Number Cruncher

(List of Authors (Mohamad Abbas, Paul Shammas, Feras Zari)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
E-mails: mabbas@oakland.edu, pshammas@oakland.edu, feraszari@oakland.edu

***Abstract*— The purpose of this project was to create a simple calculator using the Nexys 4-DDR board along with the skills we leaned while programming and implementing in Vivado throughout the semester. This calculator will allow basic 4-bit math operations to be computed such as addition, subtraction, multiplication, and division. We found that programming a 4-bit rather simply when it comes to addition, subtraction and multiplication but becomes more complicated when implementing division.**

I. INTRODUCTION

For this project we designed a basic number cruncher that can compute numbers no bigger than 4-bits. The number cruncher first takes two 4-bit positive number and through are circuit design either adds, subtracts, multiplies or divides and displays the answer on a seven-segment display.

We decided to build a number cruncher that does simple arithmetic because basic math is the foundation of all engineering. We used almost all concepts learnt in class to implement our design including all digital logic gates, building components such as multipliers and registers, structural VHDL and finite state machines.

The application of our project is simply, to help people do basic arithmetic.

II. METHODOLOGY

A. Design

The top-level design of our project included 11 different components. We have four major components that perform the arithmetic operations. We also built a multiplexer which selects the arithmetic function requested and we used a seven-segment display which displays the answer and the inputs. To better understand the design a block diagram of our design is shown in figure 1

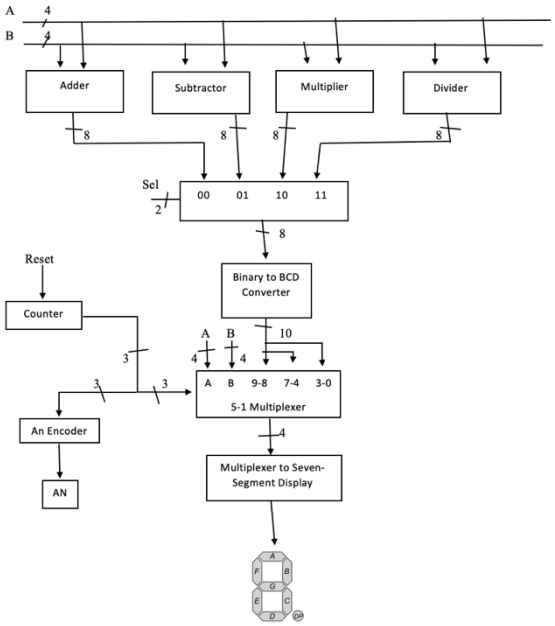

Figure 1: Number Cruncher Block Diagram

B. Operations

The addition component consists of 4 full adders with the input of it being A and B from the switches. Coming out of the addition portion is 4 bits with a cout of 1 bit which was concatenated using the 3 bits from the left. The subtraction component includes 9 full adders and 4 not gates. The result of the subtraction will always be positive as the number cruncher only accepts unsigned binary numbers. The multiplication component had a very simple design consisting of just 12 full adders. [1]. The division component is a large one. The division component included a 4-bit left register, a register that accepts input from the switches of the FPGA, a left shift registers with SCLR, finite state machine, adder, and a counter. The left register shifts and accepts 4 bits and also outputs 4 bits. Another register is included that is related to the b input and takes in only 4 bits. Next the Finite State Machine (FSM) was construed with 3 different states. Using items learned in lab 6, a counter which counts from a 0-3 output was designed. Instead of having a hex to 7 segment component which was mandatory in lab 6, our designs output from the left shift register went straight to the 4 to 1 multiplexer and the

reminder goes to the LEDs on the board. The addition, subtraction, multiplication and division components were designed separately and all feed to a 4 to 1 mux. The 4 to 1 mux consists of a 2 bit select line which corresponds to SW11-S10 on the board. The inputs are two 4-bit numbers and the output is 8-bits. A binary to BCD converter accepts bits from the 4 to 1 mux, which takes 8 bits in binary and outputs a 10-bit BCD number using 2 loops. From the BCD converter, a 5to1 mux was implemented. The reason behind the 5 to 1 mux was to take two select between 5 different inputs. The inputs of the 5 to 1 mux are as follows: A and B inputs taken from the switches of the FPGA, then we split the 10-bit output of the binary to BCD converter and broke it into three separate 4-bit numbers. Bits 10-9 were concatenated "00" and represents the 100s place, bits 7-4 represent the 10s place, and bits 3-0 represent the 1s place of a 3-digit answer to the calculation. To control the select lines, a counter was used. The counter outputs 3 bits. By designing it like this, we were able to display one digit at a time on the seven-segment display, but fast enough so it appears as if the inputs and the 3-digit result are all on. The output of the counter feeds into the 5to1 mux and an anode encoder. A mux to seven segment decoder takes the 4 bits from the multiplexer and converts them to seven bits which get displayed on the seven-segment display.

III. Experimental Setup

For the testbench, five combinations were implemented for each operation so a total of 20 combinations were built and simulated. This was tested to determine that the right values were being displayed on the simulation

IV. Results

The number cruncher provided many different challenges for design and implementation. Each component was able to be construed using the knowledge of previous labs and a little bit of figuring out and repetition. In the figures below, there are a few different highlighted basic math operations that were performed.

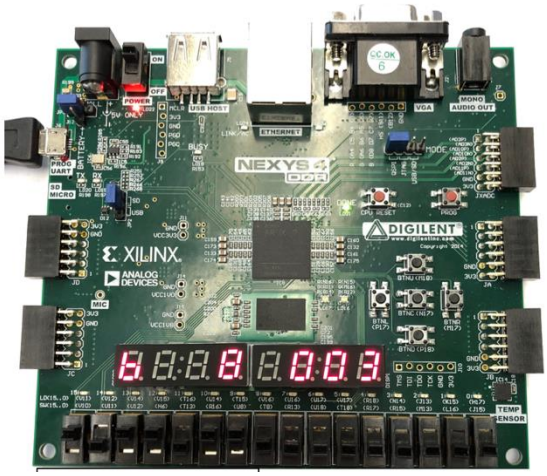

Figure 2: Addition 15(F)+12(C)=27

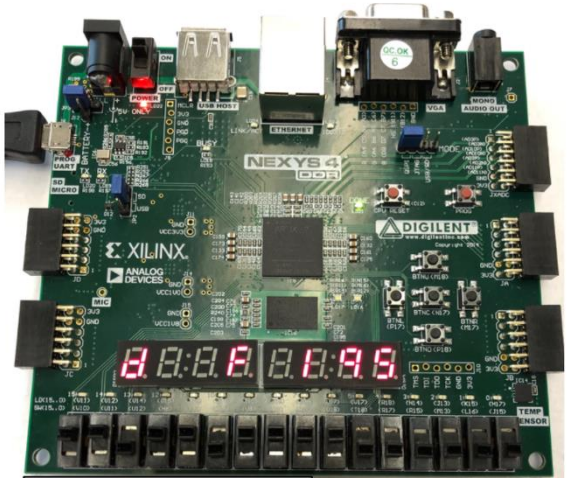

Figure 3: Subtraction: 11(B)- 8= 3

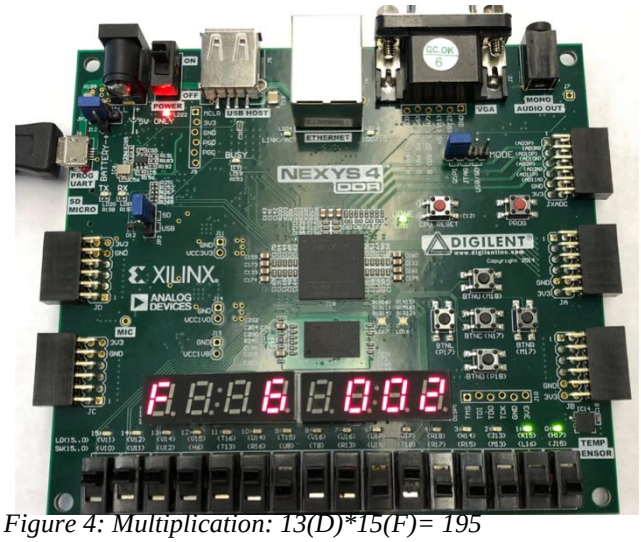

*Figure 4: Multiplication: 13(D)*15(F)= 195*

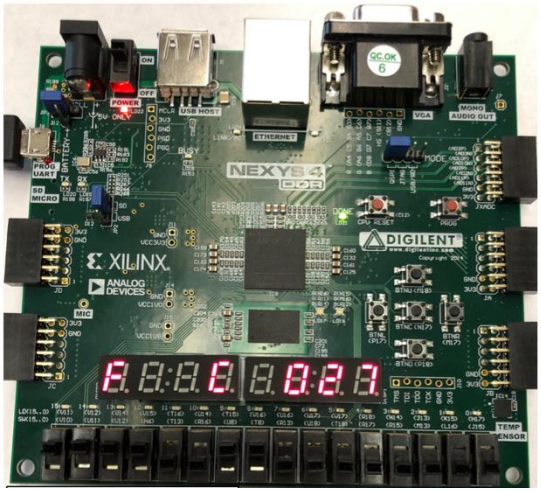

Figure 5: Division (SW5 ENABLED): 15(F)/6= 2 remainder 3(LEDS)

CONCLUSION

The Nexys number cruncher consisted of many fragments such as the multiplier, divider, adder and subtractor. Each fragment consists of several sources that give each fragment its own respected task. All of these are summed up and put together into the top file and programmed into a Nexys 4 DDR board. The number cruncher was coding intensive, the adder and subtractor were simple. However, the multiplier and divider required a lot of work and research the divider in particular stood out to be the most challenging portion of our project due to the several sources that were required to design it. Lab 6 was referenced many times to code the divider and lab 4 was referenced for the majority of the project. A few improvements that could have been made to the project was having the number cruncher registering the previous number that was calculated so a new form of arithmetic can be done to it. Throughout designing the project, the group strongly considered this and were seeking ways to implement this design into the number cruncher. The goal was to make the Nexys number cruncher as close to a standard everyday calculator as possible. After many tests however, the group decided to move on from that design source.

REFERENCES

[1] Llamocca, Daniel. "VHDL Coding for FPGA's." Reconfigureable Computing Research Laboratory. N.p., n.d. Web. 4 APR 2019.

[2] "Nexys 4 DDR." *Nexys 4 DDR [Reference.Digilentinc]*, reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/start.

[3] LBEBooks, director. VHDL Example 19: 8-Bit Binary-to-BCD Converter-for Loops. Youtube, 24 Oct. 2012, [3] https://www.youtube.com/watch?v=VKKGyOc4zRA.

[4] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 1

[5] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 2

[6] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 3

[7] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 4

[8] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 5

[9] DIGITAL LOGIC DESIGN VHDL Coding for FPGAs Unit 6

[10] Lab 4

[11] Lab 6

[12] Lab 3