Quinn Roemer

Engineering – 303

Lab 14

5/5/2017

Introduction/Description

In this lab, I finally got to finish constructing my single cycle computer and run a program on it! The goal of this lab was to take all of the parts that we have been working on for the past several weeks and incorporate them all into one design for a computer. In this lab, I had to create a total of two circuits. The first was a Verilog circuit that was designed to program the computer. The second was the computer itself.

Design

Part 0 – Instruction Memory

The first circuit that I had to create for this lab was called Instruction memory. This circuit was designed to push a program into the computer. Although most computers are programmed through software we are going to program the computer we made in class through hardware.

Here is the Verilog program for the circuit.

```
1  //A simlulated assebly language program implemented as a hard wired circuit.
2  module InstructionMemory (Address, IR);
3
4  input [7:0] Address;
5  output [15:0] IR;
6
7  reg[15:0] IR;
8
9  //Parameters being renamed for easier readability.
10 //This makes it easier to write new programs.
11 parameter //See pg 469.
12 //Register Instructions,      Opcode, DR, SA, SB
13 MOVA = 7'b0000000,
14 INC =  7'b0000001,
15 ADD = 7'b0000010,
16 SUB = 7'b0000101,
17 DEC = 7'b0000110,
18 AND = 7'b0001000,
19 OR = 7'b0001001,
20 XOR = 7'b0001010,
21 NOT = 7'b0001011,
22 MOVB = 7'b0001100,
23 SFTR = 7'b0001101,
24 SFTL = 7'b0001110,
25 LOAD = 7'b0010000,
26 ST = 7'b0100000,
27 //Immediate Instructions      Opcode, DR, SA, OP
28 LDI = 7'b1001100,
29 ADI = 7'b1100001,
30 //Jump Branch Instructions, Opcode, AD, SA, AD
31 BRZ = 7'b1100000,
32 BRN = 7'b1100001,
33 JMP = 7'b1110000,
34 //Registers
35 RO = 3'b000,
36 R1 = 3'b001,
37 R2 = 3'b010,
38 R3 = 3'b011,
39 //Numbers
40 ZERO = 3'b000,
41 ONE = 3'b001,
42 TWO = 3'b010,
43 THREE = 3'b011,
44 FOUR = 3'b100,
45 //NULL
46 NULL = 3'b000;
```

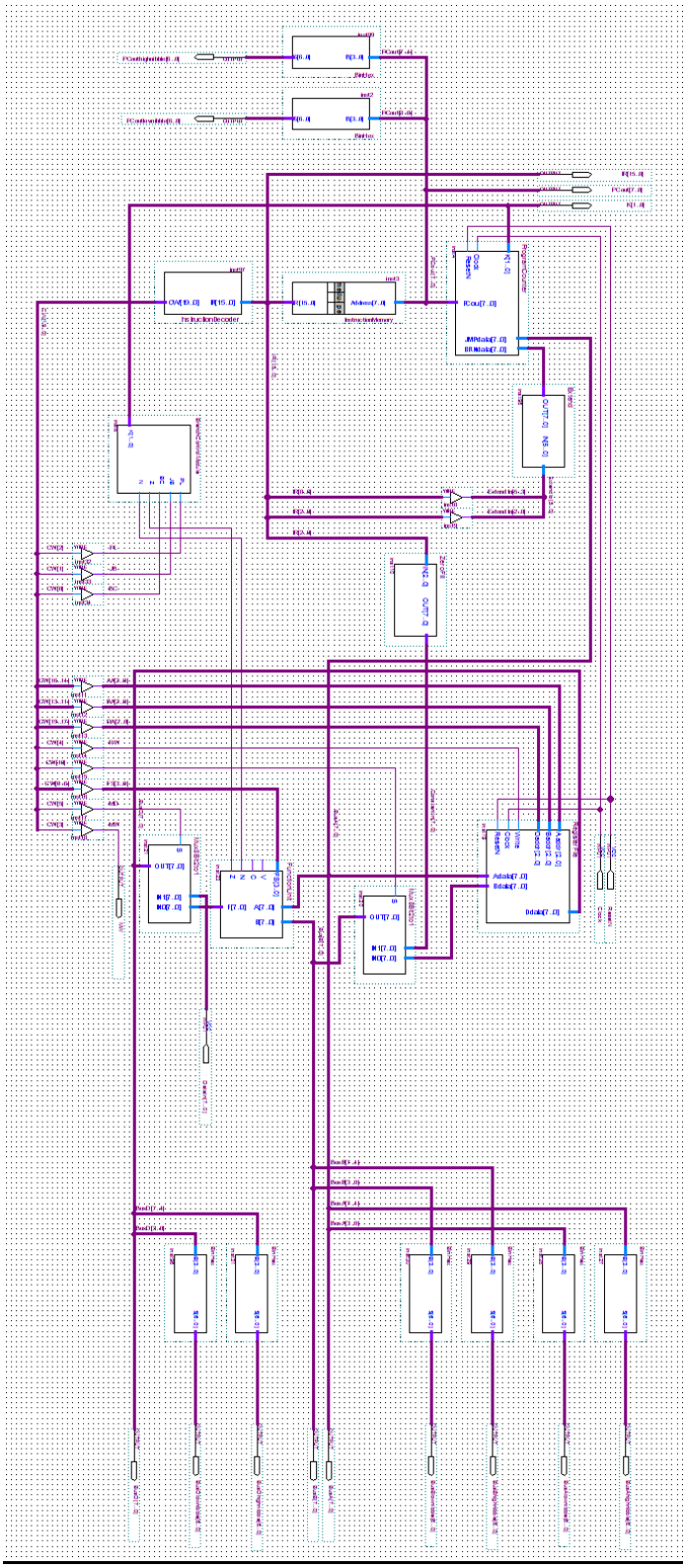
Continued on next page…

```
48    //The actual simulated assembly language "program" starts here
49    //This program multiplies two given inputs A * B = P
50    always@(Address)
51  begin
52      case(Address)
53        0: IR <= {LOAD, RO, NULL, NULL}; //Load A
54        1: IR <= {LOAD, R1, NULL, NULL}; //Load B
55        2: IR <= {LDI, R2, NULL, ZERO}; //Set P to 0
56        3: IR <= {LDI, R3, NULL, FOUR}; //Load Jump Address
57    //Loop
58        //If A = 0 then branch to done (binary 8)
59        4: IR <= {BRZ, 3'b001, RO, 3'b000};
60        5: IR <= {ADD, R2, R2, R1}; //Add B to p
61        6: IR <= {DEC, RO, RO, NULL}; //Decrement A
62        7: IR <= {JMP, NULL, R3, NULL}; //Jump to Loop
63    //Done
64        8: IR <= {ST, NULL, NULL, R2}; //Output Answer
65        default
66           IR <= 255;
67        endcase
68
69    end
70    endmodule
```

Part 1 – Single Cycle Computer

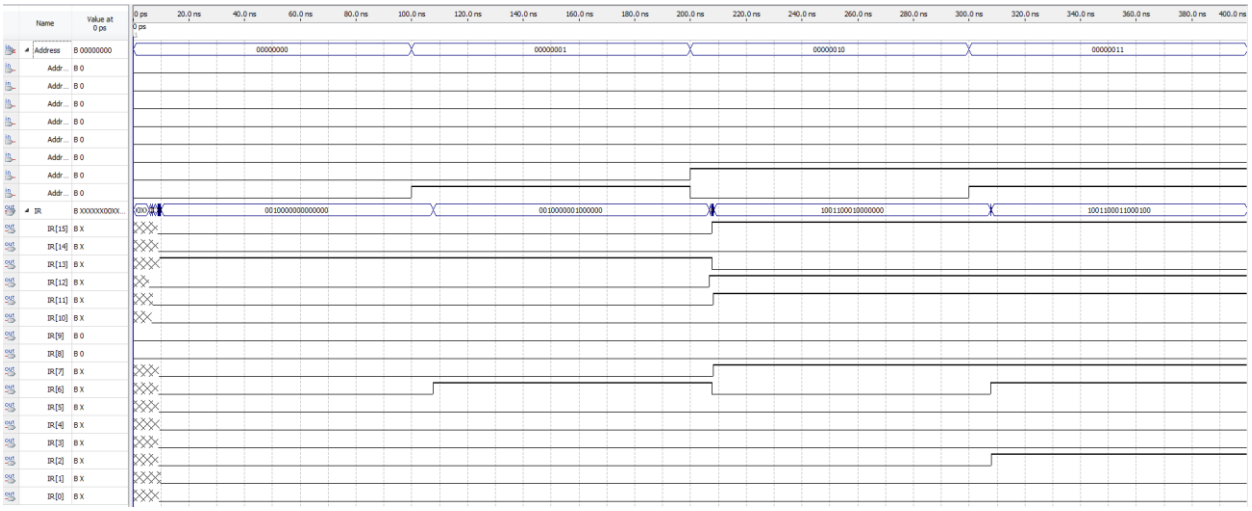
The final circuit I had to create in this lab was called the Single Cycle Computer. This circuit incorporated elements from many of my previous labs. This circuit is capable of accepting a programing circuit that will tell it to perform certain operations in a certain order. Please note, because this circuit is so large I inserted the picture sideways.

Here is the Block-Diagram for the circuit.

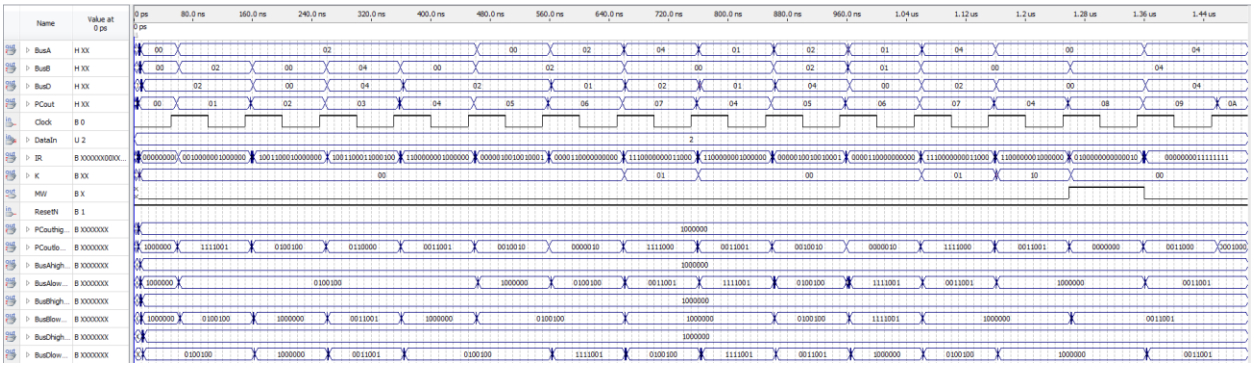


Testing

When testing the Instruction Memory I encountered no problems and it performed as expected for every single input combination.



When testing the Single Cycle Computer I encountered one small problem. Since the circuit was so huge I missed a couple spots where bus wires should've connected to each other. However, after fixing this issue this circuit performed as expected for every single input combination.



Conclusion

In this lab, I learned how to create a simple computer from many of the parts that we made in class. I learned that even the smallest mistake in wiring your circuit can produce a whole boatload of errors. I am delighted that this small computer that we have been working toward the entire semester is finally complete. I personally am very excited at learning how to program this thing

but I am also a little intimidated. The Verilog code that we need to write to run programs on this circuit looks very intense.