3x3 Matrix Multiplier

List of Authors (Stone Maguire, Jared Panizzoli, Kelly Koscielski)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: stonemaguire@oakland.edu, jpanizzoli@oakland.edu, kellykoscielski@oakland.edu

***Abstract*—3x3 matrix multiplication which stores each sector of the matrix in memory, multiplies the values individually, and then decodes from hexadecimal to be able to display on a 7-segment display. Each row of the resultant matrix is displayed at one time on the 7-segment displays of the FPGA and the three rows can be displayed by flipping the respective switches.**

I. Introduction

The 3x3 matrix is designed to create an issue free matrix multiplier. While doing matrix multiplication by hand, there is room for human errors along the way. With this 3x3 multiplier designed in Vivado and using the Nexys board, there is no room for human error. Since this 3x3 matrix multiplier can only go from 0 to 7, it will be used for smaller projects where the numbers may not be so high.

The purpose of this project is to implement a 3x3 matrix multiplier using VHDL on the Nexys A7 50T FPGA. The design of this project is greatly inspired by the Lab 5 Random Access Memory Emulator. Matrix multiplication can be long and difficult, especially for a 3x3 matrix, to do by hand and having a computer solve the calculations required will eliminate any potential errors. For this matrix multiplier, the user will select nine unsigned binary numbers for each matrix ranging from zero to seven to store in the registers. The numbers in the registers will be read and then multiplied together using binary multiplication to create the final matrix, which will then be displayed in hexadecimal on the seven segment display located on the FPGA.

II. Methodology

There are three main sections to the implementation of the 3x3 matrix multiplier: Using switches to control the matrices and input data into the matrix, the mathematical operations being performed to solve the matrix, and lastly displaying the resulting matrix on the seven segment displays.

A. Controlling the Matrix Using Switches

For the 3x3 matrix, the Nexys A-7 50T FPGA will be used, along with Vivado to write code for the matrix. A total of 11 switches will be used. The function of these switches are as follows: Two switches to switch between matrices, one for each matrix, switch in the high position meaning that matrix is selected. Each sector of the matrix will be stored in a register, with a total of nine registers being used. Four switches will be used to select each register to store a number into. Three switches will be used to choose a number to put into the matrix, zero through seven, that will be then stored in each register. When both matrix enables are low, writing to the registers is disabled, allowing the values to be read, so the seven-segment display will be on. To display each row, two switches will be used to select which row of the matrix will be displayed on the seven segment display of the FPGA in hexadecimal values, since there are not enough seven segment displays to show all of the values at once.

B. Mathematical Operations

To accomplish the multiplication of the matrices a six-bit binary multiplier will be used. Each inputted number will be sign extended three bits for a total of six bits each to account for product results being greater than the three bits of the numbers. Then using binary addition, the three resulting values for each sector of the matrix will be added together using full adders. The resulting values will each be stored in a register, totaling eighteen registers. These values will then go to a multiplexor, which will use the two switches to select the row to be displayed.

C. Display of Data

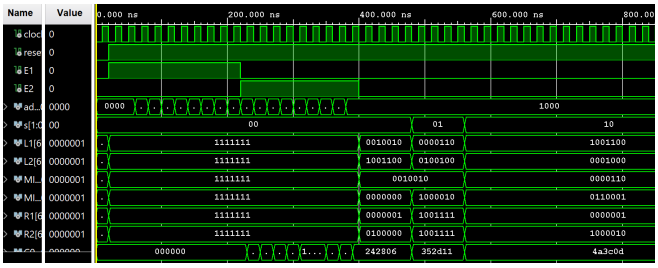
To display each row of the matrix on the seven-segment display, six out of eight of the displays will be used. Each row is able to be displayed all at once using a counter, finite state machine, and a 3-to-6 decoder. When the counter is high, the finite state machine switches states. There are a total of six states, one state for each value in the row of the matrix. This change is happening every 1 millisecond. The 3-to-6 decoder is selecting which display is being lit up. With changes being made every 1 millisecond, it appears that the seven-segment displays are all lit up at the same time, allowing for multiple numbers to be displayed at one time.

III. Experimental Setup

A simulation of the circuit was done to show the output of the matrix when certain values were inputted. The numbers chosen to test the circuit were coded into the test bench file. To begin, E1, E2, resetn, s, address, and DI signals were set to 0. Then E1 was set to 1 while keeping E2 at 0 so that matrix A could be written to the registers. The values of DI were cycled through while changing the values of address as well so that the data was going to the correct registers. Then, E1 was set to 0 and E2 was set to 1 so that this process could be repeated for writing the matrix B values into the registers. Finally, E2 was set to 0 and the

values of s were cycled through to show each row of the final matrix as seen in the simulation screenshot below. By doing this, it allows the output values to be verified to check the functionality of the circuit.

Then this circuit was tested experimentally on the FPGA with the same values from the test bench, using the switches on the board. The enable switches (E1 is SW15, E2 is SW14) were tested to be sure the board was reading and writing to the registers at the correct times, verifying that the seven-segment display turned off when an enable switch is high and turned on only when both enable switches are low. Then, the switches for the register addresses (SW3-6) were tested to choose which sector of the matrix the data will be inputted in. After the data (SW0-2) is inputted for each row and column for the matrix, the switches for the multiplexor (SW12-13) were tested to choose the row that is to be displayed. Below is the simulation showing the output of the matrix on the last row.



Simulation with matrix inputs

IV. Results

The following matrix was inputted into the FPGA in order to test the functionality of the circuit to get an outcome of the correct resulting matrix. The result is also seen above in the timing diagram simulation for the circuit. The overall block diagram of the circuit with all the components connected together can be seen at the end of this report.

$$\begin{bmatrix} 2 & 4 & 2 \\ 5 & 0 & 6 \\ 7 & 4 & 3 \end{bmatrix} X \begin{bmatrix} 7 & 3 & 1 \\ 4 & 6 & 0 \\ 3 & 5 & 2 \end{bmatrix}$$

Inputted matrices in decimal

$$\begin{bmatrix} 24 & 28 & 06 \\ 35 & 2D & 11 \\ 4A & 3C & 0D \end{bmatrix}$$

Resulting matrix in hexadecimal

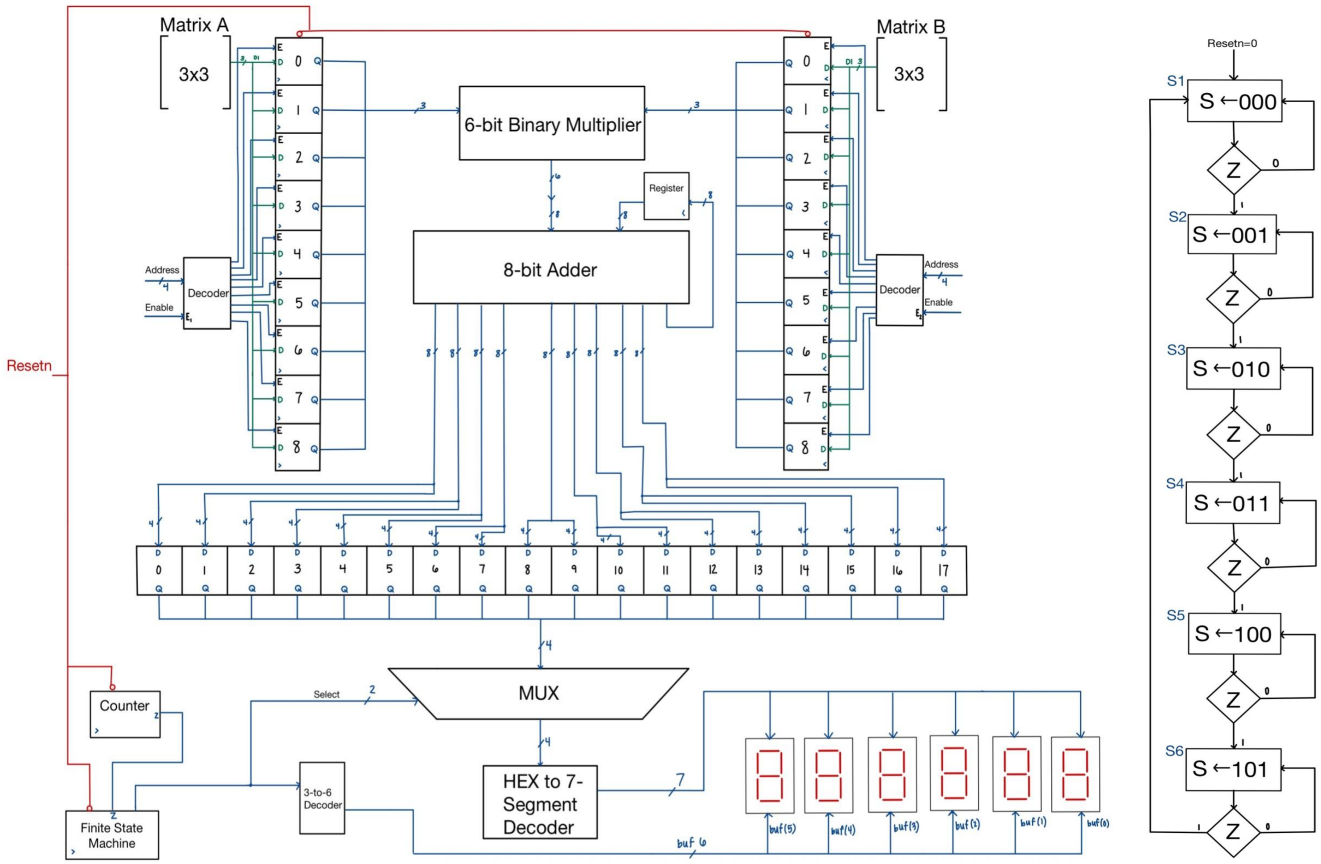
Conclusions

There was a lot of planning involved in the process of creating the 3x3 matrix multiplier, especially when it came to deciding how to input values into the matrix and to display the values of the matrix.

In the end, the 3x3 Matrix Multiplier successfully multiplied two matrices together and displayed each row correctly on the seven-segment displays one at a time. Although there were some challenges faced in the process, the final outcome of the project correctly demonstrated the plan of how the multiplier was to work. The biggest challenge was determining how to display an entire row of the matrix at once. This called for the use of a counter with a finite state machine and 3-to-6 decoder in order to make it work.

References

[1] D. Llamocca. Digital System Design [PowerPoint slides]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.h



Block Diagram