CSE 370 Introductory Laboratory Assignment

Finite State Machines and Design

Assigned: Monday, November 10, 2008
Due: End of Lab Section

Objectives

Welcome to lab 7, in this lab you will build a tug of war game based on specifications we give you. Follow the steps below to begin:

Follows These Steps Exactly

1. Open Active-HDL.
2. When prompted for a workspace, CREATE A NEW WORKSPACE.
3. Name your new workspace: lab7, set the location for the workspace folder to be on your Z: drive, UNCHECK: Add New Design to Workspace
4. Download the following file to your desktop: tugofwar.zip
5. Select (Design), (Restore Design)
6. Set the [Archive Path] to tugofwar.zip you downloaded
7. Set the [Restore To] to the folder you set your workspace on the Z: drive
8. Click [Start]
9. Right click (Workspace) on the left side of your screen and select (Add Existing Design to Workspace)
10. Find the folder you set your workspace on the Z: drive, open up the new folder (tugofwar) and restore the file (tugofwar.adf)

The design should be restored, you should see a design called (tugofwar) with the following files: top_level.bde, tugofwar.v, hex_decoder.v, pulse_gen.v, score.v. You will have to write the verilog code for tugofwar.v, pulse_gen.v, and score.v. Write your code based on the specifications below:

Tug of War Game

pulse_gen.v:

1. Pulse Gen needs to generate a HIGH (1) pulse for 1 clock cycle when it detects that the user has pressed the button.

   Tip 1: Remember not to generate more than one HIGH (1) pulse for each time the user pushes the button.
   Tip 2: You can detect when the user has pressed a button by using a state machine described in

the next tip. You need to check for when the button goes from HIGH (1) to LOW (0). This is because the buttons are active LOW (0) Active LOW means that when the button is pressed the signal goes from HIGH to LOW.
Tip 3: The suggested design for this is a state machine with three states. The first state should be the wait state. You sit here and wait for the user to push the button. In the last state you should sit and wait for the user to release the button before going back to the first state. This is because many clock cycles can occur while the user has the button held down and you don't want to generate multiple cycles.

score.v:

2. Score needs to keep track of the score. Each time the input: Scored, goes HIGH (1) the player's score should be increased by 1. Score needs to signal when a player has won, when a player reaches FIVE points, the signal: Win, goes HIGH (1). Score should respond to RESET, when RESET goes HIGH(1), the score should be reset to 0, and Win should be reset to LOW(0).

   Tip: You can use a register to keep track of score, try using a non blocking statement like x <= x + 1.
   Tip2: Use a comparison like an if statement to check when a player gets 5 points.

tugofwar.v:

3. TugofWar needs to keep track of the position of the LED, and report to Score when a player has won a round. A game of tug of war goes as follows:

   1. Someone resets the game, the only lit LED should be the one in the middle. For example if 0 means the LED is off and 1 means the LED is on, the value would be: 0000000100000000

   2. Player 1 and Player 2 press their respective buttons, each time Player 1 presses their button the lit LED should move right. For example: 0000000100000000 changes to 0000000010000000. If Player 2 presses their button the lit LED should move left. For example: 0000000100000000 changes to 0000001000000000. If both players press their buttons at the same time, the lit LED should not move.

   3. The round continues until one player manages to get the lit LED to their side. For example : 0000000000000001 means Player 1 scores for the round, otherwise 1000000000000000 means Player 2 scores for the round.

   4. When a player scores a round, their score should go up by 1, and the lit LED is automatically returned to the start, meaning it should look like this: 0000000100000000 and the two players should keep playing.

   5. This repeats until a player wins the game by scoring 5 rounds. When a player has won, the LEDs should light up signifying a winner. If Player 1 wins, all the LEDs on Player 1's side should light up: 0000000011111111, and if Player 2 wins all the LEDs on Player 2's side should light up: 1111111000000000. The buttons should no longer move the lit LEDs until the game is reset.

   Write the verilog code in tugofwar.v so that it behaves as described above.

Tip: You should use a 17 bit register to keep track of the lit LED. You can use the signal that Pulse_Gen passes you to check when a player has pressed their button.
Tip 2: You should use the shift feature to move the LED. For example >> 1 shifts everything to the right by 1, and <<1 shifts everything to the left by one. Use this to move the lit LED back and forth for the players.
Tip 3: You can check if a player has won by using the signal that Score passes you. It goes high when a player has won the game.

Lab Demonstration/Turn-In Requirements

A TA needs to "Check You Off" for each of the tasks listed below.

1. Have a TA check off your working Tug of War game.

Comments to: cse370- webmaster@cs.washington.edu