**ECE324 Lab 6: Tennis Circuit in FPGA**

Name(s):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **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.

# Learning objectives:

1. Improve SystemVerilog HDL skills by making an asynchronous sequential design synchronous.
2. Improve FPGA design skills by using Xilinx Vivado software to implement a sequential logic function, including block RAM and look-up tables used as shift registers, and demonstrate it on an FPGA prototype board.
3. Learn how shift registers in LUTs, and BRAMs may be inferred.
4. Verify that a completed digital system implementation meets timing constraints.

# Part A Procedure:

Lab6\_Tennis.sv is an attempt using SystemVerilog to implement on a Nexys4DDR board the asynchronous logic in ECE324Lab6Schem.pdf (this was a lab implemented in TTL for WSU Vancouver’s ECE214 Digital Logic Design course). Add to Vivado all of the source files required for Lab6\_Tennis.sv (**TennisDisplayMem.txt doesn’t need to be added, but just needs to be in the same directory as your .xpr file**), and, without making any changes, run Synthesis, run Implementation, and try to Generate a Bitstream. In Messages, display warnings (but not infos nor statuses), and learn what the warnings and errors look like which are due to this design’s asynchronous logic (in my experience, every critical warning won’t give you what you want, so should be treated as an error). You do not need to demonstrate this to the professor.

# Part B Procedure:

Next, change the design in **ECE324Lab6Schem.pdf** to be totally synchronous to CLK100MHZ, and improve the display output, making the following changes in **Lab6\_Tennis.sv**:

1. Synchronize the input buttons, and minimize to an insignificant level the chance of metastability causing problems, by replacing the three assign statements for *swingLeft*, *swingRight*, and *toss*, with three instantiations of *free\_run\_shift\_reg* with 4 bits each.
2. Replace the asynchronous SR latch with a synchronous D flip-flop whose Q output is *nServe*. On the rising edge of *CLK100MHZ*, if *nL[1]* is 0, then *nServe* will go high; or else if *nToss* is 0, *nServe* will go low; or else *nServe* will stay the same. Also when declaring *nServe*, initialize it to 1.
3. Generate a signal which indicates when *nSwing* rises synchronously from one clock cycle of *CLK100MHZ* to the next. Do this by uncommenting the declaration of *rising\_nSwing* (in the *Declarations* section), and instantiating *risingEdgeDetector*.
4. Make the *s1* and *s0* flip-flops synchronous to everything else in the design:
   1. Convert the asynchronous sets of *s1* and *s0* to synchronous sets by modifying the “always” blocks’ sensitivity lists;
   2. Change the clock inputs to the *s1* and *s0* flip-flops into *CLK100MHZ*;
   3. Change the synchronous D inputs to the *s1* and *s0* flip-flops, such that when each of *s1* and *s0* is not being set, only allow a change in *s1* and *s0* when *nSwing* is rising (use what you generated in part 3 above).
5. Make the shift register synchronous to everything else in the design:
   1. Replace the clock with *CLK100MHZ*;
   2. Change the generation of *nL* such that it only changes when *moveBall* is high.

Implement your design into the FPGA. The game is played by first pressing the center button until the “ball” is shown in the right-most digit (simulating the toss of a ball during a serve), and then releasing it. The right button is then pressed, simulating the swing of the racquet for a serve (note as in real tennis, occasionally the service will fail, in which case you get an opportunity to serve a second time; the shorter you hold down the button, the less chance of getting a service fault). The ball will travel back and forth, and each player must swing at the ball when it’s next to their racquet. If the swing happens too late or too early, all LEDs will go off, indicating the end of the point. If you like, you can wait to demonstrate the game to the professor or T.A after part C.

# Part C Procedure:

The following documents the locations of the ball, racquets, and net, as implemented in the file TennisDisplayMem.txt. The data you are given is an exact replica for when nServe is 0 vs. 1.

s1 == 0 (shifting right) ballLoc s1 == 1 (shifting left)

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_|

**|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 0 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_**|** |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_|

|\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 1 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| **|**\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_**­**| |\_| |\_|

|\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 2 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_|

|\_**|** |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 3 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|**

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_|

**|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 4 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| **|**\_|

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_**|** |\_|

**|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 5 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_|

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| **|**\_|

**|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 6 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_|

\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

|\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_| |\_| |\_| |\_| |\_| |\_| |\_­| |\_| |\_|

**|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|** 7 **|**\_| |\_| |\_| |\_| **|**\_| |\_­| |\_| |\_**|**

In real tennis, the ball is tossed and hit overhead when serving. Change the single byte in TennisDisplayMem.txt such that, when nServe == 0, ballLoc==7, s1==1, and digit=7. the tennis ball is on the top segment instead of the middle segment, and the tennis racquet is in the upper right segment instead of the lower right segment. Demonstrate the game to the professor or T.A.

# Code reused

**free\_run\_bin\_counter.sv** – from lab5

**univ\_bin\_counter.sv** – from lab5 (may be used in extra credit)

# New Code provided

Download the following Verilog modules for your use in building your circuit:

**Lab6\_Tennis.sv** – Top level design, contains asynchronous logic.

**free\_run\_shift\_reg.sv** – Adapted from Chu's Listing 4.9

**rising\_edge\_detector.sv** – Adapted from Chu’s Listing 5.5

**TennisDisplayMem.txt** – Values to load into ROM containing 7-segment display values.

**Lab6\_Tennis.xdc** – Specifies the mapping of top-level FPGA Verilog module ports to physical device pins.

# Deliverables:

1. Demonstrate operation of your synchronous FPGA design to the instructor or lab assistant.
2. Submit a brief (1-2 pages) written lab report including the following items:
   1. Introduction describing the design goal(s) of this lab.
   2. A written description of the SystemVerilog code, and what you changed.
   3. In the bottom Vivado window, select Reports/Place Design, double click Utilization Report, and find and report the number of look-up tables that are used as shift registers, and the number of RAMB18 Block RAMs that are used. Where are each of these inferred in the Verilog code?
   4. In Xilinx Vivado, in *Reports/Route Design/Timing Summary Report*, under *Max Delay Paths*, find the *Slack*. Since the period specified was 10 ns (100 MHz clock), calculate the minimum period at which this logic could have run. From the minimum period, calculate and report the maximum clock frequency at which this logic could run.
   5. Conclusion discussing your results and what you learned from completing this lab.
3. Submit your original SystemVerilog code with your lab report (no testbench is necessary for this lab). Your SystemVerilog code ***must include header comments stating your names, date and class number.*** Your SystemVerilog code ***must also include comments explaining the operation of the code***.

# Optional Extra Credit (up to 3 points):

Make the ball gradually move faster, so eventually one of the two players will miss. Demonstrate your results to the professor or lab assistant.