UNIVERSITY OF NEVADA LAS VEGAS. DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING LABORATORIES.

Class:	**CpE200L**		Semester:	**Fall 2020**
Points		Document author:	**Carlos Georges**	
		Author's email:	**Georgc4@unlv.nevada.edu**	
		Document topic:	**Final Project**	
Instructor's comments:				

FPGA Slot Machine

Junior Design Competition

Fall 2020

Video Demonstration

Design Files

1. Abstract

Slot Machines are ubiquitous in Las Vegas. They are huge revenue generators and have evolved from mechanical machines with moving parts to almost exclusively digital machines today. This project demonstrates an implementation of a slot machine on the DE2 board.

This implementation uses a mouse for ease of use, and the game is shown through the onboard displays (7-segment and LED’s). The game mechanics were designed in SystemVerilog, a popular Hardware Description Language. The game uses a truly random generator to produce results. Upon the click of the mouse, the machine samples several counters that are changing fifty million times per second, the results of the counters are used to produce an outcome. The frequency of winning clicks is easily customizable. The design also features a betting aspect. A user can bet a spin, and their credits will respond according to the result. The jackpot amounts are easily customizable.

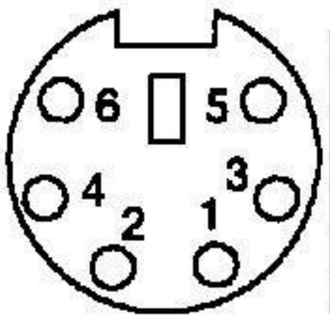
As for the psychological aspect of a slot machine, this design features a suspenseful spin animation that can easily be made longer or shorter and flashes red/green LED’s on a win or loss respectively. Once a user has run out of credits, the game will no longer allow them to bet, and requires the user to buy more credits, this feature is realized by a push-button.

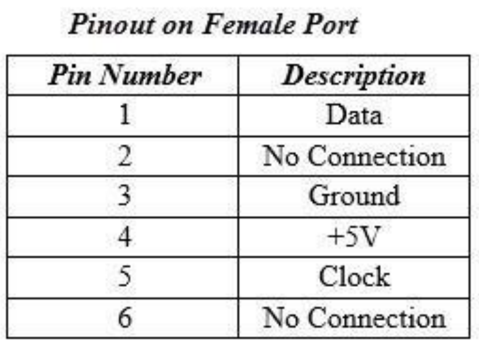
Slot machines, unfortunately, are designed to be addictive and to maximize profits. The features implemented in this design make the game engaging and if real money was involved, it can easily turn a profit.

2. Project Documentation

<u>Mouse Implementation</u>

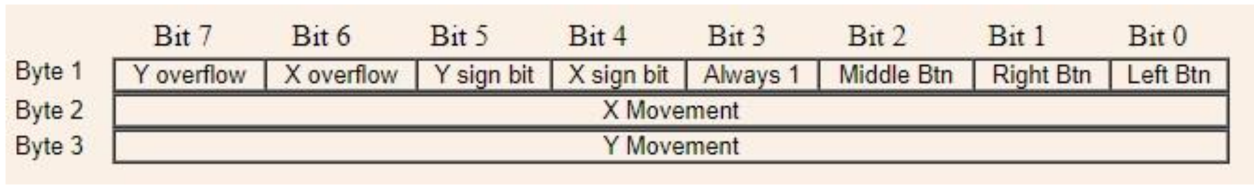
The first step in my project was interfacing the PS/2 port on the board. I purchased a standard PS/2 mouse and got to work. After some research on the PS/2 core on the DE2 board, I knew I needed a command module to listen to the mouse.





Pinout on Female Port

Pin Number	*Description*
1	Data
2	No Connection
3	Ground
4	+5V
5	Clock
6	No Connection



	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

The command module is essentially a complex state machine that has the proper timing requirements to read and store the bytes sent by the mouse. After a command module has been completed, a driver module is necessary to produce useful data from the bytes sent by the PS/2 device. An Altera PS/2 Core command module was readily available online and needed a driver to interpret the data coming from the mouse. With the permission of my instructor, I used the command module and wrote my own driver for the mouse.

The PS/2 mouse protocol sends 3 1-byte packets to be read. 2 of these bytes are reserved for the movement of the mouse. Since my design only uses the buttons on the mouse, these bytes are ignored. In the 3rd byte sent by the mouse, the buttons account for three of the eight bits. My driver extracts these three useful bits and discards the rest. The output of the driver is three bits that are high while their respective button is pressed.

Now that I had useful data from the mouse, it was time to use them in the design of the game.

Random Number Generation

At the core of all the random numbers used in this design lies a counter of some sort. This design uses 3 counters to produce a random sample on any given click. The three counters have different purposes but all operate similarly.

The base module is a 12-bit bidirectional counter. A 50Mhz clock is used for the counters, meaning the counter increments/decrements once every 50-millionth of a second. Upon the click of the mouse, the counters are sampled and whatever numbers they stopped on are used in the logic to produce a win/loss.

This technique of producing random numbers is extremely effective and for all intents and purposes it is truly random and unpredictable. Since each counter is 12-bits, there are 4096 possible combinations, meaning the frequency of wins can be made extremely precise, (+/- 0.025%). This process will be explained in the next section.

Producing a Result

Once sufficiently random numbers are obtained, then comes the fun part, producing outcomes. Here’s what we have so far. On any spin of the reels, we now have essentially 12 random bits from each counter. How you decide if a click is a win or loss is completely arbitrary and easily customizable.

For my design, a click is a win if and only if the bottom 3 bits of a counter equals ‘7’ or ‘1’, I chose these because these are somewhat ‘lucky’ numbers, but any numbers could have been selected. Since I only check the bottom 3 bits, there are 8 possible combinations on any click. Selecting 2 of these combinations means that on a given spin, there is a ¼ chance of winning, a very generous slot machine for demonstration purposes.

This is where the frequency of wins can be modified. If we look at more bits of the counter, we can select more arbitrary numbers that will produce a win. For example, if we look at all 12 bits, we have 4096 combinations to choose winning numbers from, allowing us to have a very precise and controlled win percentage. This is extremely beneficial and practical when real money is involved.

The reel displayed on a win is determined by another counter running at 20Hz, this counter is vastly slower than the counter used to determine a win, but it need not be as fast or as random. This counter runs on 20Hz for animation purposes, which will be explained in the **Psychology** section. The winning reel is simply the bottom 3 bits repeated three times, although you can easily customize what displays upon a win, (words, other number combinations).

There is another counter that determines what reel displays on a loss. This counter also runs at 50MHz and the reel displayed is simply the 12-bits (3 hex digits) that were chosen on the click.

<u>Gameplay</u>

The game runs like a regular slot machine, where you have a certain amount of credits, and you can bet on any click. The bet is controlled by the mouse and can be incremented or decremented with the right click or left click, respectively. The user can bet 0 to 15 credits on any given spin. The user also has a bank of credits, the maximum value of which is 99.

Each winning reel has a different worth. The worth is simply the reel itself multiplied by the bet. For example, if you hit 333, and you had bet 10 credits on that spin, your credits would go up by 30. This multiplier is easily customizable as well. Customizability is a common theme for the whole project since much of the gameplay is entirely arbitrary.

On a loss, the user's credits will simply decrease by however much was bet. If the user runs out of credits, the game is over and will no longer allow any bets and the user must buy more credits to continue. In my implementation, this 'buy-in' feature is simply a push-button to add 50 credits, but in reality, this would be behind a paywall.

Psychology

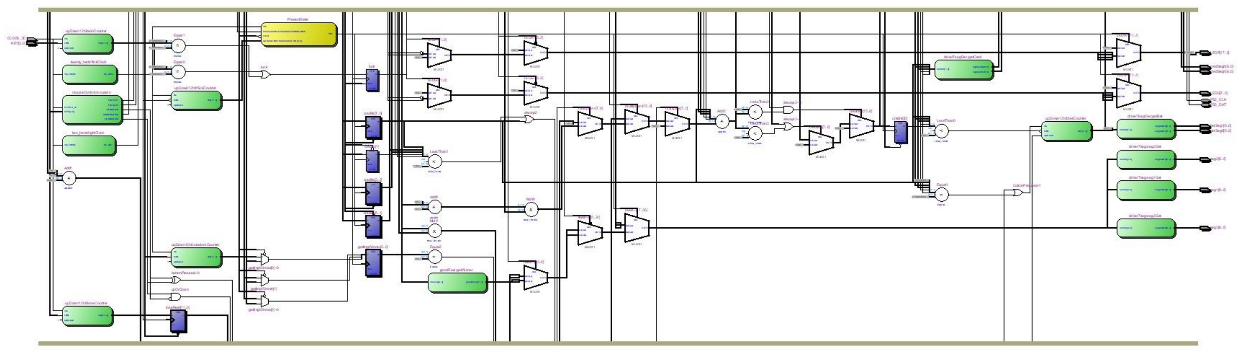
Slot machines affect our brains tremendously. It is one of their defining features and contributes highly to their success. For my design to be a viable slot machine, I had to implement the psychological factor of the gameplay somehow.

First, the user is presented with an easy-to-grasp interface: a device that everyone has used before (mouse,) and 3 displays. One display shows the credits, another the bet, and the last is the reel itself. The learning curve to play the game is practically non-existent. That is one factor that makes the game engaging.

Next, once the user understands the interface, they will place a bet and click the middle button on the mouse to spin the reel. Once the mouse is clicked¸ the reel will 'spin'. This spinning animation is achieved with an FSM and the duration of the animation is easily controlled. The animation adds an element of suspense to the game and all slot machines ever made have this feature of a delayed result. It makes the result more surprising in the end. The FSM displays a 20Hz counter on each of the three digits. To the human eye, this 20Hz counter produces a satisfying flickering/spinning animation.

As soon as the result is displayed, two important changes happen. First, the onboard LED’s will animatedly flash green/red on a win/loss respectively. Second, the credits are updated. The DE2 board has 18 red LED’s and 8 green LED’s at the output stage. I tested flashing all 18 red lights on a loss and it had a negative effect on the player, since it was bizarre to have a more prominent display on a loss. The user would expect the display to be more prominent on a win, so my design uses only 8 of the red LED’s and uses all 8 green LED’s to be more symmetric and to make a win more enjoyable. Another design feature is that the credits will only update once a result is displayed. This also adds to the suspense of the game because recalling that a win/loss is decided as soon as the button is clicked, it would be quite discouraging to see your credits subtracted as soon as you clicked.

Through my own testing, I found the game to be quite enjoyable and engaging. Remember that the money is imaginary, so if real money was involved, this game certainly has the potential to be addictive and can easily turn a profit given that the proper win frequency is selected to keep a user engaged.



RTL VIEW OF MACHINE

3. Conclusions & Summary

In conclusion, this machine is an excellent demonstration of the curriculum covered in the CpE200L course. It utilizes various HDL concepts that we covered in unique and fun ways. Through designing this machine, my ability in SystemVerilog increased dramatically. I used behavioral and structural code, instantiated many modules, had several state machines in the gameplay and had to overcome many bugs in my code. From inferred latches to real headscratching problems, I managed to research and learn more about the language to eliminate the bugs. The final design implements all the features I had in mind when I started to design the project and now, I only have further improvements.

Of course, one improvement/requirement is to make the credit register controlled by a real money sensor to keep track of actual currency. Another improvement would be to learn how to interface the LCD module on the DE2 board to display the credits/bet more professionally. An improvement that can be made to the psychological aspect is to include audio support and play different sounds for different states of the machine.

Overall, I enjoyed building this machine. I certainly learned a lot and overcame many problems. I will be including this project on my resume, and hopefully I can include a ‘first-place’ descriptor next to the entry. I look forward to entering next semester’s competition with an Arduino project.

Thank you to the professors and TAs for the rich instruction and rigorous course material.