Lab 7
Rahul Arora
3-13-20
Section B

Description:

The purpose of this lab was to learn how to use the VGA monitor of the Basys3 board, as well as our previously practiced design skills to build a game where the objective is to ensure that a car stays on a rainbow colored road. The road segments constantly scroll vertically, while the horizontal positions of the segments are randomized relative to the horizontal position of the segments below them. The car is steered left and right using buttons L and R on the Basys3 board, with btnC as the start button. In addition, there are accompanying LED animations on the board's LEDs for when the game starts and is in progress. There is also a score counter that is displayed on the 7 segment displays, which flashes when the car falls off the road and the game is over.

Methods:

Part One: VGA Display

Build a module that is responsible for the entirety of the VGA output. The VGA display's current pixel is determined by two counters that count the horizontal and vertical pixels of the 800x525 pixel grid. Horizontal sync should be high at all pixel columns except 655-750 and vertical sync should be high for all pixel rows except 489-490. The horizontal counter should always be counting up, and should reset at the highest number horizontal row (799). The CE input for the vertical counter should be the reset for the horizontal counter. In addition, the vertical counter should reset when it equals 524.

The module should take inputs for the current position of the 7 road segments, vertices of the car, width of the road segments, and the clock. Additionally, other inputs can be used for certain features like making the car blink and for timer inputs that indicate when a certain amount of seconds have passed. The outputs of this module are the horizontal and vertical sync of the display, vga color outputs, a crash output that indicates when the car is no longer in contact with the road, an active output that indicates when the horizontal and vertical counters for the VGA display are within the active region for the VGA display, and two outputs for current pixel that the VGA display is displaying.

Part Two: Road Segments

Create a new module for determining the positions of the rainbow road segments. Like for the VGA display, use vertical and horizontal counters for determining the position of the road segments. The module should take in inputs for clk, buttons L and R, load (start game), reset (restart game), the randomized offset, width of the road segments, the frame signal which is responsible for moving the segments vertically, starting reference x and y points, and a "prevX" input which is the x position of the road segment below the current, which is used with the offset to determine the next x position of the current segment. The button inputs are ANDed with the frame signal to move the segments left and right at stable rates and act as Up and Dw inputs for the horizontal counter of the road segment. The vertical counter has the frame signal as Up input and 1'b0 as Dw input. The x position for each road segment is reset every time the segment scrolls off the bottom of the screen. Therefore, it respawns back on top with a new x position. I use the center of the bottom of the road segment as the reference point for the segment.

The outputs of this module are the updated X and Y position of the reference point of the road segment. These outputs are taken from the Q outs from the horizontal and vertical counters. This output (for each segment) is taken as input in the VGA display. Within the VGA display, another module called vga_seg_control takes the position of a segment, the current position of the VGA counters, and the segment width as input. It then determines if the segment is to be currently be displayed for X and Y and provides two 1 bit outputs that can be ANDed to determine when a segment is to be displayed. These one bit outputs can be ANDed with different colors (in hex) to provide the colors for each road segment.

Additionally, determining whether the car has "crashed" or fallen off the road is also done within another module within the VGA display module, using the reference points of the segment as input, in addition to segment width and the position of the VGA display counters. There are seven instances of this module to determine, for each road segment, whether or not the car is in contact with at least a single road segment. The output of this module is a single bit, which is high when the car is in contact with the road segment. These single bit outputs for each instance of this module are ORed, and then inverted to obtain the crash output of the VGA display.

Part Three: State Machine
Build a state machine to control the states of the game. My state machine has states SPAWN, STARTING, STARTED, and STOPPED. It also takes in a crash, start, and "twoSeconds" input (from the timer). The SPAWN state is for when the game starts and the car is not moving. The states changes to STARTING upon pressing btnC. The car then flashes for two seconds and the LEDs count down across the Basys3 board for two seconds. Then the game enters the STARTED state and remains there until the car goes off the road segments. In this state, the score counts up constantly and the LED lights flash an animation. The car crash is detected through a crash signal from the VGA display. Upon crashing, the game enters the STOPPED state and remains there until btnC is pressed. In this state, the car and score flash constantly, and the LEDs turn off. The state machine has the outputs started, blinkCar, resetScore, starting, resetTimer, loadRoad, countScore, blinkScore, and resetLED. Their names explain their functions.

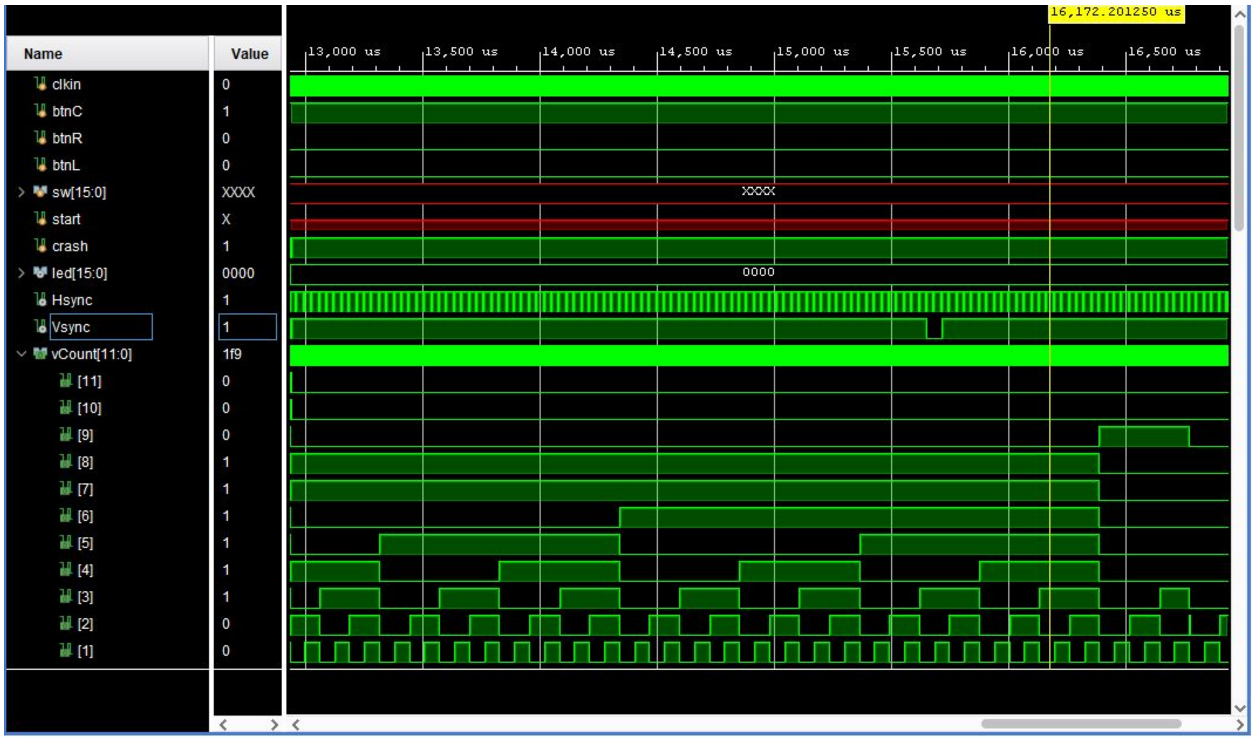
Part Four: LEDs
The LEDs are initially controlled through the state machine and a timer (counter). The timer is reset with certain state changes. For STARTING, the LEDs light up one by one and remain on until the timer is reset with the state. Upon this reset, a ring counter controls the LEDs so that they flash in a specific order.

Part Five: Score
The score uses a simple 16 bit counter that starts counting at the start of a game and resets when a game restarts. It is only displayed in certain states and flashes in the STOPPED state.

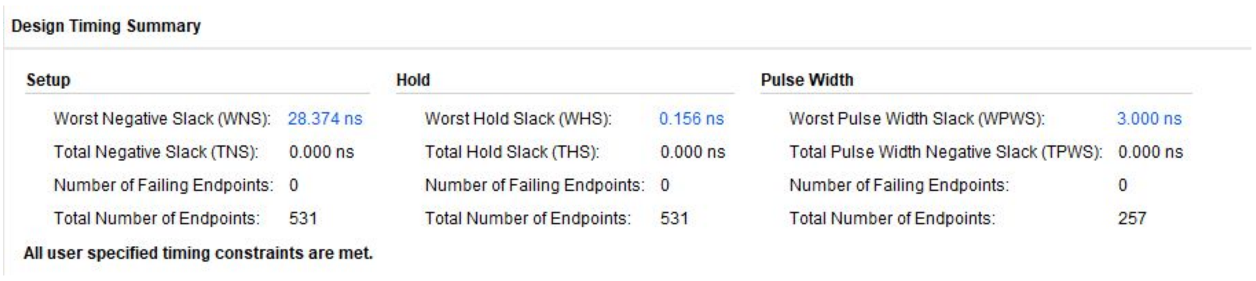
Part Six: Top Level
The top level contains 7 instances of the road segment module for each road segment. It also contains two counters for the score and timer, the VGA display module, the state machine, a ring counter for the LEDs, and the display components, including the ring counter, selector, and hex7seg modules.

<u>Results:</u>
I tested my design incrementally, testing each piece after I had completed or changed it. I started with the VGA module. Once I saw that this worked correctly, I went on to design a single segment. Once I was able to move a single segment successfully, I worked on moving 7. I spent a lot of time figuring out how to properly scroll a segment down the screen to the top, without it reappearing without a smooth transition. To fix this, I had to adjust the height for the segment when it was not fully on the screen. Additionally, figuring out how to detect when the car fell off of the road was very difficult, but I was able to do it after trying a few different methods. Everything else was relatively easy to build and test, when going piece by piece.

Remaining lab questions should have been answered in the rest of the report.



Name	Value
clkin	0
btnC	1
btnR	0
btnL	0
sw[15:0]	XXXX
start	X
crash	1
led[15:0]	0000
Hsync	1
Vsync	1
vCount[11:0]	1f9
[11]	0
[10]	0
[9]	0
[8]	1
[7]	1
[6]	1
[5]	1
[4]	1
[3]	1
[2]	0
[1]	0



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	28.374 ns	Worst Hold Slack (WHS):	0.156 ns	Worst Pulse Width Slack (WPWS):	3.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	531	Total Number of Endpoints:	531	Total Number of Endpoints:	257

All user specified timing constraints are met.

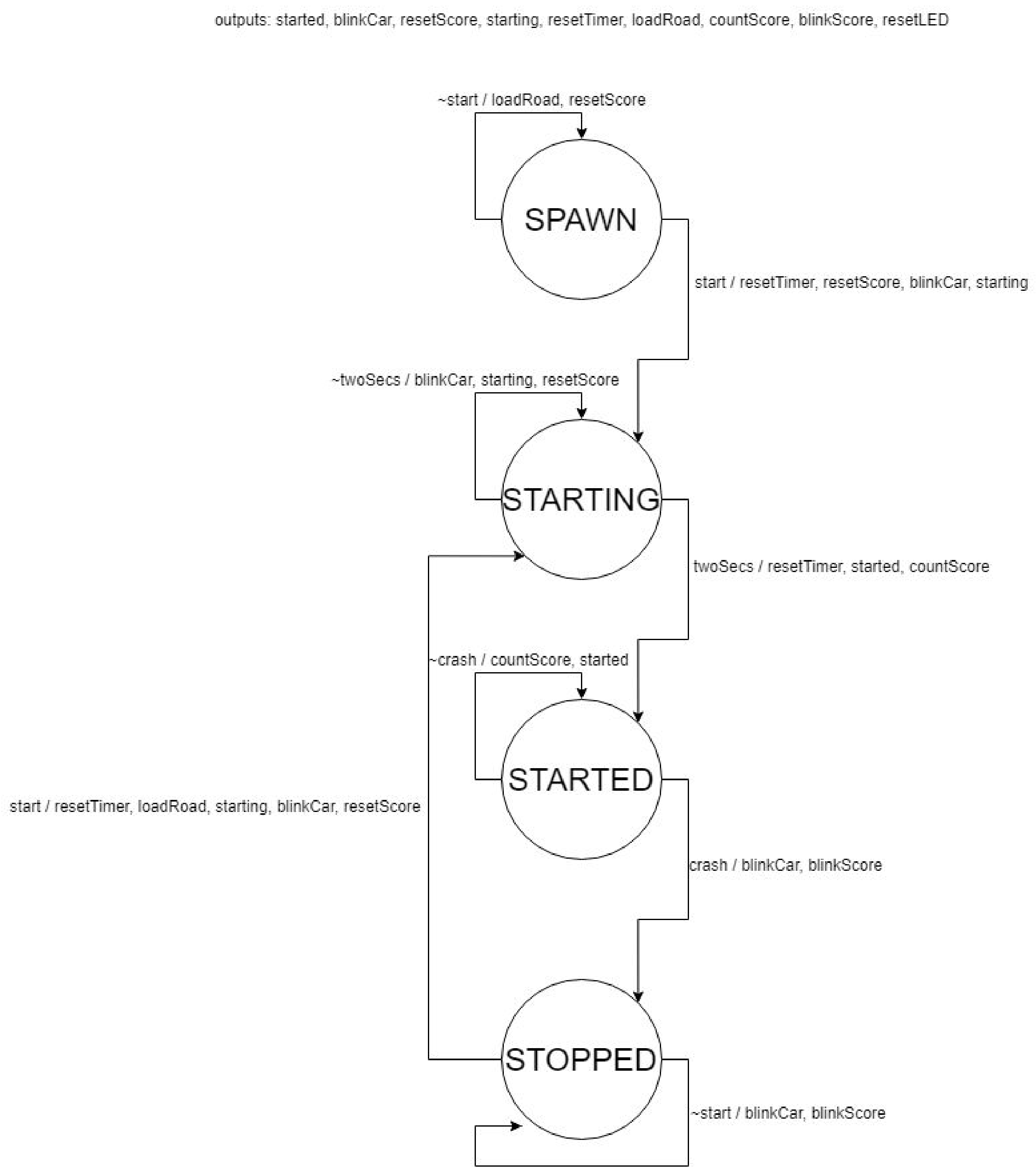

outputs: started, blinkCar, resetScore, starting, resetTimer, loadRoad, countScore, blinkScore, resetLED
~start / loadRoad, resetScore
SPAWN
start / resetTimer, resetScore, blinkCar, starting
~twoSecs / blinkCar, starting, resetScore
STARTING
twoSecs / resetTimer, started, countScore
~crash / countScore, started
STARTED
start / resetTimer, loadRoad, starting, blinkCar, resetScore
crash / blinkCar, blinkScore
STOPPED
~start / blinkCar, blinkScore

Figure 1: State Diagram

SPAWN = SPAWN & ~start
STARTING = ((SPAWN | STOPPED) & start) | (STARTING & ~twoSecs)
STARTED = (STARTING & twoSecs) | (STARTED & ~crash)

STOPPED = (STARTED & crash) | (STOPPED & ~start)
blinkCar = STOPPED | STARTING
blinkScore = STOPPED
loadRoad = (SPAWN & ~start) | (STOPPED & start)
resetTimer = ((SPAWN|STOPPED) & start) | (STARTING & twoSecs)
started = STARTED
starting = STARTING
resetScore = SPAWN | STARTING
countScore = STARTED
resetLED = STARTING & twoSecs

Conclusion:
In this lab, I learned to control the VGA output of the Basys3 board. I also learned how to properly design a game using many of the modules I had already made in the past 6 labs. I mentioned my main difficulties in the results portion. If I had to do this lab again, I would have a better introduction to the VGA display and how it works, as it took me a very long time just to figure that out.

Appendix
Supplementary material is appended at the end of this report.