Khoa Tran

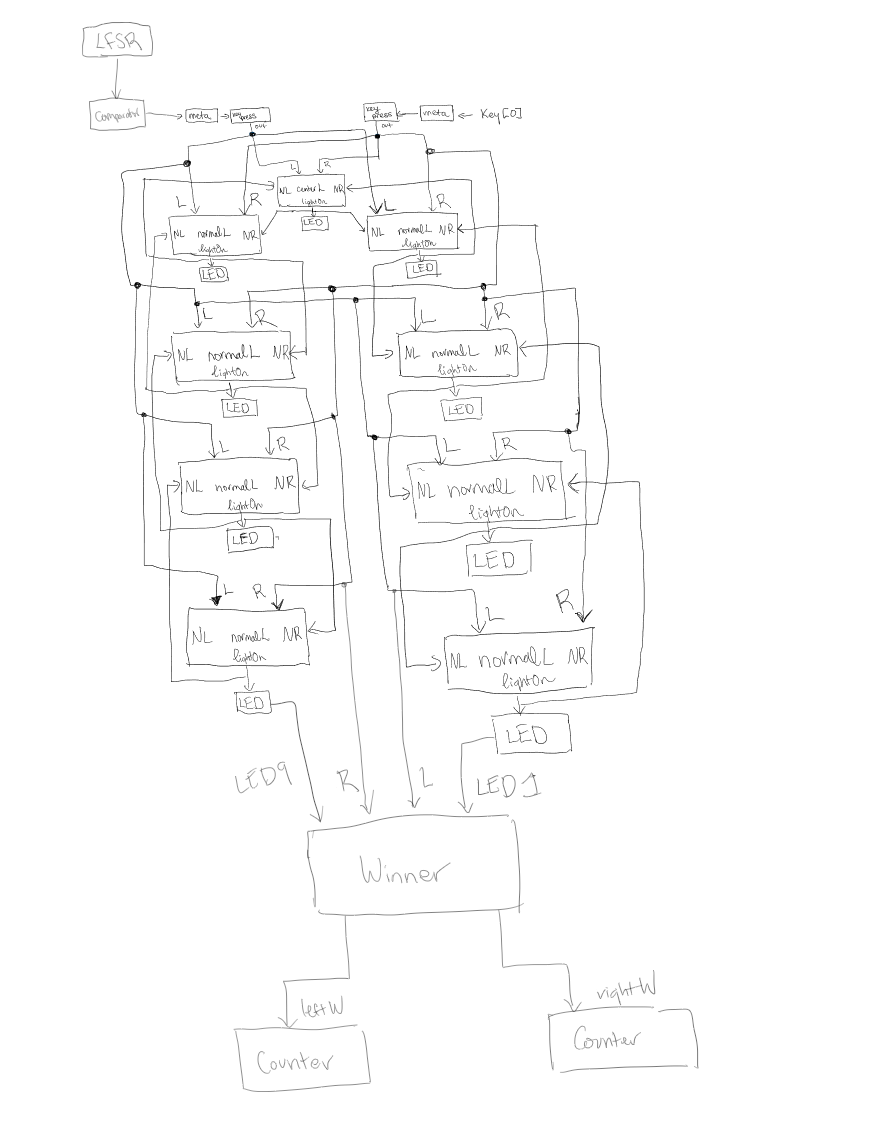
EE 271

August 08, 2020

Lab 5 Report

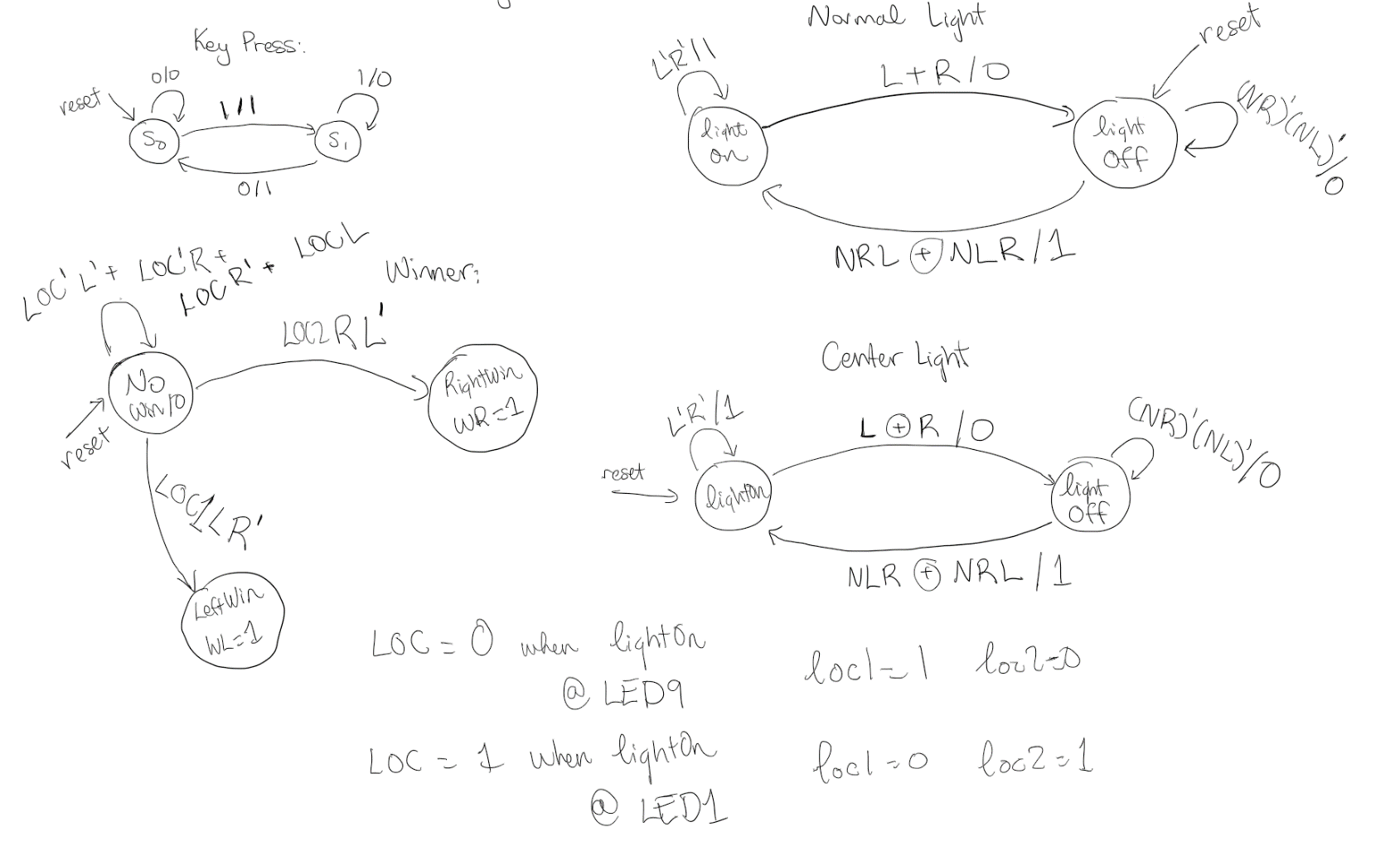
**Procedure**

**Tug of War Against Computer**

Approaching this problem, I first drew up the block diagram to figure out how to connect each FSM in order to produce the correct output for each user input that is possible. I drew the connection for the meta and keypress module to control the metastability of the user input and allows for any holding of the button to only count as a press. Next, it is connected to the center light and the light being on, sends out to the adjacent normal lights on the left and right side. This concept is applied to the other 8 normal lights surrounding the center one. Afterwards, I drew up the state diagram for each FSM in order to capture the state conditions that each FSM needs. The state diagram then allows me to transfer the FSM on picture to code. Simulating each module and connecting towards the FPGA board.

With the addition of the computer input, I drew up the LFSR state diagram, in order to understand how the output of LFSR and input of SW[8:0] will be compared in the comparator in order to produce an output for the computer player. Lastly, the counter takes in inputs from the winner and counts each player current score. I drew up the state diagram to understand how to implement the system.

Figure 1: Block diagram for Tug of War System against computer



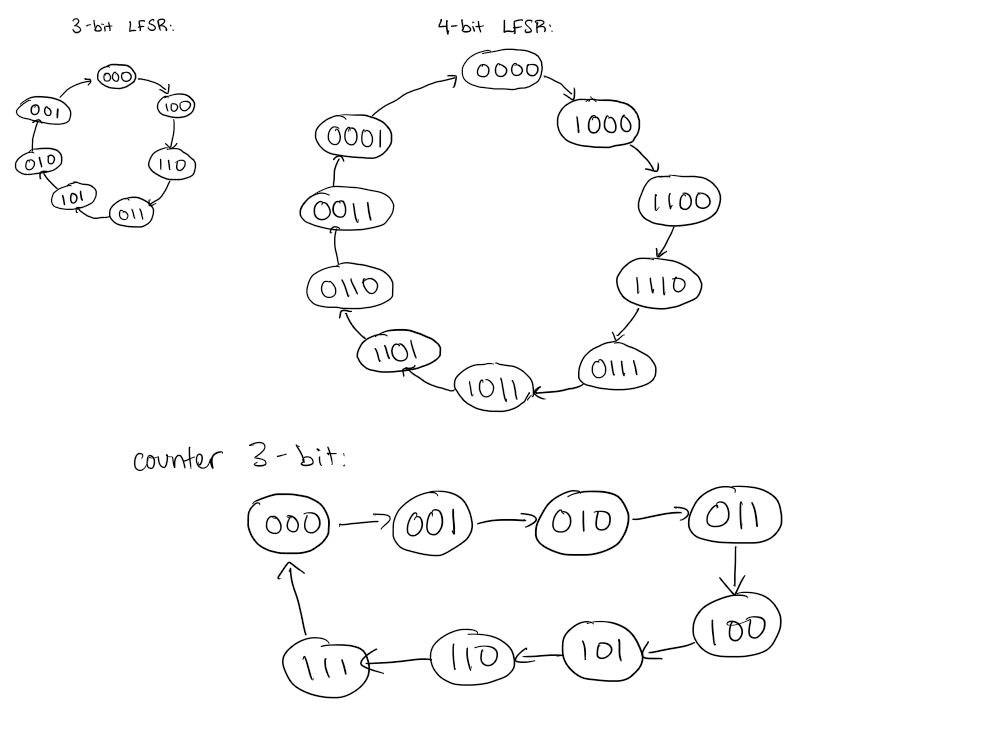


Figure 2: State diagram for each FSM in Tug of War system against the computer

**Results**

**Meta:**

For the first part, I tested the meta module to check if the metastability works with the input from the user as it allows for a two-clock cycle delay of the inputs. Also, the input and output progresses if the clock is at the positive edge.

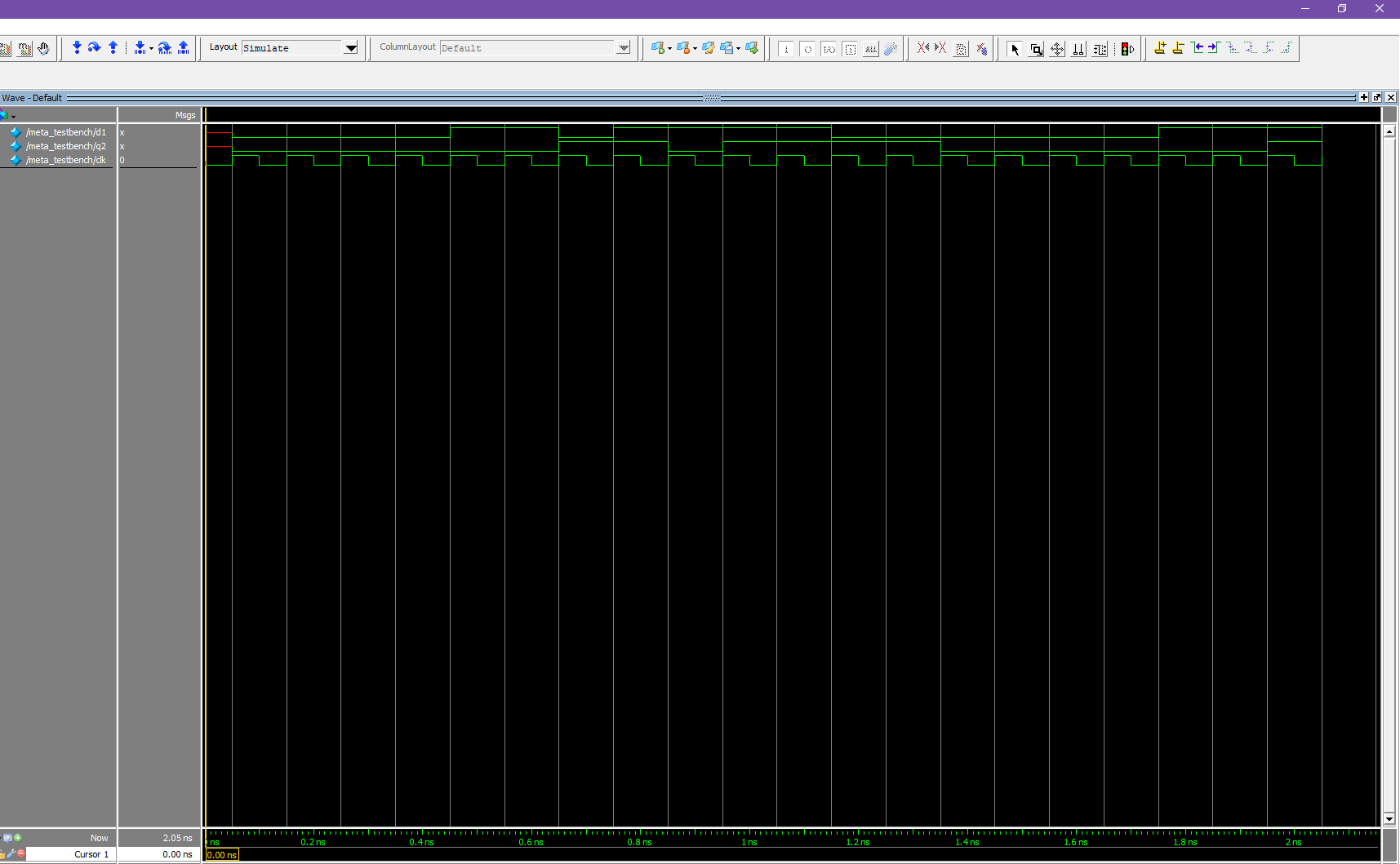
****

Figure 3: The waveform generated by the meta module for DFF

**Keypress:**

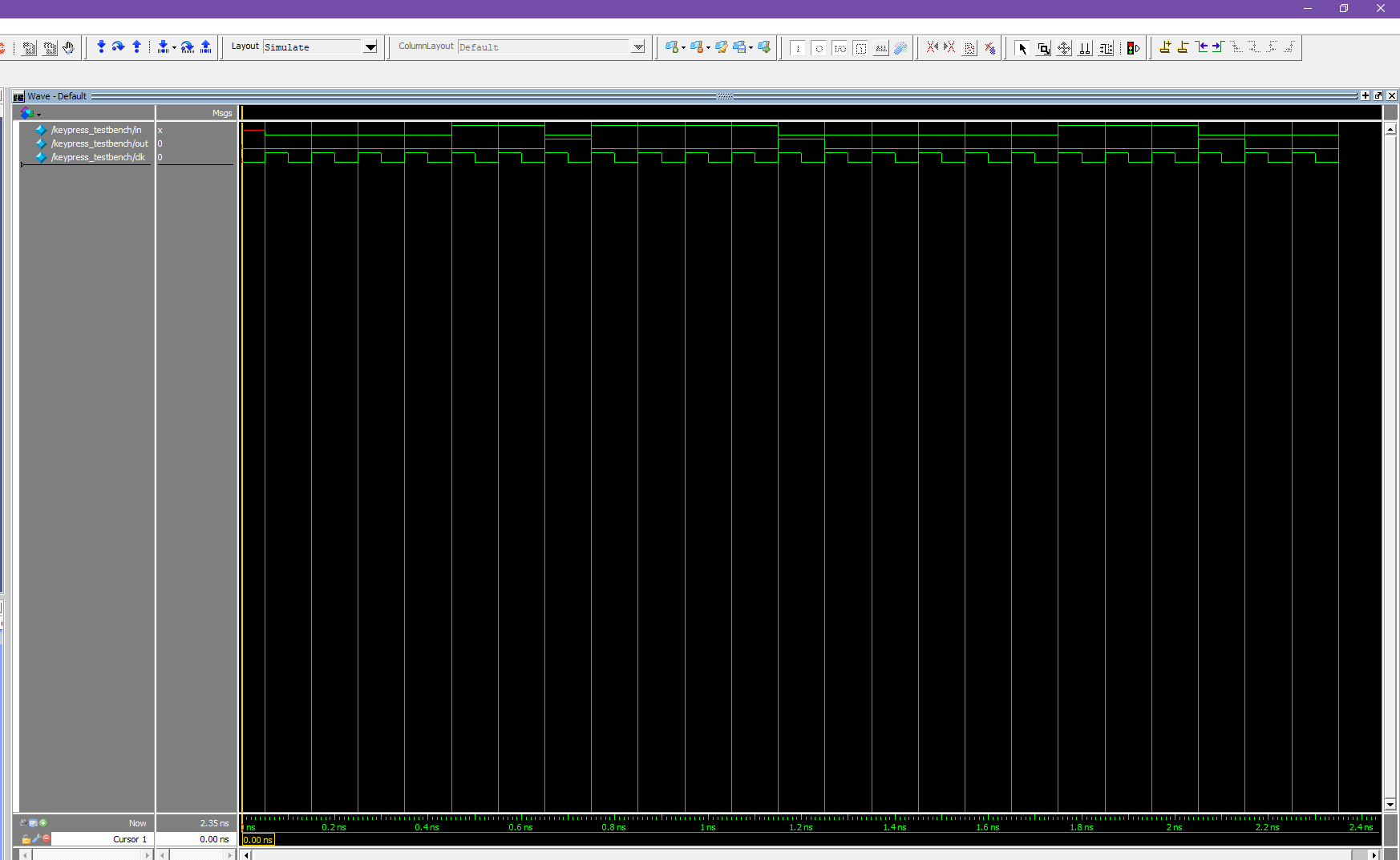
This simulation tested the inputs of in with the output of out. This FSM controls the input to allow a hold of the button as a press and forcing the user, not to hold the input. The inputs and outputs only take effect when the clock hits the positive edge.

Figure 4: The waveform generated by FSM of keypress

**normalLight:**

This simulation tested the output of the lightOn with the inputs of L, R, NL, and NR. This FSM allows for the light to be off when reset is called but turns on when either L and NR is true or when R and NL is true. When the light is on, either call to L or R would turn the light off. The inputs and outputs only take effect when the clock hits the positive edge.

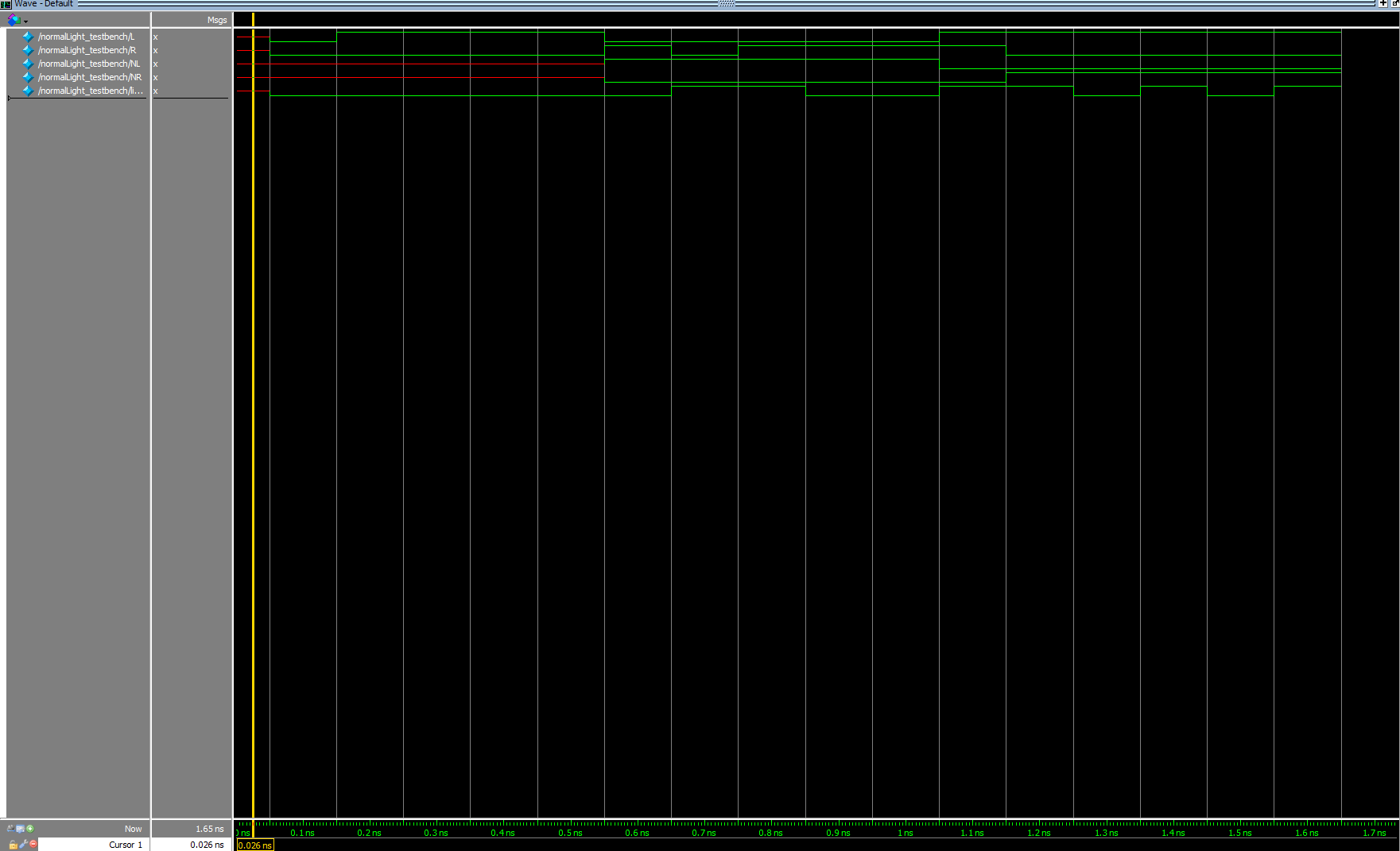
****

Figure 5: The waveform generated by FSM of normalLight

**centerLight:**

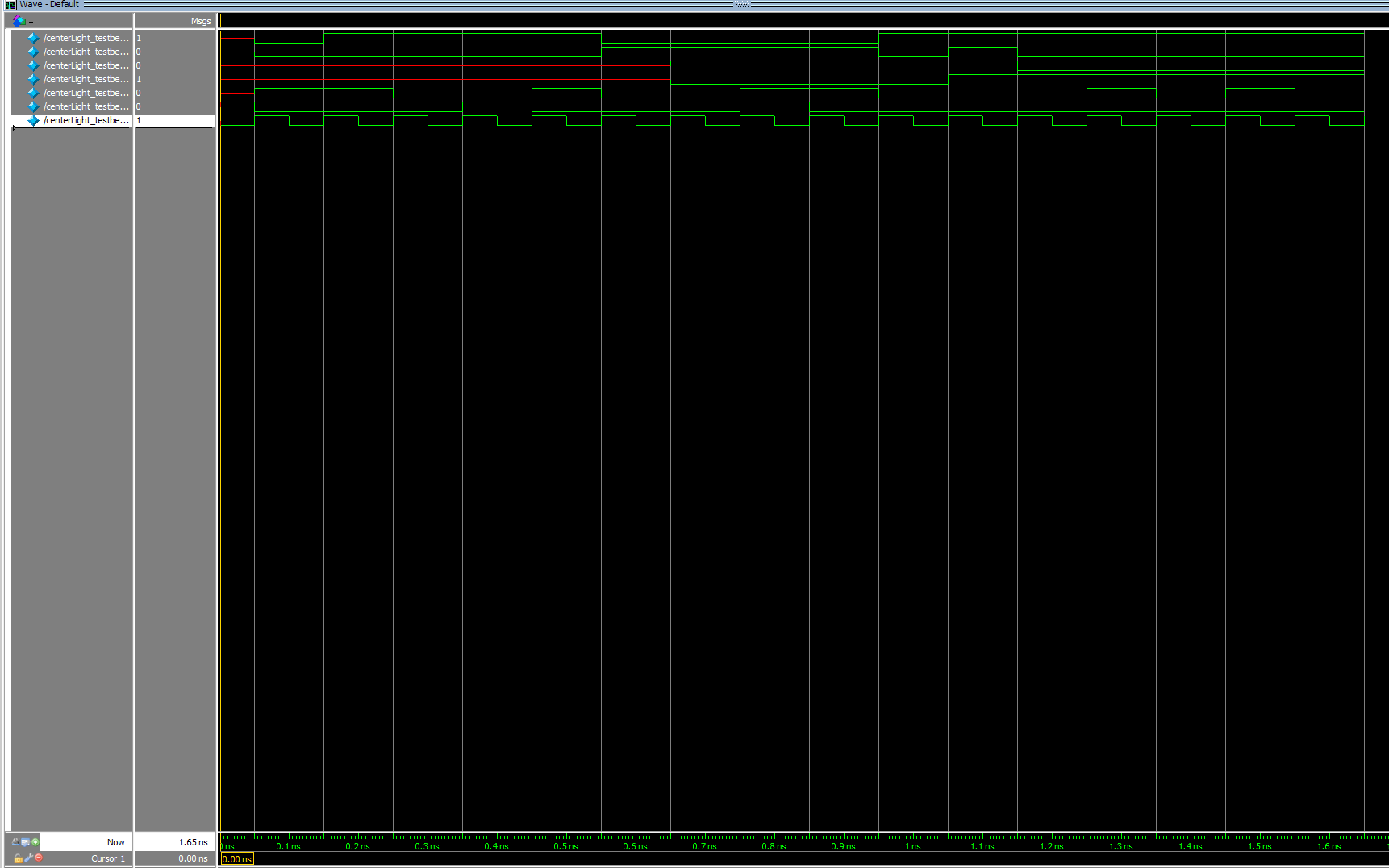
This simulation tested the output of the lightOn with the inputs of L, R, NL, and NR. This FSM allows for the light to be on when reset is called but turns on when either L and R is called. When the light is off, a call to L and NR is true or R and NL is true, allows for the light to turn on. The inputs and outputs only take effect when the clock hits the positive edge.

Figure 6: The waveform generated by FSM of centerLight

**Winner:**

This simulation tested the output of out1, out2, and temp as output for either left player or right player as the winner and for when there’s a winner to reset the other modules. The inputs are L, R, loc1, and loc2. The loc1 and loc2 are variables to indicate if either LEDR[9] or LEDR[1] is on to test if when the light are in those positions, how the calls on L and R have an effect on the out. The inputs and outputs only take effect when the clock hits the positive edge.

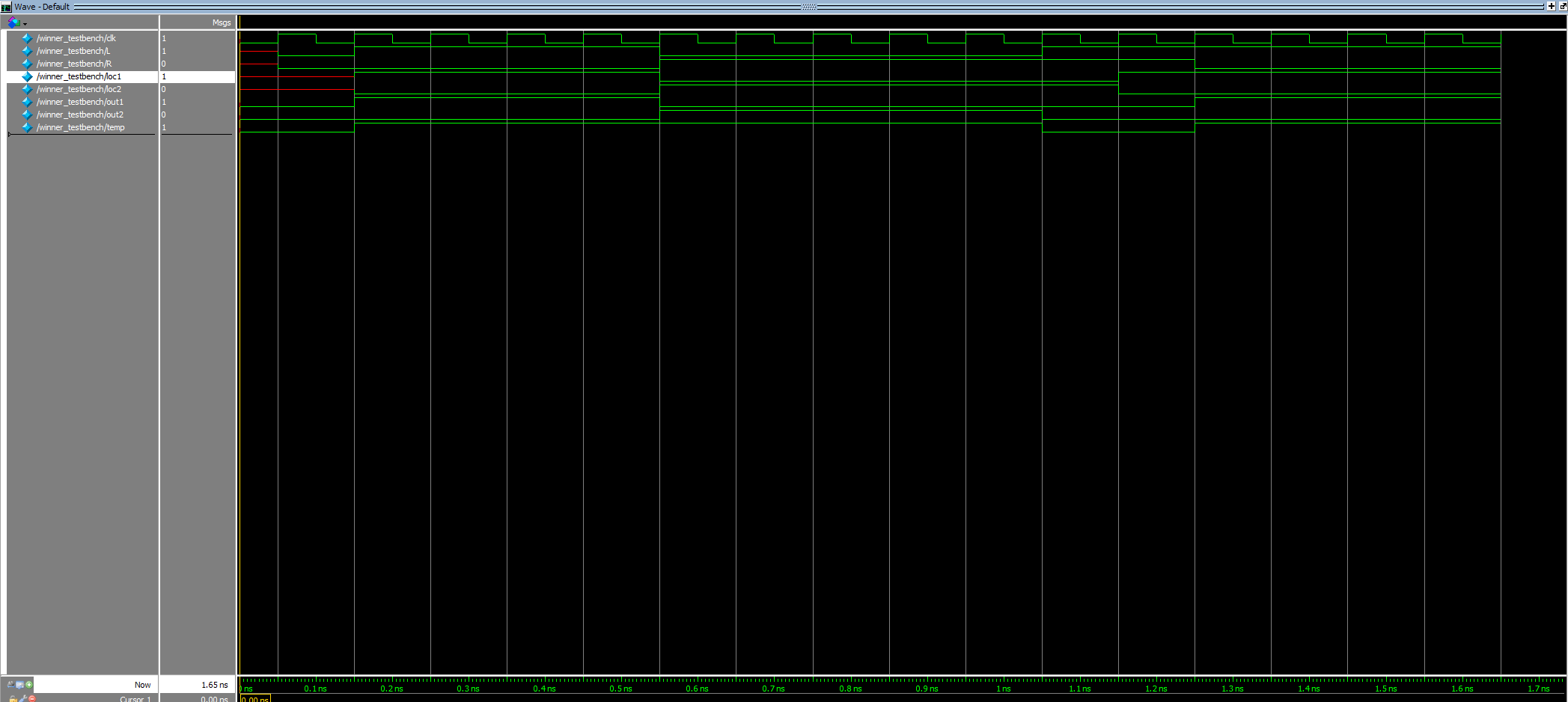
****

Figure 7: The waveform generated by FSM of winner

**LFSR:**

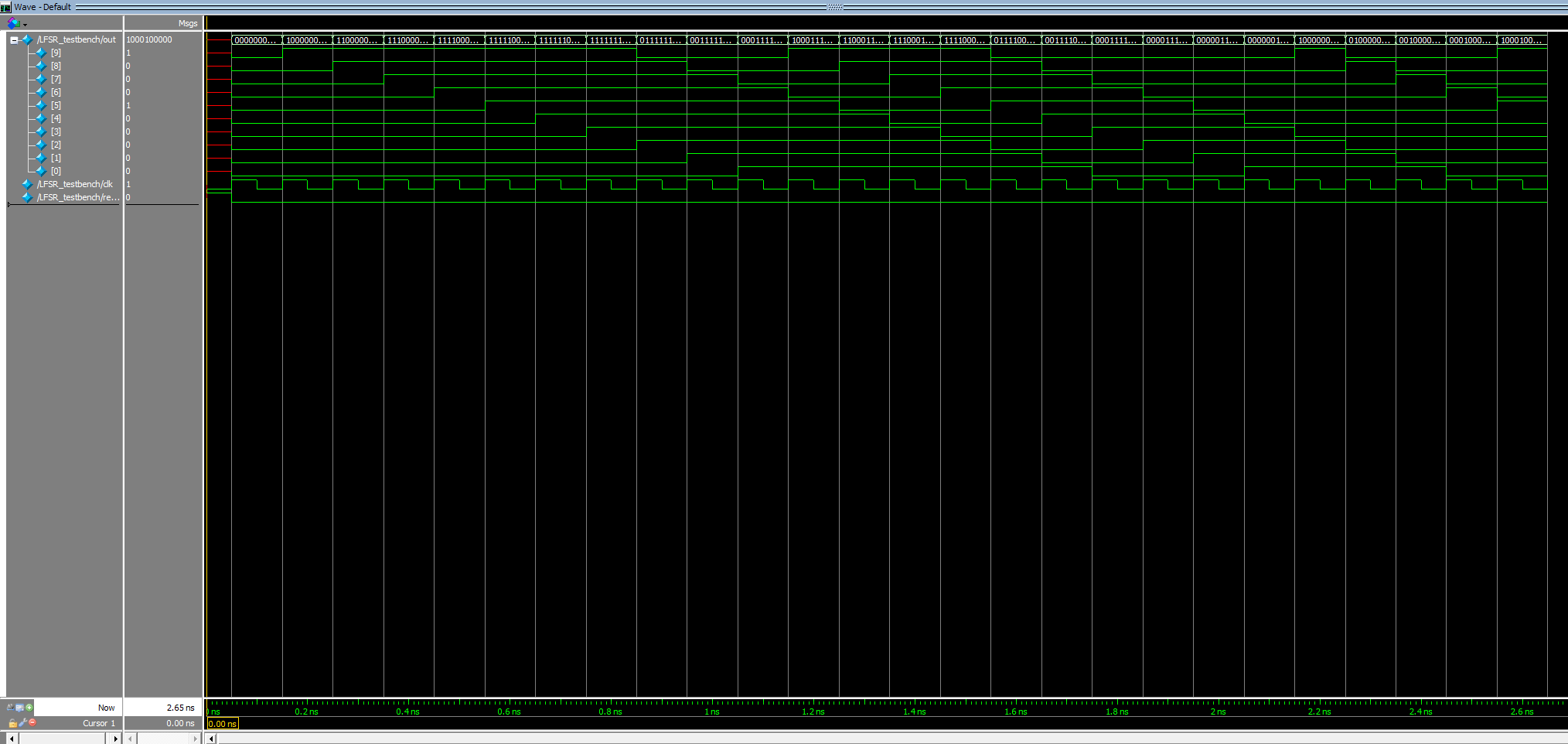
This simulation tested the output, out, as the random 10-bit binary value that is generated from a 10-bit LFSR with tabs at DFF 7 and 10 to maximize the number of random values generated. This simulation test for the positive clock edges and reset for the output, out.

Figure 8: The waveform generated by LFSR module

**Counter:**

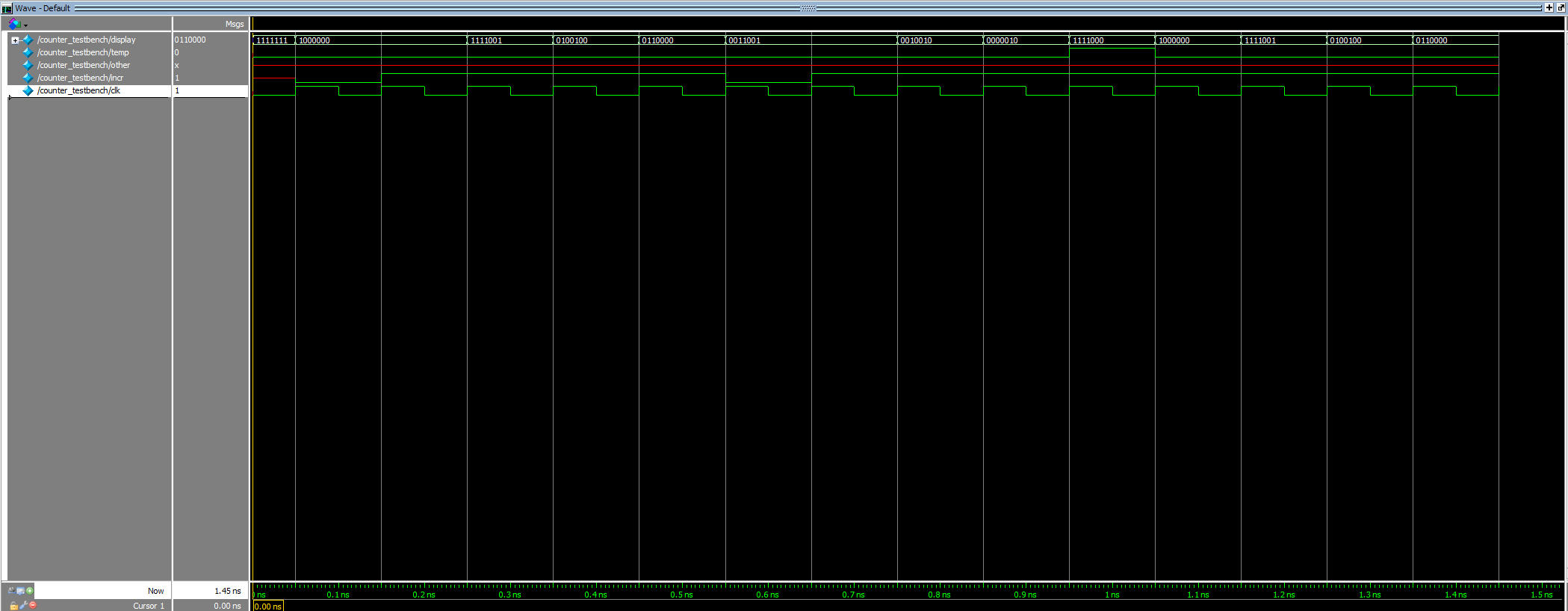
****This simulation test for the output of the display as the 7-bit HEX board display outputting the current number in the counter. The counter takes in, incr, and increments the current value by 1 whenever incr is true. With the random number of incr, simulation test for display to output a certain 7-bit binary output for the HEX board, which is active low.

Figure 9: The waveform generated by counter module

**Compare:**

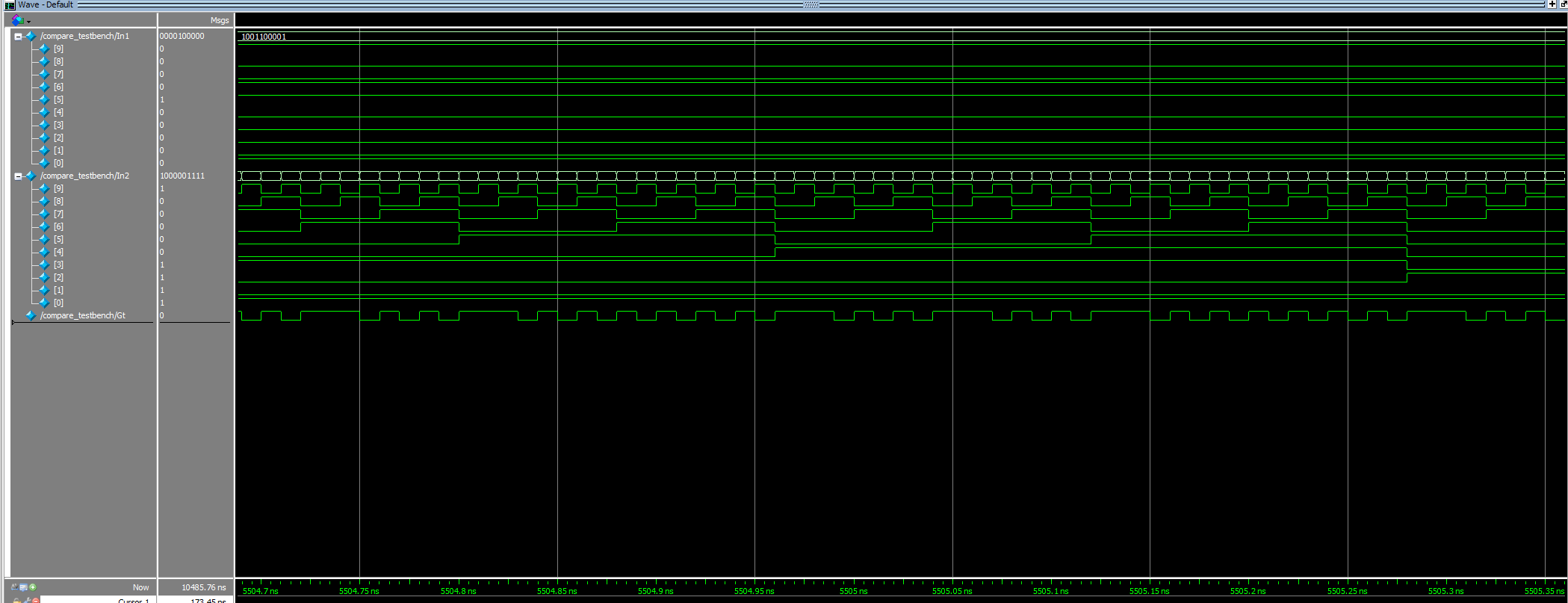
This simulation test for the comparison between two 10-bit inputs and outputs True, when the first input is greater than the second. The output, Gt, outputs true when In1 is greater than In2, and otherwise, outputs false. This simulation shows the accuracy of the comparison between the two input values.

Figure 10: The waveform generated by compare module

**DE1\_SoC:**

This simulation tested the inputs of the KEY[3] and SW[0] on the outputs of the different LEDR, HEX5, and HEX0 display. When the inputs of KEY[3] and SW[0] compare with LFSR reaches either LEDR[9] or LEDR[1] and the right input is entered, the counter module counts up the current score of the player that won, and outputs to either HEX5 or HEX0 the current score of the player on either the left or right side. The inputs and outputs only take effect when the clock hits the positive edge.

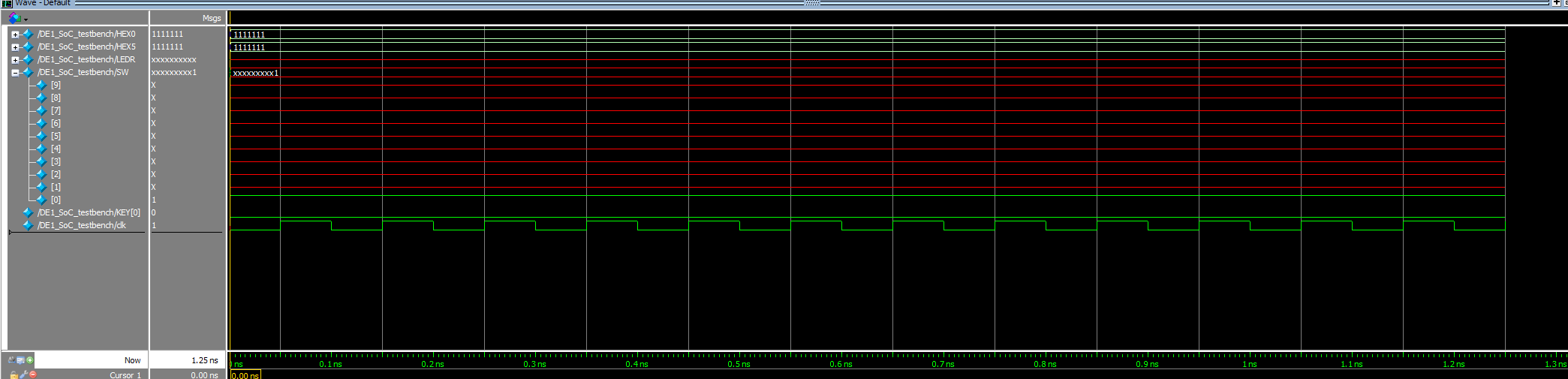
****

Figure 11: The waveform generated by DE1\_SoC

**Size:**

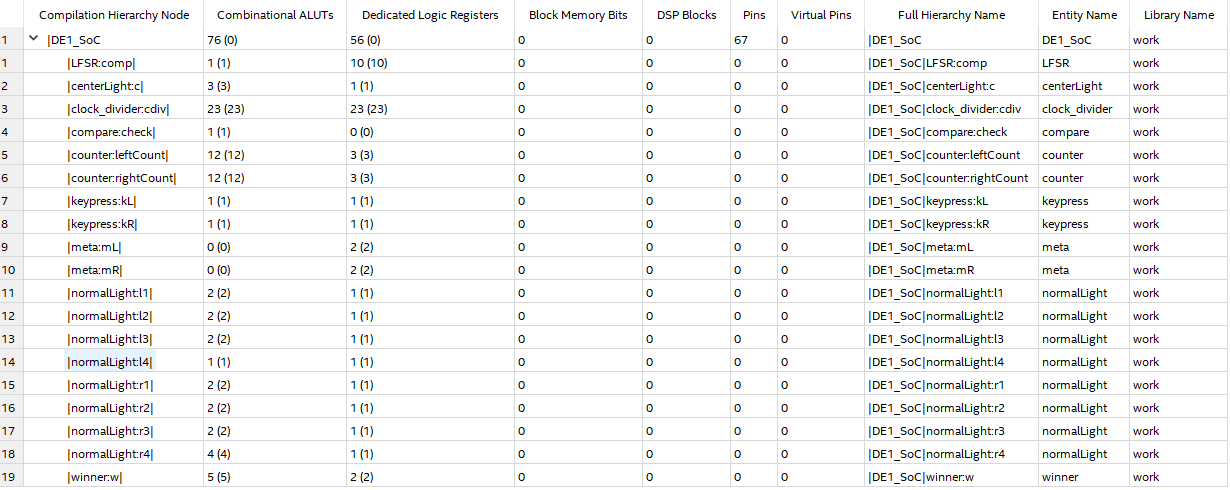
 The size of system is 76+56-23-23, which equals 86. This is the size of the design in terms of FPGA logic and DFF resources subtracted by the clockdivider.

Figure 12: Analysis and Synthesis Resource Utilization of the DE1\_SoC module

**Final Product**

Overall, this project was designed to learn how to develop a system that implements a design of Tug of War against the computer using a series of FSM along with D-Flip-Flops for metastability with the constant user input. For the computer input, this system uses the random outputs of a 10-bit LFSR and compares it to the input of the SW[8:0], and proceeds with the light towards the computer side when the SW input is larger than the LSFR. The project asked to develop a system to control the metastability of the input as well as how an input was perceived. Additionally, with the computer input, this system operates like as asked. Afterwards, the focus is on developing the transition of the LEDR with the user input. I implemented this part with controlling the center light and each of the other normal lights separately and allowing for the input of each light to be from the output of the lights adjacent to it. This allows to track the current state that the light is on and allows for transition with the user input. Lastly, the spec ask for a tracker of each player’s current score and reset the game when one player reaches a score of 7. From the counter module, it implements the display of the current score as well as the ability to reset the entire game when one reaches a score of 7. From what is asked, this project implements all facets of the Tug of War game, allowing the user to reset to the original state as well as controlling metastability of the input with the 768Hz clock. Overall, there are no complications with this project as it allows for seamless play of Tug of War against the computer with the inputs from the FPGA board.

**Appendix**

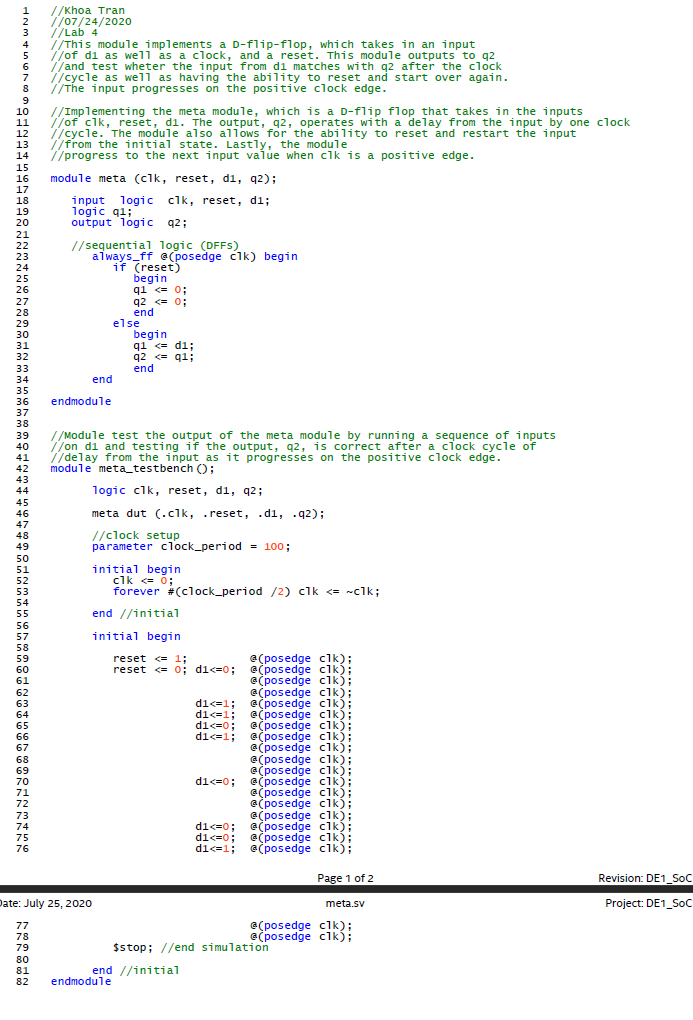


Figure 13: The Meta module

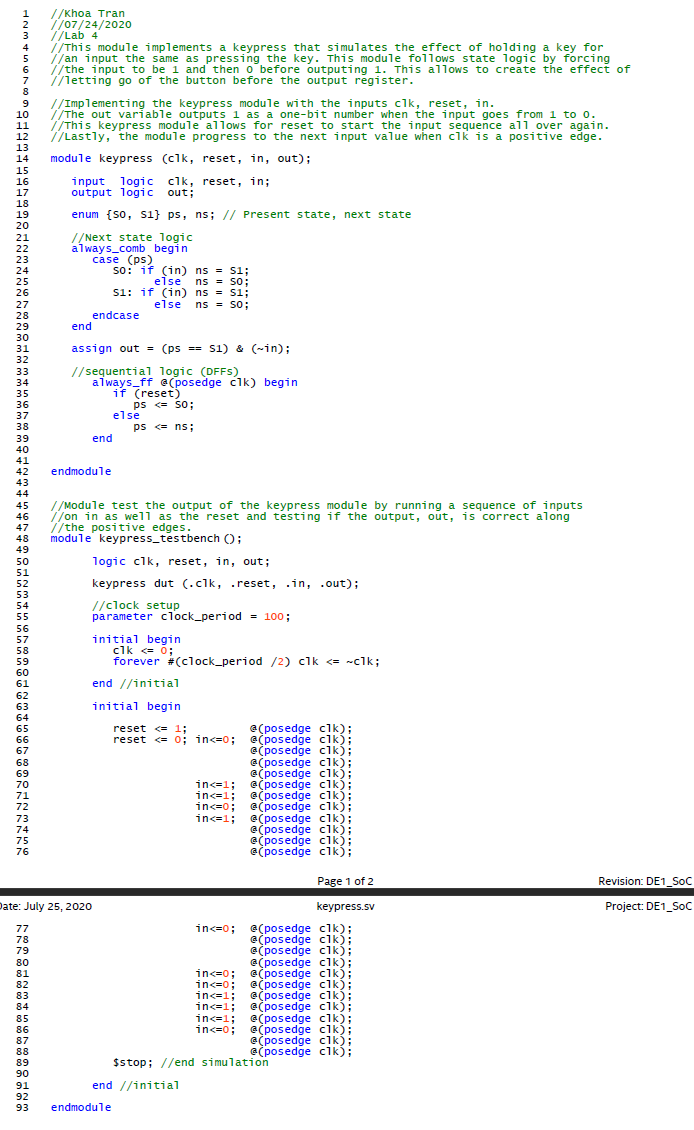


Figure 14: The keypress module

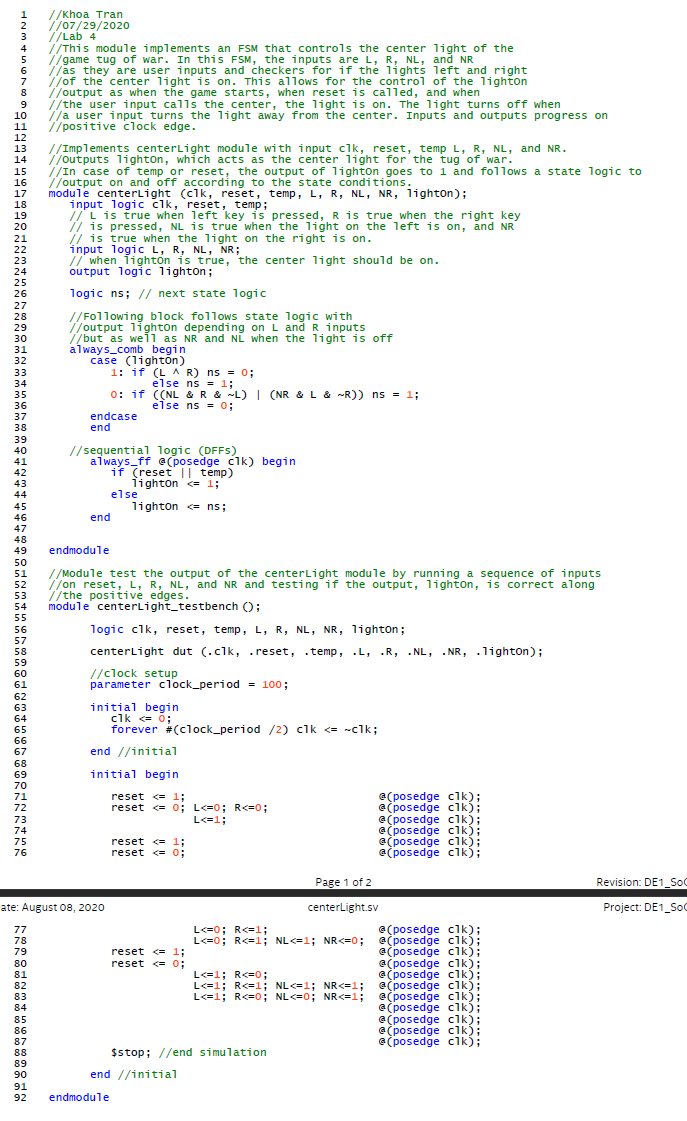


Figure 15: The centerLight module

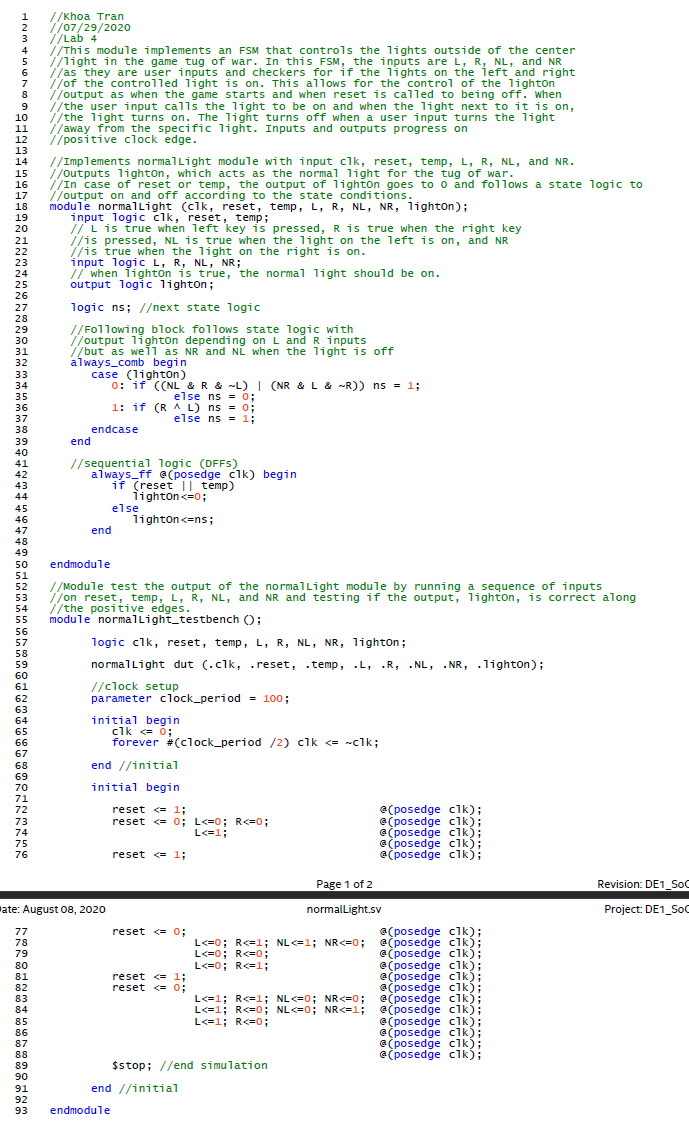


Figure 16: The normalLight module

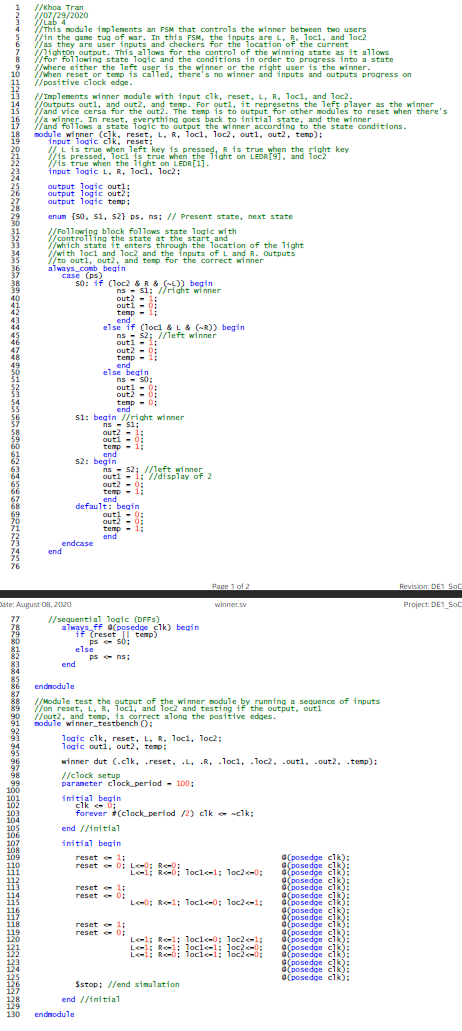


Figure 17: The winner module

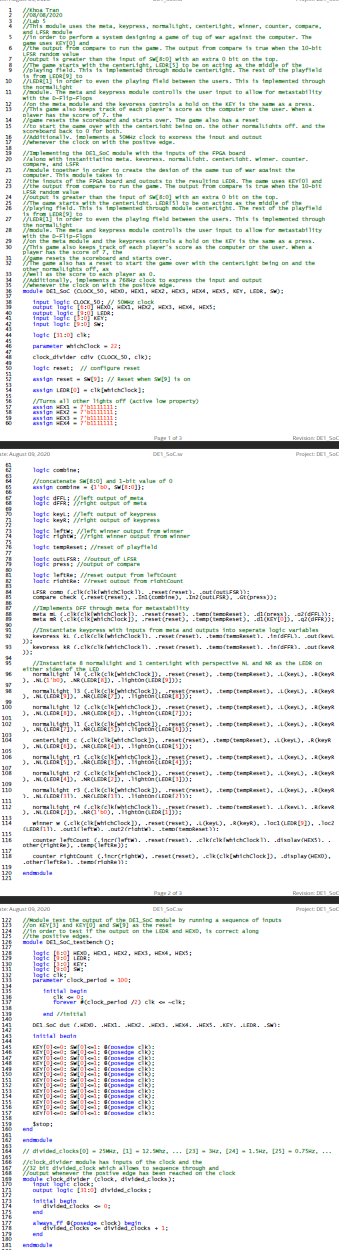
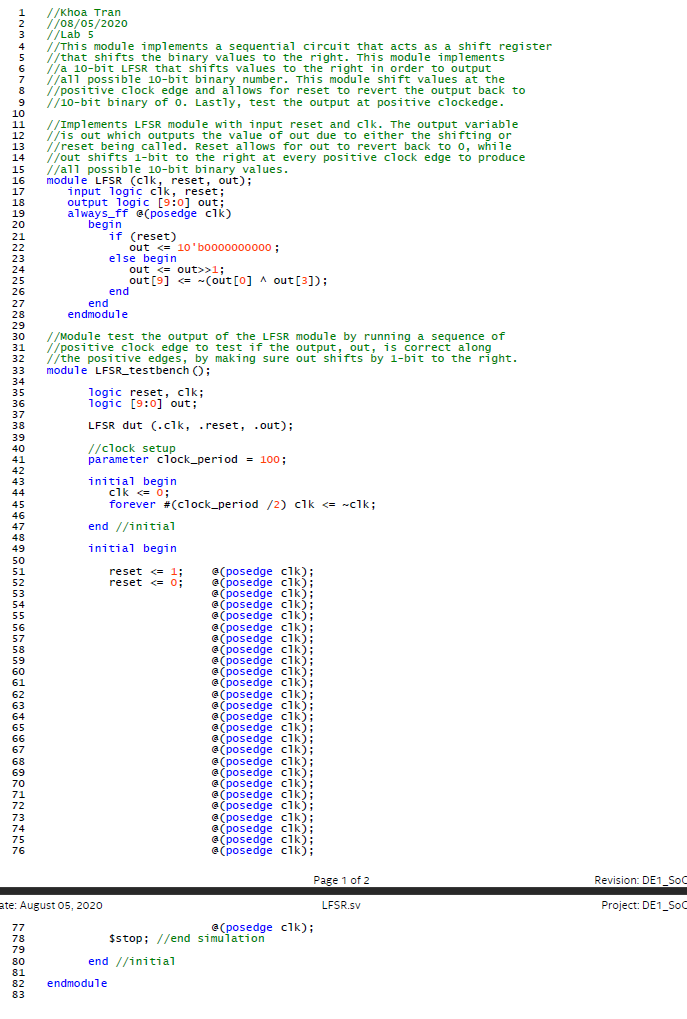


Figure 18: The DE1\_SoC module

Figure 19: The LFSR module

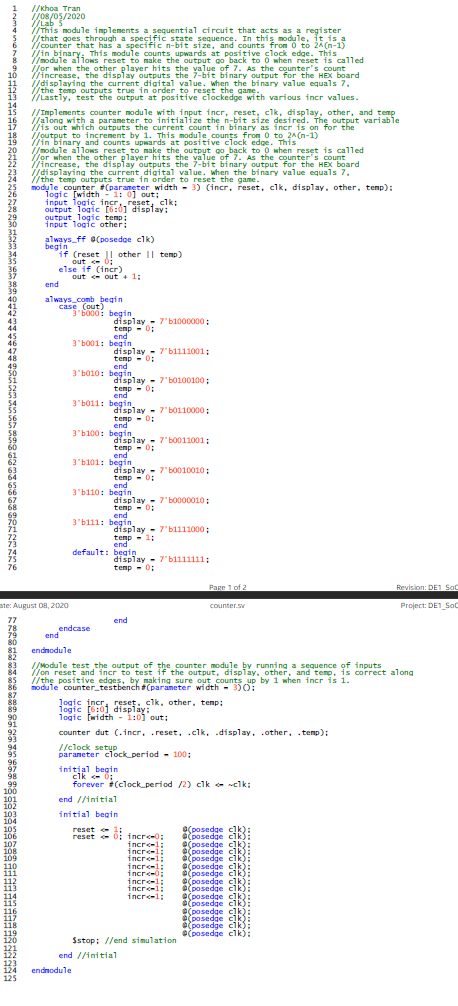
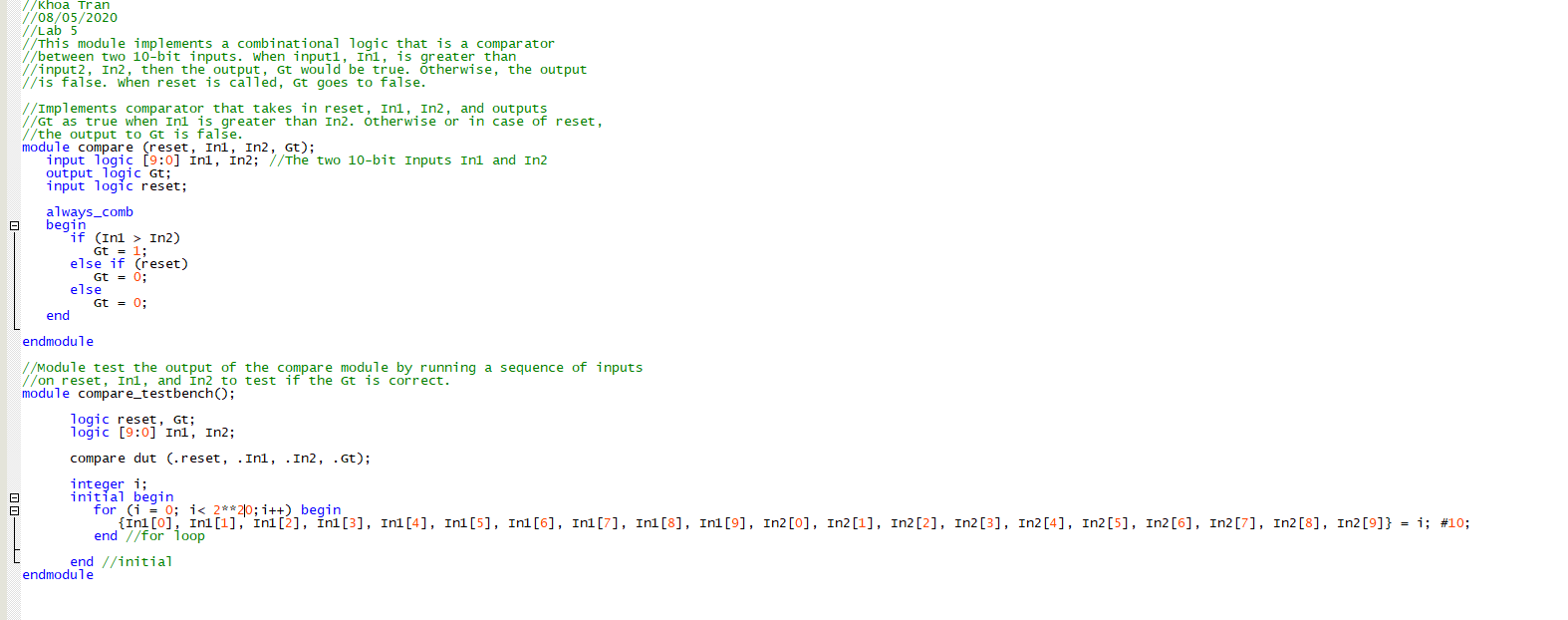


Figure 20: The counter module

Figure 21: The compare module