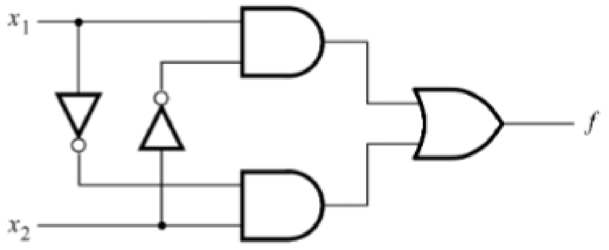
COMP ENG 2DI4
Digital Logic

Lab 3: Programmable Logic

Due date: Friday, November 8th, 2021
Submitted By:
Jil Shah
Shiv Thakar

Pre-Laboratory Preparation

1. Review the Quartus II Introduction file specified in section Resources -item 3. Complete the outlined example for the light controller circuit (an XOR circuit). Implement the model as Schematic Capture. Provide an annotated screen capture of your successful functional simulation. The annotation should show the relation between each input combination and respective output.[10 marks]



x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	0

Figure 1: The light controller circuit and truth table

Circuit

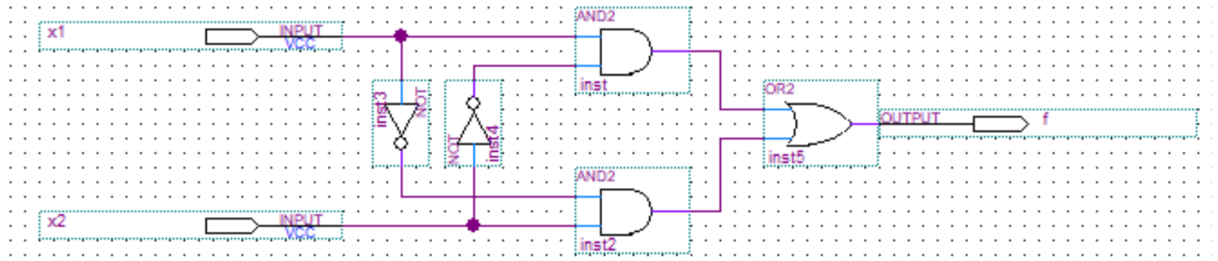


Figure 2: The schematic circuit built in Quartus

Simulation Results

The simulation tests for all possible cases.

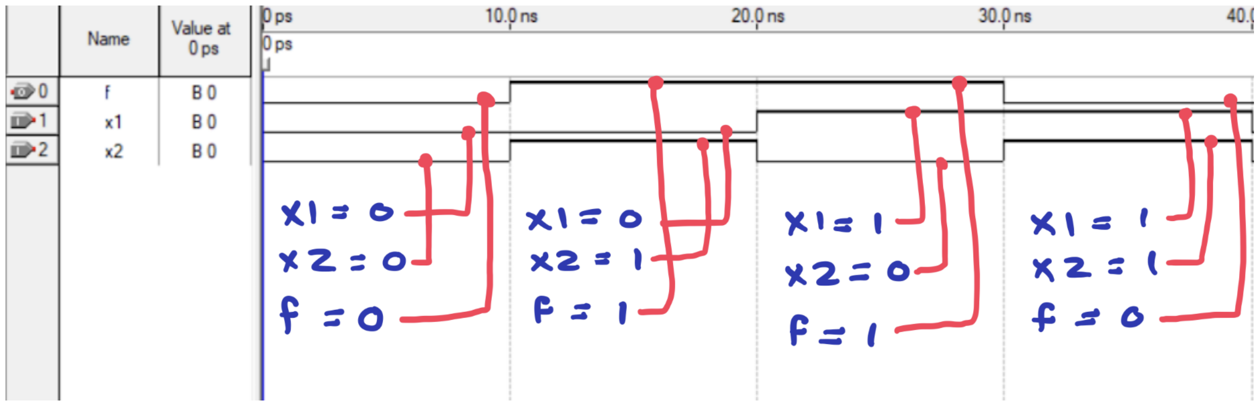


Figure 3: Simulation results with annotations for the light controller circuit.

2. Study the data sheet for the seven-segment display (MAN34x0A,36x0A,38x0A,70A). Design a minimized combinational logic circuit that takes a 2-bit binary code and drives the display to show the decimal equivalent (e.g., 00 = 0, 01 = 1, 10= 2, 11 = 3). Carefully note the required input to the seven-segment-display to light an individual segment. Show your design steps. Describe the resultant design using a Verilog modelling method of your choice. [10 mark]

Truth Table (an active HI was implemented)

X	Y	A	B	C	D	E	F	G
0	0	1	1	1	1	1	1	0
0	1	0	1	1	0	0	0	0
1	0	1	1	0	1	1	0	1
1	1	1	1	1	1	0	0	1

Table 1: Truth Table for a 2-input Seven-Segment-Display

K-map

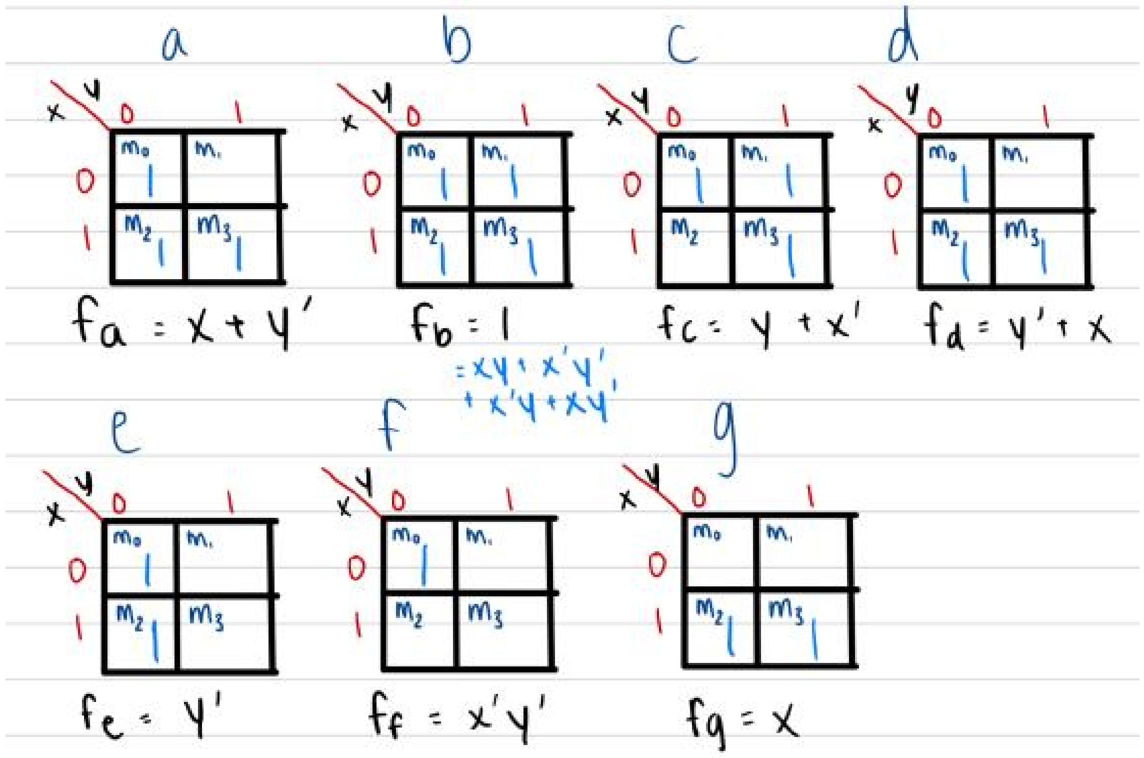


Figure 4: K-map for the 2-input Seven-Segment-Display

Circuit

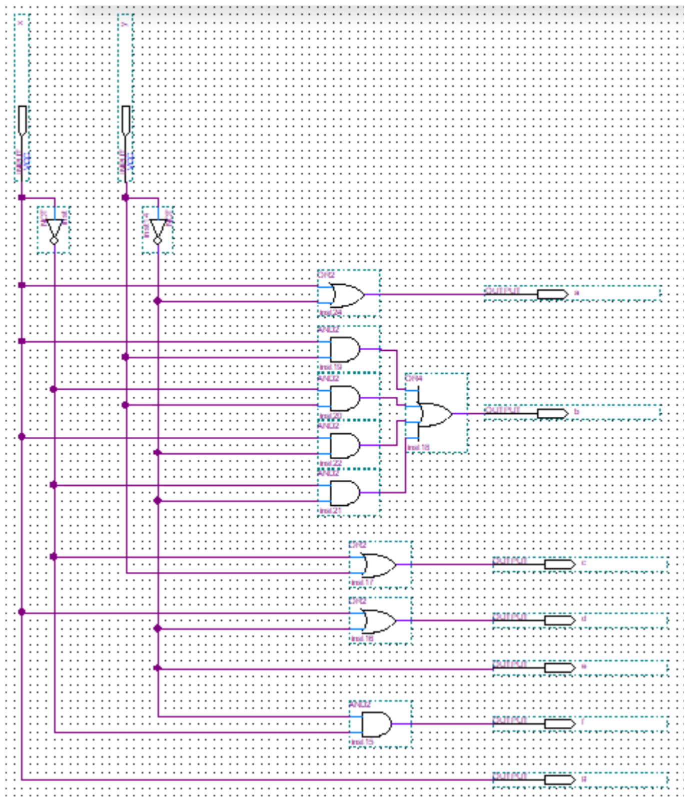
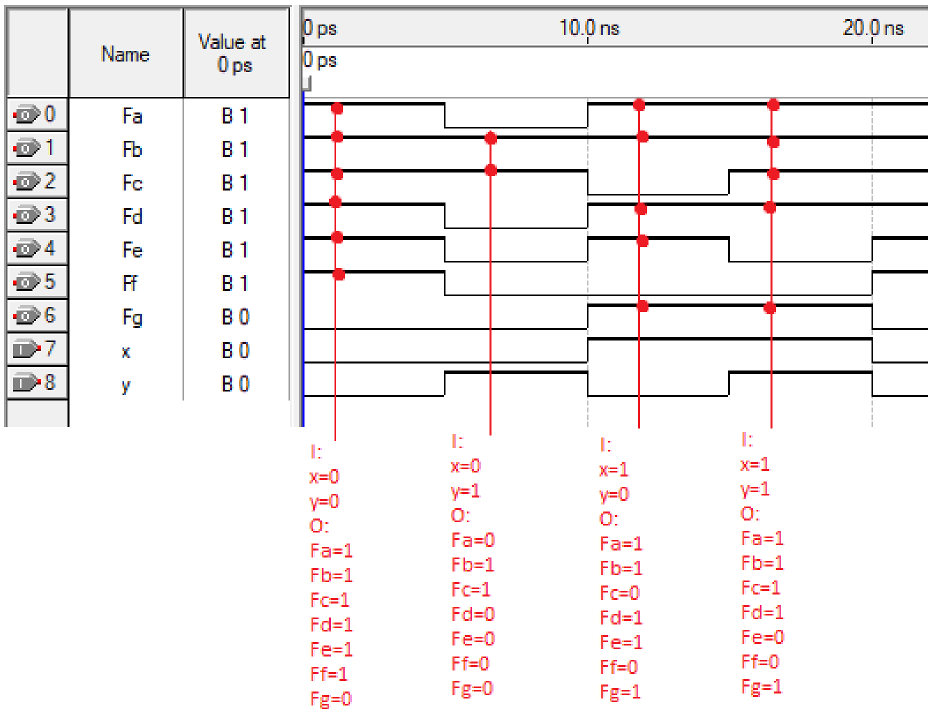


Figure 5: Circuit built in Quartus for the 2-input Seven-Segment-Display

Simulation Results



	Name	Value at 0 ps
0	Fa	B 1
1	Fb	B 1
2	Fc	B 1
3	Fd	B 1
4	Fe	B 1
5	Ff	B 1
6	Fg	B 0
7	x	B 0
8	y	B 0

0 ps
10.0 ns
20.0 ns
0 ps
I: x=0 y=0 O: Fa=1 Fb=1 Fc=1 Fd=1 Fe=1 Ff=1 Fg=0
I: x=0 y=1 O: Fa=0 Fb=1 Fc=1 Fd=0 Fe=0 Ff=0 Fg=0
I: x=1 y=0 O: Fa=1 Fb=1 Fc=0 Fd=1 Fe=1 Ff=0 Fg=1
I: x=1 y=1 O: Fa=1 Fb=1 Fc=1 Fd=1 Fe=0 Ff=0 Fg=1

Figure 6: Simulation results with annotations for the 2-input Seven-Segment-Display

3.5.1: The Lamp Controller in HDL

Truth Table

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Table 2: Truth table for the lamp controller

Verilog HDL Code

Placed in the appendix.

Simulation Results

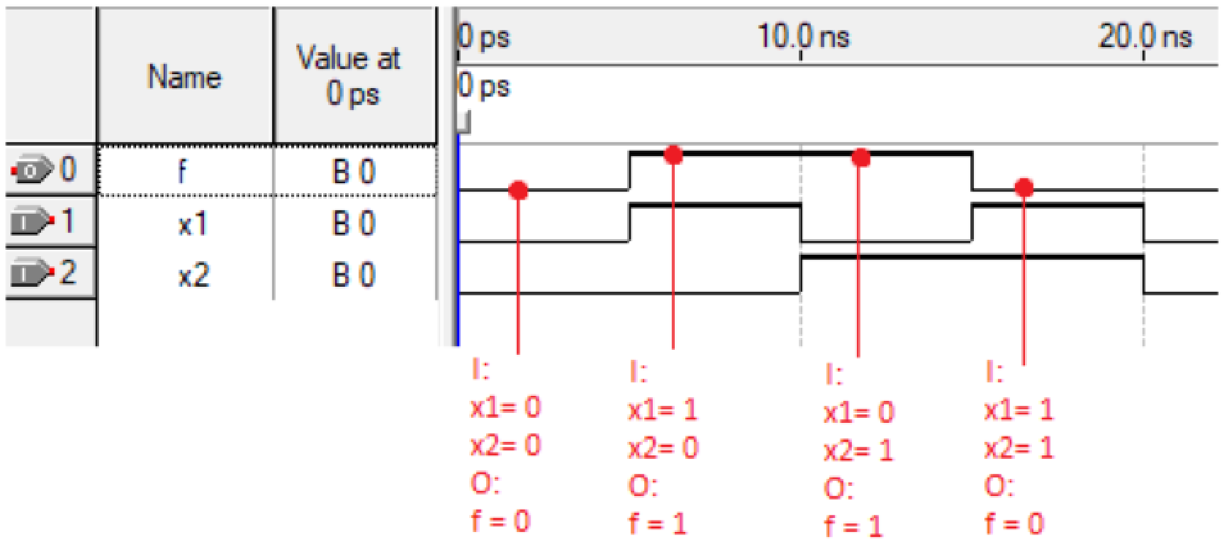


Figure 7: Simulation Results with annotations for the lamp controller

Description

Using the Vector Waveform file in Quartus II, we set the inputs (x1 and x2) to different binary values, so we can see if we are getting the proper output (f). Compared to the truth table we verified that our Verilog HDL code was correct.

3.5.2: 3-to-8 Decoder in HDL

Truth Table

x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Table 3: Truth Table for the 3-to-8 Decoder

Verilog HDL Code

Placed in the appendix.

Simulation Results

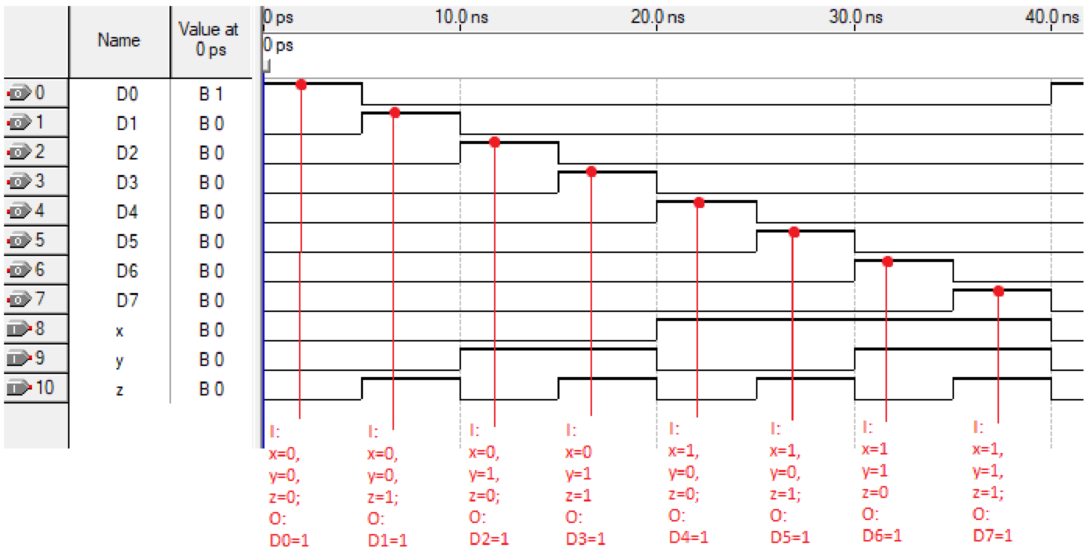


Figure 8: Simulation results and annotations of the 3-to-8 decoder

Description

After reviewing the truth table, we wrote a Verilog HDL code that describes a 3-to-8 bit decoder. To verify that the code is correct, a simulation was used to test different inputs to give different outputs. We then compared the simulation to our original truth table and therefore, we can confirm that the Verilog HDL code describes the right circuit.

3.5.3 “Programming” 3-to-8 Line Decoder

Circuit

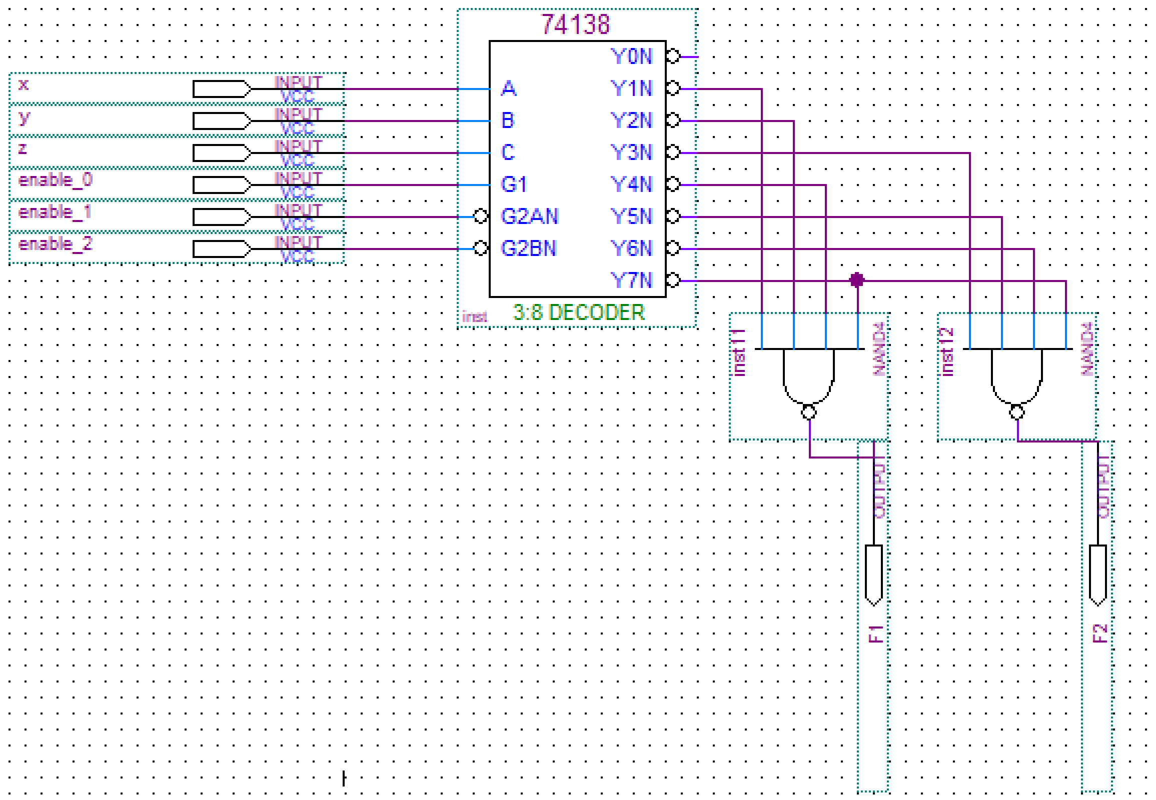


Figure 9: Circuit of the 3-to-8 Decoder, Full Adder

Circuit Simulation

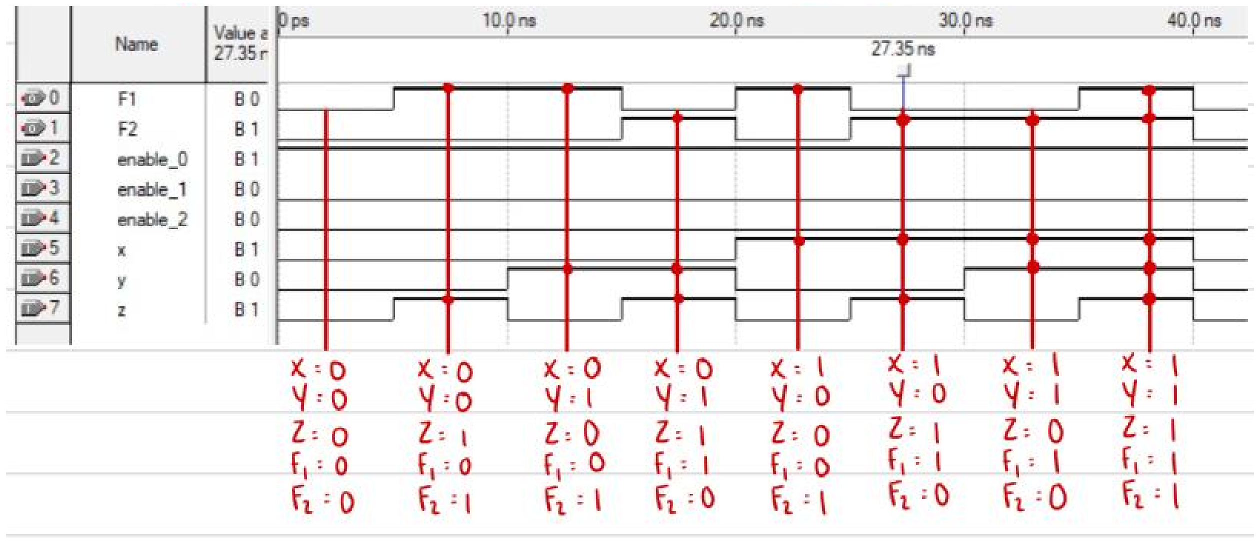


Figure 10: Simulation results and annotations of the 3-to-8 Decoder, Full Adder

Truth Table

x	y	z	F1 (Sum)	F2 (carry-out)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 4: Truth table for 3-to-8 Decoder for Full Adder

Verilog HDL Code

Placed in the appendix.

Simulation Results

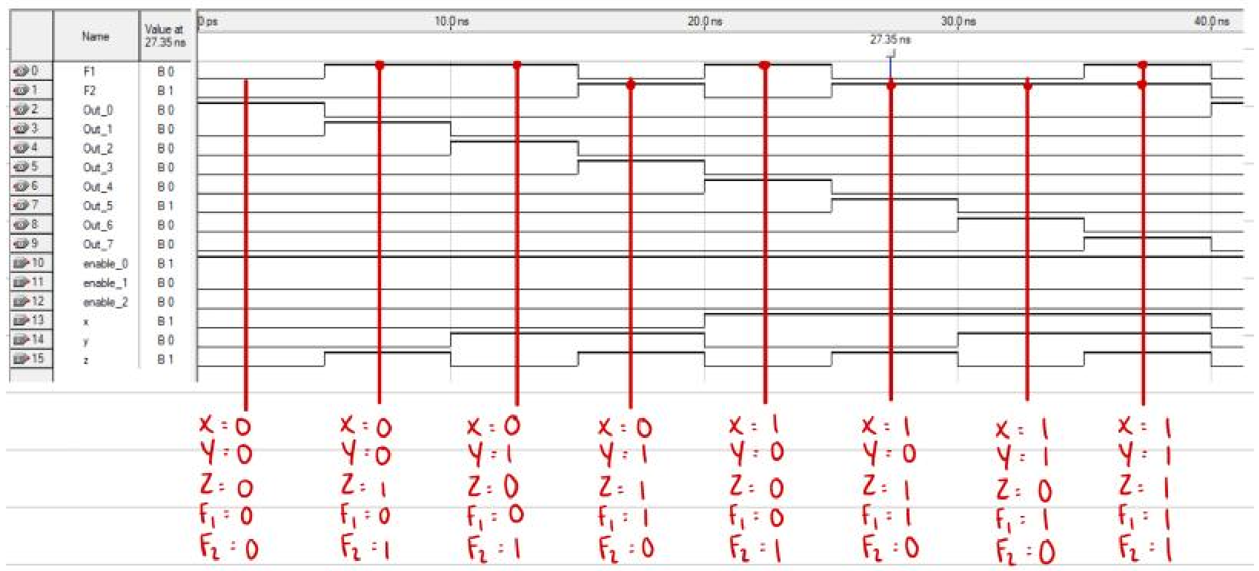


Figure 11: Simulation results and annotations of the 3-to-8 decoder (verilog), full adder.

Description

After reviewing the truth table, we wrote a Verilog HDL code that describes a 3-to-8 bit decoder with Full Adder. To verify that the code is correct, a simulation was used to test different inputs to give different outputs. We then compared the simulation to our original truth table and therefore, we can confirm that the Verilog HDL code describes the right circuit.

3.5.4 Binary to Hexadecimal Display Driver in HDL

Truth Table

	w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0

6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	1	1	0	0
A =10	1	0	1	0	0	0	0	1	0	0	0
b =11	1	0	1	1	1	1	0	0	0	0	0
C =12	1	1	0	0	0	1	1	0	0	0	1
d =13	1	1	0	1	1	0	0	0	0	1	0
E =14	1	1	1	0	0	1	1	0	0	0	0
F =15	1	1	1	1	0	1	1	1	0	0	0

Table 5: Truth table for the binary to hexadecimal display driver

K-Map Minimization

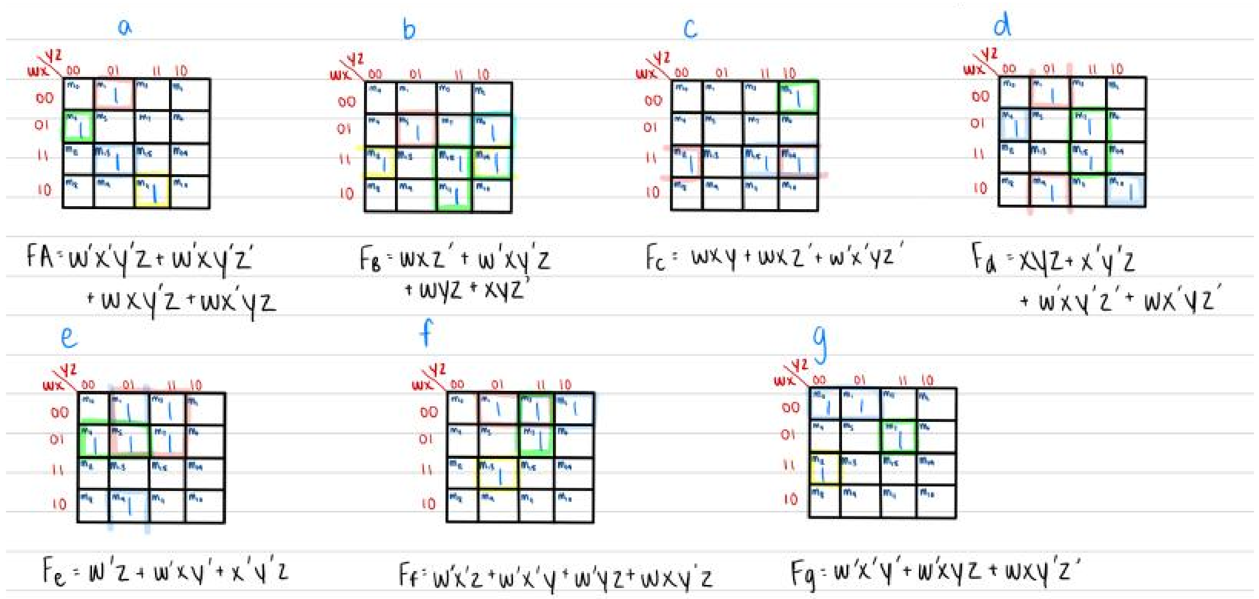

a
b
c
d
FA = w'x'y'z + w'xy'z' + wxy'z + wx'yz
FB = wxz' + w'xy'z + wyz + xyz'
Fc = wxy + wxz' + w'x'yz'
Fd = xyz + x'y'z + w'xy'z' + wx'yz'
e
f
g
Fe = w'z + w'xy' + x'y'z
Ff = w'x'z + w'x'y + w'yz + wxy'z
Fg = w'x'y' + w'xyz + wxy'z'

Figure 12: K-map Simplification of the binary to hexadecimal display driver

Verilog HDL

Placed in the appendix.

Simulation Results

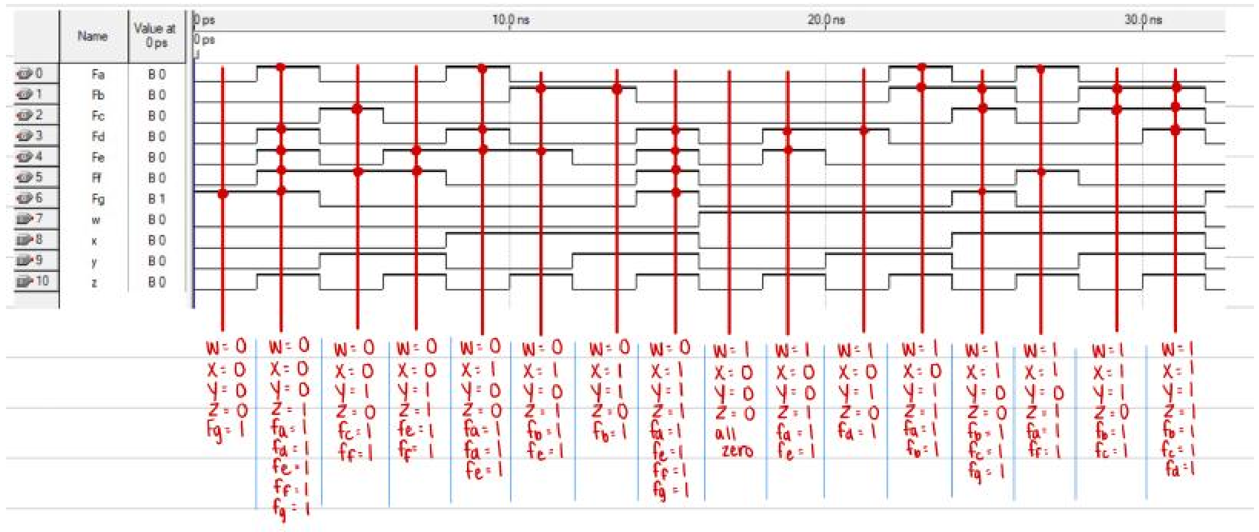
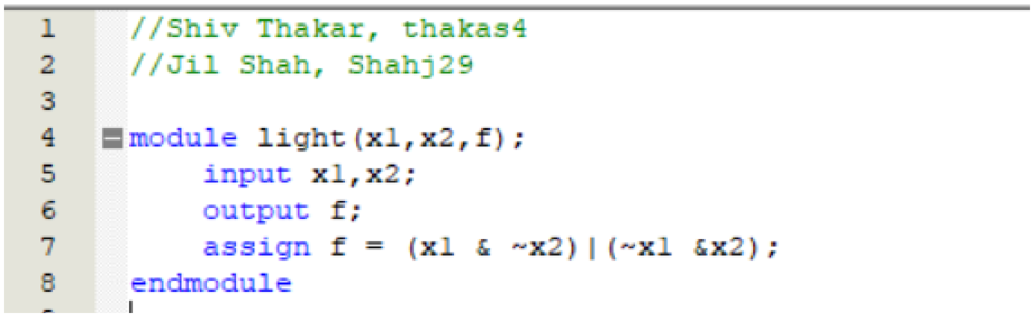


Figure 13: Simulation results and annotations of the binary to hexadecimal display driver

Description

After reviewing the truth table, we wrote a Verilog HDL code that describes a 4-to-7 bit decoder using a hexadecimal display. We used the truth table to create multiple K-Maps to determine the minimized boolean expression for each segment in the 7-segment display. These expressions were used to develop the Verilog HDL code to describe the circuit. To verify that the code is correct, a simulation was used to test different inputs to give different outputs. We then compared the simulation to our original truth table and therefore, we can confirm that the Verilog HDL code describes the right circuit.

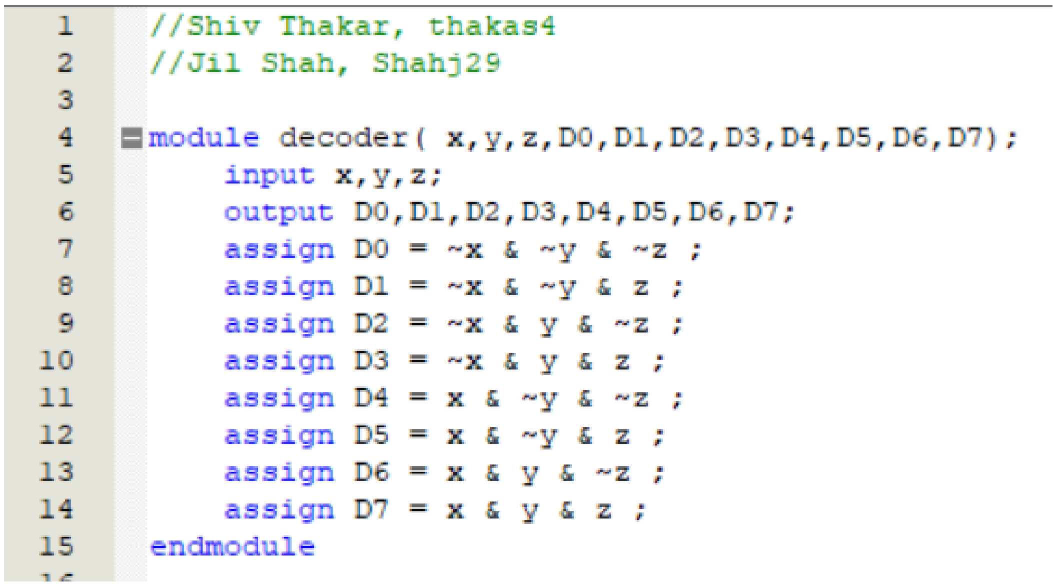
Appendix



```verilog
//Shiv Thakar, thakas4
//Jil Shah, Shahj29

module light(x1,x2,f);
    input x1,x2;
    output f;
    assign f = (x1 & ~x2)|(~x1 &x2);
endmodule
```

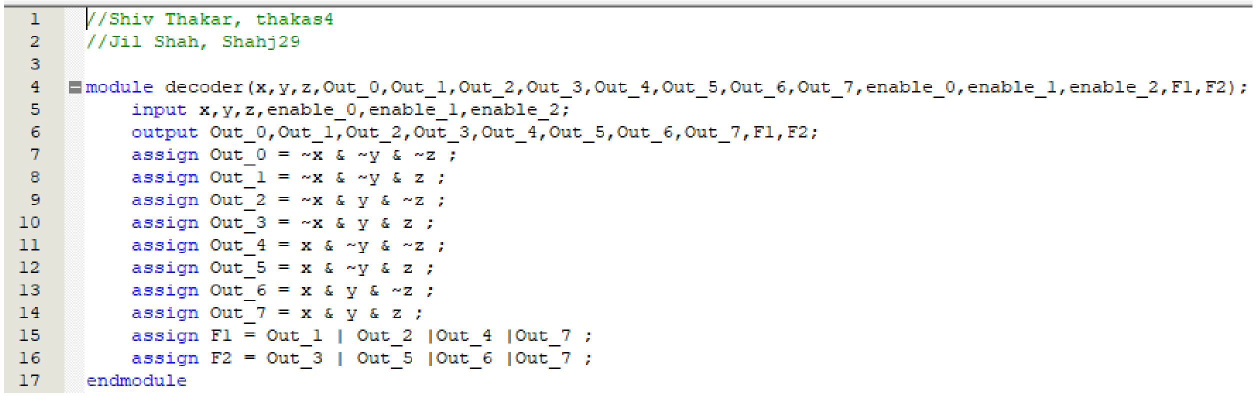
Figure A1: Code entry for the Lamp Controller in 3.5.1



```verilog
//Shiv Thakar, thakas4
//Jil Shah, Shahj29

module decoder( x,y,z,D0,D1,D2,D3,D4,D5,D6,D7);
    input x,y,z;
    output D0,D1,D2,D3,D4,D5,D6,D7;
    assign D0 = ~x & ~y & ~z ;
    assign D1 = ~x & ~y & z ;
    assign D2 = ~x & y & ~z ;
    assign D3 = ~x & y & z ;
    assign D4 = x & ~y & ~z ;
    assign D5 = x & ~y & z ;
    assign D6 = x & y & ~z ;
    assign D7 = x & y & z ;
endmodule
```

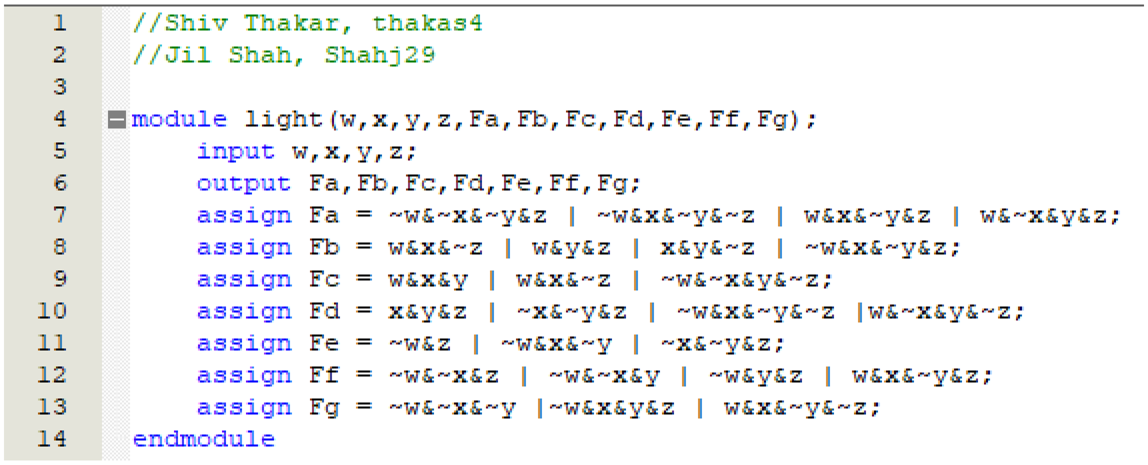
Figure A2: Code entry for the 3-to-8 Decoder in 3.5.2



```verilog
//Shiv Thakar, thakas4
//Jil Shah, Shahj29

module decoder(x,y,z,Out_0,Out_1,Out_2,Out_3,Out_4,Out_5,Out_6,Out_7,enable_0,enable_1,enable_2,F1,F2);
    input x,y,z,enable_0,enable_1,enable_2;
    output Out_0,Out_1,Out_2,Out_3,Out_4,Out_5,Out_6,Out_7,F1,F2;
    assign Out_0 = ~x & ~y & ~z ;
    assign Out_1 = ~x & ~y & z ;
    assign Out_2 = ~x & y & ~z ;
    assign Out_3 = ~x & y & z ;
    assign Out_4 = x & ~y & ~z ;
    assign Out_5 = x & ~y & z ;
    assign Out_6 = x & y & ~z ;
    assign Out_7 = x & y & z ;
    assign F1 = Out_1 | Out_2 |Out_4 |Out_7 ;
    assign F2 = Out_3 | Out_5 |Out_6 |Out_7 ;
endmodule
```

Figure A3: Code entry for the 3-to-8 Decoder, full adder in 3.5.3



```
//Shiv Thakar, thakas4
//Jil Shah, Shahj29

module light(w,x,y,z,Fa,Fb,Fc,Fd,Fe,Ff,Fg);
    input w,x,y,z;
    output Fa,Fb,Fc,Fd,Fe,Ff,Fg;
    assign Fa = ~w&~x&~y&z | ~w&x&~y&~z | w&x&~y&z | w&~x&y&z;
    assign Fb = w&x&~z | w&y&z | x&y&~z | ~w&x&~y&z;
    assign Fc = w&x&y | w&x&~z | ~w&~x&y&~z;
    assign Fd = x&y&z | ~x&~y&z | ~w&x&~y&~z |w&~x&y&~z;
    assign Fe = ~w&z | ~w&x&~y | ~x&~y&z;
    assign Ff = ~w&~x&z | ~w&~x&y | ~w&y&z | w&x&~y&z;
    assign Fg = ~w&~x&~y |~w&x&y&z | w&x&~y&~z;
endmodule
```

Figure A4: Code entry for the binary to hexadecimal display driver in 3.5.4