3x3 Matrix Multiplier using a NEXYS-A7-50T

List of Authors (Nishchay Kulkarni, Hannah, Theo)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: nkulkarni2@oakland.edu, tzink@oakland.edu, hegeisler@oakland.edu

***Abstract*— For this final project we will try to create a 3x3 or a 2x2 matrix multiplier. This logic will be implemented on the Nexys A7 FPGA board. The inputs will be given via the 16 switches that are present on the board.**

I. Introduction

The goal of this project is to create a 2x2 or a 3x3 matrix multiplier on the FPGA board. In mathematics, a matrix is a rectangular array of numbers. The application of a matrix can be seen when using the KVL or KCL representations of a circuit when trying to find the voltages or the currents. The transformation of equations to a matrix can greatly simplify calculations such as these in higher levels of mathematics.

When calculating values from equations with many variables, a matrix can be used to simplify and condense each line in a quick and concise manner. It is due to the amount of variables that many matrices are composed of arrays. By applying this concept to a FPGA board, we allow for a simpler tool which can compute the solution to the matrix much faster than if calculations were carried out by hand.

Rudimentary concepts taken from EGR 2700 Laboratory assignments can be applied to generate the multiplication of two 3x3 matrices.

II. Methodology

The design must be based off of a real matrix multiplier and the functionality should match that of a real 3x3 or 2x2 matrix calculation. By implementing binary addition and multiplication the dot product can be executed between each matrix to generate the resulting 3x3 product. Each calculation should follow as expressed in the image below:

$$\begin{bmatrix} F1 & F2 & F3 \\ G1 & G2 & G3 \\ H1 & H2 & H3 \end{bmatrix} \begin{bmatrix} X1 & X2 & X3 \\ Y1 & Y2 & Y3 \\ Z1 & Z2 & Z3 \end{bmatrix} \Rightarrow \begin{bmatrix} T1 & T2 & T3 \\ U1 & U2 & U3 \\ W1 & W2 & W3 \end{bmatrix}$$

T1 = F1X1 + G1X2 + H1X3 T2 = F2X1 + G2X2 + H2X3 T3 = F3X1 + G3X2 + H3X3

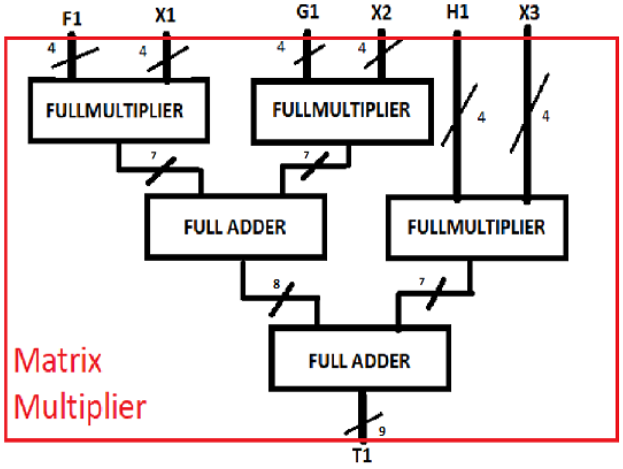
U1 = F1Y1 + G1Y2 + H1Y3 U2 = F2Y1 + G2Y2 + H2Y3 U3 = F3Y1 + G3Y2 + H3Y3

W1 = F1Z1 + G1Z2 + H1Z3 W2 = F2Z1 + G2Z2 + H2Z3 W3 = F3Z1 + G3Z2 + H3Z3

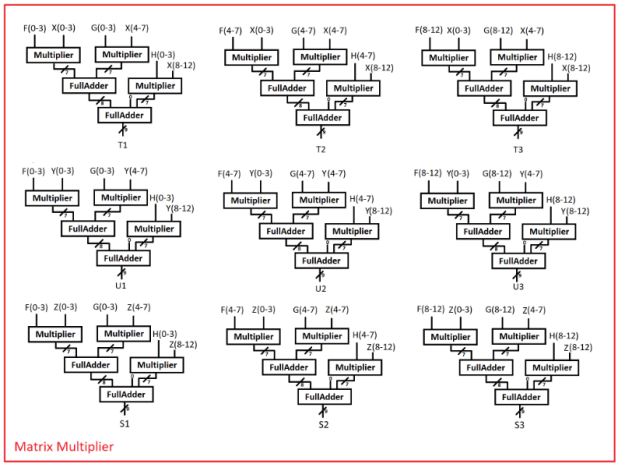
Generally speaking, each row must follow a clearly cut guideline for multiplication and addition. This guide made it relatively simple to structurally design in Vivado using full adders and multipliers generated in previous laboratory assignments required by ECE 2700.

III. Experimental Setup

When using adders and multipliers with few variables the concept of assigning and port mapping is relatively simple. Each calculation follows a waterfall type technique where each variable starts in one section, adds to the next, and finishes in the same spot. In all, the calculations for a single value in the resulting matrix followed this design:



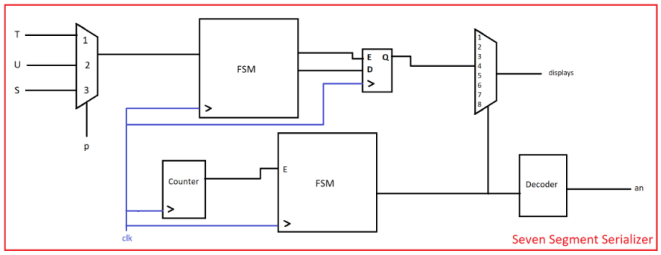
While appearing simple in nature, this calculation would have to be carried out nine different times and requires meticulous planning. Each signal coming from each multiplier would need to be individually designated and connected. Every expression would have to accommodate for overflow. All in all, there could be no umbrella signal as using it for more than one variable would generate mixed signals and incorrect results. After quite a few hours of painstaking port mapping, and using over one-hundred-and-twenty carry out signals the structure of the matrix multiplier was finished following the template provided above and stacked together nine times as shown below:



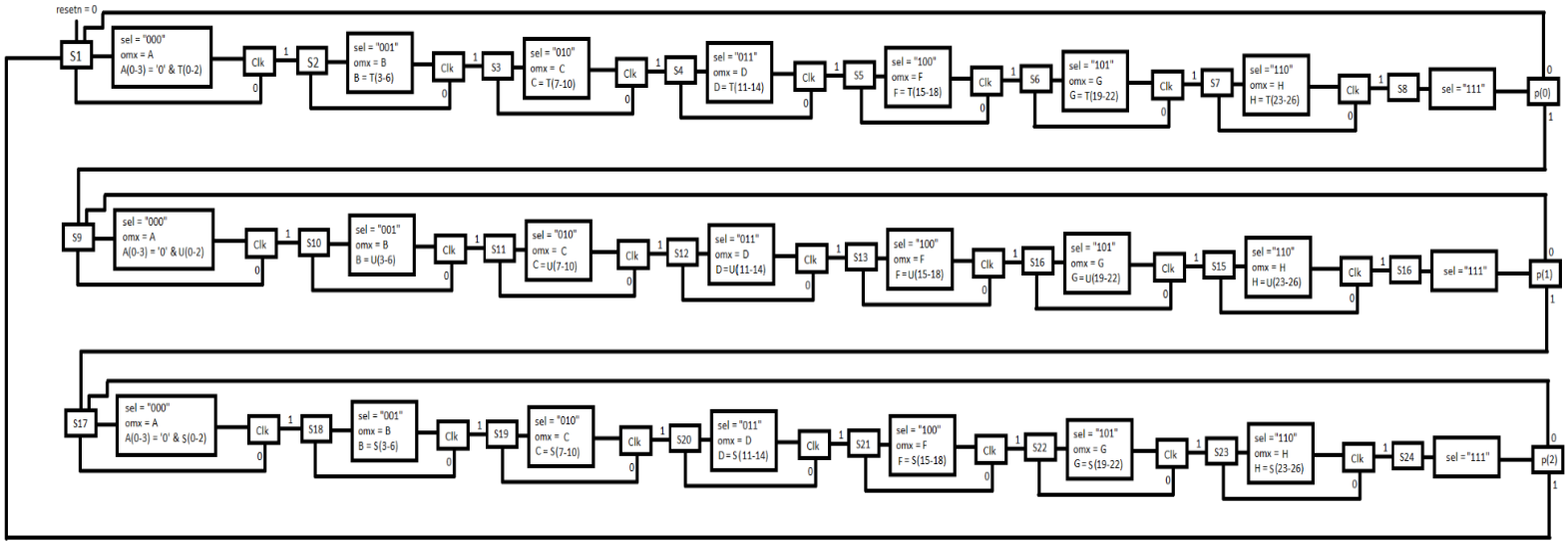
The next hurdle became saving each input and output. Something that should have been considered at the start if not for the overbearing display problem. Since the amount of switches provided on the Nexys-A7-50T would not accommodate for the eighteen four-bit inputs. However, it is possible to use twelve switches for the inputs, and three switches to toggle between each input address. Using a ram style memory system it is possible to use two switches as addresses and the following as a stored number. Then all that's left is to split the twelve-bit input into four-bit segments giving us the three numbers needed for each row of the matrix.

Lastly, displaying each output. Given the amount of seven segment displays on the Nexys-A7-50T board, a nine variable matrix could not be displayed even if presented in single digits. However, one equation could be displayed in one instance, followed by the next, and so forth. So the problem becomes displaying a different value on each seven segment display. The simplest idea of implementing this concept is to use one display at a time for each value in a small enough amount of time such that looping the desired grouping of variables for an allotted time segment makes it appear that each display is lit up at the same time. It's in this way that we can get each equation to display itself at once, then after a few seconds display the next, and so on when following the same design for each.

For this, a twenty-four state finite state machine (FSM) was used. The three rows were fed in through a multiplexor which, with the use of buttons, chose when each was displayed. Each of the inputs were broken up into four bit pieces then converted to hex to upload onto a seven segment display. Given the twenty-six digits each had the four bit segments were fed into a seven input multiplexor. The selector for that multiplexor was shared by a decoder which turned on the displays in a quick enough succession that it appeared as though each was used separately. Each state passed at the tick of the clock and showed a different four bit hex number on the specified display. Based on which button was physically pressed on the board the displays would cycle through the output equations showing the final result. The design for this looked as follows:



However, the code was much more difficult. To create this an FSM needed to be created, and as previously mentioned it needed twenty-four different states. After a few ideas, this is the final ASM chart that was decided upon for this FSM model.



Conclusions

This project seems like a great undertaking given the hardware and the complexity of the calculations. However, when taking a closer look it can be seen that everything can be broken down into smaller bits that are much easier to tackle than handling it all at once.

The most difficult aspect of this design was to create the FSM controlling the seven segment displays. All said and done, the circuit functions as desired and has simplified the calculations to that of a few switches and buttons.

The final design looked as follows: