THE AMERICAN UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

CSCE 2301/230 – Digital Design I – Spring 2023
Project 2: Sequential Signed Multiplier

Yehia Ragab - 900204888
Mostafa El Shamy - 900204869
Salma El sayed

Instructor: Dr Mohamed Shalan

Table of Contents

Introduction

This project displays the functionality of the 8 bit signed multiplier on a Artix 7 FPGA Basys 3 FPGA board. In order to attain this functionality we have gone to three stages of designs. The first stage is creating a block diagram that has a clear logic functionality of the 8 bit unsigned multiplier. The second stage included a logisim circuit that was created from the block diagram from the first stage which was also for the 8 bit unsigned multiplier. Finally, the third stage was writing the verilog code that was used to program the 8 bit signed multiplier on the FPGA board. In the first two stages it made it clear on how the verilog code must be structured in order to create the 8 bit signed multiplier which made it an easier process in developing the verilog code. In this report the functionality of the three stages will be clearly explained. In order to create the 8 bit signed multiplier we used the algorithm that we learned in the lecturer which utilizes the shift add algorithm to multiply both 8 bits. The switches used were (SW7-SW0) that was for the multiplier and switches (SW15-SW8) for the Multiplicand. One LED was used to indicate that the multiplication ended. The Artix 7 FPGA Basys 3 FPGA board has only 4 seven segment displays, and we needed to use one seven segment to display a negative if the output is negative and will output nothing on the seven segment if the output of the multiplication is positive. This meant that there were only 3 seven segment displays left to output the 5 digit output of the multiplication, which made it necessary to have two push buttons, one button to shift left and the other to shift right. The third button was used to start the multiplication of the two 8 bit-binary numbers that were identified by the switches: the MSB for the first 8 bit-binary number is SW15 and the MSB for the second 8 bit-binary number is SW7. The verilog sources were neatly written and indented to make the code more clear to the user. Github was used in every stage of the 3 stages which shows our proper workflow, repo structure, and a readme file.

Block Diagram

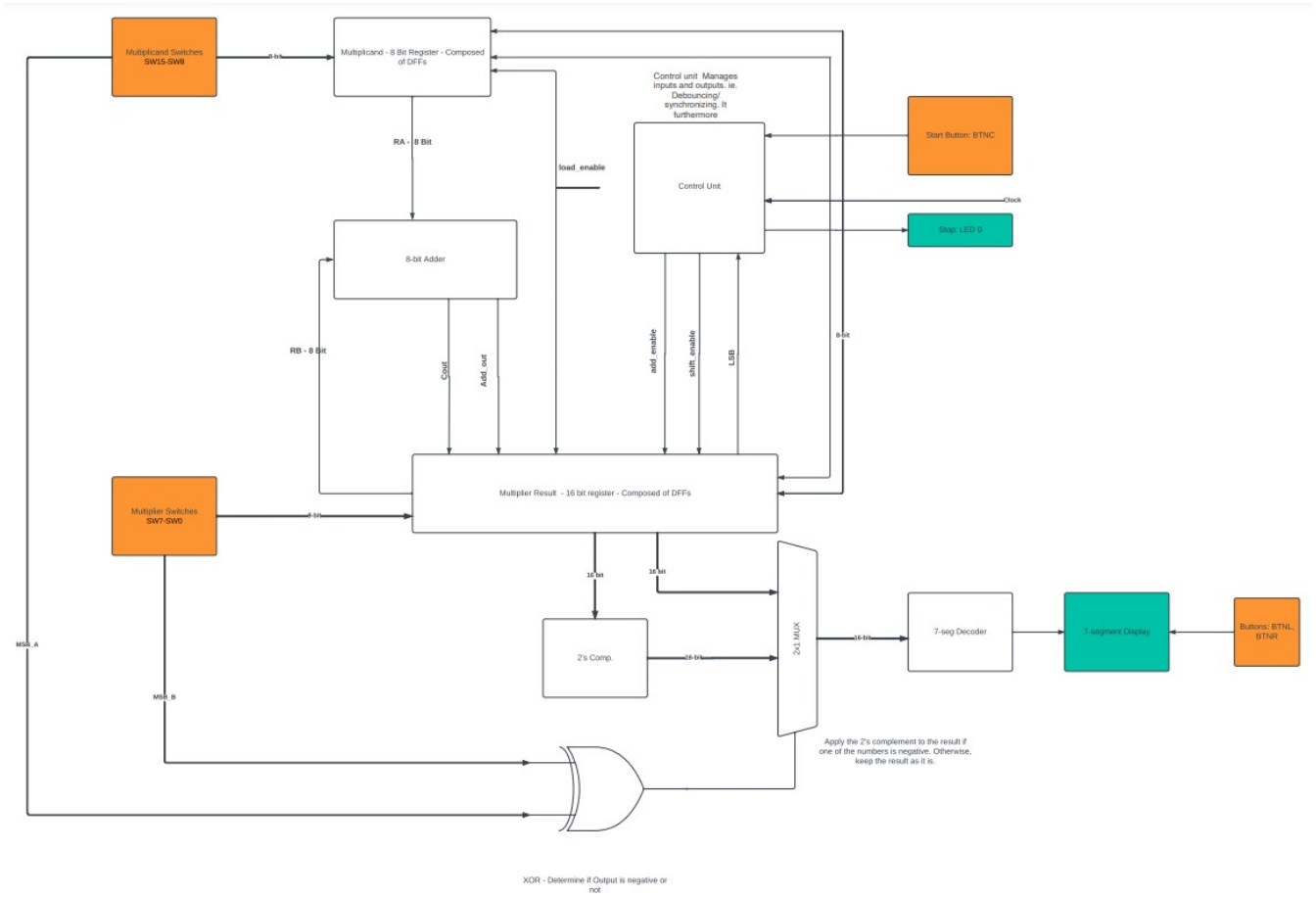


Figure 1

The block diagram will show the main functions that were used as blocks all connected by wires with their respective bits written on them. In figure 1 uses 2 inputs: the multiplier 8 bits and multiplicand 8 bits. The multiplicand 8 bits is inserted in an 8 bit register created by flip flops which will then be inserted with the multiplier in the multiplication function which includes the 8 bit adder and the 16 bit register which will be created by flip flops. The output will then be converted into its two's complement and then will be inserted to the seven segment decoder and displayed on the seven segment display if it is negative, and if its positive it will be inserted to the seven segment decoder then displayed on the 7 segment. Identifying if the output is positive or negative will be done by inserting the MSB of the multiplicand and the multiplier in an XOR gate which will then be a selector in the 2 by 1 mux. The control unit will be in access with the 16 bit register to do 3 functions: scroll left, scroll right, and a push button to start the multiplication. The indicator in the controller is the LED that switches on when multiplication is

done. The block diagram displays the main functions as a whole which was used as a guid path to create the logisim and verilog sections of the project.

Logisim Overview

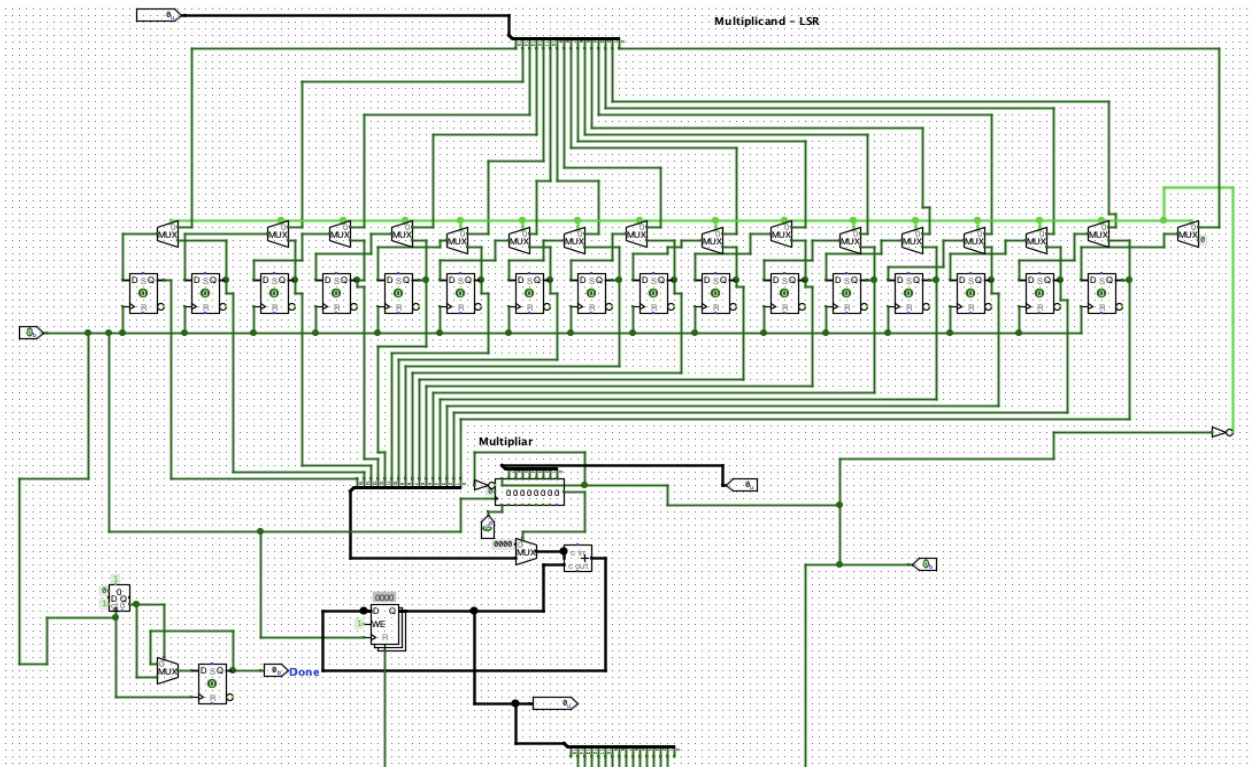


Figure 2

Logisim followed the same plan as the block diagram from the previous section. The inputs are the multiplier and the multiplicand which will be inserted by the switches in the verilog section, which were inserted in their respective MUX and D flip flop which mimics the multiplication algorithm that will be used in the verilog code. Then the start of the multiplication is connected in the middle next the output of the multiplier which signals the start of the shifting and adding. There is also a done functionality that turns one after the shifting and adding is finished which is represented on the bottom left corner. There were 16 MUX and 16 D flip flops to multiply the 8 bit numbers. The use of logisim was to mimic the functionality same as the hardware of the FPGA board. This logisim functionality was produced by the clock which

connected to the start of multiplication and connected to the done function that outputs 1 when the multiplication is done which mimics the LED that will be used on the FPGA in the verilog section.

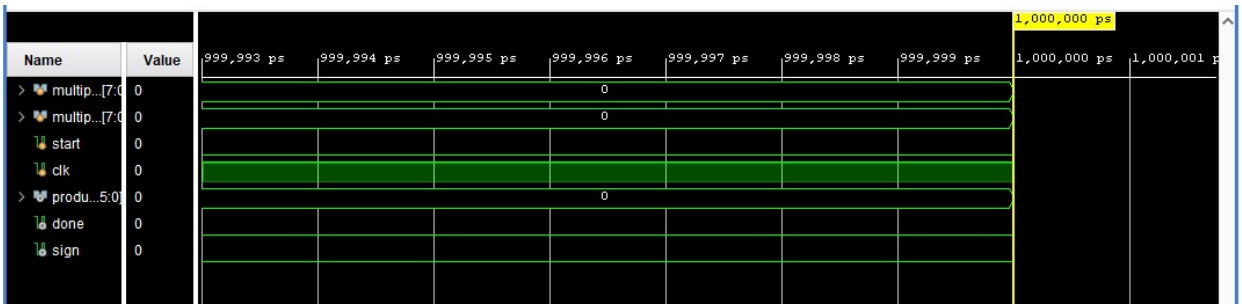
Verilog Overview

The verilog code starts by having a push button detector that will use the middle push button on the FPGA board. The inputs are two 8 bits signed numbers that will be used from the switches. The switch SW15 is MSB for the multiplicand and its LSB is SW8. The switch SW7 is the MSB for the multiplier and SW0 is the LSB. This push button module uses a clock divider that outputs to the debouncer which will then be detecting an edge of the clock to start the multiplication process. The edge detector is implemented using a finite state machine as required. The multiplication implementation starts by XOR the MSB of the multiplicand with the multiplier which is used to indicate the sign of the output of the multiplication. Then if the MSB of any of the numbers that will be multiplied is negative then a two's complement will be used. Then after checking whether a two's complement is needed we use the shift and add algorithm to produce the output on the FPGA and the LD0 will be turned on indicating that the multiplication is finished. The FPGA has the most left seven segment display as the sign for the output when the number is negative then a negative sign will appear, and when positive nothing will appear on the seven segment display. The other three seven segment displays will display three digits from the five digit output. Then we used a button to scroll left and a button to scroll right in order to be able to display all five numbers part by part. The shifting buttons were used by shifting all digits to the side you are scrolling at in order to reveal the elements that have been scrolled too. The seven segment verilog code implementation first started by using dividing and modulus to extract each digit in the ones,tens, hendreds, one thousandth and ten thousandth. Then we assigned the three digits to their respective seven segment displays according to which state the scrolling is pointed to. Then by having the BCD we then output the digit on the seven segment display.

Validation Activities

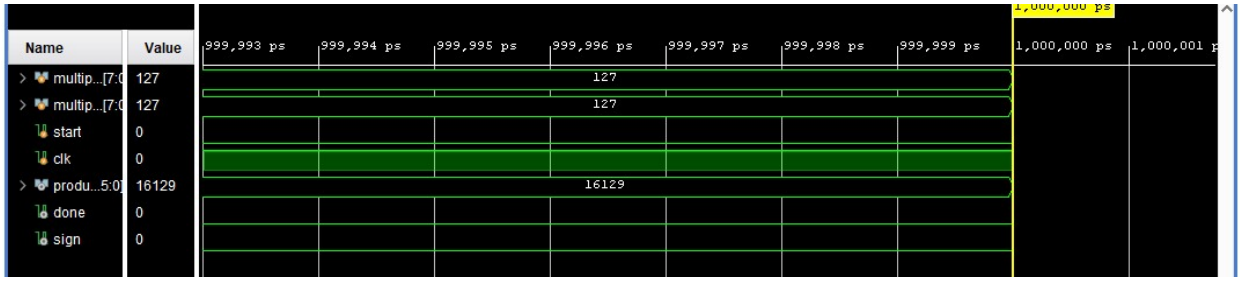
The following test cases where chose, as they demonstrate the functionality of the multiplier on edge cases.

Case 1. Zero edge-case



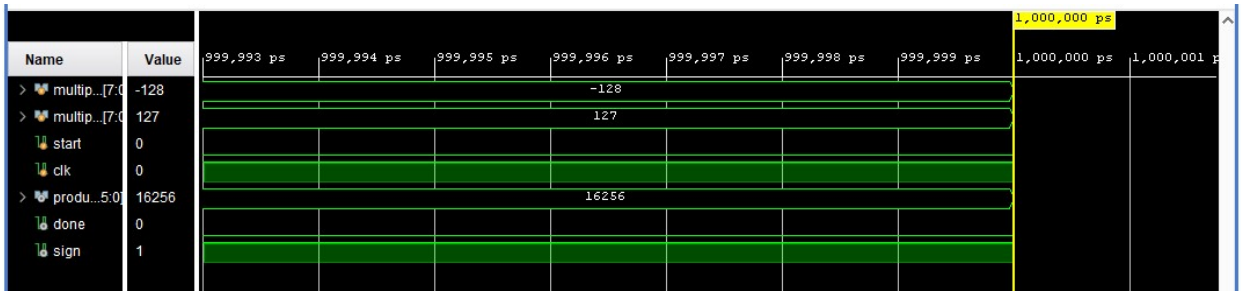
Multiplicand =0, Multiplier= 0 then an output =0, sign= 0. Which correctly outputs the multiplication.

Case 2. Positive x Positive maximum edge case



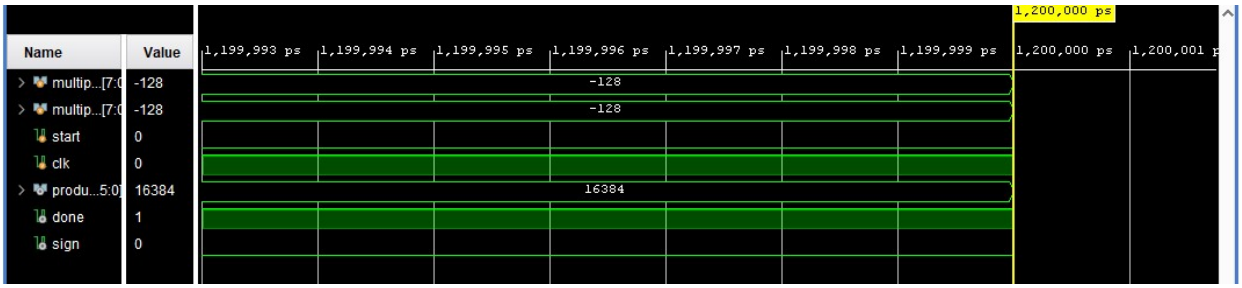
Multiplicand =127, Multiplier= 127 then an output =16129, sign= 0.
This case checks on whether the multiplication algorithm that was created by the shifting and adding technique that was taken in the lecture. This correctly outputs the multiplication.

Case 3. Positive x negative maximum edge case



Multiplicand =-128, Multiplier= 127 then an output =16256, sign= 1.
This test case tests the 2 complements code used and whether the code for the sign of the output will be negative or not.
This correctly outputs the sign of the XOR of the MSB of both numbers output the variable sign as 1.

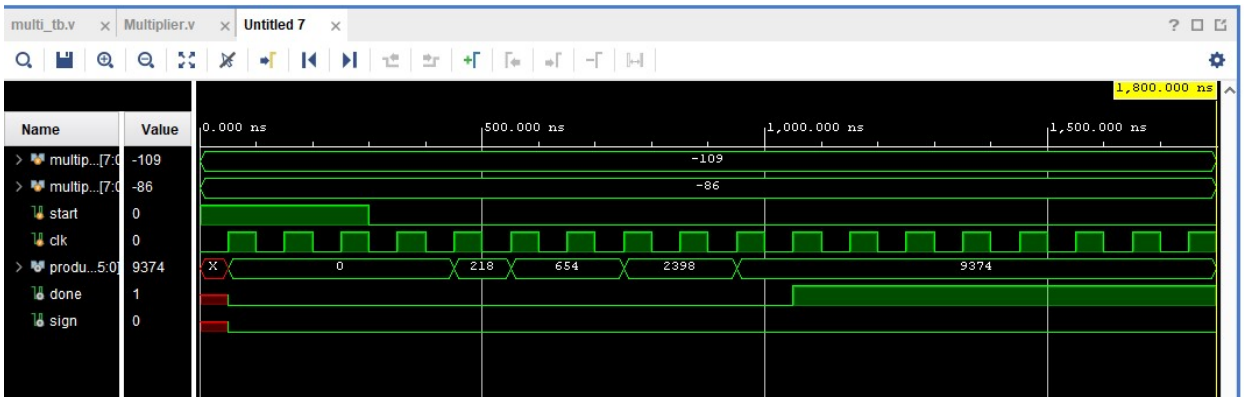
Case 4. Negative x negative maximum edge case



Multiplicand =-128, Multiplier= -128 then an output =16384, sign= 0.
This test case tests the use of two's complement on both the multiplicand and the multiplier.

This correctly outputs the sign of the XOR of the MSB of both numbers output the variable sign as 0.

Case 5. Standard test case



Multiplicand =-109, Multiplier= -86 then an output =9374, sign= 0. Which correctly outputs the multiplication as the XOR of the MSB of both numbers outputs 0.

Acknowledgements and Implementation issues

Throughout the project we have faced several problems that we were able to overcome. Designing the BCD to be able to extract all five digits had occurred to be wrong many times. This was a cause of knowing what is the best method to this which was by using division and modulus operators to be able to extract all 5 digits. The shifting algorithm involved tedious attempts of using different approaches to do it. However, we have been able to overcome this challenge. In this project the use of creating the block diagram and the logisim as a plan to be able to use them as a guidepath to be able to write the verilog code helped in having a clear vision on what will be used to finish the project. The block diagram helped in formulating the vision on what we thought how the components can be used together for this task. The logisim stage helped us test the block diagram's plan on whether it will be correct. Logisium was used as a software with capabilities mimicking how the 8 bit signed multiplication process will work on the hardware FPGA. This step was a breakthrough in decreasing the time we needed to finish the verilog task as we have already built our clear path on how we will formulate the verilog task.