

Lab 7: Racquetball



Introduction

This is another challenging design problem which incorporates knowledge of combinational logic, FPGA programming, flip-flops, counters, and multiplexers. The goal of this laboratory is to create a digital representation of a racquetball game. This lab will take a modular approach in the design, and an incremental approach to implementation. The overall description of the game and major block diagram are given in the next section. Each of the modules is further divided into submodules, and the implementation will be discussed in the following sections. This lab will give the students an opportunity to design and implement using several different techniques using SSI and MSI parts. The severe difficulty of this project will test the student's design, organization, team interaction, and implementation skills.

Description of the game:

The game consists of the following: two push buttons (that function as rackets) and a display of six LEDs in a row (that shows the position of the ball on the Court). There are two additional LEDs which indicate the player whose turn it is to hit the ball. The figure 1 illustrates what the game looks like.

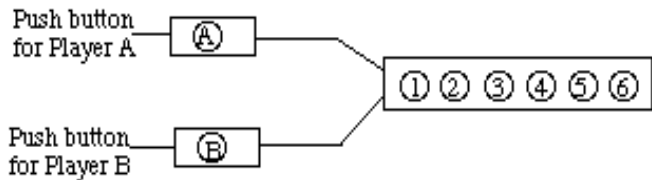


Figure 1: Overall racquetball layout. *

LED A (resp., LED B) is ON when it's player A's (resp., B's) turn to hit the ball. LEDs 1 through 6 give the position of the ball in the court. (Note that this is a very narrow court.) LED 6 corresponds to the wall at the end of the court. LED 1 is where the players are. If LEDs 1 through 6 are OFF then the ball is out of play, and so one of the players must serve. The player with its LED ON serves by pushing his/her button. (For example, if LED A is ON then it's Player A's turn to serve.) The ball commences down the court starting at LED 1. It bounces off the wall, and then returns back to LED 1. If the other player hits the button (i.e., "swings" his/her racket) when LED 1 is ON then the ball goes back down the court again (i.e., LED 2 goes ON and so forth). Otherwise, the player "misses" and the game is over (i.e, the ball is out of play). The player that wins the game gets to serve in the next game. The circuit is organized into three parts: Court Display, Racket Module, and Controller as shown in figure 2. These will be described in subsequent sections.

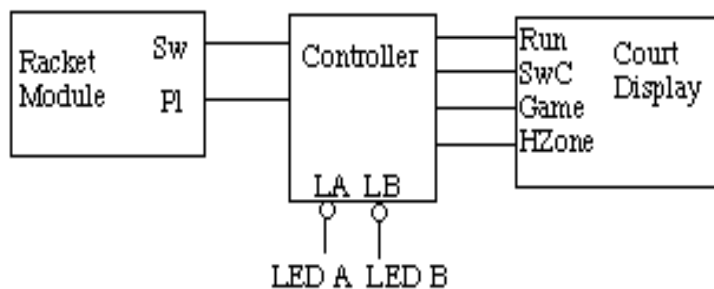


Figure 2: Racquetball block diagram.

You will design three racquetball games. The first is a simplified version of the above game with only a single player which will get the Court Display and Racket Modules implemented. The second is the full two player game described above, and is called Racquetball 1. For this game, the ball moves with each clock period, so that the ball speed is constant. The third, called Racquetball 2, is modified from the first. For this game, the ball has two speeds, depending on when a player swings his/her racket.

* Please excuse the interchange of the terms "Racketball" and "Racquetball". They both mean the same thing, but "Racquetball" is the correct spelling of the term. Since this is an adaptation of a UH Manoa laboratory, the two terms will be used sparingly throughout this manual.

New equipment information

- Time for you to start building the habit of knowing where to obtain IC information. The world wide web is a great resource to finding almost any IC data sheet. A suggested place to look is, <<http://www.futurlec.com/IC74LS00Series.shtml>>. Furthermore, when in doubt, Google it.

Design constraints

- Since this is an adaptation of a UH Manoa laboratory, a few components that are available to them are not available to us. Fortunately we have the Altera FPGA to substitute for any lack of components such as a 74138. There will be a limitation on the number of I/O channels used in this design project. Only the J1 and J2 expansion prototype connectors will be allowed for use in the design.
- Though in industry, implementing an entire design on a FPGA for testing is the standard, a racquetball implementation done purely with the Altera FPGA will receive a 40% grade deduction for this entire lab exercise. The full implementation of racquetball I and II must incorporate more than the equivalent of 5 completely-used physical ICs in order to avoid the 40% grade deduction.
- Concerning racquetball 0, the 5 completely-used physical ICs constraint does not apply, but it is advised to start incorporating physical ICs into the design in order to meet the constraint for the future.
- Regarding physical ICs, you are free to use any 74 series IC available. You are even allowed to bring in 74 series ICs that are not in your parts kit or available in the lab such as a 7474 (Quad D flip flops).
- Since we have more groups than dependable Altera FPGA boards, if your group wishes to use it, proof that the group is fully prepared to use the FPGA must be presented to the TA. If sufficient proof is given, a time limit will still be in effect. The effective limit for time with the FPGA will be 35 minutes if another group is waiting in line, and 1 hour-35 minutes if no other groups are in queue.
- Since the Altera UP3 and DE2 FPGAs have different I/O layouts, the amount of allowable I/O channels will be restricted to 32 regardless of which board is used. The push button and DIP switches do not count toward this constraint.

Racquetball 0 (Solitaire):

The first implementation of the racquetball game is a single player game for which you will build several of the major modules for the final game. The following are descriptions of the Racket Module and Court Display.

Description of Racket Module:

The racket module is shown below in figure 3.

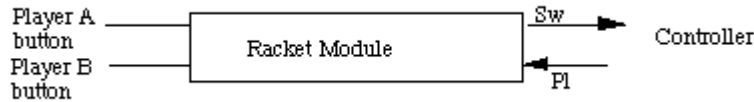


Figure 3: Racket module

The module has an output "Sw" (for "Swing") and an input "Pl" (for "Player"). The controller can choose the Player it wishes to observe by setting "Pl". If $Pl = 0$ then the controller chooses to observe Player A, and if $Pl = 1$ then the controller chooses to observe Player B. The output "Sw" equals 1 if the player chosen has just pushed his/her button (i.e, "swung" his/her racket). Otherwise, "Sw" equals 0. The Racket Module is actually composed of submodules as shown in figure 4.

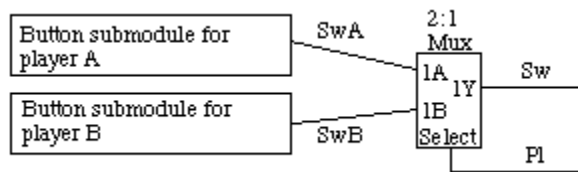


Figure 4: Racket submodules.

Note that a circuit called a 2:1 multiplexer (also written as 2-to-1 multiplexer, or 2:1 mux, or 2:1 selector) is used. A description of a generic 2:1 multiplexer is given in the textbook on pp. 605-606. It has three inputs (A, B, Select) and one output Y. (In the textbook, the inputs are (I0, I1, S) and the output is Z.) The inputs A and B are the "data" inputs, and the Select input is the "control." If $Select = 0$ then $Y = A$. On the other hand, if $Select = 1$ then $Y = B$. A multiplexer is also called a "selector" because it selects which input (A or B) is connected to the output Y.

The multiplexer is to be implemented using the 74157, which is actually a collection of four 2:1 multiplexers with a common Select input. The first multiplexer has inputs (1A,1B) and output 1Y; the second multiplexer has inputs (2A,2B) and output 2Y; the third multiplexer has inputs (3A,3B) and output 3Y; and the fourth multiplexer has inputs (4A, 4B) and output 4Y. See pp. 608-609 of the textbook for a description of the '157 under the subsection "Quad Two-Input MUX (74ALS157/HC157)".

The 74157 also has an additional "Strobe" input (or "enable G" input). If the Strobe = H then all outputs (1Y, 2Y, 3Y, 4Y) are L which basically deactivates the multiplexers. If the Strobe = L then the multiplexers are active and perform as follows. If $Select = L$ then $1Y = 1A$, $2Y = 2A$, $3Y = 3A$, and $4Y = 4A$. If $Select = H$ then $1Y = 1B$, $2Y = 2B$, $3Y = 3B$, and $4Y = 4B$.

The button submodules are connected to the button switches, which are single pole single throw (SPST) switches [1]. The submodule outputs are "SwA" and "SwB" for players A and B, respectively. They indicate when a player has "swung" his/her racket. A player "swings" by pushing the button. Note that if the player keeps the button down then that should not correspond to the player continuously "swinging." The output of a button submodule should have the following timing diagram shown in figure 5.

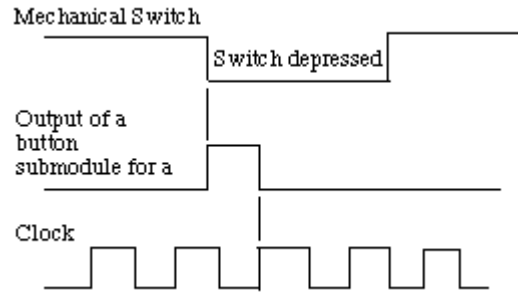


Figure 5: Timing diagram of racket swing submodule.

Note that as soon as the switch is depressed (pushed down), the output of the button submodule goes to "1," and then goes to zero after the next upward clock transition. In this way, after the racket is swung, the output is "0." The button submodule for Player A is organized as shown in figure 6 (the submodule for Player B is identical).

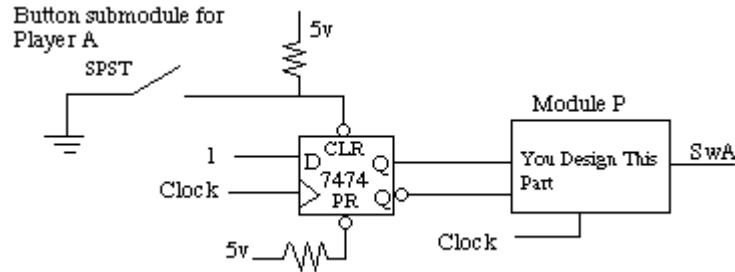


Figure 6: Racket button submodule diagram.

The D flip flop on the left (attached to the SPST switch) is a 7474 device. Unfortunately, as all of the students should know, we do not have 7474's available to us and adapt 7476's to the purpose of making D flip flops. The same also applies to this lab. The way the 7474 D flip flop is set up in figure 6, when the switch is pressed down, the "clear" input is asserted and the output Q immediately goes to 0. Note that Q will change back to 1 at the next clock transition, because the input D is permanently set to 1. To complete the design for the button submodule, the students must design Module P, which requires at most one D flip flop.

Task 1.

Draw the ASM chart for Module P. Note that its input is Q (from the D flip flop) and that its output is SwA. Module P should have two states: "Button was not pressed" and "Button was pressed."

Task 2.

Construct the Racket Module in Multisim. Note that it's composed of two button submodules and a 2:1 multiplexer. Use the 74157 device for the multiplexer. Get a print out of your working Multisim circuit.

Task 3.

Your group should build a working Racket Module circuit on a breadboard. Make sure to convert the circuit diagram into a schematic diagram by including the device numbers and pin numbers for each device.

Description of the Court Display.

Shown in figure is a block diagram of the Court Display (the clock input is implied, and not shown in the diagram).

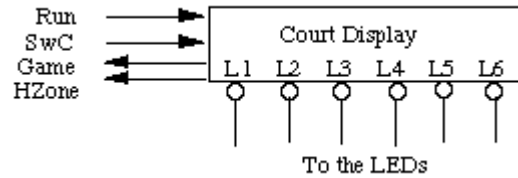


Figure 7: Block diagram of the Racquetball court display.

Note that there are six outputs (L1, L2,..., L6) that are connected to LED 1, LED 2,..., LED 6, respectively. Note that they have the negative logic convention. There are two more outputs: Game and HZone. Game = '1' if and only if the ball is in play (i.e., one of the LEDs is ON). HZone = '1' if and only if the ball is returning and is at LED 1. Thus, HZone = '1' means that a player can Hit the ball. There are two inputs: Run and SwC. SwC = '1' means that a player is swinging his/her racket. If the ball is out of play then SwC = '1' will mean that a serve is taking place. Run = '1' will advance the ball in the court display. If Run = '0' then the ball will hold its position. For Racquetball I, the input Run should be set to '1'. However, for Racquetball II, where ball speeds may change, we'll have Run equal '0' and '1' to control the speed.

The display should have the following states:

State	Interpretation
S0	Ball is at LED 1 and is proceeding in the direction of the Wall
S1	Ball is at LED 2 " " " "
S2	Ball is at LED 3 " " " "
S3	Ball is at LED 4 " " " "
S4	Ball is at LED 5 " " " "
S5	Ball is at LED 6 and bounces off the wall
S7	Ball is at LED 5 and is proceeding toward the players
S8	Ball is at LED 4 " " " "
S9	Ball is at LED 3 " " " "
S10	Ball is at LED 2 " " " "
S11	Ball is at LED 1 and is proceeding past the players and is about to go off the court
S12	Ball is out of play

Question 1:

In what state will HZone = 1?

Question 2:

In what state will Game = 0?

Question 3:

Suppose the circuit is in state S12 (ball not in play). What do the inputs have to be before the circuit transitions to S0 (which corresponds to "game commences")?

Question 4:

Suppose the circuit is in state S11 (i.e., ball is at LED 1 and is proceeding past players). For what inputs will the circuit hold its state? For what inputs will the circuit transition to S12 (ball out of play)? For what inputs will the circuit transition to S1 (ball proceeds down the court)?

Task 4:

Draw the ASM chart for the Court Display circuit.

To design the Court Display, you will use the following parts: 74193 (Binary Up/Down Counter with Clear), the Altera FPGA board, and the 74138 (3:8 decoder/demultiplexer, !!unfortunately we do not have this part, so it will be implemented through the FPGA!!). The parts are connected as shown in figure 8.

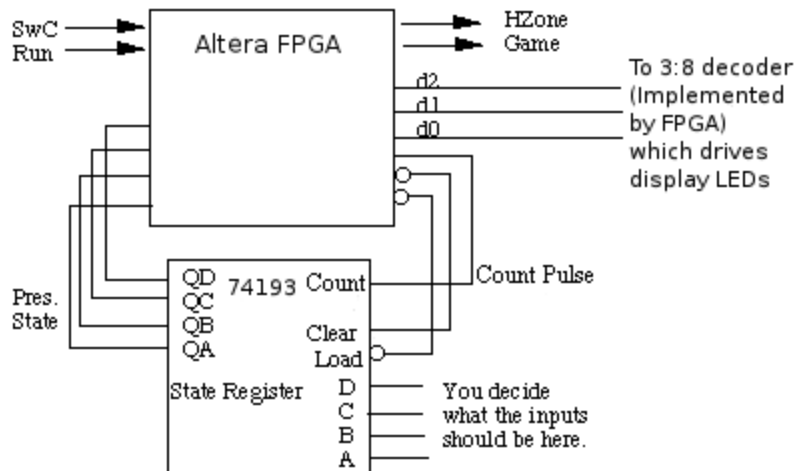


Figure 8: Court display module diagram.

The following is a brief description of the 74193 4-bit binary up/down counter and a generic 3-to-8 decoder. The fact that its count depends on a pulse input means that its state changes only at clock transitions. If its Clear input is asserted then its count (QD, QC, QB, QA) is cleared. If the CP_U pin is

used, then the count is incremented by one (it counts up). Conversely, if the CP_D pin is used, then the count is decremented by one (it counts down). Please refer to the datasheet for the 74193 in order to obtain further information about pins CP_U and CP_D which is not displayed in figure 8. If the Load is asserted then the value at (D,C,B,A) is loaded into the counter. The 74193 is used as the state register. (See pg. 403-410 of the textbook for a description of the '193.)

A generic 3-to-8 decoder (or 3-to-8 demultiplexer or demux, or sometimes written as 3:8 demultiplexer or decoder, etc) has a single "data" input G1 (which is sometimes called the "enable" input), three select inputs (C,B,A), and eight outputs (Y0, Y1,..., Y7). Figure 9 illustrates this concept.

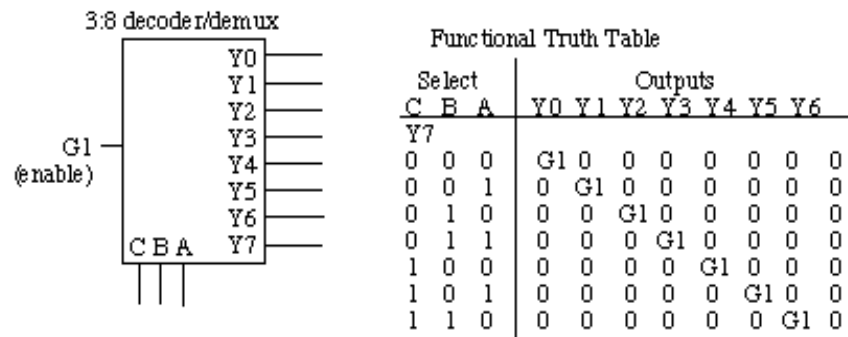


Figure 9: Generic depiction of a 3:8 decoder/demultiplexer.

The select input value is a 3-bit number that determines which output that G1 will be connected to. For example, suppose $(C,B,A) = (0,1,1)$. Since 011 corresponds to the value 3, the output Y3 is connected to G1. As another example, suppose $(C,B,A) = (1,0,1)$. Then $Y5 = G1$. If an output is not selected then its value is 0. The following is the truth table of a 3-to-8 decoder.

<u>Inputs</u>				<u>Outputs</u>							
Enable	Select										
G1	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

(See pp. 584-586 of the textbook for another description of a decoder/demultiplexer.)

The 3-to-8 decoder will be used to drive the six LEDs (i.e., LED 1, LED 2,... LED 6). It has the inputs A, B, C and the outputs Y0, Y1, ..., Y7. For the Court Display circuit, let d2, d1, d0 (from the Altera FPGA) be connected to C, B, A of the 3:8 decoder (also implemented in the Altera FPGA). The outputs Y1, Y2, ..., Y6 of the decoder correspond to L1, L2,..., L6, respectively.

Question 5:

If the circuit is in state S0, what should the value of (d2,d1,d0) be?

Question 6:

If the circuit is in state S10, what should the value of (d2,d1,d0) be?

Question 7:

If the circuit is in state S12, what should the value of (d2,d1,d0) be?

Task 5:

Design the circuit in Multisim. Use the following encoding for the states: S0, S1, S2.... is represented by "0000", "0001", "0010",..., respectively. Get a print out of your working circuit. Attach the VHDL code that is used to program the Altera FPGA.

(Hint: Suppose the circuit is in state S11, and Run = '1'. If SwC = '0' then the player has missed the ball and the next state should be S12. On the other hand, if SwC = '1' then the player hits the ball and the next state should be S1, i.e., the ball is at LED 2 and proceeds down court. To get to S12, have the 3:8 decoder continue counting. To get to S1, have the 3:8 decoder load the encoding for S1, which is "0001".)

Task 6:

With your group, build the Court Display on the protoboard.

Task 7:

Now that you have built these modules, it would be nice to see if they work. On your protoboard, connect the modules together as shown below to get a single player game (similar to paddleball). This helps to determine if your modules are inter-operable. Demonstrate to the TA that your circuit works.

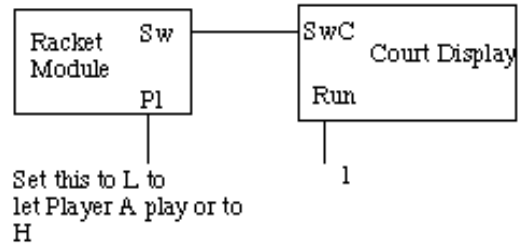


Figure 10: First milestone design.

Racquetball I (The Sequel):

You are to design, build, and demonstrate (to the TA) Racquetball 1. This game has only one ball speed, so the Run input of the Court Display should be set to '1'. The following block diagram shown in figure 11 depicts the complete Racquetball 1 circuit.

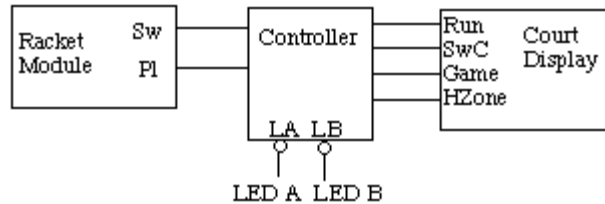


Figure 11: Complete racquetball 1 layout.

Task 8:

Draw the ASM chart for the controller module. (Hint: Let the controller have two states, keeping track of who's turn it is to hit or serve.)

You are to design the circuit using D flip flops, NAND, NOR, and NOT gates. You should minimize your circuit, since it should make your circuit simpler. Simpler circuits are easier to debug.

Task 9:

Construct the Multisim and VHDL files for the Racket Module and Court Display. Store them for later use as components. Now build the controller in Multisim and VHDL. Connect the controller to the modules to get a complete Racquetball 1 circuit. Get a print out of the circuit diagram. Include your design files into the weekly progress/lab report.

Task 10:

Build Racquetball 1 on your protoboard. Demonstrate it to the TA when it is working.

Racquetball II (The Final Showdown):

Racquetball 1 was okay, but does not compare with the available hand held toys on the market. So we would like to make the game a little more interesting by having two ball speeds: "slow" and "fast". The "slow" speed means that the ball advances every two clock periods, while the "fast" speed means that the ball advances every clock period (just like in Racketball 1). Note that you can control the rate at which the ball advances by using the Run input.

When the ball is served, it goes at the "slow" speed. When the ball returns from the wall and is at LED 1 (i.e., the state of the Court Display is S11), it will stay at LED 1 for two clock periods. If the player swings during the first clock period then the ball is returned at the "fast" speed. If the player swings during the second clock period then the ball is returned at the "slow" speed. Once the ball goes "fast", it stays fast until the game is over.

Task 11.

Draw another ASM chart for the controller to realize Racquetball 2.

Task 12.

Build Racquetball 2 in Multisim and VHDL. Use the Racket Module and Court Display modules that you have pre-built as components. Get a print out of your circuit. If you use an FPGA for your controller, attach the VHDL code you used to program it with a screen shot of your FPGA pin assignments.

Task 13.

Build Racketball 2 on your protoboard. Demonstrate it to the TA when it works.

Deliverables:

- Do the tasks and answer the questions in each section. Each lab group should turn in the
 - Answers to all questions
 - Circuit diagrams for the Racket Module, Court Display, Racquetball 1, and Racquetball 2
 - Working simulations of Racquetball 1 and Racquetball 2
- You must also demonstrate to the TA, protoboard versions of the single player game, Racquetball 1, and Racquetball 2.

To be able to complete this entire project in the time allotted, here are a set of milestones to strive for. There is an extra week in the schedule to allow for schedule slips, but do not let the schedule slip too far, or you will not be able to finish.

Week 1: Racquetball 0. Design and build the Racket Module and the Court Display (these may take a little longer than a week).

Week 2: Design and build Racquetball 1.

Week 3: Design and build Racquetball 2.

Week 4: Complete any unfinished tasks, questions, and work on the final report.

- As a safety measure to keep each group on track with the milestones, a group report will be due for each week of the lab. The submission scheme will remain the same where reports will be due on Mondays, but some leniency will be allowed in the form of extra credit. If the weekly report is submitted on Monday, then a +7% extra credit will be assigned to the overall lab grade for the entire group. Otherwise, the weekly report can be submitted by the following week's lab meeting day for a regular scoring scheme. Weekly reports submitted later than the following week's lab meeting day will be counted as a 0% for that certain report. There will be two chances for this extra credit opportunity.
- For the final report relating to racquetball 2, the extra credit opportunity will not be available. The final group lab report is due on May 1st, 2006.
- As stated in the homework policy, lab reports must be turned in at the beginning of the class on the designated due date unless the lab report is being submitted electronically. If submitted electronically, the submission time must occur before midnight on the designated due date and must be submitted in .odt format. Failure to comply will lead to an automatic 0% for the respective report.
- Like any team design project, an evaluation at the end will be required. Each individual team member must submit a evaluation report regarding the performance on this project. Things to include into this evaluation report are:
 - Written description of each teammates' strengths and weaknesses in the design project
 - Numerical ratings for their teammate's performance(scale of 0 to 10 with 0 being “Poor” and 10 being “Excellent”)
 - Written description of your own performance on this design project
 - Numerical rating of your own performance
 - Conclusions concerning the overall lab

This evaluation report will have the constraints of:

- 2 page minimum
- Double space
- 0.75” page margins
- 12 point font
- Times New Roman
- Will be due the next class meeting date AFTER the submission of the final-group lab report (May 3rd, 2006).
- If submitted electronically, must be done in .odt format
- Failure to comply with these constraints will lead to an automatic 0% for the entire lab

These evaluation reports will be kept confidential and will not be revealed to any third parties under any circumstances.

Miscellaneous notes

- This design project will count as four lab grades.
- For god's sake, start early.
- Budget your time and distribute the workload across the team in a manner to work efficiently.
- Allow time for redesign and debugging.
- Make use of the open labs being offered!! Learn to meet outside of “class time”. If you think all the work can be done only on the lab meeting days, you will not be able to finish this design project.
- Because of the demonstration-time limit constraint due to the availability of the Altera board, plan your time with the board efficiently.
 - Being unsure of how to program the board can lead to precious time being wasted. A demonstration video of how to program the FPGA is posted on the class website.
 - Planning out your FPGA pinouts in advance can also save some time.
- This project is fricken huge and contains many submodules. Try to break up the workload and have people designing, implementing and testing different submodules in parallel.
- In the fabrication phase, build and test sub-circuits individually. Assembling the whole design together without testing smaller sub-portions can lead to a lot more trouble than its worth. -- *Quoted from the U.S. Navy S.E.A.L.S. doctrine: “Slow is fast, and fast is slow”*
 - This is a good area to apply divide-and-conqueror
 - Make each person in the team responsible for fabricating and ensuring a sub-circuit works correctly.
- When assembling the whole circuit it is usually better to let one person handle the wiring, and the other teammates watch carefully each connection. Having people alternate in wiring the whole circuit can lead to confusion between personal wiring conventions.
- Take the time to ensure the planning of your design is solid. If the plan is good, the fabrication and debugging process will be quick. A poor plan will result in an insane amount of time dedicated to fabrication and debugging.
- Cutting corners in the design approach can lead to faults that are very hard to find. Take your time, but also work diligently and efficiently.
- To make the board remember a program, use the “Active Serial” mode instead of “JTAG” mode when your on the “Programmer” window. Also, make sure the programming cable is connected to the AS mode port and not the JTAG port on the Altera board.

References

- [1] Dobry, Tep; “*Introduction to Digital Design*”; Univ. of Hawaii at Manoa; accessed on Feb. 20, 2006; <<http://www-ee.eng.hawaii.edu/~tep/EE260/F04/>>.
- [2] Futurelec; “74LS Series Integrated Circuit Components”; accessed on Feb. 24, 2006; <<http://www.futurelec.com/74LS/74LS76.shtml>>.