University of **Southern California**

Asteroid Game

2013 Fall EE201L- Final Project

Richard Chan - EE

Sam Chordas - CECS

Abstract

For our final project we created a simple version of the arcade game Asteroids. Our game controls are very similar to that of typical arcade game controls. Using Verilog, we were able to code the game onto a Nexys-3 board and output the game screen using the board's VGA port. The game uses a Digilent joy stick to control the direction, while a button on the board fires the jet's weapon.

Introduction and Background

Throughout the semester, we learned how to create many modules and systems through lab assignments that we were able to apply to our final project. For example: the debouncer module and the concept of state machines. We used the debouncer module frequently in our game. It helps us achieve the desire result from pressing buttons; the shooting functionality is an example of this. We also created ways to start/stop the game with the help of state machine.

The Design

User Interface:

The control of this game is based primarily on the joy stick and the five buttons: left, right, center, up and down. The joy stick controls the movement of the jet. The jet has three horizontal movement speeds and two vertical speeds; both direction speeds depend on how far the joy stick is being pushed away from the center. The jet is able to move diagonally as well. The center button on the board shoots bullets. The user can hold down the center button and the jet will continue shooting until the user releases the button. However, to make this game challenging, the shooting will paused briefly every 22 bullets and then continue shooting (simulating reload). The Left and Right buttons control the speed of the bullet. Right increases the speed and Left decreases the speed. The default bullet speed is 7 and it can go up to 11 and down to 6. To start the game, user will need to press the Down button on the board once. The Down button can also be pressed again to pause/unpause the game. A fun extra feature of this game is being able to change the jet's color. By adjusting the position of the switches (Switch 0 to Switch 7), the user can adjust the 8-bit RGB setting for the jet. The Right two switches (0 – 1) change the blue bits and the other 6 switches split the left 3 switches and right 3 switches, they change red and green bits respectively. If an asteroid hits the user, the game will stop and the user will have to press the Up button to reset the game.

The 7 Segment Display on the board will tell the user's current score. The score increases by 1 each time an asteroid is hit. The Left four LEDs tells the user the current game level (1 to 4). Level 1 being the easiest and level 4 the hardest. The Right four LEDs tell the user the current bullet speed. Default bullet speed is 7 and the LED is displayed as binary. LED 3 is the MSB and LED 0 is the LSB. The higher the value the faster bullets.

Implementation:

In this game, arrays are used to store the pixel positions of the bullets and asteroids. We declare 4 eight-object arrays that will hold the x and y positions of bullets as well as asteroids. Jet positions are controlled by two 10 bits registers and these register's values (movement of the jet) depend on the number output of the joy stick. The Joy stick transmits x and y positions to our program and the values have a range from 0 – 1023. We determined the stable value of the joy stick (when it is at rest in the middle) and use it as the center of the control knob.

All the buttons are tied to a debouncer module so we can adjust the desired output.

When the user hits the Center button, the x and y coordinate of the jet will be recorded and stored as the bullet's starting position. The bullet counting mechanism will also increase each time a new bullet position is entered. The maximum bullet count is 8. If the game is in START mode, the bullet's y position will keep decreasing until it reaches the edge or hits an asteroid. It will hit an asteroid if the y position of the bullet is less than the asteroid's y position when their x position are close. The shooting will stop briefly every 22 bullets as the users keep holding down the Center button to simulate reloading.

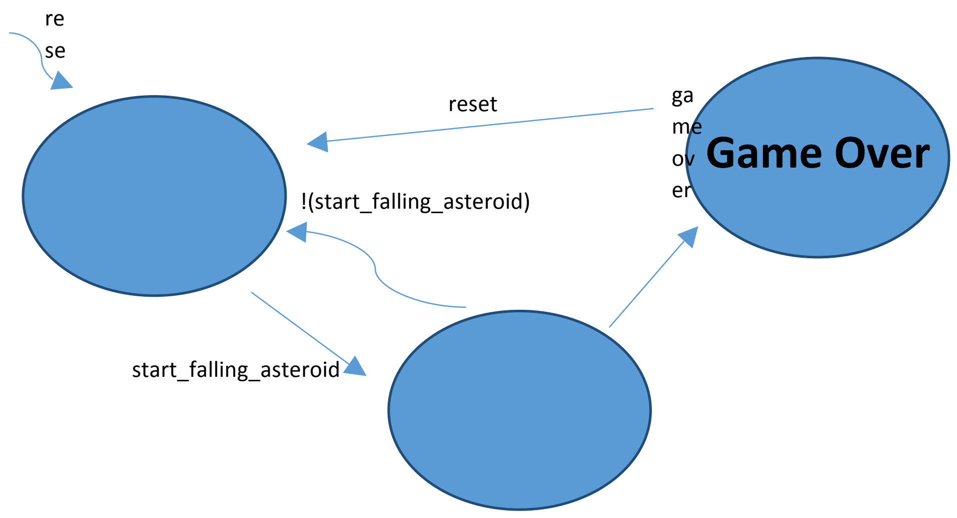
We modified a LFSR counter from *outputlogic.com* to produce a 10-bit random number. The asteroid will take a random number from the LFSR counter and store it as its x position. If the number is greater than 620, it will right shift the number by 1 bit so that the number will fit on our screen. After obtaining an x position, the asteroid will increase its y position from 0 to 510. The speed of the asteroid will be determined depending on the game level. There are 4 game levels and the level will increase every 6.8 seconds. Similar to the bullets and asteroids, if any asteroid's x and y positions are extremely close to the jet, it will hit the jet and the game will stop and all objects turn red as indicator that the game is over. When the game is over, only the reset button can start the game again. The generation of the asteroid depends on the game level and will have a rate of approximately 2-4 asteroids per second.

We used the vga_demo as the foundation of coloring each pixel. As the counterX and counterY pass each pixel, numerous "if" conditions determine if any objects exist in that pixel and color the pixel accordingly. If no object exists in that pixel, it will be painted black.

To start or pause the game, the user will have to press the Down button. A 1-bit register will invert its value each time the Down button is pressed and the movement of all asteroids, bullets, and jet act according to this register. The Left and Right buttons control a 5-bit register which has a default value of 7. The value of the register determines how much of the Y position of the bullet needs to be subtracted.

State Machine:

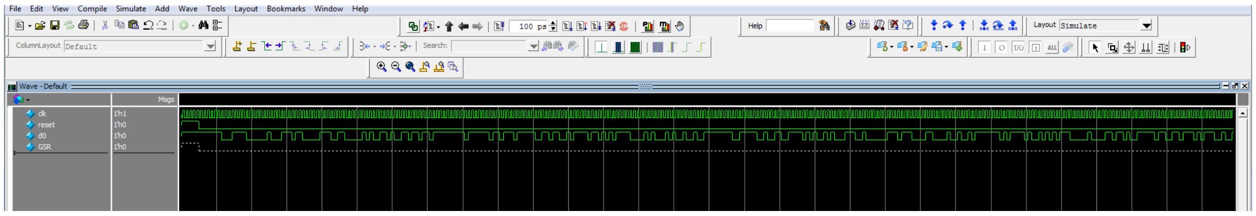
There are total of three states in this design: Start, Pause and Game Over.



Initially, the state is at PAUSE. When start_falling_asteroid becomes 1, it will enter the START state. In this state, all the objects are allowed to move and the user can interact with the game. However, when an asteroid hits the jet while in START state, the state changes to Game Over and everything will stop. The user is required to press the Up button to enable the reset signal.

Test Methodology

We tested the program every time we made a small change in an attempt to track and fix bugs efficiently. As a result, we regenerated the bit file every time we made a change in our code. The downside to this method was facing the long compile time before we were able to run the program on the board to test our changes. However, it allowed us to achieve more concrete results as opposed to testing it in the test bench file. The random number generator was not generating the desired result when we first implemented it. Since the output signal is much faster (100Mhz), we wrote a test bench to see what was happening in this module. Using this method we finally determined our desired signal. Below is the waveform of our modified LSFR counter.



```
clk    1'h1
reset  1'h0
d0     1'h0
GSR    1'h0
```

Conclusion and Future Work

This project provided us the opportunity to apply some of the knowledge we gained in lab to practical projects of our own creation. We were exposed to new problems we had not faced before without the same guidance we would have received in lab. Due to this, we gained practical engineering experience that will be useful in future engineering ventures. In the future, we will be able to apply the same knowledge of hardware, software languages, and algorithms to complete similar projects; furthermore, we will be prepared to apply other tools that we learned about in lab that were unused in this project but may be necessary for others.