CS M152A

Introductory Digital Design Laboratory

Lab Module #4

Self-Designed Program

Name:	Huiwen Zheng
	Jiaming Zheng
	Rujun Yao
Section:	LAB 4
TA:	Luna Li
Date:	2 December 2016

Introduction

The purpose of this lab is to work on our self-designed project with the technics and knowledge we learnt in the previous lab about utilizing the FPGA board. In this project, the initial project idea proposal, the creativity of our ideas, the complexity of the end of the result, the quality of the design and this lab report are all grading criterion. Our design is a FPGA version of the car racing game on retro handheld game consoles back in the 90's. The project will incorporate Nexsys Board's buttons for controlling the game, the VGA adapter on the Nexsys Board for displaying the game interface, and the Seven Segment display for showing the score.

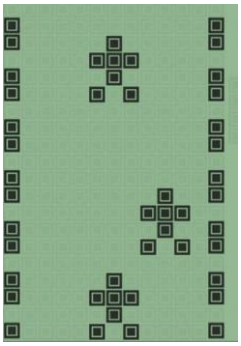

Figure 1: Inspiration of the game.

The main purpose of the game is to control a car by moving it left or right on a track with three lanes. As the car moves, it will encounter traffics— other obstacle cars blocking specific lanes. The goal of the player is to control the car to change lanes rapidly so that it can dodge every obstacle car in front of it. When the car hits any obstacle cars, accident happened, and the game is over. If the car is able to survive without hitting the obstacles, the game will continue and the player can earn more points. Also, as the player survive longer in the game, the difficulty increases as the speed of the overall traffic will increase.

Game Design

Specifically, the game will support the following features. These are slightly different from the ones in the original proposal, as revisions were made while building the actual product.

- Two buttons on the board will be used to control the racing car to move left or right in order to avoid hitting any obstacles.
- The point will be displayed on the Seven Segment Display and will be updated whenever the player gains more score and will stop adding points when game is over.

- The game interface: the tracks and all racing cars will be displayed on the screen through the FPGA using VGA connection.
- Another button will be implemented to restart the game.
- Extra feature that as the player survives longer in game, the overall traffic speed increases.

The final project takes different inputs. There are three buttons: two of them are used to control the car, making it moving left or right; and one button in the middle is used to restart the game. And our design has different outputs: one of them is the seven-segment display output and the other output are the VGA outputs, controlling different colors, and some specific output that are required to make the VGA display functions normally.

The overall design is shown in the high level diagram in Figure 2. Detailed descriptions of modules are discussed in the next part.

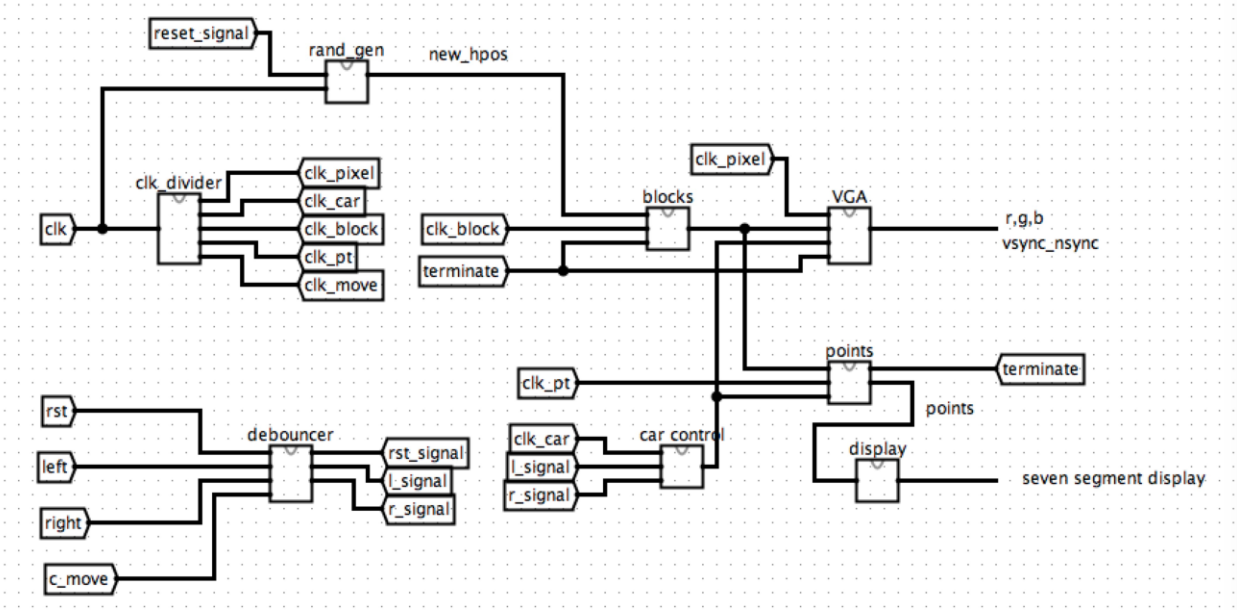

Figure 2: Schematic diagram of our design

Design Description

The design description part is mainly focus on the detailed explanation of the modules.

1. Clock Divider

The clock divider is one of the most important modules of our design. It takes the 100 MHz master clock signal as the input. The job of the clock divider is to divide the clock into five different frequencies to satisfy our design requirement:

clk_move	500 Hz
clk_car	200 Hz
clk_pixel	25 Hz
clk_pt	10 Hz
clk_block	100 Hz

Table 1: different output of the clock divider.

2. Debouncer

This module is used to stabilize the signal generated by the push buttons on the board. Originally, when you press the button once, the button will generates an oscillation of signals due to the poor metal contact and thus causing considerable noise in the input. The debouncer module is used to filter out such noises. Thus the output signal can be delivered more precisely.

In our design, three signals need to be debounced: the left and right control buttons and reset button.

3. Blocks

This module is used to generate all the obstacle cars. The obstacle cars are generated randomly on one of the three lanes and will constantly move down towards the bottom of the screen.

To achieve this, we have two sub-modules to control the positions of the blocks. The horizontal positions are given by a LFSR random number generator. We generate a random number from 0, 1 or 2 to decide whether a block should be in the left, middle or right lane, respectively. The vertical position is designed in a way such that blocks will be continuously generated from the top of the screen. While the vertical position of a block is within the active display region of the screen, we simply let it fall down according to the signal clk_block. Once a block drops outside of the active display region, we simply wrapped in back to the top of the screen by resetting its vertical position. Thus, blocks will be falling down continuously.

Another issue is to avoid the situation where all three lanes are blocked by obstacle, leaving the player with no chance to win. We therefore design the game so that there can only be a maximum of two blocks on the screen. Since a new obstacle will only show up when its old self disappear, the dead situation is avoided.

For the speeding up mechanics, instead of using a constant clock for the blocks, we implement clk_block such that its counter value is decreased by 1000 every second. The clock cycle will stop speeding up if it hits the maximum value we set. In this way, the movement in vertical direction for the blocks will be increased over time, thus increasing the difficulty of the game.

4. Car Control

The car control module is used to control the player's car. The module takes the debounced left and right signals and the car_clk signal as inputs, and adjust the horizontal position of the car on the screen. Basically, we modify the horizontal position of the car by ±2 unit depending on whether the left or right signal is received. Special cases are when the car collides with the left or right boarder of the screen. In that situation, we simply stop modifying on that direction.

5. Points and Seven Segment Display

The points of the game is calculated in the points module, and its result is displayed on the board. An anode counter and a binary decoder are used to decode the value that will be displayed.

The points are calculated at each rising edge of clk_pt and the point is simply incremented by 1 each time. In this way, we are able to achieve a points system that depends on how long the player survives in game. The terminated condition is also tested in the points module. Whenever the player's car collide with a block, the terminated condition is set to one, points will stop adding, and the game is paused, awaiting for a reset signal.

6. VGA Display

The VGA module will use two counters to traverse through each pixels units on the screen and thus determining the color of each pixel. We pass in all the position data of the blocks and the car, so that scanning through the entire screen, the two counters know the specific regions that different color should be applied. For example, the blocks are set to have color of dark blue and the player's car have the color of green.

Results and Final Product

Since the project is a game, the testing are done gradually throughout the development process. First of all, we test the VGA display to find a suitable display boundary so that we can proceed with all the other elements. Then we implement the randomly generated and falling obstacle cars. After that, the car control is implemented. We spent relatively longer time in adjusting the left and right push buttons to achieve a stable and smooth movement of the car. After that, the points and display modules are implemented. Figure 3 shows the photos of our game interface and the points displayed on the seven segment display.

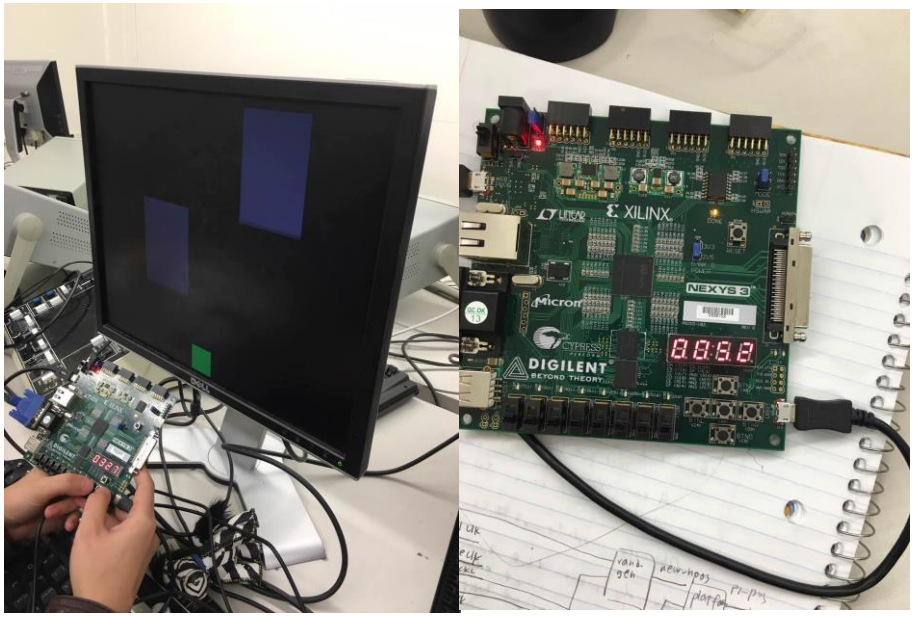

Figure 3: Photo of the design

Conclusion

The basic features proposed in the original design plan are mostly finished. Incorporating FPGA board, VGA display and knowledge from previous labs such as clock divider, the lab is definitely a fun and informative experience. There are also some major difficulties we encounter during the process. First, we are unfamiliar with how the VGA display works so it takes us some time trying to figure that out from the sample code. Our original idea was to design the cars so that they are better looking and more similar to the game on the console. However, we are unable to display complex shapes correctly on through VGA. Therefore, we ended up using only

rectangle and square blocks as our racing cars. Also, we spent relatively more time on implementing suitable sensitivity level for different buttons. Especially, the left and right signal debouncer took us the most time in order for the movement of the car to be both accurate and stable at the same time.