Digital Design and Computer Architecture

Lab 4: Behavioral Verilog FSM

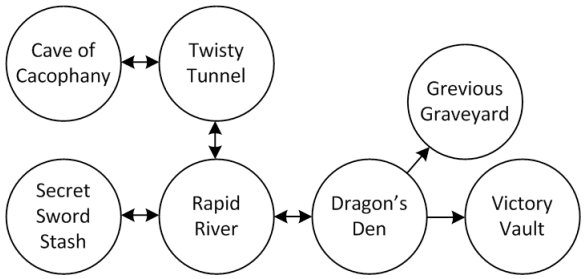
Objective

The purpose of this lab is to learn to design a finite state machine using behavioral SystemVerilog, write your own self-checking testbench, and debug your design. You will not need to download it onto the FPGA board.

In this lab may you will design a **Finite State Machine** (FSM) that implements an adventure game. You will write a testbench to apply inputs to play the game.

1. Adventure Game

The adventure game that you will be designing has seven rooms and one object (a sword). The game begins in the Cave of Cacophony. To win the game, you must first proceed east through the Twisty Tunnel and south to the Rapid River. From there, you will need to find a Vorpal Sword in the Secret Sword Stash to the west. The sword will allow you to pass east from the Rapid River through the Dragon Den safely into Victory Vault (at which point you have won the game). If you enter the Dragon Den without the Vorpal Sword, you will be devoured by a dangerous dragon and be tossed into the Grievous Graveyard (where the game ends with you dead). The game remains in the Graveyard or Vault until you reset it. The game map is shown below.



This game can be implemented using two separate state machines that communicate with each other. The room state machine keeps track of which room you are in, while the sword state machine keeps track of whether you currently have the sword. The machines must communicate back and forth.

Sketch a state transition diagram for each of the FSMs. The state machines should receive an asynchronous reset and a clock. The system inputs are N, S, E, and W, corresponding to the four directions. The system should produce two outputs, WIN and DIE. The room FSM should have one state for each room and transitions based on the directions you could move. Assume the player will apply exactly one input each cycle and will never apply an invalid input.

Write behavioral SystemVerilog for your FSM. Be sure to think about the hardware you want and write the appropriate idiom rather than approaching this like a programming exercise.

Write a self-checking testbench and test vectors to play the adventure game. Provide one set of vectors that demonstrates both the win and die cases. You will need to generate reset appropriately between games. Run the simulation, showing all the inputs and outputs and also the current room.

Look at your synthesized schematic in the RTL viewer and make sure it matches your expectations. Click into each of your two state machines and look at the schematic or state diagram and again verify that it matches your expectations.

What to Turn In

1. Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.
2. State transition diagrams for the room and sword FSMs.
3. Behavioral SystemVerilog code for the system.
4. A single testbench and test vectors illustrating both the win and die cases.
5. Simulation waveforms including the inputs, outputs, and current room. Do they pass your testbench and match expectations?
6. RTL Viewer schematics (including each FSM). Do they match your expectation?
7. EXTRA CREDIT: It is a little known fact that the Twisty Tunnel is located beneath Hoch and that by heading north one can reach the dormitories. Extend your adventure game with more interesting rooms or objects. There will be a prize for the most interesting working enhancement!

If you have suggestions for further improvements of this lab, you're welcome to include them at the end of your lab.