Lab 3: Decoder/Encoder Circuits Lab

Logic Lab

By: Erich Wanzek, Nathan Phipps

Abstract

In this text we analyze our Verilog code, including those schematics (block diagrams, wave diagram simulations), by which we developed, as well as the implementation of these schematics/code in the ability to create hex displays, decoders and encoders (both active high and active low). **Through performing our tasks, found within the guidelines of this lab, we felt it was of high importance to note that all of our wave diagrams include an initial period offset of 10 nanoseconds in order to simulate the case where the enable inputs were set to a logic one.** Our process in developing these schematics (block diagrams, wave diagram simulations), and Verilog codes pertains to the guidelines set forth in our Lab 3 rubric for Logic Lab, Summer 2019. From our trials in this lab we are able to present a full report pertaining to the aforementioned concepts (active low and active high encoders, as well as decoders, and 7 segment displays) which will be discussed in the body of this text, as follows:

1.) **Analysis:** Analysis of Each (Parts 1 - 6):
   - I.) Analysis of Part 1: Seven Segment Display Decoders,
   - II.) Analysis of Part 2: Seven Segment Display Implementation,
   - III.) Analysis of Part 3: Active Low Decoder - NAND Gate Schematic Implementation,
   - IV.) Analysis of Part 4: BCD to Decimal Decoder Verilog,
   - V.) Analysis of Part 5: Decimal to Binary Decoder - Verilog,
   - VI.) Analysis of Part 6: Putting It All Together,

2.) **Group Member Contributions:** A brief discussion on the contributions of each group member,

3.) **Components for Reference:** Compiled Lab Components for reference, including block diagrams, wave diagram simulations, Verilog codes, “FPGA board” trial photographs.

Analysis

Analysis Part 1

In part 1 of Lab 3, we examined a seven segment display, including the Verilog code necessary to output hexadecimal to an “FPGA” board. To elaborate, we used Karnaugh maps to create individual expressions utilized to light up segments on the hex display. These segments were then grouped into sets to form Hex expressions, as outputs, including numbers 1 - 9, and letters A - F.

Analysis Part 2

From part 2, we created a top level in order to implement our verilog code from part 1 (seven segment display), and upload it to our “FPGA” board.

Analysis Part 3

In part 2, we designed an active low 2 to 4 decoder and an active low 3 to 8 decoder. We created the decoders using both block symbol diagrams and Verilog code modules. The 3 to 8 decoder was created by using two instantiations of the 2 to 4 decoder. We verified that the active low 2 to 4 decoder and an active low 3 to 8 decoder functioned properly by running functional timing simulations and analyzing the timing diagrams. We then created a top level for the 3 to 8 decoder to run on the FPGA board. The active low 3 to 8 decoder ran as expected.

Analysis Part 4

Next, in part 4, we designed a binary to decimal decoder. We created a BCD to Decimal Decoder in Verilog. The aforementioned code was generated using “POS (products of sum)” form allowing us to examine and translate the truth table outlined within the lab, resulting in assignment expressions for our Verilog code.
We verified that the active low BCD to Decimal Decoder functioned properly by running a functional timing simulation and analyzing the timing diagrams.

Analysis Part 5

Within part 5, we produced code for a “decimal to binary encoder.” The aforementioned code was generated using “SOP (sum of products)” form allowing us to examine and translate the truth table outlined within the lab, resulting in assignment expressions for our Verilog code. We verified that the decimal to binary encoder functioned properly by running a functional timing simulation and analyzing the timing diagrams.

Analysis Part 6

Finally, in part 6, we designed a top level. We created block symbols for the BCD to decimal decoder, the decimal to binary encoder, and the seven segment display module. We then connected them using data bus wires in a block diagram. We uploaded the top level to the FPGA board and got the results we expected. Pictures of the results are shown in part 6.

Group Member Contributions

Both group members, Nathan Phipps, and Erich Wanzek performed equal contributions on this lab, which involved tasks such as coding in Verilog, working with k-maps, creating block diagrams, block symbol files, top levels, wave diagram simulations, typing up this report and so forth.

Components for Reference

Part 1: Seven Segment Display Decoders

Part 1: a

Symbol	Segment Sets	Symbol	Segment Sets
0	(0,1,2,3,4,5)	A	(0,1,2,4,5,6)
1	(1,2)	b	(6,5,4,3,2)
2	(0,1,3,4,6,)	C	(0,3,4,5)
3	(0,1,2,3,6)	d	(0,1,2,3,4,6)
4	(1,2,5,6)	E	(0,3,4,5,6)
5	(0,5,6,2,3)	F	(0,4,5,6)
6	(0,2,3,4,5,6)		0
7	(0,1,2)		
8	(0,1,2,3,4,5,6)		
9	(0,1,2,5,6)		

Part 1: b

Symbol	None	S6	S5	S4	S3	S2	S1	S0	Hex Encoding of S6-S0
0	0	1	0	0	0	0	0	0	0x40
1	0	1	1	1	1	0	0	1	0x79
2	0	0	1	0	0	1	0	0	0x24
3	0	0	1	1	0	0	0	0	0x30
4	0	0	0	1	1	0	0	1	0x19
5	0	0	0	1	0	0	1	0	0x12
6	0	0	0	0	0	0	1	0	0x02
7	0	1	1	1	1	0	0	0	0x78
8	0	0	0	0	0	0	0	0	0x00
9	0	0	0	1	1	0	0	0	0x18
A	0	0	0	0	1	0	0	0	0x08
B	0	0	0	0	0	0	1	1	0x03
C	0	1	0	0	0	1	1	0	0x46
D	0	0	1	0	0	0	0	1	0x21
E	0	0	0	0	0	1	1	0	0x06
F	0	0	0	0	1	1	1	0	E

Part 1: c

	INPUTS				Output						
	A	B	C	D	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	0	1	0	0	0	0	0	0
1	0	0	0	1	1	1	1	1	0	0	1
2	0	0	1	0	0	1	0	0	1	0	0
3	0	0	1	1	0	1	1	0	0	0	0
4	0	1	0	0	0	0	1	1	0	0	1
5	0	1	0	1	0	0	1	0	0	1	0
6	0	1	1	0	0	0	0	0	0	1	0

	INPUTS				Output						
	A	B	C	D	S6	S5	S4	S3	S2	S1	S0
7	0	1	1	1	1	1	1	1	0	0	0
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	1	1	0	0	0
A	1	0	1	0	0	0	0	1	0	0	0
B	1	0	1	1	0	0	0	0	0	1	1
C	1	1	0	0	1	0	1	0	1	1	0
D	1	1	0	1	0	1	0	0	0	0	1
E	1	1	1	0	0	0	0	0	1	1	0
F	1	1	1	1	0	0	0	1	1	1	0

Part 1: d (K - Maps)

S6 (A, B, C, D) = Σ(0,1,7,12)

S6		CD			
		00	01	11	10
AB	00	1	1		
	01			1	
	11	1			
	10				

S6 = A' B' C' + A' B C D + A B C' D'

S5 (A, B, C, D) = Σ(1,2,3,7, 13)

S5		CD			
		00	01	11	10
AB	00		1	1	1
	01			1	
	11		1		
	10				

S5 = A' B' D + A' CD + A' B' C + A B C' D

S4 (A, B, C, D) = Σ(1, 3, 4, 5, 7, 9)

S4		CD			
		00	01	11	10
AB	00		1	1	
	01	1	1	1	
	11				
	10		1		

S4 = ?

$S4 = A'BC' + A'D + B'C'D$

S3 (A, B, C, D) = Σ(1, 4, 7, 9, 10, 15)

S3		CD			
		00	01	11	10
AB	00		1		
	01	1		1	
	11			1	
	10		1		1

S3 = ?

$S3 = A'BC'D' + B'C'D + BCD + AB'CD'$

S2 (A, B, C, D) = Σ(2, 12, 14, 15)

S2		CD			
		00	01	11	10
AB	00				1
	01				
	11	1		1	1
	10				

S2 = ?

$S2 = A'B'CD' + ABD' + ABC$

S1 (A, B, C, D) = Σ(5, 6, 11, 12, 14, 15)

B1		CD			
		00	01	11	10
AB	00				
	01		1		1
	11	1		1	1
	10			1	

S1 = ?

$S1 = ABD' + A'BC'D + BCD' + ACD$

S0 (A, B, C, D) = Σ(1, 4, 11, 13)

S0		CD			
		00	01	11	10
AB	00		1		
	01	1			
	11		1		
	10			1	

S0 = ?

$S0 = A'BC'D' + A'B'C'D + ABC'D + AB'CD$

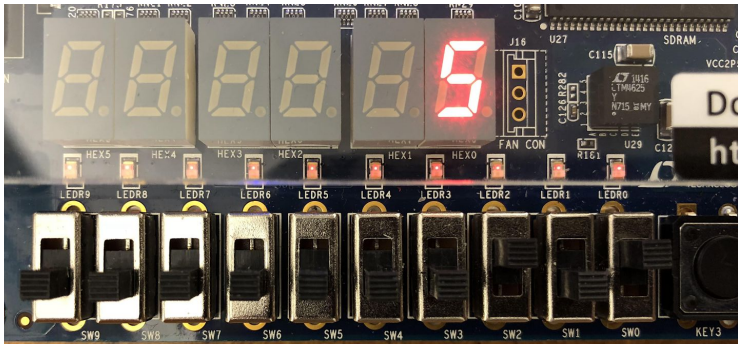
Part 1: e (Verilog Code: Active Low - Hex Display)

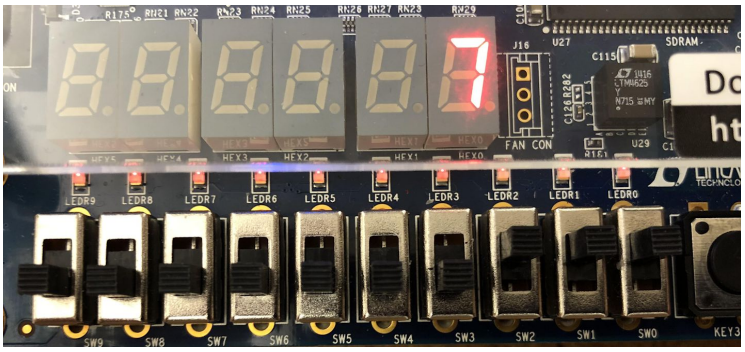
```
module hex_7seg_bitwise (X, segment); //module name 7 segment display, part 1e
        input wire [3:0]X; //defines a wire that's 4 bits wide
        output wire [6:0]segment; //defines an output wire that is 7 bits wide

        // Base for copying and pasting to make typing segments easier: A[3]&B[2]&C[1]&D[0] |

        // S0 = Sum(1,4,11,13)
        // Segment[0] = ~A[3]&~B[2]&~C[1]&D[0] | ~A[3]&B[2]&~C[1]&~D[0] | A[3]&B[2]&~C[1]&D[0] | A[3]&~B[2]&C[1]&D[0];
        assign segment[0] = ~X[3]&~X[2]&~X[1]&X[0] | ~X[3]&X[2]&~X[1]&~X[0] | X[3]&X[2]&~X[1]&X[0] | X[3]&~X[2]&X[1]&X[0];

        // S1 = Sum(5,6,11,12,14,15)
        // Segment[1] = A[3]&B[2]&~D[0] | ~A[3]&B[2]&~C[1]&D[0] | B[2]&C[1]&~D[0] | A[3]&C[1]&D[0];
        assign segment[1] = X[3]&X[2]&~X[0] | ~X[3]&X[2]&~X[1]&X[0] | X[2]&X[1]&~X[0] | X[3]&X[1]&X[0];

        // S2 = Sum(2,12,14,15)
        // Segment[2] = ~A[3]&~B[2]&C[1]&~D[0] | A[3]&B[2]&~D[0] | A[3]&B[2]&C[1];
        assign segment[2] = ~X[3]&~X[2]&X[1]&~X[0] | X[3]&X[2]&~X[0] | X[3]&X[2]&X[1];

        // S3 = Sum(1,4,7,9,10,15)
        // Segment[3] = ~A[3]&B[2]&~C[1]&~D[0] | ~B[2]&~C[1]&D[0] | B[2]&C[1]&D[0] | A[3]&~B[2]&C[1]&~D[0]
        assign segment[3] = ~X[3]&X[2]&~X[1]&~X[0] | ~X[2]&~X[1]&X[0] | X[2]&X[1]&X[0] | X[3]&~X[2]&X[1]&~X[0];

        // S4 = Sum(1,3,4,5,7,9)
        // Segment[4] = ~A[3]&B[2]&~C[1] | ~A[3]&D[0] | ~B[2]&~C[1]&D[0]
        assign segment[4] = ~X[3]&X[2]&~X[1] | ~X[3]&X[0] | ~X[2]&~X[1]&X[0];

        // S5 = Sum(1,2,3,7,13)
        // Segment[5] = ~A[3]&~B[2]&D[0] | ~A[3]&C[1]&D[0] | ~A[3]&~B[2]&C[1] | A[3]&B[2]&~C[1]&D[0]
        assign segment[5] = ~X[3]&~X[2]&X[0] | ~X[3]&X[1]&X[0] | ~X[3]&~X[2]&X[1] | X[3]&X[2]&~X[1]&X[0];

        // S6 = Sum(0,1,7,12)
        // Segment[6] = ~A[3]&~B[2]&~C[1] |  ~A[3]&B[2]&C[1]&D[0] |  A[3]&B[2]&~C[1]&~D[0]
        assign segment[6] = ~X[3]&~X[2]&~X[1] |  ~X[3]&X[2]&X[1]&X[0] |  X[3]&X[2]&~X[1]&~X[0];

endmodule //ends module
```

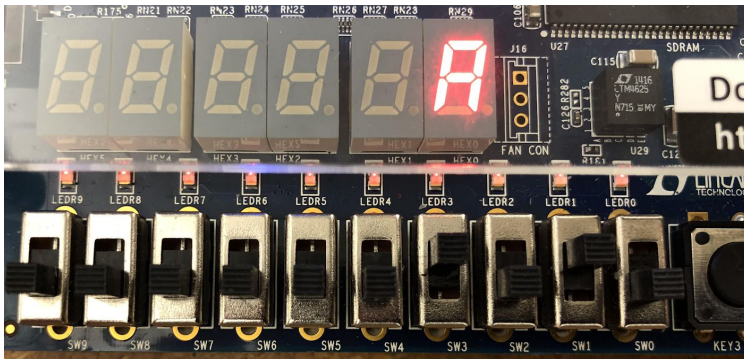
Part 2: Seven Segment Display Implementation (Verilog Code, Top Level)

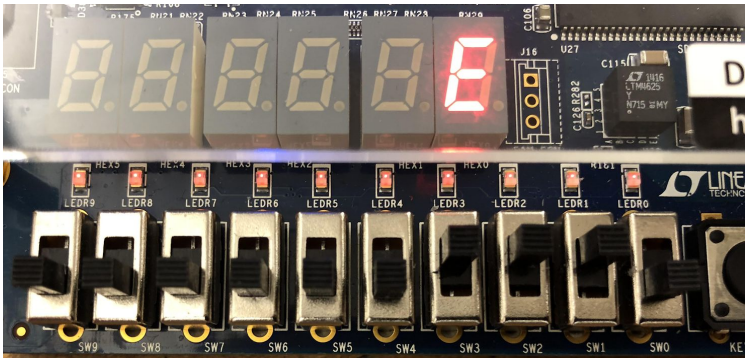
```
module part01_top (input[3:0]SW, output[6:0]HEX0); //defines module, with 4 input switches, and outputs 7 bits to a segment display.

        hex_7seg_bitwise inst0(SW, HEX0); //instantiates the 7 segment bitwise verilog code

endmodule //ends module
```

Pictures of FPGA board for test cases of 5,7,A,E:



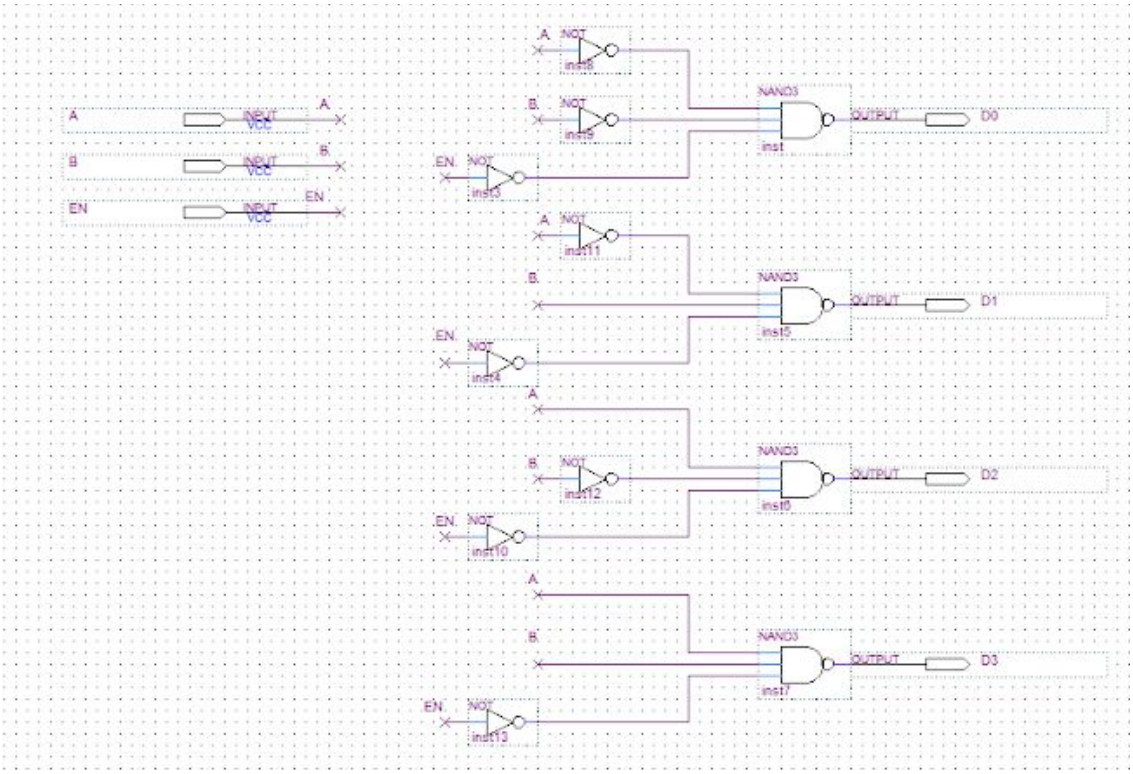






Part 3: Active Low Decoder - NAND Gate Schematic Implementation

Active Low 2 to 4 Decoder Block diagram:

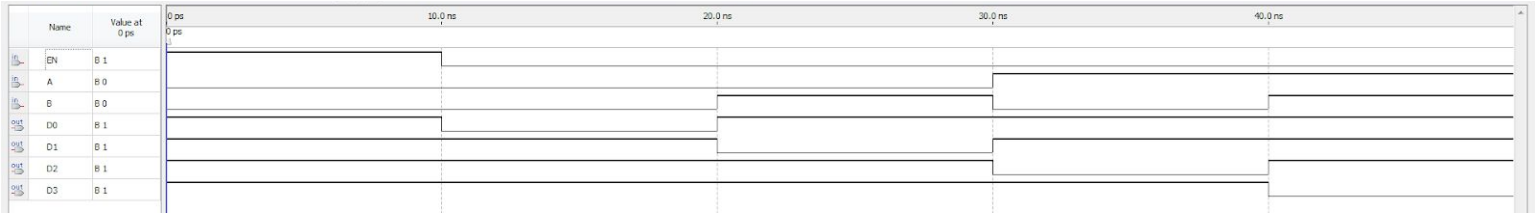


Verilog Code for 2 to 4 Decoder (*note: we made both a verilog module and block diagram for the 2 to 4 decoder because we were unsure of if you wanted a BDF file, or .v file for this part)

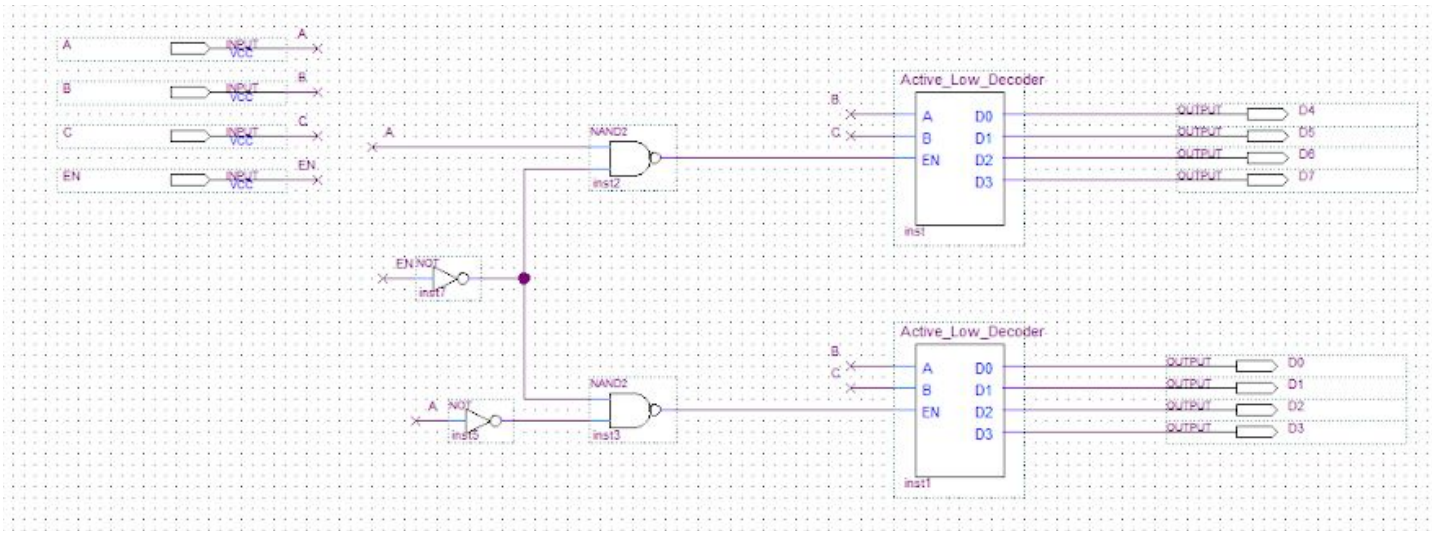
```
module  Active_Low_2_to_4_converter(A,B,EN,D0,D1,D2,D3); //defines module

        input wire A; //defines an input A
        input wire B; //defines an input B
        input wire EN; //defines an input Enable, "EN"

        output wire D0; //defines an output wire D0
        output wire D1; //defines an output wire D1
        output wire D2; //defines an output wire D2
        output wire D3; //defines an output wire D3

        assign D0 = ~(~A & ~B & ~EN); //assigns logical expression to D0
        assign D1 = ~(~A & B & ~EN); //assigns logical expression to D1
        assign D2 = ~(A & ~B & ~EN); //assigns logical expression to D2
        assign D3 = ~(A & B & ~EN); //assigns logical expression to D3

endmodule //ends module
```

Timing Diagram for Active Low 2 to 4 Decoder:

(*Note: In this wave diagram simulation an initial period offset of 10 nanoseconds in order to simulate the case where the enable inputs were set to a logic one.)



Active Low 3 to 8 Decoder Block diagram:

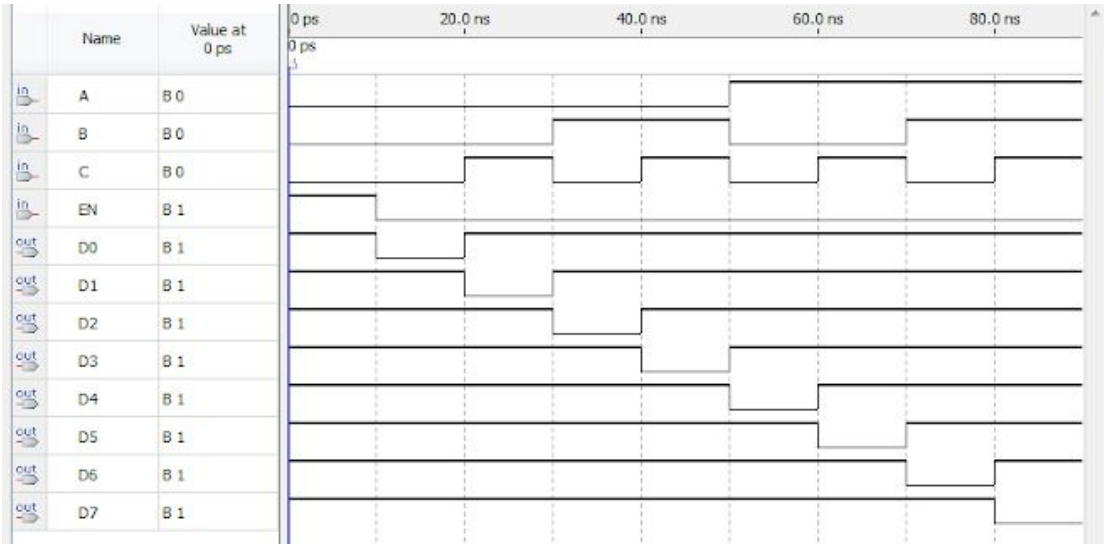


Verilog Code for 3 to 8 Decoder (*note: we made both a verilog module and block diagram for the 3 to 8 decoder because we were unsure of if you wanted a BDF file, or .v file for this part)

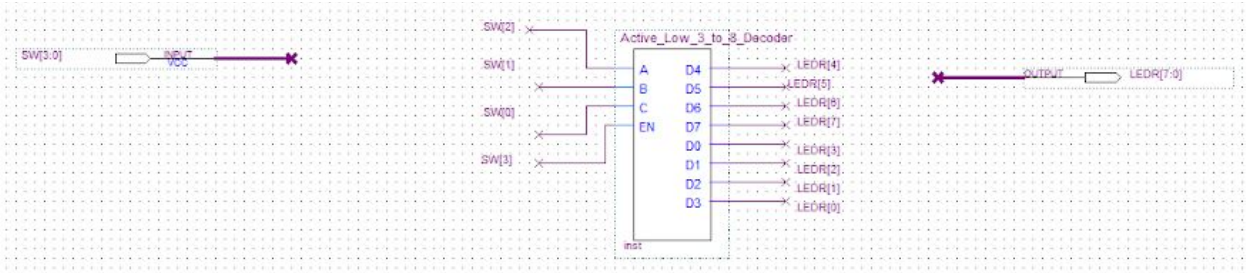
```
module  Active_Low_3_to_8_Decoder_V(A,B,C,EN,D0,D1,D2,D3,D4,D5,D6,D7); //module name

        input wire A; //defines an input A
        input wire B; //defines an input B
        input wire C; //defines an input C
        input wire EN; //defines an input Enable

        output wire D0; //defines an output wire D0
        output wire D1; //defines an output wire D1
        output wire D2; //defines an output wire D2
        output wire D3; //defines an output wire D3
        output wire D4; //defines an output wire D4
        output wire D5; //defines an output wire D5
        output wire D6; //defines an output wire D6
        output wire D7; //defines an output wire D7

        Active_Low_2_to_4_converter inst0 (B , C , ~( A & ~EN), D4,D5,D6,D7); // instantiates Active_Low_2_to_4_converter inst0
        Active_Low_2_to_4_converter inst1 (B , C , ~( ~A & ~EN), D0,D1,D2,D3); //instantiates Active_Low_2_to_4_converter inst1

endmodule //ends module
```

Timing Diagram for Active Low 3 to 8 Decoder:

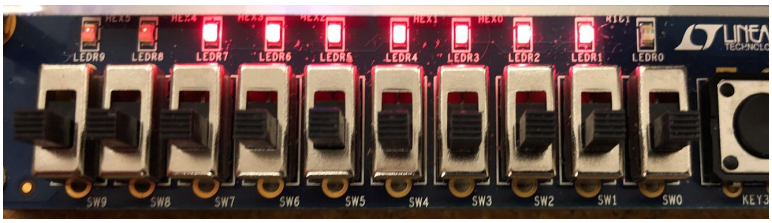
(*Note: In this wave diagram simulation an initial period offset of 10 nanoseconds in order to simulate the case where the enable inputs were set to a logic one.)

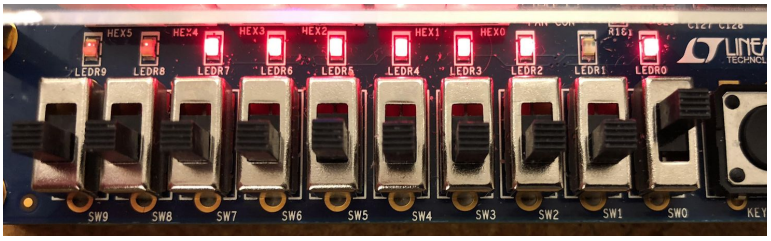


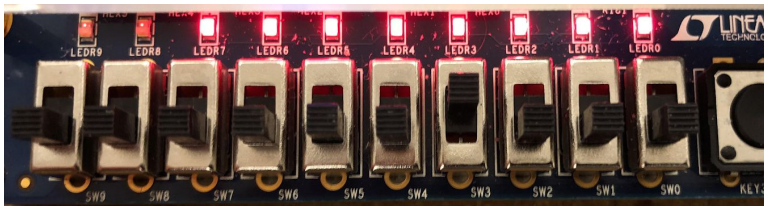
Active Low 3 to 8 Decoder Top Level Block diagram



Random Trial Pictures 3 to 8 Decoder





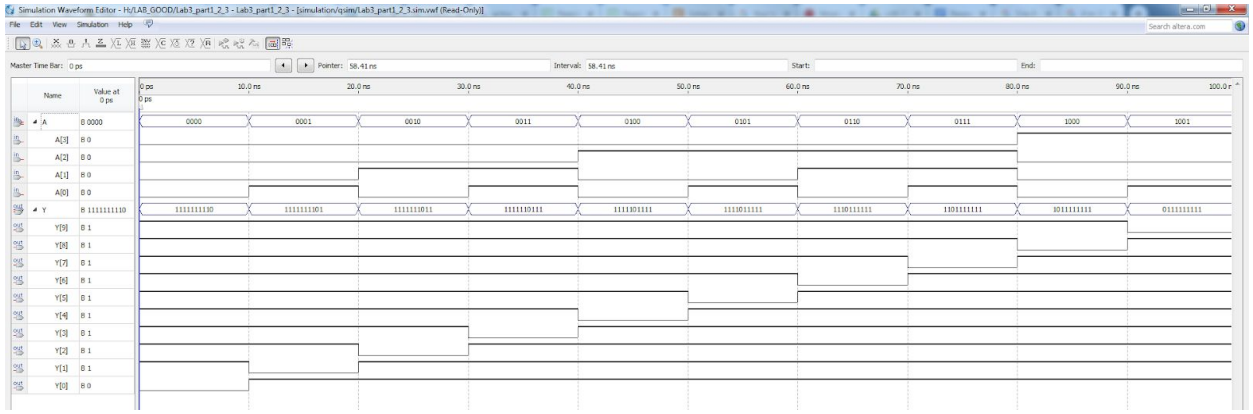


Part 4: BCD to Decimal Decoder - Verilog

```
module Decimal_Decoder (A, Y); //defines decimal decoder module

        input [3:0]A; //defines a 4 bit input for A
        output[9:0]Y; //defines a 10 bit output for Y

        assign Y[0]=  A[3] |  A[2] |  A[1] |  A[0]; //assigns POS form for Y[0]
        assign Y[1]=  A[3] |  A[2] |  A[1] | ~A[0]; //assigns POS form for Y[1]
        assign Y[2]=  A[3] |  A[2] | ~A[1] |  A[0]; //assigns POS form for Y[2]
        assign Y[3]=  A[3] |  A[2] | ~A[1] | ~A[0]; //assigns POS form for Y[3]
        assign Y[4]=  A[3] | ~A[2] |  A[1] |  A[0]; //assigns POS form for Y[4]
        assign Y[5]=  A[3] | ~A[2] |  A[1] | ~A[0]; //assigns POS form for Y[5]
        assign Y[6]=  A[3] | ~A[2] | ~A[1] |  A[0]; //assigns POS form for Y[6]
        assign Y[7]=  A[3] | ~A[2] | ~A[1] | ~A[0]; //assigns POS form for Y[7]
        assign Y[8]= ~A[3] |  A[2] |  A[1] |  A[0]; //assigns POS form for Y[8]
        assign Y[9]= ~A[3] |  A[2] |  A[1] | ~A[0]; //assigns POS form for Y[9]

endmodule //ends module
```

Timing Diagram for BCD Decoder:

(*Note: In this wave diagram simulation an initial period offset of 10 nanoseconds in order to simulate the case where the enable inputs were set to a logic one.)

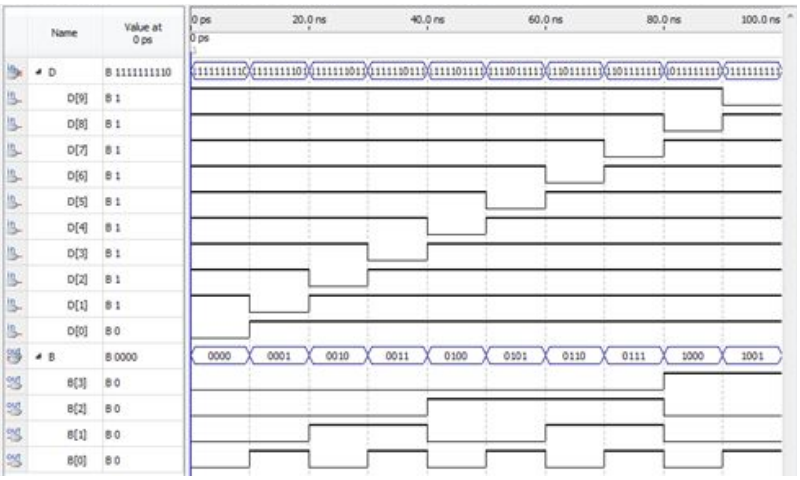


Part 5: Decimal to Binary Encoder: Verilog

```
module Decimal_Encoder (D, B); //defines Decimal Encoder Module

        input [9:0]D; //defines a 10 bit input wire D
        output[3:0]B; //defines a 4 bit output wire B

        assign B[0]=  ((D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  ~D[1] &  D[0])
                    |   (D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  ~D[3] &  D[2] &  D[1] &  D[0])
                    |   (D[9] &  D[8] &  D[7] &  D[6]  & ~D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |  (D[9] &  D[8] &  ~D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |  (~D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])); //assigns SOP form for B[0]

        assign B[1]=  ((D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  ~D[2] &  D[1] &  D[0])
                    |   (D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  ~D[3] &  D[2] &  D[1] &  D[0])
                    |  (D[9] &  D[8] &  D[7] &  ~D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |   (D[9] &  D[8] &  ~D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])); //assigns SOP form for B[1]

        assign B[2] = ((D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  ~D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |   (D[9] &  D[8] &  D[7] &  D[6]  & ~D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |  (D[9] &  D[8] &  D[7] &  ~D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |   (D[9] &  D[8] &  ~D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])); //assigns SOP form for B[2]


        assign B[3]=  ((D[9] &  ~D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])
                    |   (~D[9] &  D[8] &  D[7] &  D[6]  & D[5] &  D[4] &  D[3] &  D[2] &  D[1] &  D[0])); //assigns SOP form for B[3]

endmodule //ends module
```

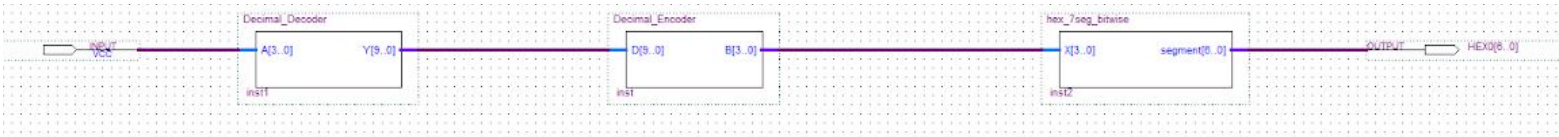
Timing Diagram for Decimal to Binary Encoder:

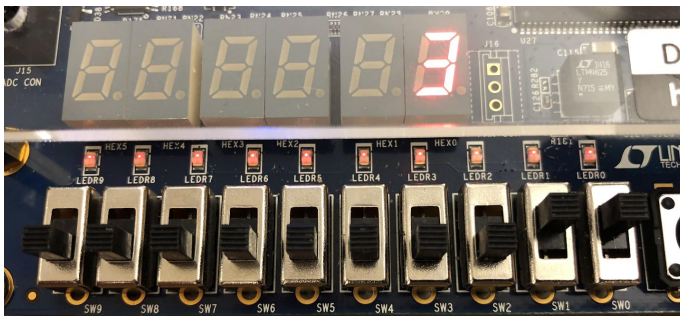
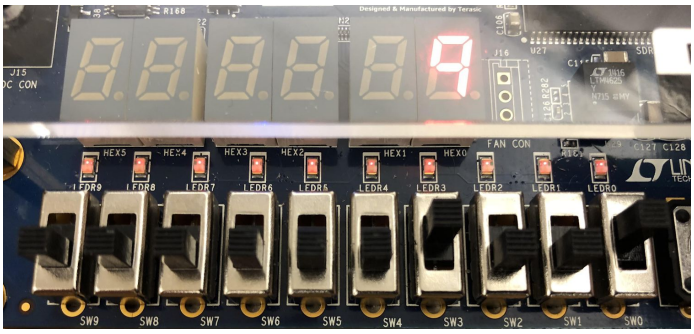
(*Note: In this wave diagram simulation an initial period offset of 10 nanoseconds in order to simulate the case where the enable inputs were set to a logic one.)



Part 6: Putting It All Together

Top Level Block Diagram



Pictures of FPGA board for test cases of displaying 3,9

.