Lab 4: State Machine Behavioral Modelling in VHDL

Seth Kurtenbach, Dylan Samson, Angelique Taylor

March 4, 2014

Lab Objective

In this lab, we were given a state machine and directed to produce a single process behavioral model in VHDL using ModelSim. The state machine had three input signals, x, y, and a clock, and one output signal z. In this report, we describe how we achieved the objective, and the results of our lab.

Lab Work

1. Design

In order to model the state machine as a single process, we decomposed it into two units. The first unit modeled the state machine's transition from one state to another, based on its current state and the x and y input values. We used VHDL's built in case analysis structure for this purpose, with each current state marking a distinct case. Within each case, we used if...then...elsif... statements to respond to each of the four possible xy combinations. We nested this case analysis structure inside a process sensitive to the clock. To model the rising edge activation, we conditionalized the execution of the case analysis on the following if statement:

if CLK'event and CLK = '1' then ...

Figure 1 shows a relevant portion of our code.

```
architecture stateProcess of StateMachine is
signal State: integer range 0 to 3 := 0;
begin
  process(CLK)
  begin
  if CLK'event and CLK = '1' then    -- rising clock edge trigger
    case State is
    when 0 =>                        -- next state for current state = 0
      if X&Y = "00" then State <= 0;
      elsif X&Y = "01" then State <= 1;
      elsif X&Y = "10" then State <= 2;
      else State <= 1;
      end if;

    when 1 =>                        -- next state for current state = 1
      if X&Y = "00" then State <= 1;
      elsif X&Y = "01" then State <= 0;
      elsif X&Y = "10" then State <= 2;
      else State <= 3;
      end if;

    when 2 =>                        -- next state for current state = 2
      if X&Y = "00" then State <= 2;
      elsif X&Y = "01" then State <= 3;
```

Figure 1: Beginning of single process, with case analysis portion.

The second unit modeled the state machine's output, which is a function of its current state. In order to limit our model to a single process, we included this unit outside of the above process, running concurrently with it. It is also conditional on the rising edge clock tick. Figure 2 shows the relevant portion of the code.

```
    when 2 =>                          -- next state for current state = 2
      if X&Y = "00" then State <= 2;
      elsif X&Y = "01" then State <= 3;
      elsif X&Y = "10" then State <= 3;
      else State <= 1;
      end if;

    when 3 =>                          -- next state for current state = 3
      if X&Y = "00" then State <= 3;
      elsif X&Y = "01" then State <= 0;
      elsif X&Y = "10" then State <= 1;
      else State <= 0;
      end if;

  end case;
end if;
end process;

-- Z output values, depending only on state
Z <= '1' when (State = 2 or State = 3) else '0';

end stateProcess;
```

Figure 2: End of single process, separate unit modelling output Z.

This concludes our discussion of the model design. We now turn to our results.

2. Results

For this lab, we were given a sequence of x, y, and CLK inputs for testing purposes. Figure 3 shows the results of our model simulation given the inputs.

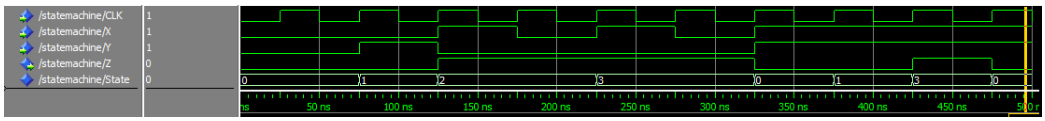

Figure 3: Results of the given input sequence.

We determined that this is the correct behavior for a single process model of the state machine. In the assignment, there are slightly different results shown, but that is because the lab assignment shows the results of a two process model, which results in an additional delay between the state transition and outputs.

In order to further test our model, we altered the given input sequence in the following way. We left all of the timing instructions unchanged, but reversed the forced value assignments. Initially, we forgot to include the *CLK* assignment sequence, and so we observed no changes in the simulation. However, once we corrected for this error, we generated the following results, shown in Figure 4.

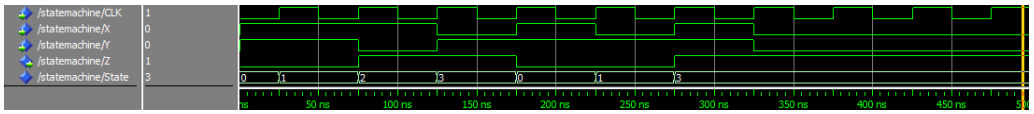

Figure 4: Results of the altered input sequence.

Conclusion

To conclude, we have shown that we achieved the lab objective, which was to construct a single process behavioral model for the given state machine. Our construction relied on a single process sensitive to the *CLK* signal, with an if statement representing rising edge triggered action. Within the if statement, we modelled the state machine's state transitions with case statements comprised of if...then..elsif statements for each possible xy combination. We modelled the state machine's ouput signal, z, with a concurrently executed statement. Other than forgetting to include the *CLK* sequence in our altered input sequence, we experienced no problems with this lab.