Simple Binary Calculator

Tod Rocco, Connor Willcock, Scotty Knight, Bradly Pfeiffer

Electrical and Computer Engineering Department

School of Engineering and Computer Science

Oakland University, Rochester, MI

e-mails: trocco@oakland.edu, connorwillcock@oakland.edu, soknight@oakland.edu, bpfeiffer@oakland.edu

Abstract:

The overall goal of this project is to create a simple binary calculator that adds, subtracts, multiplies and divides sign and magnitude binary numbers. Calculators are used on a daily basis all around the world and have made problems in math and engineering significantly more accurate and expedient. We will elaborate on the major findings, conclusions and recommendations upon the completion of the project.

Introduction:

Throughout this document we will cover the major sections of this project. This includes how we went about the overall design of the calculator, each individual component, and how we put it all together. We also will include any challenges we faced and how we went about solving them. This project is important because it demonstrates how to create a useful tool we use every day.

This project is very closely incorporated with what we did in class as the adder, subtractor, multiplier and divider were all previous labs. Four bit values were chosen for the two numeral inputs that the user would enter. The main topic we had to figure out on our own was how to handle signed numbers.

Methodology:

Overall Top File:

The overall top file connects all of the major sections of the project. If both binary values are positive it goes into the unsigned portion of the program and if one of the values is negative it goes into the unsigned portion. After the multiplexer selects the correct positive or negative value, the 8-bit number goes into the decoder. Finally, the output from the decoder goes into the serializer to display multiple values on the 7-segment displays.

Unsigned Top File:

The unsigned top file is triggered when both sign and magnitude values are 0. The

unsigned top file is composed of five files, a full adder, subtractor, multiplier, and divider. The full adder is a simple adder that adds the two four bit numbers. If there is a carry out, a 1 is added to the left side of the result. The program then sign extends the number with 0's to make the resulting number 8-bits. The subtractor is very similar to the adder expect the second input's logic is flipped and then added to the first input. We also decided to make the output value the absolute value. The multiplier works through a series of port maps that connect adders to multiply the two binary numbers. Finally, the divider works by connecting state machine with 3 registers, a counter and a full adder. The result of the multiplier and divider are always positive because the input numbers are always positive. The remainder from the divider is mapped to the overall top file MUX. The multiplexor is then used to select the adder, divider, subtractor or multiplier.

Signed Top File:

The signed top file is very similar to the unsigned top file. The inputs are sent to the signed top file when either both or one of the signs of the S&M inputs are 1, meaning it's a negative number. The signed top file is composed of 6 different components: sign and magnitude converter, adder, subtractor, multiplier, divider, two's complement converter, and multiplexor. The inputs first go into the sign and magnitude to 2's complement convertor. If the number is negative, the convertor works by taking in the 4-bit number, flipping the logic, and then adding it to 0001. However, if the input number is positive, it simply outputs the original number. Finally, the program concatenates the sign bit onto the left of the 4-bits. The signed adder works very similar to the unsigned adder except the program sign extends the numbers. It then adds the two 8-bit numbers in 2's complement and outputs the result. The subtractor works in a very similar way except it flips the logic of the second number. The 8-bit numbers from the adder and subtractor then go into the 2's complement to sign and magnitude convertor. This convertor works essentially the same as the past convertor. However, the output number is always positive and there is no sign concatenation. For simplicity, the multiplier and divider are the exact same as the unsigned top file ones. It is important to note that this means the output from all of these segments will be positive. Finally, a multiplexor is used to select the adder, divider, subtractor or multiplier.

Signed or Unsigned Multiplexor:

This multiplexor selects the data from the signed or unsigned top file. The code itself works by using a simple if statement. The if statement checks if either SM bit is 1. If they are both 0, the multiplexor selects the unsigned data, otherwise, it selects the signed data. This multiplexor also chooses which remainder to use for the divider function.

7-Segment Decoder

The main purpose of the decoder is to take the 8-bit input from the previous multiplexor and convert it into 2 numbers. It also determines if the output number should be negative. To determine the 2 output numbers, the program uses a case statement to select values from 00 to FF and create two 7-bit led variables. For example the 8-bit number '10011010' translates to 9A. The decoder makes the first

led value for A ‘0001000’ and the second led value for 9 ‘000100’. The final thing the decoder determines is if the negative sign should be applied to the number. The program works by checking a variety of if statements and if the number should be negative it makes the third led variable value ‘1111110’ (-). For example, if the SM value of A is 1 and the SM value of B is 0 and the two are multiplied together, the result should be negative.

Seven Segment Display Serializer:

The serializer makes it possible to utilize multiple 7 segments on the Nexys A7 board simultaneously [1]. The board only has provisions for one set of data to be displayed on all segments at one time. Our result will utilize three segments to display the sign and two hexadecimal digits. This file displays each segment separately in a rapid fashion so that the human eye can not tell it is flashing each segment one after the other [2]. The program itself utilizes a state machine where each number is represented in a state value.

Experimental Setup:

The first step of testing this project was to simulate the programs using the testbench. We varied each variable to ensure each combination was working correctly. After the simulation aspect of the project was functioning correctly we transition to test on the board. This project was very simple to test with the hardware because we only needed the board. Each input corresponds to a specific switch on the board and the output is displayed on the rightmost 7-segment displays. The simulation and the testing on the board both worked as expected after multiple combinations on input values.

Results:

The best results obtained from this experiment were the simulations. The signed waveform are included in the page below. Figure 1 represents the signed adder, figure 2 represents the signed subtractor, figure 3 represents the signed multiplier, and figure 4 represents the signed divider. Each waveform also includes at least one case where both SM values are 0, representing the unsigned case. Our results were exactly what was expected therefore we did not see and unexpected values.

Conclusions:

The main thing we learned in this project was how to translate unsigned versions of files to the signed versions. We also learned how to combine multiple complex subsystems into one large system. With more time we could have improved the project by making the user interface easier to use. This could be done by inputting the numbers using a keyboard and then displaying the result on a serial monitor.

References:

The majority of the code from this project was derived from previous labs. However, the serializer and counter were found in the tutorial website. The links are shown below:

[1] http://www.secs.oakland.edu/~llamocca/Tutorials/VHDLFPGA/ISE/Unit_7/serializer.vhd

[2] http://www.secs.oakland.edu/~llamocca/Courses/ECE495/Lab/my_genpulse.vhd

Methodology Diagrams:

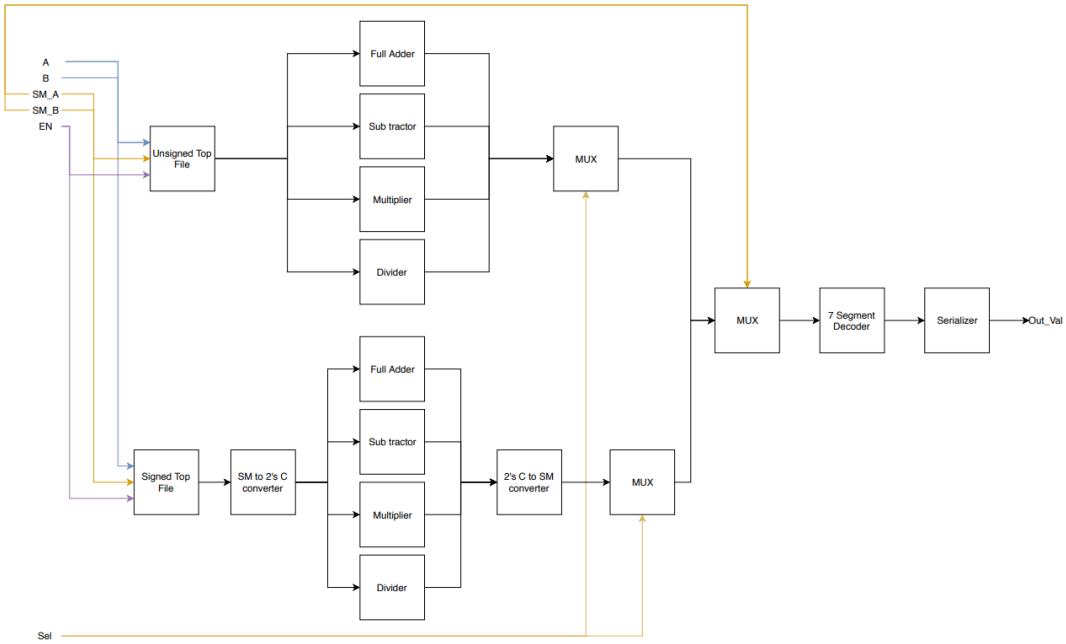

Figure 1: Simple Calculator Block Diagram

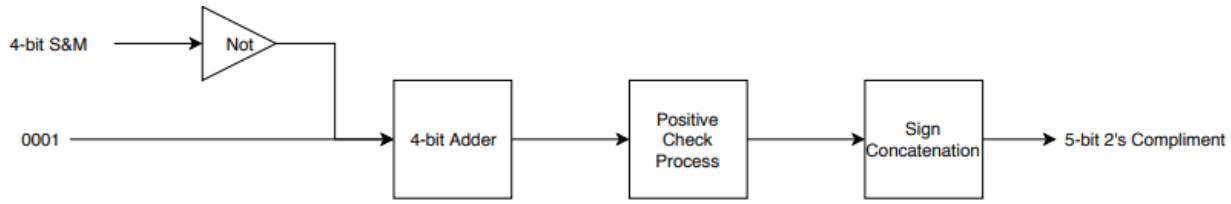

Figure 2: Sign and Magnitude to 2's Complement Convertor

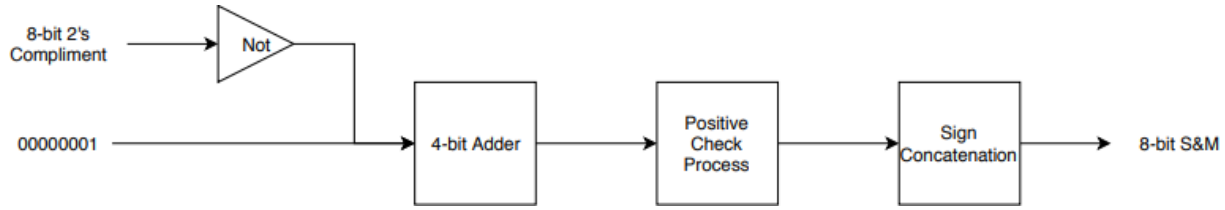

Figure 3: 2's Complement Convertor to Sign and Magnitude

Resulting Waveforms:

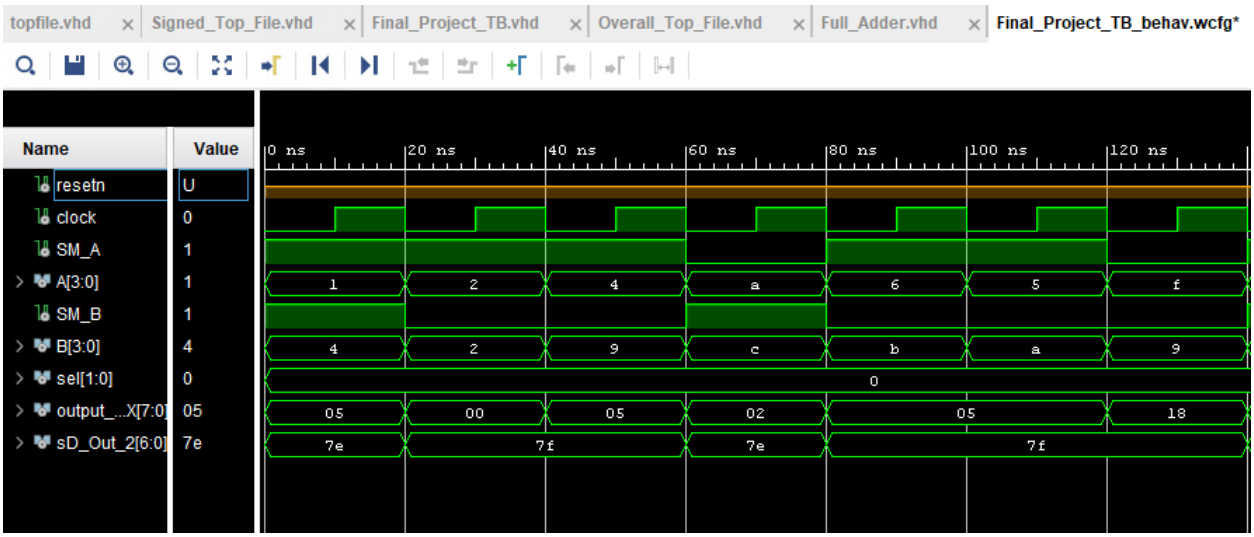

Figure 4: Signed Adder Simulation

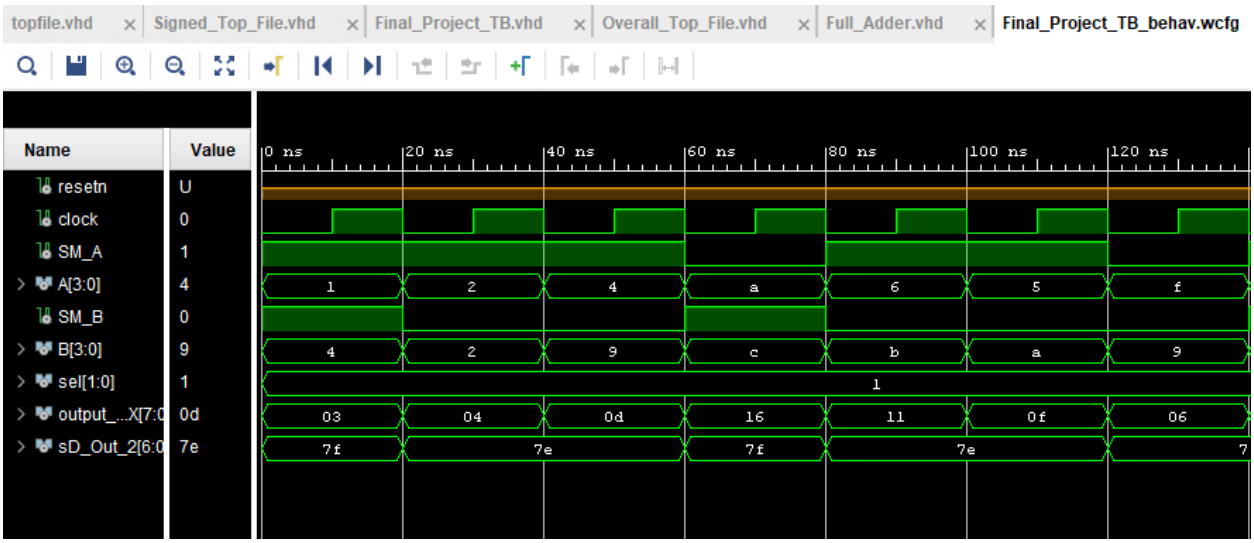

Figure 5: Signed Subtractor Simulation

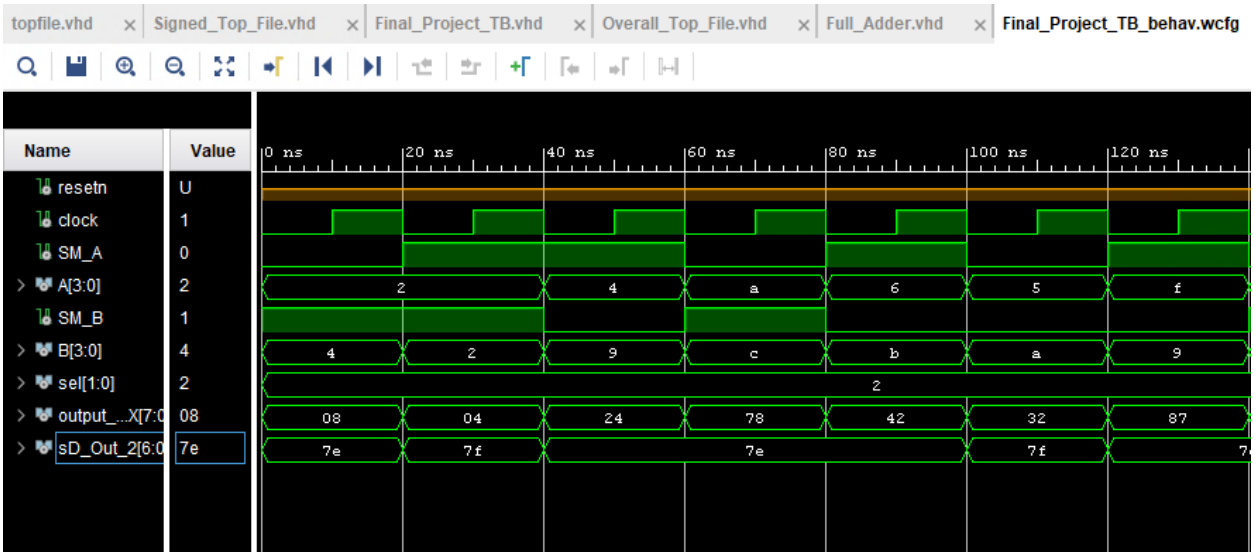

Figure 6: Signed Multiplier Simulation

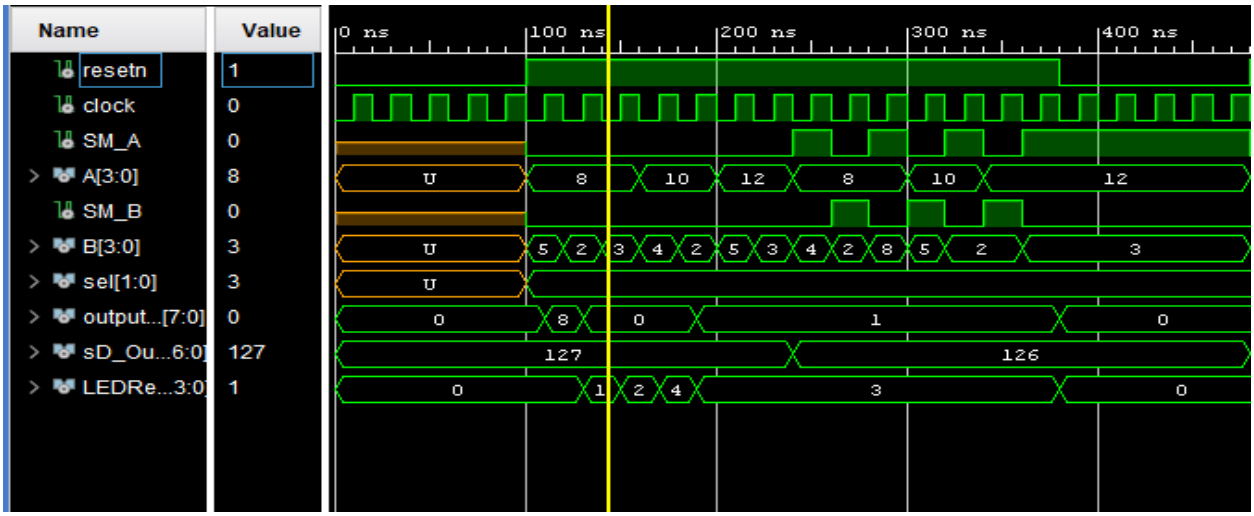

Figure 7: Signed Divider Simulation