Simple Calculator
ECE 2700 - Digital Logic Design Final Project Paper

List of Authors (Benley Mathew, Matthew Stopyak, Matthew Wagner)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: bpmathew@oakland.edu, matthewstopyak@oakland.edu,
matthewwagner@oakland.edu

ABSTRACT

This project demonstrates how users can use two eight-bit unsigned integer numbers for the four main operations of math, which are addition, subtraction, multiplication and division. While this concept sounds simple in nature the actual task of implementing this is complex and can be confusing. The actual math of it was easy to understand and implement, while the input and output mapping and control was more challenging. We came to the conclusion that an 8 bit calculator is not very useful but after the heavy lifting, implementing the calculator states and I/O integration, it could easily be expanded to accomplish larger more useful tasks.

I. INTRODUCTION

For this project, a simple calculator was created. The motivation of this project was that the main math operations were created in the labs of this course. Other main components were a BCD to Binary converter and a seven segment display. Some other components also include registers to store the data values of the two numbers. What we learned from this project is to clear variables in the sensitivity of the process. Another thing that we had to learn on our own was how to converter BCD to Binary and Binary to BCD. This method is called Double-Dabble, where you convert numbers from Binary to BCD and vise versa. We also learned how to troubleshoot error messages from our project. This project can be used for any person who needs to use a calculator for simple math, such as elementary school students who are just starting to learn basic mathematics.

II. METHODOLOGY

A. Inputs and Outputs

The first parts of this project is the input and output values. The input values' range is determined by the number of bits this calculator is built for. Since the decision of this calculator is eight-bit unsigned integer numbers, the range for the inputs is from 0 to 255 binary. Furthermore, since we are using a 4x4 Keypad, it is further limited to the range, 0 to 99. Once the range was decided, the next step was to figure out how many input values that will be used. To avoid complex calculations, two numbers would be enough. These values would then be converted from BCD numbers into Binary.

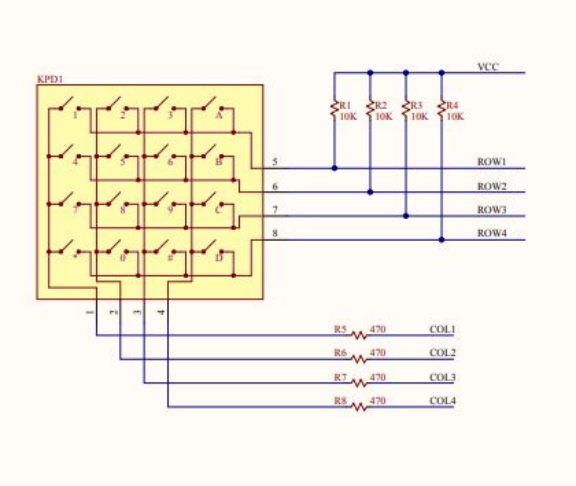


Figure 1: 4x4 Keypad Schematic [2]

After the numbers go through the operations, the results will be converted back from Binary to BCD and will be outputted onto a seven segment display. For division, if the number has a remainder, the remainder will be shown on segments 3 and 4 of the seven-segment display. Additionally, for the seven-segment display to work with the multiple digits, we had to create a multiplexer that would select each digit one at a time and display the correct value.

For a static system the human can not discern rates above 30Hz so we arbitrarily selected a value of 40kHz for this scan cycle related to the operation of the seven segment display.

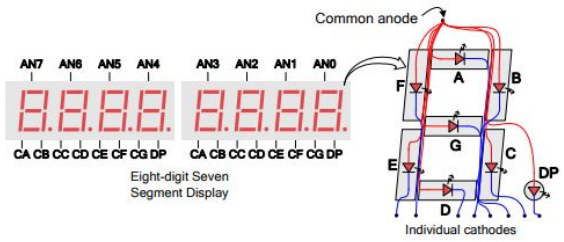


Figure 2: Seven Segment Display [1]

Additionally, leds 15 through 12 were programmed to display the state of the calculator. This helped immensely when troubleshooting and debugging the systems.Led 0 is the “carry out” of the adder/subtractor circuit and led 1 is the “overflow” from this same circuit. Blue led 17 is programmed to come on whenever a button is pressed on the 4x4 keypad.

B. Operations

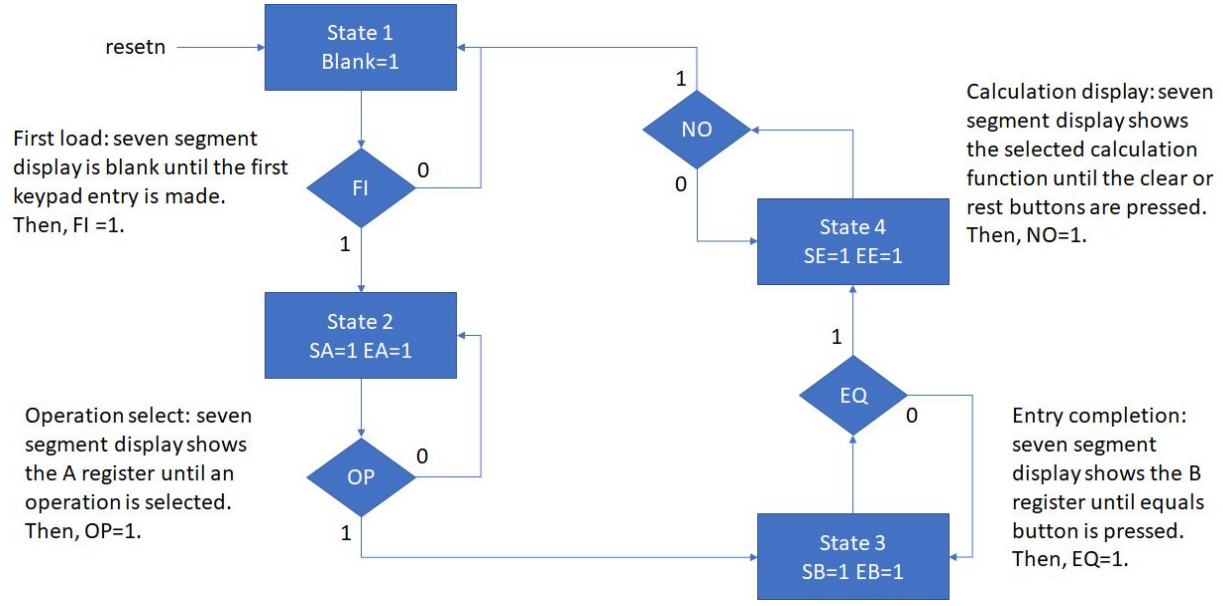
The four mathematical operations that were programmed were addition, subtraction, multiplication and division.

The addition operation was created from a lab from the course and was modified to accommodate eight bit unsigned numbers.

The subtraction operation was based of the addition, the “addsub” input is set to a one enabling the subtraction function of this circuit.

The multiplication operation was also created from a lab from the course and was modified to accommodate eight bit unsigned numbers.

Figure 3: State Diagram of the Calculator



The division operation was created from a lab from the course and was modified to accommodate eight bit unsigned numbers with an LED system to display the remainder.

All of these operations will be connected to the operational keys on the keypad. The operations will be executed to the two inputs once the equal key is pressed.

To control the flow of data through the calculator, we implemented a four position finite state machine. This

State 1 is the initialize state in which the calculator is waiting for its first input value. Upon receiving the first numeric value, the calculator moves to State 2. In State 2, the calculator is collecting the first numeric entry for the seven-segment display. Once an operation button is pressed, the calculator moves to State 3, and the first numeric value is locked into its register. In State 3, the calculator is collecting the second numeric entry and displaying this data on the seven-segment display. Once the “equals” button is pressed the calculator moves to State 4 and locks the second numeric entry into its register. In State 4, the calculator displays the value of the selected function that has been performed on the numeric inputs. This value will remain until either the “Clear” or “Reset” buttons are pressed. At this point the calculator will move back to State 1 awaiting its first numeric entry.

III. EXPERIMENTAL SETUP

For this project, we have a keypad that is used to input the two values, an operation, and an equal sign. The model of the keypad is the PmodKYPD. We also used the NEXYS 4 DDR Board that was purchased for the class and the Vivado Program used in the lab. We defined the operations for the calculator on the keypad as the following; ‘A’ is addition, ‘B’ is subtraction, ‘C’ is multiplication, ‘D’ is division, ‘E’ is equals, and ‘F’ is clear. For the NEXYS Board, the reset button will reset the program if pressed and the seven-segment display to present the answer to the user. The Vivado Program is used for our coding for the the calculator. The expected results will be that it calculates the answers to what the user asks of it. Example, 1 + 1 = 2.

For the Vivado Board, these are the things we used on the board.

1. Reset button just resets the program
2. LED 15: State 1
3. LED 14: State 2
4. LED 13: State 3
5. LED 12: State 4
6. LED 11: Enable Register A
7. LED 10: Enable Register B
8. LED 9: Enable Register E
9. LED 17 (Blue): Key Press

10. LED 0: Carry Out
11. LED 1: Overflow
12. Seven-Segment:
    a. Digit 8: “=” sign
    b. Digit 5 and 6: Number Display
    c. Digit 4: “r” for remainder
    d. Digit 3: remainder value

IV. RESULTS

Table 1: Calculator Project Test Results

Operation	Data A	Data B	Solution
Addition (+)	12	36	48
	3	27	30
	99	99	98 carryout
Subtraction (-)	48	12	36
	98	8	90
	54	52	2
Multiplication (*)	3	6	18
	20	4	80
	9	9	81
Division (/)	29	10	2 r9
	48	12	4
	56	5	11 r1

Link to calculator demo: https://photos.app.goo.gl/EvsWD4Uwr1oxDwWv5

In all case the results were as expected. However, do to the unsigned nature of our subtraction the results of the arithmetic, when Data B is greater than Data A, were not representative of the true nature of the operation because the result value is unsigned. As a future improvement the calculator should be converted to using 2’s complement binary arithmetic.

V. CONCLUSIONS

Some issues we faced included the integration of the keypad, including signed numbers in our project.

The decoding and button debouncing of the keypad were extremely difficult to decode due to the nature of the 4x4 wiring. It was especially difficult to determine when the same button was pressed in successive order. For each column scan if that particular row did not have a button depressed, then it would have have an intermediate value set to high. If all column read were high, then no button was pressed.

The second sticking point when integrating the components of this calculator was the shift nature of the BCD output. For each button press, four bits had to be shifted into the four least

significant bits of the appropriate register. Initially we were trying to use a “button event” function to accomplish this task but it wasn’t working. After some research and reading about VHDL and Vivado, it seems the “event function” is specifically reserved for clock functions which are synchronous in nature. We were attempting to use this in an asynchronous manner. This makes sense since VHDL is a hardware descriptive language. We then developed a definite “button press state” and any change latched a synchronous clock would mean that a button was pressed. The register would then shift in four bits of data.

Potential improvements

1. Using 2’s complement binary arithmetic
2. Drop the leading zeroes on the display
3. Institute an “all clear” and clear function
4. Expand the arithmetic capabilities to handle a greater number of bits.
5. Rework the FSM and integration of coding so it is easier to read
6. Replace the seven segment display with LCD display for greater functionality

Overall, this project was much harder than anticipated. VHDL may look like a coding language but it is not. When you forget that it is hardware descriptive and treat it is a coding language, the problems you will encounter with your code will intensify. This was a good learning experience for beginners and we are walking away with some valuable lessons.

REFERENCES

[1] Nexys 4 DDR Reference Manual. (n.d.). Retrieved from https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual

[2] Pmod KYPD. (n.d.). Retrieved from https://reference.digilentinc.com/reference/pmod/pmodkypd/start