# Matrix Multiplication using FPGA

List of Authors (Junbang Chen, David Huang, Ethan Postma)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: junbangchen@oakland.edu, dhuang2@oakland.edu, ethanpostma@oakland.edu

***Abstract*—We attempted to implement a 2x2 and 3x3 matrix multiplication circuit into an Artix A-7 FPGA. The user will be able to input numbers into each cell of the input matrix and the circuit will produce a result.**

## I. Introduction

The purpose of this project is to make a circuit capable of doing 2x2 and 3x3 matrix multiplication using unsigned numbers. Matrices are rectangular arrays of numbers or expressions arranged into cells in a certain row and column.

Matrix multiplication seemed like an interesting project because it wouldn't be difficult to do in a standard programming language, even ones without a GUI. However, implementing the design into an FPGA proved to be a challenge as our Nexys A7-50T can only handle one input/output at a time. For our project, we used many components from class like a finite state machine, decoders, registers, and a serializer. One thing we used that we had to do additional research for were the libraries, "IEEE.std_logic_arith.all" and "IEEE.std_logic_unsigned.all" which greatly simplified our code allowing us to do arithmetic operations without needing to using adder and multiplier circuits.

## II. Methodology

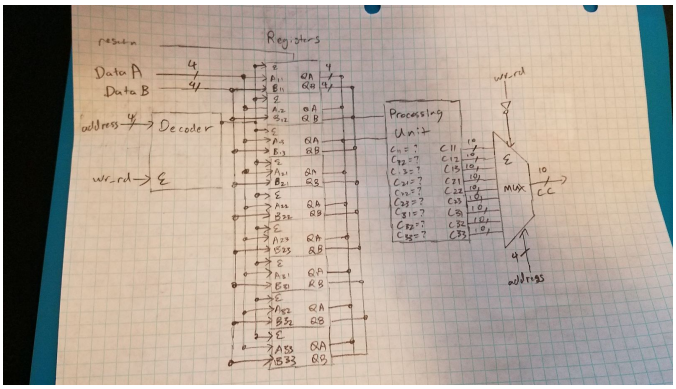
We had to figure out how to design our circuit architecture before we could start coding our components. Our setup initially used an 18 to 1 decoder that fed data into 18 registers. The data for those 18 registers would then be fed into a processing top file to do the mathematical operations required. Once the product matrix was calculated it would be fed into a 9 to 1 multiplexer, controlled by the address bus and display our output to LEDs. Initially we want to use 18 registers to store data, one cell for each cell in the 3 by 3 input matrices, A and B. This would have required us to use a 5-bit address bus to control 18 addresses with our decoder. We managed to simplify the circuit significantly by using 9 registers that each stored two data values, and having two data input buses. This meant that our address bus would correspond to the same cells in both our input and output matrices. We also used the IEEE libraries for arithmetic and unsigned numbers eliminating the need for adders and multipliers in our circuit. We were also initially constrained to 4-bit inputs and 8-bit outputs in our due to the IEEE library for arithmetic. This however would not work as an input of 15 (1111) into every cell of our input matrices would result in each cell of our output matrix having a value of 675 (1010100011), a 10-bit number. To solve this issue in our code, we changed each input "XXXX" into '0'&"XXXX", so that the multiplication of two 5-bit numbers would result in a 10-bit result, accounting for our worst case scenario.

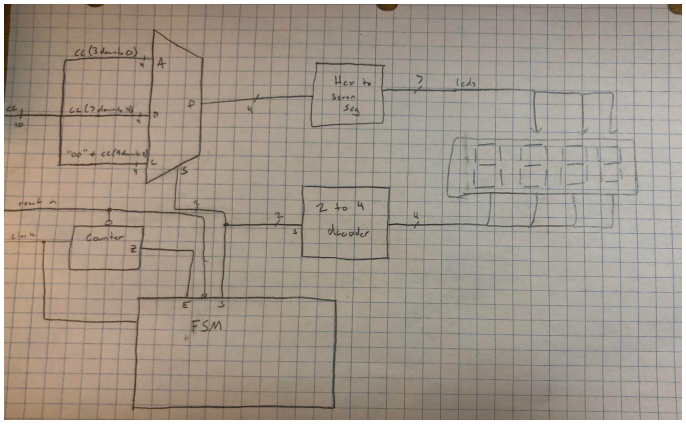
### *A. Block Diagram*

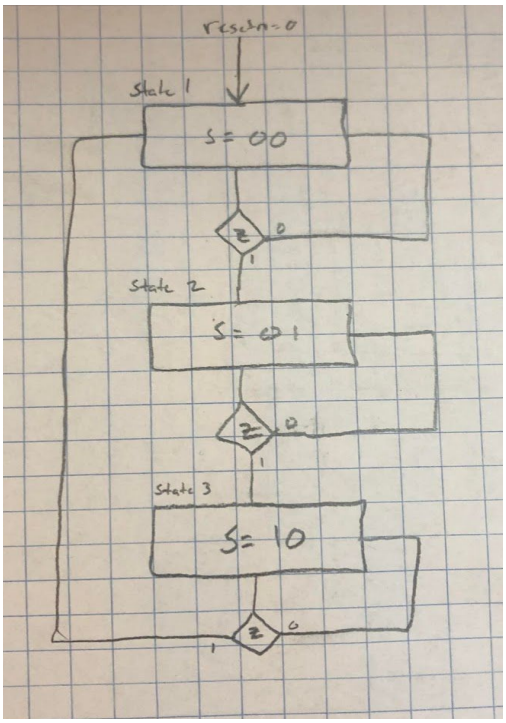
For our block we have a decoder reading an address telling it to enable 1 of 9 registers to store all the possible numbers we may need for 2x2 or 3x3 matrix multiplication. That decoder is enabled by a write read switch that writes when wr_rd = '1'. Each register stored two 4-bit data values, one for data A and the other for data B. Data entry was controlled using a 9-to-1 decoder, with a 4-bit address bus. Two 4-bit numbers, dataA and dataB were imputed simultaneously. From there we take all the values stored in the registers and put them into our processing unit (PU) to handle the arithmetic operations of matrix multiplication. Our PU had 18 4-bit inputs for each cell of the input matrices and 9 10-bit outputs for each cell of the output matrix. The 9-output values were fed into a 9 to 1 MUX controlled by the 4-bit address bus and an enable of ~(wr_rd), so it is enabled when the decoder is disabled and vice versa.

The output of the MUX, CC, then gets split up into 3 bit signals, CC(3 downto 0), CC(7 downto 4), and "00"&CC(10 downto 9), and those signals are fed into a HEX to seven segment serializer that outputs to an LED display. The serializer is controlled by a finite state machine that cycles between three states to display only 1 LED seven segment display. In order to enable 2 by 2 mode, the third column and third row of each input matrix should be zeros

so then the third column and third row of the output matrix are also zero.




## III. Experimental Setup

This project was created and programmed in Xilinx Vivado Webpack 2018.3. The FPGA used for our project was a Nexys A7-50T and all code is available upon request. We also used code provided by Dr. Llamocca for the registers, serializer and finite state machine.

## IV. Results

The result of this project was a working matrix multiplier for both 3 by 3 and 2 by 2 matrices. It would have been possible to do the circuit without a FSM if the output was displayed using LEDs. We learned about using IEEE libraries to simplify our code and learned about some of the quirks to said libraries. The hardest and most rewarding part of the project was designing the interface given the limited amount of both input and output methods.

## Conclusions

Our project went smoothly for the most part and we got a working circuit. We would only recommend using this circuit for actual matrix multiplication only to the most adventurous of masochists. It would be preferable and simpler to do matrix multiplication using a standard programming language or even a website with a matrix calculator as those would allow for the user to put in all the inputs simultaneously and see all the results at once. Just use the computer programming the FPGA to do matrix multiplication. We had ideas of implementing a hardware switch for 2 by 2 mode, but that would have resulted in more complex code.

## References


[1] "Fundamentals of Digital Logic with VHDL Design" THIRD EDITION, Stephen Brown and Zvonko Vranesic,Department of Electrical and Computer Engineering University of Toronto

[2] "Matrix Multiplication." *From Wolfram MathWorld*, mathworld.wolfram.com/MatrixMultiplication.html

[3] Bryan J. Mealy, James T. Mealy, Digital McLogic Design, Free Range Factory, 2012.

[4] S. Brown, Z. Vranesic, Fundamentals of Digital Logic with VHDL Design, 3rd ed., McGraw Hill, 2009.

[5] "Standard VHDL Packages," Standard VHDL Packages. *From University of Maryland, Baltimore County,* https://www.csee.umbc.edu/portal/help/VHDL/stdpkg.html.