Calculator

Sander Babby, Alin Stan, Dustin Hoskins
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
sgbabby@oakland.edu, amstan2@oakland.edu, dhoskins@oakland.edu

Abstract

The purpose of the project was to build a simple calculator using a keypad, LCD, and FPGA (Nexys 4). The calculator would be able to add, subtract, multiple, and divide 4 bit numbers. The project was a great learning experience for digital logic and VHDL. An example is interfacing multiple components and performing computations on the FPGA. The project was able to perform properly in simulation, but it did not perform correctly in real life because of a timing issue with the outputting the result to the LCD. A recommendation would be to keep the signal high long enough for the LCD to print the result.

I. Introduction

The scope of the project was building a 4-bit calculator using a keypad, LCD, and FPGA (Nexys 4). The team wanted a project that would involve many topics covered in ECE 378 and challenge the team to complete the project. This was a good project to do because it incorporates many different digital logic principals such as external interfaces and architecturally based with arithmetic operations. Some of the topics that were covered in class were decoders, Arithmetic Logic Unit (ALU) design, and Finite State Machines (FSMs). Topics that had to be researched outside of class were interfacing a keypad and LCD to the Nexys 4. This project has many applications such as being used as a traditional calculator, app on a smartphone, or a part of a larger project that would need calculations performed. Many modern electrical and software applications depend on calculations to be performed in real time in the background using a calculator such as the one in this project.

II. Methodology

A. Keypad

A decoder in the project represented the keypad. The decoder would receive a signal from the ALU to determine to clear or not. It would also receive an input from the keypad. The way this works is if a key is being pressed it determines it by matching the row being pressed with what column in that row is being pressed. That is how the keypad determines which key is inputted. The keypad in addition to the value being sent that is being pressed, it would send two additional signals to the ALU. These signals were a zO and a ZN. These would be high or low depending if a number was pressed or if an operator was pressed. If the value of output from the value from the decoder was less than 10, then zN would be high and zO would be low. Likewise, if the value from the decoder was greater than 9, then zO would be high and zN would be low. This would help the ALU determine if numbers or if operators were being pressed. The LCD did not need this data since it displays what is pressed and not running background computations.

The keypad was switched to the switches halfway throughout the project. This was done to remove any variation coming from the keypad. The original keypad had a bend in the ribbon cable, which was causing bouncing between rows. The decoder stayed the same in what it was inputting and outputting. Later in the project, a new keypad was received and tested. This newer keypad was able to work well with minimal bouncing. The newer keypad was not connected to the project at the end because the team did not want the minimal bouncing to hinder test to allow for quicker debug time. The decoder code did account for bouncing inside of the code. This is what took care of most of the bouncing in the keypad.

B. Arithmetic Logic Unit

The ALU represented the brain of the project. Its job was to be able to receive the numbers that the user inputs and output the result. The use would input numbers in decimal and would expect to see a decimal output. This was the first obstacle that we faced. The numbers would be coming in and it was up to the ALU to figure out what the input was, and what the operand was. If a user enter a 1, 2, and then 3. The ALU had to take those 3 separate inputs and make it the decimal number 123. This was done by using a state machine. In the first state the machine would receive the first number, store it in a variable, and then go to the next state. This state would multiple the first number by 10 and then add the second input to this number before going to state three. In each state it would take the previous states result and multiple it by 10, then add the new input. If at any time the ALU received an operator (+ - * %) the state machine would stop and hold its current value. There were two of these state machines used. One to capture the first number, and one to capture the second. There was a parent State machine which controlled all aspects of the ALU's functions. In state 1 of the ALU state machine it would enable to first number catcher. It would only change states when an operator was received. Then in state 2 it would disable the first number catcher and enable the second. It would stay in state 2 until it received the input for equals. It would then transition to state 3. This state performed the calculations using a series of if statements and would then output the result to the LCD’s divider. The FSM is outlined in below in figure 1.

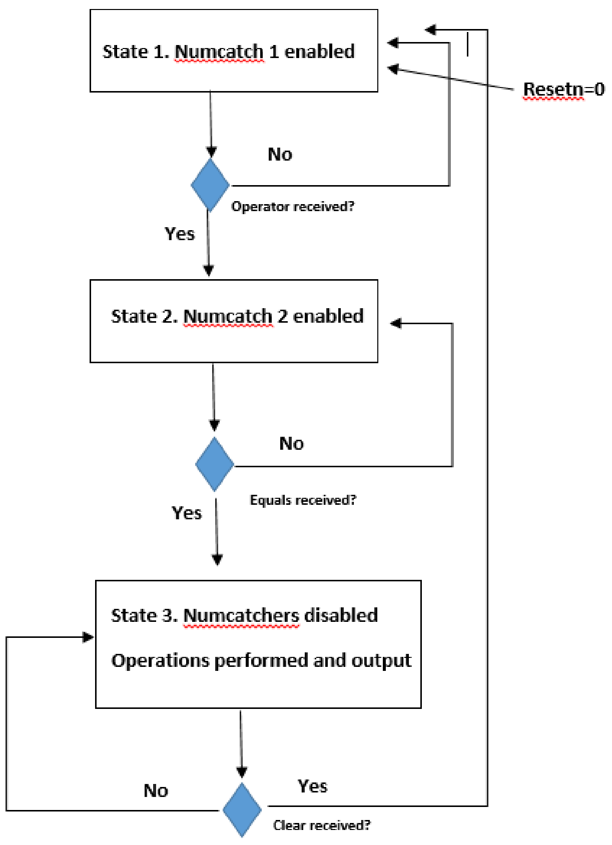
State 1. Numcatch 1 enabled
Resetn=0
No
Operator received?
Yes
State 2. Numcatch 2 enabled
No
Equals received?
Yes
State 3. Numcatchers disabled
Operations performed and output
No
Yes
Clear received?

Figure 1. FSM of project ALU.

C. LCD

We attached an LCD Unit to our Nexys 4 Board. The main code came from Dr. Llamocca [1][2]. The rest of the code we created ourselves. For the first portion of the equation, which we consider as the first number input, the operator, the second number input and the equals input is a direct link to the LCD. When you select an input on the switches, or if using the keypad, pushing a button, the LCD automatically write the character received. There is a component called a divider, which is really just a state machine, which takes in a 4-bit number and converts that to an ASCII character and then that gets sent to the LCD. The divider stay in the same state and keeps writing to the LCD until the equals sign is pushed. When the equals sign is pushed the divider moves into state two. In state two the divider no longer receives an input from the switches; it now receives an 8 bit input from the ALU, it also sends an enable to the next component called result catch. The number from the ALU can be anywhere from zero to 225 in decimal. The divider then starts ripping the number apart in order to try and get the characters. It has 3 outputs. All of them are zero at first and the outputs are in ASCII 8 bit notation. First the divider checks if the number is greater than 200, if it is then the first output will be a 2 and subtracts 200 from the input and saves it into input2, if it is between 100 and 199 then the first output is a 1 subtracts 100 from the input and saves it into input2, if it is less than 100 then output one is blank and saves input to input2. Then the divider checks for the numbers by 10’s outputs the number to output2 and subtracts the largest multiple of 10 from it and saves the remainder into input3. It then finally outputs input3 to output3 and send a finished signal to the result catch.

The 3 outputs are now inputted into result catch, which is another state machine. This state machine is designed to slow down the writing process and output each number one at a time to the LCD. The LCD was receiving the numbers to fast so we tried to slow down the output. This is where things started going wrong. We were never able to output the final result to the LCD. We believe this is due to the timing required by the LCD. We also had to implement an enable signal for the LCD, which also came from this result catch. We were able to show that the calculator does actually work through a simulation; however we are not able to show the actual result on the LCD. Based on the timing diagram show below we believe that we were sending the signal out at the wrong time and the LCD was not ready to be written to, and was not correctly enabled at the right time and for the right amount of time. We should have seen the rs as high and e as high in order for us to write to the LCD, which in our case did not happen as it should have as represented in figure 2 below [3].

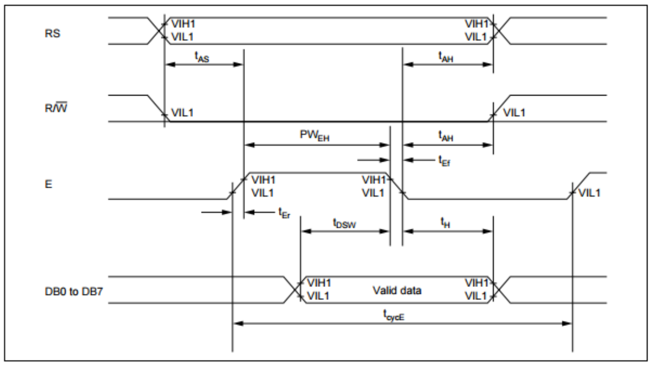

Figure 2. Timing required to write to LCD.

III. Experimental Setup

To verify the functionality of the calculator different hardware and software test methods were used. For the software, Xilinx to code, debug, and simulate the project. Each individual component was simulated in Xilinx’s ISim to verify it was working correctly. After this was accomplished, the components were combined into a top level and simulated to verify the functionality of the top level.

For the hardware, the components were tested individually to verify functionality. An example of this is connecting the keypad to the Nexys 4, pressing a button, and outputting the result to the seven-segment display of the Nexys 4. This was also done with the ALU by inputting values from the switches and outputting the result to the seven-segment display. For the LCD, values were inputted using the switches and then sent to the LCD by pressing a pushbutton on the Nexys 4. The value was being displayed in ASCII on the LCD.

After components were individually tested, they were combined together to verify the functionality of the system. This was accomplished by connecting the keypad and LCD to the Nexys 4. On the keypad, entering a number followed by a arithmetic operator then another number and the equals operator will send the necessary signal to the ALU and LCD. As buttons are being pressed on the keypad, the LCD is displaying what is being entered. The ALU is performing the necessary calculations as buttons are pressed so that once the equal operator is pressed, the result is sent to the LCD to display it. The LCD should then display the result next to commands that were entered. Also, to rule out any bouncing by the original keypad, switches were used to send the signal to the ALU and LCD in testing. This should display the numbers entered, the arithmetic operator, equal operator, and the result on the LCD.

IV. Results

The project produced results as expected for the most part. In simulation, the project was able to send the signals correctly to and from components then send the final result to the LCD. In implementation, the calculator performed similarly to the simulation except with not displaying the calculated result. The LCD would receive the result but would not show the result being displayed. Besides the LCD not displaying the result, the rest of the projected would worked properly. For example, the signals being sent to the ALU and the LCD from the keypad were correct. The ALU would then take the signals, properly perform the necessary calculations, and then send the result to the LCD. The LCD would take the result as an input, but a timing delay issue would cause the LCD not have enough time to write the result on the display. Besides the result, the LCD would show the numbers being entered into the system, the arithmetic operator, and the equal operator on the display.

The project was able to perform correctly because of topics covered in ECE 378. FSMs were able to work as a control for the calculator within the ALU to send, receive, and calculate signals as needed for the project. The LCD was also implemented using a FSM. The keypad was able to work by using a decoder. Besides the LCD not displaying the calculated result, the calculator performed as expected. The results in the projected were explainable. For example, switching keypads removed the bouncing for the most part because the connection was not bent as such in the original keypad. Although the result did not display on the LCD, this was explainable because the signal was not being sent long enough to the LCD for it to write the result.

V. Conclusions

This project was an exceptional learning experience for implementing the topics covered in ECE 378 in a real world project. The team was able to gain experience with important topics such as FSMs, ALU, decoders, computer arithmetic, and VHDL programming. Another important take away from this project was working as a team to integrate individual components that work separately on its own but require work to implement correctly together. This will provide as excellent experience when moving into industry and working with teams in large corporations.

As discussed, the project did not perform 100% as expected. The main issue remaining was the result not being displayed on the LCD and connecting the keypad to the final system. Displaying the result to the LCD could be improved by determining the proper timing when displaying a result from the ALU instead of the keypad. Once the correct timing delay is found, then the result could be displayed on the LCD. Another alternative would be to displaying the result to the seven-segment display. This could have been done either for demonstration purposes in class or as a permanent solution to the LCD not displaying the result. Furthermore, the keypad could have been improved by improving the minimal bounce that was left in the keypad.

The project was able to enhance the team's understanding of digital logic and provide lessons learned for moving forward in school and industry.

References

[1] Llamocca, Daniel. “[ECE378] Final Project.” Message to the authors. 2 Apr. 2015. E-mail.

[2] Llamocca, Daniel. “Re: ece 378 Final project.” Message to the authors. 20 Mar. 2015. E-mail.

[3] Hitachi. HD44780U (LCD-II). Tokyo, Japan: Hitachi, 1998. PDF.

[4] Haskell, Richard E., and Darrin M. Hanna. Digital Design Using Digilent FPGA Boards. Rochester Hills, MI: LBE Books, 2009. Print.