ELC 2137 Lab 11: FSM: Guessing Game

Aaron Mendoza

November 17, 2020

Summary

The purpose of this lab is to build a guessing game in which the user presses buttons on a Basys3 board that correspond to certain states in order to win.

The first module built during this lab was the debounce module. Although pressing a button is seemingly straightforward, there is actually something called a "bounce" where the output cycles on and off a few times when the button is pressed or a switch is moved. In order to fix this, I built a debounce module, where it changes between four states: zero, wait1, one, wait0. In order to switch from zero to one, the output from zero must remain on for a long enough time during wait1, in which case it will switch to one. If the output from zero does not remain on long enough during wait1, then it returns back to zero. This eliminates the bounce behavior from pressing the pushbuttons on the Basys3 board. This module has three inputs (in, reset, clock) and two outputs (out and tick). Tick corresponds with if the button was pressed, and out corresponds with how long the button was pressed.

The second module built during this lab was the guess-FSM module. This module contains the main states of the game. The guess-FSM module has six states: s0, s1, s2, s3, swin, and slose. Whenever there is no input given, the FSM cycles through s0 to s3. When there is an input given at one of those states, it will either go to swin or slose based on logic. My guess-FSM module has three inputs (clock, enable, and b) and three outputs (y, win, and lose).

The third module built was just a counter module built from a previous lab. This controls the time it takes to switch between the states of my guess-FSM module. In other words, this controls the difficulty of the game. This is made possible by using a mux, where turning a switch on and off changes which counter is used. The only difference between the counters is the parameter N (number of bits) given within the code.

My top level module guessing-game combines all of these previous modules. Each of the buttons on the Basys3 board(btnU, btnL, btnR, and btnD) feed into the input of a debounce module whose outputs go into the "b" input of the my guess-FSM module. The output of the mux whose inputs are from the counters are decided by sw[0] of the board, which goes into the enable input of my guess-FSM to keep the states switching from s0 to s3 at a constant rate. Finally, I made my LED's correspond to the output "y" of the guess-FSM. This makes four LEDs light up one at a time, switching between each other. If the player wins, then all of the lights on the seven segment display turn on and if they lose, then only dashes appear on the display of the board.

Q&A

1. At what time in the simulation the did debounce circuit reach each of the four states (*zero, wait1, one, wait0*)? The debounce circuit reached zero at
2. Why can this game not be implemented with regular sequential logic? This game cannot be built using regular sequential logic because the outputs do not repeat or exhibit a specific pattern. So, in this lab I implemented a finite state machine. An FSM has a limited number of internal "states" that switch between each other based on the inputs. This is crucial for the game; the states are not just repeating from s0 to s1 to s2 to s3 all the time. Instead, s0 can go to swin, slose, or s1 based on the input, so an FSM is needed in order for this game to work.
3. What types of outputs did you use for your design (Mealy or Moore)? Explain. Mealy because these outputs depend on not only the current state, but the input as well. So, reaching the output of win, lose, or hold is dependent on the button pressed or if any of the buttons are pressed and the current state that the machine is in. Therefore, Mealy outputs are what I used for my design.

Results

In this section, put your simulation waveforms, results tables, pictures of hardware, and any other required items.

Code

code