**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 $2^n - 1$ 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.

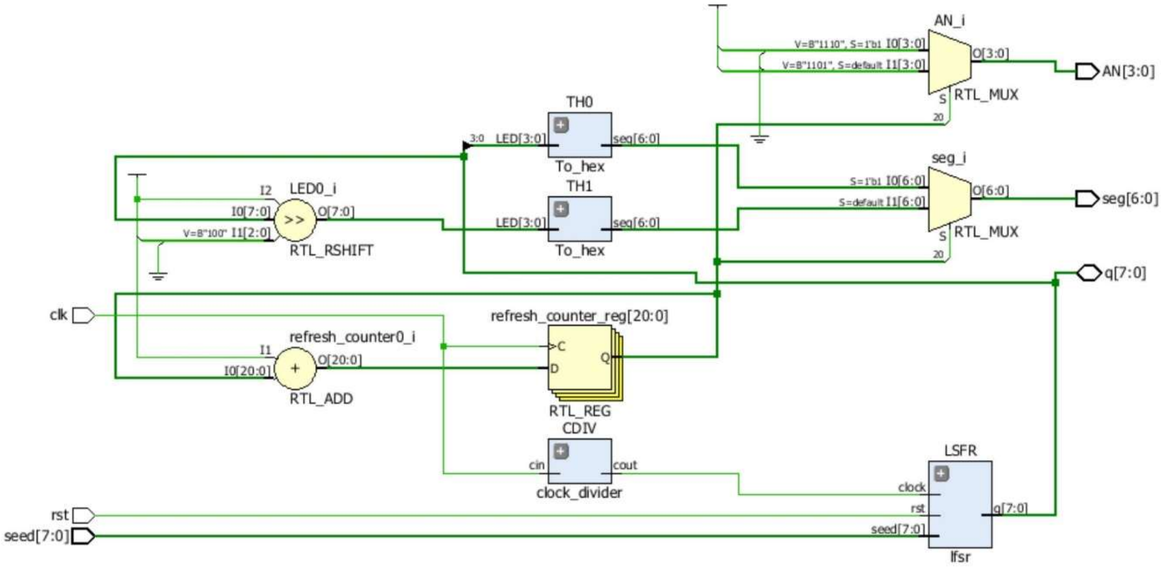

*Figure 1: The full circuit schematic of the 8-bit pseudo random number generator*

## 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.

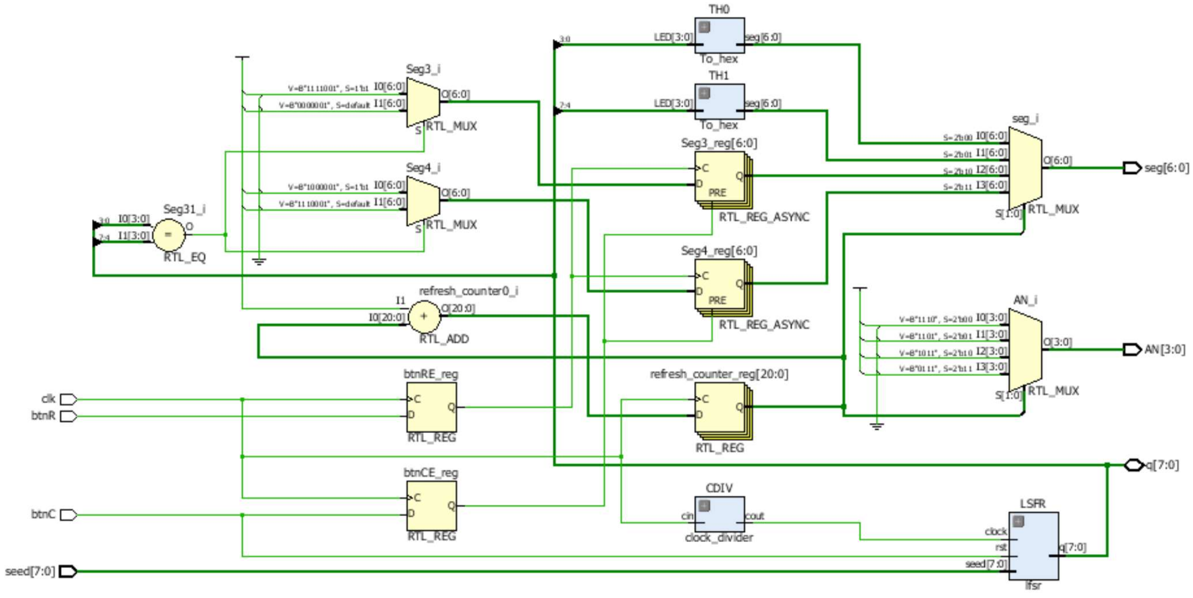

*Figure 1: The full circuit schematic of the casino-type game*

**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.

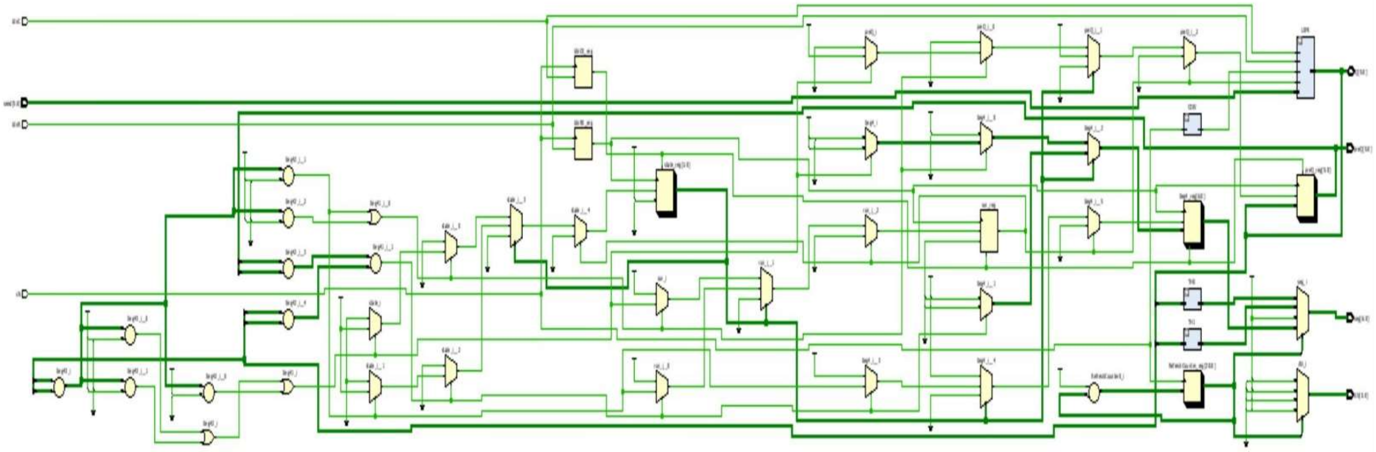

*Figure 1: The circuit schematic that implements full adder and full subtractor*

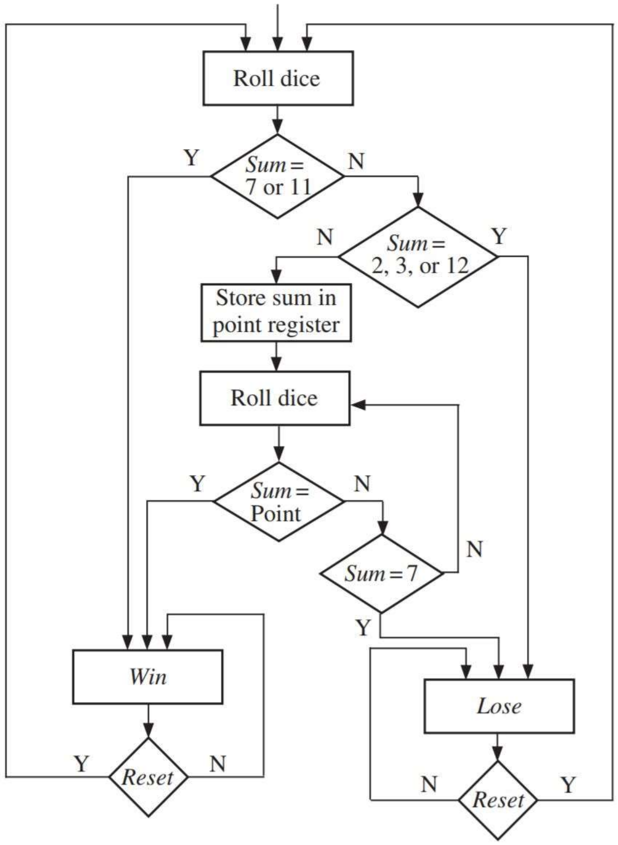

*Figure 2: Flowchart for Dice Game that is used to implement the state machine in the circuit*

**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.

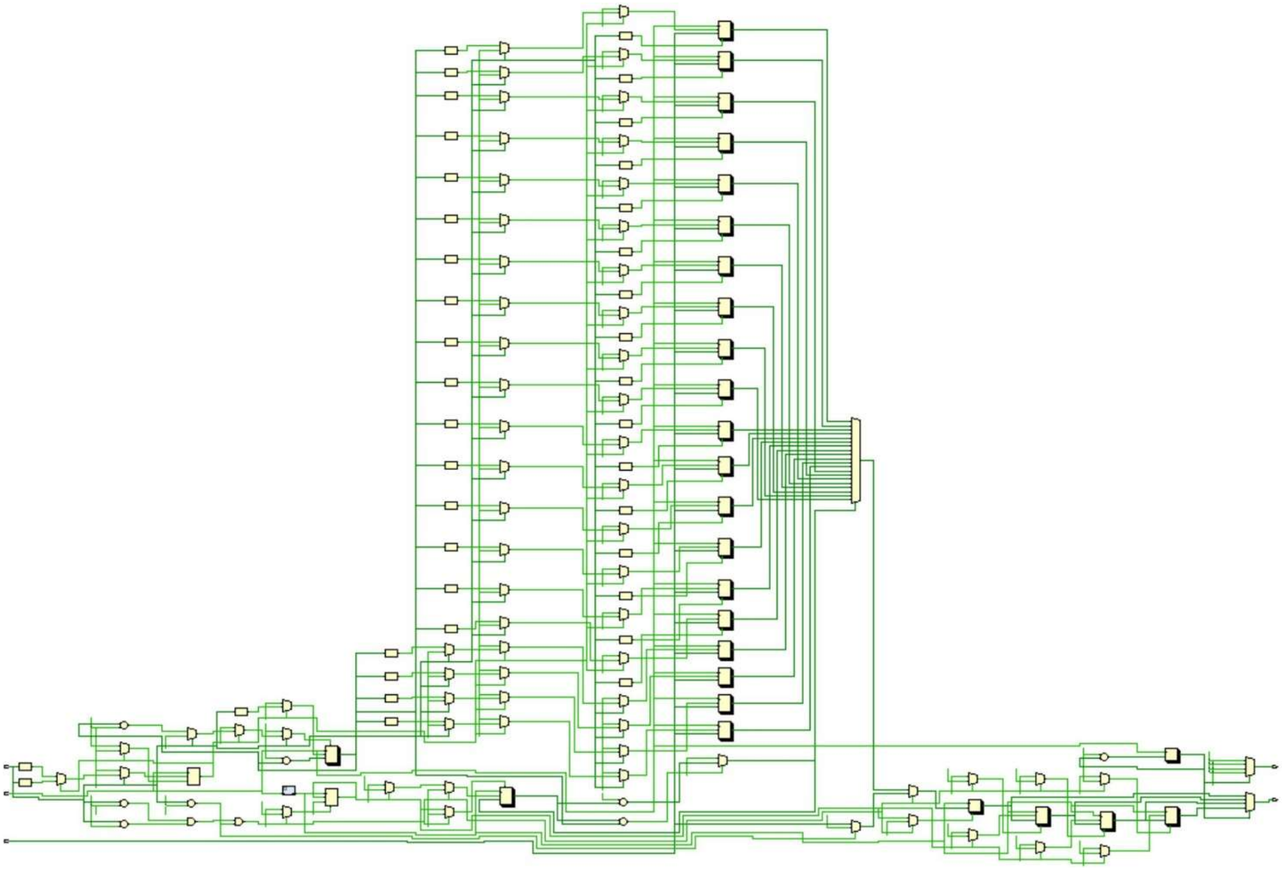

*Figure 1: The full schematic of storing and moving text circuit*

**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.

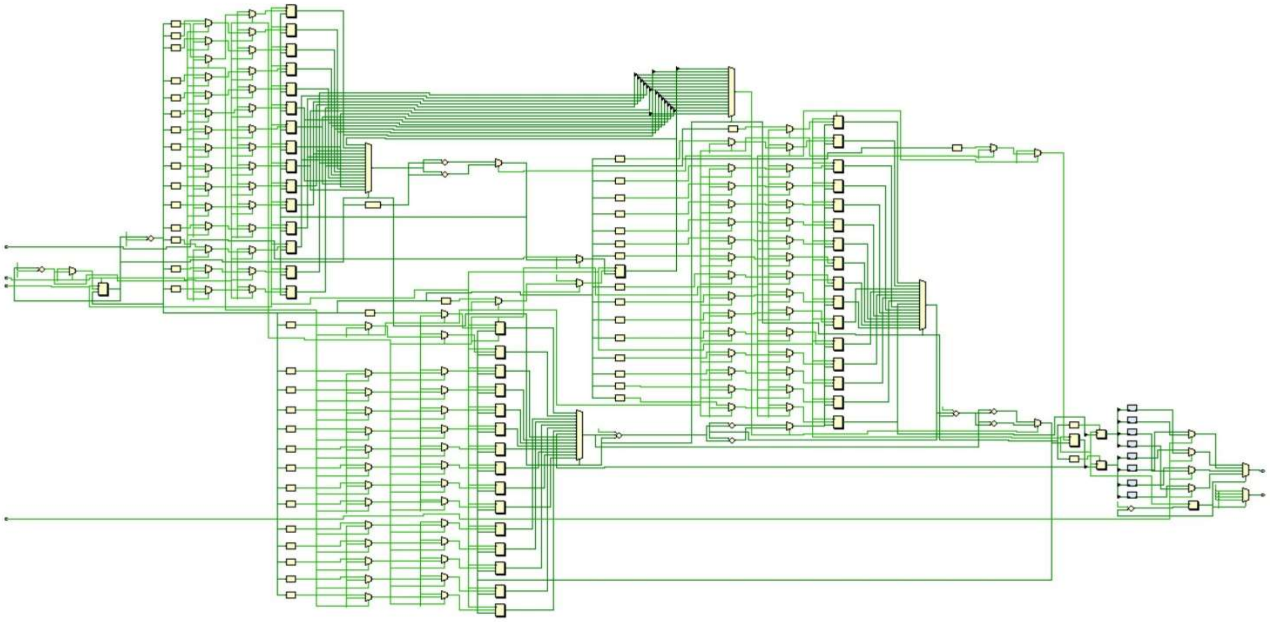

*Figure 1: The full schematic of 16-bit cordic computer*

**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.