**Yahya Alhinai**

EE 4301-LAB

Tuesday

June 14, 2018

LAB 1

**VERILOG DESIGN ENTRY, SYNTHESIS, AND BEHAVIORAL SIMULATION**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab was to be exposed to Xilinx Vivado. Using Verilog to describe logic gates, we design a simple circuit. We have learned Synthesis, and Simulation the created circuit. As well as we have learned how to do simple behavioral simulation of Verilog designs.

**DISCUSSION OF RESULTS:**

We wrote a Verilog code to model an 8-bit full adder using 8 modules compute the results of two 8-bits numbers. The synthesis of the our designed was successful. The 8-bit full adder schematic generated by Verilog code is shown below (figure 1) As well as the logic design for the building block of the full adder modular (figure 2).

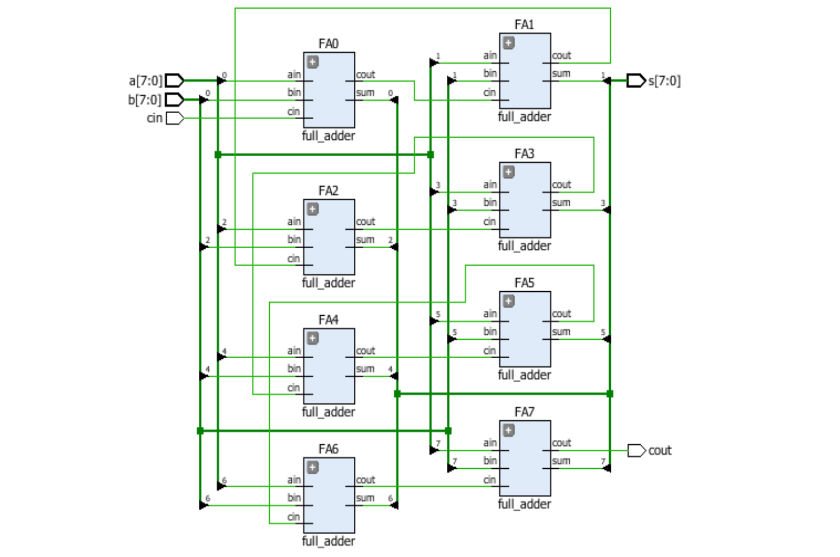


Figure 1: 8-bit full adder schematic generated by Verilog code

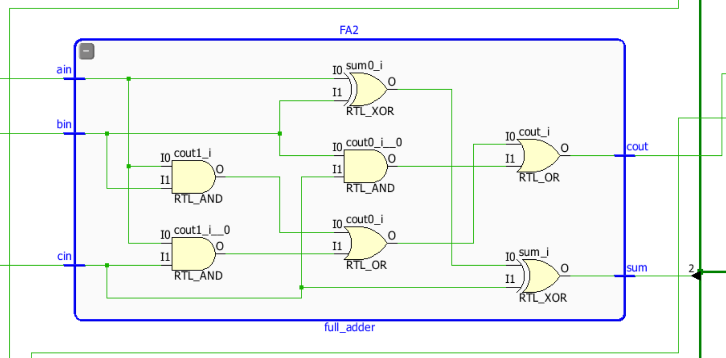


Figure 2: logic design for the building block of a single full adder modular

After making sure that the circuit has no syntax error and it working it meant to be, behavioral simulation of Verilog designs has been made. Different test cases had been simulating to make sure the circuit is fully functioning including overflow triggering. For instance, (figure 3) shows an example of adding A = 8’hF0 with B = 8’h0F and Cin = 1’h01 to give a result of S = 8’h00 and Cout = 1’h01 which indicate an overflow have had happened. All of the cases have behaved as it was predicted previously.

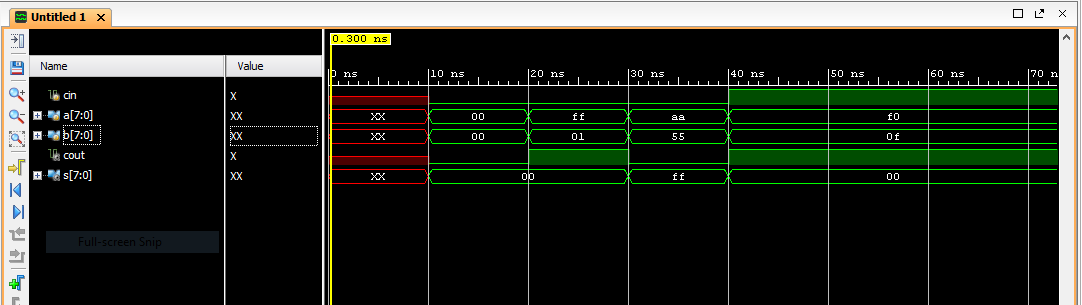


Figure 3: simulated results for the 8-bits full adder

**SUMMARY OR CONCLUSION:**

All of the predicted results have been met. No difficulties have been faced conducting the 8-bits full adder. There are no prior improvements that can be made.

**SOURCE CODE:**

The source code that have been created for this and it had been attached separately. The attached files are **full adder modular code**, **8-bits full adder code**, and **a** **testbench code** to verify the functionally and the behavior of 8-bit full adder.

Wednesday

June 16, 2018

LAB 2

**IMPLEMENTATION AND TIMING ANALYSIS**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab was to test different implementation and timing analysis within Xilinx Vivado’s environment of the circuit that we have created in the previous lab. We have tried different implementation to test different optimization to the same circuit. We have competed Vivado’s default implementation with area explore implementation which have minimize the gates the where used at the expense of power consumption.

**DISCUSSION OF RESULTS:**

This lab was a continues of what we have started last lab by adding the constrain file and try different optimization of circuit’s implementation. The first implementation was Vivado’s default. The schematic design on FPGA basys 3 is shown below (figure 1).

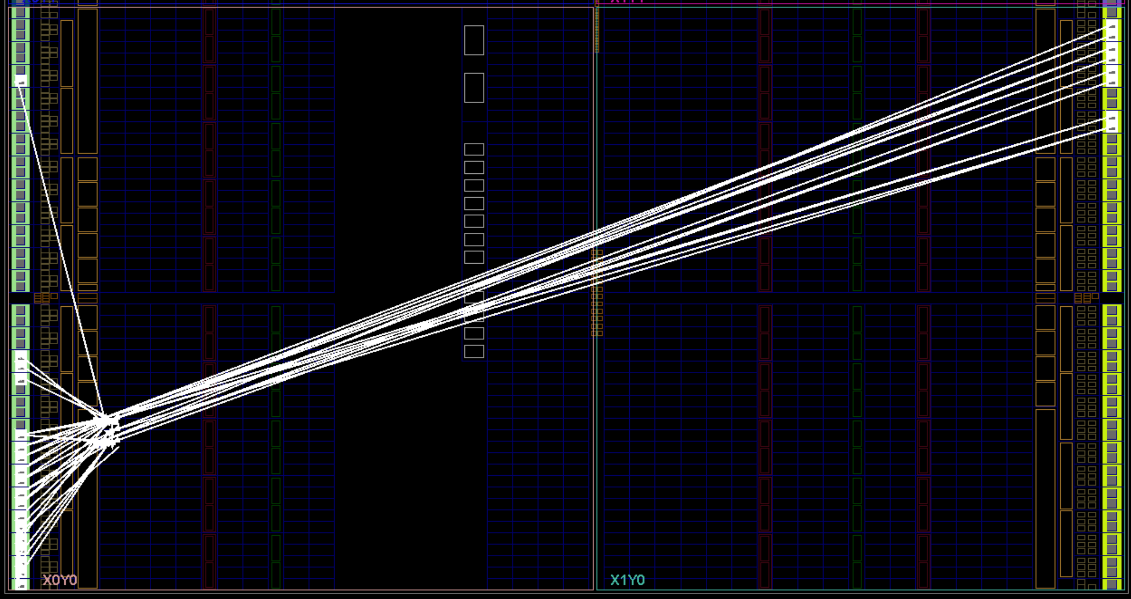


Figure 1: FPGA basys 3 schematic design

Afterwards, area explore implementation had been conducted to mark the difference it and Vivado’s default. The two implementations were exactly the same in terms of power consumption and the number of gates used to build up the circuits. This is might be because of the fact that the circuit the was conducted was too simple to be optimized further; therefore, no noticeable difference has been spotted. The power estimation for Vivado’s default implementation is shown (figure 2).

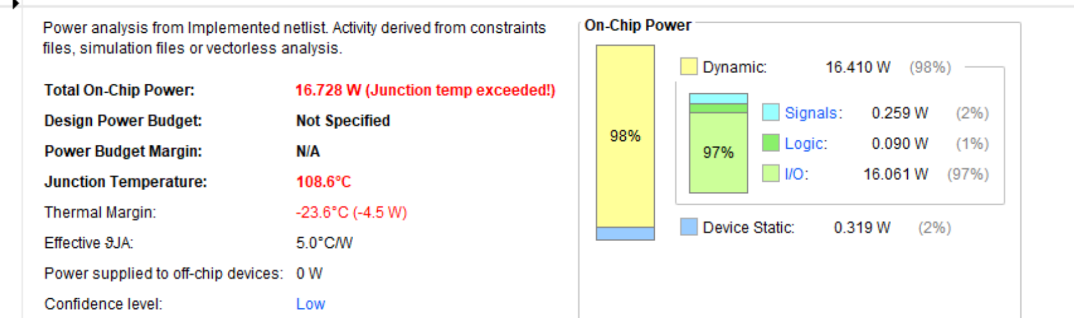


Figure 2: power estimation for Vivado’s default implementation

Unlike behavioral simulation which shows ideal transition between different states, timing analysis provide an observation of the delay when switching between two different states. The delay was estimated to be 1.8 ns for the results to be stable. Timing analysis is shown (figure 3). Unlike the prediction that have been made, there is no noticeable difference in timing analysis between the two conducted implementations.

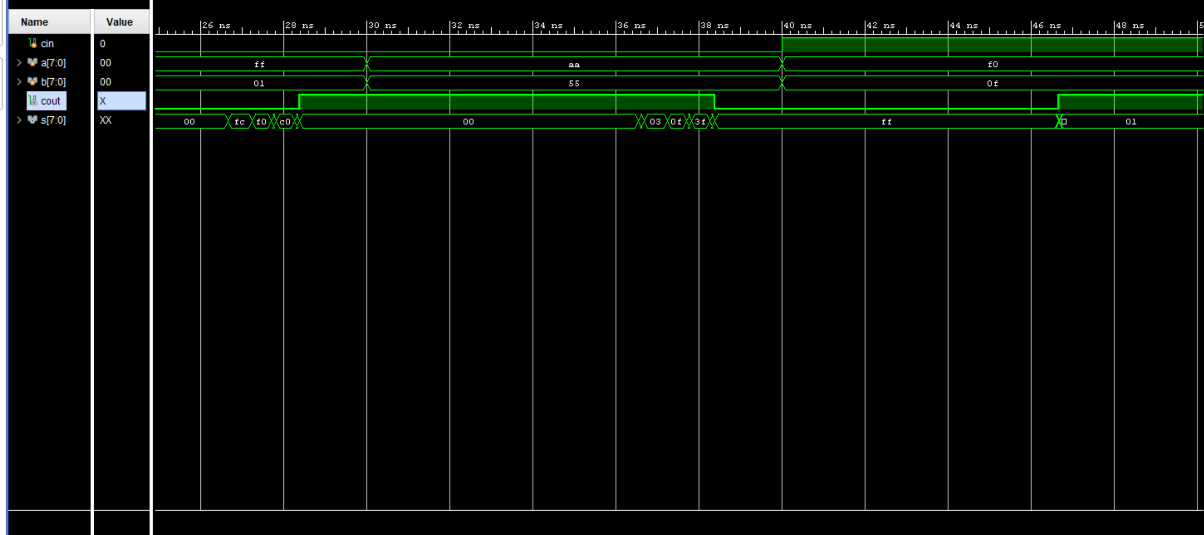


Figure 3: timing analysis for the conducted circuits

**SUMMARY OR CONCLUSION:**

No difference was observed in power consumption, the number of gates utilizations, and timing analysis unlike predicted results. No difficulties have been faced integrating the constrain file into the project. There are no prior improvements that can be made as far as the lab goes.

**SOURCE CODE:**

The source code that have been created for this lab and it had been attached separately. The attached files are the same as Lab 1 with further additions. **Full adder modular code**, **8-bits full adder code**, **a** **testbench code,** and **the constrain file** had been attached.

Tuesday

June 19, 2018

LAB 3

**DOWNLOADING TO THE FPGA BOARD**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab was to push a bit stream software file to the basys3 board burring the circuit we designed using Verilog code. We have used the instantaneous programming that erases as soon as the board get restarted. The other type of pushing a code to be preeminently burred into the board by generating a bin file that can be stored in the memory chip on the board. Therefore, every time the board is restarted the writing code in the memory chip kicks in. The design of the full adder from pervious labs had been upgraded to include a full subtractor when a button is pressed.

**DISCUSSION OF RESULTS:**

In order to add full subtractor modules, I have changed the logic equations of the sum and the carrier to function as a full subtractor. As well as change the constrain file accordingly to accommodate when a button is pressed. When the button is not pressed, the board will compute the given inputs as it activates the full adder modules. When the button is not pressed, however, the two inputs will be subtracted from each other as the full subtractor modules are activated. The results are shown in the board using the 16 LEDs that are in the board. The circuit schematic that implements full adder and full subtractor is shown (figure 1).

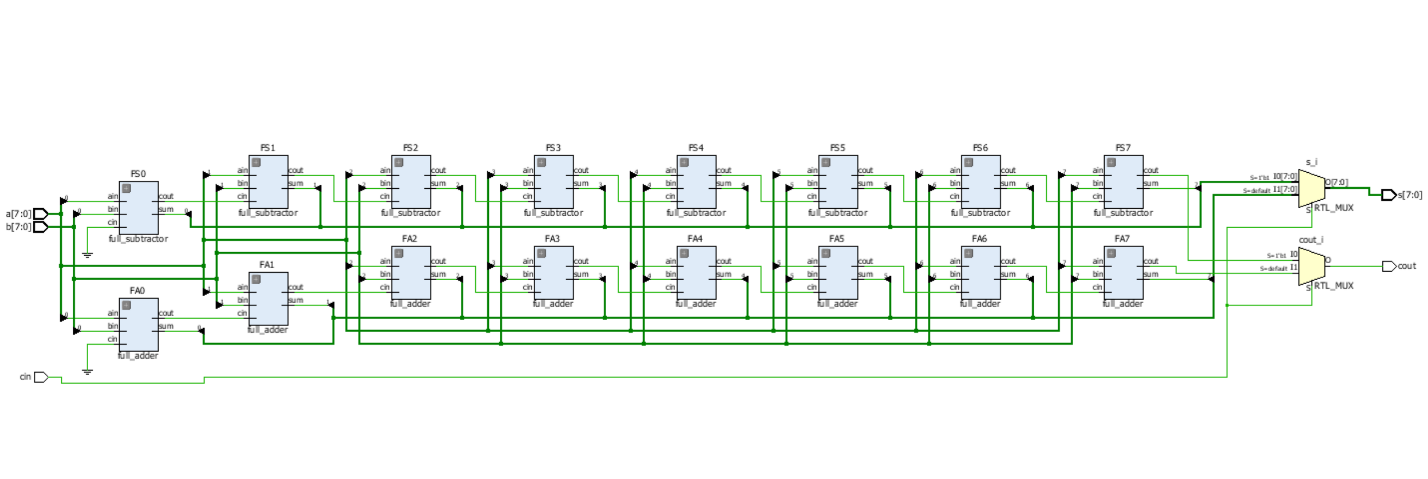
****

Figure 1: The circuit schematic that implements full adder and full subtractor

After the bin file was uploaded to the board, different inputs have been provided to make sure the full adder and the full subtractor are working as they meant to be.

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave as it was predicted. No difficulties have been faced pushing the bin file into the board. There are no prior improvements that can be made to improve power consumption or gates utilization.

**SOURCE CODE:**

The source code that have been created for this lab and it had been attached separately. The attached files are a continuation of previous labs and it has some further additions. **Full adder and subtractor modular code**, **8-bits full adder code**, 8**-bits full subtractor code,** and **the constrain file** had been attached.

**Yahya Alhinai**

EE 4301-LAB

Thursday

June 21, 2018

LAB 4

**AN 8-BIT PSEUDORANDOM NUMBER GENERATOR**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab is to design and implement an 8-bit pseudo random number generator on the Basys3 board. We used Linear Feedback Shift Register (LFSR) to generate random numbers which shift the bits in the register to one direction at one with two or more of the flip-flop outputs XORed to generate a bit that will work as an input to the shifted register.

**DISCUSSION OF RESULTS:**

In a Linear Feedback Shift Register there are possible outcome excluding the number 0 because it will repeat itself with no change in the number’s bits. Since we are using 8-bit pseudo random number generator, there are going to be 255 possible output.

Clock divider has been implemented in this lab as well to make the 7-segment display changes at a noticeable rate. It utilized the 100 MHz internal clock to generate slower clock to be used for the intended purpose.

Finally, shown the bits of the random number at 8 of the board’s LEDs and make sure the speed base of the number changes is suitable. The refresh rate of the 7-segment display has been adjusted so it can represent 4 digits 7-segment simultaneously as the display designed to show one digit at a time.

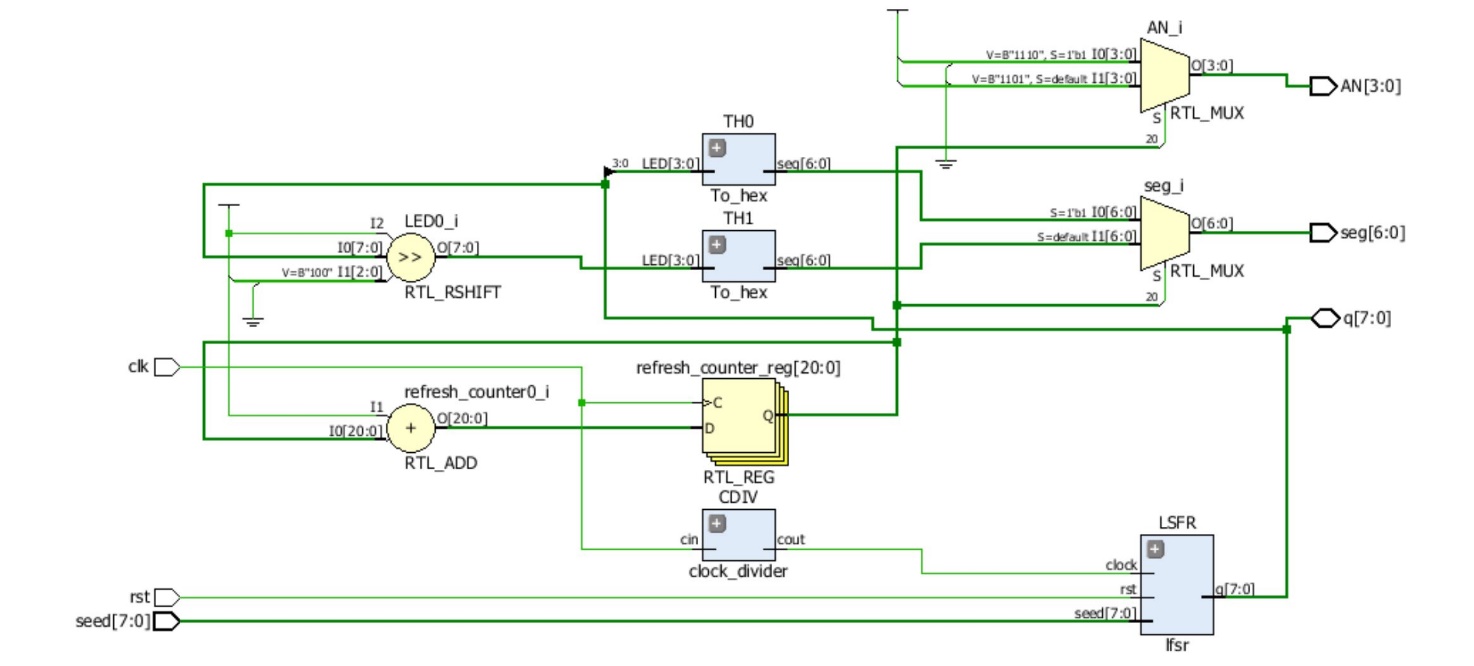
****

Figure 1: The full circuit schematic of the 8-bit pseudo random number generator

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave it was intended. No difficulties have been faced implementing the circuit design on the basys3 board. There are no prior improvements that can be made to improve power consumption or gates utilization. Also, the code can’t be further optimized.

**SOURCE CODE:**

The source code that have been created for this lab is updated and attached as a zip file with the rest of labs’ source code.

Thursday

July 28, 2018

LAB 5

**CASINO-TYPE GAME**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab is to build up on an 8-bit pseudo random number generator to create casino type game and implement it on the Basys3 board. The game will start to generate random numbers and stop when it is in win/lost state after the button is being pressed. Otherwise, it will keep generating random numbers. The circuit is implemented in a way that shows random number and win/lost state at the 7-segment display.

**DISCUSSION OF RESULTS:**

The first step in this lab was to create a modular to convert number bits to 7-segment numbers. Those modules take 4 bits at a time to convert it into a base of hexadecimal number that can be displayed.

Clock divider has been implemented in this lab as well to make the 7-segment display changes at a noticeable rate. It utilized the 100 MHz internal clock to generate slower clock to be used for the intended purpose.

The refresh rate of the 7-segment display has been adjusted so it can represent 4 digits 7-segment simultaneously as the display designed to show one digit at a time. The full circuit schematic of the casino-type game is shown in Figure 1.

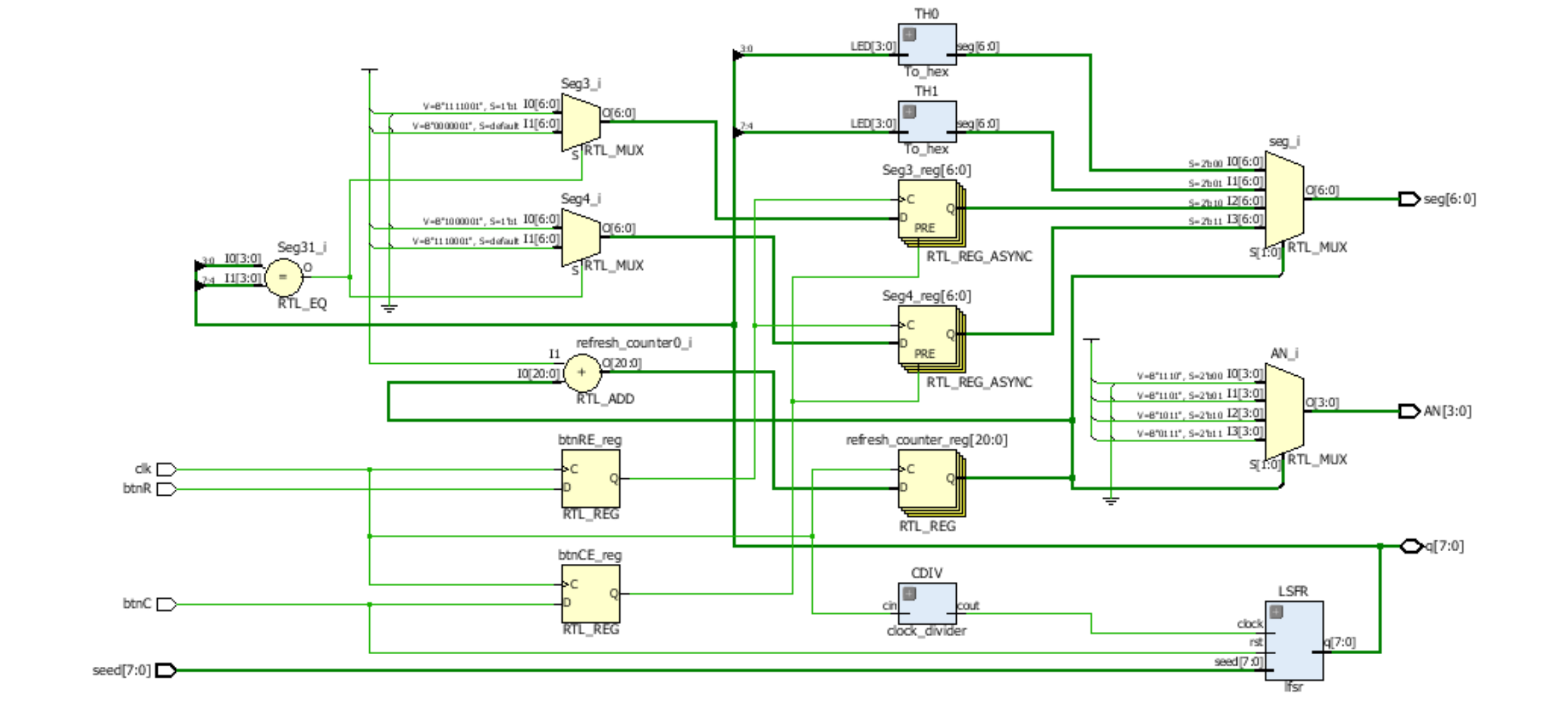


Figure 1: The full circuit schematic of the casino-type game

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave it was intended. No difficulties have been faced implementing the circuit design on the basys3 board. There are no prior improvements that can be made to improve power consumption or gates utilization. Also, the code can’t be further optimized.

**SOURCE CODE:**

The source code that have been created for this lab is updated and attached as a zip file with the rest of labs’ source code.

Tuesday

July 10, 2018

LAB 6

**WRITING CODE FOR SYNTHESIS. ELECTRONIC DICE GAME**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab is to further modify the last lab that generate random number to work as an electronic dice game. Through this code transformation, we were asked to implement state machine Verilog code. We learn, as well, how to create separate modules that is controlled by top module to simulate the implication of a real-world application.

**DISCUSSION OF RESULTS:**

Machine state have been integrated in the circuit to create a proper transformation between the games state. Transferring from one state to another is due to button pressing until it lands on one of the two states win/lose. Otherwise, random numbers will continue to be generated. The full circuit schematic of the electronic dice game is shown in Figure 1. Also, The Flowchart for Dice Game that is used to implement the state machine in the circuit is shown in Figure 2.

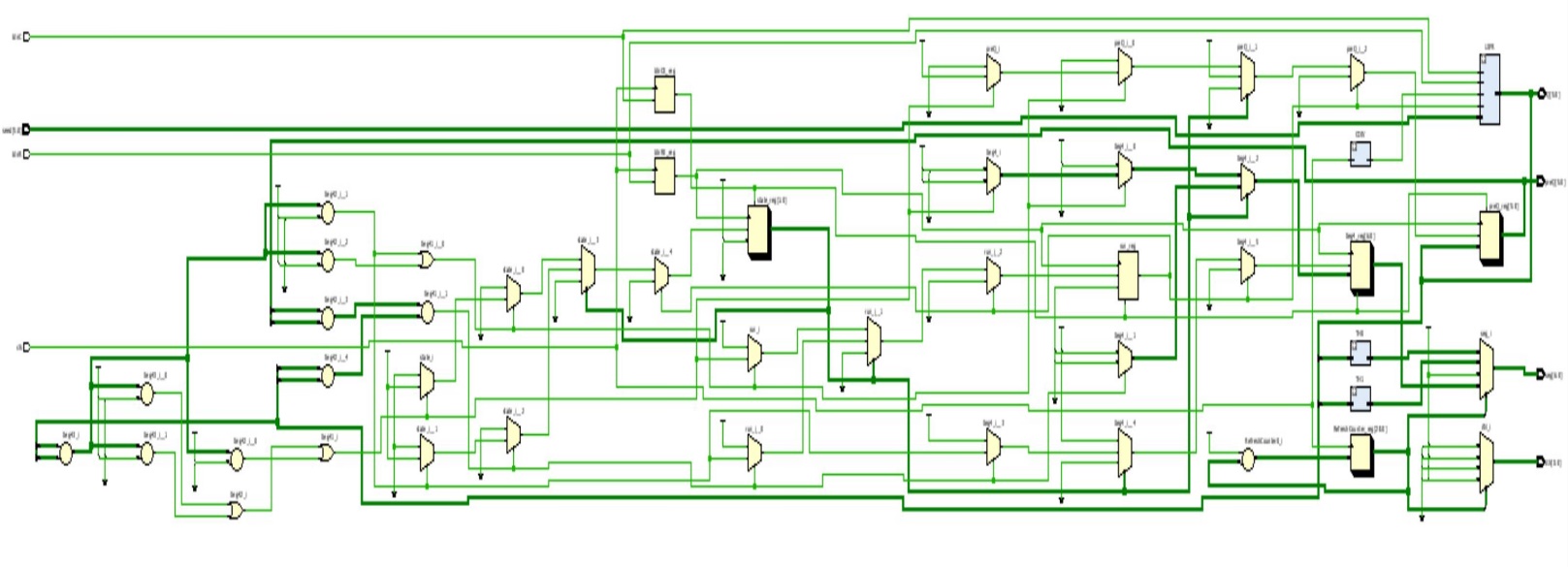
****

Figure 1: The circuit schematic that implements full adder and full subtractor

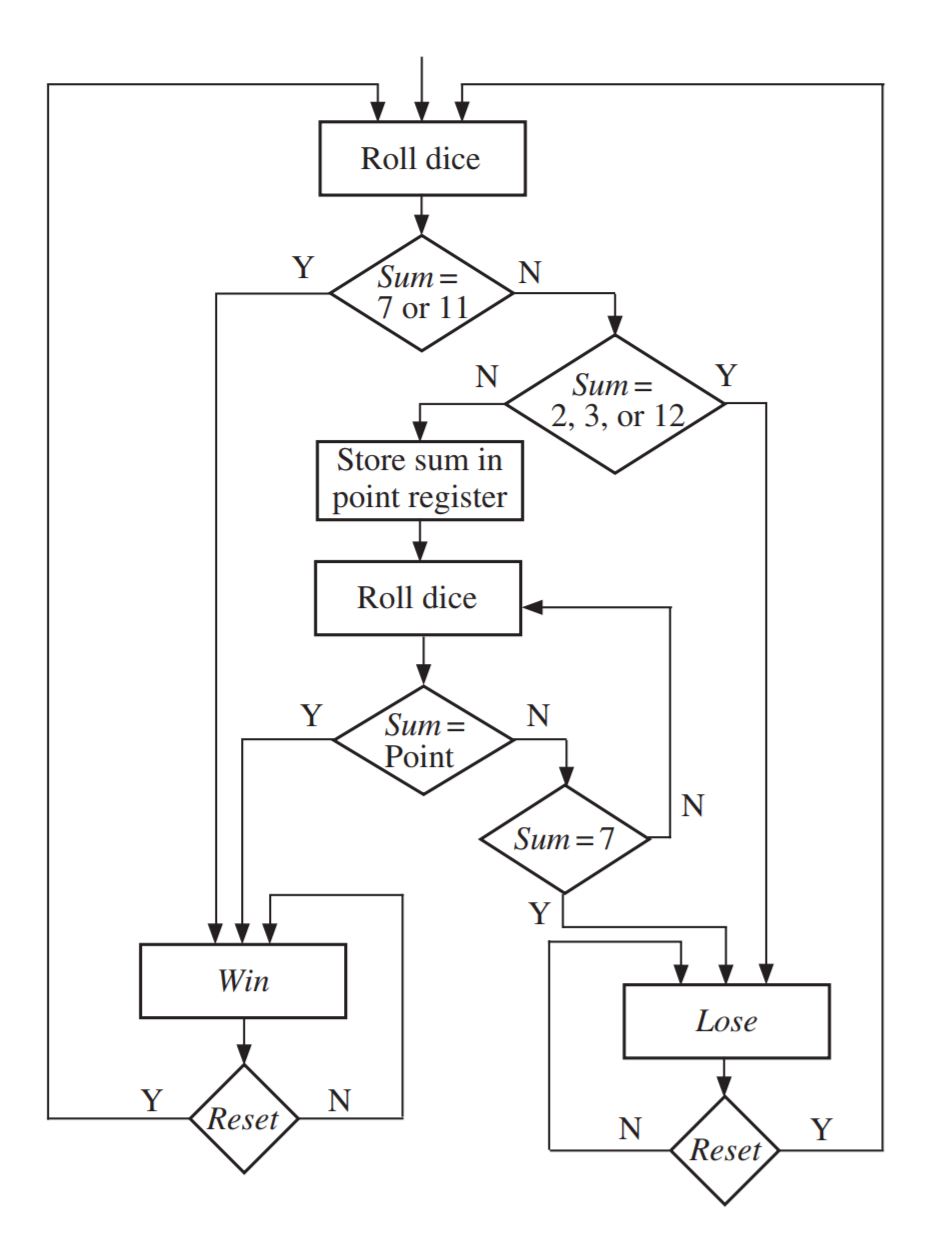
****

Figure 2: Flowchart for Dice Game that is used to implement the state machine in the circuit

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave it was intended. No difficulties have been faced implementing the circuit design on the basys3 board. There are no prior improvements that can be made to improve power consumption or gates utilization. Also, the code can’t be further optimized.

**SOURCE CODE:**

The source code that have been created for this lab is updated and attached as a zip file with the rest of labs’ source code.

Tuesday

July 17, 2018

LAB 7

**STORING AND MOVING TEXT**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab is to design a circuit that can enter a number as an input and store it. Then display the stored test that was previously enter character after another. These characters are created using the basys’s switches. This lab utilized two-dimension array as a memory to store text.

**DISCUSSION OF RESULTS:**

One of the most important method utilized in this lab was pointer. Pointers were useful to keep track of the number of characters that was entered. Three buttons were used to reset, store, and modify. State machine was used to distinguish the different state and to seamlessly transform from one state to another. The full schematic of storing and moving text circuit is shown in Figure 1.

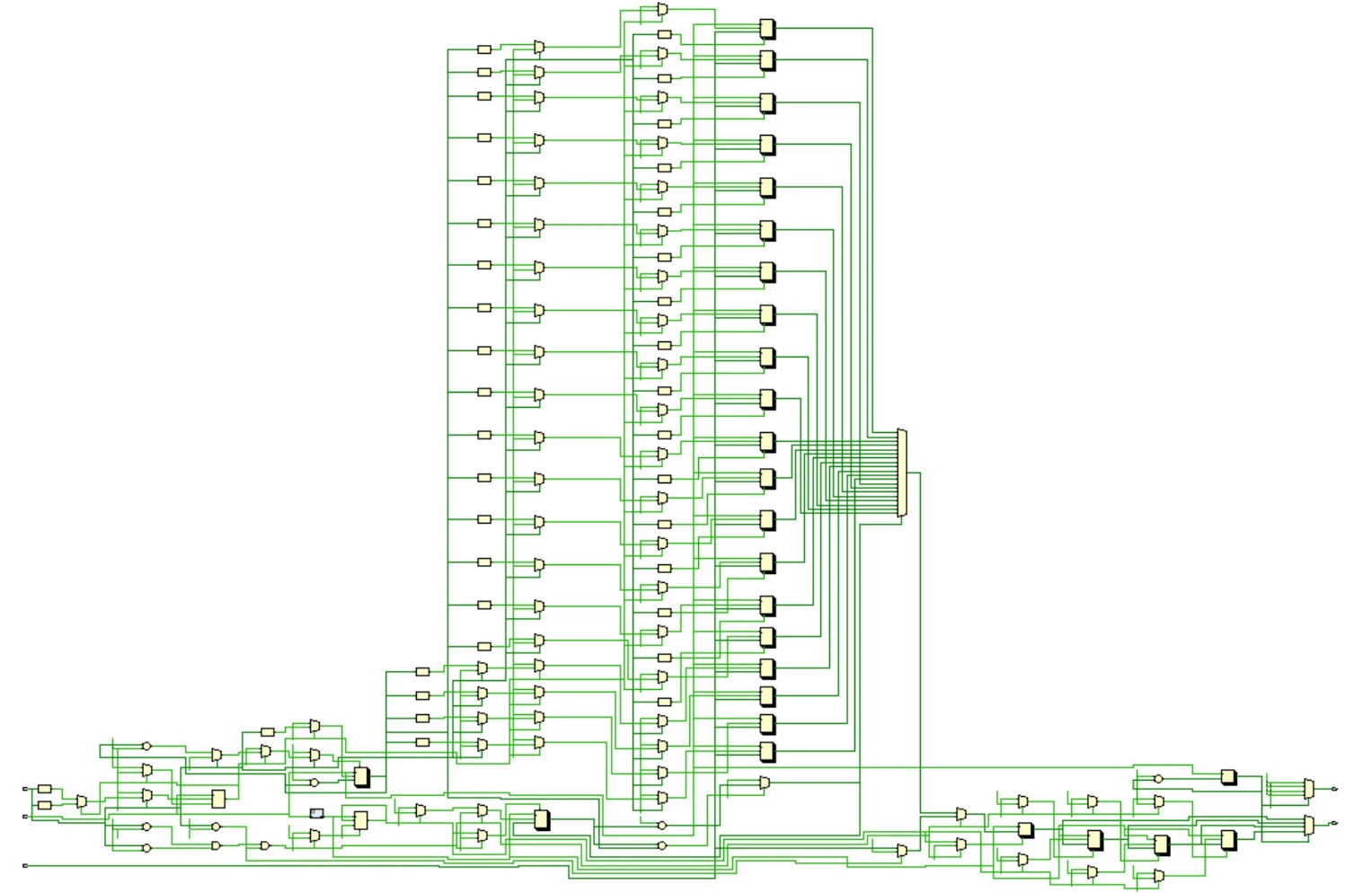
****

Figure 1: The full schematic of storing and moving text circuit

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave it was intended. No difficulties have been faced implementing the circuit design on the basys3 board. There are no prior improvements that can be made to improve power consumption or gates utilization. Also, the code can’t be further optimized.

**SOURCE CODE:**

The source code that have been created for this lab is updated and attached as a zip file with the rest of labs’ source code.

Tuesday

July 24, 2018

LAB 8

**DESIGN OF A 16-BIT CORDIC COMPUTER**

**DESCRIPTION OF THIS LAB:**

The purpose of this lab is to design and implement a 16-bit CORDIC computer. The usefulness of the algorithm is that it can approximate the values of sin(θ) and(θ) cos by the provided angle (θ). The simplicity and power efficiency is what makes this algorithm stood out.

**DISCUSSION OF RESULTS:**

A bit-series implementation of CORDIC has been implemented to find the value of sine and cosine. There are 16 bits in the series; therefore, it’s necessary to shift and adjust the value 16 times. The whole idea behind this implementation is that It consists of 3 sets of shift registers, serial adder-subtractors, and two input multiplexers. It’s going through 3 values simultaneously: sine, cosine and the angle adjustment. After 16 times of shifts and adjustments, the value for sine and cosine would have a a really good approximation. The full schematic of 16-bit cordic computer is shown in Figure 1.

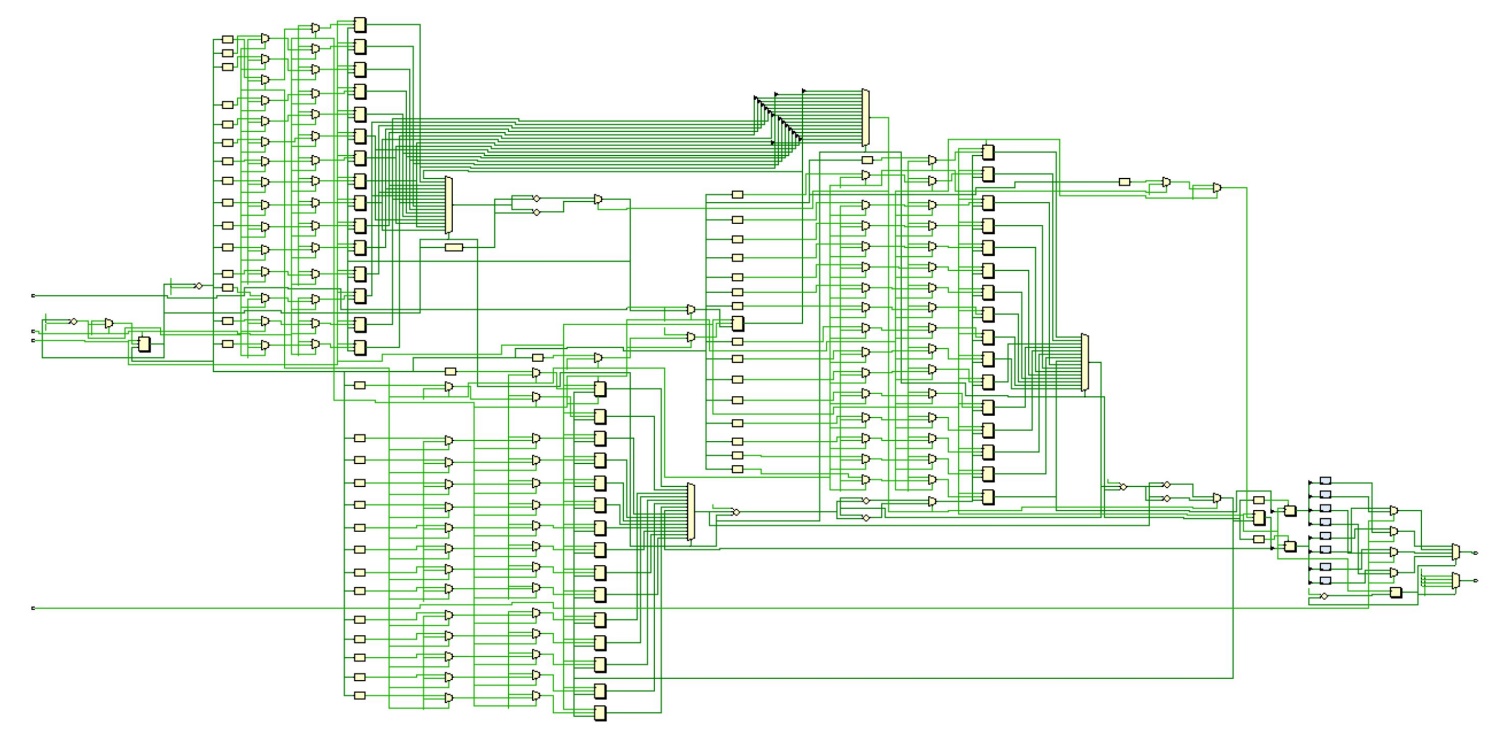
****

Figure 1: The full schematic of 16-bit cordic computer

**SUMMARY OR CONCLUSION:**

The goals for this lab was fully met. The board behave it was intended. No difficulties have been faced implementing the circuit design on the basys3 board. There are no prior improvements that can be made to improve power consumption or gates utilization. Also, the code can’t be further optimized.

**SOURCE CODE:**

The source code that have been created for this lab is updated and attached as a zip file with the rest of labs’ source code.