EAST WEST UNIVERSITY

Post Lab Report-05 & 06

Course Title: Digital Logic Design

Course Code: CSE345

Section: 04

Experiment 4

Submitted to:

Touhid Ahmed

Lecturer

Department of Computer Science & Engineering

Submitted by:

Name: Adri Saha

ID: 2019-1-60-024

Date of submission: 4 May 2021

Experiment Name: Priority Encoder, Decoder & Multiplexer, and their Use in Combinational Logic Implementation

Abstract:

Digital Logic Design is the one of the introductory courses for Electrical and Electronics Engineering students. It can introduce students to circuit design, problem solving, testing, and feature verification. In this experiment, students were asked to design and construct **Priority Encoder, Decoder & Multiplexer** and simulate the vector waveforms. They also verified the truth table with the simulated results. And they also learned behavioral Verilog coding of Priority Encoder, Decoder & Multiplexer using procedural model and continuous assign statement. Here, they also learned to implement and test encoder & decoder using discrete gates and get to know the uses of these. The digital logic design lab is a learning experience that most students enjoy because it gives them their first hands-on experience designing and constructing miniature structures.

Introduction:

Encoders are digital circuits that perform digital information decoding, while decoders are digital circuits that decode the coded digital information. An encoder with enable pins is called multiplexer while a decoder with enable pins is called demultiplexer. Discrete inputs to digital systems are encoded to binary-coded form by encoder circuits. In a reverse process, binary coded data are decoded to decoded to discrete information by decoder circuits. 2^n discrete symbols or elements of coded information can be represented by a binary code of n bits. Multiplexer means transmitting many information units over a smaller number of channels or lines.

Theory:

Binary Encoders:

The encoder is a circuit, a device, or a transducer. The encoder converts data from one format to another, such as electrical signals to counters or a PLC, for example.

A binary encoder, also known as a digital encoder, is a combinational circuit capable of converting 2^n input signals into n signals (such as BCD). In simple

terms, a Binary Encoder is a device that is used to encode binary codes.

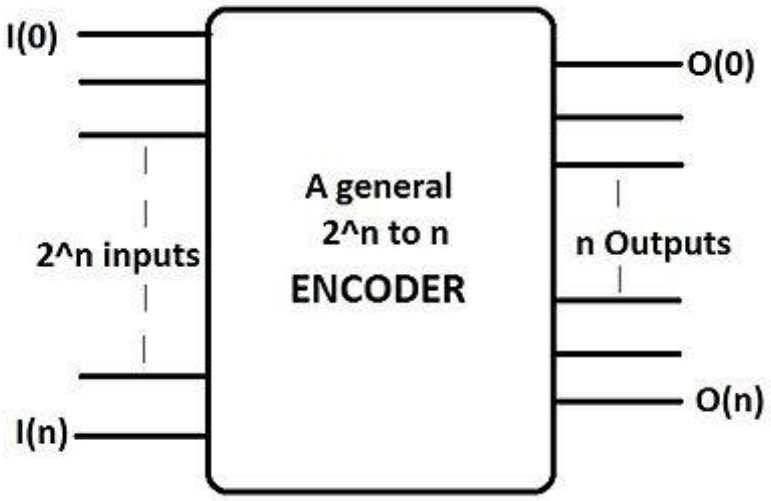


Figure 1: Example of binary encoder logic diagram

Uses:

Generally, we use an encoder when we need to process data from many data lines but only have a few input ports.

- Ensures that the unit to be controlled, such as a machine tool, does not exceed a preset position or direction of travel.
- In textiles it provides feedback for speed, direction, and distance for precisely controlled operations.
- Aerospace: Encoders provide high-precision motion feedback while operating in extreme environmental conditions.
- Conveying: Applied to drive motor, to a head-roll shaft to a pinch-roller, or to a lead screw.
- Autonomous Vehicles & Robots: Encoders provide precise, reliable motion feedback in automated robotics applications.
- Cut-to-length: Generates fixed number of pulses, sends them to the controller, which determines length of travel.
- Ball Screw Positioning: Attached to the end of the ball Screw shaft or drive motor, encoder provide motion feedback.
- Filling: Encoders ensure that the item to be filled is in the correct position before the filling mechanism activates.

- Food & Beverage: Provide feedback for processing, filling, packaging, material handling, labeling etc.
- Textiles: Provide critical feedback for speed, direction, and distance.
- Printing: Provide feedback in a variety of automated machinery used in the printing industry.
- X-Y positioning: Provide feedback on two axes of motion to determine X-Y coordinates.

Priority Encoder: The priority encoder's output corresponds to the highest-priority input that is currently active. As a result, any lower-priority inputs will be ignored if a higher-priority input is present.

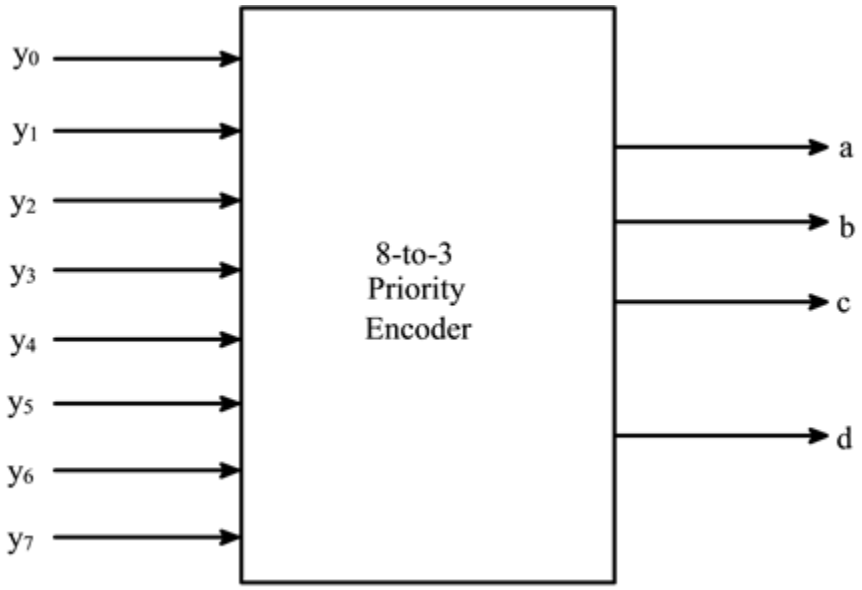


Figure 2: 8-to-3 Priority Encoder where Y_7 highest & Y_0 lowest priority

Uses: Microprocessor interrupt controllers are widely used in digital and computer systems to monitor the highest priority input.

- Used to reduce the number of wires used in circuits or applications with multiple inputs.
- Used to control interrupt requests by acting on the highest priority interrupt input.

- Useful for error checking: The binary encoder's drawback is solved by a priority encoder. It generates a coded output by prioritizing the bits of data. The values of the lower priority bits are irrelevant.
- Another more common application is in magnetic positional control as used on ships navigation or for robotic arm positioning etc.

Enable Input: A chip-enable input is a clock or strobe input that has a direct impact on the integrated circuit's power dissipation. It may be a dynamic memory's cycle control input or a static memory's power-reduction input, for example.

Binary Decoders: The decoder is a system that performs a series of signals into a code. A decoder is a circuit that converts binary data from n input lines to a maximum of 2^n unique output lines using a combinational circuit. When the decoder is activated, one of these outputs will be active High based on the combination of inputs present. That is, the decoder recognizes a specific code.

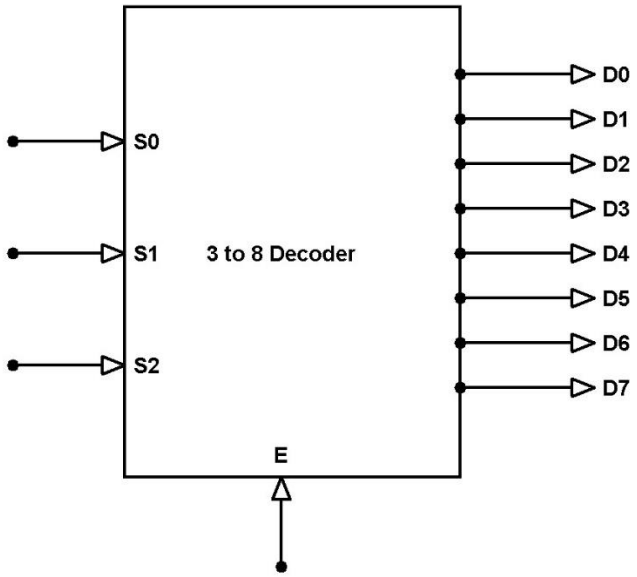


Figure 3: 3-to-8-line Decoder logic diagram

Uses:

- Mainly used in logical circuits, data transfer.
- In analog decoders, it's used in the analog to digital conversion.

- when the decoder is enabled. That means decoder detects a particular code.
- Used in electronic circuits to convert instructions into CPU control signals.

Multiplexer: A multiplexer is a system that accepts multiple inputs and outputs a single line. It selects between several analog or digital input signals and forwards the selected input to a single output line.

A simple example of an non-electronic circuit of a multiplexer is a single pole multi-position switch.

Selection Pin: The select pins control the inputs are connected to which outputs, as well as increasing the amount of data that can be sent over a network in each amount of time. It is also called data selector. If the input lines are n the selected pin will be ceil of $\log_2(n)$. In example: In 4-to-1 line multiplexer, selected pins are $\log_2(n)= \log_2(4)$ =ceil of 2. That means 2 selected pins S_0, S_1.

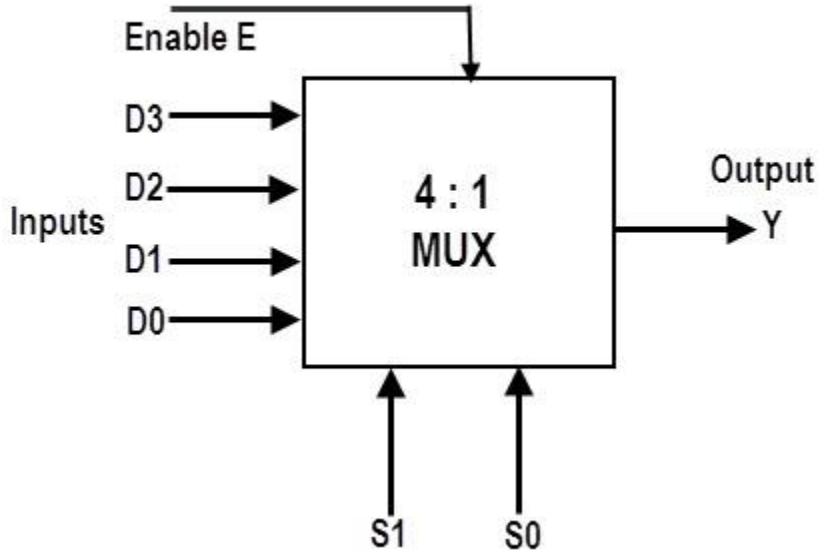


Figure 4: 4 to 1 line multiplexer

Uses:

- Used to maximize the amount of data that can be sent over a network in a specified amount of time with a given bandwidth.
- **Communication System:** Used to improve the communication system's efficiency by allowing data such as audio and video to be transmitted from different channels through cables and single lines.
- A digital multiplexer has digital input signals coming from multiple data-acquisition networks.
- **Computer Memory:** used in computer memory to store a large volume of data and reduce the number of copper lines required to link the memory to other components of the computer.
- **Telephone Network:** used in telecommunications networks to combine different audio signals on a single transmission line.
- **Transmission from a Satellite's Computer System:** GSM communication is used to relay data signals from a satellite's computer system to the ground system.

Objectives: Our main purpose is to design Priority Encoder, Binary Decoder and Multiplexer using discrete logic gates. Using Behavioral Verilog Simulation**,** students can implement these circuits and verify with truth table. Here they can also utilize their functions & utilize binary encoder, decoder & multiplexer in many other simulations & designs. Here in procedural statements, students can use if-else & case statements to design these.

- To implement 8 line to 3line Priority Encoder with active low input lines and active high output lines.
- To implement 4 line to 16
  line Binary Decoder with active low enable input, active low output lines and active high input lines and test combinational logic functions.
- To implement and test 12 line to 1
  line Multiplexer with active low enable input, active high input and output lines.
- To learn how to encoder, decoder & multiplexer works.
- Verify the designs with truth table.

Experimental results & discussions:

1. **Priority Encoder**: 8 line to 3-line Priority Encoder with active low input lines and active high output lines. Priority sequence is given as I2>I4>I0>I7>I5>I6>I1>I3

Here, inputs are active low which means when an input is active that time become it become 0 & for other inactive pins it is 0 on that time. Output lines are active high so 1 represent active output. No enable input is attached in the circuit. So, by following the priority sequence the truth table can be written like below:

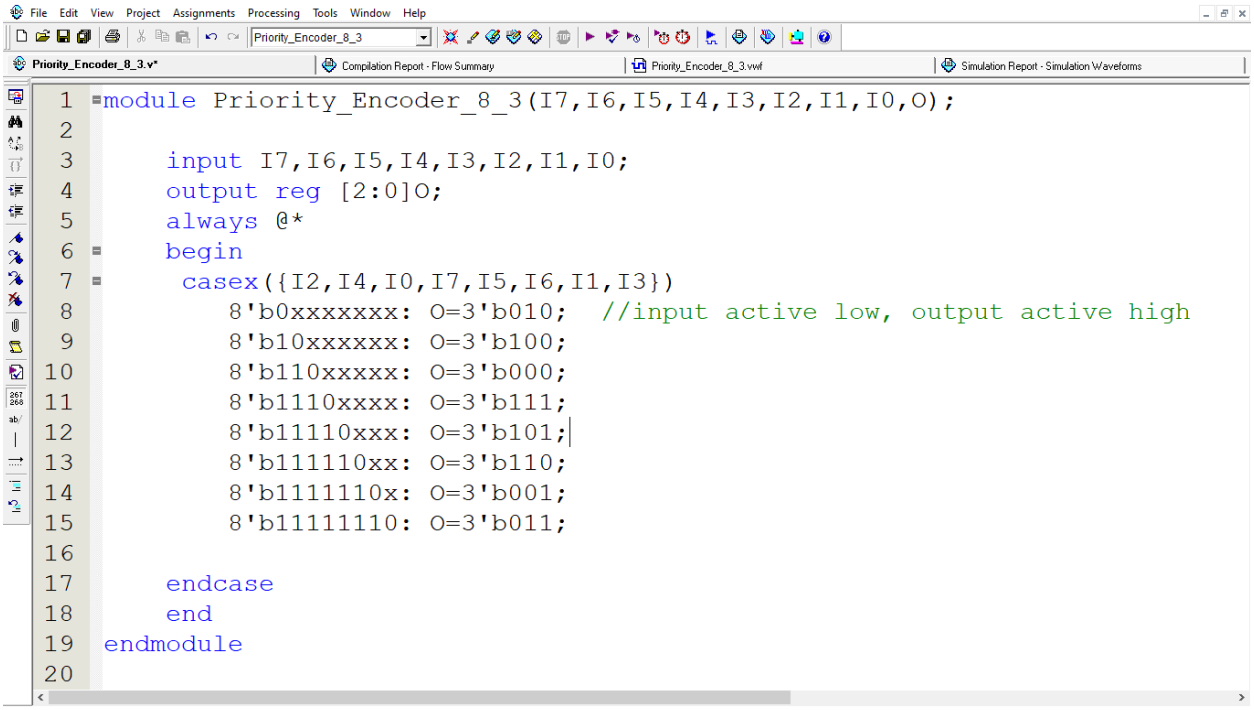
Truth Table: Here 8-to-3 priority encoder truth table where Priority sequence is I2>I4>I0>I7>I5>I6>I1>I3

Inputs								Outputs		
I7	I6	I5	I4	I3	I2	I1	I0	O2	O1	O0
0	X	X	X	X	X	X	X	0	1	0
1	0	X	X	X	X	X	X	1	0	0
1	1	0	X	X	X	X	X	0	0	0
1	1	1	0	X	X	X	X	1	1	1
1	1	1	1	0	X	X	X	1	0	1
1	1	1	1	1	0	X	X	1	1	0
1	1	1	1	1	1	0	X	0	0	1
1	1	1	1	1	1	1	0	0	1	1

Figure 5: 8-to-3 priority encoder

In this table, X represents don't care term when output is not affected by these inputs.

Verilog Code: In this code, we used case statement for each binary input & binary outputs.



```
1  module Priority_Encoder_8_3(I7,I6,I5,I4,I3,I2,I1,I0,O);
2
3      input I7,I6,I5,I4,I3,I2,I1,I0;
4      output reg [2:0]O;
5      always @*
6      begin
7       casex({I2,I4,I0,I7,I5,I6,I1,I3})
8          8'b0xxxxxxx: O=3'b010;  //input active low, output active high
9          8'b10xxxxxx: O=3'b100;
10         8'b110xxxxx: O=3'b000;
11         8'b1110xxxx: O=3'b111;
12         8'b11110xxx: O=3'b101;
13         8'b111110xx: O=3'b110;
14         8'b1111110x: O=3'b001;
15         8'b11111110: O=3'b011;
16
17     endcase
18     end
19 endmodule
20
```

Figure 6: 8 line to 3-line priority encoder

Waveform Simulation:

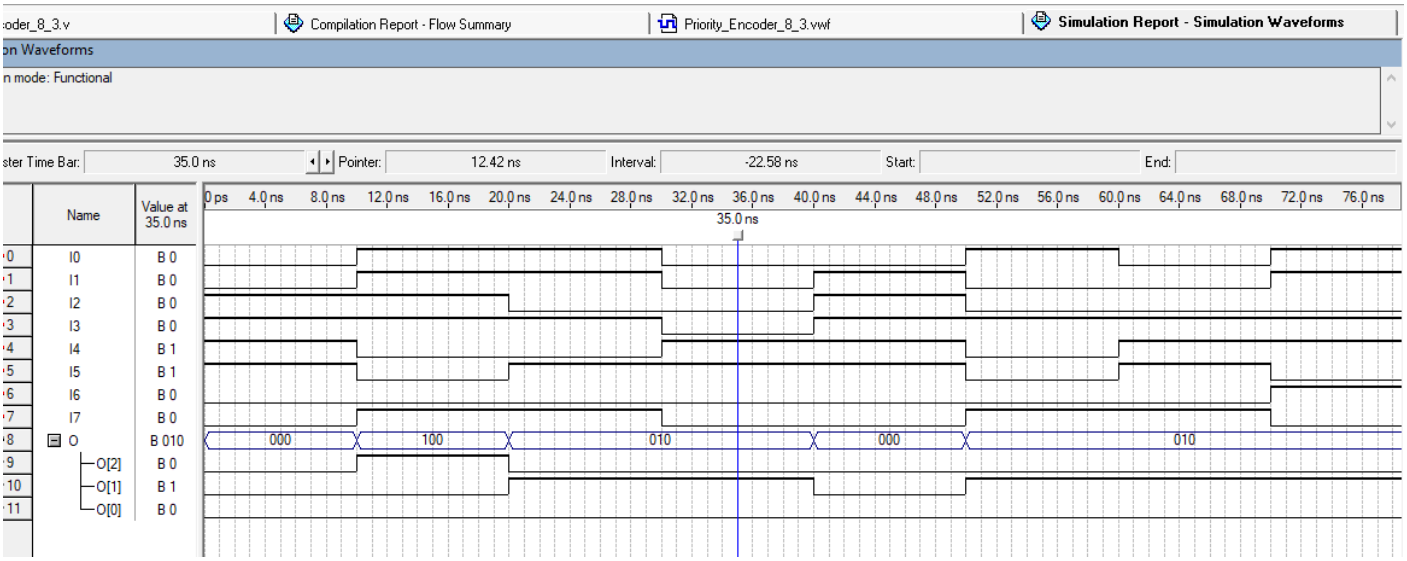


Figure 7: 8 line to 3 priority encoder simulation

In this simulation, end time was taken 80 ns to check at least 8 values. Inputs are taken in random value of fixed interval 10ns for each node. Here. We can verify the simulation by checking outputs. We can see, in the 2nd output where 3 input lines were active which was I_4, I_5, I_6. Among them I_4 has highest priority that's why

output shows 100 and no 2 line is active which represents decimal no 4(I_4). Rest other are in same way. So we can verify the simulation also by checking with above truth table.

2. **4 line to 16 binary Decoder:** 4 line to 16-line Binary Decoder with active low enable input, active low output lines and active high input lines.

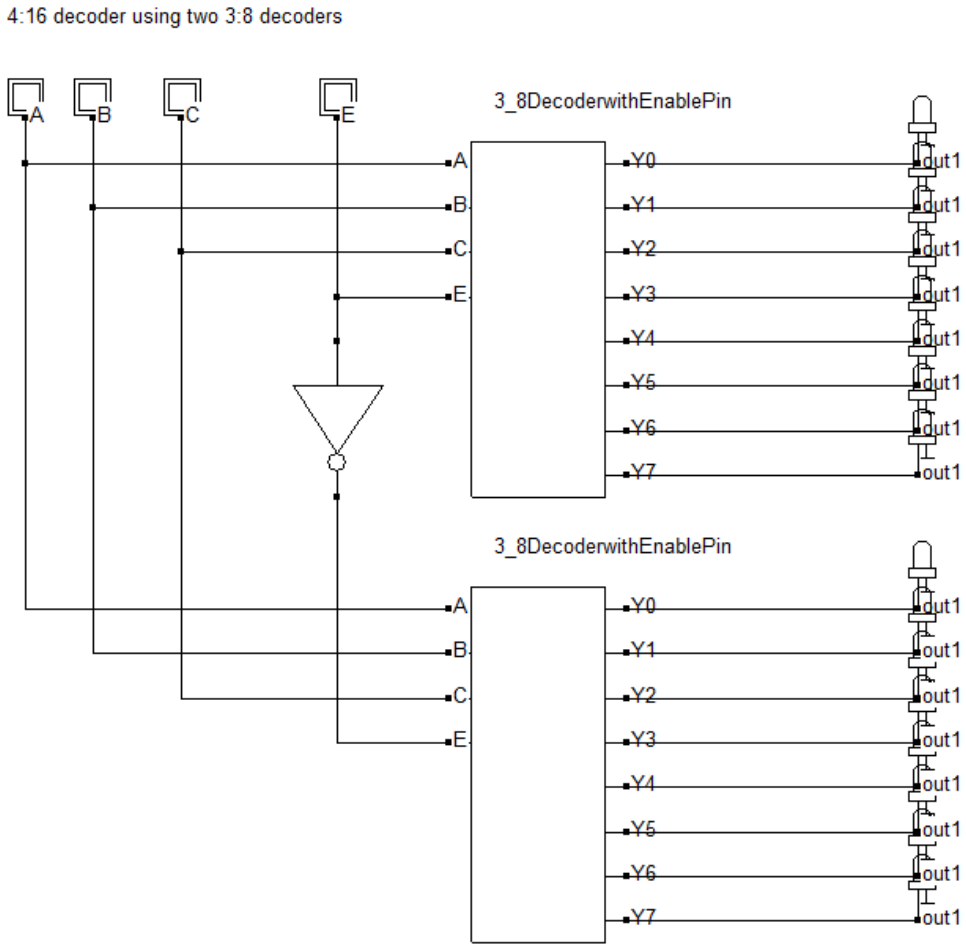


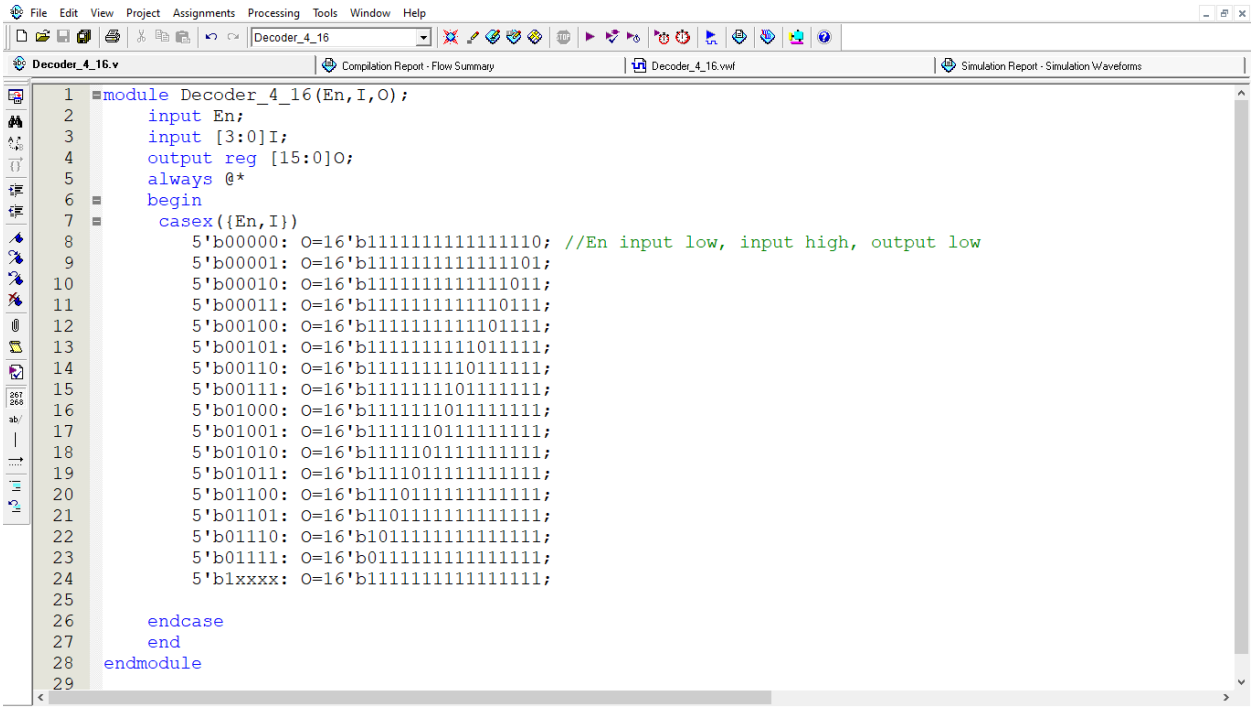
Figure 8: 4 line to 16 binary Decoder

Truth Table: Inputs are active high, but enable input and output are active low. So by following this the truth table is:

Input					Output														
En	I3	I2	I1	I0	O_{15}	O_{14}	O_{13}	O_{12}	O_{11}	O_{10}	O_9	O_8	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
0	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1
0	1	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1
0	1	0	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	X	X	X	X	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Figure 9: 4 line to-6 binary decoder truth table

Verilog Code: Here we also used procedural case statement to implement 4 line to 16 decoders. Here, by adding a enable input reduced more 16 cases. Enable input was low that is why when enable input became high or 1 all outputs are inactive. When Input is high output become active with low or 0. As enable input is there, so input became 4+1=5 binary digits number. We can see in each row, for a single input, 1 line is active in output and others are 0. So that is the default case for all high enable input which is 0.



```
1  module Decoder_4_16(En,I,O);
2      input En;
3      input [3:0]I;
4      output reg [15:0]O;
5      always @*
6      begin
7       casex({En,I})
8          5'b00000: O=16'b1111111111111110; //En input low, input high, output low
9          5'b00001: O=16'b1111111111111101;
10         5'b00010: O=16'b1111111111111011;
11         5'b00011: O=16'b1111111111110111;
12         5'b00100: O=16'b1111111111101111;
13         5'b00101: O=16'b1111111111011111;
14         5'b00110: O=16'b1111111110111111;
15         5'b00111: O=16'b1111111101111111;
16         5'b01000: O=16'b1111111011111111;
17         5'b01001: O=16'b1111110111111111;
18         5'b01010: O=16'b1111101111111111;
19         5'b01011: O=16'b1111011111111111;
20         5'b01100: O=16'b1110111111111111;
21         5'b01101: O=16'b1101111111111111;
22         5'b01110: O=16'b1011111111111111;
23         5'b01111: O=16'b0111111111111111;
24         5'blxxxx: O=16'b1111111111111111;
25
26     endcase
27     end
28 endmodule
29
```

Figure 10: 4 line to-16 binary decoder Verilog code

Waveform Simulation:

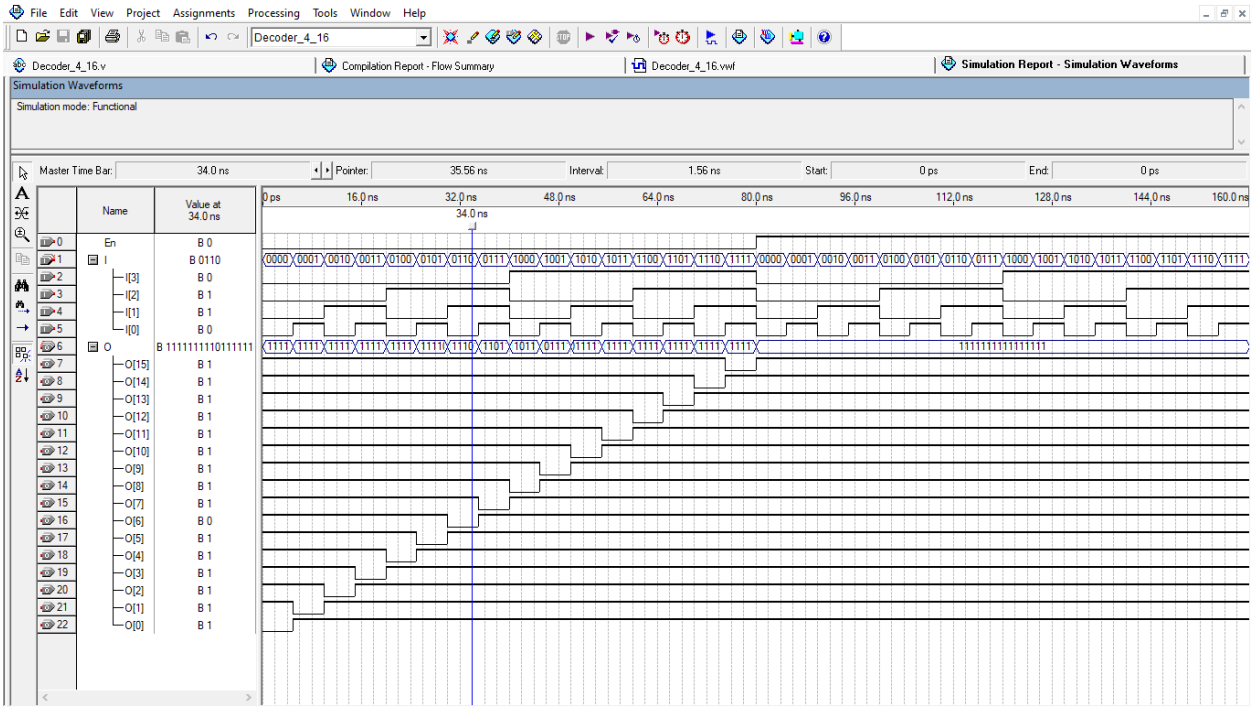


Figure: 4 line to-16 binary decoder waveform simulation

In this simulation, end time was taken 160ns. We can see after 80ns no output line is active. It is because enable input is 1 after 160/2=80 ns as we have taken 160 ns end time. Then in input lines time period was taken 80,40,20 & 10 ns for I_3, I_2, I_1, I_0 sequentially by using override clock. So now if we verify the output, we can see O_0, O_1, O_2…… consequently become active for each inputs and others are inactive. So, the decoder has been simulated well. And we find no error in this design.

3. **12 line to 1 line Multiplexer:** 12 line to 1 line Multiplexer with active low enable input, active high input, and output lines.

In 12-to-1 line multiplexer, selected pins are $\log_2(n)= \log_2(12)$ =ceil of 3.584. That means 4 selected pins: S_3, S_2, S_1, S_0 and Output is O.

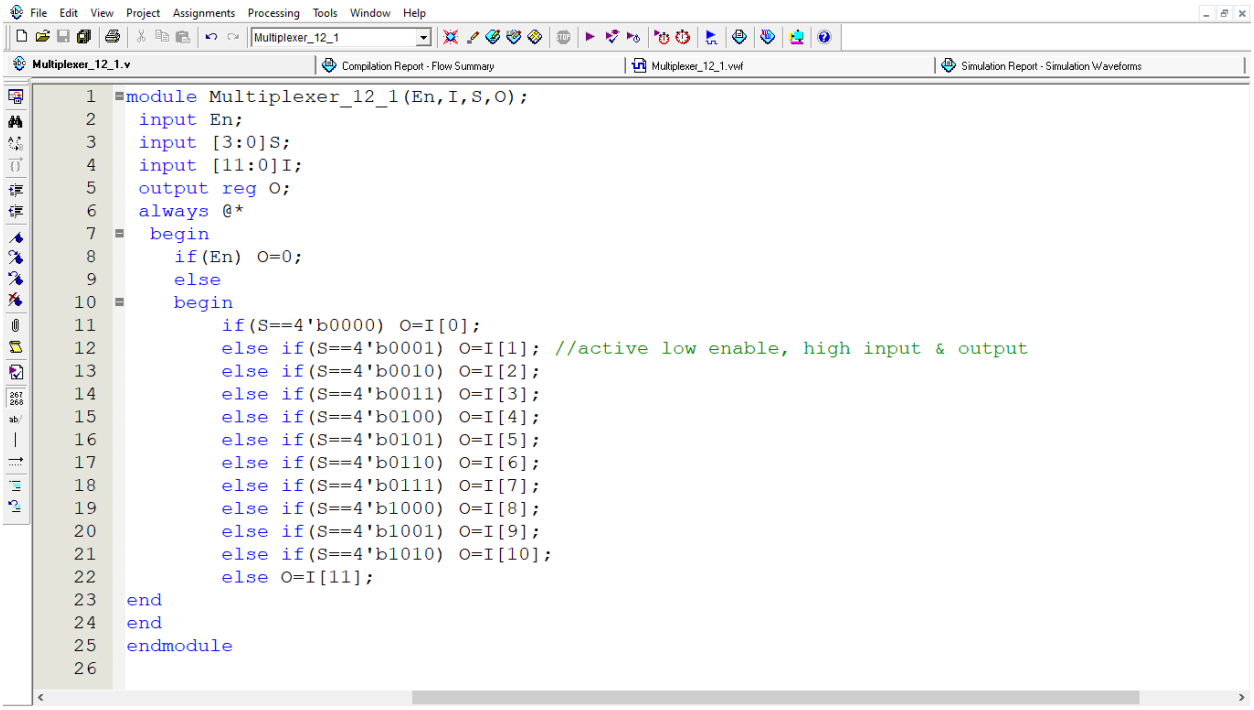
So, the truth table is:

12-to-1 line multiplexer (Enable input active low & input, output both are active high)

Input					Output
En	S_3	S_2	S_1	S_0	O
0	0	0	0	0	I_0
0	0	0	0	1	I_1
0	0	0	1	0	I_2
0	0	0	1	1	I_3
0	0	1	0	0	I_4
0	0	1	0	1	I_5
0	0	1	1	0	I_6
0	0	1	1	1	I_7
0	1	0	0	0	I_8
0	1	0	0	1	I_9
0	1	0	1	0	I_{10}
0	1	0	1	1	I_{11}

Figure: 12 line to 1-line Multiplexer truth table

Verilog Codes: Here, I used procedural if-else statement to design 12 to 1 multiplexer. In this code selection pin,4+enable input,1= total 5 inputs. As enable input was low, so the first condition is if enable input==true or 1, output is 0. Then in else statement when enable input is low or 0, there is another if else block for different outputs.



```
1  module Multiplexer_12_1(En,I,S,O);
2   input En;
3   input [3:0]S;
4   input [11:0]I;
5   output reg O;
6   always @*
7    begin
8      if(En) O=0;
9      else
10     begin
11         if(S==4'b0000) O=I[0];
12         else if(S==4'b0001) O=I[1]; //active low enable, high input & output
13         else if(S==4'b0010) O=I[2];
14         else if(S==4'b0011) O=I[3];
15         else if(S==4'b0100) O=I[4];
16         else if(S==4'b0101) O=I[5];
17         else if(S==4'b0110) O=I[6];
18         else if(S==4'b0111) O=I[7];
19         else if(S==4'b1000) O=I[8];
20         else if(S==4'b1001) O=I[9];
21         else if(S==4'b1010) O=I[10];
22         else O=I[11];
23  end
24  end
25  endmodule
26
```

Figure: 12 line to 1 line Multiplexer Verilog code

Waveform Simulation:

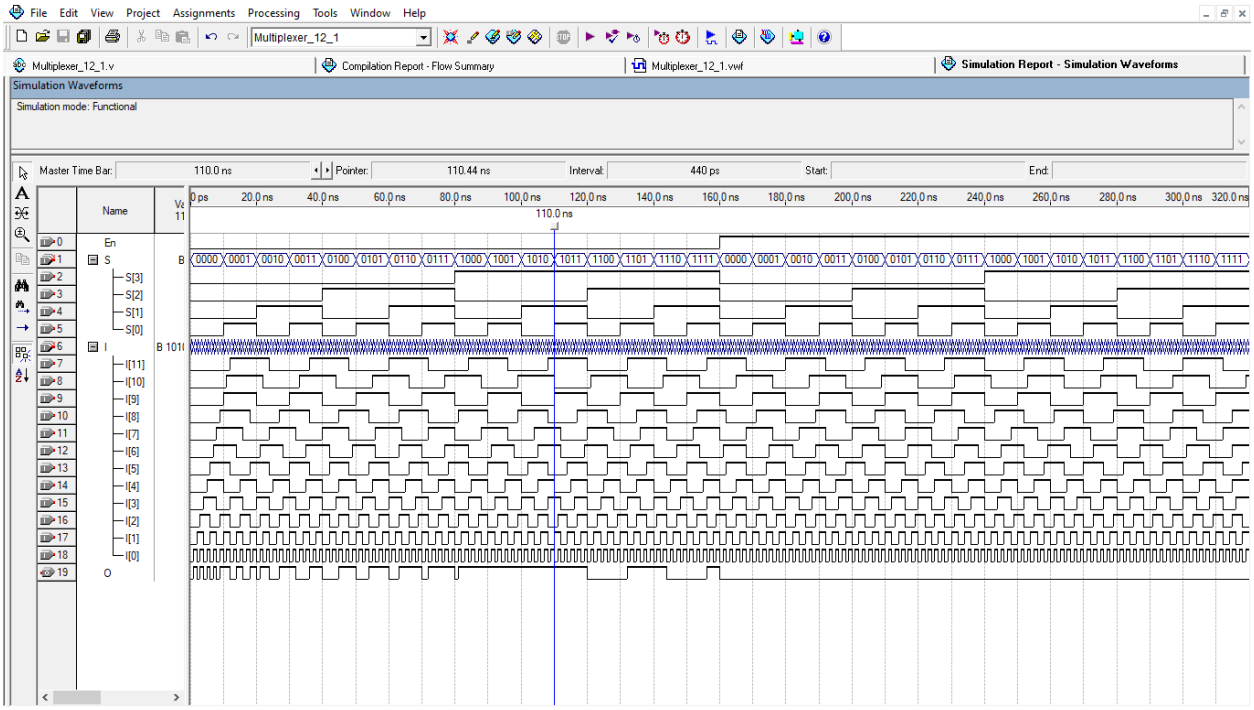


Figure: 12 line to 1 line Multiplexer waveform simulation

In this simulation, the end time was taken 320 ns as there were 12 inputs. Time period of enable input is 320 ns, then time period of selection pins S_3, S_2, S_1, S_0 was 160,80,40,20ns. Then time period of inputs I_0, I_1, …. ,I_{11} were taken consequently 2,4,6,8,10,12,14,16,18,20,22,24 ns. Now, if we simulate the waveform we can see in results, output's particular time is match for each input lines. Like, for 0-10 ns output shows like I_0, then from 10-20 ns output shows similar waveform of I_1, …. And so on. After 160 ns the output was inactive as enable input is 1 on that time period. So, Output shows 0 on that period. Before that it resembles all input line for a particular time. So, we can verify the design by this and truth table.

So, after all this simulation of design we can find the perfect result for each design. So, we can say there is no error doing this experiment.

Discussion:

So, after all this simulation of design we can find the perfect result for each design. Moreover, all the designs implemented by procedural behavioral Verilog codes. We use if else & case statement to implement these designs. All the outputs have given the correct result with truth tables. That is why, we can say there is no error doing this experiment.

Conclusion:

From this experiment, students learned to design encoder, decoder & multiplexer circuit by using behavioral Verilog code. They also learned to implement priority encoder using procedural Verilog code. They also learn to design multiplexer using discrete logic gates & encoder, decoder. Finally, students would be able to verify their logic design by truth tables. They also get to know the uses of encoders & decoders. The experiment verified his/her hands-on experience by simulating these designs in Quartus Software.

References:

1. Book: Fundamentals of Digital Logic with Verilog Design by Stephen Brown, Zvonko Vranesic-McGraw-Hill Science_Engineering_Math (2013)
2. https://www.electronics-tutorials.ws/combination/comb_4.html
3. https://www.encoder.com/encoder-applications
4. https://www.watelectronics.com/what-is-multiplexer-and-types/