CS/EE 120A
(Section: 021)

Lab #3: Programming Combinatorial Logic on the BASYS FPGA Board

Dan Murphy; SID: 862xxxxxx
dmurp006@ucr.edu

Partner:
Jesse Garcia

Subject Overview

In this lab, we will be implementing the schematics of a sprinkler systems and a seven-segment decoder onto our FPGA board and essentially control what the board outputs/displays.

Part 1:

After implementing the schematic from our previous lab, we mapped all of our inputs from the board to the schematic then implemented onto the FPGA board to show output via LED lighting.

Part 2:

We utilized a seven-segment decoder schematic to implement into our FPGA board. Next, we converted the schematic into Verilog code and simulated it through the FPGA board. This was demonstrated through the display of numbers 1-9 and A-F.

New Concepts

1. Describe how this lab built upon previous ones:

This lab was highly dependent on a successful previous lab. We essentially used the previous lab as a building block to create this lab, thus allowing us to see the direct communication between software and hardware.

2. Describe the most difficult part of this lab for you:

By far, the most difficult part was figuring out how to maneuver around the errors within Verilog. Not being fluent let alone familiar with the language

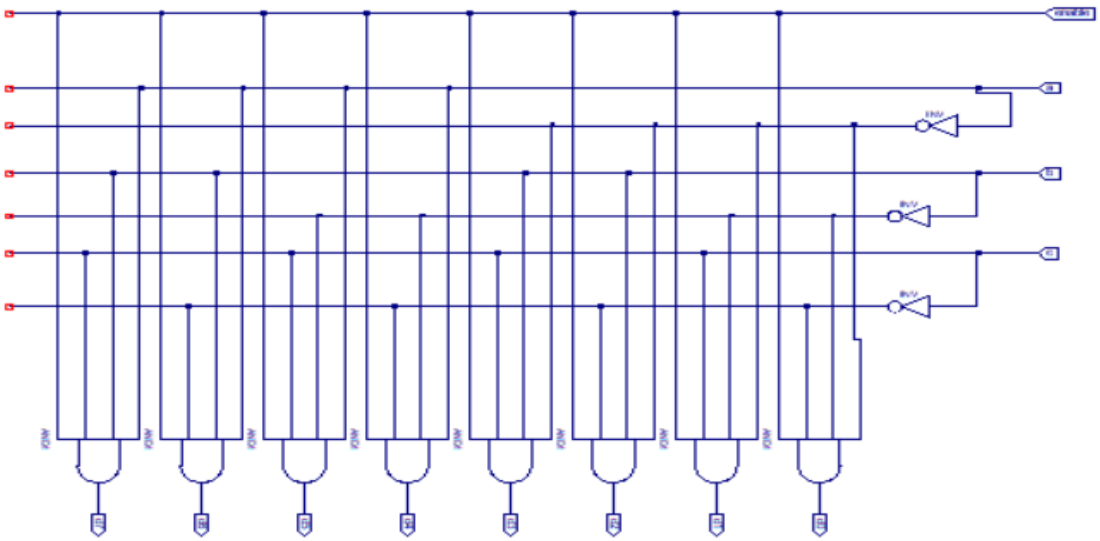
3. Describe problems you faced and how you solved them:

After realizing that we were not the only group suffering from frustration and confusion, we pooled knowledge and debugging-type techniques from various groups and the TA. Eventually, brute force plug-and-chug methods permitted successful implementation.

4. Do you verify that the code included with this report is yours and your partner's original work (yes/no)?

Yes. For the purpose of verifying it is our original code, yes; with the caveat of group collaboration outside of myself and Jesse Garcia. Ideas were shared, code was not. Thus, our code is inherently original.

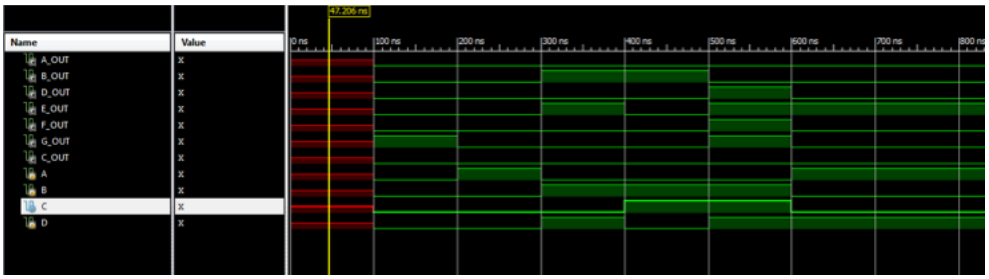
Sprinkler Schematics



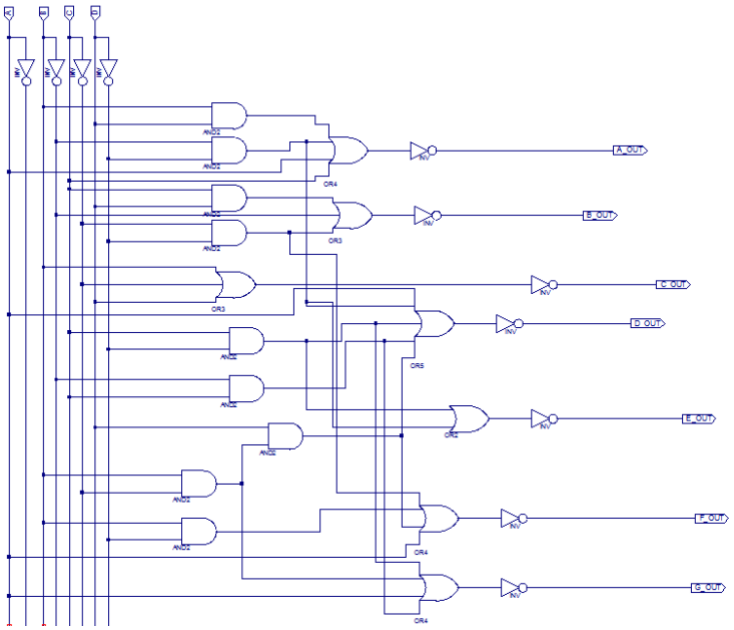
Sprinkler UCF to FPGA

```
1   NET "A" LOC = "A7" ;
2   NET "B" LOC = "M4" ;
3   NET "C" LOC = "C11" ;
4   NET "E" LOC = "G12" ;
5
6   NET "d0" LOC = "G1";
7   NET "d1" LOC = "P4";
8   NET "d2" LOC = "N4";
9   NET "d3" LOC = "N5";
10  NET "d4" LOC = "P6";
11  NET "d5" LOC = "P7";
12  NET "d6" LOC = "M11";
13  NET "d7" LOC = "M5";
```

Sprinkler Simulation Output



Decoder Schematics



Test Harness

```
38  // Initialize Inputs
39  initial begin
40  #100;  //0
41  A = 0;
42  B = 0;
43  C = 0;
44  D = 0;
45  #100; //8
46  A = 1;
47  B = 0;
48  C = 0;
49  D = 0;
50  #100; //5
51  A = 0;
52  B = 1;
53  C = 0;
54  D = 1;
55  #100; //6
56  A = 0;
57  B = 1;
58  C = 1;
59  D = 0;
60  #100; //7
61  A = 0;
62  B = 1;
63  C = 1;
64  D = 1;
65  #100; //9
66  A = 1;
67  B = 0;
68  C = 0;
69  D = 1;
70
```

Seven-Segment Decoder UCF on FPGA

```
NET "sw3" LOC = "A7" ;
NET "sw2" LOC = "M4" ;
NET "sw1" LOC = "C11" ;
NET "sw0" LOC = "G12" ;

NET "a" LOC = "L14";
NET "b" LOC = "H12";
NET "c" LOC = "N14";
NET "d" LOC = "N11";
NET "e" LOC = "P12";
NET "f" LOC = "L13";
NET "g" LOC = "M12";

NET "an0" LOC = "K14";
NET "an1" LOC = "M13";
NET "an2" LOC = "J12";
NET "an3" LOC = "F12";
```

Seven-Segment Test Harness

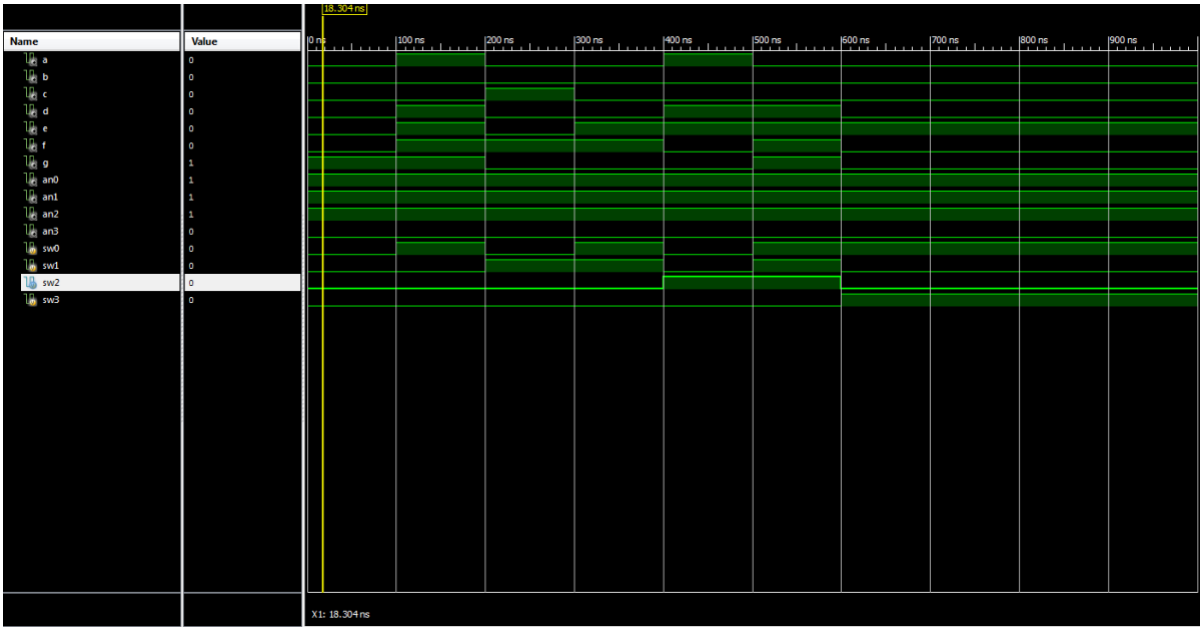
```
case ( bundle )
4'b0000 : begin // 0
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b0 ;
end

4'b0001 : begin // 1
a = 1'b1 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b1 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b1 ;
end

4'b0010 : begin // 2
a = 1'b0 ;
b = 1'b0 ;
c = 1'b1 ;
d = 1'b0 ;
e = 1'b0 ;
f = 1'b1 ;
g = 1'b0 ;
end

4'b0011 : begin // 3
a = 1'b0 ;
b = 1'b0 ;
c = 1'b0 ;
d = 1'b0 ;
e = 1'b1 ;
f = 1'b1 ;
g = 1'b0 ;
end
```

Seven-Segment FPGA Simulation



Questions

1. Can there be a difference in logical behavior between the intended logic entered and simulated and, the logic actually synthesized for FPGA? Why?

   Of course! How the programmer intends their logic to work can be vastly different from the interpretation from the hardware's frame of reference – this is why debugging is very important. Something can vary from an overlooked expression within code to a wonky or mismatched circuit within the schematic. Simulations show what the hardware interprets from the information its given and intended logic from the programmer does not necessarily reflect the same information (conversion of human language to computer language is an art in itself).

2. Why do we need a configuration file?

   Configuration files are used to structure parameters and initialize (i.e. configure) settings within the program.

3. Is there a functional difference in circuitry between Lab 1, Part 3 and BASYS board for this particular application?

   There is no functional difference between the two for this particular application.

4. What must be done in order to use switches SW3 and SW7 instead of SW0 and SW1? How about using LED5 instead of LED0?

   In order to use SW3 and SW7 instead of SW0 and SW1, the UCF file needs to be modified where the values simply need to be swapped where the appropriate values are mapped on the FPA board.

Conclusion

The lab served its purpose of allowing us to become more familiarized with Verilog and implementing code which directly affects hardware. In particular, we learned about combinatorial logic within our FPGA board through Verilog code. This was shown through the seven-segment LED's on our FPGA board via testbenches.

As a whole, the circuit design and Boolean logic allowed us to see the implementation of the code within Verilog. Once the results expected were yielded, our LED output confirmed successful software to hardware communication.