Signoffs and Grade:

Name:_____

Component	Signoff	Date	Time
Servo moves from Min = 46 and Max = 134			
Servo moves from Min = 80 and Max = 100			
Servo stays at Min = 90 and Max = 90			

==========

Component	Received	Possible
Prelab		10
Signoffs		90
Penalties • after the first 15 minutes of lab session 6: -10 • after the first 15 minutes of lab session 7: -25 no signoffs after the first 15 minutes of lab session 8:	-	
Total		100

Education Objective
The educational objective of this laboratory is to investigate the creation of custom IP to interface with the NIOS II on the Avalon Switch Fabric. This lab is two parts, combined with Lab 4. Lab 4 focuses on the creation of the core, and Lab 5 on the software interfacing with the component.

Technical Objective
The technical objective of this laboratory is to design an embedded system for the Nios II processor and DE1-SoC development board that will control a servo motor to sweep between two angles.

The embedded system should contain a custom IP that creates the appropriate Pulse-Width Modulation (PWM) signal to sweep the servo-motor's arm. The system also contains 8 switches for entering the start and stop angles, 2 pushbuttons for "locking" in the angles and 5 seven-segment displays for displaying the start and stop angles.

The system operates as follows:

- The user inputs the start or minimum angle (> 45) in binary using the switches and locks it in with KEY3. You can assume for this lab you will not be asked to enter a minimum angle larger than 99.
- The user inputs the stop or maximum angle (<135) in binary using the switches and locks it in with KEY2.
- The C program passes the angle information to the servo controller.
- The servo controller outputs a waveform to the servo motor that causes it to continuously sweep from the minimum angle to the maximum angle and back again.
- The angles can be changed at any time and in any order.

Software Design
The C code for this embedded system if very straight forward. Both the pushbutton and servo controller interrupts need to be registered in the C code using the alt_ic_isr_register() function from Lab 3. The pushbutton interrupts need to be enabled in Platform Designer as well. There should also be an initial write to the servo controller setting its min angle to 45 degrees and its max angle to 135 degrees in main before you begin you while(1) looping.

There will be two interrupt service routines (ISR). The ISR for the pushbuttons should change the variable for the minimum or maximum angle depending on which pushbutton was pushed.

The ISR for the servo controller writes to both of the servo controller's registers. Remember you will be loading the registers with the number of counts necessary to create the pulse width for the specified angle.

The main loop of the program should display the configured minimum angle (in decimal) on HEX5 and HEX4 and display the maximum angle (in decimal) on HEX2-HEX0. If the angles are less than the number of digits allotted, just blank the display or front fill with 0. The displays should be updated every time the min and/or max angle changes. Remember that ISRs need to be executed as quickly as possible. As such, do not update the displays in the ISR, but rather set a flag and perform the task once the program has returned to main.

Prelab

Part 1 – Add custom component to new Lab 5 project.

1. Open your Quartus II project from Lab 4. Confirm your servo_controller.vhd file is in the project.
2. Open Platform Designer and create a system with the following components
   a. Nios II/e processor
   b. On-chip memory for program code and data
   c. 8-bit input PIO for switches
   d. 4-bit input PIO for pushbuttons enabled for a falling edge interrupt
   e. 5 7-bit output PIO for HEX5, HEX4, HEX2, HEX1, HEX0
   f. JTAG Uart
   g. Sysid
3. Using the custom component demo as a guide, create a custom component for the servo controller and add it to the nios_system.
4. Verify that the jtag_uart, the pushbutton PIO, and the servo controller are all connected to the NIOS II IRQ port.
5. Save your system as nios_system.qsys
6. Generate the VHDL. *NOTE* if you need to go back and edit the servo_controller VHDL code, you need to regenerate the VHDL in QSYS.
7. Return to the Quartus project and add nios_system.qip to the project.
8. Create the top_level VHDL file. The <project>/nios_system/nios_sytem_inst.vhd file contains the component declaration and the port map template to instantiate nios_system in the top_level.
   a. Remember, you will want to name your top level entity ports the same names as the provided *.qsf pin assignement file, otherwise the pins will not assign correctly.
9. Use Assignments > Import Assignments… to import the pin assignments in the DE1_SoC.qsf file
10. Compile the design.
11. Connect a servo motor's signal pin to use the same GPIO pin as in Lab 4.
12. Program the DE1_SoC board.

Procedure

Part 1 – Write the necessary C program code and demonstrate sweeping servo at different angles.

1. Write the NIOS II C program described in the software design section above.
2. Open NIOS II Software Build Tools for Eclipse. Create a new App and BSP, generate the BSP, move your C program to the app folder, build the project and choose Debug as NIOS II hardware.
3. Click the run icon and verify your program works as expected. The servo motor should start by sweeping from 45 to 135 degrees. If that seems to be working correctly, try changing the angles.
4. If your system is not working as expected, you will need to debug the software and possible look deeper in the hardware using Signal Tap methods learned from the last demo.
5. **Obtain signoffs for different angles on signoff sheet.**