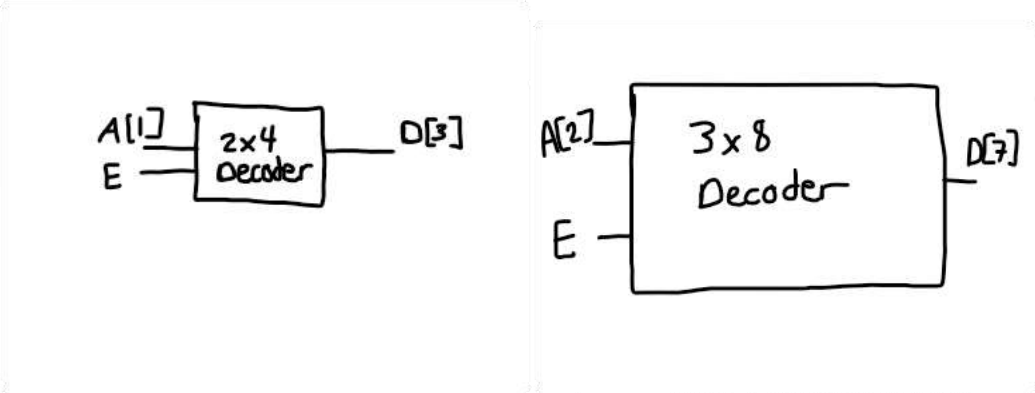
Laboratory Report #3

Name: Josh Ratificar **Date Completed:** 09-23-2023

Laboratory Exercise Title: Structural Modeling of Combinational Circuits

Block Diagrams:



Above includes the block diagrams for this Laboratory Exercise. Based on the diagrams, the following can be concluded:

2x4 Decoder

3-Input Ports:

- **A[0], A[1], E**

4-Output Ports:

- **D[0], D[1], D[2], D[3]**

3x8 Decoder

4-Input Ports:

- **A[0], A[1], A[2], E**

8-Output Ports:

- **D[0], D[1], D[2], D[3], D[4], D[5], D[6], D[7]**

Exercise 3A:

Table 1.0 – *Truth Table for a 2x4 Decoder*

INPUTS			OUTPUTS			
E	A_0	A_1	D_0	D_1	D_2	D_3
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

Boolean Expressions:

$D_3 = A_0'A_1'E$

$D_2 = A_0'A_1E$

$D_1 = A_0A_1'E$

$D_0 = A_0A_1E$

```
/*************************************
*  FILE:              Decoder2x4.v
*  AUTHOR:            Josh Ratificar
*  Class:             Gr.3 CpE 3101L Introduction to HDL
*  Group/Schedule     Friday, 7:30 AM to 10:30 AM
*  Description:       Decoder2x4.v module
*************************************/

module Decoder2x4(                  // A[0] = A; A[1] = B;
    input     [1:0] A,
    input     E,
    output    [3:0] D
);
    wire      [1:0] w;              // Inverted Bus

    not       N1(w[0], A[0]);       // Not A <- w[0]
    not       N2(w[1], A[1]);       // NOT B <- w[1]
    and       A0(D[0], w[0], w[1], E);
    and       A1(D[1], w[1], A[0], E);
    and       A2(D[2], A[1], w[0], E);
    and       A3(D[3], A[0], A[1], E);
endmodule
```

Figure 1.0 – *Decoder2x4.v Script*

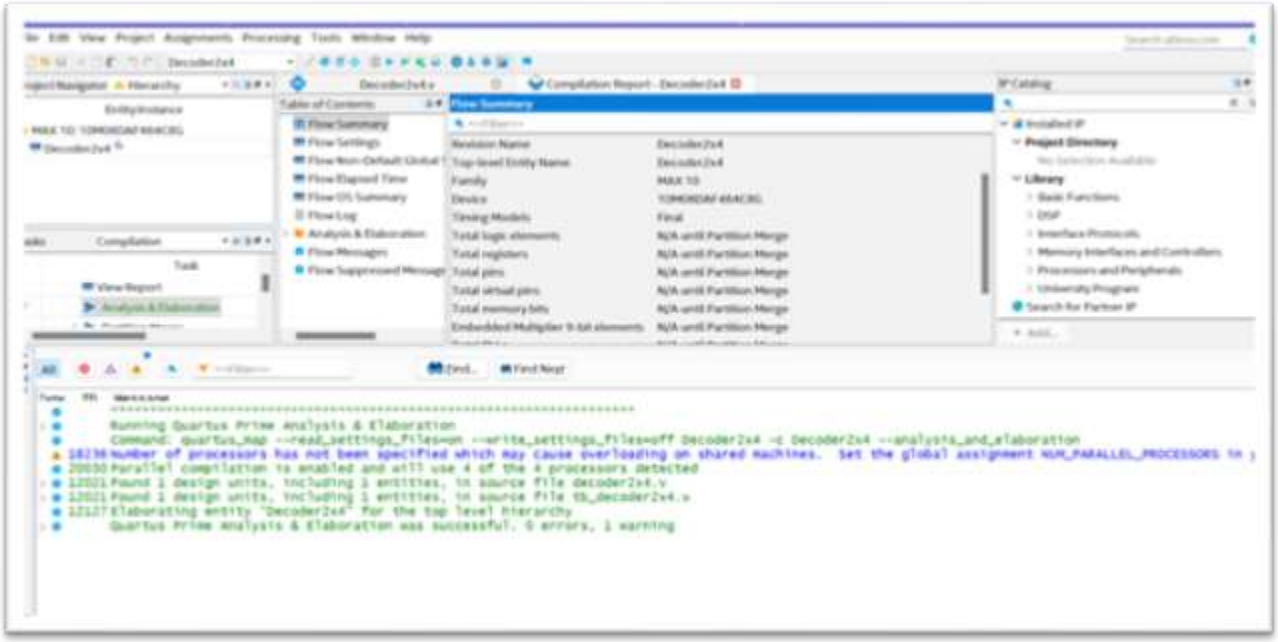


Figure 1.1 – *Decoder2x4.v Analysis and Elaboration Test Results*

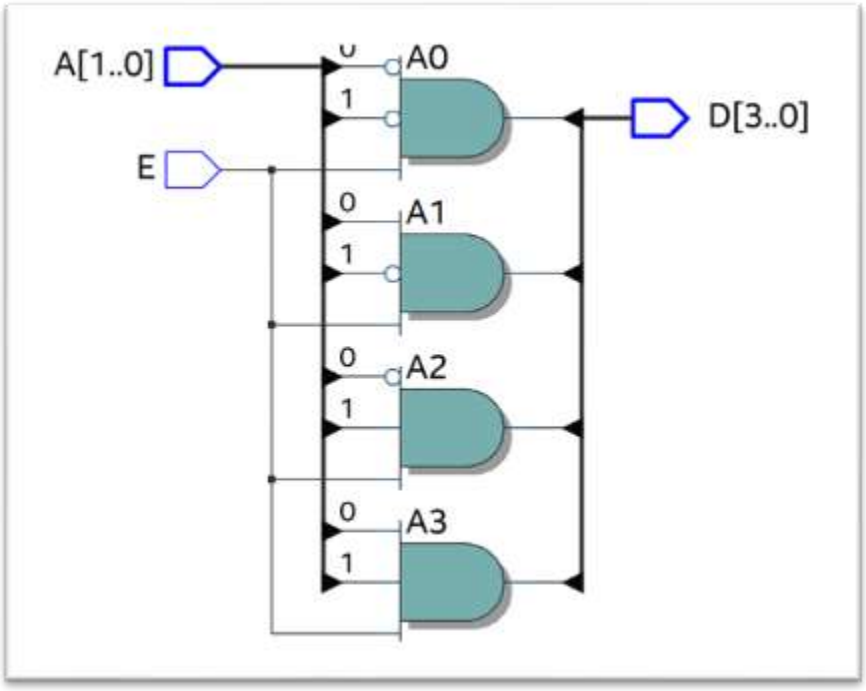


Figure 1.2 – *2x4-Decoder RTL View Output*

```
 2   *  FILE:              tb_Decoder2x4.v
 3   *  AUTHOR:            Josh Ratificar
 4   *  Class:             Gr.3 CpE 3101L Introduction to HDL
 5   *  Group/Schedule     Friday, 7:30 AM to 10:30 AM
 6   *  Description:       Decoder2x4 Test Bench
 7   ************************************/
 8   `timescale 1 ns / 1 ps
 9   module tb_Decoder2x4();
10      reg  [1:0] A;
11      reg   E;
12      wire  [3:0] D;
13
14      Decoder2x4 UUT(A[1:0], E, D[3:0]);
15      initial begin
16         $display("2to4 Decoder Simulation");
17         E = 1'b0;    A = 2'b00;          #10
18         E = 1'b0;    A = 2'b01;          #10
19         E = 1'b0;    A = 2'b10;          #10
20         E = 1'b0;    A = 2'b11;          #10
21
22         E = 1'b1;    A = 2'b00;          #10
23         E = 1'b1;    A = 2'b01;          #10
24         E = 1'b1;    A = 2'b10;          #10
25         E = 1'b1;    A = 2'b11;          #10
26         $stop;
27      end
28      initial begin
29         $monitor("Time = %d ns \t Enable = %1b \t A = %2b \t D = %b", $time, E, A, D);
30      end
31   endmodule
```

Figure 1.3 – tb*_Decoder2x4.v Script for Testing Module*

Figure 1.4 – *2x4-Decoder RTL Simulation Output*

```
# 2to4 Decoder Simulation
# Time =                  0 ns   Enable = 0   A = 00   D = 0000
# Time =                 10 ns   Enable = 0   A = 01   D = 0000
# Time =                 20 ns   Enable = 0   A = 10   D = 0000
# Time =                 30 ns   Enable = 0   A = 11   D = 0000
# Time =                 40 ns   Enable = 1   A = 00   D = 0001
# Time =                 50 ns   Enable = 1   A = 01   D = 0010
# Time =                 60 ns   Enable = 1   A = 10   D = 0100
# Time =                 70 ns   Enable = 1   A = 11   D = 1000
```

Figure 1.5 – *2x4-Decoder Test Bench Monitor Output (Annotations to **Figure 1.4**)*

Discussion of Results (Exercise 3A)

From what we can observe, the circuit is operating how it is expected. This is apparent by observing **Figure 1.5's** results in comparison to **Table 1.0's** truth table. The "*$monitor*" command reflects the observable values in **Figure 1.4** and thus confirms that the test bench is appropriately designed. The output of this circuitry is dependent on the enabler. When the enabler is off, the output is 0.

Exercise 3B:

Table 2.0 – *Truth Table for a 3x8 Decoder*

INPUTS				OUTPUTS							
E	A_0	A_1	A_2	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

```
1  /************************************
2  *  FILE:              Decoder3x8.v
3  *  AUTHOR:            Josh Ratificar
4  *  Class:             Gr.3 CpE 3101L Introduction to HDL
5  *  Group/Schedule     Friday, 7:30 AM to 10:30 AM
6  *  Description:       Decoder3x8.v module
7  ************************************/
8  module Decoder3x8(
9      input       [2:0] A,
10     input       E,
11     output      [7:0] D
12 );
13     wire [3:0] D1, D2;
14     Decoder2x4 Dc2x4_1(
15         .A(A[1:0]),
16         .E(E & A[2]),
17         .D(D[7:4])
18     );
19
20     Decoder2x4 Dc2x4_2(
21         .A(A[1:0]),
22         .E(E & ~A[2]),
23         .D(D[3:0])
24     );
25 endmodule
```

Figure 2.0 – *Decoder3x8.v Script*

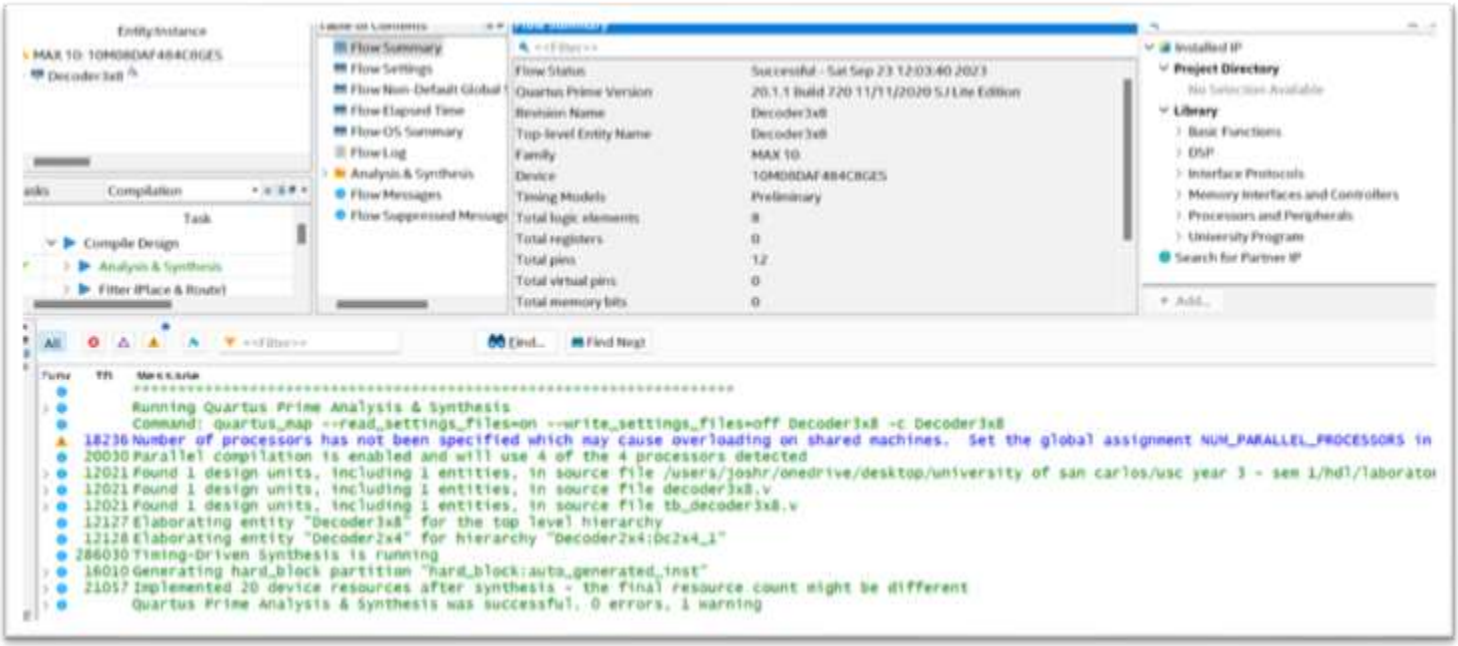


Figure 2.1 – *Decoder3x8.v Analysis and Elaboration Test Results*

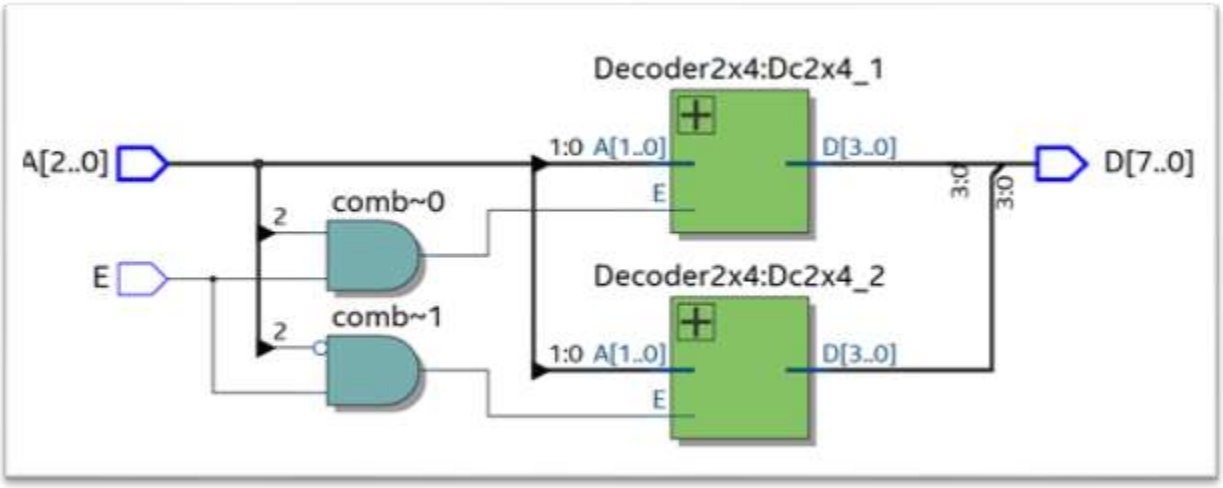


Figure 2.2 – *3x8-Decoder RTL View Output*

```
/************************************
*  FILE:              tb_Decoder3x8.v
*  AUTHOR:            Josh Ratificar
*  Class:             Gr.3 CpE 3101L Introduction to HDL
*  Group/Schedule     Friday, 7:30 AM to 10:30 AM
*  Description:       Test bench for Decoder3x8.v
************************************/

`timescale 1 ns / 1 ps
module tb_Decoder3x8();
   reg   [2:0] A;
   reg   E;
   wire  [7:0] D;
   Decoder3x8 UUT(A[2:0], E, D[7:0]);

   initial begin
      $display("Decoder 3x8 Simulation:");
      $display("Enable STATUS [OFF].");
      E = 1'b0;   A = 3'b000; #10
      E = 1'b0;   A = 3'b001; #10
      E = 1'b0;   A = 3'b010; #10
      E = 1'b0;   A = 3'b011; #10
      E = 1'b0;   A = 3'b100; #10
      E = 1'b0;   A = 3'b101; #10
      E = 1'b0;   A = 3'b110; #10
      E = 1'b0;   A = 3'b111; #10

      $display("Enable STATUS [ON].");

      E = 1'b1;   A = 3'b000; #10
      E = 1'b1;   A = 3'b001; #10
      E = 1'b1;   A = 3'b010; #10
      E = 1'b1;   A = 3'b011; #10
      E = 1'b1;   A = 3'b100; #10
      E = 1'b1;   A = 3'b101; #10
      E = 1'b1;   A = 3'b110; #10
      E = 1'b1;   A = 3'b111; #10
      $stop;
   end
   initial begin
      $monitor("TIME = %d ns \t E = %1b \t A = %3b \t D = %8b", $time, E, A, D);
   end
endmodule
```

Figure 2.3 – tb*_Decoder3x8.v Script for Testing Module*

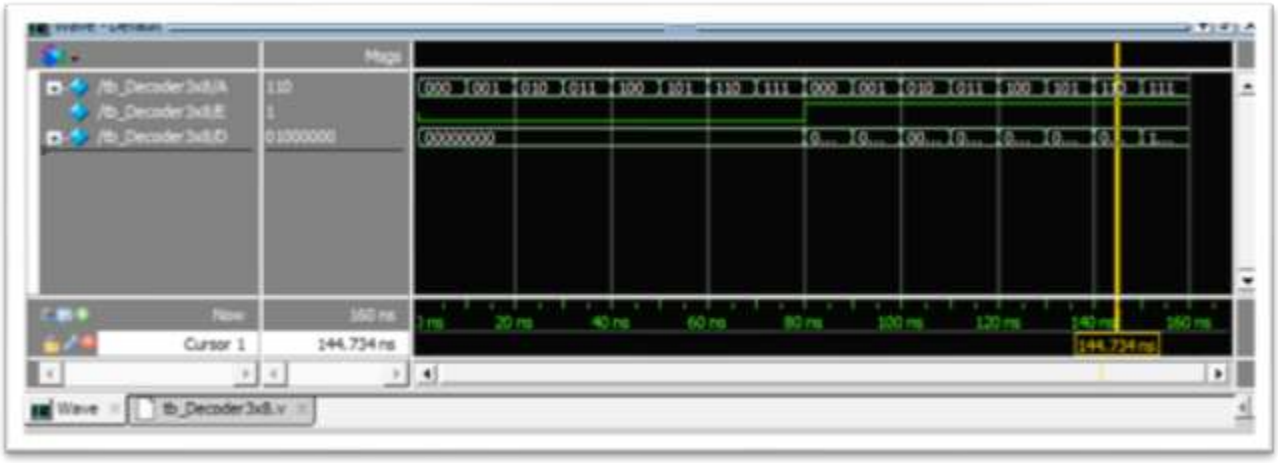


Figure 2.4 – *3x8-Decoder RTL Simulation Output*

```
# Decoder 3x8 Simulation:
# Enable STATUS [OFF].
# TIME =                  0 ns   E = 0   A = 000   D = 00000000
# TIME =                 10 ns   E = 0   A = 001   D = 00000000
# TIME =                 20 ns   E = 0   A = 010   D = 00000000
# TIME =                 30 ns   E = 0   A = 011   D = 00000000
# TIME =                 40 ns   E = 0   A = 100   D = 00000000
# TIME =                 50 ns   E = 0   A = 101   D = 00000000
# TIME =                 60 ns   E = 0   A = 110   D = 00000000
# TIME =                 70 ns   E = 0   A = 111   D = 00000000
# Enable STATUS [ON].
# TIME =                 80 ns   E = 1   A = 000   D = 00000001
# TIME =                 90 ns   E = 1   A = 001   D = 00000010
# TIME =                100 ns   E = 1   A = 010   D = 00000100
# TIME =                110 ns   E = 1   A = 011   D = 00001000
# TIME =                120 ns   E = 1   A = 100   D = 00010000
# TIME =                130 ns   E = 1   A = 101   D = 00100000
# TIME =                140 ns   E = 1   A = 110   D = 01000000
# TIME =                150 ns   E = 1   A = 111   D = 10000000
```

Figure 2.5 – *3x8-Decoder Test Bench Monitor Output (Annotations to **Figure 2.4**)*

Discussion of Results (Exercise 3B)

By modularizing, a 3x8 was created with the use of two 2x4-decoders. By creating a test bench (**Figure 2.3**), it was straight-forward verifying the results of the RTL simulation (**Figure 2.4**). As shown by the "*$monitor*" command, the results of **Figure 2.5** reflect the behaviour observed in the truth table (**Figure 1.0**). As observed, the output is dependent on the enabler being on, which is the desired outcome of the enabler.