8-bit Signed Calculator with Hexidecmial Display and Keyboard Input

EGR 2700 Final Project

Francesca Cipriano, Andrew Fergan, Austin Nieporte, Paige Seyfarth
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI

fcipriano@oakland.edu, afergan@oakland.edu, austinnieporte@oakland.edu, paigeseyfarth@oakland.edu

Abstract

The goal of this project is to create an 8-bit signed calculator that is capable of 4 operations. These operations are addition, subtraction, division, and multiplication. To make data input easy for users, it uses a USB keyboard, and the results are displayed on a 7-segment screen. The answer will then be shown on the Nexys A7 FPGA board as a 16-bit, two's compliment signed number in hexadecimal form. Just like a normal calculator the inputs and outputs can be positive or negative.

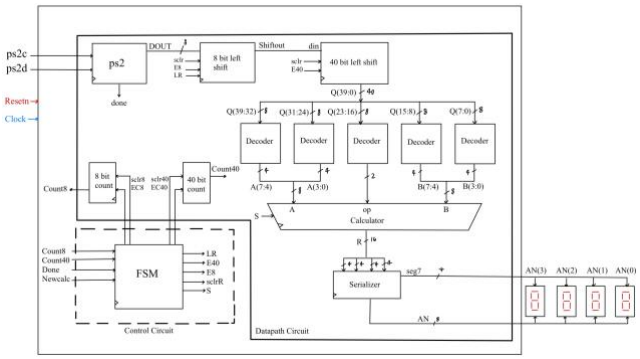

Figure 1: Top File Block Diagram

I. Introduction

The purpose of this project is to create an 8-bit signed calculator. This calculator should perform addition, subtraction, multiplication, and division to function like a normal calculator. The knowledge gained in previous labs have laid the groundwork for construing this calculator. Topics that were most helpful were taught in lab 2, 3 and 6. Lab 2 focused on addition, lab 3 focused on multiplication, and lab 6 was focused on division. Another lab that helped with the design of the calculator was lab 4 because of its similarities to the calculator's architecture. Lab 4 was composed of 4 circuits that were then connected through a multiplexor (MUX) and were output to a hexadecimal to 7-segment decoder. The calculator receives input from a USB keyboard, consisting of five inputs, and outputs the results as four hexadecimal numbers on a 7-segment display.

II. Methodology

As previously noted, the fundamental structure of the calculator consists of four arithmetic circuits, each dedicated to one of the mentioned operations, and they are all connected via a MUX. A standard USB keyboard, as shown in figure 2, was used for user input of desired values and operations. Since a keyboard was used, the use of the PS/2 Interface code, provided by Dr.Llamocca in the Unit 7 notes [1], was needed. A 40-bit register stored all the input data as a string of scan code bits which then went through the decoders as either a hexadecimal value or operation (0-F, +, -, x, /). The output of the decoders were the inputs for the calculator. To display the signed hexadecimal result, the 7-segment display on the FPGA was used. Displaying each hexadecimal place simultaneously required the 7-Segment Serializer code from Dr.Llamocca in the Unit 7 notes [1]. To control the data processing circuit with a USB keyboard, a finite state machine (FSM) was created. All of the circuits in this project are found in the data processing circuit, as well as the two left shift registers and two counters. By having all circuits as a part of the data processing circuit, it allows all calculations to take place through inputs on the keyboard.

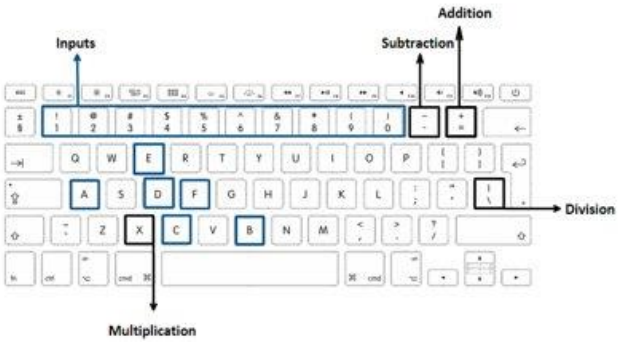

Figure 2: Keyboard Inputs

A. Addition/ Subtraction

The adder and subtractor operand takes place in the 4 to 1 mux when either the addition or subtraction operands are used. Component of the calculator was taken directly from lab 2 with a few minor changes. This is a 9 bit adder for sign extension to avoid oveflow. This block diagram has a one as the input for subtraction and zero as an input for addition. Note that figure 3 is a 4 bit adder/subtractor while out project was modified to be a 9 bit adder/subtractor.

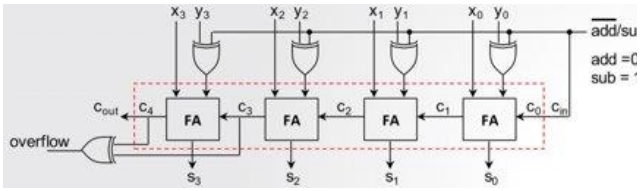

Figure 3: Addition and Subtraction Block Diagram

B. Multiplication

To create the multiplication function, the unsigned multiplier from lab 3 was used. To change the multiplier from unsigned to signed, a two's component was added. The component has two inputs, the 8-bit input and an enable. If the enable is not active, known as being 0, then the 8-bit input will be passed through to the output. If the enable is active, known as being a 1, the component outputs the two's compliment of the original 8-bit input. Both 8-bit inputs, A and B, are put through the two's compliment component before they enter the multiplier. In the two's compliment component, the most significant bit (MSB) is what is given to the enable to decide if the input is passed through or if the input is changed to its two's compliment. The output of the multiplier is also put through the two's compliment component. The enable on this two's compliment is the MSB of A xored with the MSB of B. This corrects the sign of the output of the circuit to its intended value. The MUX then receives the output through a 16-bit bus. This 8-bit signed multiplier is shown in Figure 4.

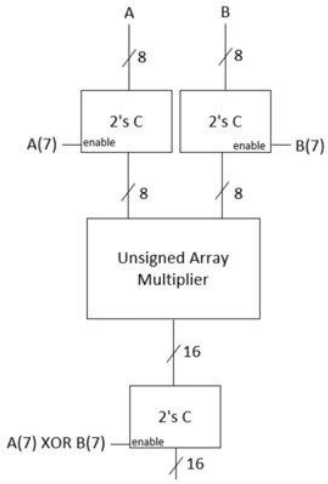

Figure 4: Multiplication Block Diagram

C. Division

To create the division function, the unsigned divider from lab 6 was used. To change the divider from unsigned to signed, a twos's complement component was added similar to the multiplier function. The most significant bit of each input is what determines if the input will pass through or if the input will be changed to two's compliment. The divider used a clock, shift register, counter, and an FSM to control the circuit. Since the input to the mux is 16 bits, the output of the divider function was sign extended to 16 bits, in order to be able to pass through the signal.

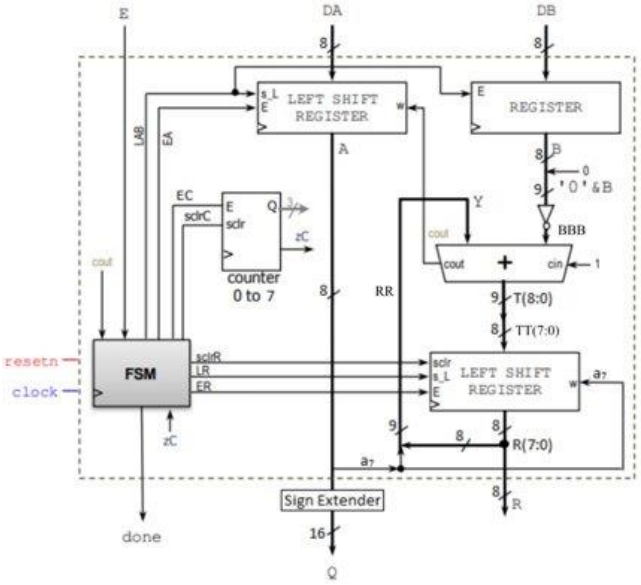

Figure 5: Division Block Diagram

D. PS/2 Interface

The PS/2 interface transfers data synchronously with the clock. The scan code is comprised of 8 data bits and is subsequently stored in the 40-bit register. The PS/2 interface

outputs the scan code through an output signal with a "done" signal lasting one clock cycle. This "done" signal plays a crucial role as an input to the Finite State Machine, instructing the data path circuit to store the output scan code in the 40-bit register.

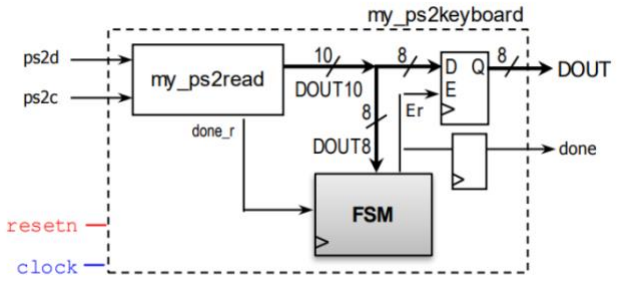

Figure 6: PS/2 Interface Block Diagram

E. Mux Operation Selection

In figure 7, the encoders encode the data from the 40 but register as either hexadecimal input or an operation. The outputs from these encoders are then inputs to the 4 to 1 mux. The purpose of the multiplexer is to take the multiple input signals and synthesize them into a single output. These arithmetic operations are each accessible through a select line. Therefore, the output of the multiplexer is 16 bits and is determined by which operation is being selected by the user.

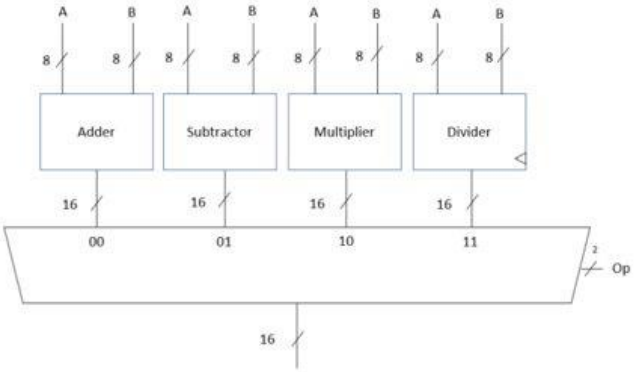

Figure 7: 4 to 1 Mux Operand Block Diagram

F. 7-segement Serializer

The function of the serializer is to display the 16-bit hexadecimal calculation on the Nexys A7 board. The serializer's components were a finite state machine (FSM), a 4 input multiplexor, a counter, a 2-to-4 decoder, and a hexadecimal to 7-segment decoder. The counter controls the FSM and generates a 2-bit output. The FSM controls the select line of the multiplexor. The output of the FSM also passes through the 2-to-4 decoder which basically just activates one of the four 7-segment digits each millisecond. This allows for the 16-bit result to be visible to users. For displaying the calculation, the calculator's output is connected to either the input of the serializer or the multiplexor. For the bus size of the of the calculator to match the input of the multiplexor, the 16-bit bus is split into 4, 4-bit busses. These 4 busses are connected to the multiplexor's input. The figure below shows the block diagram of the serializer. The 7-segment serializer was given in the Unit 7 notes [1].

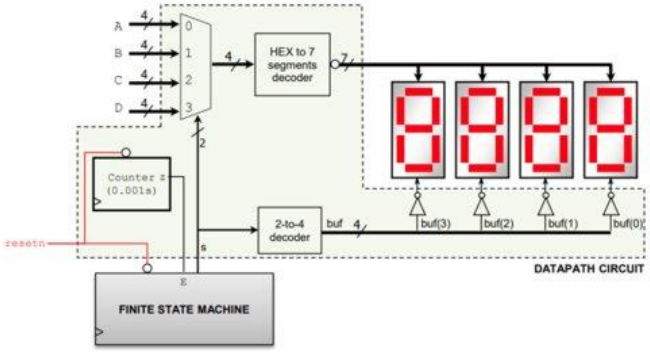

Figure 8: 7 Segment Serializer Block Diagram

G. FSM

This Finite State Machine State consists of 3 sequential states while having signals connecting to the two counters and two registers. In state 1, the machine waits for an input from a key stroke on the keyboard. If an input is detected, it is then loaded into the 8-bit register. State 2 is responsible for detecting all 40 bits from the 5 input keys. When state 2 counts each bit, the output is 0 and returns to state 2 until all 8 bits are detected. When all 8 bits are detected the output becomes 1 and enters the count40 signal. Count40 signal returns output 0 and returns to state 1 after 8 bits are detected. Once all 40 bits are accounted for, then state 2 can finally enter state 3. When state 3 is hit, the calculation process is complete and the middle button on the Nexy's board bush buttons as in figure10 is used to start a new calculation. If the user wanted to restart a calculation, they could simply press the reset button.

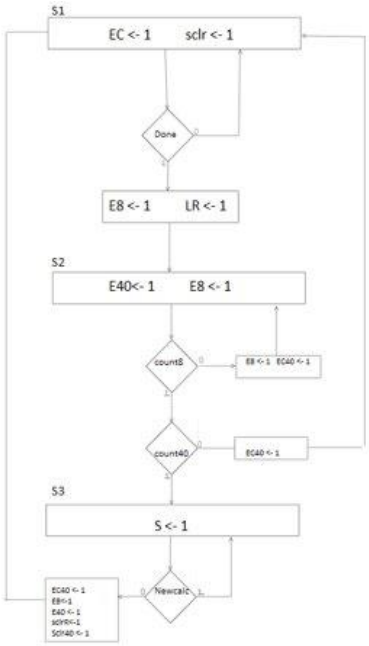

Figure 9: Finite State Machine Block Diagram

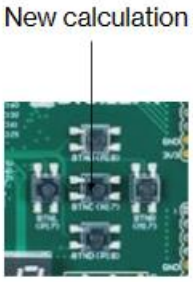

Figure 10: Nexy's Push Buttons

H. Left Shift Register and Counter

The left shift registers and counters work in parallel with each other and the finite state machine. These are actively working during state 2 in the finite state machine. It is important for each bit to be accounted for in order for the calculator to work properly. The shifted output bit of the 8 bit register is then the input to the 40 bit register. These values keep shifting left until all 40 bits are in the left shift register and accounted for by the 40 bit counter. The counters have importance by counting how many bits in the register in order to output a 1 or 0 which then determines the path on the finite state machine. The output 1 means the counter is full whereas the output 0 signifies that the counter has not reached is maximum bits.

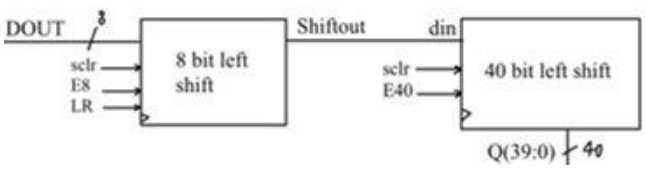

Figure 11: 8 Bit and 40 Bit Left shift Registers

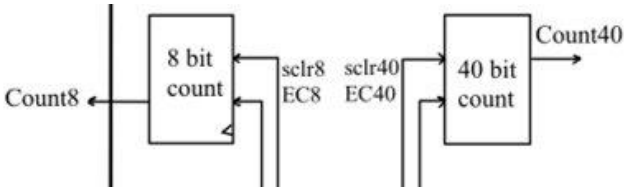

Figure 12: 8 Bit and 40 Bit Counters

III. EXPERIMENTAL SETUP

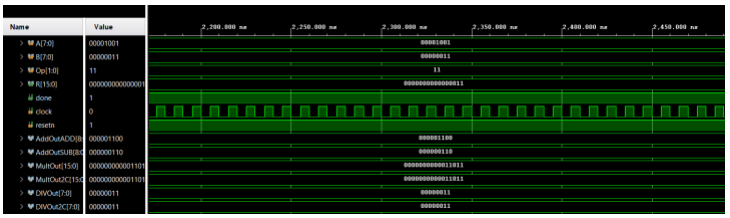

Figure 13: Behavioral Simulation

Experimental results from this lab are shown in the behavioral simulation created from cases in a test bench. It displays all of the results of all operations as well as the overall result based on the operation input. The testbench was used for troubleshooting errors in the code. One encountered was with division and this was found when we noticed that the "cout" signal was always on causing for incorrect calculations. This was resolved by rewriting the code for the subtractor in division to resolve this.

IV. RESULTS

The calculator developed in this project functions similarly to a normal calculator, because it incorporates various functions like addition, subtraction, multiplication, and division. This calculator uses a USB keyboard for inputs instead of a keypad. The keys that are used as operations are "+, -, x, and /". The output number will always be in hexadecimal. This is to account for a negative result. The results of the inputs are displayed using the seven segment displays on the Nexys A7 board.

CONCLUSIONS

After the completion of this calculator, some conclusions were drawn. The creation of this calculator was an excellent method to show what topics were learned in class and how to put them to use. One topic that was used a lot in this project was port mapping from the different components to the top file. To create a working calculator, there were numerous design files, a test bench, and a constraint file. When these files were integrated correctly, the calculator was finished. Once the calculator was working, it was fun to test out all the operations. When reflecting on this project, it is easy to see why learning VHDL coding and using Vivado is important for our future careers.

REFERENCES

http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

[1] D. Llamocca. (2021). ECE2700 - Unit 7: Introduction to Digital System Design [PDF].