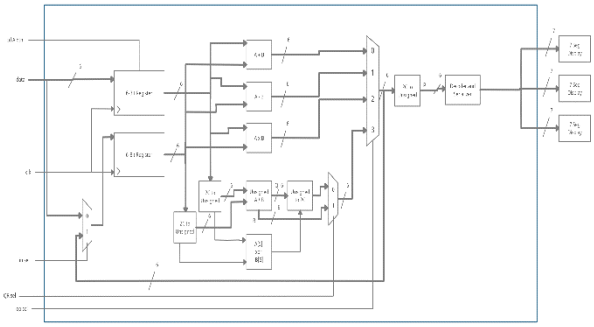
Signed Calculator with Seven Segment Display

Names: Michael Dunnigan, Andrew Waite, Shahul Hameed Arif Batcha

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: michaeldunnigan@oakland.edu, agwaite@oakland.edu, sarifbatcha@oakland.edu

Abstract- *Our goal is to create a 6-bit signed calculator that has the ability to add, subtract, multiply and divide signed numbers. We also want to display the output of these calculations on the built in seven segment display on the FPGA. On top of that, registers will also be used to cut down on the number of inputs needed to use the calculator.*



I. Introduction

For our project, we will be using Vivado to create a signed 6-bit calculator capable of performing the four fundamental calculations (adding, subtracting, multiplying, and dividing). The results of the calculations will then be displayed in decimal form on a seven-segment display.

Our main reason for choosing a calculator was to further increase our understanding of binary data manipulations. This project will also expand our knowledge of not only seven segment displays but registers as well. We plan on using multiple registers to save our input data and multiple seven segment displays to show our solution. Our calculator will cover the three main tasks of any digital system - receiving/saving data, manipulating data, and displaying the results. Many of the modules used in this calculator such as the register and multiplier have been taught to us in class. However, the calculator will also utilize some logic circuits that were not covered in class such as the multiplier and a special 5-bit hexadecimal decoder. Our end goal is to utilize the base logic circuits taught to us in class to create a robust yet easy to use calculator.

II. Methodology

The calculator will be composed of many logic circuits split into three different segments. The first segment is the input/storage section. This section will include two 6-bit registers to store the two input values. Also, we plan to feed the outcome of any calculation back into the second register for sequential calculations.

The second segment of the calculator will be the data manipulation section. This section will contain all of the circuits necessary for performing the fundamental calculations. This segment will also require a component that can change between signed and unsigned binary, as the divider being used is not capable of signed division without them.

The final segment of the calculator is the data display section. This section will contain only a single yet rather complex circuit capable of receiving a signed 6-bit signal and converting it into a 2-digit hexadecimal signal with and extra signal to indicate positive or negative. Each of the three group members worked consecutively on all three segments to increase our general understanding, however, for the report, each student chose a specific segment of the calculator to explain in detail.

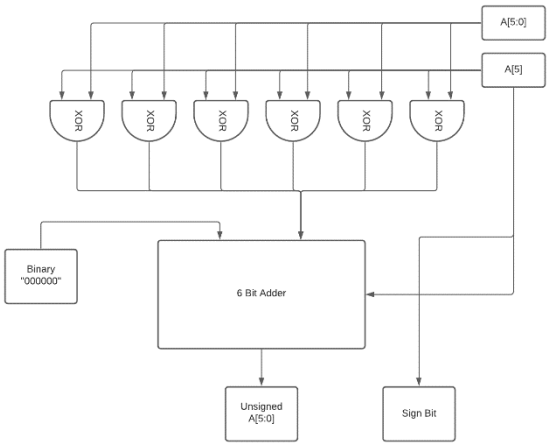
A. Input and Storage Segment

The main goal of this segment is to allow the user to enter data quickly and easily to the device while still using the fewest inputs to avoid confusion from the operator. For that reason, the decision was made to connect both registers to the same input bus. Each register also has a button on the FPGA connected its ‘enable’. This reduces the number of inputs needed from 12 to 8. Along side of this, we also plan on feeding the output of the calculation back into one of the registers to make sequential calculations easier. This will require a 2 to 1 multiplexer to connect to one of the

registers. The multiplexer will allow the user to choose whether they would like to load the previous calculation(sequential calculation) or to load a new data stream.

B. Data Calculation Segment

The biggest problem that had to be solved when designing the data calculation segment was which signing convention to use for the data. Both sign and magnitude as well as 2's complement were considered, however the latter was ultimately chosen. We had originally planned on using sign and magnitude as it would make switching from signed binary to unsigned binary trivial, however, we soon realized that using sign and magnitude would require a complete redesign of our adder and subtractor components, so 2's complement was chosen instead. This left us with the task of creating two components that switched from unsigned binary to 2's complement and vice versa for use in the unsigned divider. This proved to be an easier task than expected and after creating the 2C to unsigned component using XOR gates and a 6-bit adder, we were able to modify the code to create a component that could change back to 2C.



C. Data Output and Display Segment

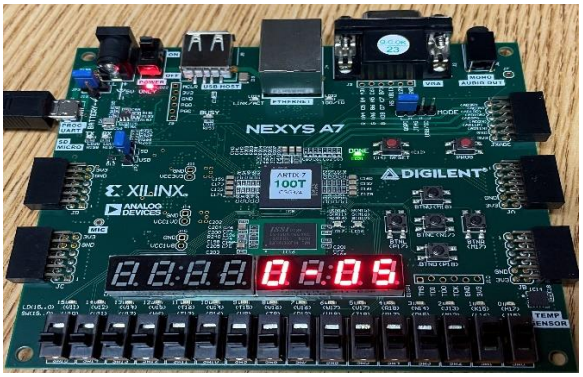
Our group had expected the data display segment of the project to be the easiest part, but it proved to be a lot trickier that initially thought. A lot of thought had to be put into designing components that converted a six bit 2's complement number into a decimal answer. This problem was solved by first converting the 2's complement into an unsigned binary number with a sign bit concatenated to it (essentially sign and magnitude). Then this number was sent into a custom decoder that converted every possible output answer into a twelve-bit BCD number. In this BCD, the four most significant bits were used as the sign bit by decoding a negative sign as the binary representation for 10 (1010). Finally, the 12-bit BCD was segmented into 3 four bit groups and sent to a general seven segment serializer.

I. Experimental Setup

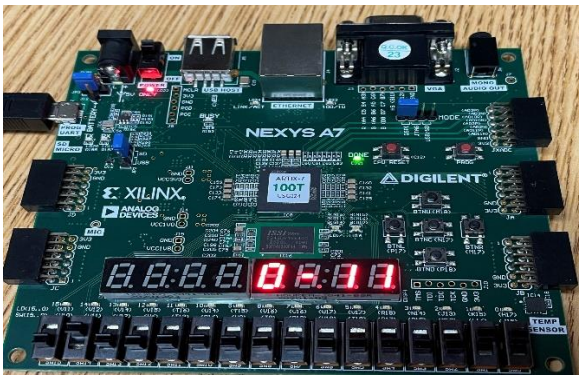
Once all of the project files were completed, we began testing our project by using the simulation function on Vivado. Immediately, we noticed that quite a few things were not functioning as expected. There were a few minor problems that were fixed immediately such as typos in the BCD decoder and a few port mapping typos in the top file. After these issues were fixed, however, we noticed that there were more errors that could not be fixed as easily. One such problems was with the multiplier. Despite not showing any problems when ran in a testbench individually, no outputs were given when the multiplier was compiled with the rest of the files. Our solution was to replace the file with the simple code "F <= A*B" and let Vivado generate a circuit to describe it. Another critical issue we encountered when trying to testbench our project was with the seven-segment display serializer. The state-machine used in the serializer would not recognize the clock signal, resulting in no state change and only the first seven-segment display to be enabled. While we never knew what truly caused this problem, we were able to fix it by creating a new project in Vivado and loading the project files into it.

II. Results

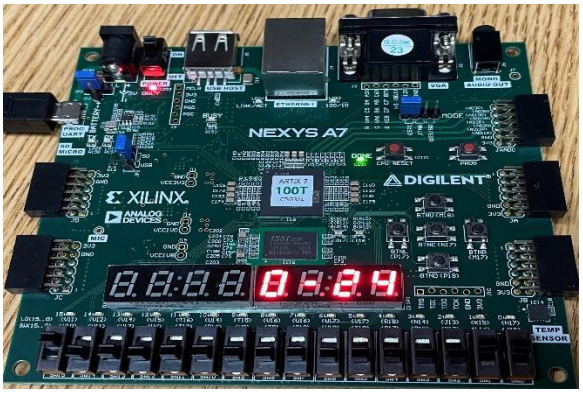
The timing diagram above shows all four operations of the calculator computing both negative and positive numbers. Originally, certain results like the multiplier were not outputting any data at all, but after some heavy modification to the code, we were able to get the timing diagram to display all the expected outputs. Shown below is the calculator performing all operations including the remainder with A= -8 and B= 3.



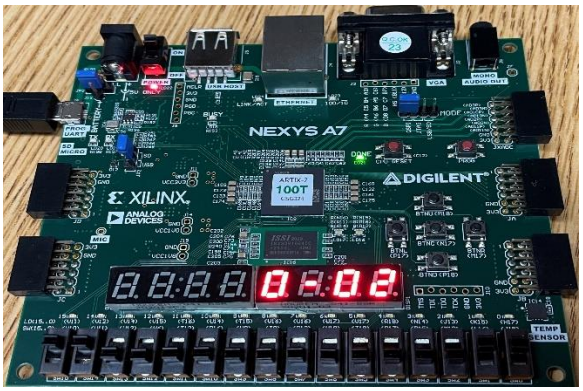
-Addition of A=-8 and B=3



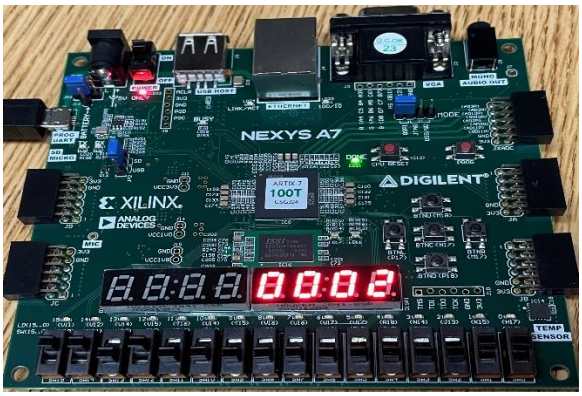
-Subtraction of A=-8 and B=3



-Multiplication of A=-8 and B=3



-Division of A=-8 and B=3



-Remainder of the division of A=-8 and B=3

CONCLUSIONS

After completing the project, our group was able to make a few important conclusions. One lesson we learned early on was the importance of designing a detailed and easy to understand block diagram. Coordinating the project would have been very difficult without a visual aid like the block diagram to decide what problem each group member should work on. Similar to the block diagram, our group also recognized how useful the use of the error codes that Vivado would give after an implementation would fail. The error codes allowed us to pinpoint exactly where and how the project was failing. Overall, our group was very satisfied with the final outcome of our project. It taught us a lot about how data is managed and displayed in digital devices. If there was one thing our group could have improved on, it would have been changing the calculator to compute number higher than -32 to 31. We feel, however, that his project was a great demonstration of our knowledge with Vivado/VHDL and we look forward to expanding our knowledge even further in the future.

REFERENCES

Llamoca, D. (n.d.). *RECR Class Website*. Fall 2020 - ECE2700: Digital Logic Design. Retrieved December 11, 2021, from http://www.secs.oakland.edu/~llamocca/Fall2020_ece2700.html.