Floating Point Calculator

Kurtis Rhein, Ghaith Kachi, Brandon Holtz
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: rhein@oakland.edu, gkachi@oakland.edu, baholtz@oakland.edu

The goal of this project is to develop and demonstrate successfully a piece of computer hardware capable of calculating various operations of addition, subtraction, multiplication, and division on floating point numbers of 32-bit size inputter through a keyboard and displaying the result on a 7-segment display.

I.INTRODUCTION

In our project, we aim to make a floating point number calculator on a Digilent Nexys-4 DDR FPGA board. The hardware will take an input of a floating point number, manipulate it based on an input (add, subtract, multiply, divide) and display the output on 7-segment displays in Hexadecimal format. The input mechanism will be via USB using a USB keyboard. The project will utilize the topics of floating point arithmetic and external interfacing.

II.METHODOLOGY

In this project, a USB keyboard and an FPGA board are used to input data and output a result. All the data is in HEX format. The 2 numbers get inputted via a USB keyboard, then the operation desired is also selected using that keyboard. The result is displayed on the seven segment display after the registers A and B are fired through the ALU, with the operation controlled by a decoder that is selected with a code given to each operand. The operand has its own finite state machine made up for it so that it "locks" the operand selection in. It looks for a code according to the F1, F2, F3, and F4 keys on the keyboard and sets a flag high to keep the finite state machine from reverting back to another type.

III.EXPERIMENTAL SETUP

The plan is to have a Top File that does each operation individually along with a Test Bench that tests that operation, then all the modules will be combined in one Top File and a 7-segment display will be connected to show the final results.

The control finite state machine dictates the order that the calculator proceeds. It has four states, referring to the "Get A", "Get Operation", "Get B", and "Calculate" steps that the calculator needed in order to function. The first state, "Get A", set up the according pathways to transfer the inputs from the registers to the 32-bit register designated for the A input once the Enter key was struck. Once this was completed, a "Done" flag was raised, and State 2 was performed.

State 2 is dedicated specifically to the operand finite state machine. The FSM looks for an input and changes its output as described before, which then sends a flag back to the control circuit to continue onward to State 3, "Get B." This was similar to State 1, almost identical actually. Once the value was recorded, State 4 triggers the operand register and sends the 32-bit inputs through the ALU. The display is also changed to show the output.

The output was done on a set of 7-segment displays. As there were 8 of them on the Nexys 4 DDR board being used, it was perfect to display the 32-bit output in hexadecimal. The displays were serialized using the serializer from Lab 4 (SPI Accelerometer Lab) so that each individual hexadecimal digit could be represented when State 4 came. When the output was not being displayed, a signal from the input registers was mirrored and displayed here so that the user could see what values they were entering with the keyboard.

IV.RESULTS

The final output of the project is exactly like it's supposed to be. We're successfully entering data using a keyboard (numbers A, B, and an arithmetic key). Each number is being shown on the 7-segment display on the FPGA board. After both numbers are inputted, we show the result on the 7-segment display upon hitting the Enter key.

V.CONCLUSIONS

The floating point calculator worked as expected. We have successfully combined components such as the adder, subtractor, divider, and multiplayer together to create a floating point calculator. When inputting in various 32-bit size numbers and selecting an operation the result would display on the seven segment display in a hexadecimal format. The implementation plan of having a separate ALU unit for one part of the group and another I/O framework to place that ALU in worked wonderfully, allowing for seamless integration and collaboration. If new ALU units had to be replaced, the simple "A, B, S" I/O design allowed this to be easy to do. The implementation of the PS/2

Keyboard circuit was also a success, however was met with some challenge during design as the wrong “Done” signal was used for cycling through the registers, causing a set of 3 inputs per keystroke to be made. This was resolved by reevaluating the PS/2 Keyboard circuit and finding the correct “Done” signal to use. This project was not met without challenge, however the material learned in this class reinforced the ability of problem solving in VHDL programming, which saw the success of the project pull through.

1- Adder/ Subtractor (Figure 1)
2- Multiplier/ Divider (Figure 2 and Figure 3)
3- Keycode
4- LUT
5- Finite State Machine
6- Hex to Seven Seg Display
7- Registers
8- MUX
9- Leading Zero Detector
10- Barrel Shifter

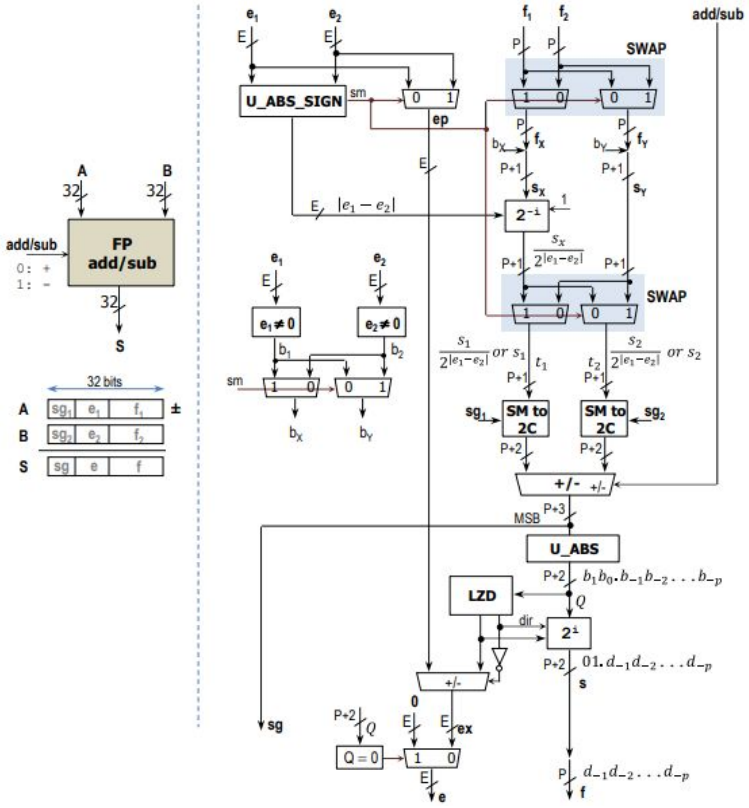
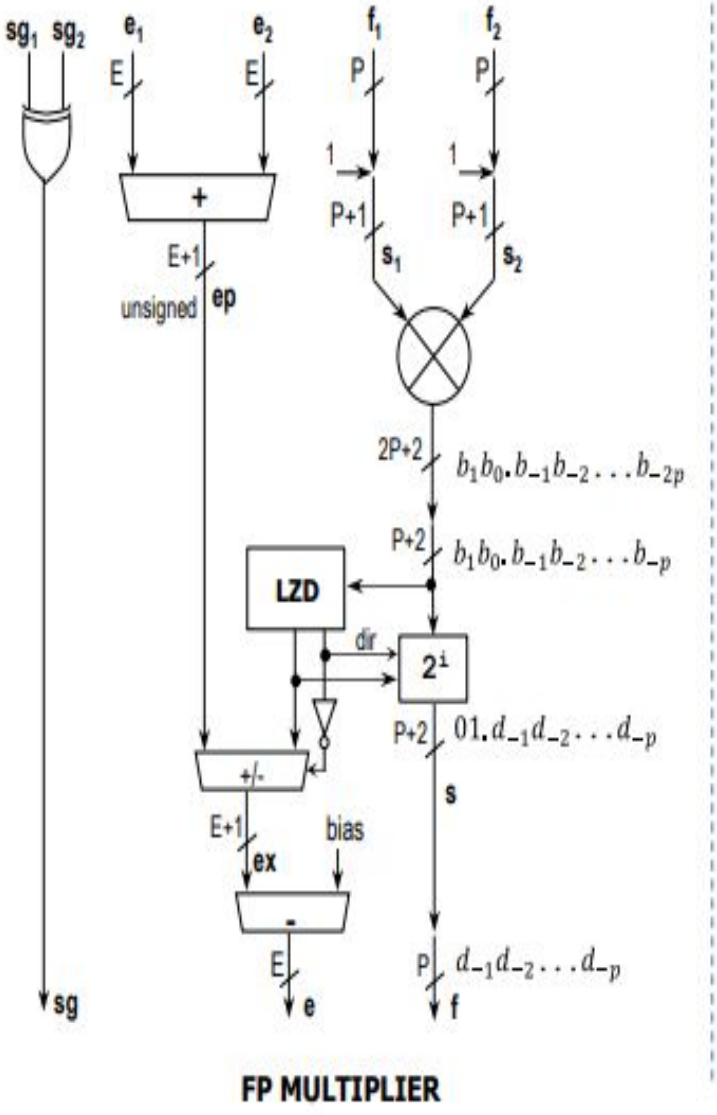


Figure 1: Adder/ Subtractor Block Diagram

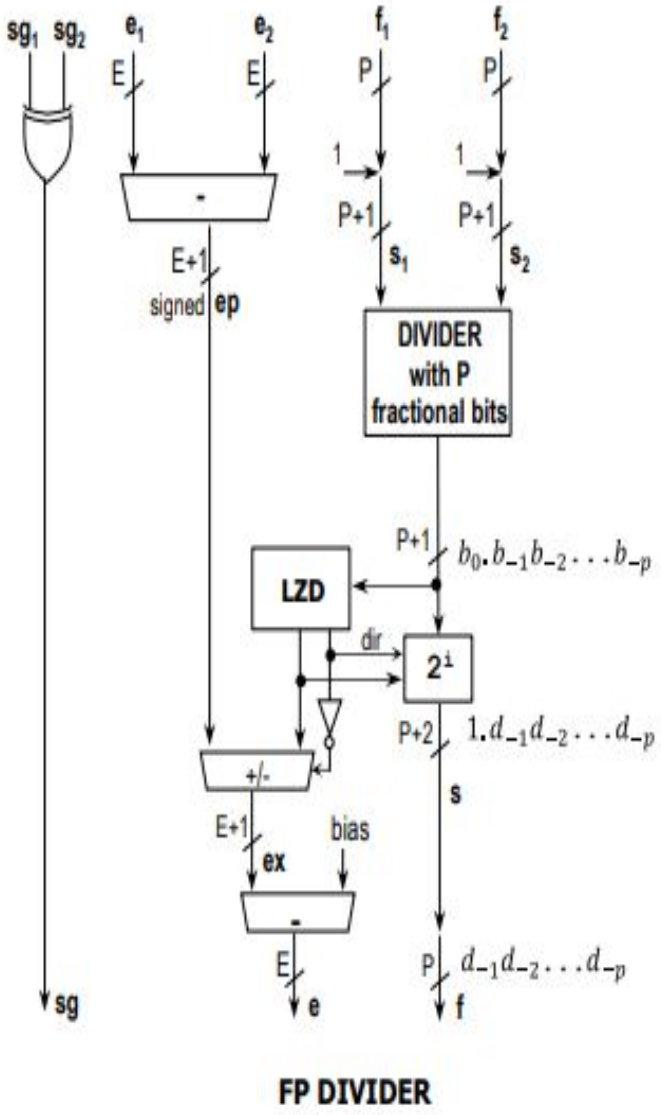
This circuit was created and tested in Lab 3 of ECE 4710 and was reused in this project.

VI.REFERENCES

[1] Daniel Llamocca, Class notes-Unit 2, “Computer Arithmetic”, W2020.
[2] Daniel Llamocca, Class notes-Unit 4, “Special-Purpose Arithmetic Circuits and Techniques”, W2020
[3] Daniel Llamocca, “Laboratory 2”, W2020
[4] Daniel Llamocca, “VHDL Coding for FPGAs”
[5] Daniel Llamocca, Class notes-Unit 3, “External Peripherals: Interfacing”



FP MULTIPLIER



FP DIVIDER

Figure 2: Multiplier Block Diagram **Figure 3: Divider Block Diagram**

These two circuits had to be constructed with information in Dr. Llamocca's class notes. The Divider and Multiplier were essential to get correct due to the useless nature of an inaccurate calculator.

Figure 4. Addition Simulation

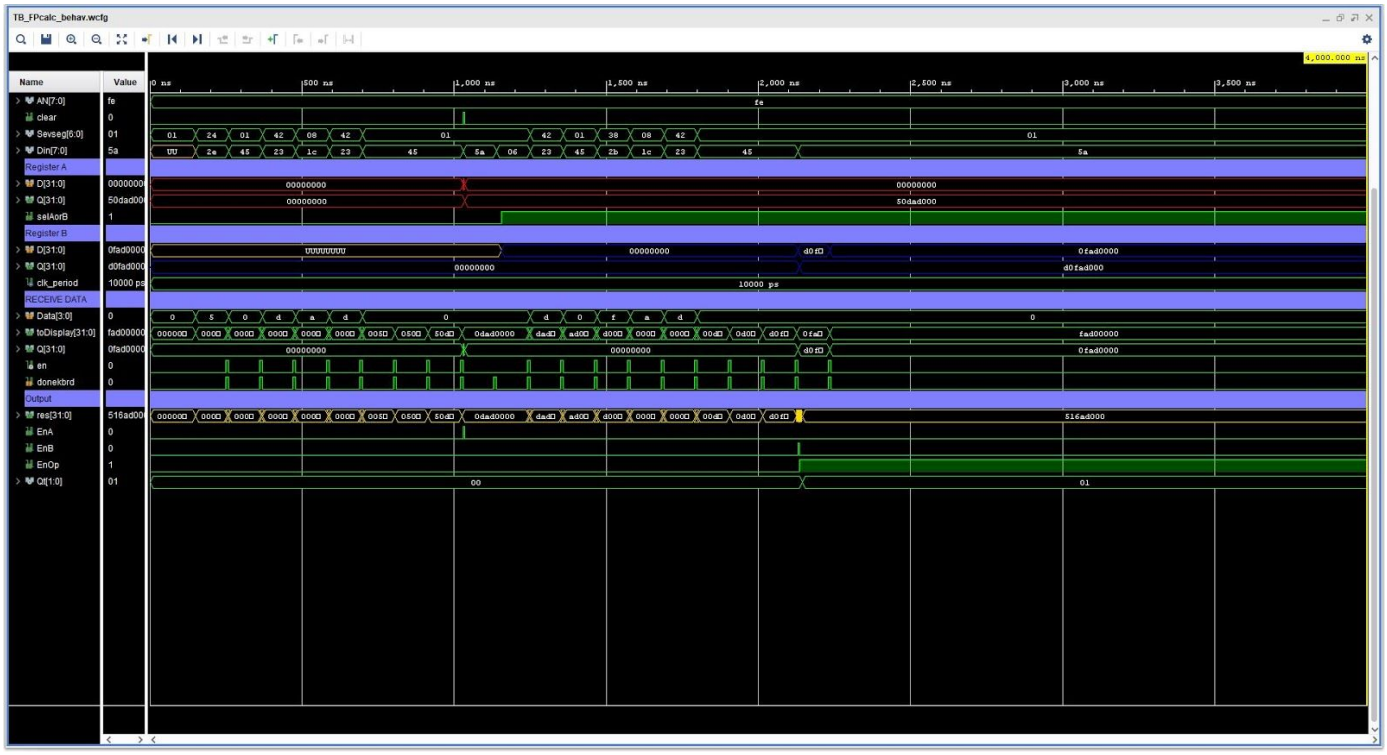


Figure 5. Subtraction Simulation

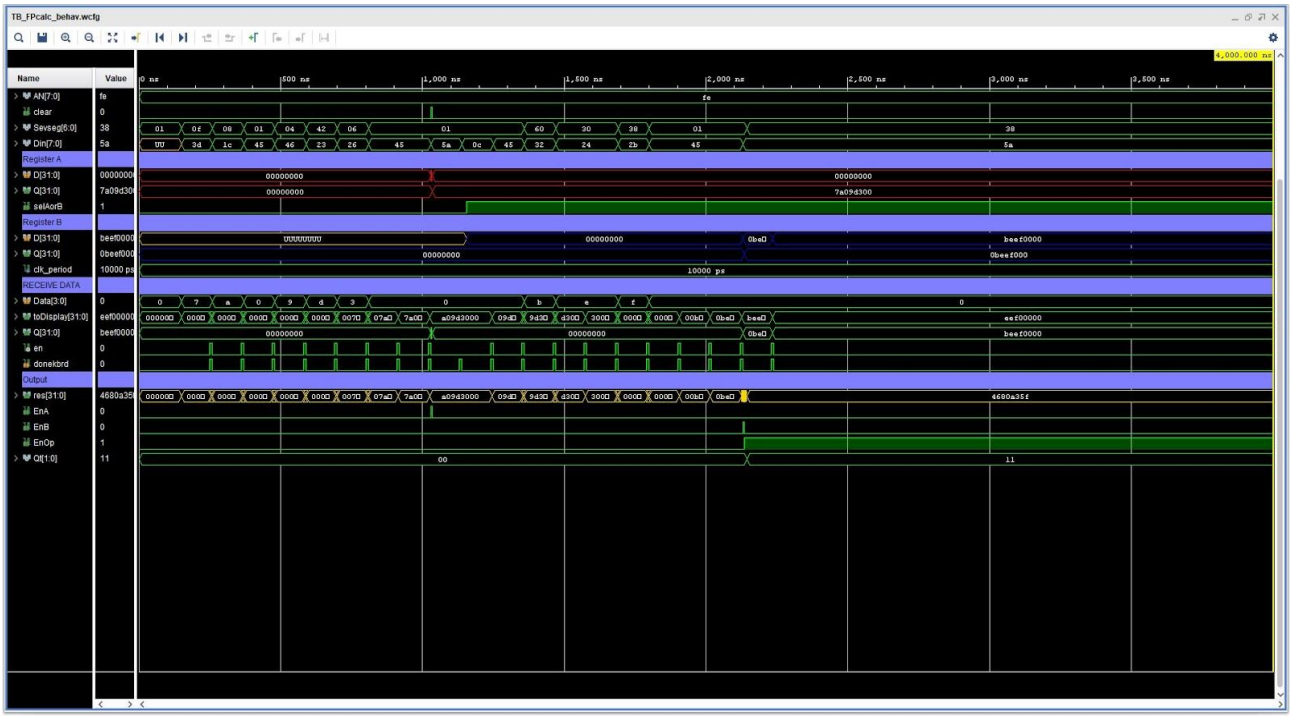


Figure 6: Multiplier Simulation

Figure 7: Division Simulation

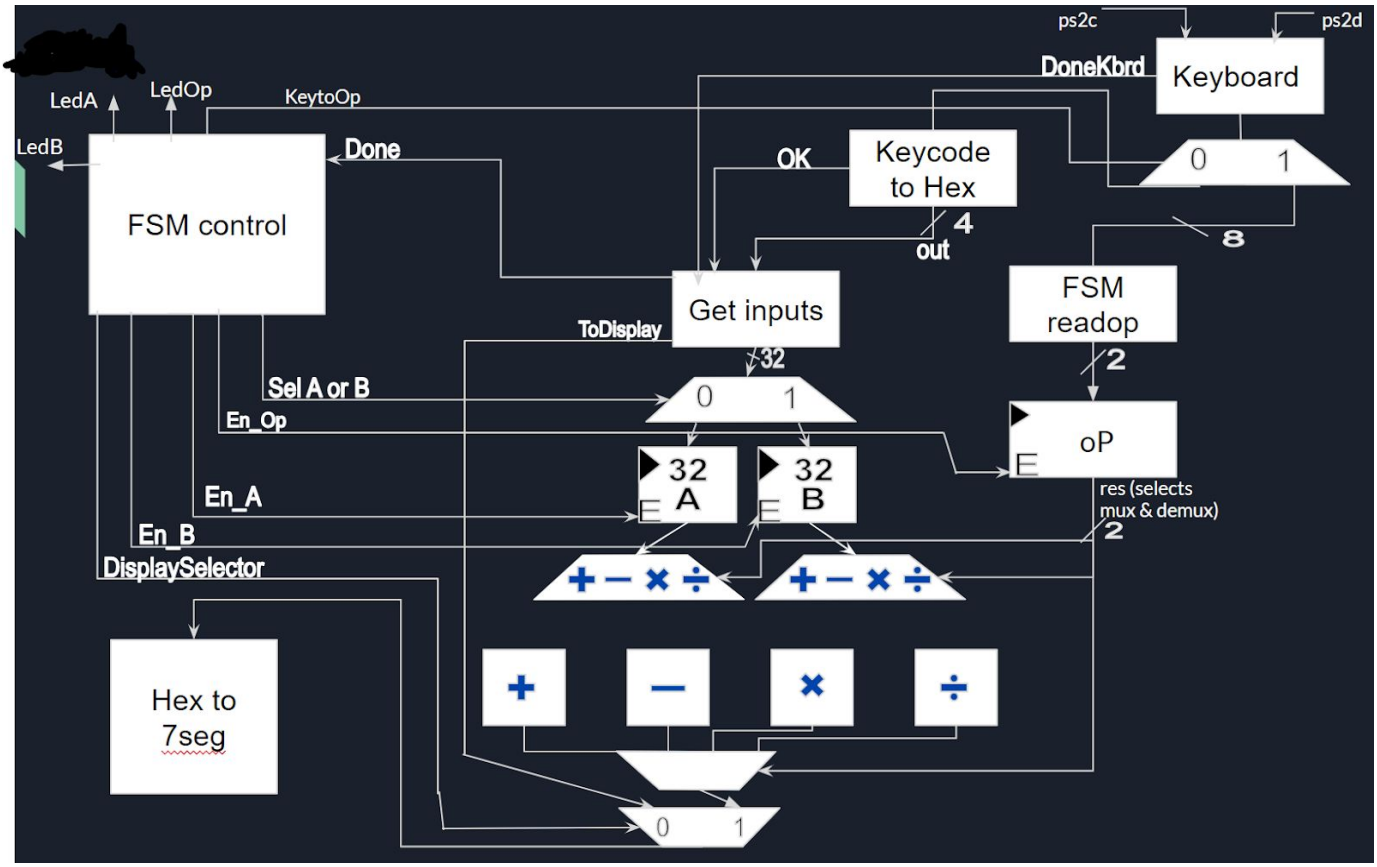


Figure 8: Top Level

The overall design of the circuit, with data being input at the top right by the Keyboard module and output in the Hex to 7Seg module to output to the FPGA. Important design choices to note are the MUX and DEMUX modules for the ALU, which are selected by the 2-bit code that the operation register puts out, allowing the user to choose an operation.

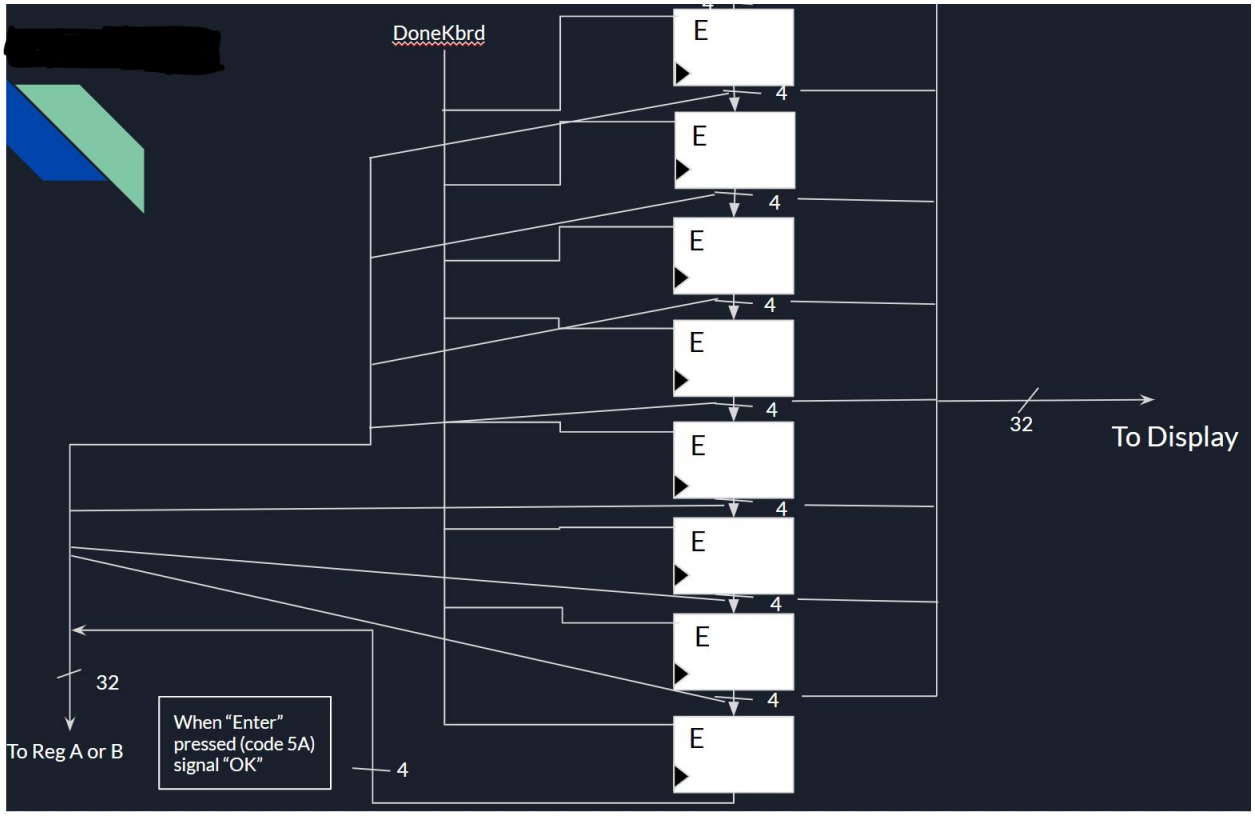


Figure 9: Get Data

This circuit is a series of registers that get enabled and shift down when a new input gets put in from the Keyboard (hence "DoneKbrd"). This helped to input one value at a time and also is the sole force behind the active 7-segment display array that showed what was typed in.

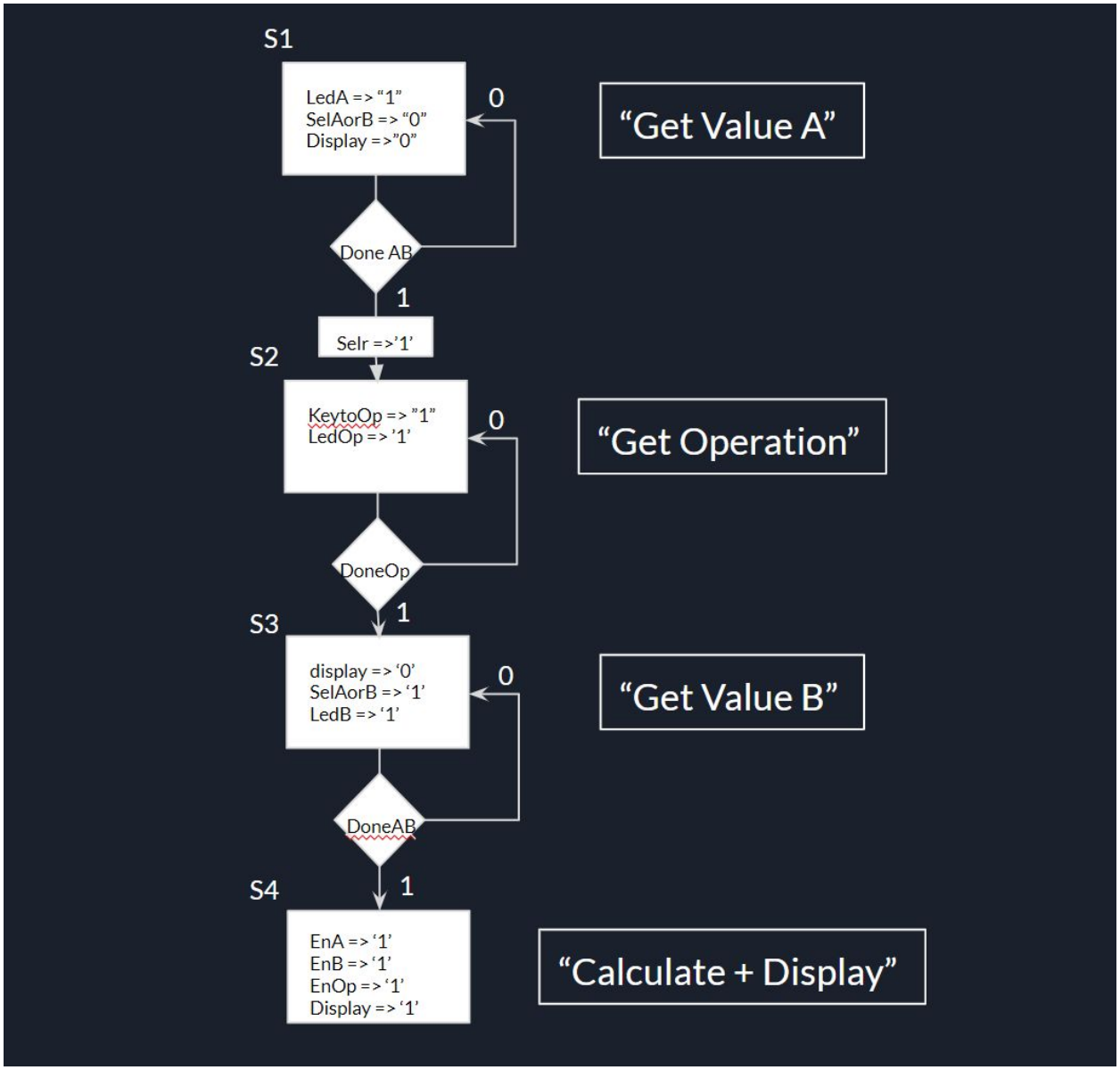


Figure 9: FSM Control

The Control FSM is the driving force of the entire calculator. It defines what goals are being performed and the credentials to move onto the next requirement. There was a slight adjustment while debugging where EnA and EnB were moved to states S1 and S2 respectively to secure the input data, preserving it for use in the ALU.

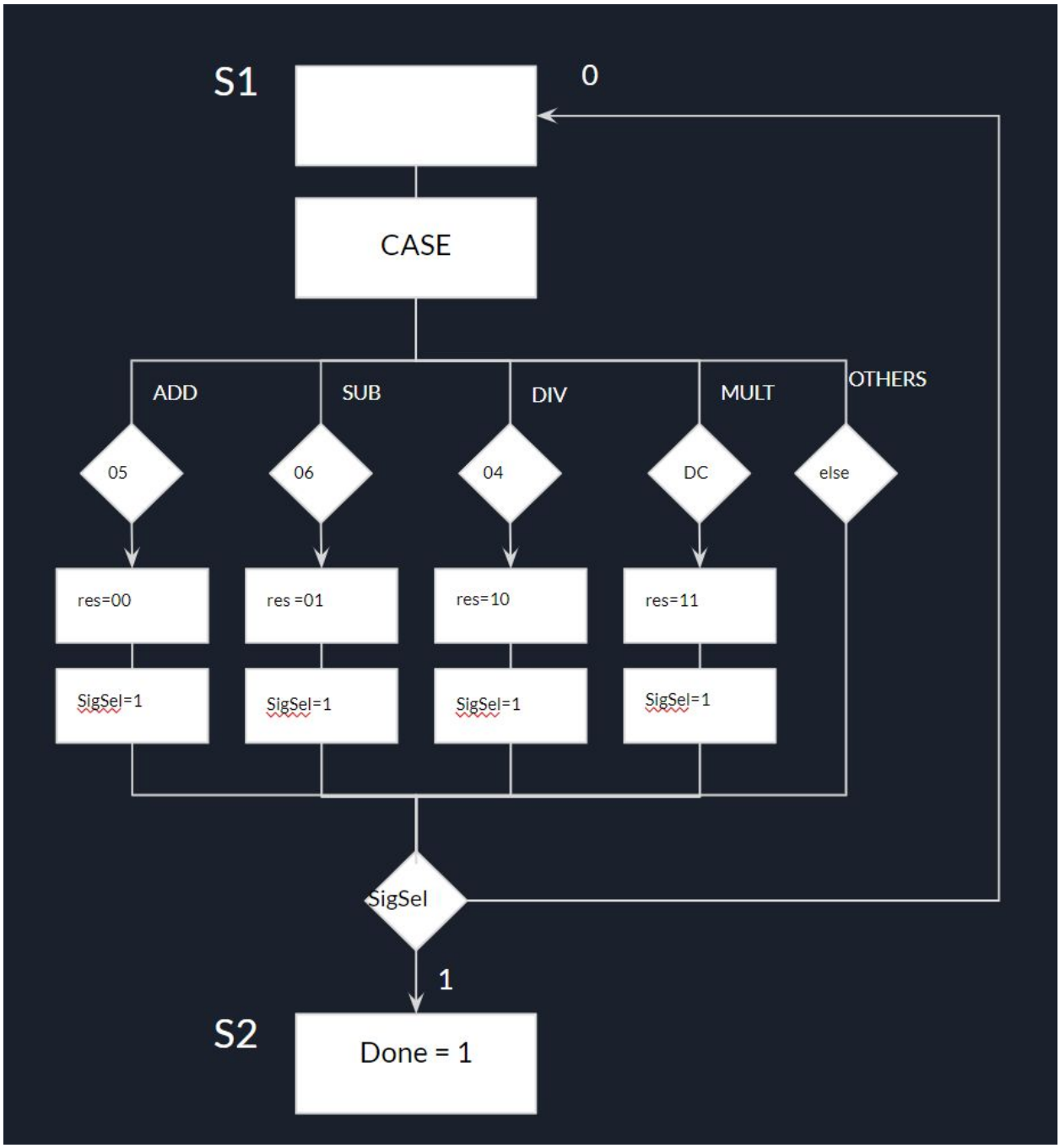


Figure 10: FSM readOp

This circuit was designed as a case statement that was looking for four distinct inputs from the keyboard itself to assign operands to the rest of the calculator circuit.