Laboratory 4, Total Points: 30

Due Date: March 05, 2019 (Tuesday), 11:59 PM CST

Textbook Sections: Digital Design and Computer Architecture: 3.4, 4.6

In this lab, students will implement finite state machines (FSM) in VHDL. Students may refer the above-mentioned textbook sections and Quartus tutorial to complete the lab exercises.

Acknowledgement: The lab assignment is adopted from authors' website.

Exercises

1. [15 pts] In this exercise, you will design a **Finite State Machine** (FSM) in VHDL that implements an adventure game. The game has seven rooms and one object (a sword). The game begins in the Cave of Cacophony (**CC**). To win the game, you must first proceed through the Twisty Tunnel (**TT**) and the Rapid River (**RR**). From there, you will need to find a **Vorpal Sword** in the Secret Sword Stash (**SS**). The sword will allow you to pass through the Dragon Den (**DD**) safely into Victory Vault (**VV**) (at which point you have won the game). If you enter the DD without the Vorpal Sword, you will be devoured by a dangerous dragon and pass into the Grievous Graveyard (**GG**) (where the game ends with you dead).

This game can be factored into two communicating state machines. One state machine keeps track of which room you are in, while the other keeps track of whether you currently have the sword. The Room FSM is shown in Figure 1. In this FSM, each state corresponds to a different room. Upon `reset`, the machine's state goes to the CC. The player can move among the different rooms using the inputs **`n`**, **`s`**, **`e`**, or **`w`**. When in the SS, the **`sw`** output from the Room FSM indicates to the Sword FSM that the player is finding the sword. When in the DD, signal **`v`**, asserted by the Sword FSM when the player has the Vorpal Sword, determines whether the next state will be VV or GG; the player must not provide any directional inputs. When in GG, the machine generates the **`d`** (dead) output, and on VV, the machine asserts the **`win`** output.

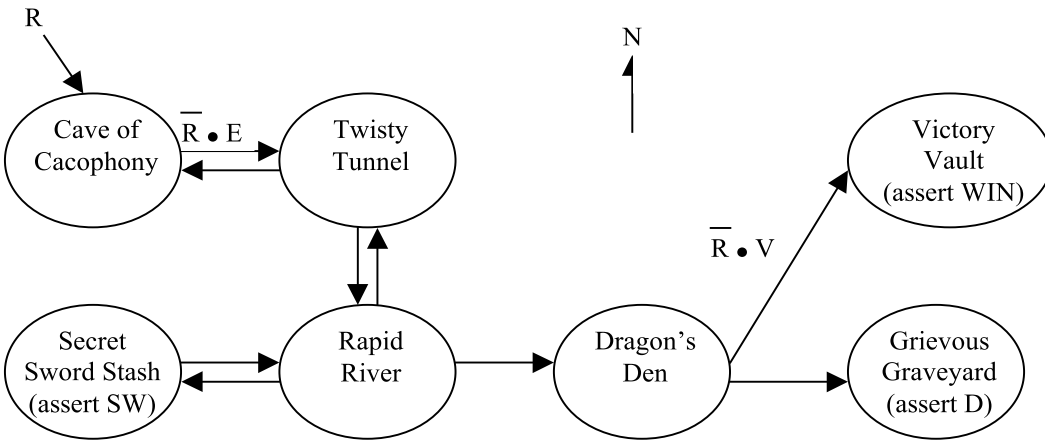

Figure 1: Partially Completed State Transition Diagram for Room FSM

In the Sword FSM (Figure 2), the states are "No Sword" and "Has Sword." Upon **`reset`**, the machine enters the "No Sword" state. Entering the Secret Sword Room causes the player to pick up a sword, so the transition to the "Has Sword" state is made when the **`sw`** input (an output of the Room FSM that indicates the player is in the Secret Sword Stash) is asserted. Once the "Has Sword" state is reached, the **`v`** (Vorpal sword) output is asserted and the machine stays in that state until **`reset`**.

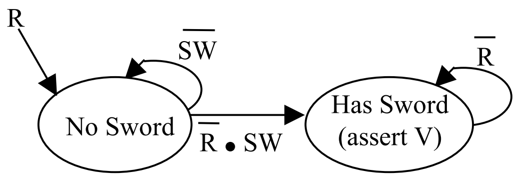

Figure 2: State Transition Diagram for Sword FSM

So far, we have given a description and a state transition diagram for each of the two FSM's. You may have noticed, however, that the diagram in Figure 1 is incomplete. Some of the transition arcs are labeled, while others are left blank. Complete the State Transition Diagram for the Room FSM now by labeling all arcs so that the FSM operates as described.

The next step in the design process is to enumerate the inputs and outputs for each FSM that you will use for entity design. Figure 3 shows the inputs (on the left) and outputs (on the right) of the Room FSM and Figure 4 does this for the Sword FSM. Note that for navigational purposes the Room FSM should output **`s0-s6`**, indicating which of the seven rooms our hero is in.

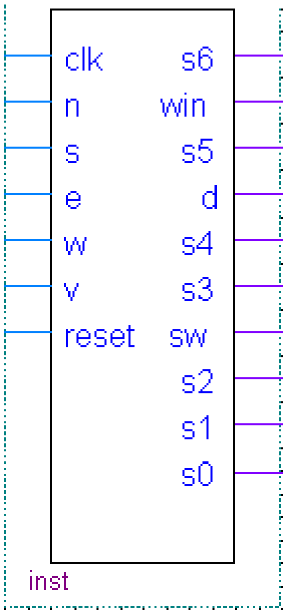

Figure 3: Symbol for Room FSM, showing its inputs and outputs

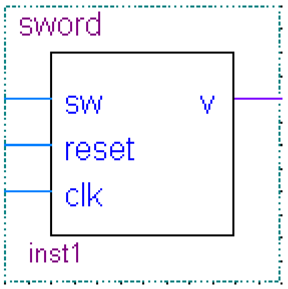

Figure 4: Symbol for Sword FSM, showing its inputs and outputs

With this information, now you can write VHDL code for your FSM design and simulate it. To test your design, write a testbench similar to the one uploaded with this assignment. Submit state transition diagrams, VHDL files for modules and testbench, and simulation results for the FSMs.

2. [15 pts] In this exercise, you will design a finite state machine (FSM) in VHDL to control the taillights of a 1965 Ford Thunderbird. Three lights on each side operate in sequence to indicate the direction of a turn. Figure 5 shows the tail lights and Figure 6 shows the flashing sequence for (a) left turns and (b) right turns.

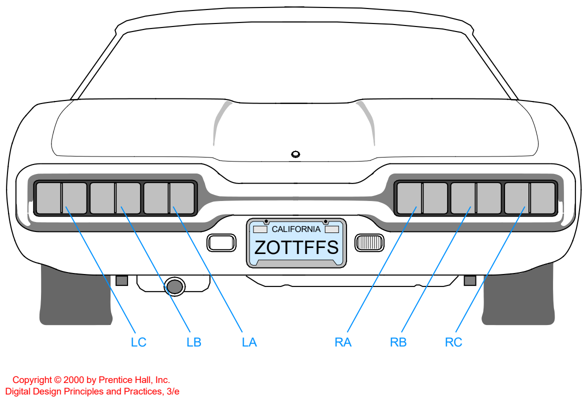
CALIFORNIA
ZOTTFFS
LC
LB
LA
RA
RB
RC
Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e

Figure 5: Thunderbird Tail Lights

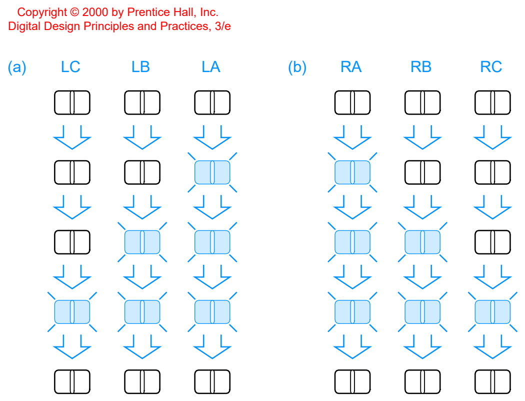
Copyright © 2000 by Prentice Hall, Inc.
Digital Design Principles and Practices, 3/e
(a)
LC
LB
LA
(b)
RA
RB
RC

Figure 6: Flashing Sequence (shaded lights are illuminated)

On reset, the FSM should enter a state with all lights off. When you press **left**, you should see LA, then LA and LB, then LA, LB, and LC, then finally all lights off again. This pattern should occur even if you release **left** during the sequence. If **left** is still down when you return to the lights off state, the pattern should repeat. The **right** is similar. It is up to you to decide what to do if the user makes **left** and **right** simultaneously true; make a choice to keep your design *easy*.
Sketch a state transition diagram. Now, you can write VHDL code for your FSM design using the state transition diagram and simulate it. Submit state transition diagrams, VHDL files, testbench file, and simulation results for the FSMs.