Digital System Design [ECE437]

Lab 3: Adventure Game [PRELAB ASSIGNMENT]

0. Introduction

Total points assigned to all lab 3 assignments- 10 points

In this lab may you will design a **Finite State Machine** (FSM) that implements an adventure game! You will then build the digital logic equations for the FSM in a new Block Design file in a new project on Quartus. You will then simulate the game using the simulation waveforms.

Please read and follow the steps of this lab closely. Start early and ask questions if parts are confusing. It is much easier to get your design right the first time around than to make a mistake and spend large amounts of time hunting down the bug. As always, don't forget to read the entire lab and refer to the "What to Turn In" section at the end of this lab before you begin.

You will design your FSM using the systematic design approach described in Section 3.4.5 of the textbook (Harris and Harris).

1. Design

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 through the Twisty Tunnel and the Rapid River. From there, you will need to find a Vorpal Sword in the Secret Sword Stash. The sword will allow you to pass 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 pass into the Grievous Graveyard (where the game ends with you dead).

This game can be factored into two communicating state machines as described in Section 3.4.4. of the textbook (Harris and Harris). 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 state machine, each state corresponds to a different room. Upon `reset`, the machine's state goes to the Cave of Cacophony. The player can move among the different rooms using the inputs `n`, `s`, `e`, or `w`. Note that these 4 inputs each represent one of the cardinal directions- **N**orth, **S**outh, **E**ast and **W**est. When in the Secret Sword Stash, the `sw` output (which denotes **Sw**ord) from the Room FSM indicates to the Sword FSM that the player is finding the sword. When in the Dragon Den, signal `v`, asserted by the Sword FSM when the player has the Vorpal Sword, determines whether the next state will be Victory Vault or Grievous Graveyard; the player must not provide any directional inputs. When in Grievous Graveyard, the machine generates the `d` (dead) output, and in Victory Vault the machine asserts the `win` output.

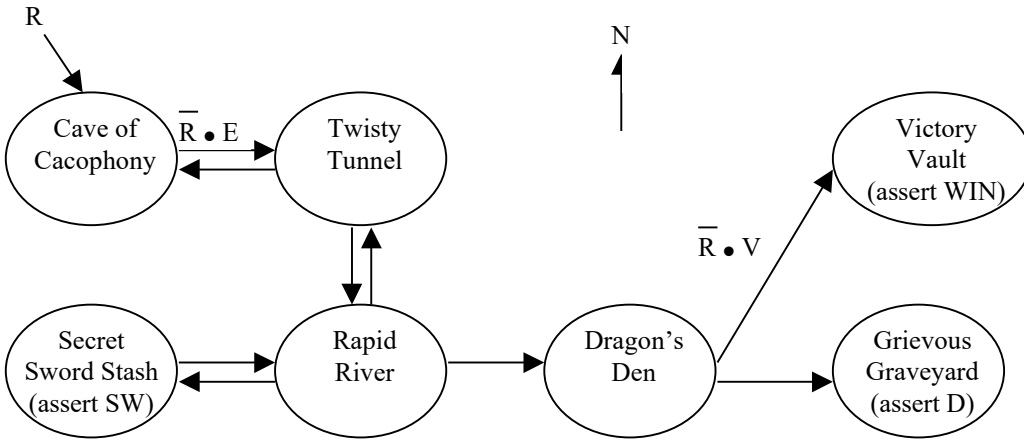

Figure 1. Partially Completed State Transition Diagram for Room FSM

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`.

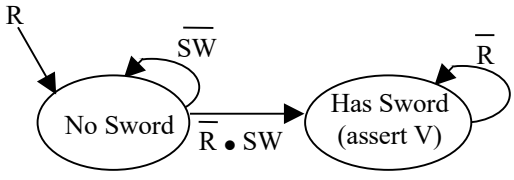

Figure 2. State Transition Diagram for Sword FSM

The state of each of these FSM's is stored using D flip-flops. Since flip-flops have a clock input, this means that there also must be a `clk` input to each FSM, which determines when the state transitions will occur.

So far, we have given an English description for each of the two FSMs as well as an equivalent State Transition Diagram. 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.

[Figure 1 in your prelab report] Complete the State Transition Diagram for the Room FSM (the incomplete one is given above in Figure 1) 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. 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. This is the last step of the design that will be given to you. Be sure to use input and output names (and capitalization) that exactly match these figures so that your design will play nicely when you simulate later.

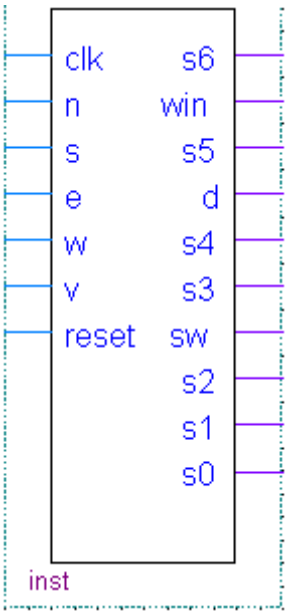

Figure 3. Symbol for Room FSM, showing its inputs and outputs

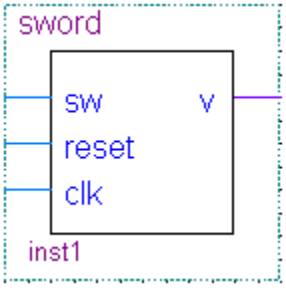

Figure 4. Symbol for Sword FSM, showing its inputs and outputs

[Table 1.1, 1.2, 1.3 and 1.4 in your prelab report] Next, draw a state transition table for each of the two FSM showing how the current state and inputs determine next state. The left side of the tables should have a column for the current state (along with the enumerate used for that state), and separate columns for each of the inputs. The right side should have a column for the next state. Also draw outputs tables, with the current state on the left, and the output(s) on the right. These four tables are a way of representing the FSMs that is an alternative to the diagrams in Figure 1 and Figure 2.

On the left side of the table for the Room FSM, you do not need to fill in every possible combination of values for all inputs (that would make for a rather large number of rows in your table). Instead, for each state you only need to show the combinations of inputs for which there is an arc leaving that state in the state transition diagram. For example, when the input N is asserted and the current state is Twisty Tunnel, the behavior of the FSM is unspecified and thus does not need to be included in the table.[1] Also, you do not need to show rows in the table for what happens when more than one of the directional inputs is specified at once. You can assume that it is illegal for more than one of the *N*, *S*, *E*, and *W* inputs to be asserted simultaneously. Therefore, you can simplify your logic by making all the other directional inputs of a row "don't care" when one legal direction is asserted. By making careful use of "don't cares", your table need not contain more than a dozen rows.

[Table 2.1 and 2.2 in your prelab report] The next step in FSM design is to determine how to encode the states. By this, we mean that each state needs to be assigned a unique combination of zeros and ones. Common choices include binary numeric encoding, one-hot encoding, or Gray encoding. A one-hot encoding is supposed to be used for the Room FSM (i.e. Cave of Cacophony=0000001) since it makes it trivial to output your current state `s0`…`s6`. Make a separate list of your state encodings for each FSM.

[Table 3.1, 3.2, 3.3 and 3.4 in your prelab report] Now rewrite the **tables 1.1, 1.2, 1.3 and 1.4** using the encoding that you chose. The only difference will be that the states will be listed as binary numbers instead of by name.

[In-lab portion of the Lab]

You are now approaching the heart of FSM design. Using your tables, you should be able to write down a separate Boolean logic equation for each output and for each bit of the next state (do this separately for each FSM). In your equations, you can represent the different bits of the state encoding using subscripts: S_1, S_2, etc. Depending on which state encoding you chose, a different number of bits will be required to represent the state of the FSM, and thus you will have a different number of equations. Simplify your equations where possible.

As you know, you can translate these equations into logic gates to implement your FSMs in block design file. That is all that you will do in the in-lab portion of the lab.

2. What to Turn In

Due Date- Before lab 3 is due to be performed in the Lab (that means week of September 22)

Points assigned to prelab Assignment- 2 points for Lab 3 come from prelab assignment report submission

[1] Since the behavior of the FSM is unspecified in cases like this, the actual behavior of the FSM that you build in these cases is up to you. In a real system, it would be wise to do something reasonable when the user gives illegal inputs. In this game, we don't care what your game does when given bad inputs, as long as the current state persists.

You must submit an electronic copy of the following items (in the correct order, and each part clearly labeled) via Canvas. Submit to the Lab 3 Prelab Assignment. These should all be included in a single file (.pdf). Most of these items relate to items highlighted in yellow in this PDF above.

1. Please indicate how many hours you spent on this assignment**.** This will be helpful for calibrating the workload for upcoming labs.
2. A completed State Transition Diagram for the "Room" FSM. Scanned, hand drawn diagrams are acceptable so long as they are neat and clearly readable. (Refer to **[Figure 1 in your prelab report]** section above)
3. Your tables listing (1) next state in terms of current state and inputs and (2) output in terms of current state. You need tables for each FSM. (Refer to **[Table 1.1, 1.2, 1.3 and 1.4 in your prelab report]** section above)
4. A list (one for each FSM) of your binary encoding for each state. (Refer to **[Table 2.1 and 2.2 in your prelab report]** section above)
5. The revised copy of your tables, using your binary encoding. (Refer to **[Table 3.1, 3.2, 3.3 and 3.4 in your prelab report]** section above)

Some Quartus and vlab Tips

- Make sure you don't have any spaces in any of your folder or file names.
- Some CAD tools are case sensitive and others are case insensitive. Never use two different capitalizations of the same signal because some tools may treat them as different signals while others may treat them as the same signal. The easiest solution is to be consistent in your choice of capitalization.
- Make sure you have a backup of the project folder that you created when you click on "Create new project" on launching Quartus for the first time in vlab/in-person at the EE103 lab. Do this by creating a zip folder of this project folder and uploading it to your google drive. This way you will have access to your project files later as well.
- Please refer to instruction set 1 on how to run Quartus from home.
- Please refer to instruction set 5 on how to install Quartus on your own computer.
- Please note that the above two instruction sets only allow you to complete the block design part and the simulation waveform verification part of the Lab assignments. To implement your design on the board, you will need to follow the same instructions as from Lab 1.