Student Name: Ricardo A Garza
Student UNT email ID: ricardogarza3@my.unt.edu 10967208

Requirements:

1. Using **Structural modelling**, design a 4-to-16 Decoder using 2-to-4 Decoder

   _____

2. Run the simulation using testbench for **all possible input combinations (Including enable)**.

   _____

3. Capture the waveforms and verify against truth table.

   _____

4. Configure Nexys 4 DDR FPGA board with switches SW0(a[0]), SW1(a[1]), SW2(a[2]), SW3(a[3]) as inputs and SW4(en) as enable pin. Use LE D's LED0(b[0]), LED1(b[1]), LED2(b[2]), LED3(b[3]), LED4(b[4]), LED5(b[5]), LED6(b[6]), LED7(b[7]), LED8(b[8]), LED9(b[9]), LED10(b[10]), LED11(b[11]), LED12(b[12]), LED13(b[13]), LED14(b[14]), LED15(b[15]) for outputs.

   _____

Learning Objectives:
The general learning objectives of this lab was to design a 4:16 decoder using 2:4 decoders and implement it on the Nexys 4 FPGA board using Xilinx.

General Steps:
1. Design a 2:4 decoder using an enable switch and using behavioral modelling.
2. Using structural modelling design a 4:16 decoder using 5 2:4 decoders.
3. Find the truth table for a 4:16 decoder
4. Write a testbench to verify the design of 4:16 decoder
5. Simulate and capture waveforms to verify against the truth table
6. Synthesize the design by using the synthesis tutorial and run on Nexys 4 FPGA board.

Detailed Steps:
Some detailed steps to complete this lab were
1. Following the example given on canvas write out the 2:4 decode file.
   a. Modify the ports little bit so that it can include an enable switch

$$en: in\ STD\_LOGIC;$$

Student Name: Ricardo A Garza
Student UNT email ID: ricardogarza3@my.unt.edu 10967208

2. Also modified the statements in the architecture

```
39
40  architecture Behavioral of decode2to4 is
41  signal enx: STD_LOGIC_VECTOR (2 downto 0);
42
43  begin
44   enx <= en&x;
45   WITH enx SELECT
46     y <= "0001" WHEN "100",
47          "0010" WHEN "101",
48          "0100" WHEN "110",
49          "1000" WHEN "111",
50          "0000" WHEN OTHERS;
51
```

3. After finishing the 2:4 decoder you make a 4:16 decoder about the same.
    a. The component will be part of the 2:4 decoder

```
39
40  architecture Structure of decode4to16 is
41    Component decode2to4
42     Port (  x : in STD_LOGIC_VECTOR (1 downto 0);
43             en : in STD_LOGIC;
44             y : out STD_LOGIC_VECTOR (3 downto 0));
45    END Component;
```

    i.
    b. The entity will be about the same except it will go from 3 down to 0 in the out port to 15 down to zero.

```
entity decode4to16 is
    Port (  x : in STD_LOGIC_VECTOR (3 downto 0);
            en : in STD_LOGIC;
            y : out STD_LOGIC_VECTOR (15 downto 0));
```

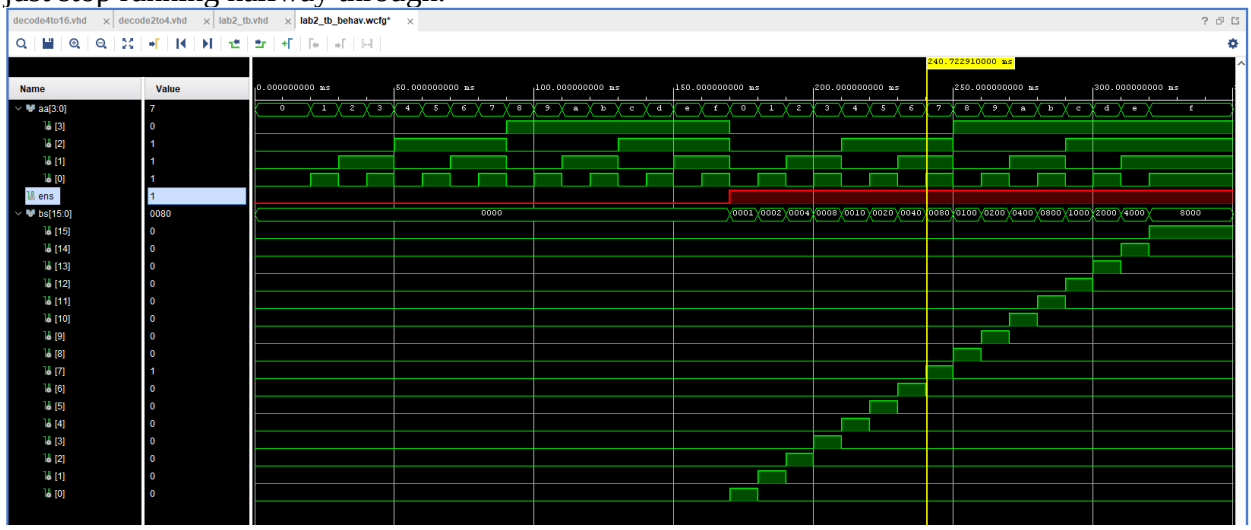
    i.
    c. After completing these two parts, you make 5 instances of the 2:4 decoders while keep track of what each one does.

```
decoder0: decode2to4 port map(x(3 downto 2), en, k(3 downto 0));
decoder1: decode2to4 port map(x(1 downto 0), k(0), y(3 downto 0));
decoder2: decode2to4 port map(x(1 downto 0), k(1), y(7 downto 4));
decoder3: decode2to4 port map(x(1 downto 0), k(2), y(11 downto 8));
decoder4: decode2to4 port map(x(1 downto 0), k(3), y(15 downto 12));
```

    i.
4. Next you write the testbench
    a. It will mostly stay the same as the example given except that you will make it run longer.
    b. You also must make sure that you call the 4:16 decoder instead of the 2:4 decoder in the component.
5. After you run the simulation
6. Attach the given constraint given on canvas
    a. Make sure to uncomment the correct lines needed
7. Run the synthesis, implementation and generate bitstream
    a. Make sure to fix any errors generate while running these parts
8. Run on Nexys 4 FPGA board.

Student Name: Ricardo A Garza
Student UNT email ID: ricardogarza3@my.unt.edu 10967208

Observations:
One thing I noticed was that you must make sure you have all your test signals in correctly or you will only get part of your waveform to display. Also, the 2:4 decoder can greatly affect how the waveform comes out. So, you must make sure that the logic is correct. I had to try out different kinds of implantation just to get the waveform to come out correctly because it would just stop running halfway through.



Truth table – will have enable ON

A	B	C	D	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	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	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Student Name: Ricardo A Garza
Student UNT email ID: ricardogarza3@my.unt.edu 10967208

Summary:
In this lab we learned how to implement a 4:16 decoder using 2:4 decoders. We also learned how implement our design on an FPGA board and run it on the board. It was very nice to see the lights turn on as we were switching through the switches. Still this taught me that doing structural modelling is different from doing behavioral modelling. They do not necessarily follow the same format as each other. Overall, the lab was more difficult than the first. It was more knowing how to implement your logic using a different architecture.