CMPE 100 Lab Report 6

Leonid Shuster

Lab Section 1C

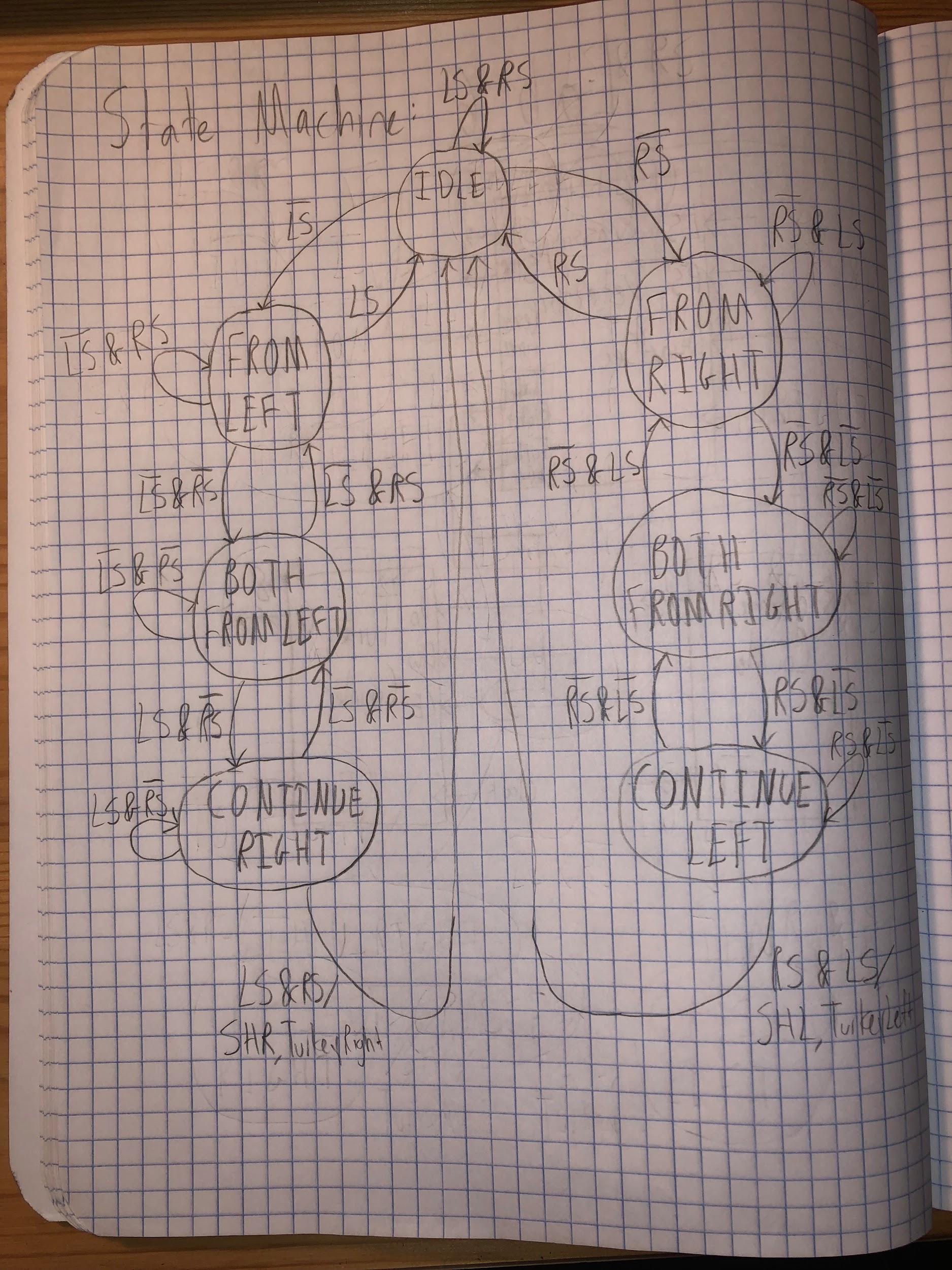
5/26/19

**Description:**

In this lab we were tasked with creating a system that counted the amount of turkeys that crossed from either side of two sensors and displayed the result on the two rightmost 7-segment displays. The sensors were simulated with the pushbuttons of the Basys3 board, where if a button was being pressed down it meant that the sensor was being disrupted. When a turkey crossed both sensors starting from the left, the count went down and several LEDs on the board began flashing right indicating the direction of crossing, and when it crossed both sensors starting from the right, the count went up and the LEDs on the board started flashing left. In this lab, we furthered our knowledge of implementing state machines into our sequential circuits. The lab consisted of the following parts:

**Part One: State Machine**

In the first part of the lab, we had to create a state machine that took in the inputs of the buttons and figure out whether a turkey had successfully crossed across both sensors and in which direction. The first thing to do was create a state diagram that mapped out every possible scenario and result. The following is the state diagram I created:

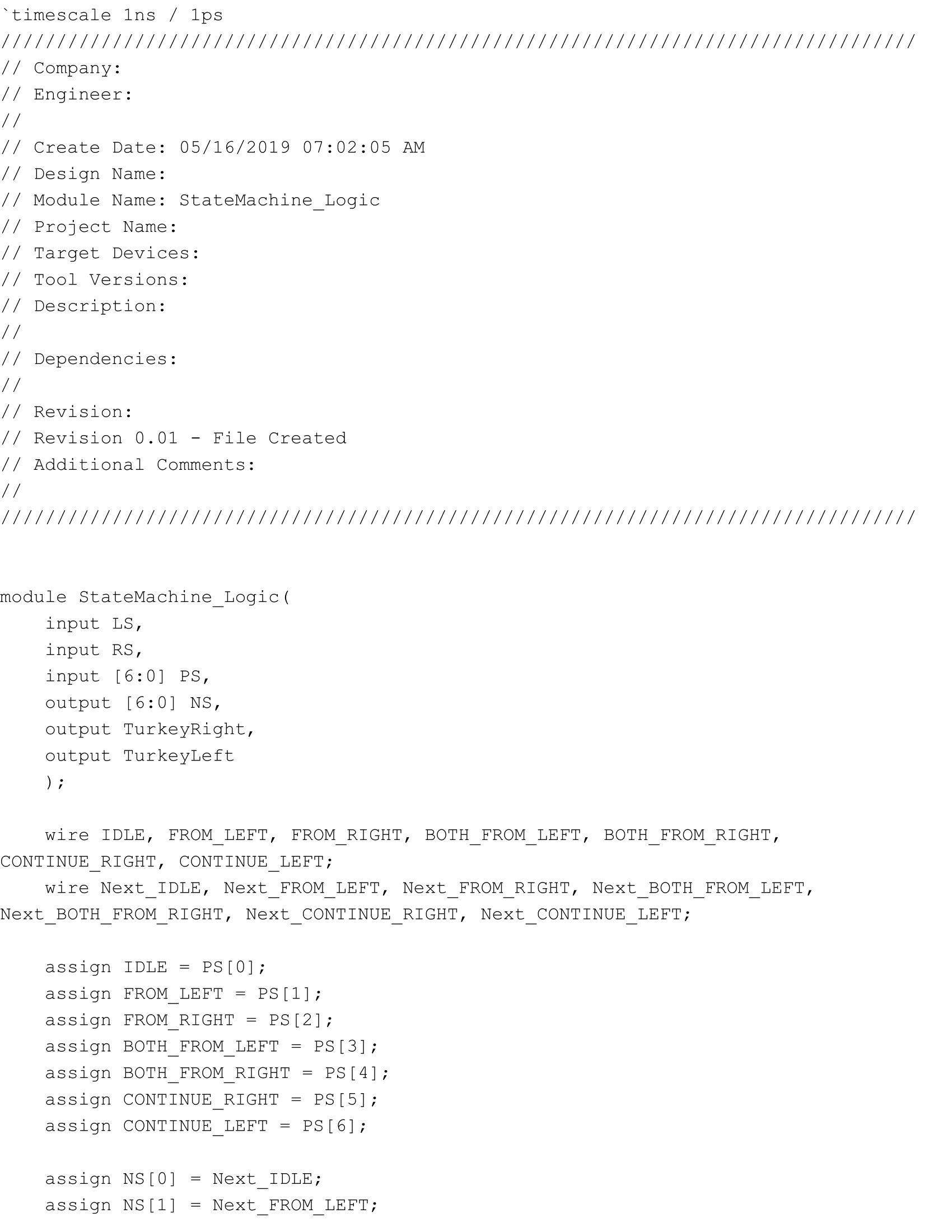


State Diagram

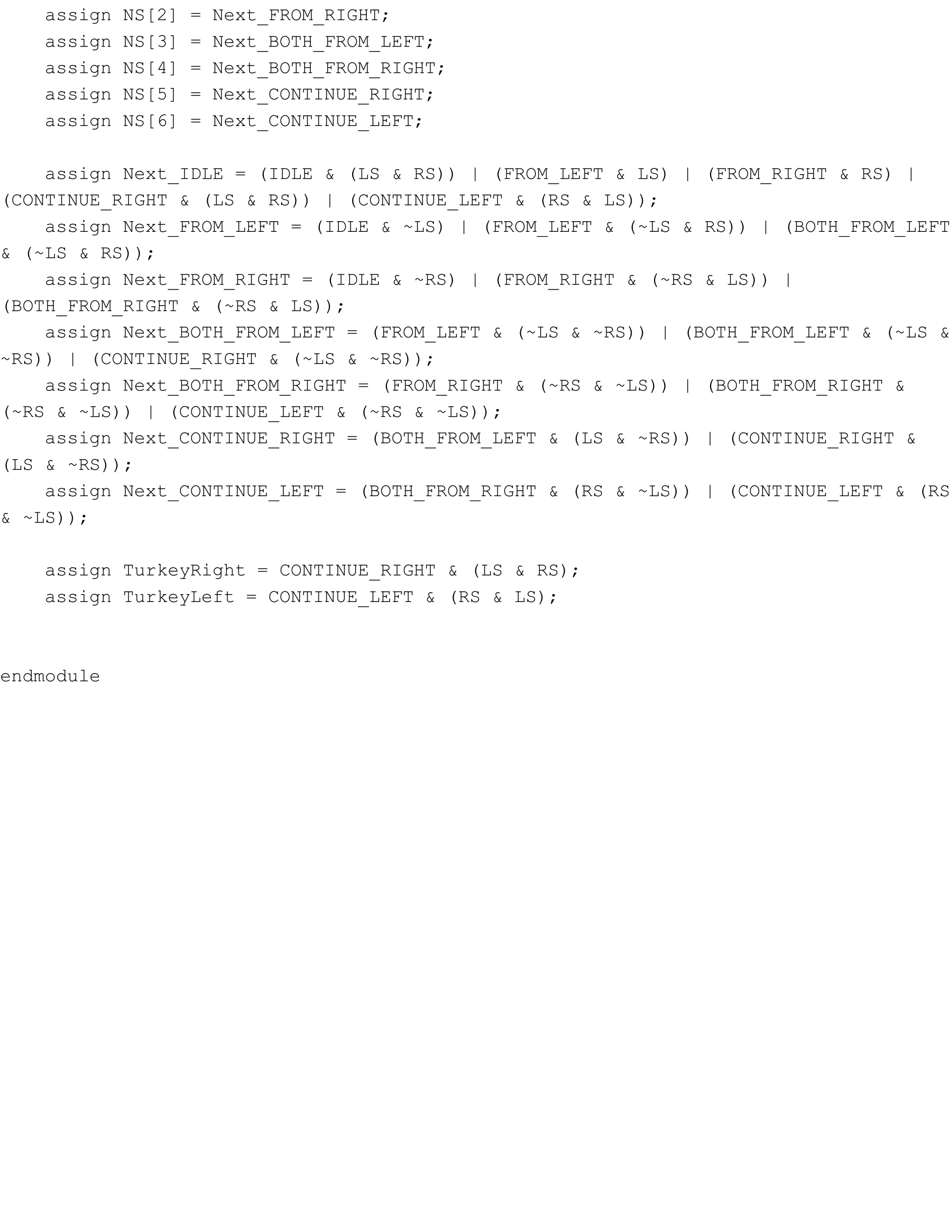
Initially, the system would be in state IDLE. While either button (btnL or btnR) are not pressed, the system continues to stay in IDLE. When either of the buttons are pressed, the following can happen:

1. If btnL is pressed first, the system goes to state FROM\_LEFT. Until the user has not pressed btnL and btnR, the system stays in FROM\_LEFT, or can go back to IDLE if btnL is released. When the user does press btnL and btnR, the system transitions to state BOTH\_FROM\_LEFT. Next, until the user releases btnL while still holding btnR, the system stays in BOTH\_FROM\_LEFT, or can go back to FROM\_LEFT if btnR is released. When the user does release btnL while still holding btnR, the system transitions to state CONTINUE\_RIGHT. Then, until the user releases btnR, the system stays in CONTINUE\_RIGHT, or can go back to BOTH\_FROM\_LEFT if btnL is pressed down again while still holding down btnR. Finally, when the user does release btnR, this means that a turkey has crossed from left to right, and the variable TurkeyRight goes high. The system returns back to state IDLE.
2. If btnR is pressed first, the system goes to state FROM\_RIGHT. While the user has not pressed btnL and btnR, the system stays in FROM\_RIGHT, or can go back to IDLE if btnR is released. When the user does press btnL and btnR, the system transitions to state BOTH\_FROM\_RIGHT. Next, until the user releases btnR while still holding btnL, the system stays in BOTH\_FROM\_RIGHT, or can go back to FROM\_RIGHT if btnL is released. When the user does release btnR while still holding btnL, the system transitions to state CONTINUE\_LEFT. Then, until the user releases btnL, the system stays in CONTINUE\_LEFT, or can go back to BOTH\_FROM\_RIGHT if btnR is pressed down again while still holding down btnL. Finally, when the user does release btnL, this means that a turkey has crossed from right to left, and the variable TurkeyLeft goes high. The system returns back to state IDLE.

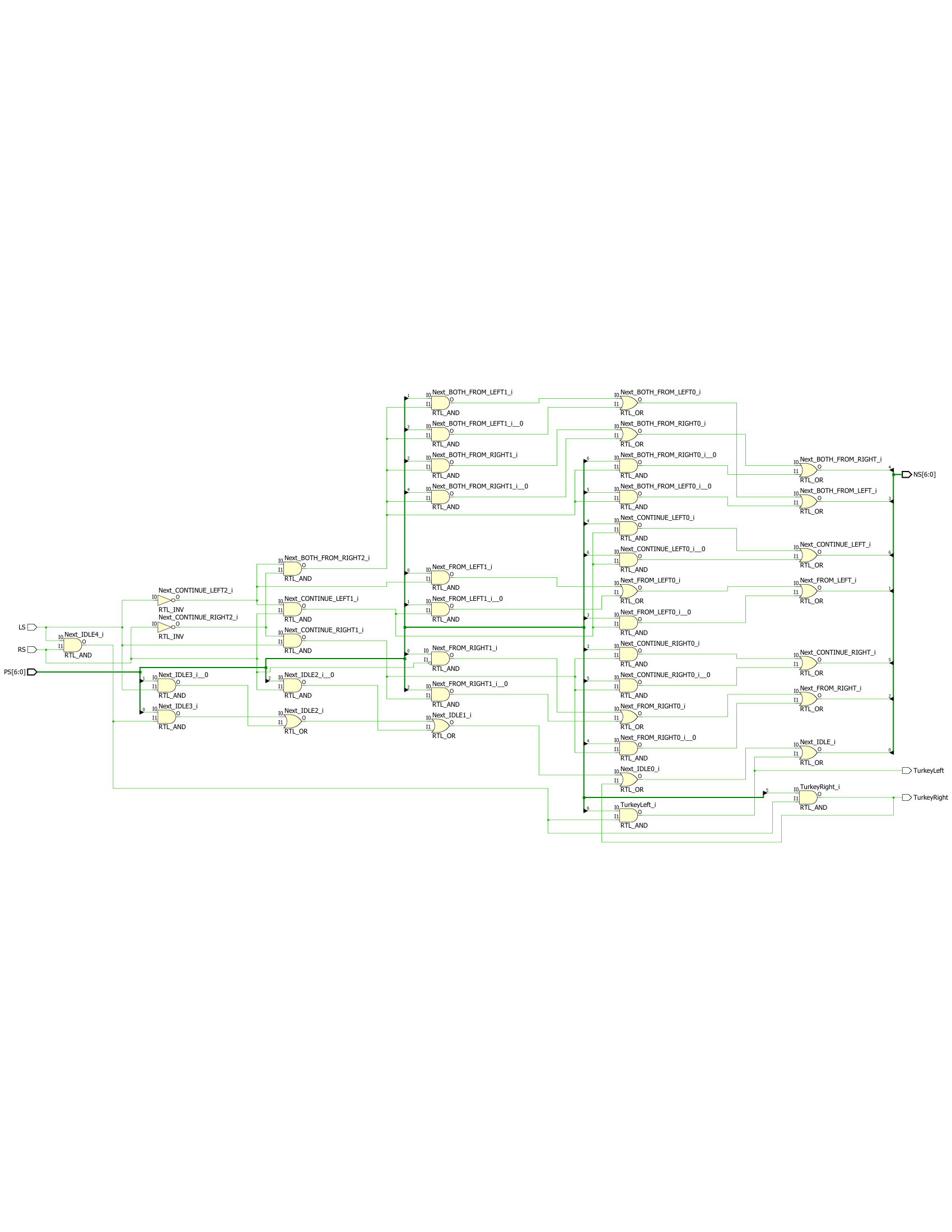
The code and schematic for the state machine’s logic is shown below.



StateMachine\_Logic.v Part 1

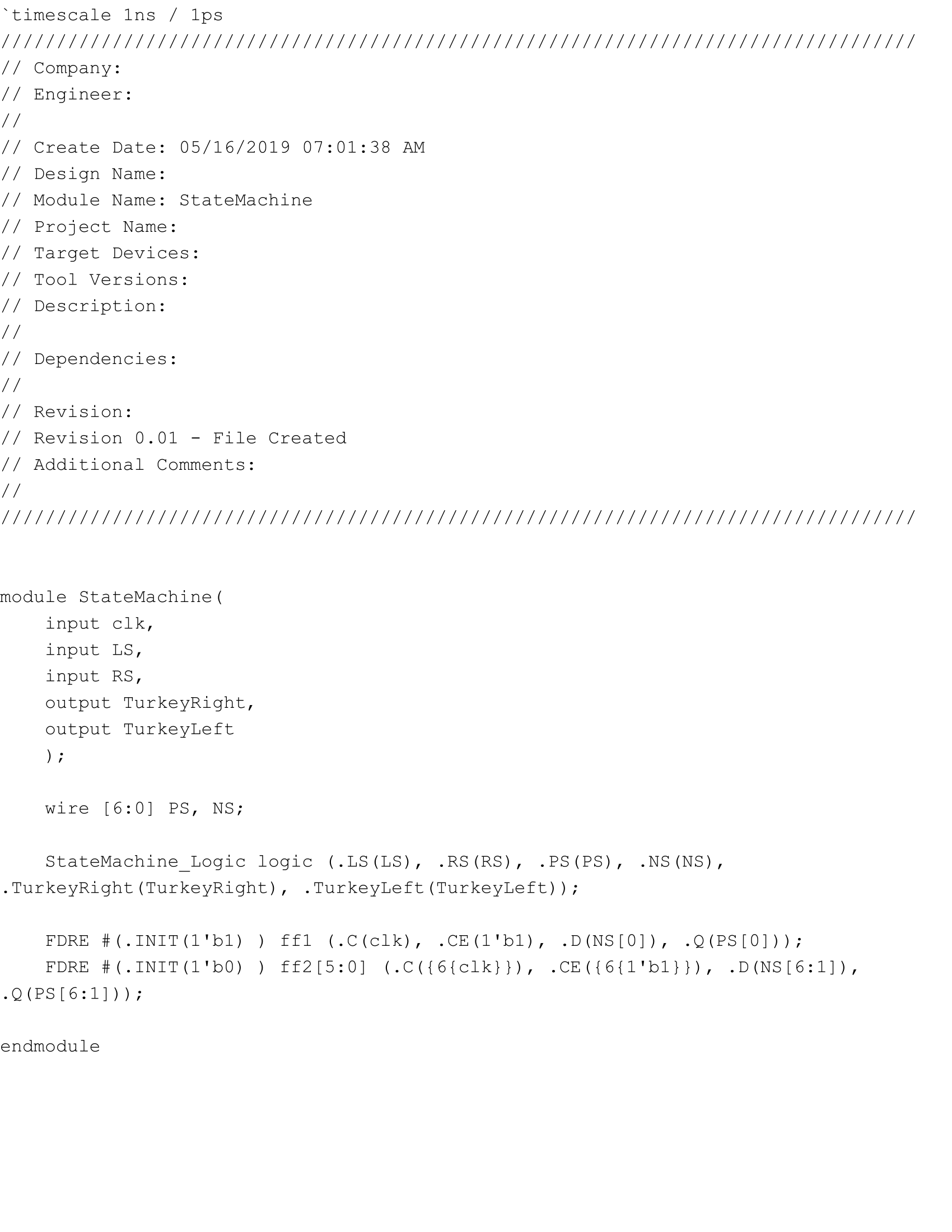


StateMachine\_Logic.v Part 2

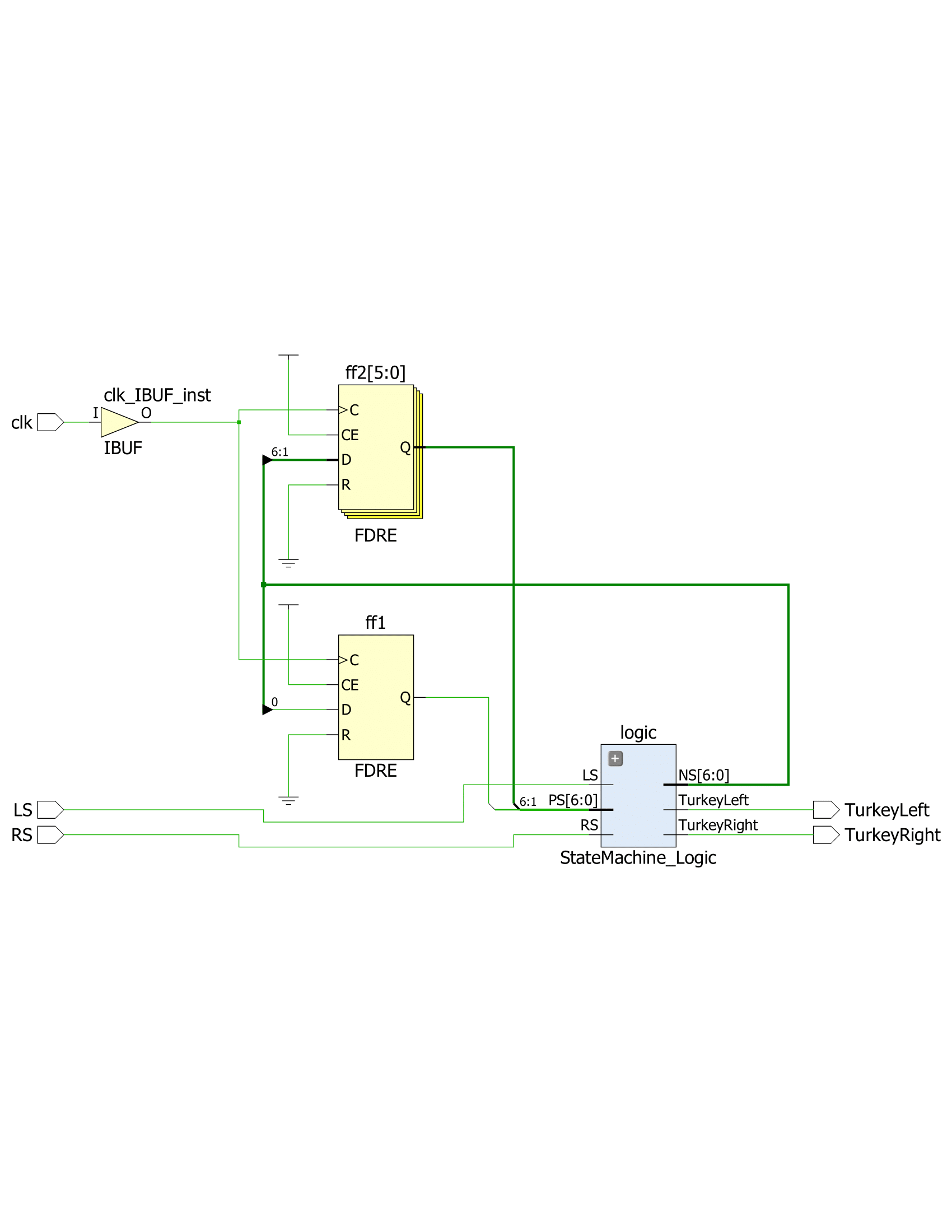


StateMachine\_Logic.v Schematic

To facilitate the inputs and outputs of the state machine, I created a module that makes use of seven flip-flops and one-hot encoding to ensure that only one state happens at a time. The code and schematic are the following:



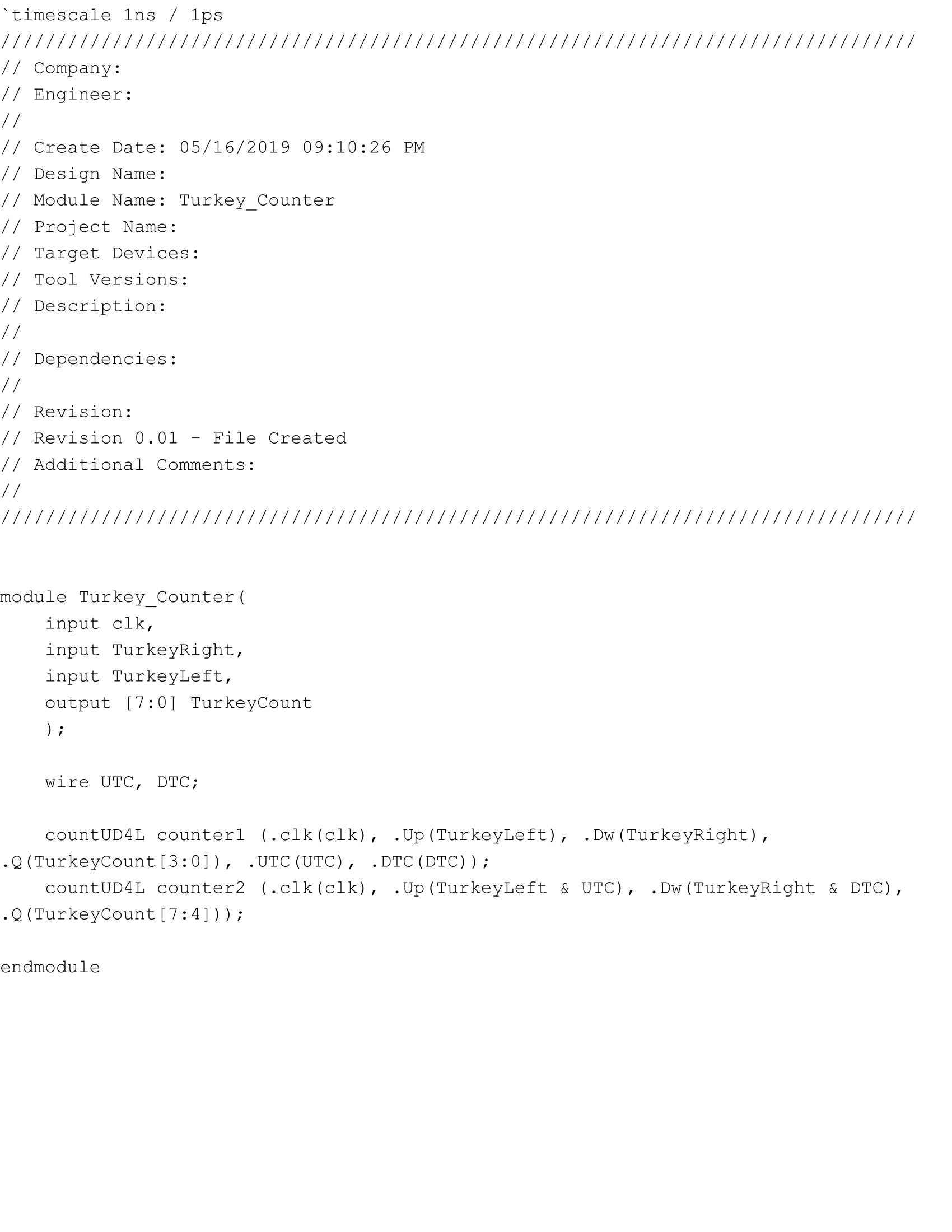
StateMachine.v



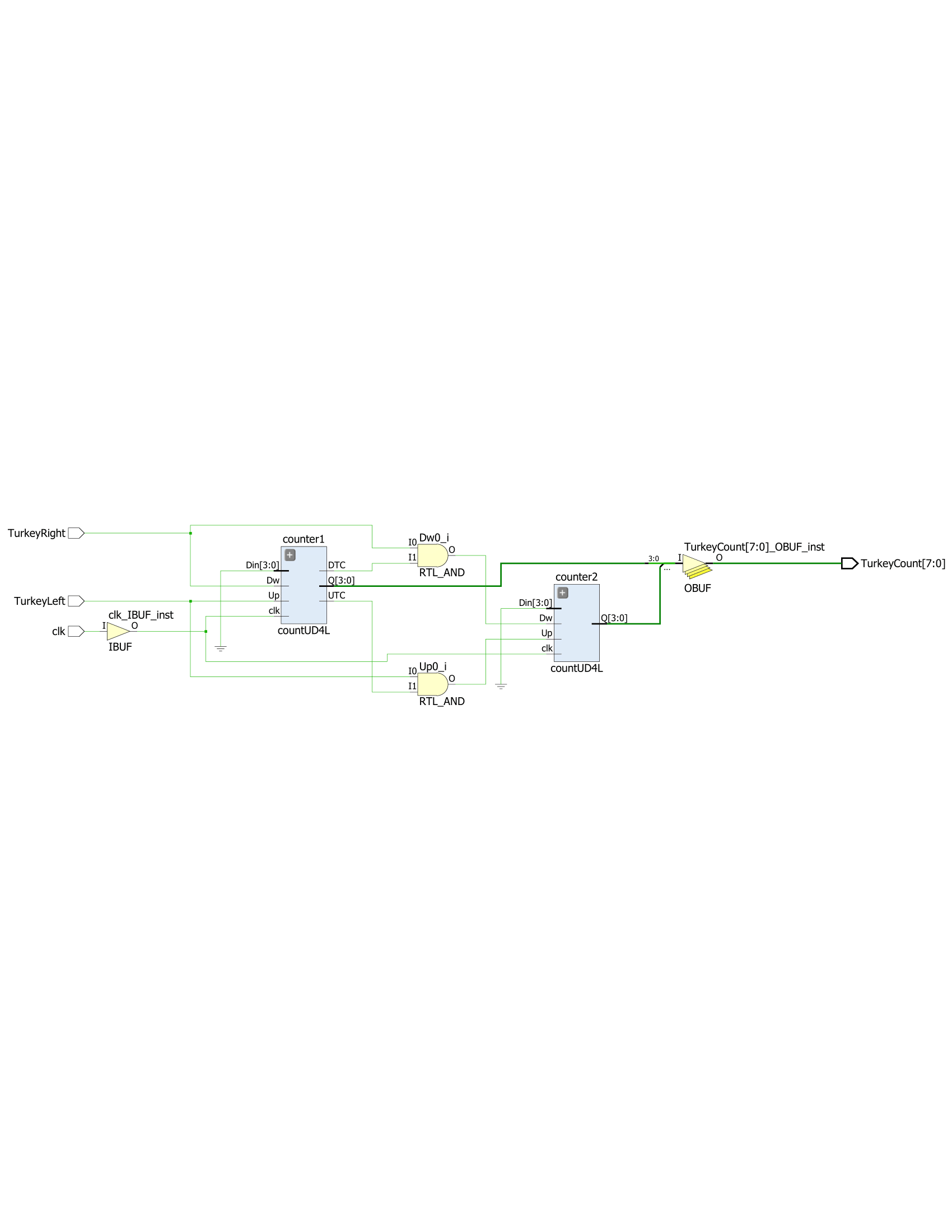
StateMachine.v Schematic

**Part Two: Turkey Counter**

In the next part of the lab, we had to create a counter that counted up or down depending on which direction a turkey had crossed. If a turkey crossed from left to right, the counter decremented, while if a turkey crossed from right to left, the counter incremented. In order to implement the counter, I simply used the same counters from a previous lab and set them up so that they could count up or down up to 8 bits. The code and schematic for the counter are the following:



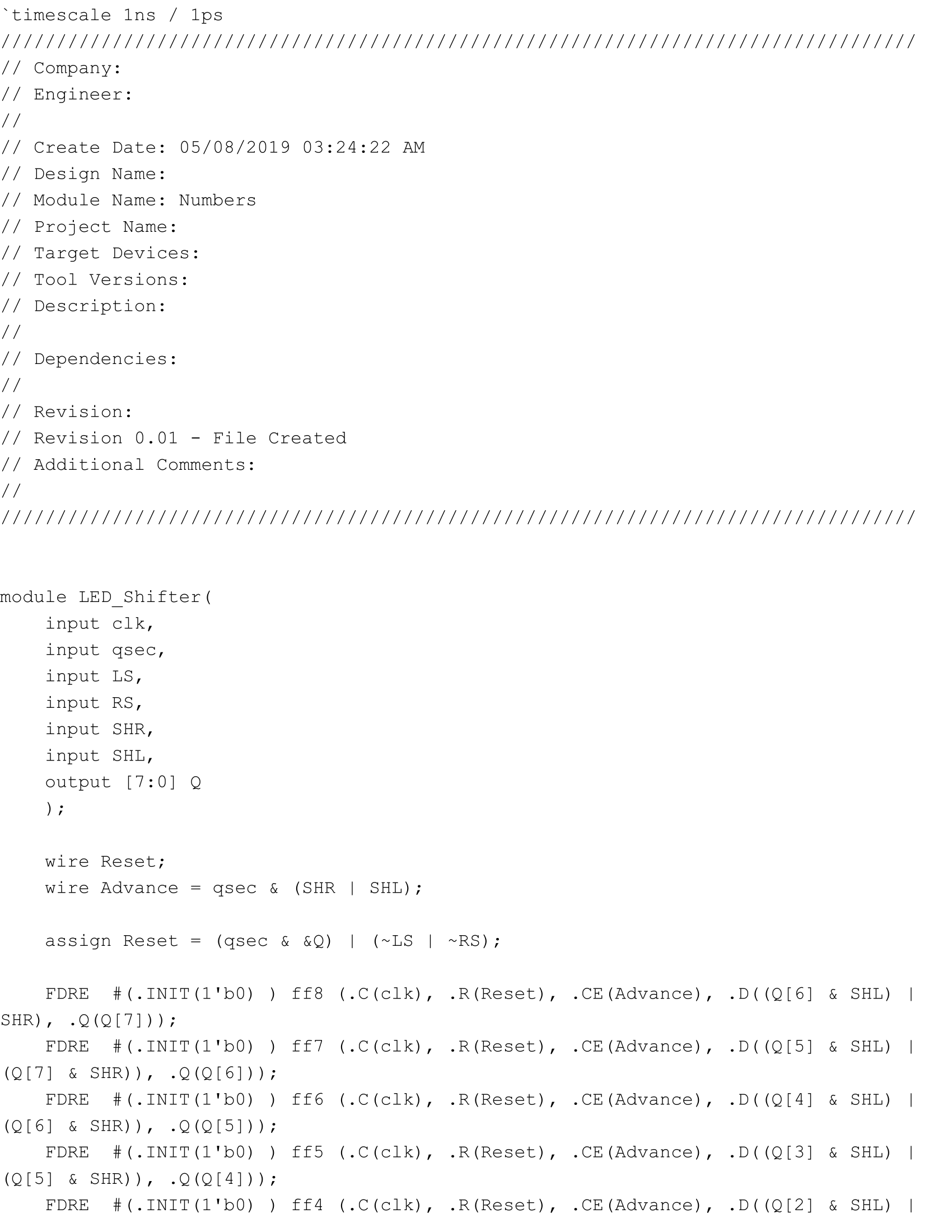
Turkey\_Counter.v



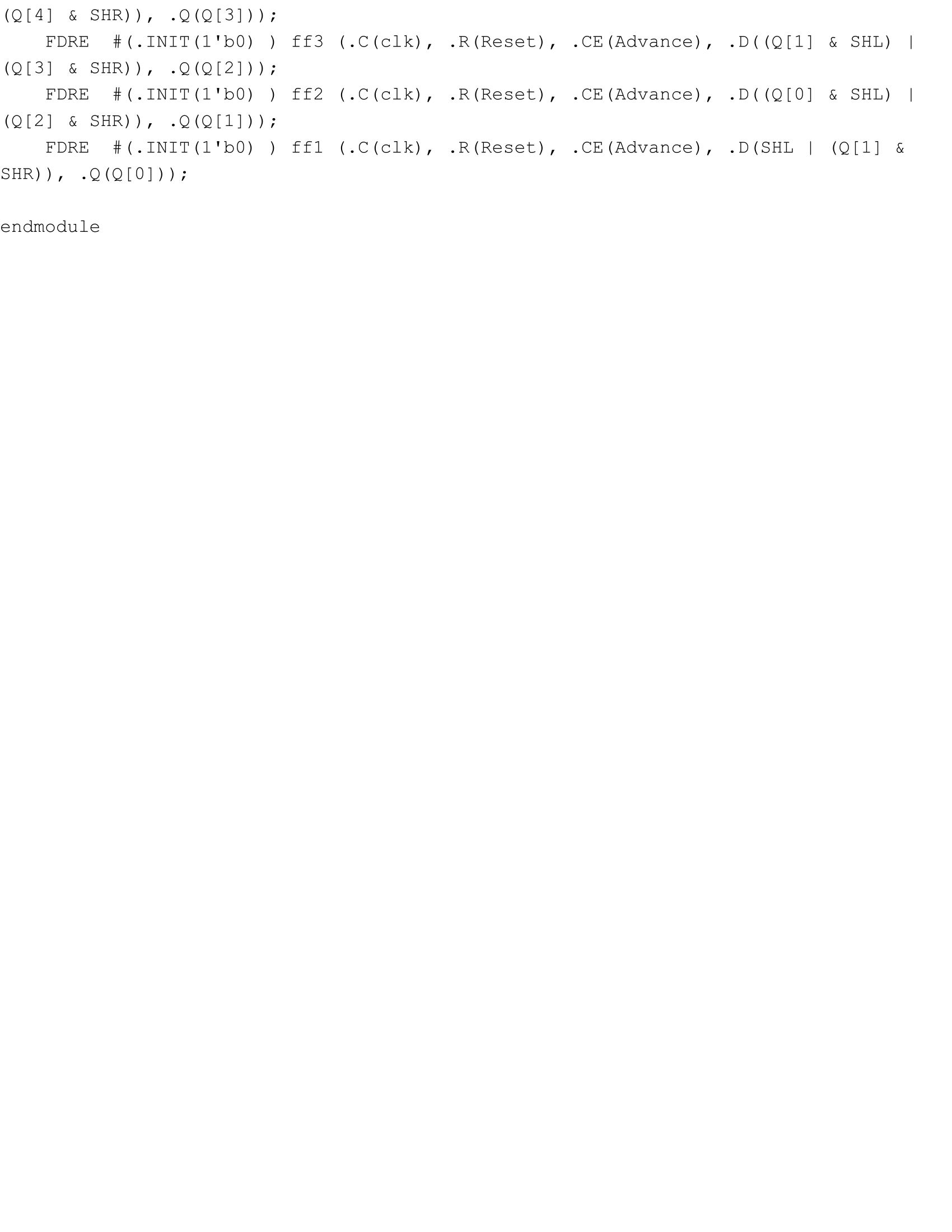
Turkey\_Counter.v Schematic

**Part Three: LED Shifter**

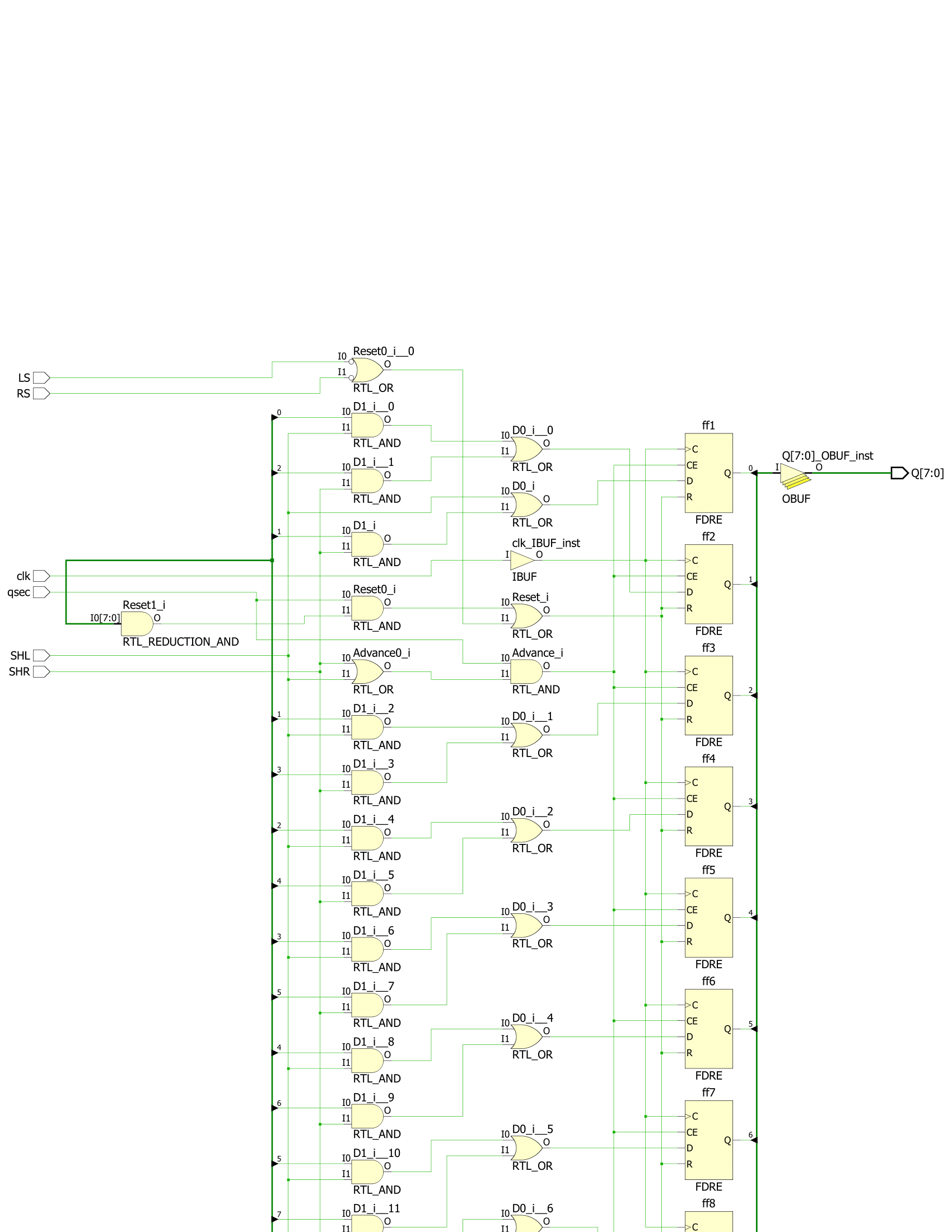
In the next part of the lab, we had to create an 8-bit shifter for the 8 rightmost LEDs that either shifted from the right or left at the rate of 0.25 seconds that would be used to indicate the direction of the previous crossing. To do so, I reused the LED shifter from the previous lab, but now used 8-flip flops for the 8 LEDs that shifted the left or right bit into the current one. The LED shifter kept shifting its values until either button was pressed again, but kept shifting the previous turkey’s direction if both buttons were released again. This was done by tying the reset of the flip-flops to either of the buttons, and have seperate flip-flops in the top-level module that kept the direction of either crossing with two flip-flops. The code and schematic for the LED shifter are the following:



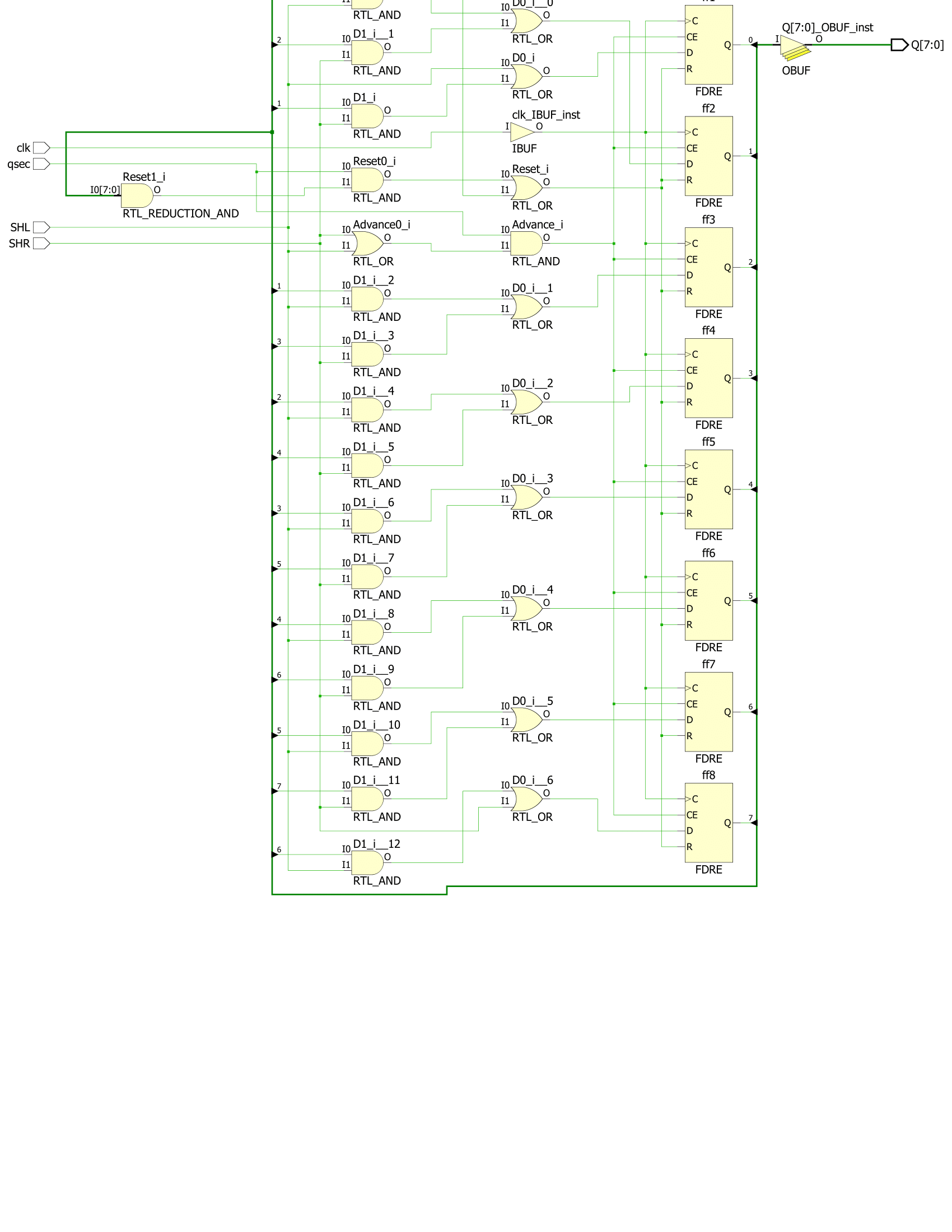
LED\_Shifter.v Part 1



LED\_Shifter.v Part 2



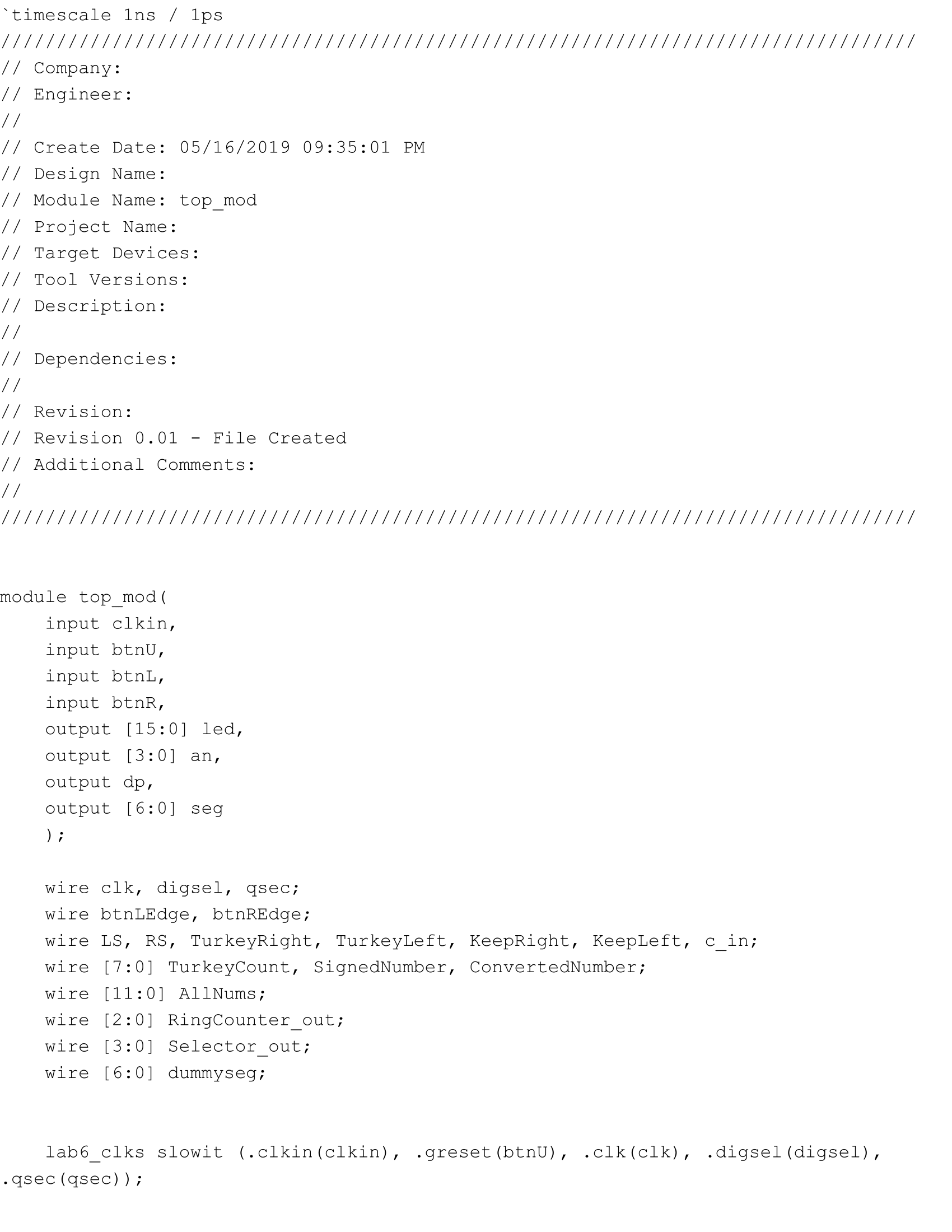
LED\_Shifter.v Schematic Part 1



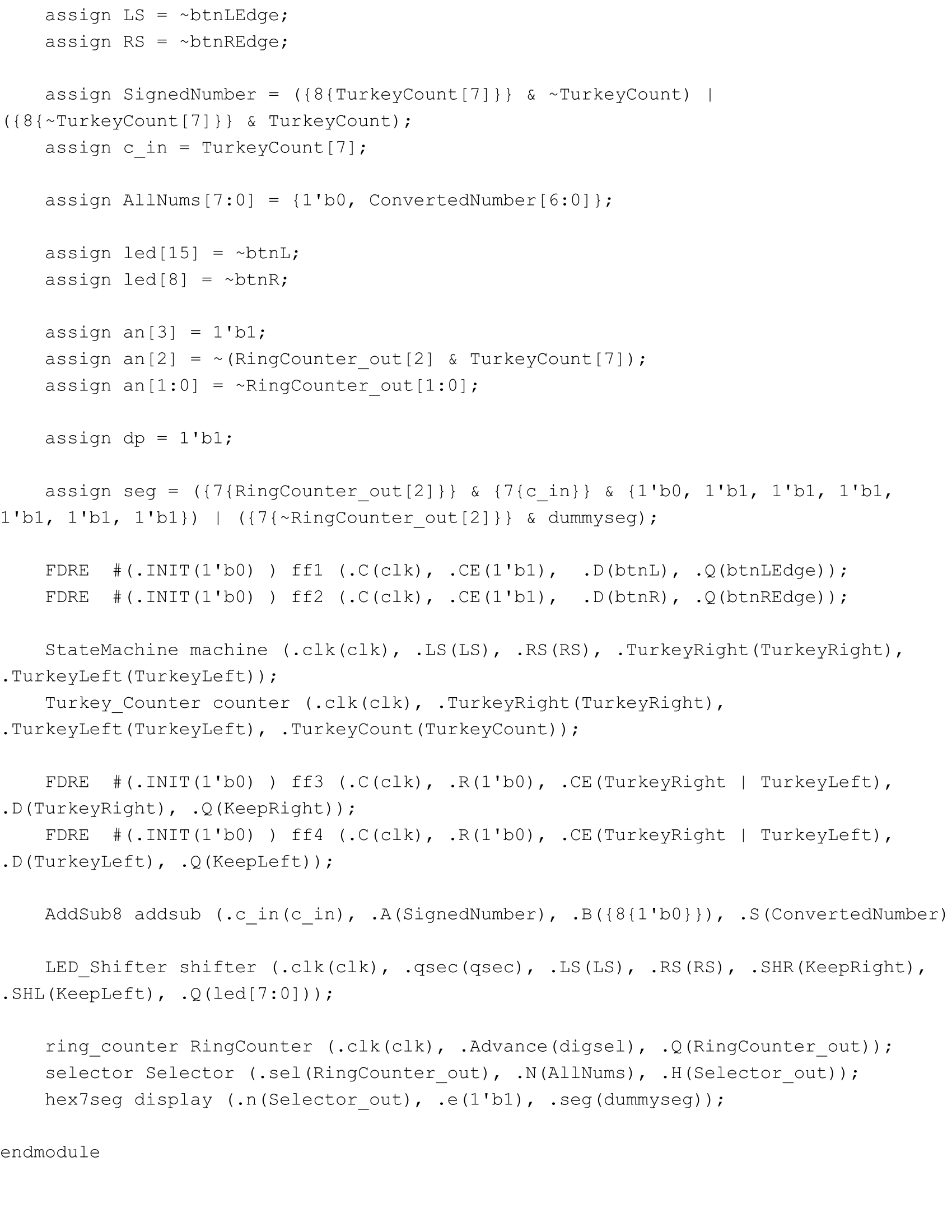
LED\_Shifter.v Schematic Part 2

**Part Four: Top Level Module**

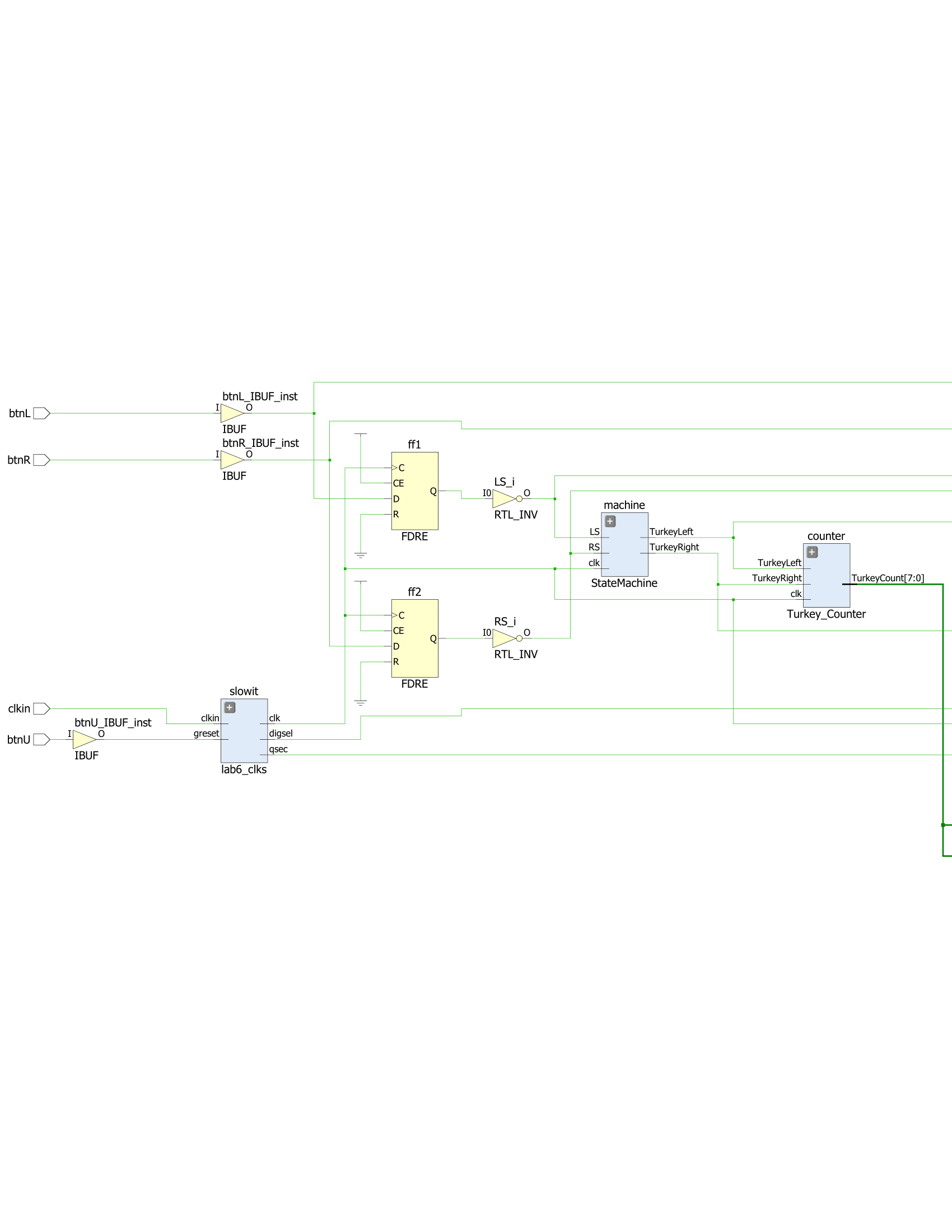
In the next part, everything had to be put together into one file that called every other module. First, the state machine was set up so that it would react to btnL and btnR and output whether a turkey had crossed left, right, or not at all. The result of the state machine was then used to decrement or increment the counter, and to also start shifting the LEDs in either direction until either button had been pressed. The total amount of turkeys from the counter would then need to be displayed onto the Basys3 board. If the total amount were a negative number, such as FF, the result had to show up on the board as 01, with the second leftmost LED displaying a negative sign. To do so, I simply checked if the result was negative, and if so, inverted the number and added one to it by using an adder subtractor from a previous lab, and displayed that number on the board along with a negative sign. Finally, the ring, counter, selector, and hex7seg were used to output the correct numbers at the right time. The code and schematic for the top level module are the following:



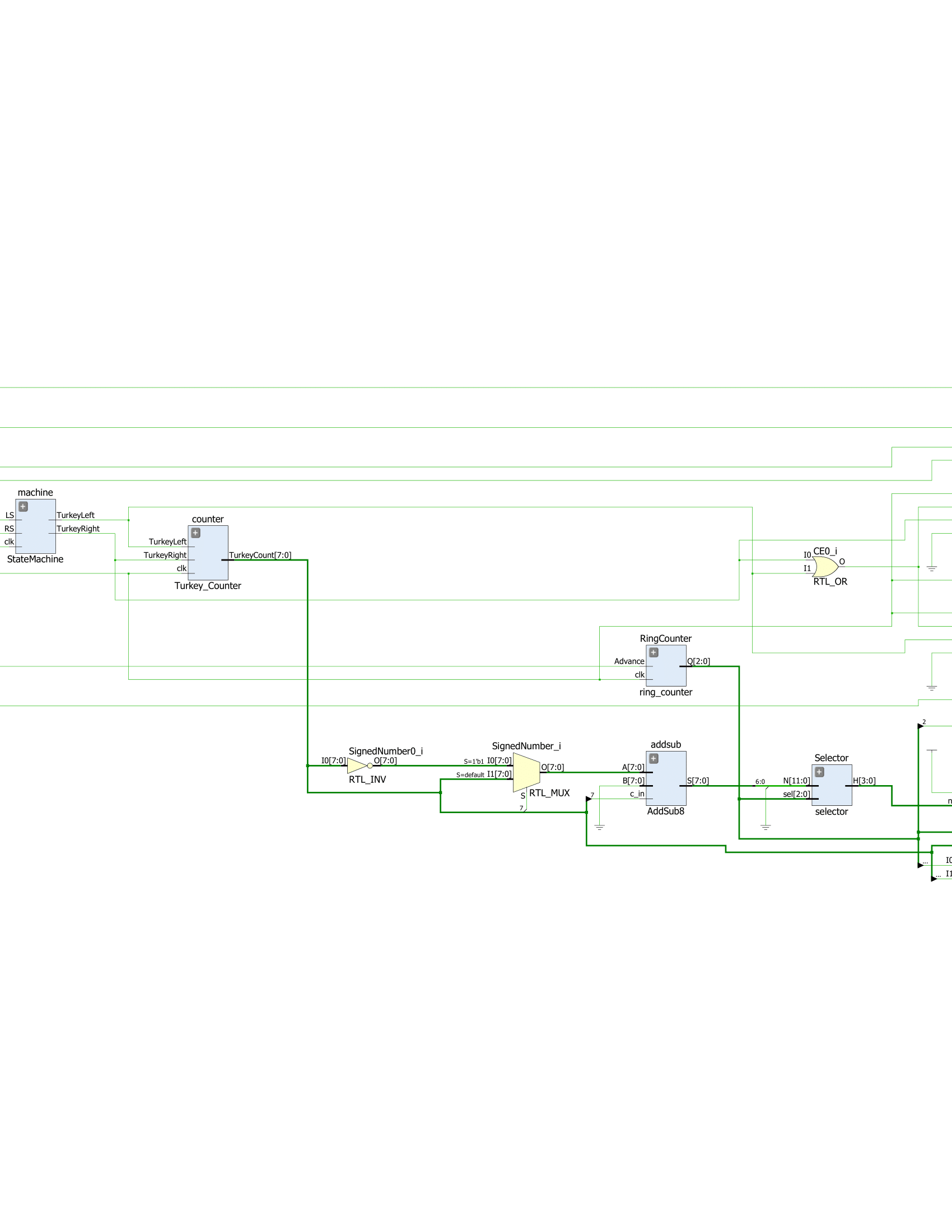
top\_mod.v Part 1



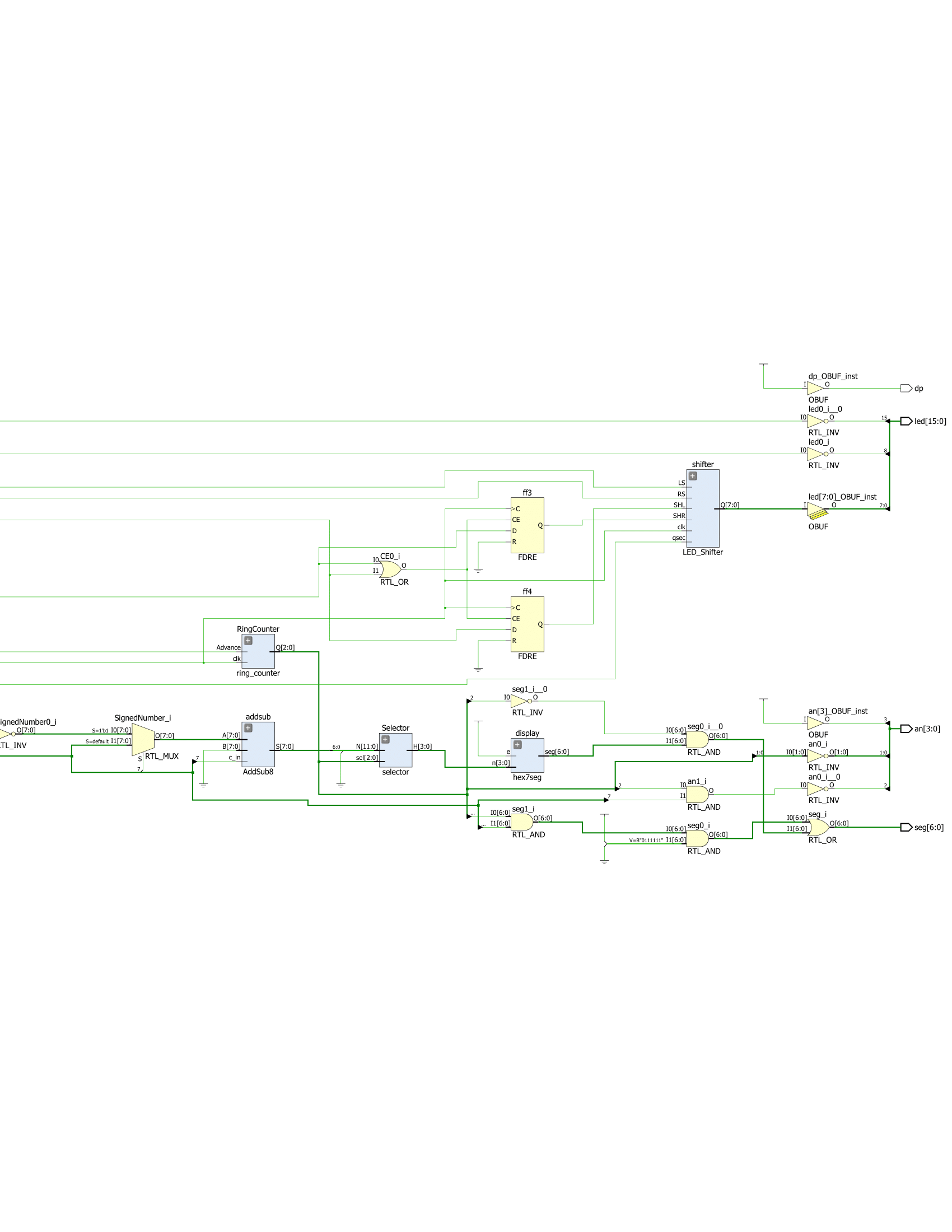
top\_mod.v Part 2



top\_mod.v Schematic Part 1



top\_mod.v Schematic Part 2

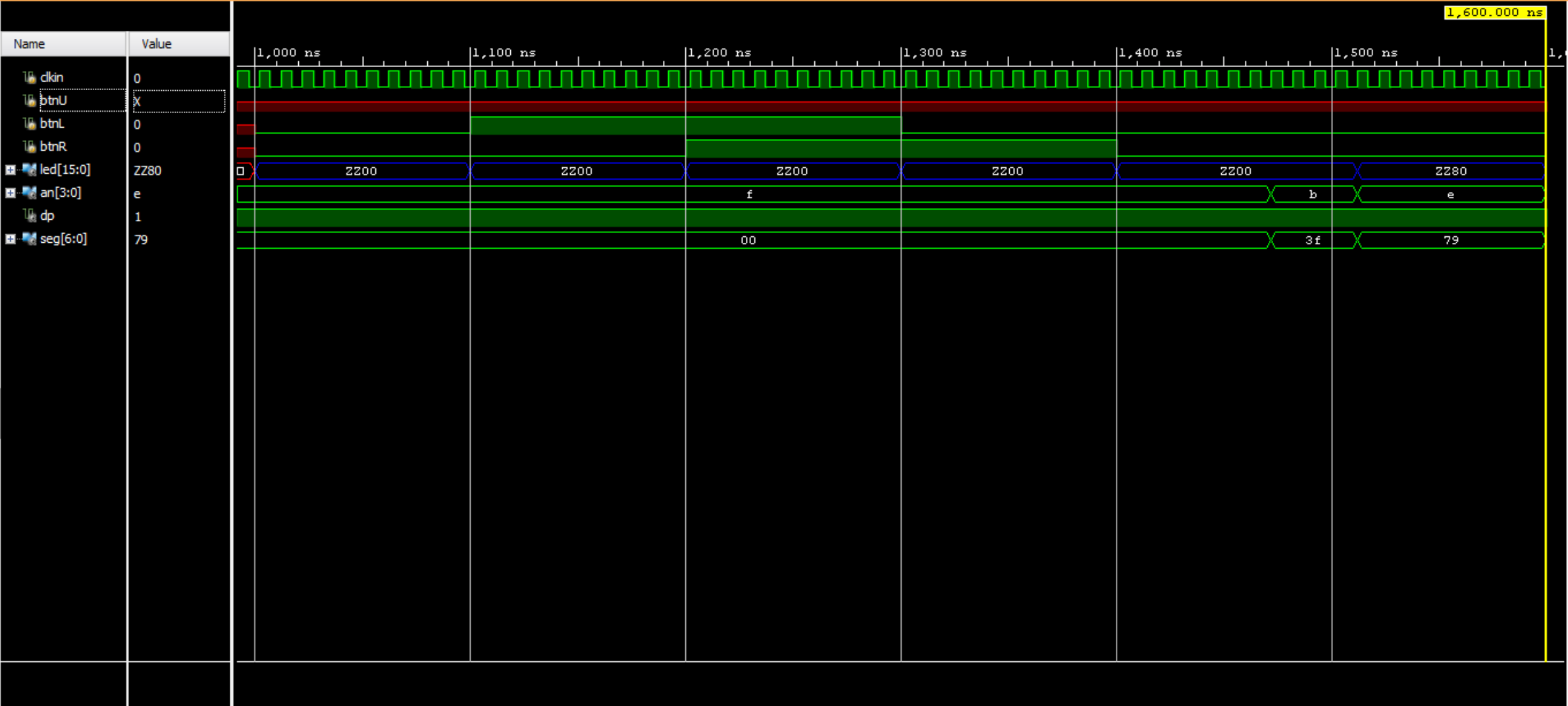


top\_mod.v Schematic Part 3

I then mapped all the outputs of the top level module to the Basys3 board, and verified that the counter did indeed work when turkeys were crossing.

**Part Five: Using the Simulator**

The last part of the lab was to use the simulator to make sure that our top level module worked. I tested the case where a turkey crossed from the left side to the right side, and verified that it worked. The following is the waveform displayed:

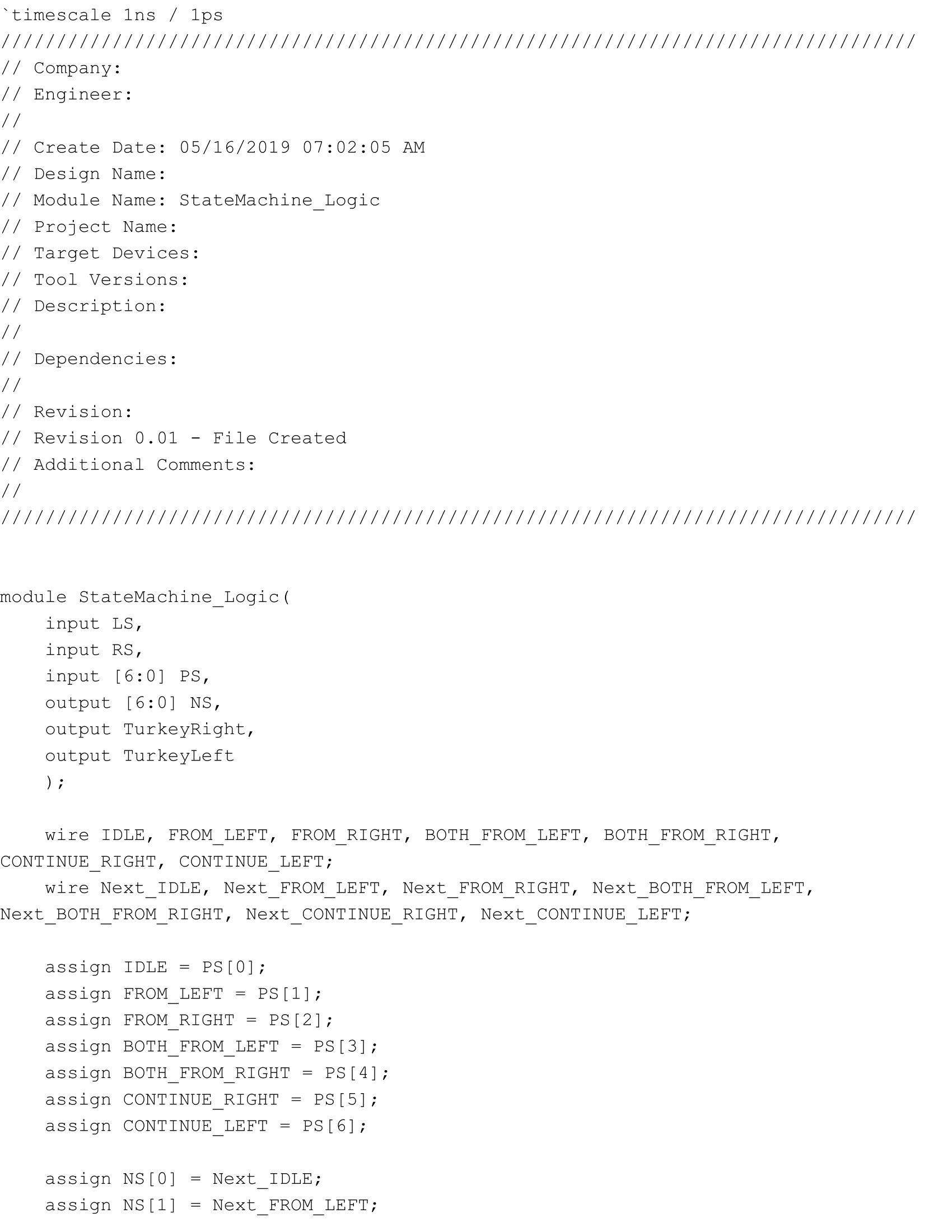


Waveform Viewer

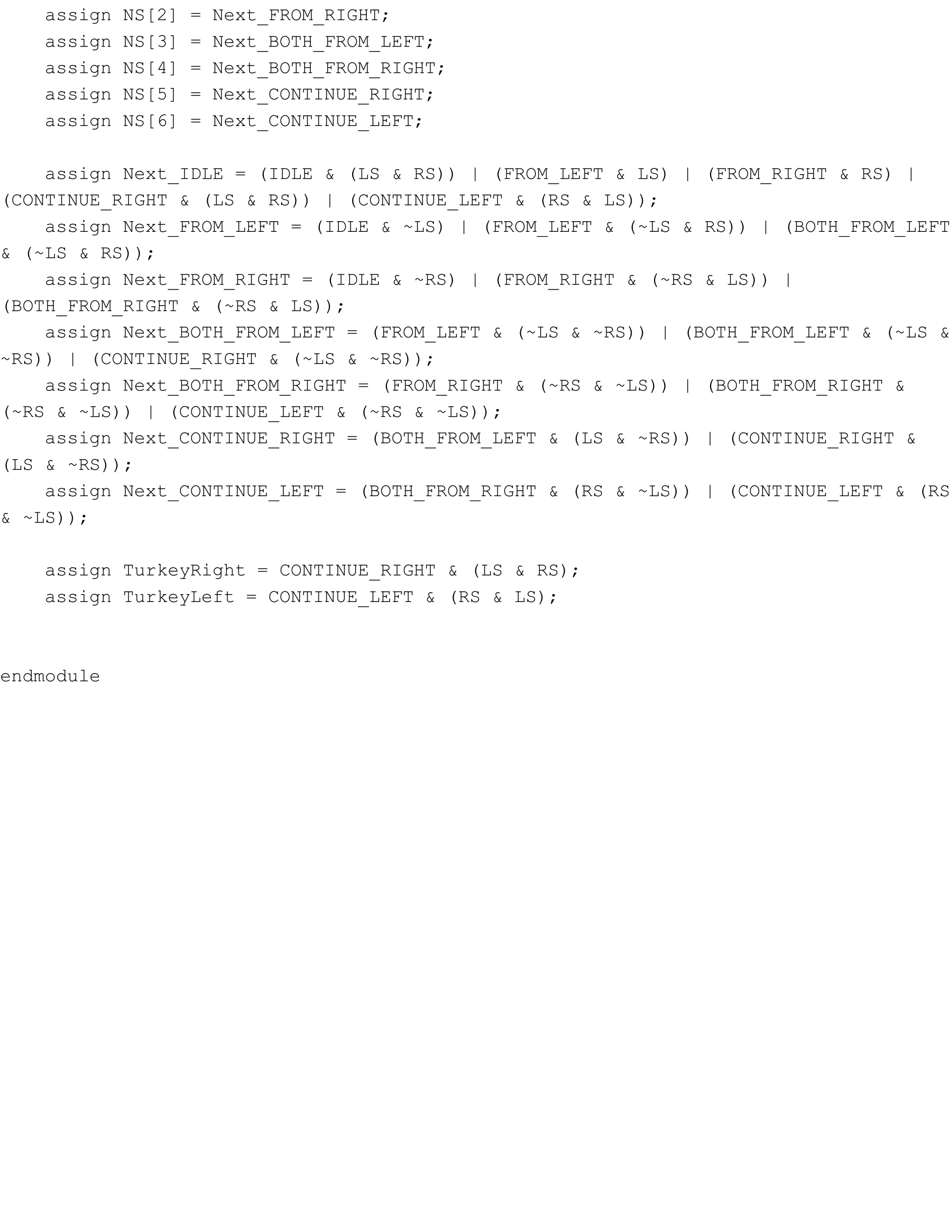
**Conclusion:**

In conclusion, we successfully built a system that counted the amount of turkey crossings, and furthered our ability of implementing state machines into our designs in order to react to certain events, and output the correct actions. If I were doing this lab again, I would make better use of the simulator by testing my code as soon as I write it instead of testing it after creating multiple files with code that could be wrong.

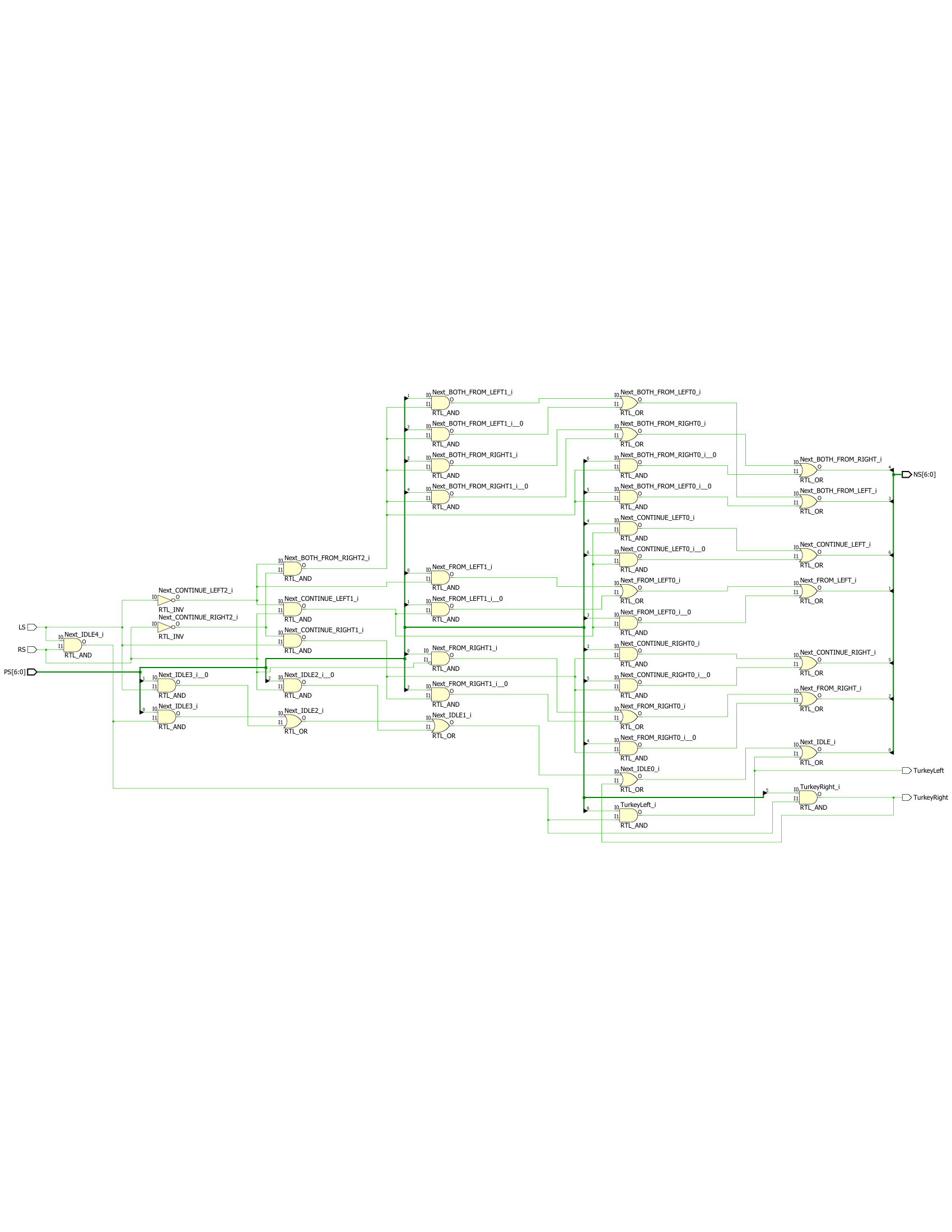
**Appendix**



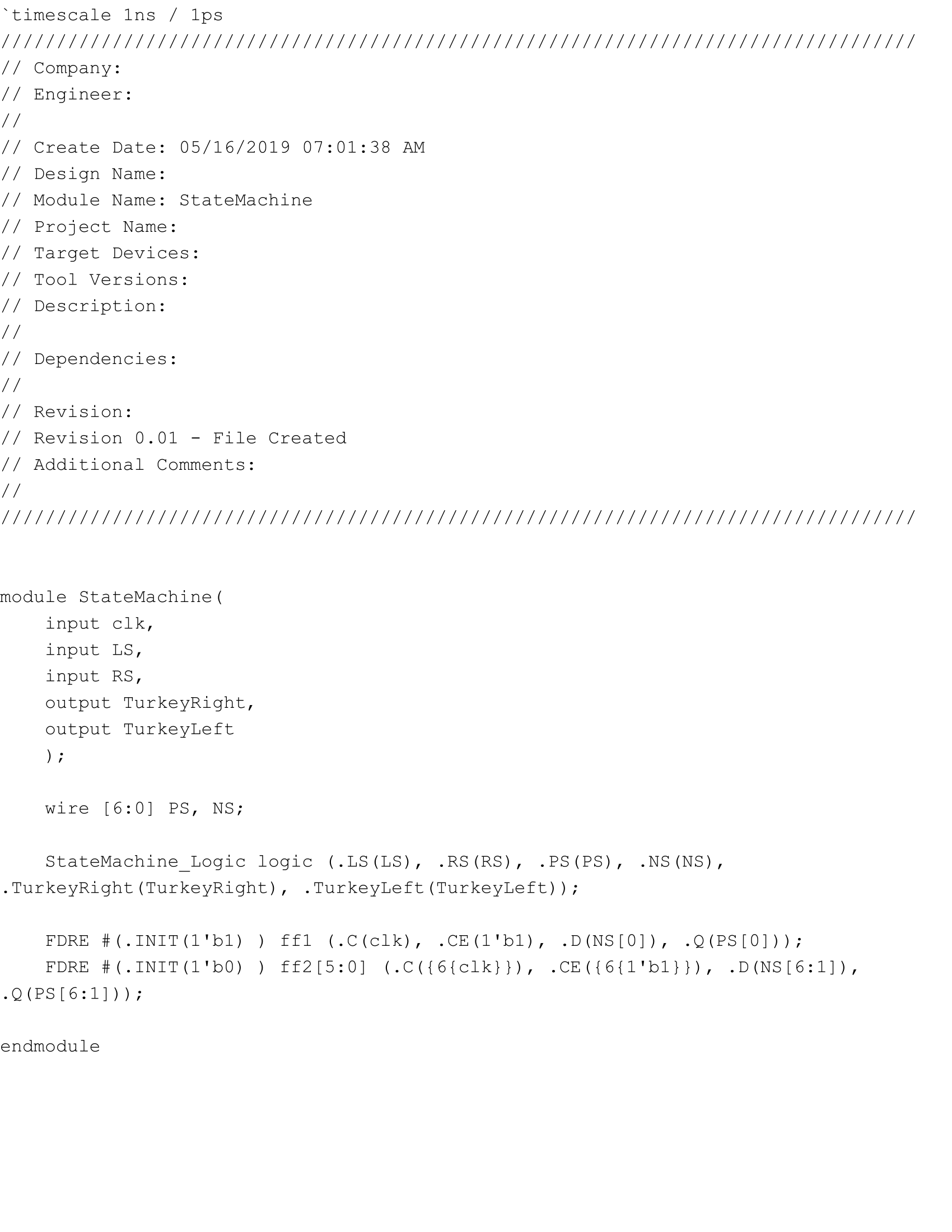
StateMachine\_Logic.v Part 1



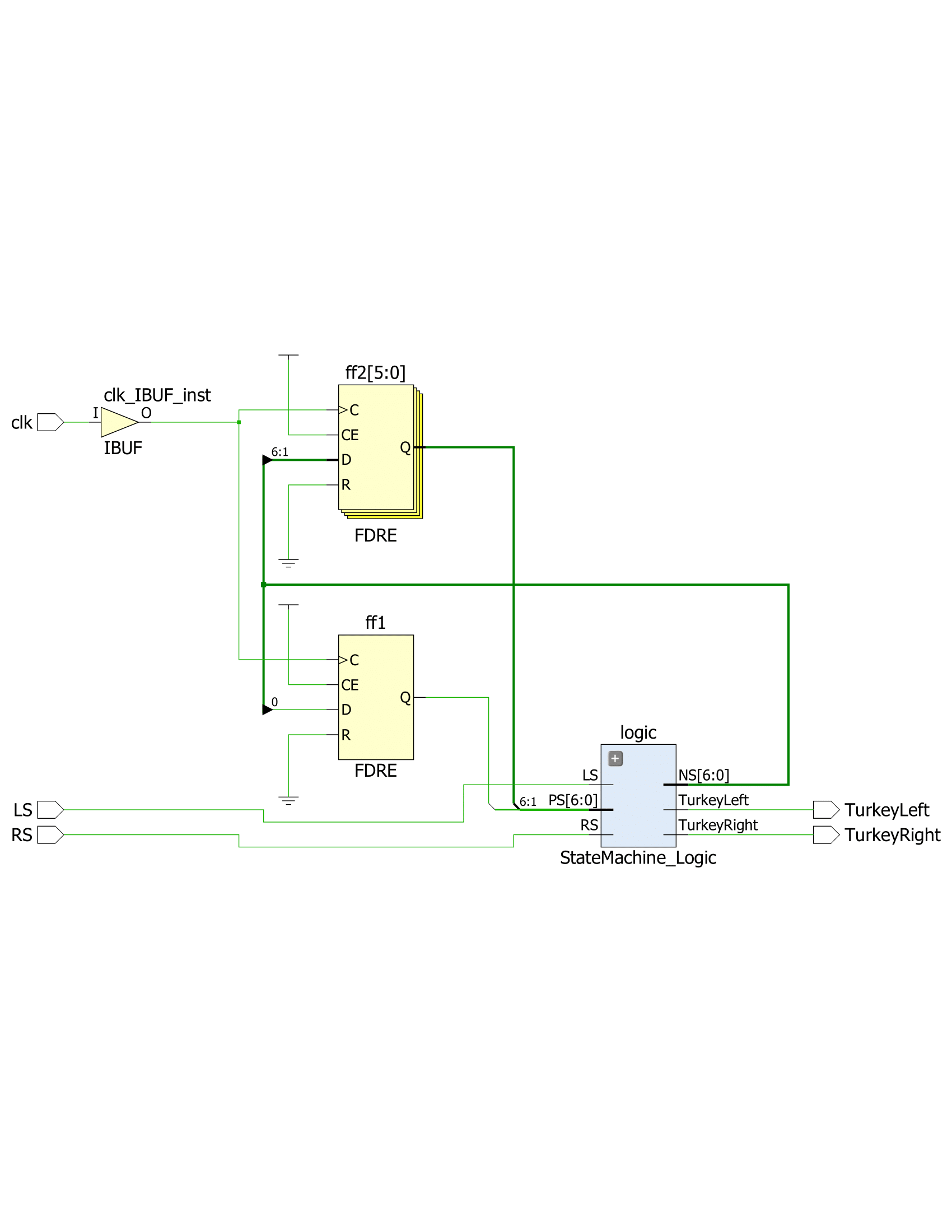
StateMachine\_Logic.v Part 2



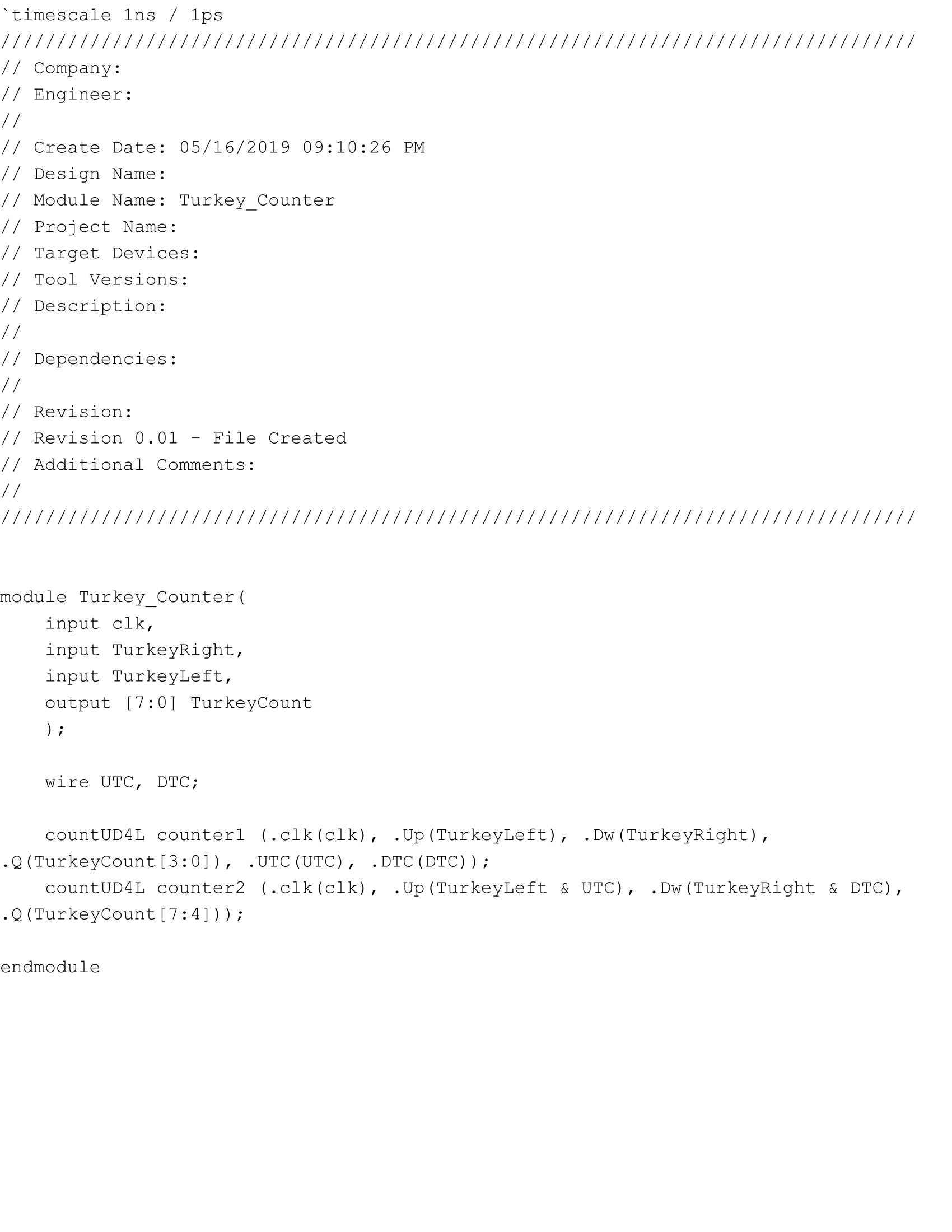
StateMachine\_Logic.v Schematic



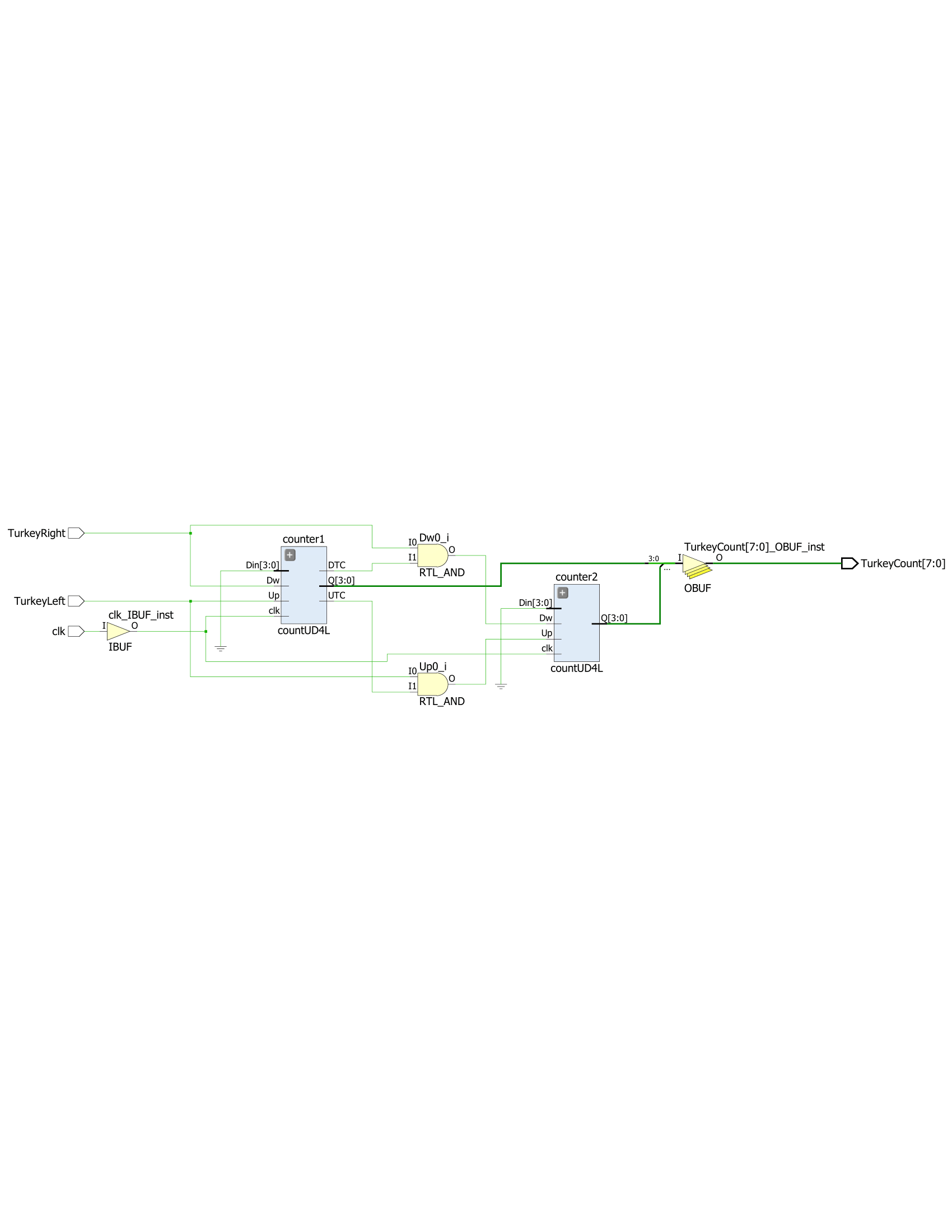
StateMachine.v



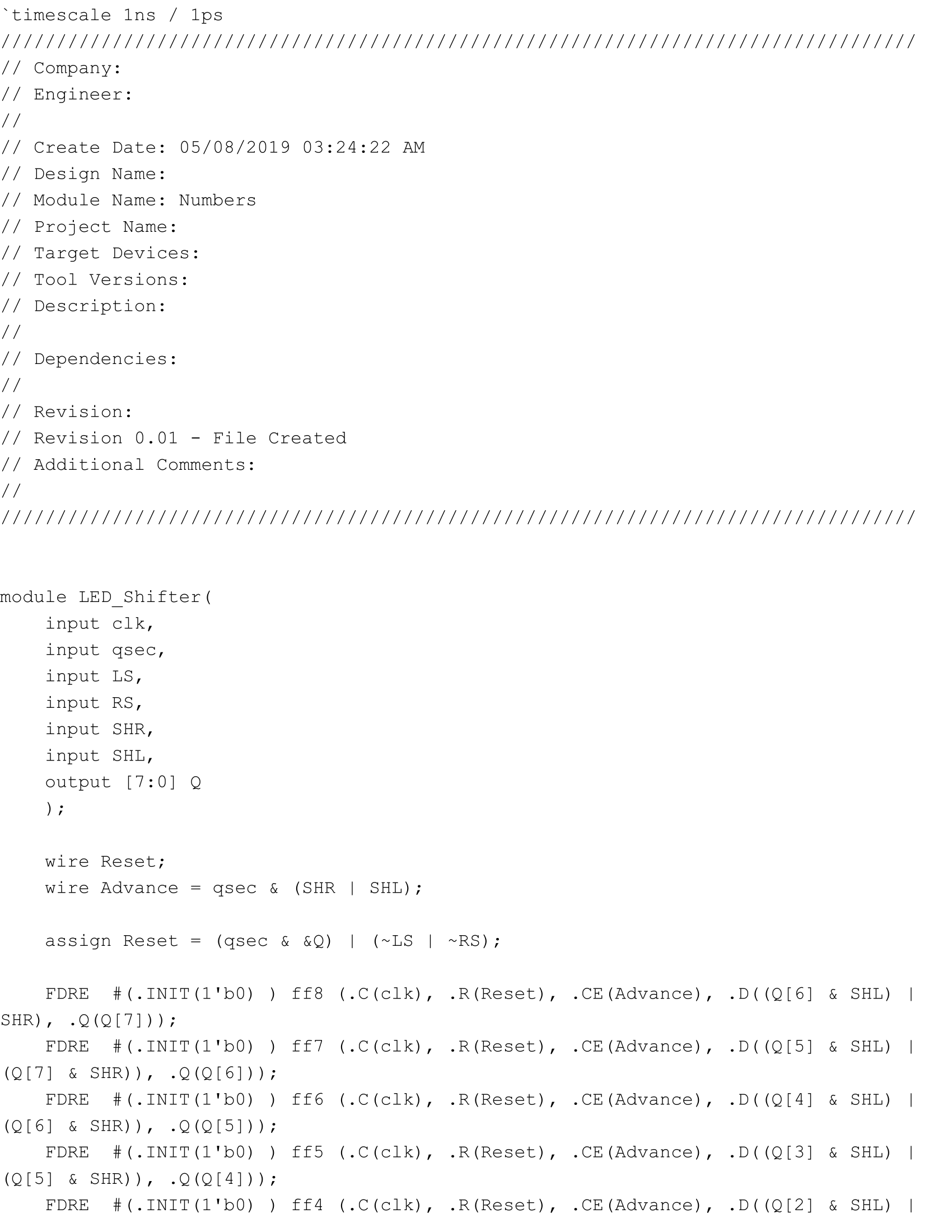
StateMachine.v Schematic



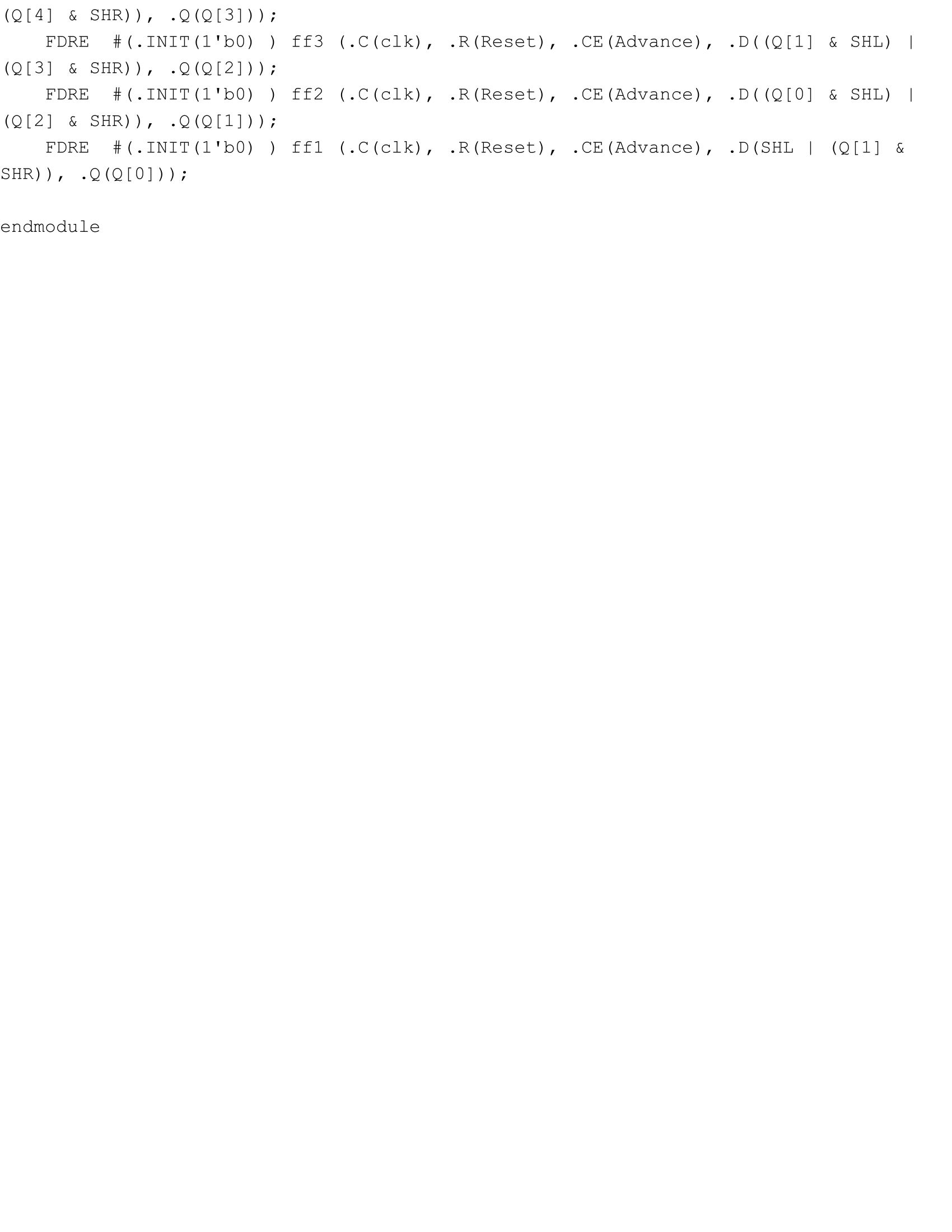
Turkey\_Counter.v



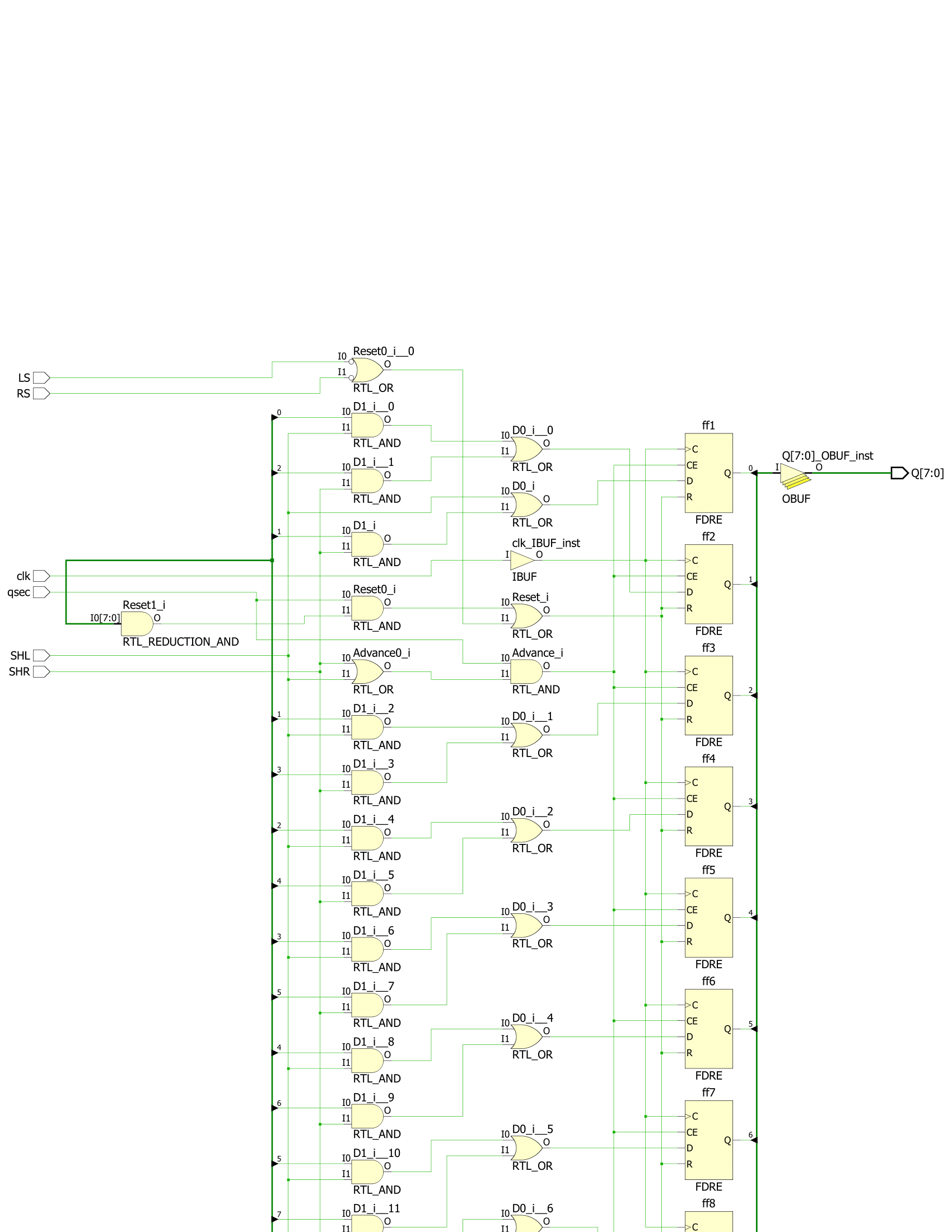
Turkey\_Counter.v Schematic



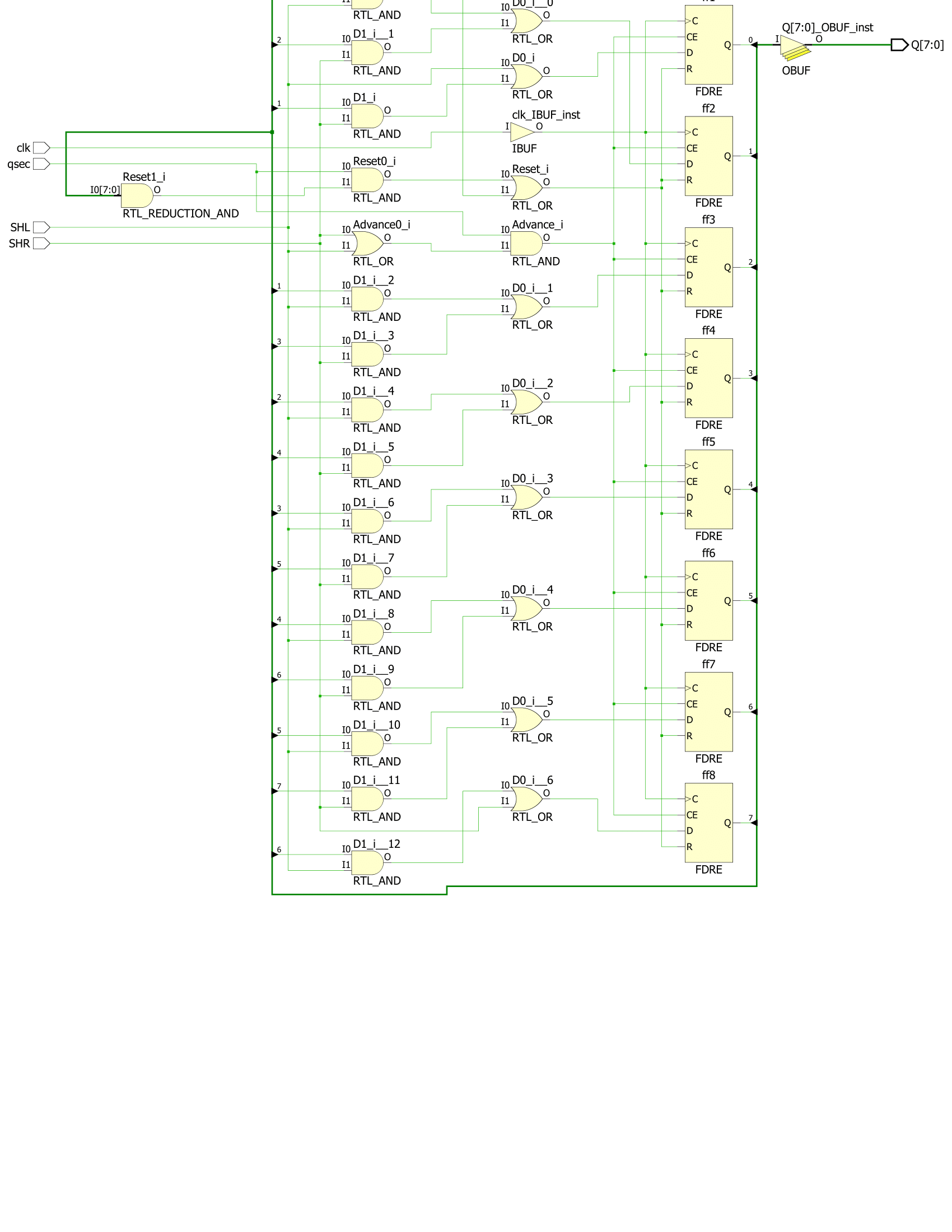
LED\_Shifter.v Part 1



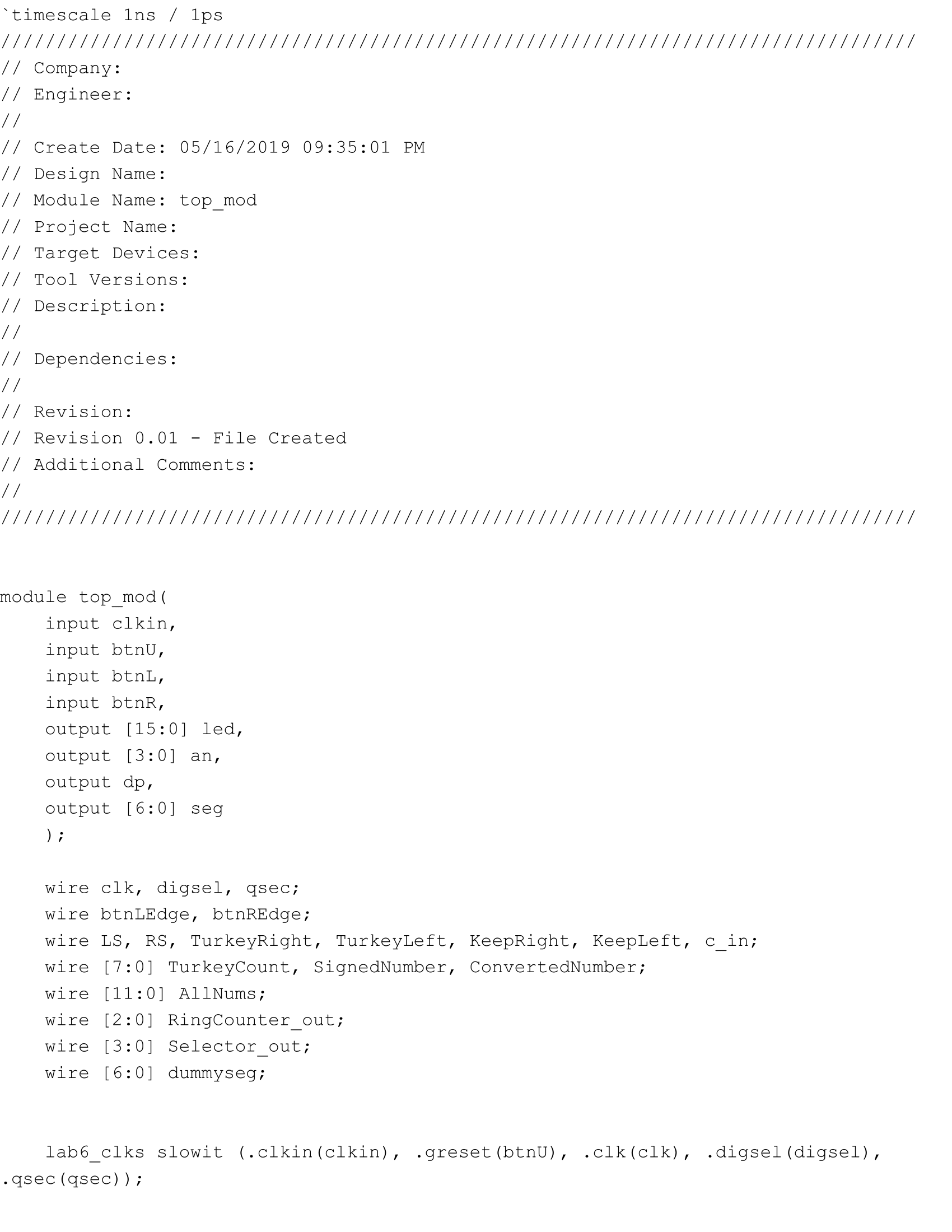
LED\_Shifter.v Part 2



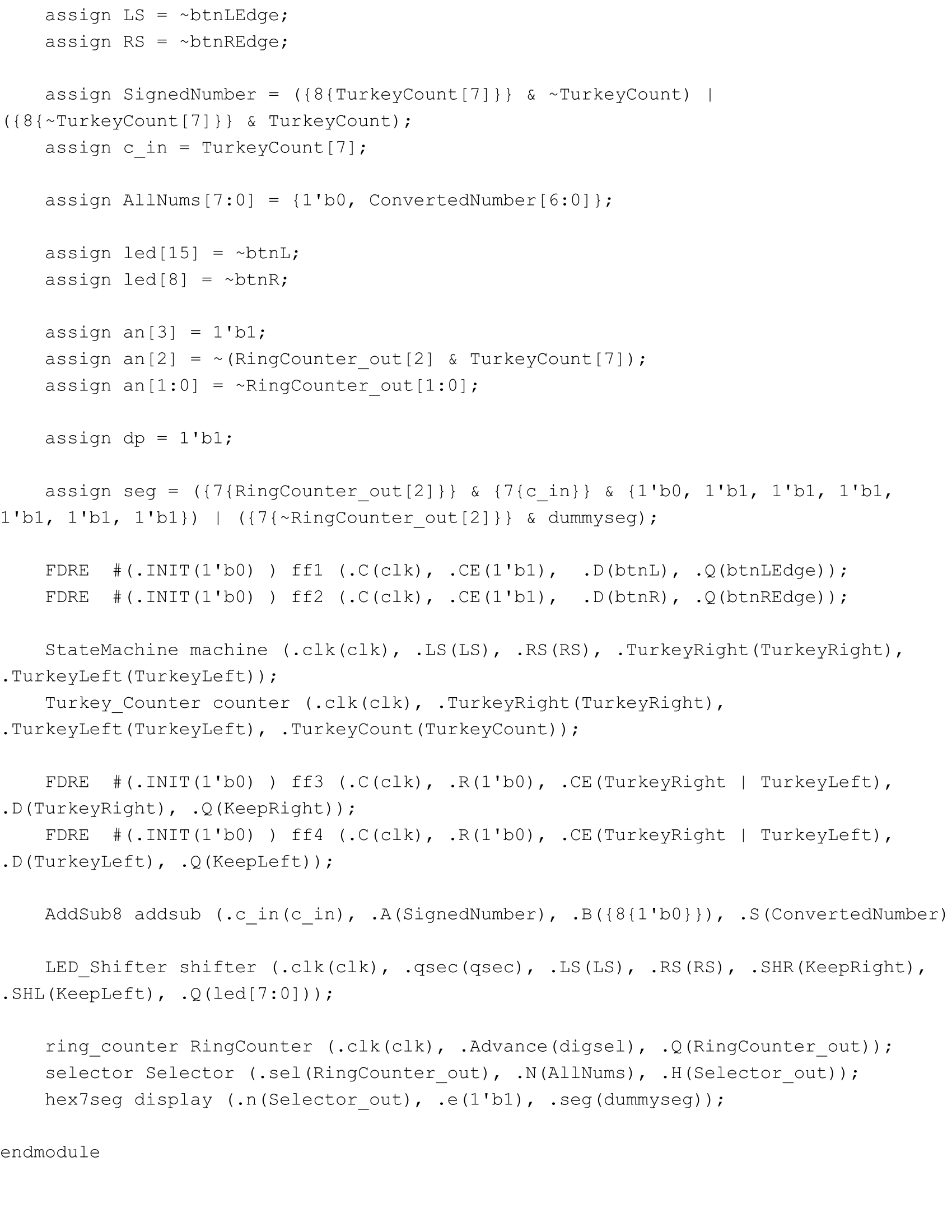
LED\_Shifter.v Schematic Part 1



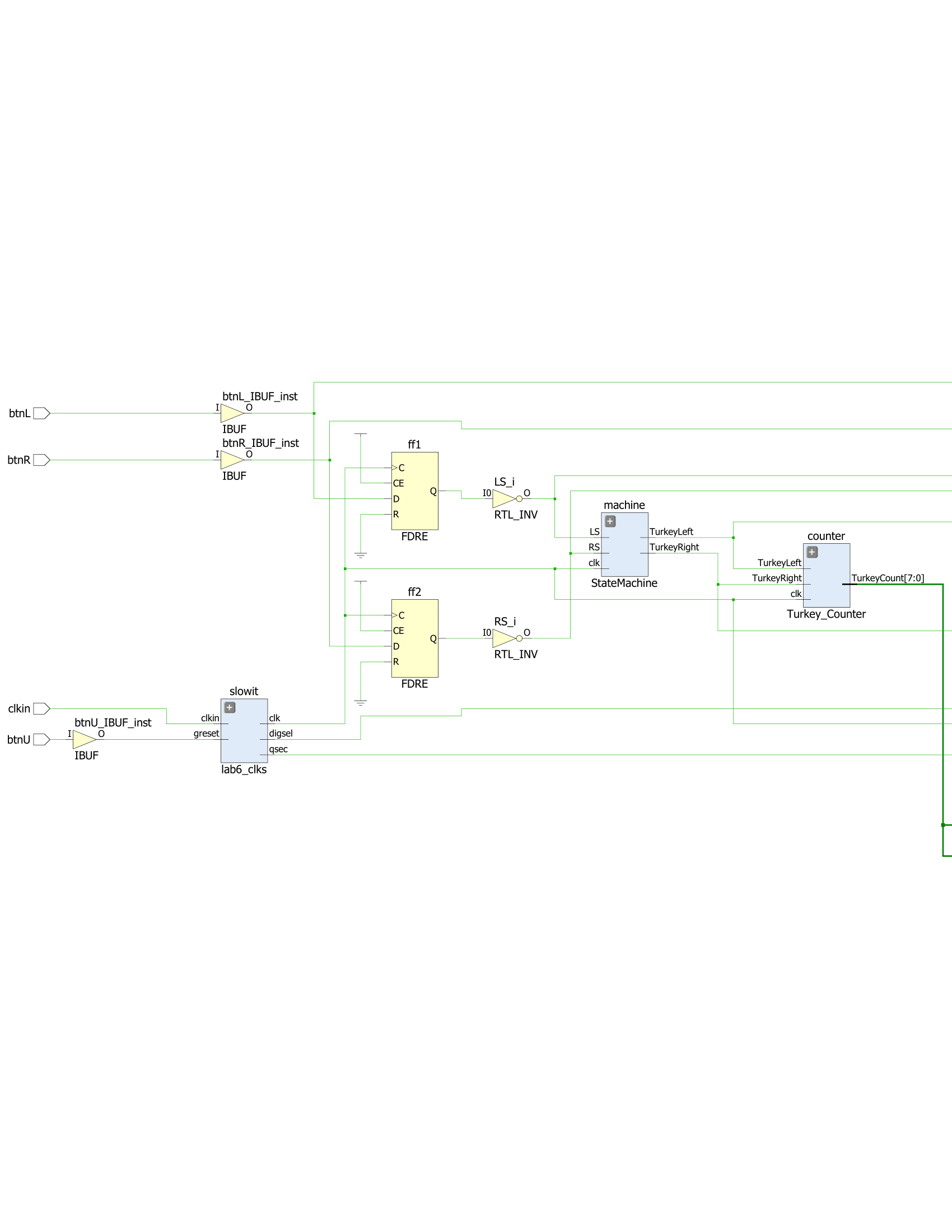
LED\_Shifter.v Schematic Part 2



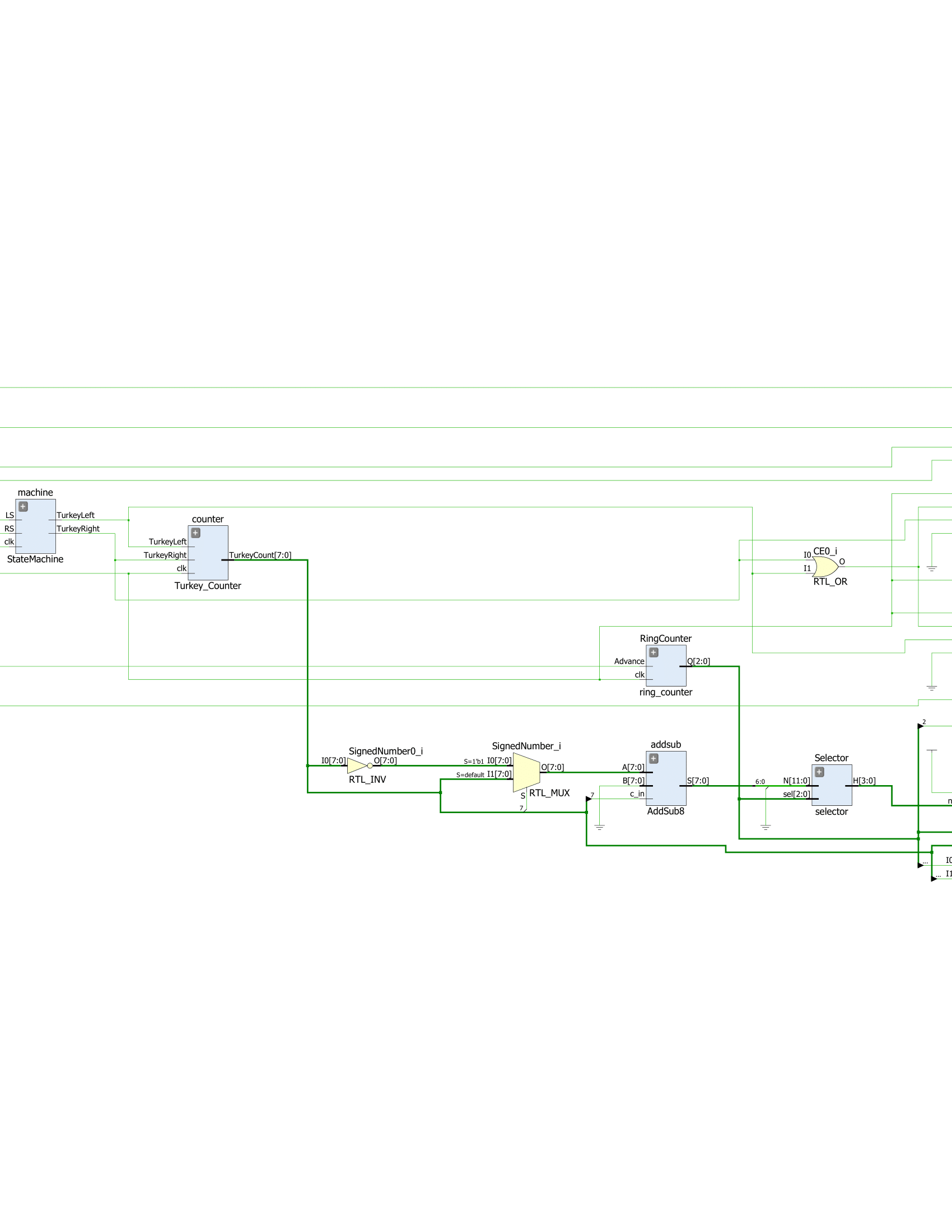
top\_mod.v Part 1



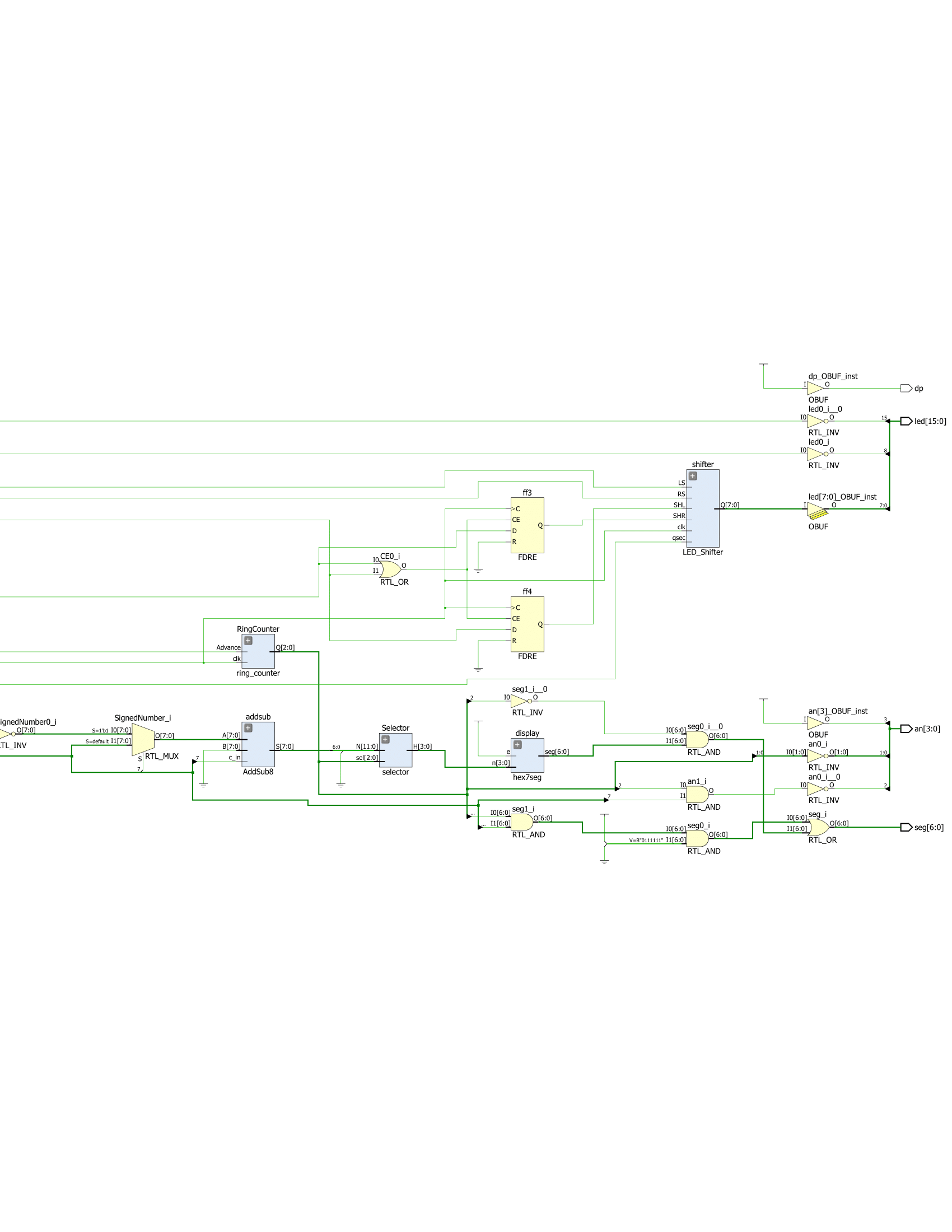
top\_mod.v Part 2



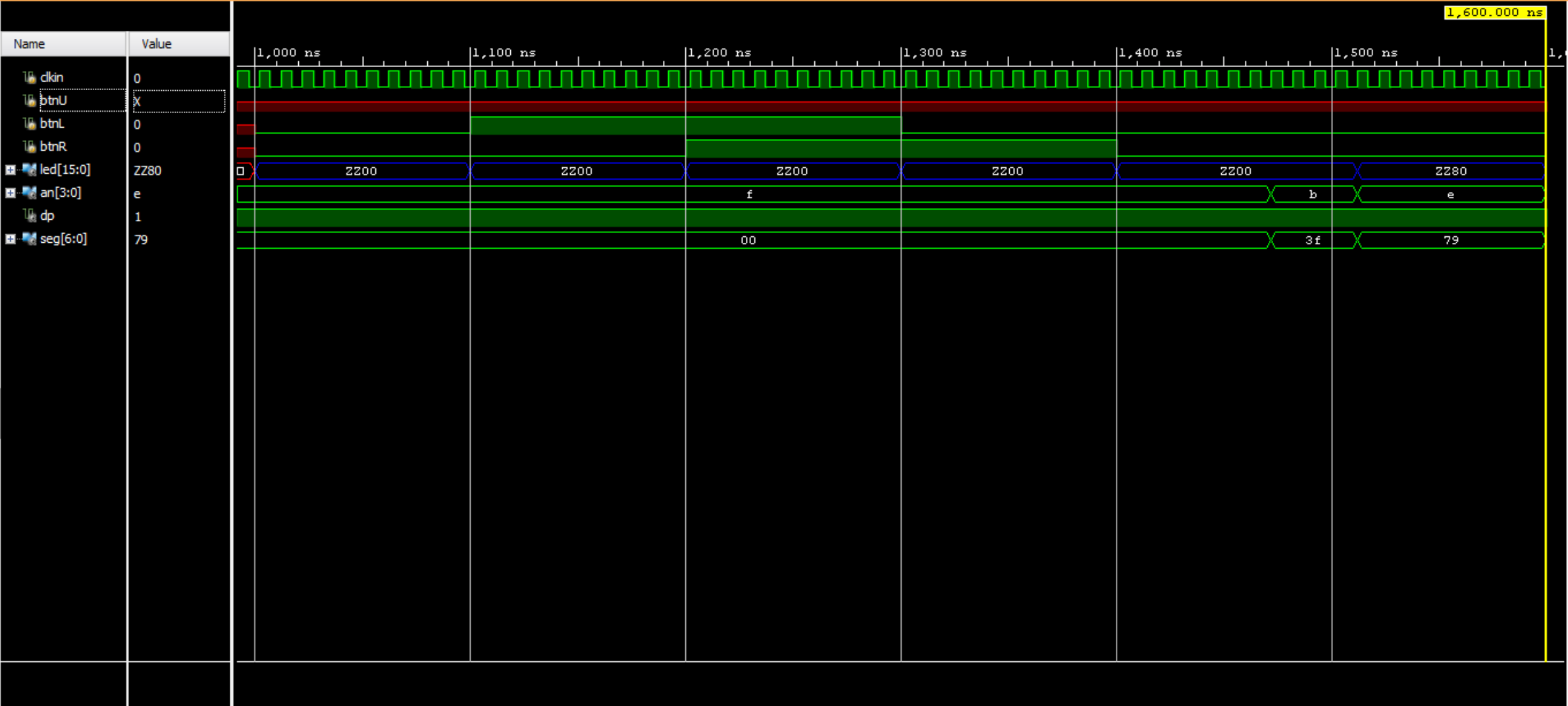
top\_mod.v Schematic Part 1



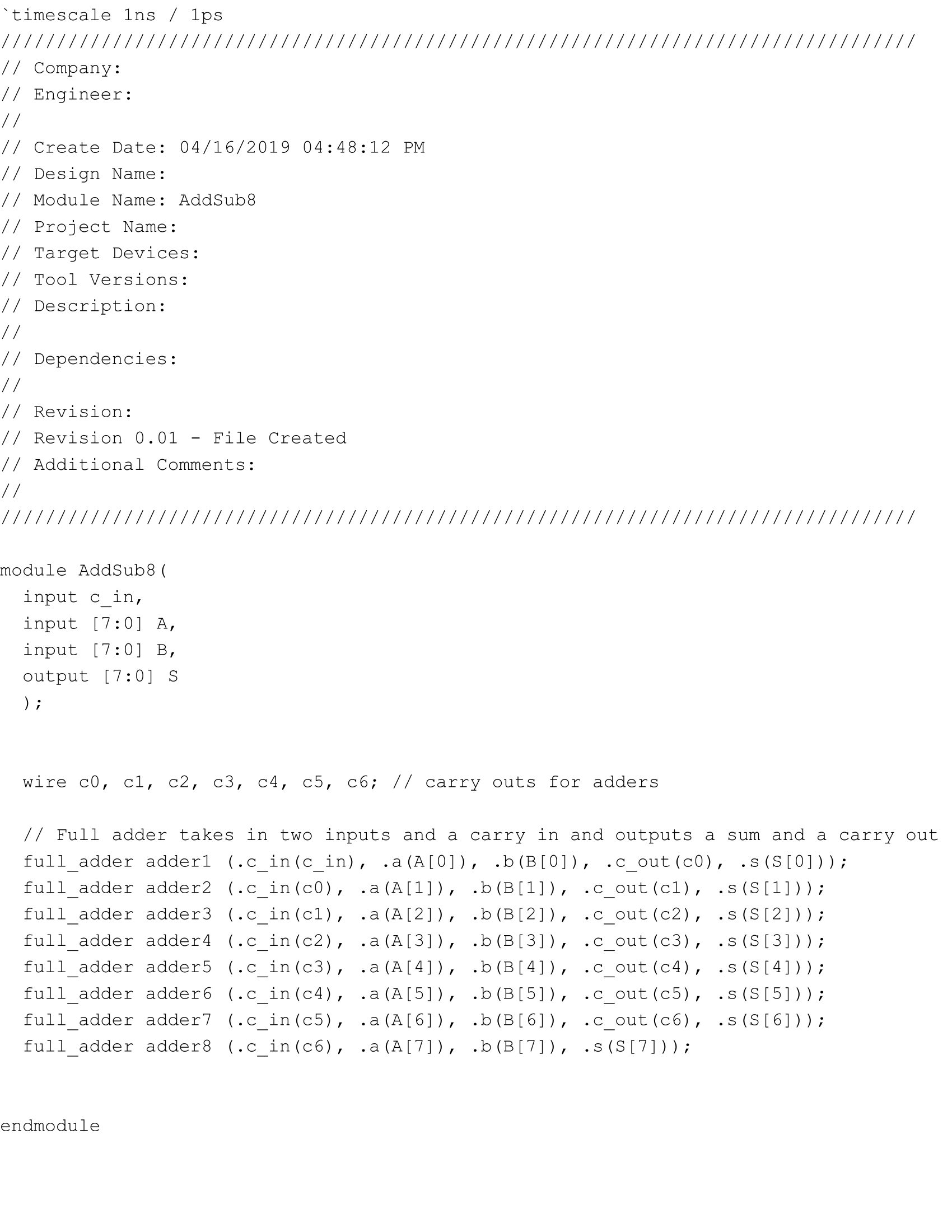
top\_mod.v Schematic Part 2



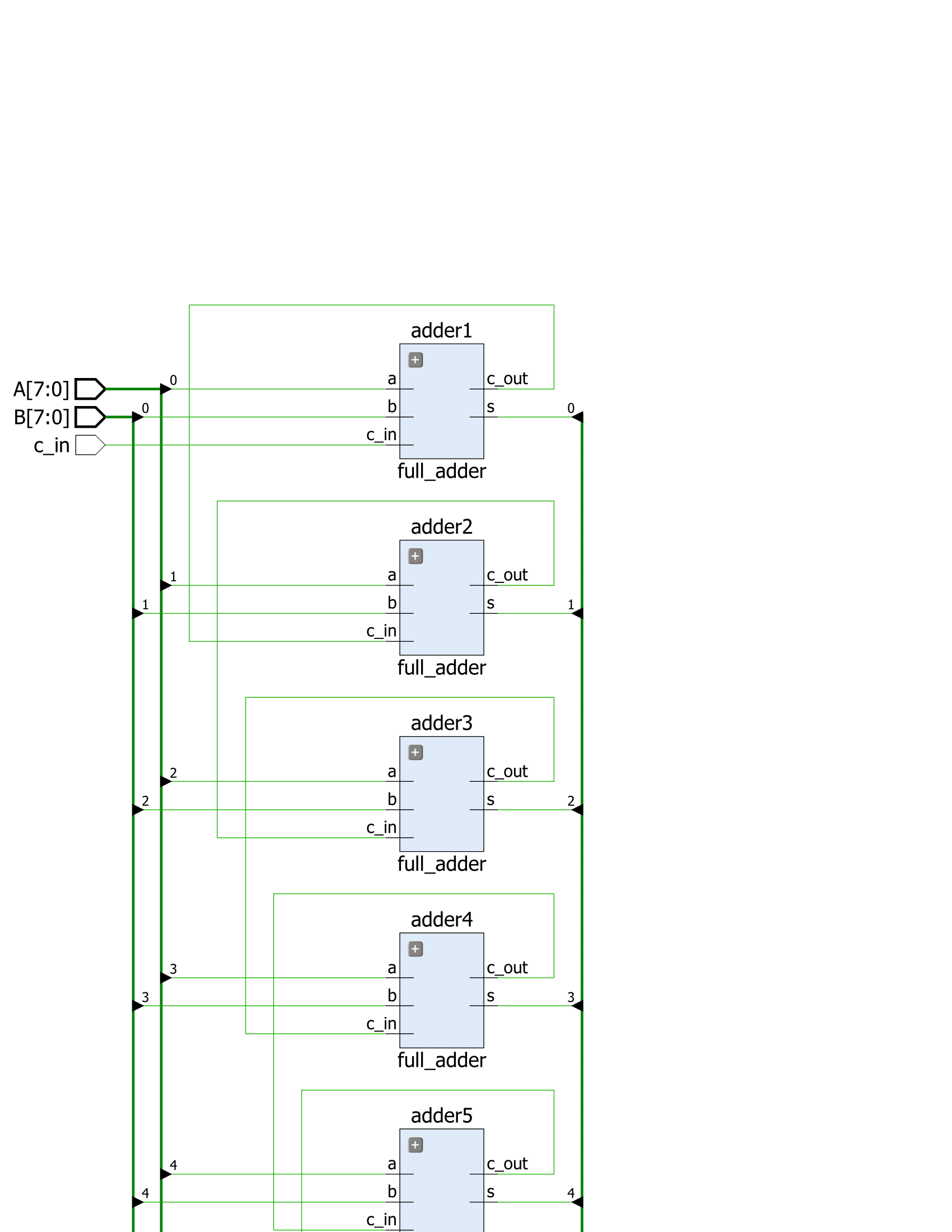
top\_mod.v Schematic Part 3



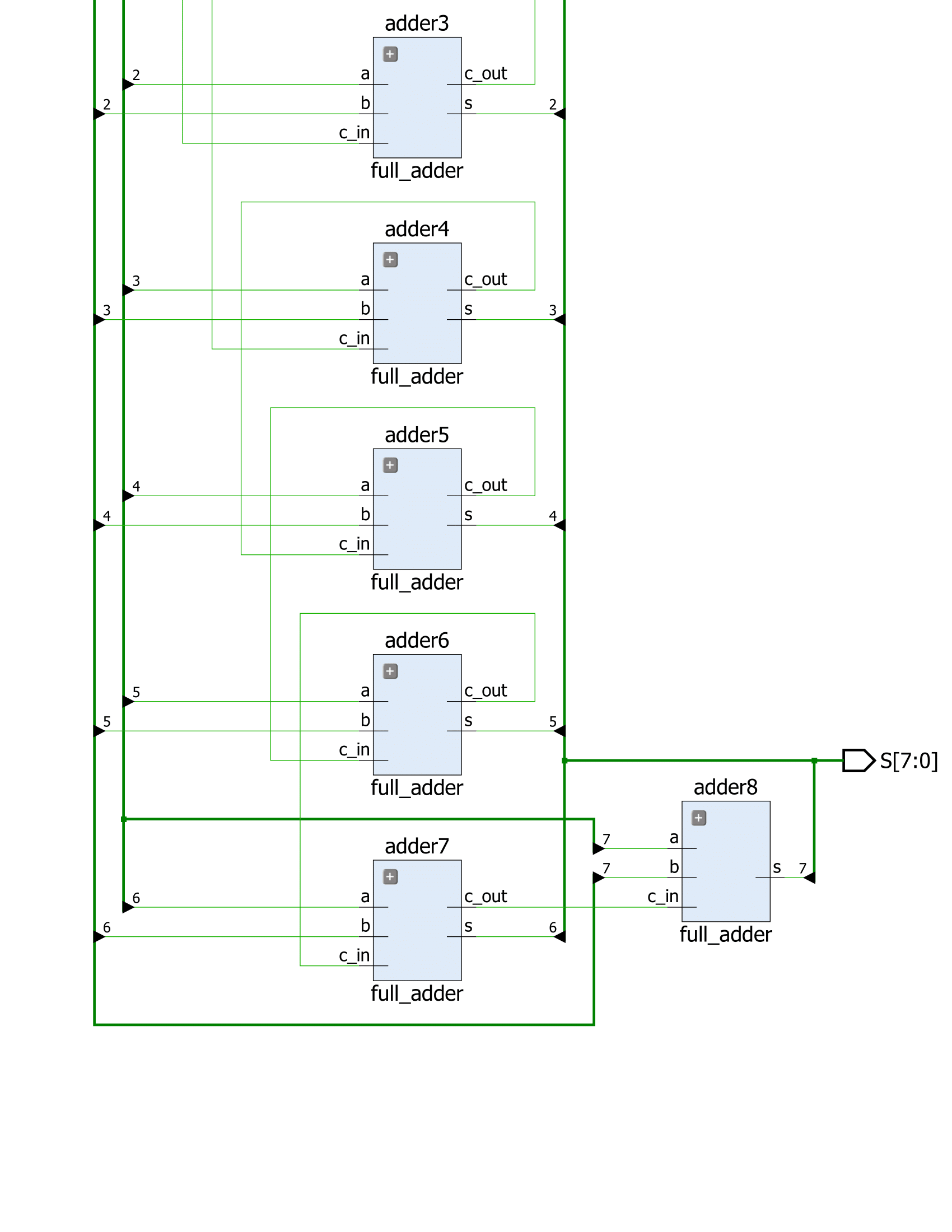
Waveform Viewer



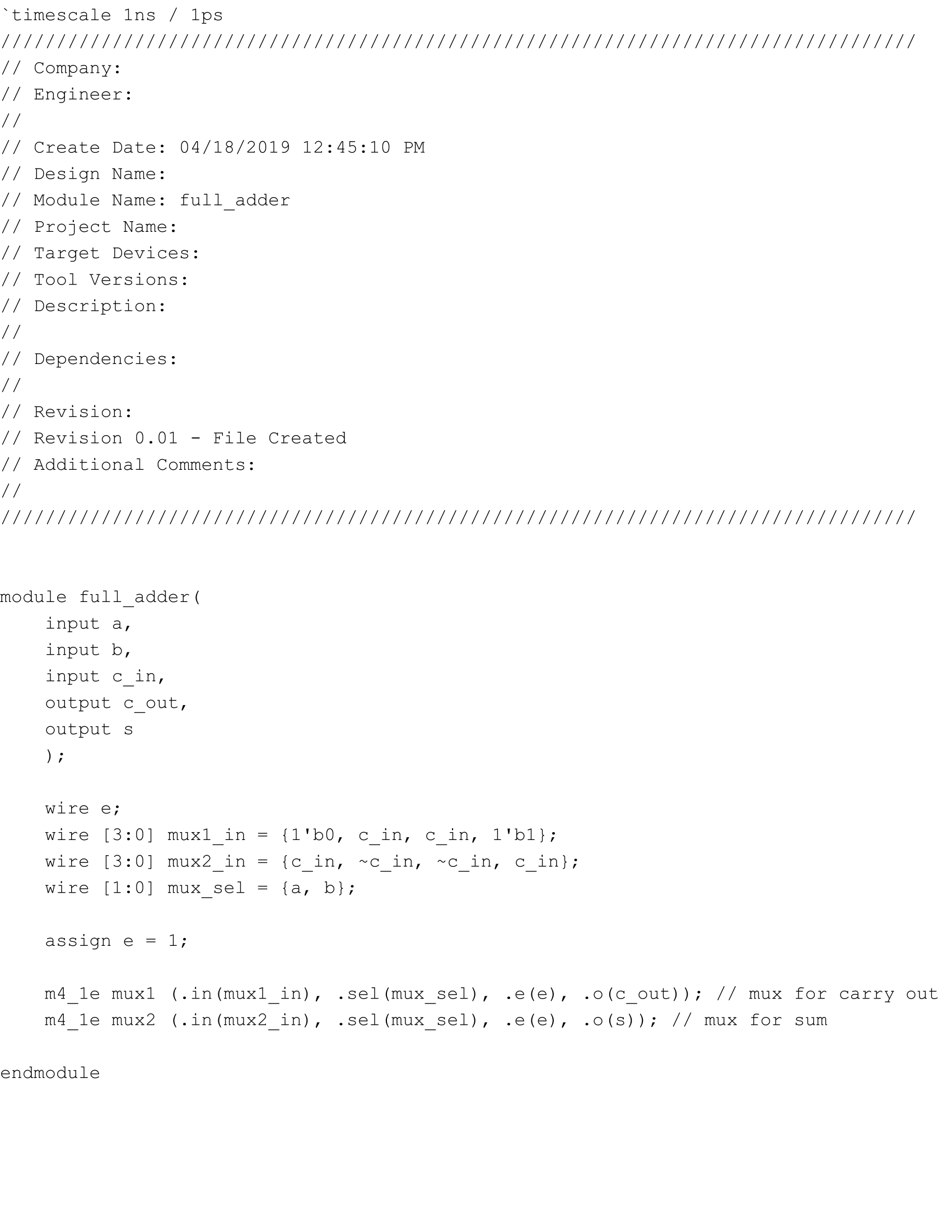
AddSub8.v



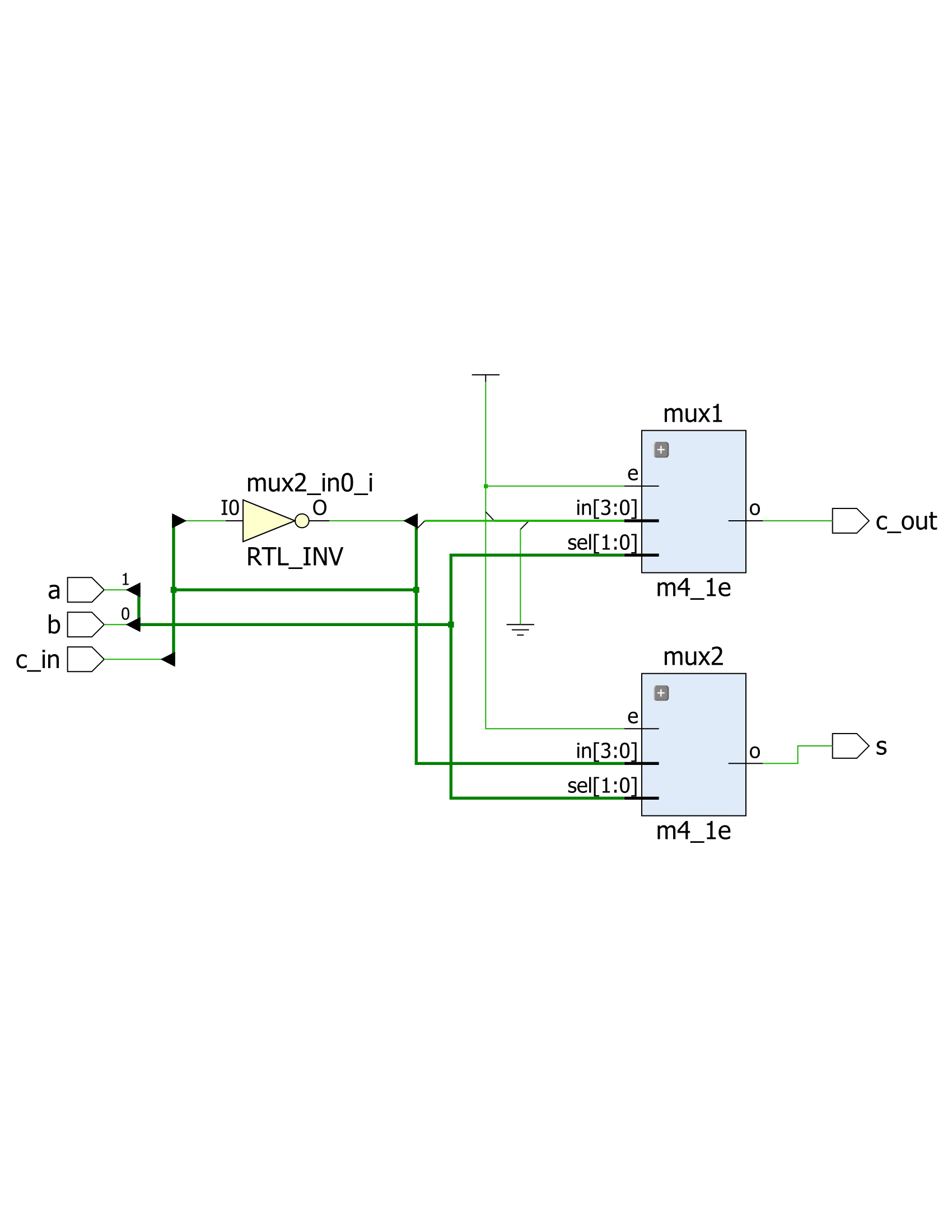
AddSub8.v Schematic Part 1



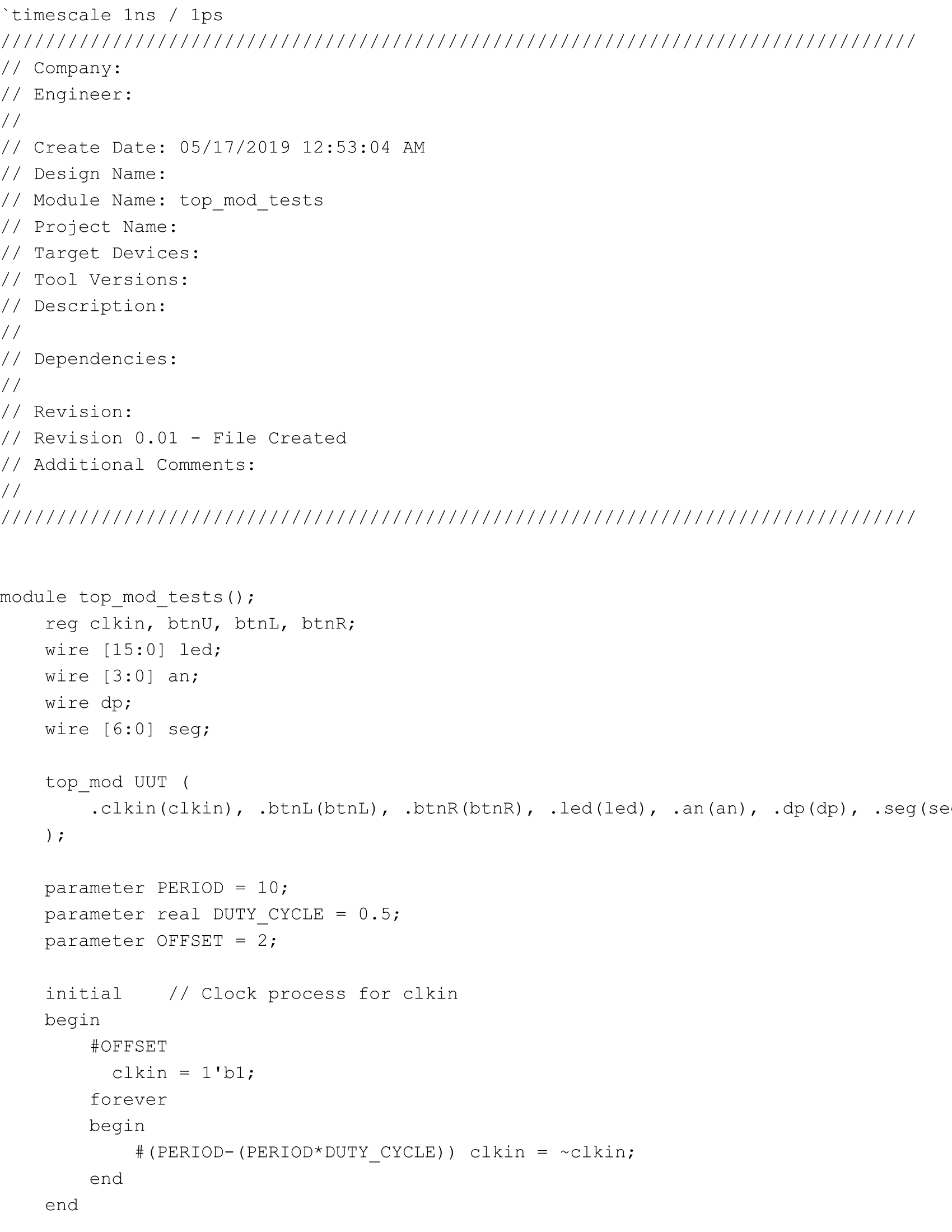
AddSub8.v Schematic Part 2



full\_adder.v



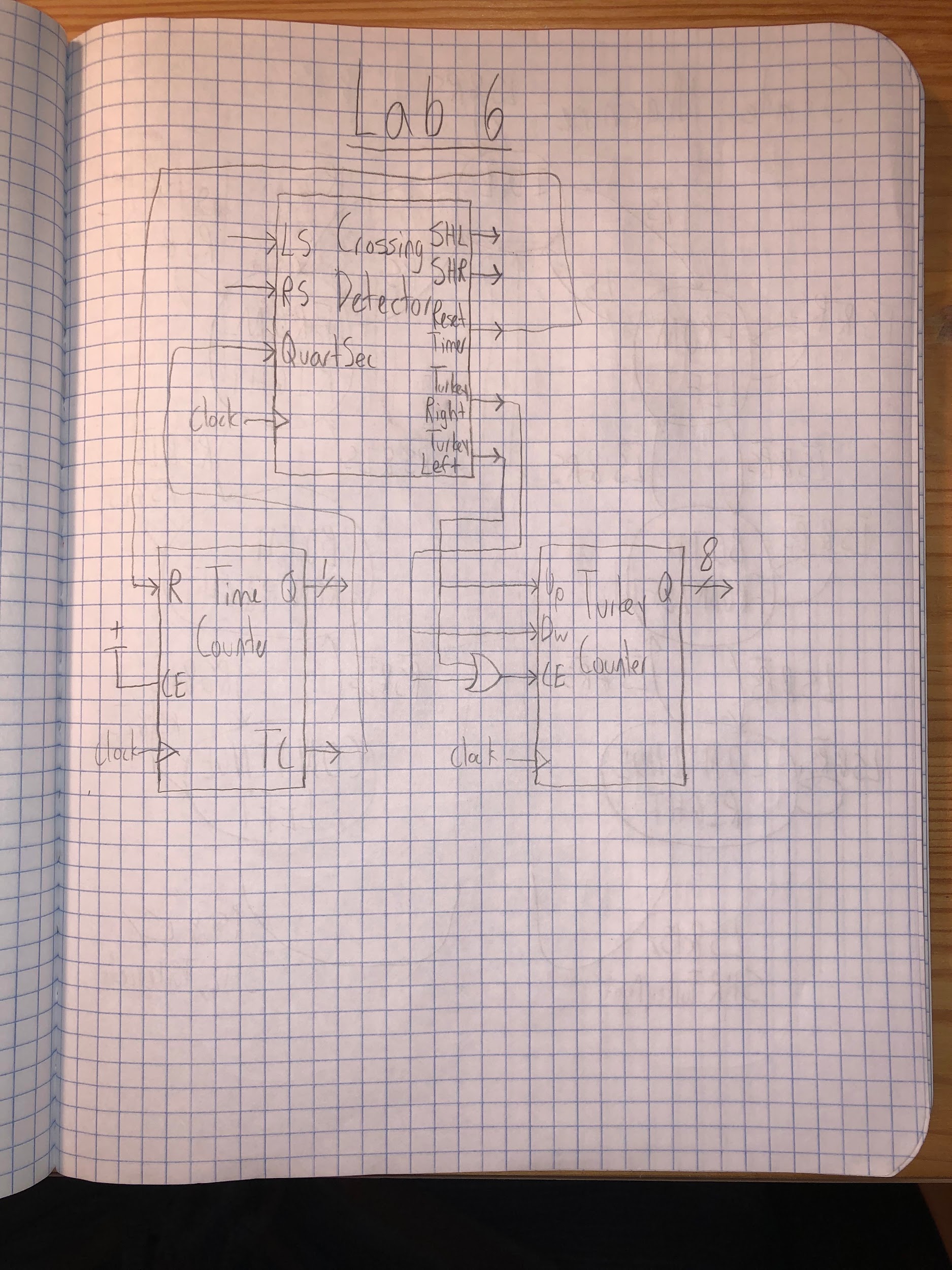
full\_adder.v Schematic



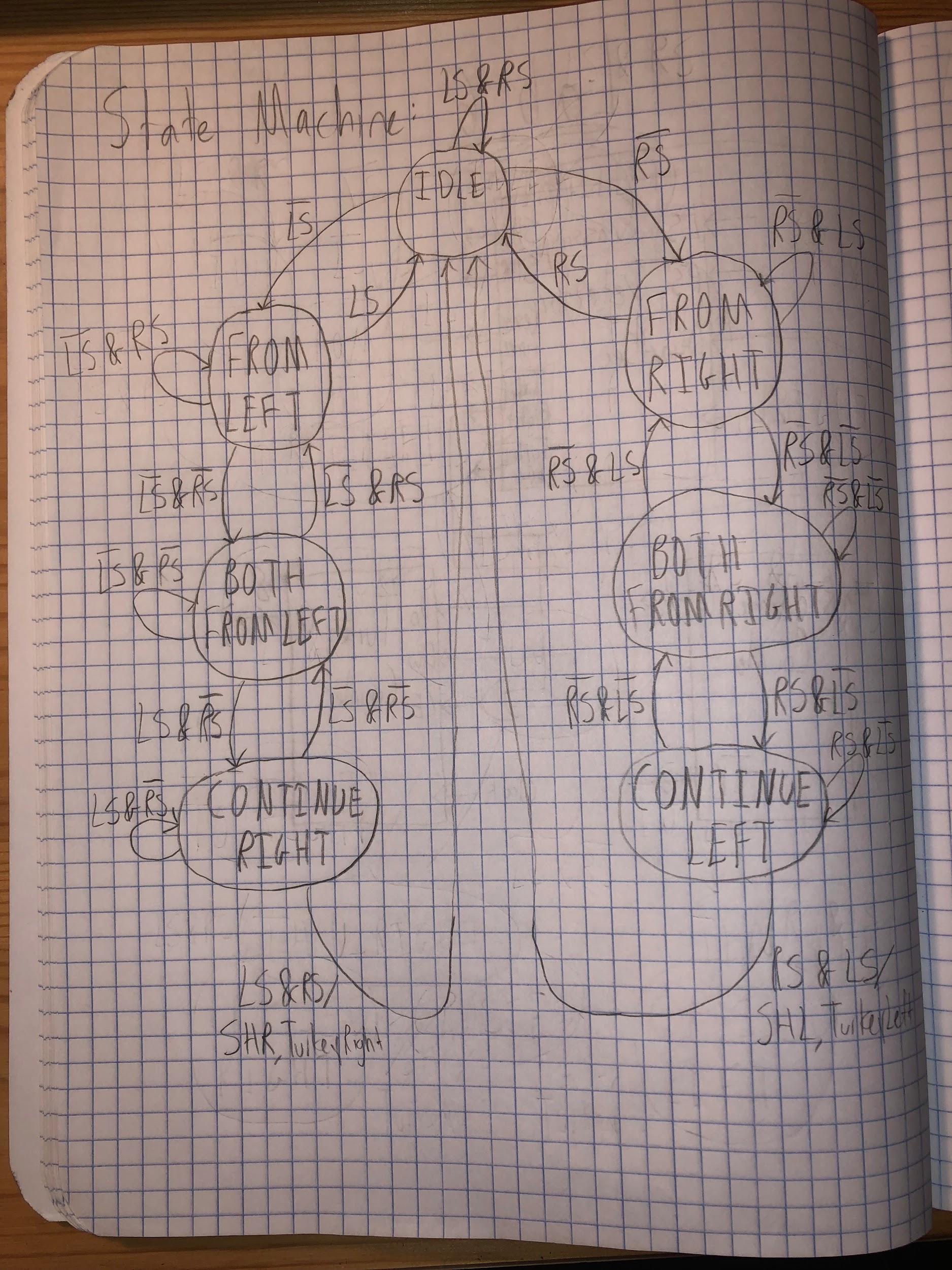
top\_mod\_tests.v Part 1



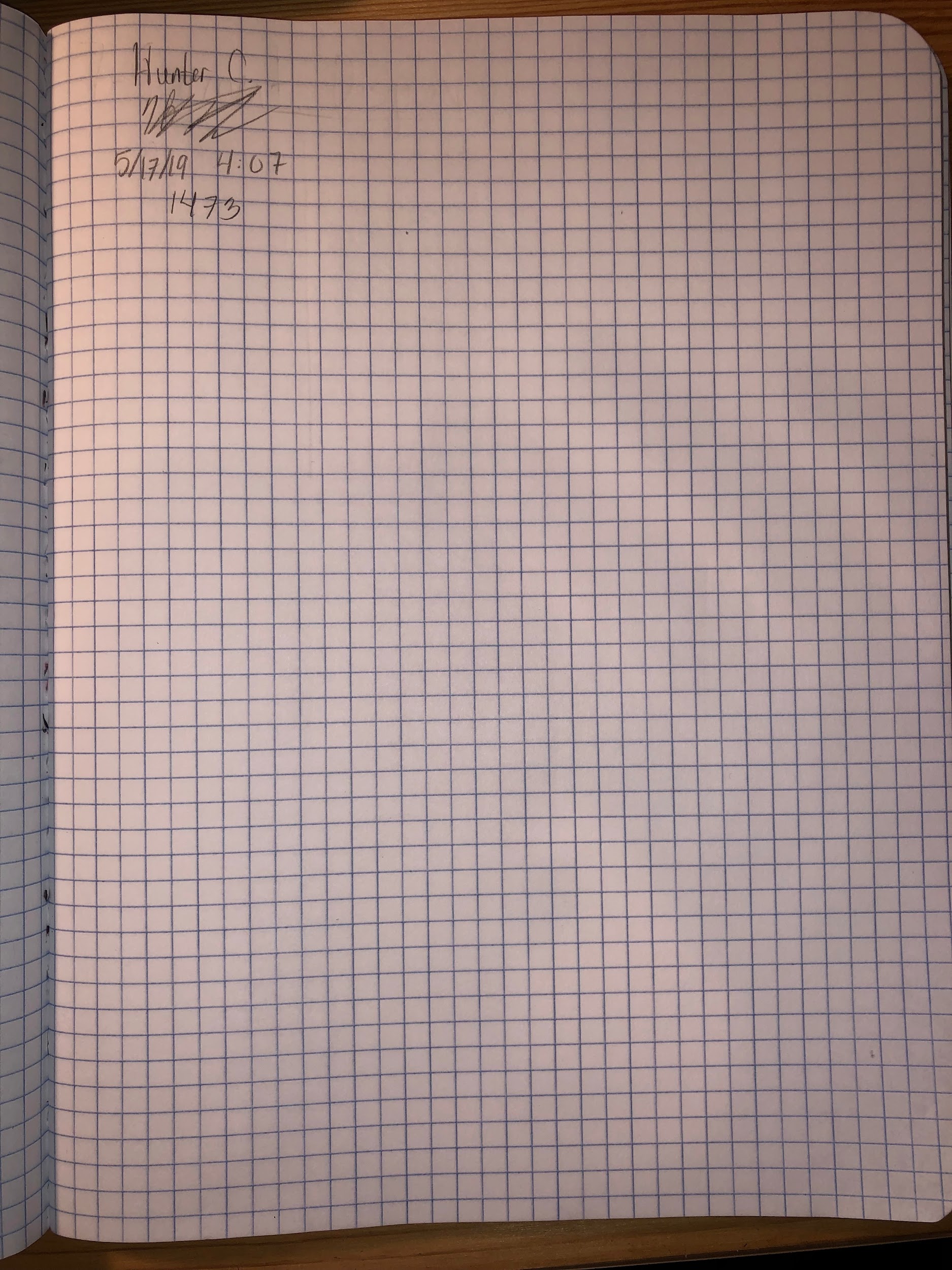
top\_mod\_tests.v Part 2



Lab Notebook Page 1



Lab Notebook Page 2



Lab Notebook Page 3