SANTA CLARA UNIVERSITY	ELEN 21 Fall 2021	Sara Tehranipoor, David Zheng, Vinay Andra, Allen Shelton, Soroor Ghandali

Laboratory #7: Slot Machine Inspired Game

For lab sections Monday-Friday November 8-12, 2021

I. OBJECTIVES

In this laboratory you will:

- Learn to use a comparator and a counter in a circuit
- Integrate several smaller designs to build a more complex circuit

PROBLEM STATEMENT

A typical slot machine is shown to the left. It is a game of chance in which the player initiates the spinning of three separate displays and tries to stop the display motion at a time when all three display images match.

A slot machine inspired game will use two seven segment displays in place of the three displays in the example. Each seven-segment display will show one of four custom designed symbols. Status information about the game will be shown in a third seven segment display.

When a button is pressed, the two displays showing the game symbols begin to change at different rates. The displays continue to change as long as the button remains depressed. When the button is released, the symbols stop changing. The objective of the game is to try to stop when both displays show the same symbol.

- A "win" symbol is shown in the status display if the two displayed **symbols match** when the button is released.
- A "lose" symbol is shown in the status display if the two displayed **symbols do not match** when the button is released.
- While the button is depressed and the symbols are changing, a neutral symbol appears in the status display to indicate that the game is in progress.

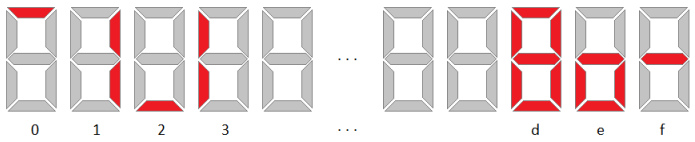
The game has four speeds: **test, slow, medium, and fast**. At test speed the symbols change so slowly that it is easy to win and easy to test the function of the circuit. As the speed increases, the game becomes more challenging. The speed is selected using two input switches to specify the four possible speeds.

II. PRE-LAB

In the pre-lab you will focus on the **display control** for the game. The display will show the two symbols which are continuously changing while the pushbutton is pressed. These symbols are defined by 4-bit values **code1** and **code2.** The game status display is controlled by the 4-bit value **codeS**. First you will design a new seven segment display controller using your own custom-designed symbols, and then you will design the logic circuit to generate **codeS** values. In the lab you will design the logic circuit to create **code1** and **code2** values.

1) **Define your custom symbols and design your special symbol generator.** Your game will need ***four special game symbols*** and ***three status symbols***. You will design your custom symbols using the method that was used to create numeric displays in Lab 3.
   a) Refer to the seven-segment tutorial in the "Seven_Segment" file used in Lab 3.
   b) Copy the text file "bin_7seg_temp.txt " to "spec_7seg_temp.txt " to create a new module "*spec_7seg.*" This module will use the same input and output specification, but it will generate your specially designed character patterns instead of 0-9 and A-F.
      i) Hex inputs 0, 1, 2, and 3 will select your four special symbol shapes for the spinning display used in the game
      ii) Hex inputs d, e, and f will select the status symbol shapes to indicate "win", "lose", and "in progress."
      iii) The other nine input possibilities will not be used, but are available "for expansion" and may be used to add special features to your game.

      An example is shown below, but **you may design your own symbols**.



2) **Design the circuit to create codeS, the code for the game status symbols.**
   a) Write a Verilog module *my_status_code* with two single bit inputs **PB1** and **EQ**. **EQ** is 1 when the two game symbols are the same, and **PB1** is 1 when the pushbutton is depressed. The logic circuit output is the 4-bit **codeS** value ("d", "e", or "f" See 1bii above). From the problem statement description of the "win", "lose", and "in progress" displays, design the logic to generate the 4-bit value.
      i) How should the two ***most significant bits*** of the special symbol code be set?
      ii) Write the logic expressions for the two ***least significant bits*** of the special symbol code.
   b) Write a **Verilog module** for my*_status_code* using **continuous assignment**.
   c) Write another version of the **Verilog module** using a **case statement** in an always block.
   d) Write a third **Verilog module** version using **if-else statements** in an always block.

3) Draw a circuit diagram for the complete circuit for the game using:
   a) Three instances of *spec_7seg.*
   b) One instance of *my_status_code*.
   c) One instance of *clock_module*, a module provided with this lab which generates two clock signals at different rates. The inputs are the system clock, **clock50**, and two switches, **SP1**, and **SP0**, which will control the speed of the output clocks. The outputs are **clk1** and **clk2**.
   d) One instance of a circuit that will be created in the lab. The inputs will be **clk1** and **clk2**, and **PB1**. The outputs are **code1**, **code2**, and **EQ**.
   e) Show all the connections between the modules.

For the prelab, submit the circuit diagram and Verilog.

III. PROCEDURE

1. **Choose one of the three versions of the Verilog module,** *my_status_code,* **designed in the prelab, and simulate it to verify its operation.**
2. **Create new components:**
   a. Compile the modules *my_status_code* and s*pec_7seg*.
   b. Compile the modules *clock_module* and *clock_counter* provided with this lab. Please note that the *clock_counter* is a module that is instantiated inside the *clock_module*. It will not be visible at the top-level schematic.
   c. Go to libraries→megafunctions→arithmetic and select *lpm_comparator* to get a 4-bit comparator that will create three outputs: equal, greater than, and less than. Select parameters for binary values, unsigned comparisons, and no pipelining.
3. **Create a circuit to generate the game codes.**
   The new module will create 2-bit game code patterns using the two clock inputs. Each clock will individually create one 2-bit pattern that will cycle through the four possible two-bit patterns.
   a. Go to libraries→megafunctions→arithmetic and select *lpm_counter* to get an 2-bit binary up counter with a clock enable. Select parameters for a width of 2, a count direction of up, plain binary, clock enable input, and no other options.The binary up counter is a circuit that will count in binary from 00 to 11 and then start over again. The rate at which it counts is controlled by a clock input. The counter can only count in response to a clock input if it is enabled.
   b. In your circuit, the up counter's clock enable will be connected to input **PB1** and the clock input will be connected to **clk1**. The outputs are q1 q0 where q0 is the least significant bit. This will be the two least significant bits of code1.
   c. Repeat step (d) except make a down counter instead of an up counter. The only difference will be that the direction of the count is down instead of up. The counter will count down from 11 to 00 and then start over.

d. In your circuit, the down counter's clock enable will be connected to input **PB1** and the clock input will be connected to **clk2**. The outputs are q1 q0 where q0 is the least significant bit. This will be the two least significant bits of code2.

4. **Connect your circuit components in a top level schematic.**
   a. Add the components from your circuit diagram from your prelab.
   b. For the circuit designed in the lab, add
      i. The two counters
      ii. The comparator
   c. Connect the circuit components.
5. **Connect inputs and outputs.**
   a. Connect PB1 to a push button input.
   b. Connect SP1 and SP0 to two adjacent switches to be the display speed control.
   c. Connect the outputs of the three special seven segment display controllers to seven segment displays.
   d. For diagnostic purposes, connect code1 and code 2 to LEDs.
   e. For diagnostic purposes, connect **clk1** and **clk2** to two LEDs so you can see if the counters have an input clock.
6. **Test your circuit**
   a. Verify that the counters are NOT counting when the pushbutton is NOT depressed.
   b. Verify that both counters are counting when the pushbutton is pressed.
   c. In the very slow test mode, (SP switches are 11), verify that the symbol display sequences. One should display your special symbols in the order 0, 1, 2, 3, 0, 1, 2, ... and the other should display symbols in the order 0, 3, 2, 1, 0, 3, 2, ...
   d. Verify that the status display works correctly.
   e. Verify that the speed select switches work correctly.
7. **Play the game**
   a. Have each lab partner do ten trials at each speed and record how many times you win at each speed.

IV. REPORT

1. Include an introduction, circuit diagrams and Verilog code from the prelab and lab.
2. How often were you able to win using each mode? Include a photo of your board for both a win and a lose.
3. From the prelab, which form of the **Verilog module** *my_status_code* was the easiest or most intuitive for you? Explain briefly.
4. If you want to use 8 symbols instead of only 4 to make it harder to win the game, list all the things you would need to change.