# Signed 4-bit Calculator

List of Authors (Rachel Dingman, Kevin Huffman, Alyssa Musienko)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: racheldingman@oakland.edu, khuffman@oakland.edu, amusienko@oakland.edu

***Abstract*— This project was created to explore the possibility of constructing a four function calculator using digital logic to program a Field Programmable Gate Array (FPGA). The design accepts user inputs to dictate the function being performed and the two separate 4-bit numbers that the calculations are being performed on. The final calculator design also allows for increased usability by displaying the calculation on a Universal Asynchronous Receiver Transmitter (UART) controlled display. Through careful implementation, and multiple design iterations, it was discovered that an FPGA can successfully be used to run addition, subtraction, multiplication, and division programs without error or miscalculation.**

## I. Introduction

This project explores the possibility of recording and processing information from an FPGA in real time and displaying that information on a UART display. The system's inputs include two 5-bit numbers in sign and magnitude, a 4-bit input to dictate what function will be performed, and an input command to run the calculation. The motivation for designing this system is expanding it to encompass more processing and more inputs. By expanding this project, a fully functional calculator that allows the user to input complex equations could be built.

This project is the culmination of many concepts covered in the lectures and labs of ECE 2700. This includes using full adders to create addition and subtraction programs, using a combination of partial and full adders to multiply binary numbers, using a complex algorithm to create a division program, the concept of finite state machines, and using user inputs to dictate when to run specific commands. This program also utilizes concepts not specifically covered in class, including the interfacing of an FPGA Board with a UART display.

## II. Methodology

This project was designed to create a simple signed 4-bit calculator using sign and magnitude inputs. The switches on the FPGA are used to enter the values, with each value having a sign switch as its most significant bit (MSB). This brings the inputs to 5 bits, with the magnitude being 4 bits. Since this is to be a signed calculator, conversion of an input to 2's complement (2C) is necessary when the sign is negative (binary value of 1). This was accomplished using if statements to determine if the 5-bit input is positive or negative. If the input is found to be negative, then the 2C operation is performed on the magnitude. If the input is positive, then it will not change and continue through the program.

Due to the nature of basic sign convention, 2C conversions only need to be performed on inputs going to the addition and subtraction blocks. For multiplication and division, simple exclusive or (XOR) gates were used on the sign bit of the inputs to determine the sign bit of the output. This helped to simplify the system by keeping the math operations in basic unsigned implementations.

The system utilizes a program called PuTTY to use as the UART display [1]. The program creates a serial connection between the board and the computer to display the desired information on a console window. To activate the UART display program, the center button on the button array of the FPGA is used. This acts much like an enter button on an actual calculator.

The Block diagram (Figure 9) and Vivado schematic (Figure 10) are both shown at the end of this report.

### A. Addition and Subtraction

To perform addition and subtraction, the most straightforward implementation is to use a Full Adder block. By linking together multiple full adder blocks, a multiple bit adder system can be created. This concept is shown below in Figure 1.

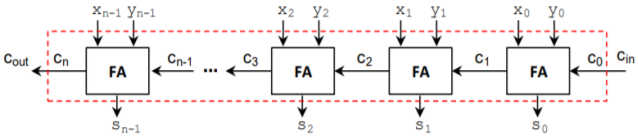


**Figure 1:** Diagram for creating n-bit adders

For this application, since we have an input of 5-bits, a 6-bit adder is used to prevent any overflow errors. To account for this system, the 5-bit input is sign-extended to 6-bits. While the concept shown above is for addition, it can be slightly altered to become a subtractor by negating one of the inputs before it enters the Full Adder block. Since this basic system is an unsigned adder/subtractor, the inputs were converted into 2's complements in order to create signed addition and subtraction. Since this will create an output that is in 2C, the output of this system must then be converted back to sign and magnitude using the previously mentioned 2C process once again..

## B. *Multiplication*

The multiplication function utilizes Dr. Llamocca's array multiplication program, which takes two unsigned 4-bit inputs and provides the 7-bit output [2]. This program utilizes Full Adder blocks and logic gates to successfully multiply the two inputs. The diagram for this concept is shown below in Figure 2.

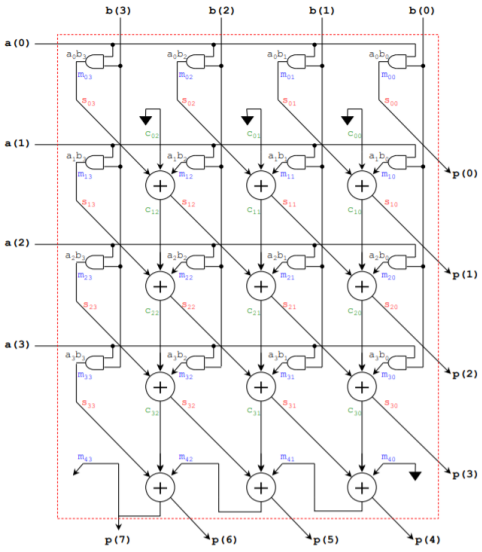


**Figure 2:** Diagram of 4-bit multiplication

This program was edited to account for the sign of the inputs, by taking the XOR of the two sign bits and assigning it to the sign bit of the output, creating an 8-bit output in sign and magnitude.

## C. *Division*

The division function was created by modifying Dr. Llamocca's Lab 6 from the Winter 2020 - EGR 2700: Digital Logic Design course in which an unsigned divider was created [2]. The inputs were adjusted so that they were four bits each and the counter was adjusted to accommodate this as well. There was also no longer a need for the hex to 7-segments decoder and it was left out. The MSBs of the inputs are put through an XOR gate to determine the sign of the output, just like for the multiplication. The division function then outputs a 5-bit result in sign and magnitude. If the division has a remainder, it is ignored.

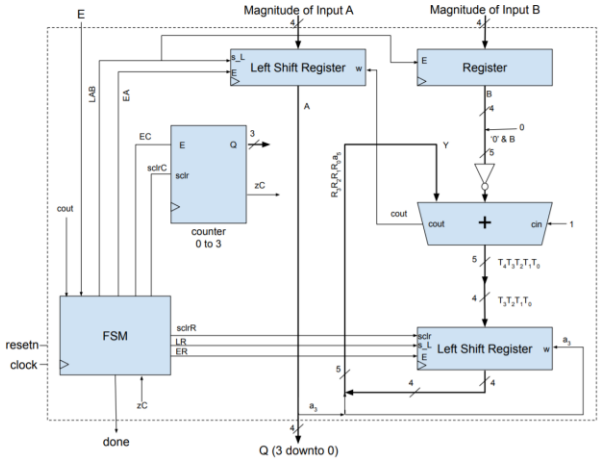


**Figure 3:** Block diagram of division function

## D. *UART Display*

The UART display was created using a program adapted from Dr. Llamocca's Digital Library of FPGA programs [2]. When the original program was run it only sent 8 bits to the UART display which allowed one ASCII character to be printed. This program utilized switch 7 down to switch 0 on the FPGA to control the 8 bits being sent to the display.

In order to adapt this program for the purpose of this project, modifications were necessary to allow for the printing of 9 ASCII characters every time the program was run. To allow for this, modifications had to be made to force the program to send 72 readable bits to the UART display or 8 bits for each ASCII character that needed to be printed. Originally this was attempted by creating a top file and porting Dr. Llamocca's UART program 9 times as shown in Figure 4.

```
Minus100: uart_tx_symbols port map (clock => clock, E => E, resetn => resetn, Sym => Minus10); --minus symbol of first input
Num1  : uart_tx port map (clock => clock, E => E, resetn => resetn, Num => A, Q => Qt); --first input
Function_Sym: uart_tx_symbols port map (clock => clock, E => E, resetn => resetn, Sym => Function1); --function symbol (add/divide/mult/subtract)
Minus200: uart_tx_symbols port map (clock => clock, E => E, resetn => resetn, Sym => Minus11); --minus symbol of second input
Num2  : uart_tx port map (clock => clock, E => E, resetn => resetn, Num => B, Q => Qt1); --second input
Equal_Symbol: uart_tx_symbols port map (clock => clock, E => E, resetn => resetn, Sym => "111"); --equal symbol
Minus300: uart_tx_symbols port map (clock => clock, E => E, resetn => resetn, Sym => Minus11); --minus symbol of answer
AnswerNum1  : uart_tx port map (clock => clock, E => E, resetn => resetn, Num => A, Q => Qt); --first answer symbol
AnswerNum2  : uart_tx port map (clock => clock, E => E, resetn => resetn, Num => A, Q => Qt); --second answer symbol
```

**Figure 4:** Initial program for the UART display

However, it was quickly discovered that this was an impossible system to implement because all 9 functions attempted to send their output data to the UART display at the same time which caused a "multiple drivers error" for the output variable that fed data to the UART. So this program was scrapped in favor of a different design.

In order to allow the asynchronous transfer of 72 bits to the UART display, only one program could be run with a one variable output that changed at a rate of 9600 Hertz in order to accommodate the UART receiving data at a rate of 9600 Baud. This meant that all output bits would have to be strung together into a single 72 bit output in the program. This was initially attempted, but the program only read some of the bits. After carefully scrutinizing the values the UART was displaying, it was discovered that a spacer bits where

necessary between the 8 bits that were being used to generate the ASCII characters shown on the UART display. Without the spacer bits in place a series of horizontal lines would appear in place of the characters due to the UART display not believing that it had a complete data set. This output can be seen in Figure 5 below.

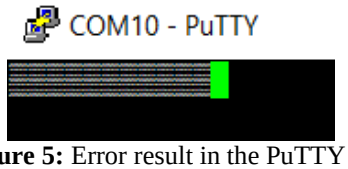


**Figure 5:** Error result in the PuTTY screen

Once this was realized the output variable was expanded to its final size of 84 bits in order to fit 12 spacer bits on top of the 72 data bits output from the program.

In this design the program allowed for some bits to remain the same for all calculations. An example of this is the equal sign which had the same 8 bit sequence no matter the function being performed. This allowed for simplification of the program. For the hexadecimal values displayed on the UART only 16 options were possible, this meant that only 4 bits of data would be needed to represent each number in the program and only 16 possible options had to be accounted for in the program. This rule also applied to the function symbol, there were only 4 possible function symbols that could be displayed so the 8 bits that dictate the function symbol can be controlled by user input, shown in Figure 6.

```
  when "00" => Num1 <= "00101011"; --add
  when "01" => Num1 <= "00101101"; --subtract
  when "10" => Num1 <= "00101111"; --divide
  when "11" => Num1 <= "00101010"; --multiply
  when others => Num1 <= "00000000";
end case;
end process;
```

**Figure 6:** Symbol selection based on user input

These simplifications are what allowed the creation of an 84 bit long output based upon only 12 bits of user input.

### *E. Top Files*

The overall project was separated into two projects: the math calculations and the UART display. Beginning with the math calculation portion, the two input values from the FPGA go through registers when the system is enabled. The division function is then enabled so that all the numerical calculations can be performed. Since it is the only synchronous circuit in this section, the "done" signal sent out when the division is finished is used to transition into the next state. These four output values are then sent to another set of registers, and then to a multiplexor (MUX). Depending on the user input, the MUX then lets through the appropriate function output. A state machine was added to control the flow of the values through this system, a diagram of which is shown in Figure 7.

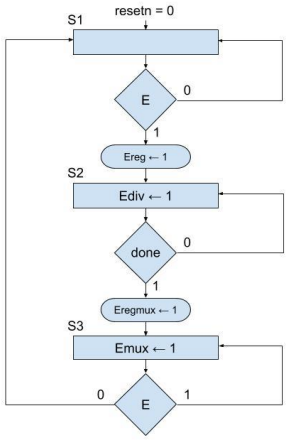


**Figure 7:** State Machine Diagram

Errors had been occurring in the division output, presumably because it needs both an enable and a certain amount of time to complete the function. The state machine allowed for a structured and controlled flow through the system so that these errors were avoided. The output of the MUX is then sent to the UART display file, along with the original 5-bit sign and magnitude inputs and function input to create the math equation. The display file outputs the results to the PuTTY screen when the center button is pressed.

## III. Experimental Setup

For the mathematical calculations portion, a simple test bench was utilized to test the results of each function. Inputs A and B were set to a certain value, and the timing diagram simulation was used to visualize the output of the program. Each math function was tested for each pair of inputs, and multiple different pairs were checked to test the accuracy of the sign convention as well as the numerical output.

For the UART display, a test statement was entered using 'X' to indicate where user input or the system output would be. This proved to be most efficient, because it allowed for testing on an individual scale before it was implemented with the rest of the project.

## IV. Results

The results of this system were how we had originally envisioned it. The user must first enter the values, then select the math operation, then turn on the enable, and finally hit the center button to enable the UART communication to display in the console window. This result is shown below in Figure 8, using inputs -10 * 5, in hexadecimal form.

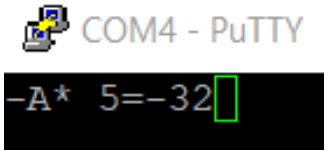


**Figure 8:** Output displayed on the PuTTY screen based on user input

The result matches in both sign convention, and the numerical output. The hexadecimal output equals 50 in decimal.

The only portion of this project that displayed unexpected characteristics was the UART display. After careful inspection of the information being displayed on the UART, it was discovered that some of the bits sent by the FPGA were not being read by the UART. This was circumvented by adding "spacer bits" in between the bits being used to control what ASCII characters appear on the display. It is believed that these spacer bits were necessary because of a mismatch in clock speed between the FPGA board and the receiving port of the UART. While both ports were set to a read/write rate of 9600 Baud, it is possible that a small clock speed difference compounded over multiple clock cycles led to the occasional misreading of data.

## Conclusions

An FPGA board can successfully be used to compile complex programs and dictate which algorithms to run based on user inputs. In the case of this project, algorithms successfully performed addition, multiplication, subtraction, or division based on the user inputs. The FPGA board can also be used to successfully display information on a UART display based upon user inputs. The board in this project displayed the signs, calculation inputs, and the calculation answer.

## References


[1] *Download PuTTY: latest release (0.73)*. [Online]. Available: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html. [Accessed: 14 April 2020].

[2] D. Llamocca, VHDL Coding for FPGAs. [Online]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html [Accessed 14 April 2020].


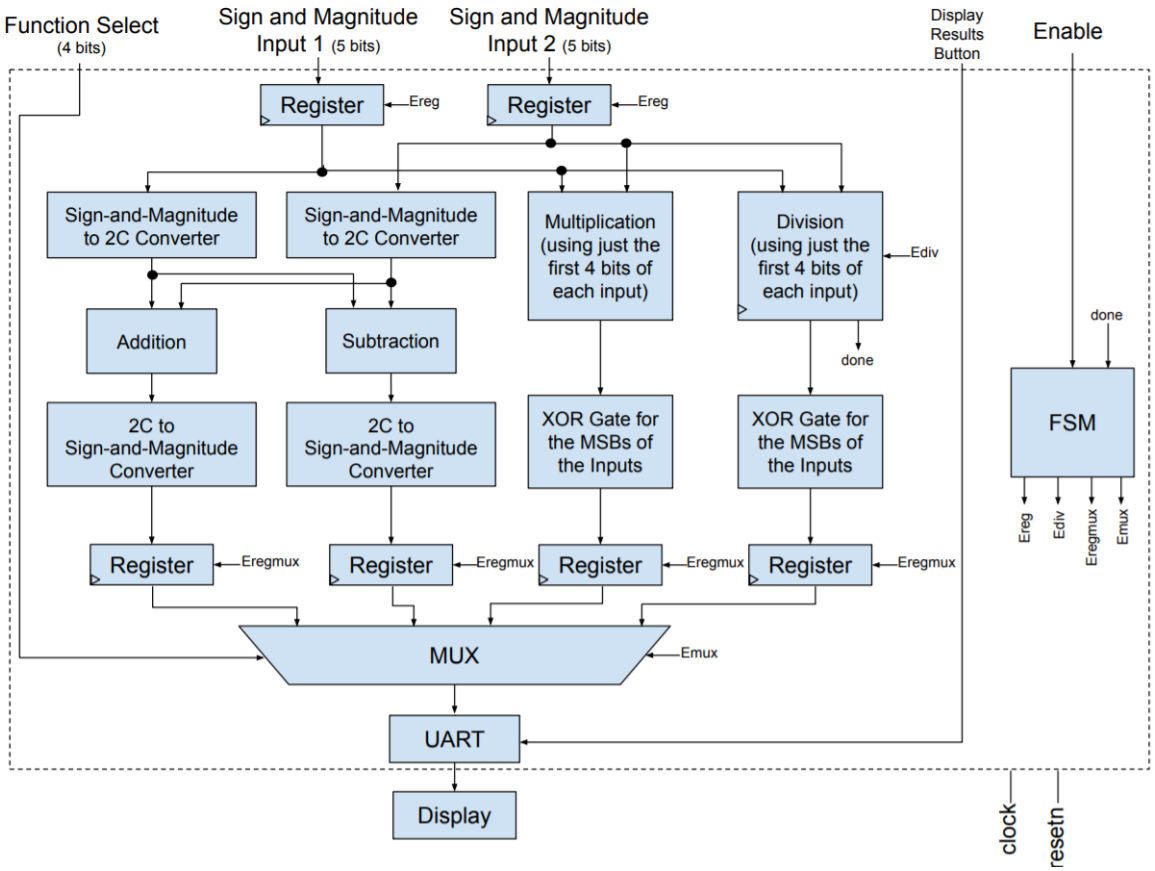


**Figure 9:** Full Block Diagram of Signed Calculator System

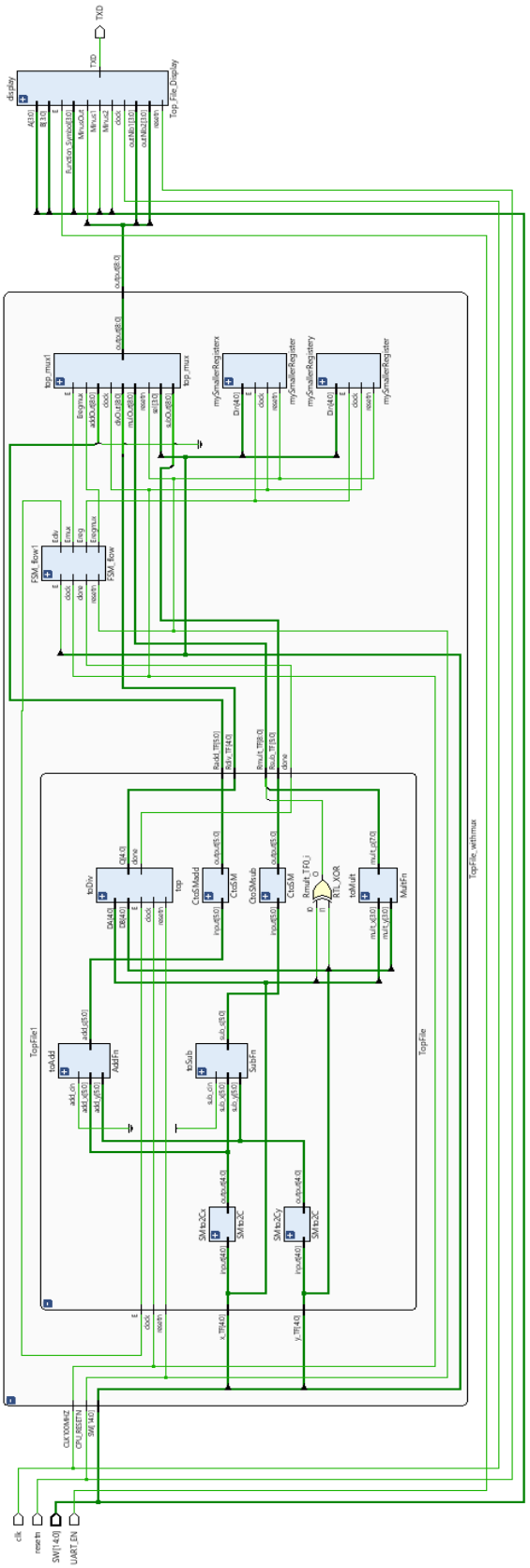

**Figure 10:** Full Schematic of Signed Calculator System