Matrix Multiplication

The architecture of 3x3 Matrix Multiplication for unsigned numbers.

List of Authors (Genbela Lifo, Nour Alnounou, Maryam Yaqo, Maryam Zadoyan)

Electrical and Computer Engineering Department

School of Engineering and Computer Science

Oakland University, Rochester, MI

e-mails: lifo@oakland.edu, nalnounou@oakland.edu, maryamyaqo@oakland.edu, maryamzadoyan@oakland.edu

Abstract- The purpose of this project was to implement the architecture of a three-by-three multiplexer for unsigned numbers through VHDL. This design only utilizes the Nexys A7-100T board and a regular Keyboard. This project requested different coding techniques to complete the architecture of the circuit.

I. Introduction

The goal of this VHDL project is to create and implement a synthesizable matrix multiplier that can calculate 3x3 matrices in size. The report outline shows the strategy of the architectural design, the implementation, and the ultimate outcomes. The architecture of the matrix multiplexer was implemented via VHDL in the Xilinx Vivado Design. We used an external keyboard as the system's input, which is connected through a USB port to the Nexys A7-100T board. The board was programmed to turn the 8-bit input from the keyboard into a 4-bit input to complete the 3x3. After the signal goes through the matrix process it is converted from Hex to a 7-segment decoder which after will display the results on the FPGA board.

II. Methodology

The purpose of this project is to be able to give an input that goes through the architecture of the 3x3 matrix multiplicator in order to display the same number as the input but into the 7-segment display in the Nexys A7-100T board. The methodology of this project consists of the matrix multiplication, block diagram, state diagram, and implementation of FPGA, where all of these parts and the inner steps will complete the structure of the 3x3 matrix multiplication architecture.

A. Matrix Multiplication

The matrix multiplication needed for this project follows the same steps as a normal math matrix multiplication but in this case, signals are replacing the numbers. Being able to multiply matrices is essential for comprehending how to use the circuit's various parts. The first rule is that the first matrix's columns must have the same number of rows as the second matrix's rows in order for matrices to be multiplied. This project focuses on 3x3 matrix multiplication, the example of this matrix using the letter component is shown in Figure 1[2].

$$\begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix} \begin{pmatrix} b11 & b12 & b13 \\ b21 & b22 & b23 \\ b31 & b32 & b33 \end{pmatrix} =$$

$$\begin{pmatrix} a11\,b11 + a12\,b21 + a13\,b31 & a11\,b12 + a12\,b22 + a13\,b32 & a11\,b13 + a12\,b23 + a13\,b33 \\ a21\,b11 + a22\,b21 + a23\,b31 & a21\,b12 + a22\,b22 + a23\,b32 & a21\,b13 + a22\,b23 + a23\,b33 \\ a31\,b11 + a32\,b21 + a33\,b31 & a31\,b12 + a32\,b22 + a33\,b32 & a31\,b13 + a32\,b23 + a33\,b33 \end{pmatrix}$$

Figure 1: Matrix Multiplication

B. State Diagram

This is our Finite state machine. First We start with the first state and after we get the data from the keyboard we will receive this done. The first enable for the decoder will be one and the address will be zero (five bits), but the enable for the D-flip flop will be one and the done will be zero. It is zero because the data is not ready yet. Then it will go to the second state and we will get one for the enabled decoder and one for the address. From state two we will go to state three and get one for the enabled decoder and the address will be one. From state three it will go to the fourth state. We will get one for the enabled decoder and two for the address. Then it will go to the fifth state and we will get the enable decoder to be one and the address will be three. From this state, we will go to the sixth state and will get one for the enabled decoder and four for the address. Then we will go to the next stage which is the seventh state and will get one for the enabled decoder and five for the address. From the seventh state, we will go to the eighth state and will get one for the enabled decoder and six for the address. Next, we will go to the ninth state and will get one for the enabled decoder and seven for the address. From state nine we will go to the tenth state and will get one for the enabled decoder and eight for the address. Then we will go to the eleventh state and will get one for the enabled decoder and nine for the address. From the eleventh state, we will go to the twelfth state and will get one for the enabled decoder and ten for the address. From that state, we will go to the thirteenth state and get one for the enabled decoder and eleven for the address. From that state, we will go to the fourteenth state. We will get one for the decoder and twelve for the address. From that state, we will go to state fifteen and will get one for the enabled decoder and thirteen for the address. From state fifteen we will go to state sixteen and will get one for the enabled decoder and fourteen for the address. From state sixteen we will go to state seventeenth and get one for the enabled decoder and fifteen for the address. From state seventeen we will go to state eighteen and get one for the to enable and sixteen for the address. From state eighteen we will go to the last state which is state nineteen. We will get one for the enabled D-flip flop and will get one for the done which means the data is ready now. In the end, we will go back to the first state.

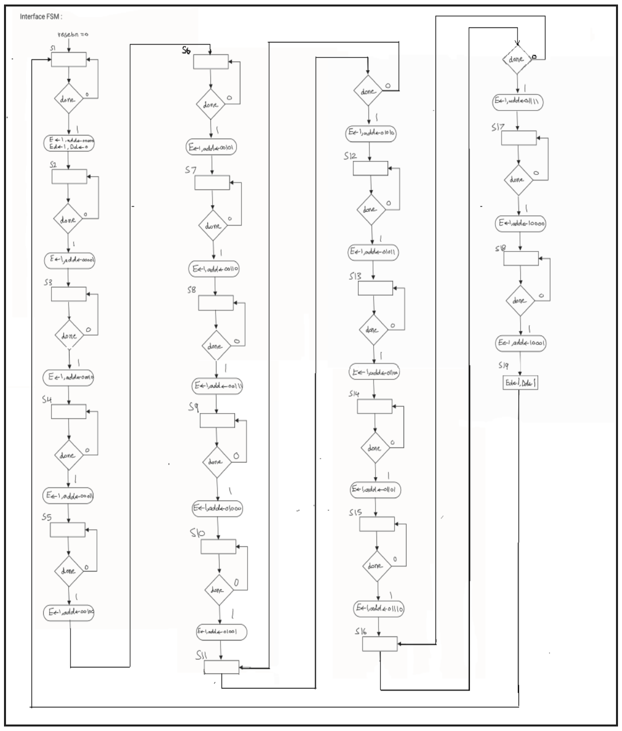


Figure 2: FSM Diagram

The first part of the circuit starts with the keyboard which provides the input. The data will move to the first decoder which uses axi code to convert the data to 4 bits. This data will be stored in the registers. After Done is received from the keyboard the FSM will make sure that the right data will be represented. So it will display the results one by one and this is stored in the second decoder and from the decoder to the register. After this entire process is done the d flip flop will turn the LED on to show that the data is received. There are a total of 18 elements stored in the register, 9 per each matrix and all of them are 4 bits each. After the matrix multiplication is complete now each element will be 8-bit. This data now will be entering the Mux which controls the output we see from the matrix multiplication using switches. The last part of the circuit is the serializer [1] which includes the necessary components that the board knows to output the results in the two 7-segment displays.

C. Circuit Design

The picture below shows the complete circuit for this project. It is quite a complicated circuit that requires many components like two Decoders, a Register, Multipliers, Muxes, FSM, and more.

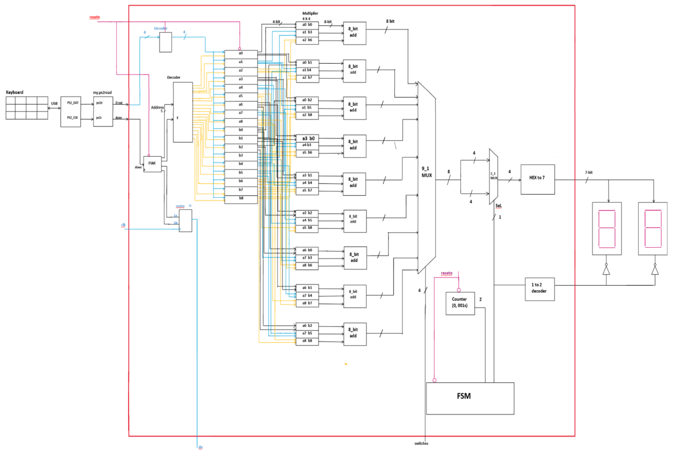
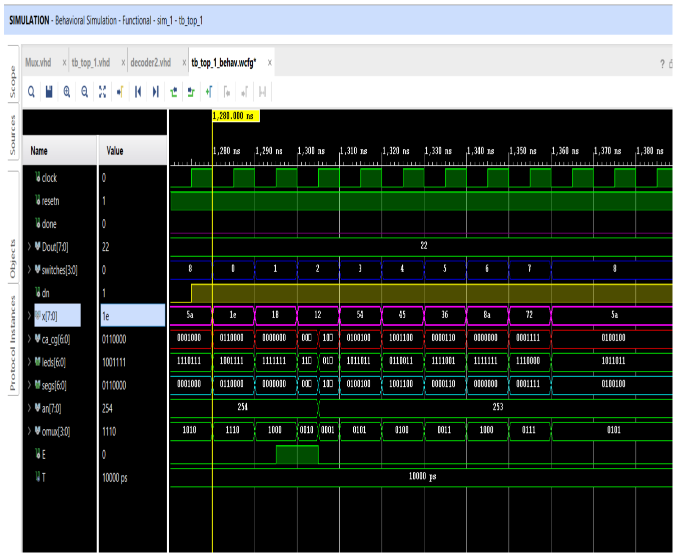


Figure 3: Circuit Diagram

III. Experimental Setup

We worked in the Vivado VHDL programming environment to build the code as well as simulated it in order to make sure that all the components of the circuit were working as expected. Separate source files were created for each component and later on combined on the top file to complete all the necessary connections between components.


SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_top_1

Name	Value
clock	0
resetn	1
done	0
Dout[7:0]	22
switches[3:0]	0
dn	1
x[7:0]	1e
ca_cg[6:0]	0110000
leds[6:0]	1001111
segs[6:0]	0110000
an[7:0]	254
omux[3:0]	1110
E	0
T	10000 ps

Figure 4: Simulation

IV. Results

Overall the board functioned as desired and the right results were displayed. The keyboard was able to provide the inputs and the LED turned on after all the data was received and later saved in the register. Very quickly as expected the 7-segment display showed the right results after the switches selected which element of the results to display.

Conclusions

This was a successful project. We were able to complete all the basic requirements. As in every case, there is always room for improvement or challenging steps that can be added to the project for example the ability to work with signed numbers or improving the concept of the project so the user doesn't need to choose which displays they want to see by using the witches on the board but maybe the keyboard as well. The options are endless but overall this project was challenging and it showed vicarious results.

References

[1] D.Llamocca's Unit 7 serializer example project. https://moodle.oakland.edu/pluginfile.php/7846520/mod_resource/content/17/Notes%20-%20Unit%207.pdf

[2] Threads: Basic Theory and Libraries. (n.d.). Retrieved December 4, 2022, from users.cs.cf.ac.uk website: https://users.cs.cf.ac.uk/Dave.Marshall/C/node29.html

[3] Laxman, S., et al. "FPGA implementation of different multiplier architectures." *International Journal of Emerging Technology and Advanced Engineering* 2.6 (2012): 292-295.

[4] Bhairannawar, Satish S., et al. "Efficient FPGA Based Matrix Multiplication Using Mux and Vedic Multiplier." *International Journal of Computers and technology* 12.5 (2014).