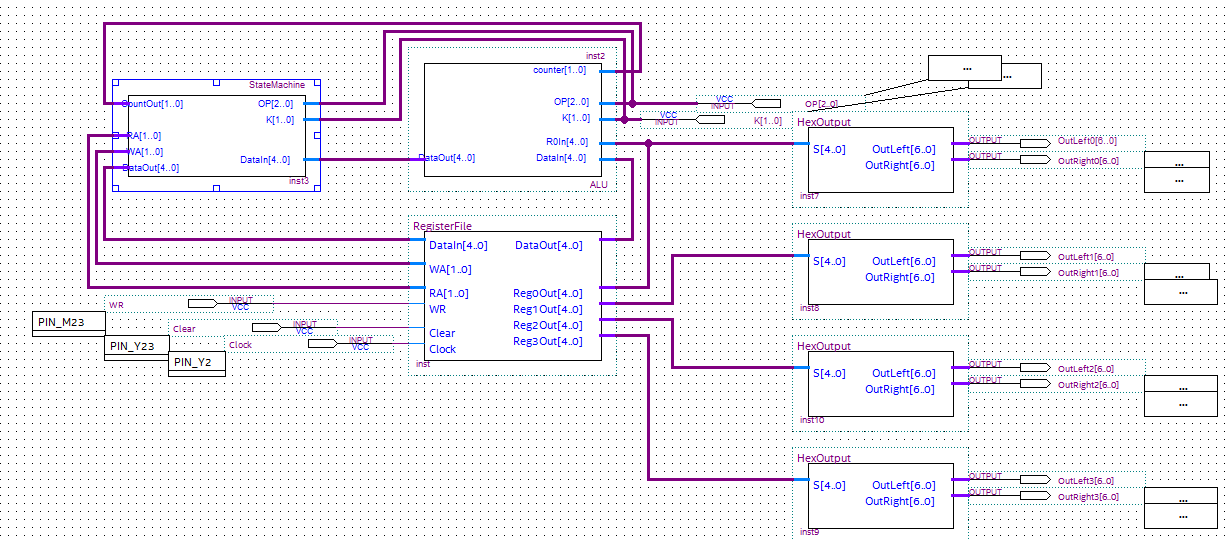
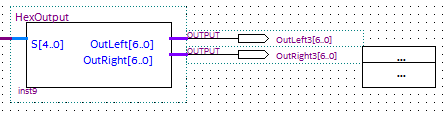
Aaron Goff  
CprE 281  
5/3/19

Final Project report

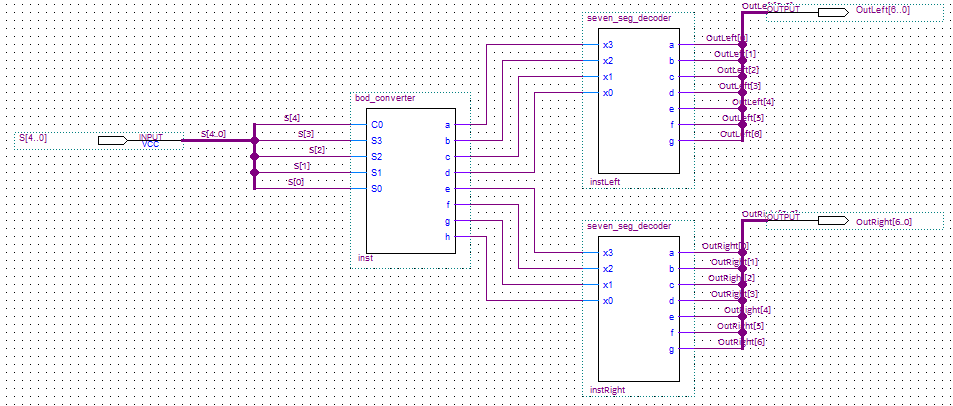
For the final project, we were told to make a calculator based from a Register File, an ALU, and a Finite State machine. The calculator will output to all 8 hex outputs. The register file stores the data and outputs to both the eight 7-digit hex outputs on the Altera board, and also feeds data back into the ALU and finite state machine for modification.



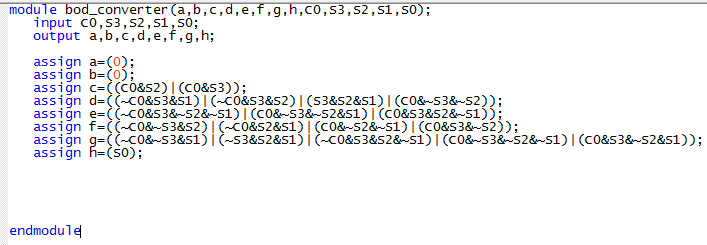
**HexOutput blocks:**  


The hex output blocks are what takes data from each register output from the register file. The HexOutput then modifies the signals in order to combine and output 5-bit numbers for display on the 7-digit hex outputs on the Altera board.

Inside the HexOutput block:

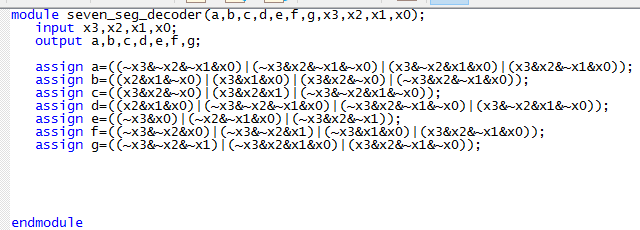


Inside the HexOutput block is a BCD Converter feeding into two separate seven seg\_decoders. A bus is used for each input and output of the file keeping the wires to a minimal. The BCD converter will take the information from the register and split it in order to read numbers 0-9 on each seven seg decoder. It does this by black magic and cheese whiz as seen below.  
  
Inside the BCD converter:



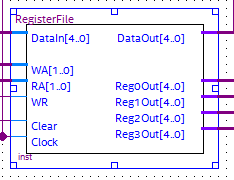
All that it’s doing here is taking a five bit input and outputting two different four bit values to be received by the seven seg decoder.

Inside the seven seg decoder:



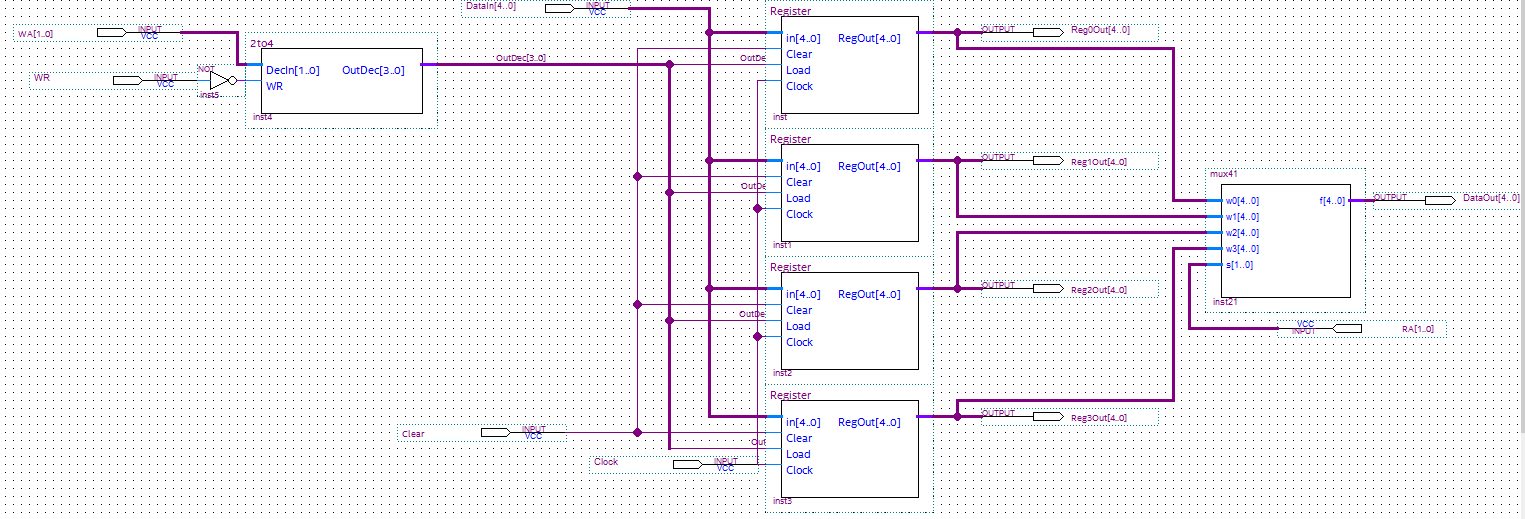
The seven seg decoder is taking those 4 bit inputs and modifying them to 7 bit outputs for use on the seven digit displays used later on in the program.

**Register File:**



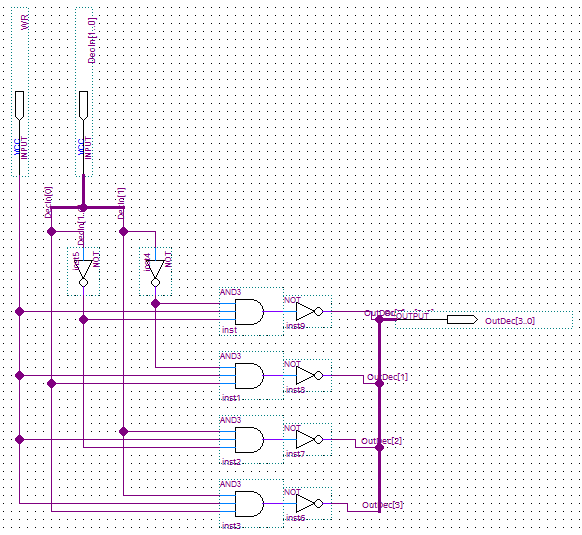
The register file takes in data that has been modified from the finite state machine and AUL, then stores the numbers and outputs to the registers. It also has a data output that feeds back into the ALU for further calculations.

Inside the register file:



Held within the register file is a 2 to 4 encoder, a 4 to 1 multiplexer, and four registers. The 2 to 4 encoder switches between two inputs and gives write access to one register at a time, depending on the position of the two input switches. The encoder also has a write access switch which will only allow to write to any register if it’s enabled. The register file will always write out to the 7 digit displays, however from there all four busses are combined to be read by a 4 to 1 mux that is switched in order to take specific information from a single decided register to output and be modified within the ALU.

Inside the 2 to 4 encoder:

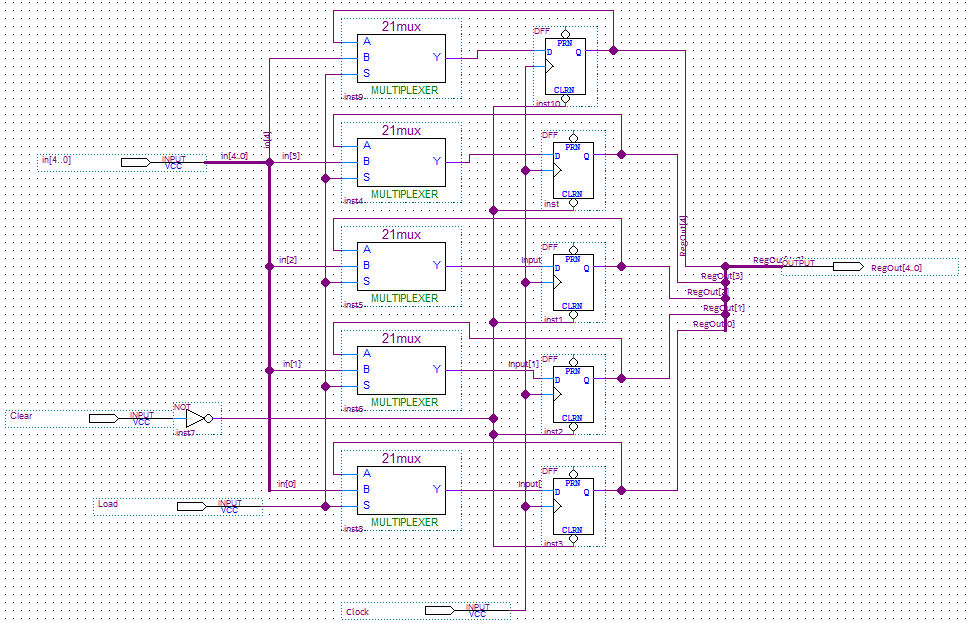


Fancy stuff. It’s basically just a truth table as shown:

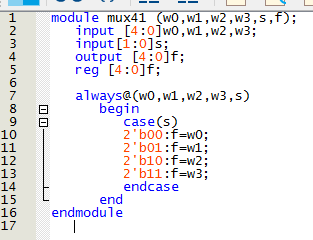
WR Y1 Y2 Reg0 Reg1 Reg2 Reg3  
 0 0 0 x x x x  
 0 0 1 x x x x  
 0 1 0 x x x x  
 0 1 1 x x x x  
 1 0 0 1 0 0 0  
 1 0 1 0 1 0 0  
 1 1 0 0 0 1 0  
 1 1 1 0 0 0 1

So it’s only allowing one register to write at a time and, as stated previously, will only allow the registers to write when the WR is enabled.

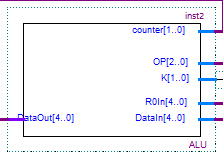
Inside each register:



Held within each register are five different 2-to-1 muxes that feed into five D-flip flops. This is a pretty standard way to storing a 5-bit number. The load allows the numbers to be passed through the multiplexer and into the D-flip flop which continues to pass back into itself each clock cycle until new information in read in. Once load is no longer enabled the number is stuck in there until load is either enabled again or the clear button has been switched, in which case it will clear all flip flops to 0. The flip flops are then combined into a 5 digit output.

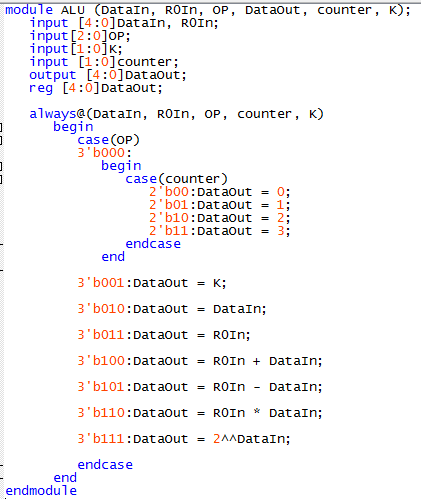
Inside the 4-to-1 mux:  
  
  
  
Pretty standard stuff with the truth table already clearly listed within the program. All that it’s doing is taking the bus from each register and combining them into a single bus for an output to be modified with the ALU and state machine. It’s being switched by the Read Access in order to pull information from only the single needed register.

**The ALU:**



The ALU is where all of the calculations are done and submitted into the state machine which is then output and stored within the register file which is then output onto the ALU board. The ALU assigns predetermined value to the register and calculates values together. The OP determines which calculation is to be performed, K is which inputs (0-3) are to be assigned, or which register to assign them to (depending on the OP positions). R0in takes information from register 0 as it’s the most commonly used register in the system, and Datain and Dataout do exactly as they’re named, they pass data in from the register we’re reading from, and passing the data out into the state machine to be modified and assigned to the register required.

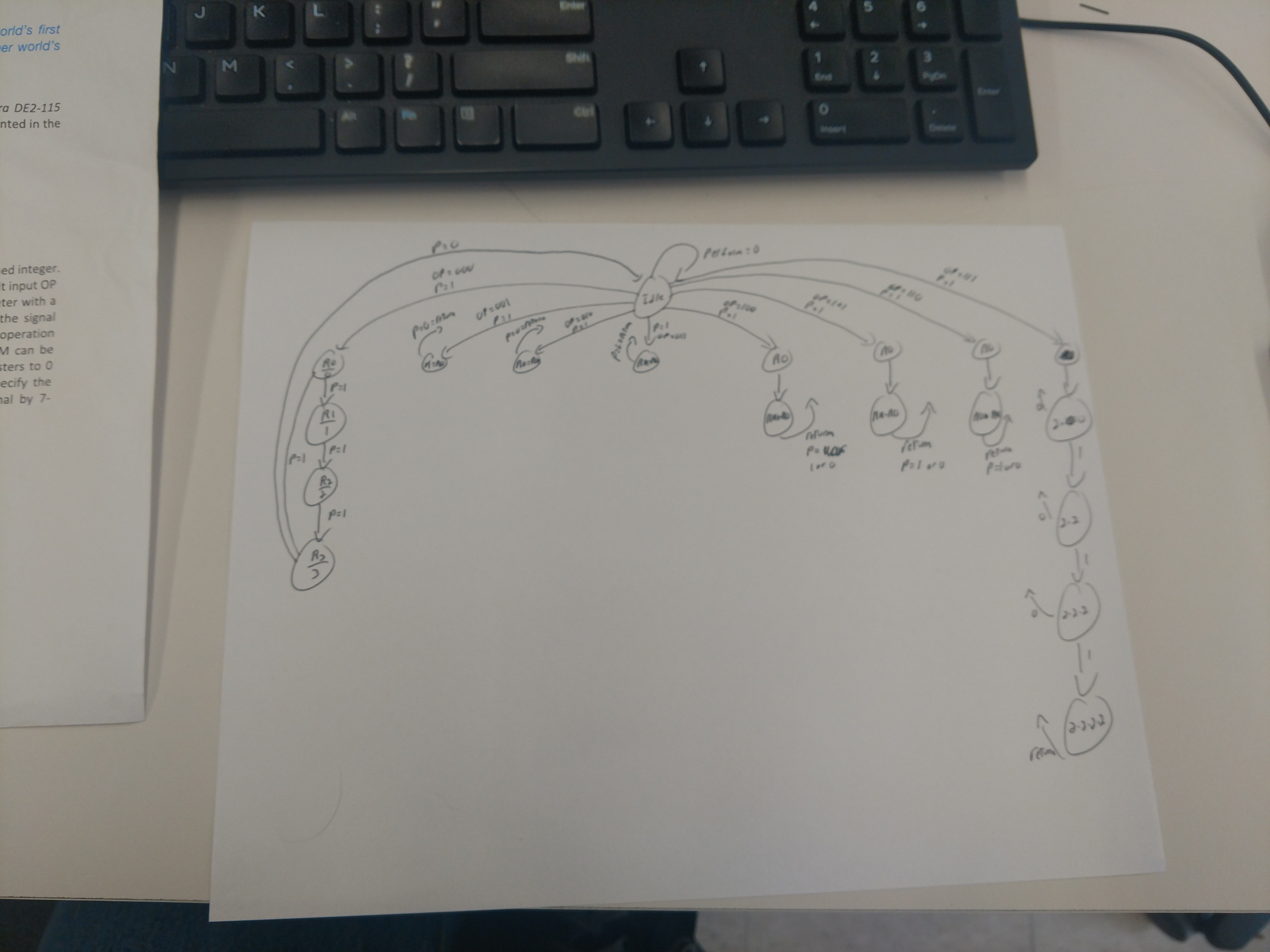
Inside the ALU:



Held within the ALU is all the calculations to be performed as seen. The OP 000 is a special case that requires a counter and a loop in order to write the values 0 1 2 3 into their respective registers. When OP code 001 is chosen it will output the value from input k into the respective register. When 010 is chosen it will assign the data in to its new (or old) respective register. 011 will take the information from register 0 and return it to its chosen register. The final codes simply do as they’re stated which are simple addition, subtraction, multiplication, and power of two.

Finite State Machine:

Conclusion:  
It was a little difficult to figure out where to start, but I started with the register file and even though it’s probably the more complicated part of the project, I feel as if it’s the easiest to understand. The ALU was confusing at first but I rate that a little more difficult to understand simply because I’m still not great at Verilog coding; but as soon as I got the general concept and a few hints from TA’s, it was probably the thing that came together the fastest.



**Test Plan:**  
*OP 000:*  
*Test 1:*  
Switch OP to 000  
Switch K to 00  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0

*OP 000:  
Test 2:*  
Switch OP to 000  
Switch K to 01  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0

*OP 001:*  
*Test 1:*  
Switch OP to 001  
Switch K to 01  
Activate enable switch  
Outputs: Register 0 = 1

*OP 001:  
Test 2:*  
Switch OP to 001  
Switch K to 11  
Activate enable switch  
Outputs: Register 0 = 3

*OP 010:*  
*Test 1:*  
Switch OP to 000  
Switch K to 00  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0  
Switch OP to 010  
Switch K to 10  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 2

*OP 010:  
Test 2:*  
Switch OP to 000  
Switch K to 00  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 3

*OP 011:*  
*Test 1:*  
Switch OP to 000  
Switch K to 00  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 0, Register 1 = 1, Register 0 = 0

*OP 011:  
Test 2:*  
Switch OP to 000  
Switch K to 00  
Activate enable switch  
Outputs: Register 3 = 3, Register 2 = 2, Register 1 = 1, Register 0 = 0  
Switch OP to 011  
Switch K to 11  
Activate enable switch  
Outputs: Register 3 = 0, Register 2 = 2, Register 1 = 1, Register 0 = 0

*OP 100:*  
*Test 1:*  
Switch OP to 001  
Switch K to 11  
Activate enable switch  
Outputs: Register 0 = 3  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 3  
Switch OP to 100  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 6

*OP 100:  
Test 2:*  
Switch OP to 001  
Switch K to 10  
Activate enable switch  
Outputs: Register 0 = 2  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 2  
Switch OP to 100  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 4

*OP 101:*  
*Test 1:*  
Switch OP to 001  
Switch K to 11  
Activate enable switch  
Outputs: Register 0 = 3  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 3  
Switch OP to 101  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 0

*OP 101:  
Test 2:*  
Switch OP to 001  
Switch K to 10  
Activate enable switch  
Outputs: Register 0 = 2  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 2  
Switch OP to 101  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 0

*OP 110:*  
*Test 1:*  
Switch OP to 001  
Switch K to 11  
Activate enable switch  
Outputs: Register 0 = 3  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 3  
Switch OP to 110  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 9

*OP 110:  
Test 2:*  
Switch OP to 001  
Switch K to 10  
Activate enable switch  
Outputs: Register 0 = 2  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 2  
Switch OP to 110  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 4

*OP 111:*  
*Test 1:*  
Switch OP to 001  
Switch K to 11  
Activate enable switch  
Outputs: Register 0 = 3  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 3  
Switch OP to 101  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 3, Register 0 = 27

*OP 111:  
Test 2:*  
Switch OP to 001  
Switch K to 10  
Activate enable switch  
Outputs: Register 0 = 2  
Switch OP to 011  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 2  
Switch OP to 100  
Switch K to 10  
Activate enable switch  
Outputs: Register 2 = 2, Register 0 = 4