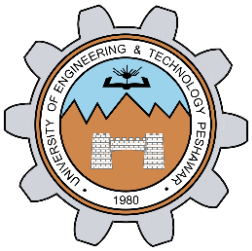
DECODER IN VERILOG

LAB # 08



Fall 2023

CSE-304L Computer Organization and Architecture Lab

Submitted by: **Ali Asghar**

Registration No.: **21PWCSE2059**

Class Section: **C**

"On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work."

Submitted to:

Dr. Bilal Habib

Date:

17th December 2023

Department of Computer Systems Engineering

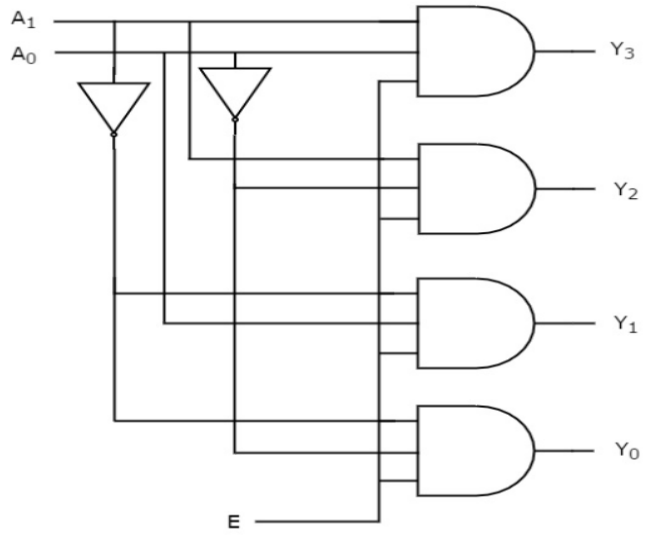
University of Engineering and Technology, Peshawar

ASSESSMENT RUBRICS COA LABS

LAB REPORT ASSESSMENT				
Criteria	**Excellent**	**Average**	**Nill**	**Marks Obtained**
1. Objectives of Lab	All objectives of lab are properly covered [Marks 10]	Objectives of lab are partially covered [Marks 5]	Objectives of lab are not shown [Marks 0]	
2. MIPS instructions with Comments and proper indentations.	All the instructions are well written with comments explaining the code and properly indented [Marks 20]	Some instructions are missing are poorly commented code [Marks 10]	The instructions are not properly written [Marks 0]	
3. Simulation run without error and warnings	The code is running in the simulator without any error and warnings [Marks 10]	The code is running but with some warnings or errors. [Marks 5]	The code is written but not running due to errors [Marks 0]	
4. Procedure	All the instructions are written with proper procedure [Marks 20]	Some steps are missing [Marks 10]	steps are totally missing [Marks 0]	
5. OUTPUT	Proper output of the code written in assembly [Marks 20]	Some of the outputs are missing [Marks 10]	No or wrong output [Marks 0]	
6. Conclusion	Conclusion about the lab is shown and written [Marks 20]	Conclusion about the lab is partially shown [Marks 10]	Conclusion about the lab is not shown[Marks0]	
7. Cheating			Any kind of cheating will lead to 0 Marks	
Total Marks Obtained:_____ Instructor Signature: _____				

Task 1:

Implement 2X4 Decoder using gate level modeling in Verilog.



Code:

DUT Code:

tb_task2.v | task2.v | task3.v | 2x4_Decoder.v | tb_2x4_Decoder.v | 3x8_Decoder.v | tb_3x_Decoder

```
1  module Decoder_2x4(A, B, E, I);
2
3      input A;
4      input B;
5      input E;
6      output [3:0]I;
7
8      wire nA;
9      wire nB;
10
11     not n1(nA, A);
12     not n2(nB, B);
13
14     and a1(I[0], nA, nB, E);
15     and a2(I[1], nA, B, E);
16     and a3(I[2], A, nB, E);
17     and a4(I[3], A, B, E);
18 endmodule
19
```

Test bench Code:

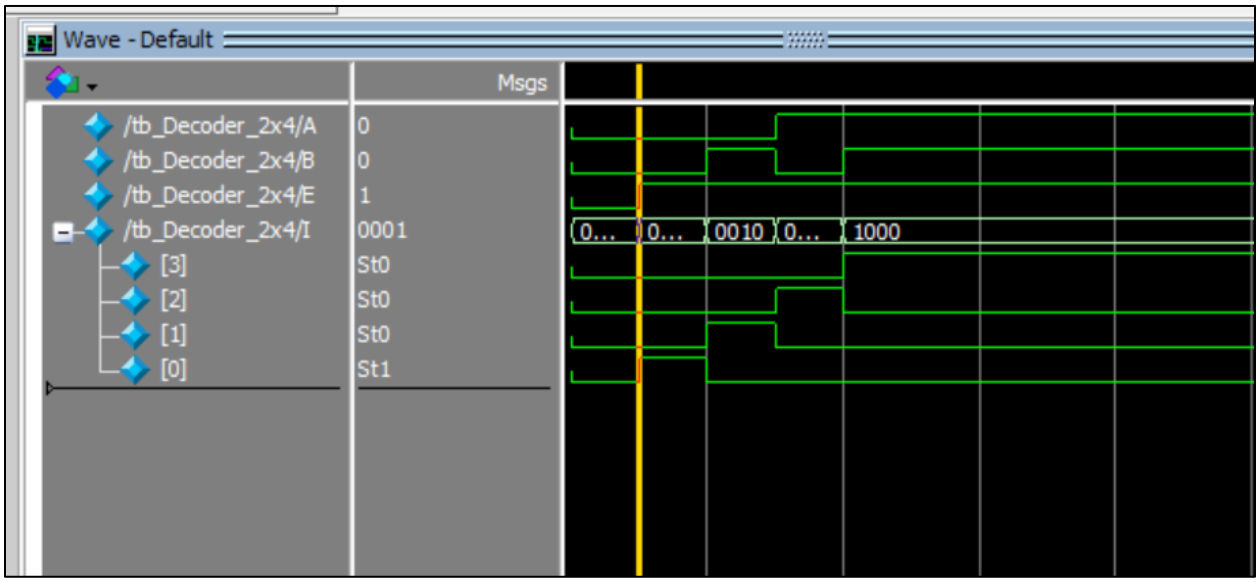
tb_task2.v | task2.v | task3.v | 2x4_Decoder.v | tb_2x4_Decoder.v | 3x8_Decoder.v | tb_3x_Decoder.v | task3.v | tb_tas

```
1  module tb_Decoder_2x4();
2
3      reg A;
4      reg B;
5      reg E;
6      wire [3:0]I;
7
8      Decoder_2x4 my_2x4_Decoder(A,B,E,I);
9
10     initial begin
11     A = 0;
12     B = 0;
13     E = 0;
14     $display("%b %b %b %b", A , B, E, {I[0], I[1], I[2], I[3]});
15     #10
16
17     A = 0;
18     B = 0;
19     E = 1;
20     $display("%b %b %b %b", A , B, E, {I[0], I[1], I[2], I[3]});
21     #10
22
23     A = 0;
```

tb_task2.v | task2.v | task3.v | 2x4_Decoder.v | tb_2x4_Decoder.v | 3x8_Decoder.v | tb_3x_Decoder.v | task3.v | tb_ta

```
21     #10
22
23     A = 0;
24     B = 1;
25     E = 1;
26     #10
27
28     $display("%b %b %b %b", A , B, E, {I[0], I[1], I[2], I[3]});
29     A = 1;
30     B = 0;
31     E = 1;
32     $display("%b %b %b %b", A , B, E, {I[0], I[1], I[2], I[3]});
33     #10
34
35     A = 1;
36     B = 1;
37     E = 1;
38     $display("%b %b %b %b", A , B, E, {I[0], I[1], I[2], I[3]});
39
40     end
41
42 endmodule
```

Output:



Task 2:

Implement 3X8 Decoder using gate level modeling in Verilog.

Code:

DUT Code:

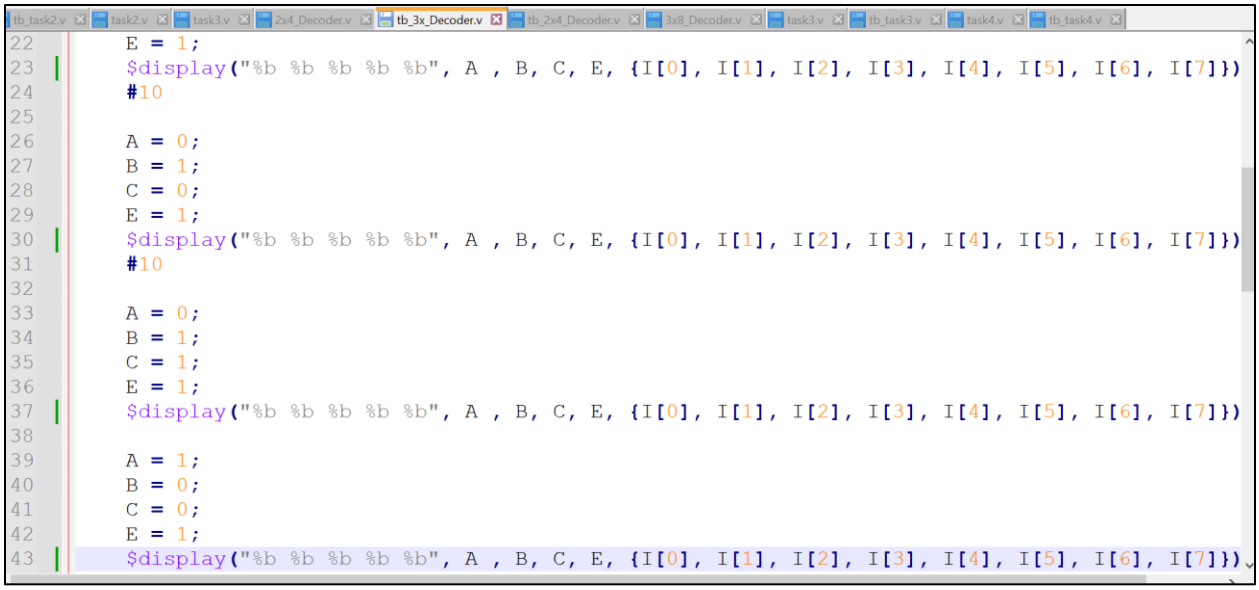
task2.v | task3.v | 2x4_Decoder.v | tb_2x4_Decoder.v | 3x8_Decoder.v | tb_3x_Decoder.v

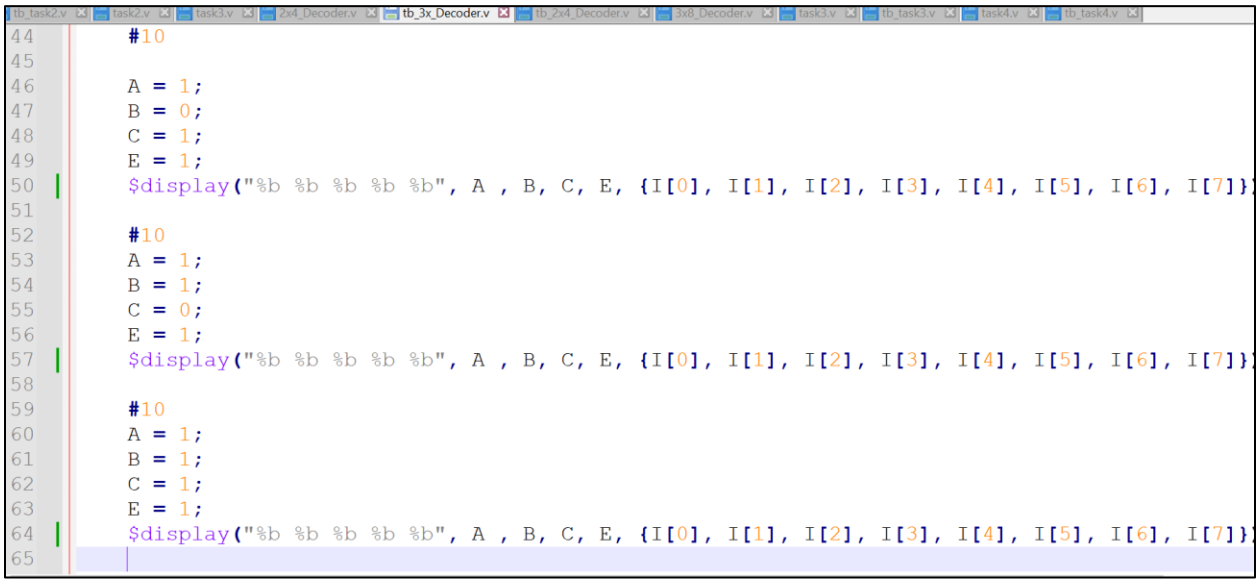
```
module Decoder_3x8(A, B, C, E, I);
    input A;
    input B;
    input C;
    input E;
    output [7:0]I;

    wire nA;
    wire nB;
    wire nC;
```

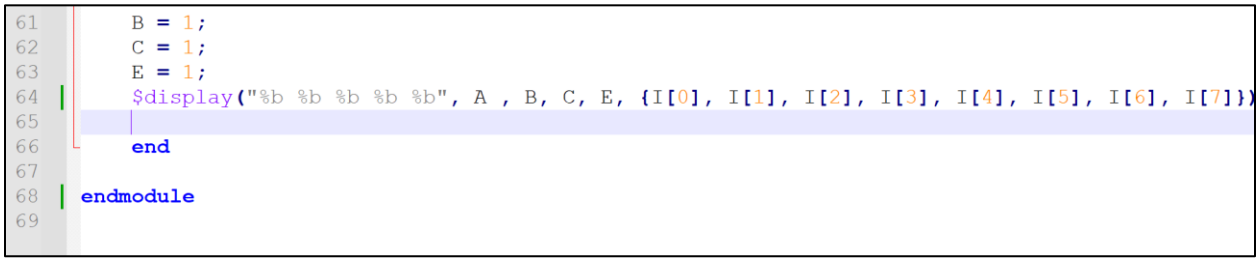
```
12      not n1(nA, A);
13      not n2(nB, B);
14      not n3(nC, C);
15
16      and a1(I[0], nA, nB, nC, E);
17      and a2(I[1], nA, nB, C, E);
18      and a3(I[2], nA, B, nC, E);
19      and a4(I[3], nA, B, C, E);
20      and a5(I[4], A, nB, nC, E);
21      and a6(I[5], A, nB, C, E);
22      and a7(I[6], A, B, nC, E);
23      and a8(I[7], A, B, C, E);
24
25  endmodule
```

Test bench Code:

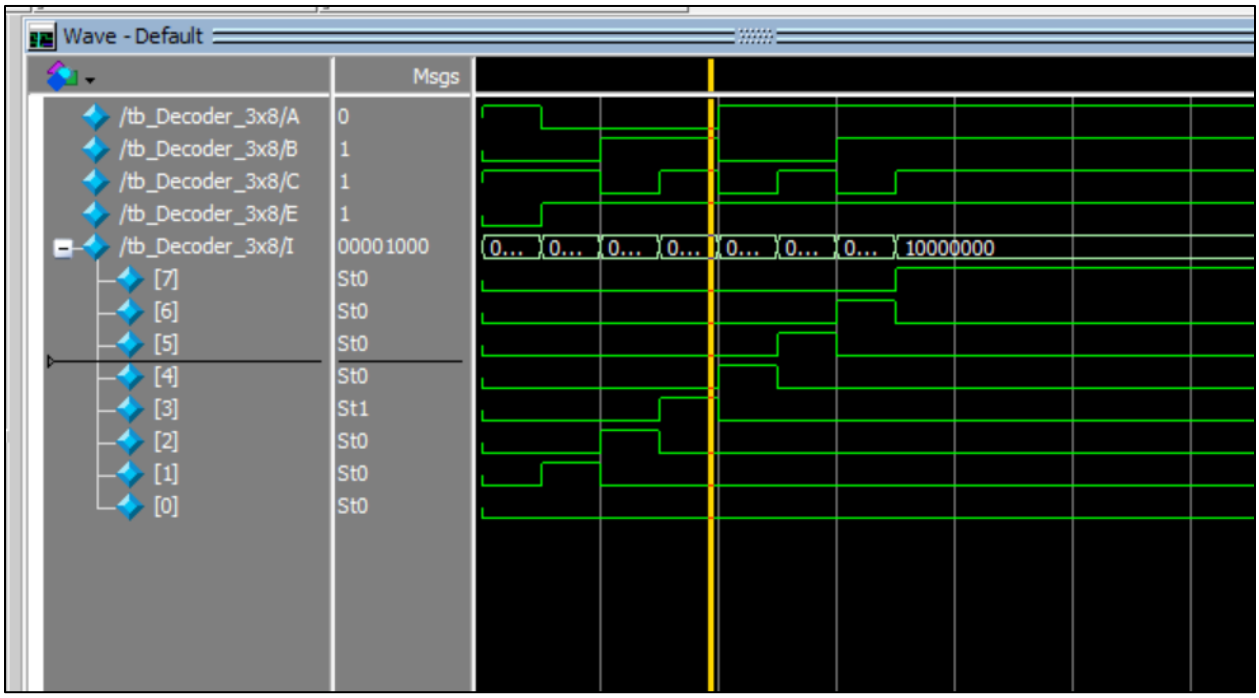
```
1   module tb_Decoder_3x8();
2       reg A;
3       reg B;
4       reg C;
5       reg E;
6       wire [7:0]I;
7
8       Decoder_3x8 my_3x8_Decoder(A,B,C,E,I);
9
10      initial begin
11
12      A = 1;
13      B = 0;
14      C = 1;
15      E = 0;
16      $display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
17      #10
18
19      A = 0;
20      B = 0;
21      C = 1;
22      E = 1;
```


tb_task2.v
task2.v
task3.v
2x4_Decoder.v
tb_3x_Decoder.v
tb_2x4_Decoder.v
3x8_Decoder.v
task3.v
tb_task3.v
task4.v
tb_task4.v
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
#10
A = 0;
B = 1;
C = 0;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
#10
A = 0;
B = 1;
C = 1;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
A = 1;
B = 0;
C = 0;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})


tb_task2.v
task2.v
task3.v
2x4_Decoder.v
tb_3x_Decoder.v
tb_2x4_Decoder.v
3x8_Decoder.v
task3.v
tb_task3.v
task4.v
tb_task4.v
#10
A = 1;
B = 0;
C = 1;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
#10
A = 1;
B = 1;
C = 0;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
#10
A = 1;
B = 1;
C = 1;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})

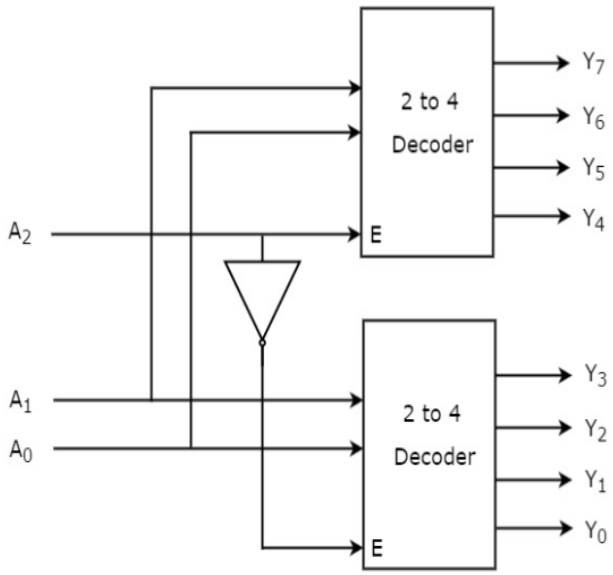

B = 1;
C = 1;
E = 1;
$display("%b %b %b %b %b", A , B, C, E, {I[0], I[1], I[2], I[3], I[4], I[5], I[6], I[7]})
end
endmodule

Output:



Task 3:

Implement 3X8 Decoder using 2X4 Decoder.



Code:

DUT Code:

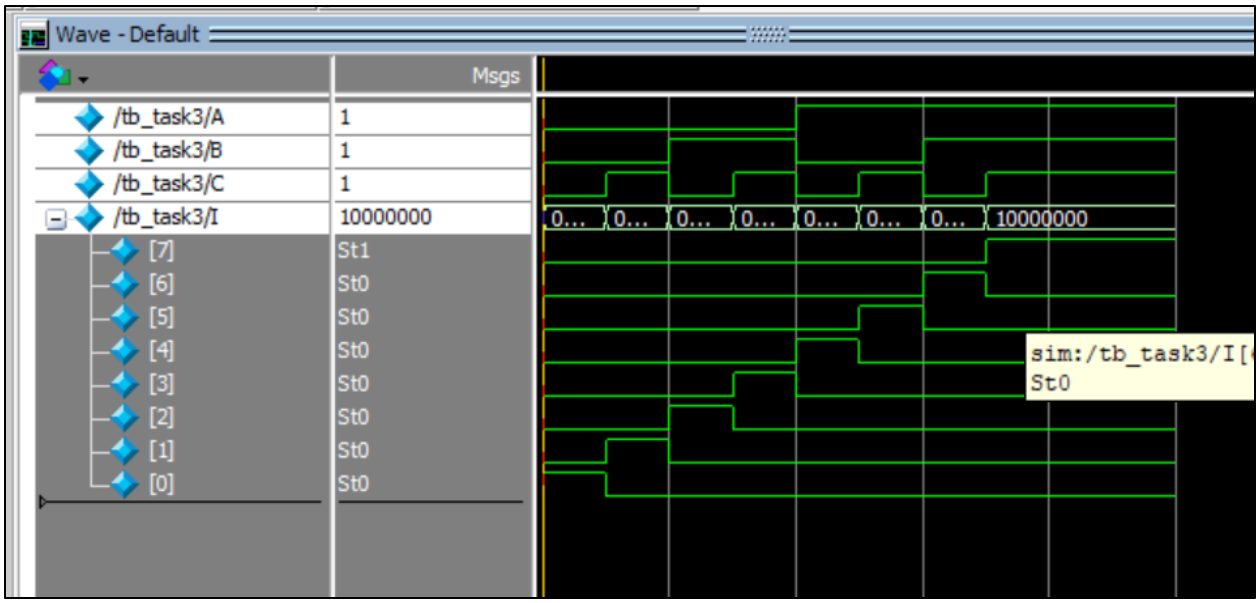
tb_task2.v | task2.v | task3.v | task3.v | 2x4_Decoder.v | tb_3x_Decoder.v | tb_2x4_Decoder.v | 3x8_Decoder.v

```
1   module task3(A, B, C, I);
2
3       input A;
4       input B;
5       input C;
6       output [7:0]I;
7
8       wire nA;
9
10      not n1(nA, A);
11
12      Decoder_2x4 d1(B, C, nA, {I[3], I[2], I[1], I[0]});
13      Decoder_2x4 d2(B, C, A, {I[7], I[6], I[5], I[4]});
14
15  endmodule
```

Test bench Code:

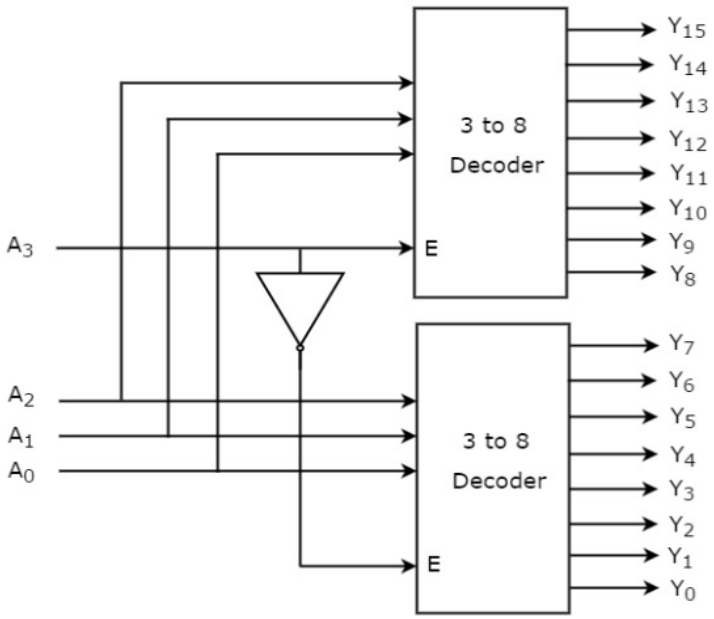
Same as task2

Output:



Task 4:

Implement 4X16 Decoder using 3X8 Decoder.



Code:

DUT Code:

tb_task2.v | task2.v | task4.v | task3.v | task3.v | tb_task3.v | 2x4_Decoder.v | tb_3x_Decoder.v

```
1   module task4(A, I);
2
3       input [3:0]A;
4
5       output [15:0]I;
6
7       wire nA;
8
9       not n1(nA, A[3]);
10
11      Decoder_3x8 d1(A[2], A[1], A[0], nA, I[7:0]);
12      Decoder_3x8 d2(A[2], A[1], A[0], A[3], I[15:8]);
13
14  endmodule
```

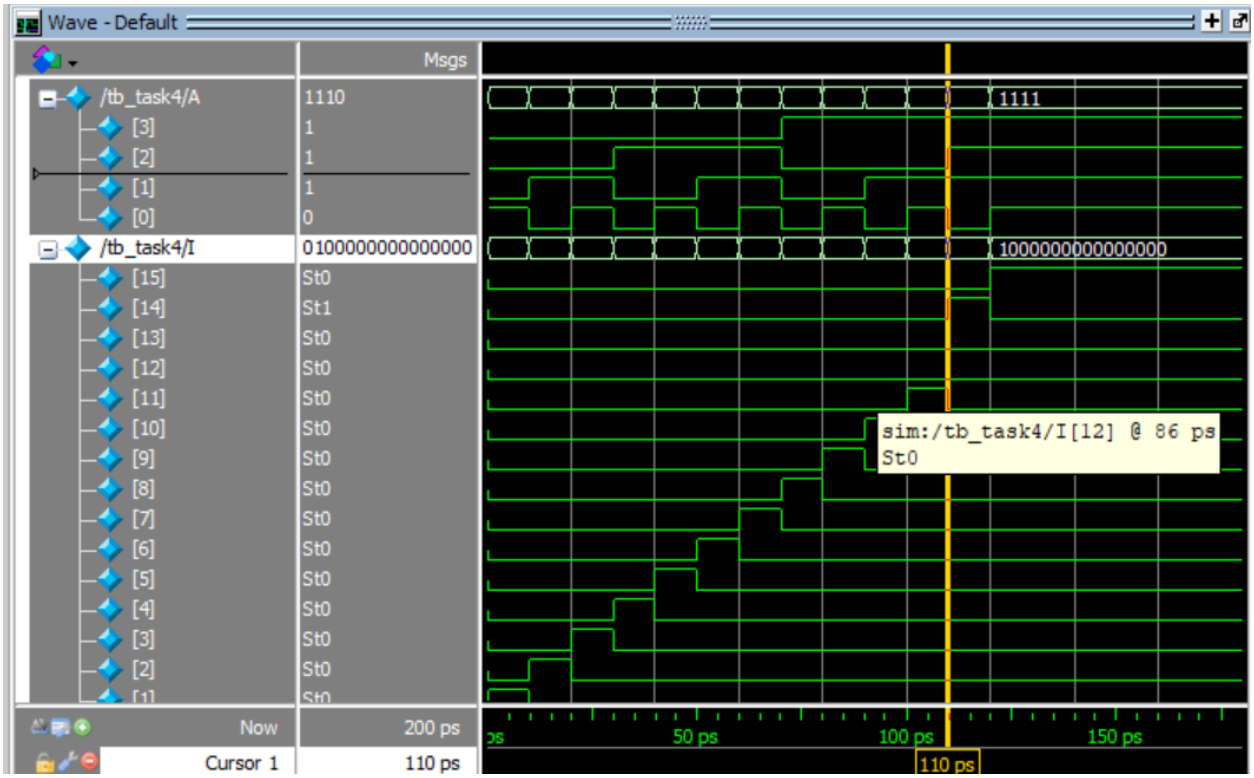
Test bench Code:

tb_task2.v | task2.v | task4.v | tb_task4.v | task3.v | task3.v | tb_

```
1   module tb_task4();
2
3       reg [3:0]A;
4       wire [15:0]I;
5
6       task4 my_3x8_Decoder(A,I);
7
8       initial begin
9       A = 4'b0001;
10      #10
11
12      A = 4'b0010;
13      #10
14
15      A = 4'b0011;
16      #10
17
18      A = 4'b0100;
19      #10
20
21      A = 4'b0101;
22      #10
23
```

```
24      A = 4'b0110;
25      #10
26      A = 4'b0111;
27      #10
28      A = 4'b1000;
29      #10
30
31      A = 4'b1001;
32      #10
33
34      A = 4'b1010;
35      #10
36
37      A = 4'b1011;
38      #10
39
40      A = 4'b1110;
41      #10
42
43      A = 4'b1111;
44      end
45  endmodule
```

Output:



Conclusion:

In this lab, I learned how to implement Decoders using gate level modelling in Verilog.