1. **The state diagram of your FSM.**

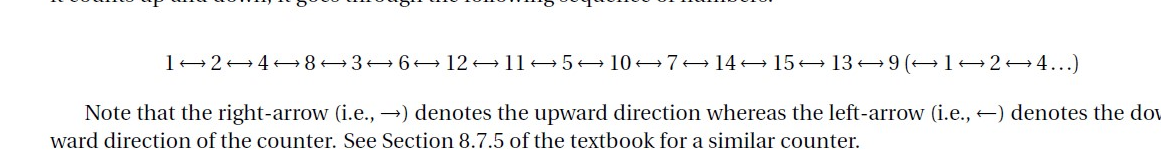


Figure 1: State Machine Possible States

The tests shown in the lab instructions are by figure 1. The each number can be represented by a letter which will be in alphabetical order, by convention. Thus, variable A represents 1, B represents 2 and so for.

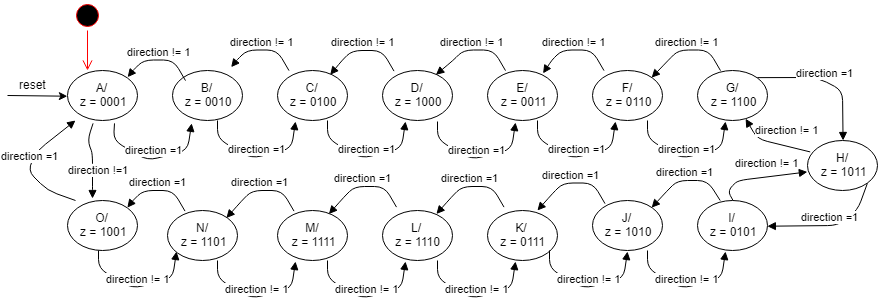


Figure 2: State Diagram

For our state diagram, A is the corresponding state from the instructions. Z is the binary representation of the value in that state. If the direction is 1, the state diagram will change to the next state. Consequently, if it is not 1, it will go to the previous state. For instance, if the current state is A, the next state is B if direction = 1 or it will be O if direction != 1. Also, the starting state (initiation) our state machine is at state A so as the reset.

1. **A description of the FSM circuit.**

The FSM circuit can be broken down into 3 parts.

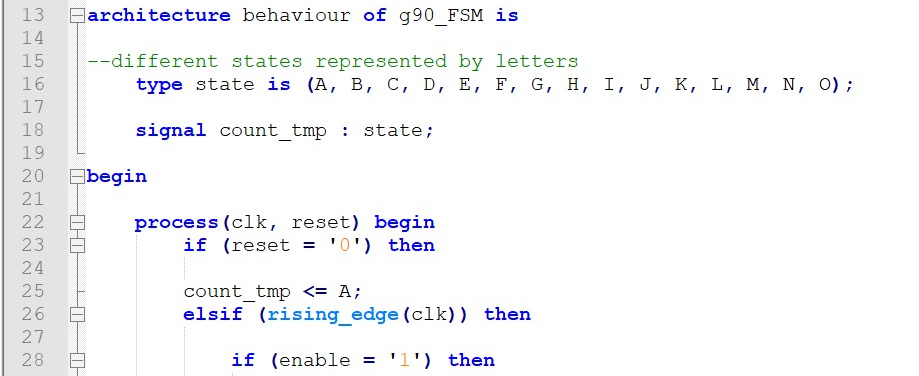


Figure 3: Part 1 of State Machine - Initialization of count and variables

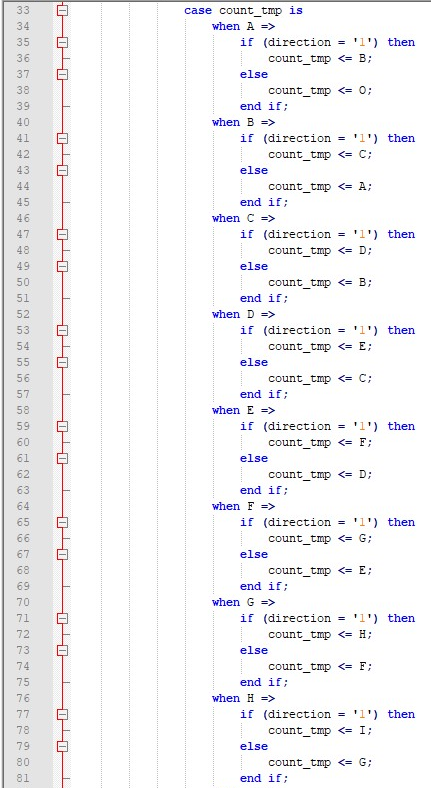
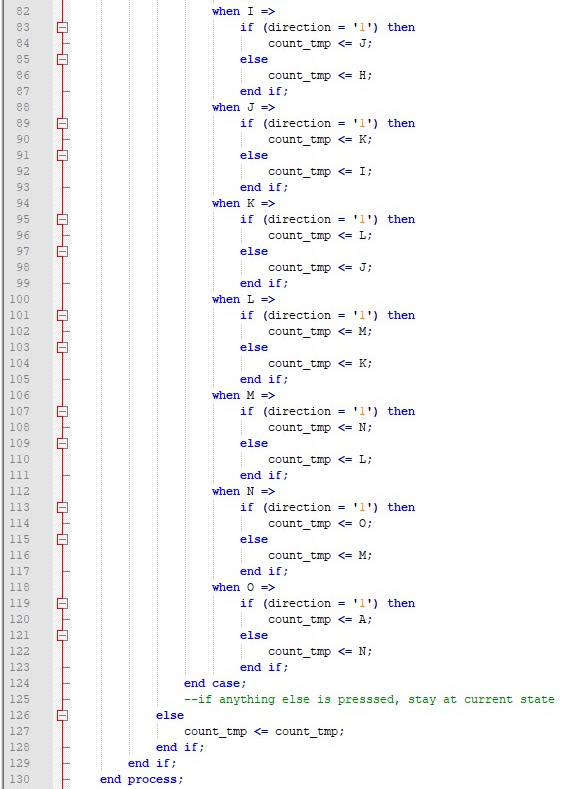


Figure 4.2: From State I to State O and other cases

Figure 4.1: From State A to State H

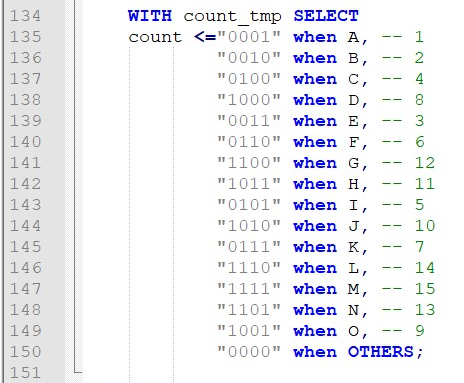


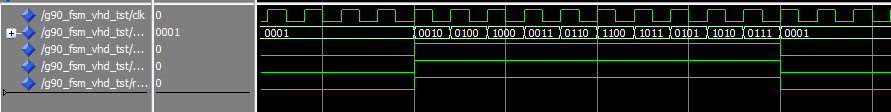
Figure 5: Conversion of States to Binary

1. **A discussion of how the FSM circuit was tested, showing representative simulation plots. How do you know that these circuits work correctly?**

The FSM circuit was tested by simulating its behavior. Observing both figures, we notice that our testbenches accurately track the clock. Given that this worked in past labs, this isn’t so much a point of interest. Instead, we want to observe the state change, that is to say, the second element, and to determine if it iterates through the states, whether or not it’s moving in the desired direction, and finally what conditions it appropriately functions under.

When observing both files, we realize that the defined state machine properly iterates through all desired states in the desired fashion. To elaborate, the simulation properly illustrates that the iteration only occurs when both enable and reset are active (i.e. rising edge). Given the logic of the written code, this demonstration describes the states only iterating, when the enable is “on”, and reset is “off”. In this regard, our simulation works correctly.

Furthermore, we can observe, that for both test files, our simulation does in fact iterate through its options in the desired fashion outlined in the lab. Depending on what the direction is set to (i.e. 0 or 1), the state machine will iterate either forward, or backward, respectively. Observe the figures below.

 Figure 6.1: FSM simulation with direction set to 0

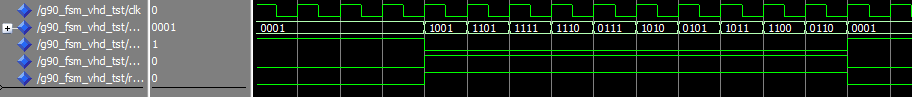


Figure 6.2: FSM simulation with direction set to 1

1. **A description of the multi-mode counter circuit.**
2. **A discussion of how the multi-mode counter circuit was tested on the FPGA board.**

The way that we tested the circuit is straight forward. Once the code is loaded onto the board, we pressed start while keeping the direction = 1. We followed what was displayed on the board with the figure 1 which was provided with the lab report. Then, we flipped the direction = 0; we looked if the FSM would go into the other direction. We had to make sure that when the state hits O it switches to A and same for the inverse. Also, we tested if the stop function to see if the board would remain at its current position. Finally, we made sure that the reset button will return the state machine into its starting state which is A (0001).

To further test our board, we randomly flipped the direction switch in order to test if the clock would go to the previous or the next state.

1. **A summary of the FPGA resource utilization (from the Compilation Report’s Flow Summary) and the RTL schematic diagram for the multi-mode counter circuit. Clearly specify which part of your code maps to which part of the schematic diagram**

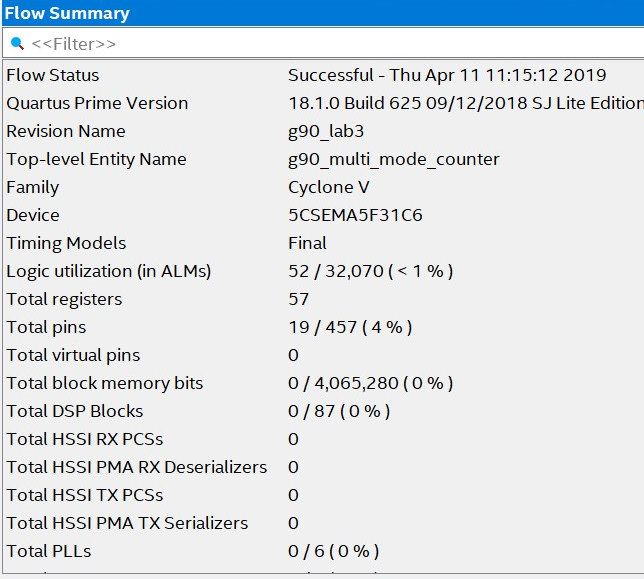


Figure 6: Flow Summary

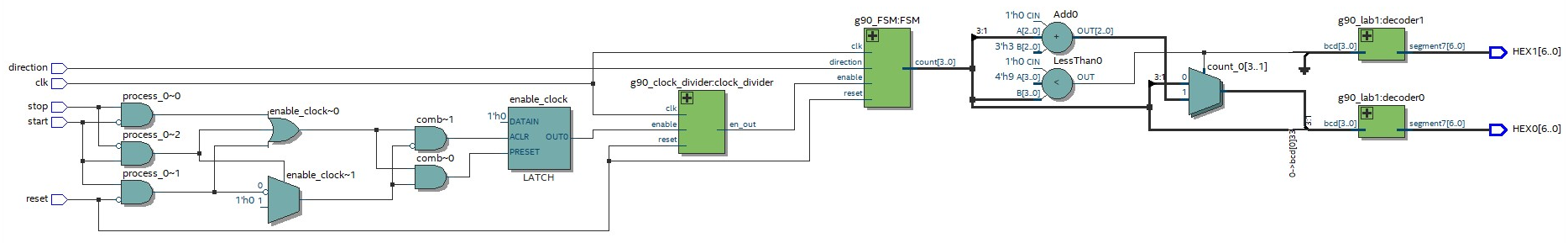


Figure 7: RTL Viewer