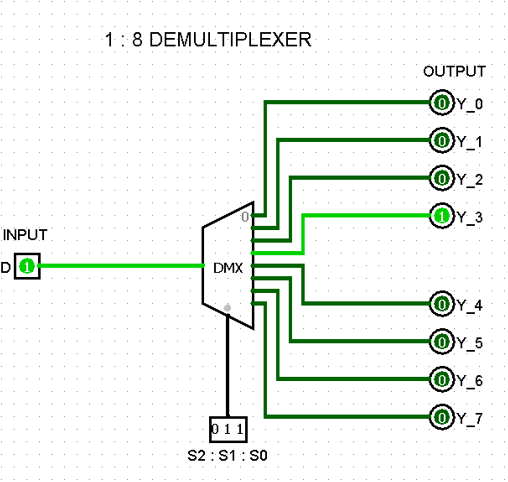
**Q 5 : Design a 4:1 Multiplexer using Logisim.**



* **Truth Table :**



**Q 6 : Design a 1:8 Demultiplexer using Logisim.**



* **Truth Table :**

