Please attach this cover sheet to your lab report

ECE324 Lab 6: Tennis Circuit in FPGA

Name(s):

Alex Blake

Jameson Shaw

Exercise	Course outcome	Grade
Lab6 Demo	2.a, 2.d, 5.c, 7.b	/15
Lab6 Extra Credit	2.a, 2.d, 5.c, 7.b	/3
Lab6 Report	2.a, 5.c, 7.b	/25
	TOTAL:	/40

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.
2.d. Produce FPGA designs that meet specified needs.
5.c. Collaborate with individuals with diverse backgrounds, skills and perspectives.
7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

In this lab, students were asked to improve existing SystemVerilog code by changing the logic of a tennis emulation circuit from from asynchronous to synchronous. This means students need to know not only how the modules work, but how they are examples of asynchronous logic and how to change them to become synchronous. This lab's code was a Tennis game where the ball is displayed by the row of seven-segment displays. The code also utilizes BRAM as well, so this is also testing students their knowledge on initializing and utilizing BRAM.

Procedure

Before beginning the lab, acquire the files requires (multiple .sv files or modules and a constraint file for the FPGA board). This was given to us by the professor and should be ready to be implemented into a new SystemVerilog project.

The first step to the lab of three is simply to run the code after it is added to a Vivado profile. The code should be synthesized, implemented, and have a bitstream generated and uploaded to the FPGA board. If this is successful, it is time to move on to the next step.

Until this point, all the logic has remained asynchronous which is a major problem for reliability and stability, so that will now be changed. There are five major changes that are required in the tennis module (top file of the project) and listed below:

1. Minimize metastability by having 'swingLeft', 'swingRight', and 'nServe' created by separate 'free_run_shift_reg' modules.
2. Replace the existing flip-flop of nServe with a synchronous D flip-flop. Make sure 'nServe' = 1 when 'nL[1]' = 0, and 'nServe' = 0 when 'nToss' = 1. If neither of those occur, 'nServe' remains constant. Make sure to initialize 'nServe' to 1.
3. Declare a wire called 'rising_nSwing' and drive it with a 'risingEdgeDetector' (synchronous). The risingEdgeDetector is already in the existing code and just needs to be uncommented.
4. 's1' and 's2' need to be converted to synchronous D flip-flops.
   a. Change always block sensitivity to be synchronous
   b. Change flip flop clock signals to 100MHz signal
   c. Make sure 's1' and 's2' can only change on 'nSwing' rising edge (using 'rising_nSwing')
5. Change the flip-flop always statement towards the end of the module that drives 'nL' that uses 100MHz as the clock signal (synchronous) and only changes when 'moveBall' is high.

These are all the changes required to make this module a synchronous logic design and is now ready to be implemented and uploaded to the board. This should work as a game of tennis.

The final step to the lab required manipulation of the memory initialized and what drives which segments are on to display the game of tennis. The memory is initialized from a text document that came with the other files. The file is commented well so knowing what display what is obvious.

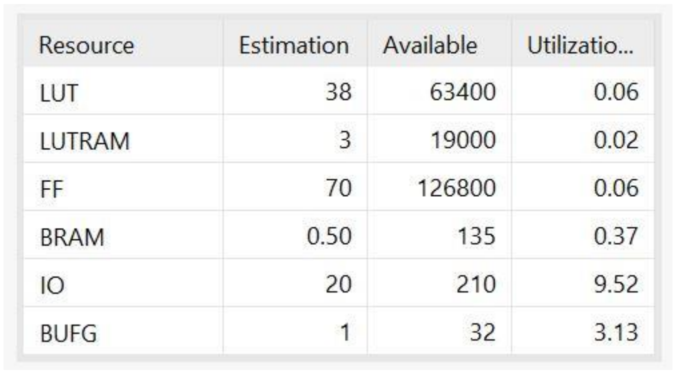
The final manipulation is to change the display at the serve; the ball should be hit overhand instead of underhand like the rest of the game. To do this, change the values within the text file so that the correct binary values which represent each segment is on, then convert it back to hex to be put in place of the existing number.

Test and confirm that the serve is overhand now. If everything works, this concludes the lab.

Results

The lab worked exactly as expected. Unlike previous labs, this lab compiled and was implemented without any issues. The code was commented well to make the process of changing everything from asynchronous to synchronous logic very simple.

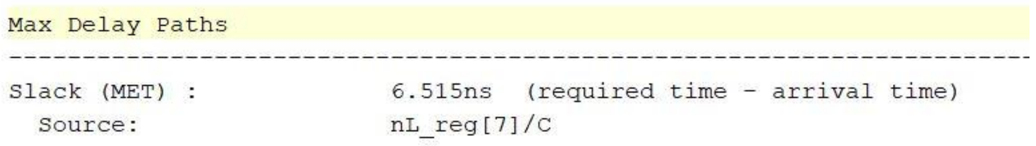
This lab also was the first lab that used BRAM so that was added in the chart (Figure 1) of what was used on the hardware to accomplish the synchronous logic project. Less than 1% of the BRAM was used in this project.



Resource	Estimation	Available	Utilizatio...
LUT	38	63400	0.06
LUTRAM	3	19000	0.02
FF	70	126800	0.06
BRAM	0.50	135	0.37
IO	20	210	9.52
BUFG	1	32	3.13

Figure 1: Utilization of FPGA (Including BRAM Used)

For the first time, this lab required the slack of the project to be used. This can be found in Figure 2.



```
Max Delay Paths
----------------------------------------------------------------------
Slack (MET) :             6.515ns  (required time - arrival time)
  Source:                 nL_reg[7]/C
```

Figure 2: Slack in Max Delay of Module

Figure 2 shows that there was a slack of 6.515 slack of delay. The following calculations show what was the fastest this could be implemented:

$$T_{MIN} = 10ns - 6.515ns = 3.485ns$$

$$F_{MAX} = \frac{1}{3.485ns} = 286.9MHz$$

In other words, this project could have been implemented to a faster clock from 100MHz to 286.9MHz and the delay between each stage of the logic would update fast enough. However, this is the limit and anything faster than this frequency would mean logic would not update fast enough and so the clock would tell it to capture before the correct outputs were updated. 100MHz gave us plenty of room for this project and we were not close to this maximum frequency.

Conclusion

This lab was good practice in understanding the difference between asynchronous and synchronous logic in a tennis module and how to convert them from one to another. It also introduced students to BRAM initialization and use on the FPGA board and how it can be useful. The techniques used are and will be useful and required for more complex projects.