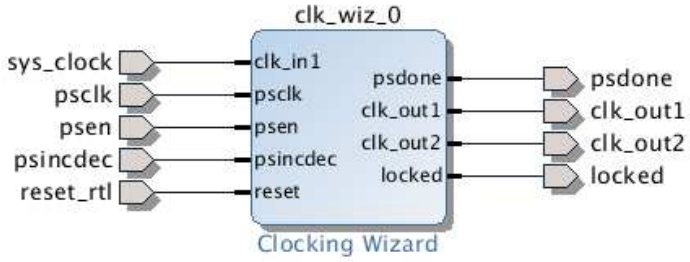
LAB Assignment #3 for ECE 522

Assigned: Mon., Sept 28, 2015
Part I: Due: Wed., Sept. 30, 2015
Part II: Due: Mon, Oct. 5, 2015

Description: Create a project with an MMCM, simulate the fine phase shift feature (part I), add a state machine that encapsulates the MMCM and provides a well defined interface

Part I:

1) Vivado component: Create a new project, 'Add IP' Clocking Wizard, double click to config, click 'Clock Options tab', select Dynamic Phase Shift, click Output Clocks, add check mark on left for 'clk_out2' and add check mark on far right to 'Use Fine PS' (ONLY FOR clk_out2), set output freq for BOTH clk_out1 and clk_out2 to 50.000, click Summary, just make note of the 'Divide Counter', 'Mult Counter' and 'CLKOUTx Divider' values. Click Okay. Run automation and use defaults, Right click each of the remaining pins, choose 'Create Port' and use defaults. Right click 'design_1 (design_1.bd)' in Sources window, choose 'Create HDL wrapper', 'Copy generated wrapper to allow user edits' and then Okay. Right click 'Generate Output Products' and use defaults, click Okay. Run 'Run Synthesis' and 'Run Implementation'.



3) Testbench: Instantiate the entity in the 'design_1_wrapper - STRUCTURE (design_1_wrapper.vhd) in a **verilog test bench** (see lab3_testbench_starter.v). We must use verilog b/c Vivado does not support VHDL for timing simulation testbenches. Simulate the fine phase adjust feature using 'isim' or the student version of 'modelsim' (available for download). You MUST run 'Run Simulation', 'Run Post-Implementation Timing Simulation'.

The following is derived from www.xilinx.com/support/.../user.../ug472_7Series_Clocking.pdf, starting on page 70, but most important material on page 73. The F_{VCO} frequency needed below is computed as:

$$F_{\text{VCO}} = F_{\text{CLKIN}}\frac{M}{D} = 100\text{MHz}\frac{10}{1} = 1\text{GHz}$$

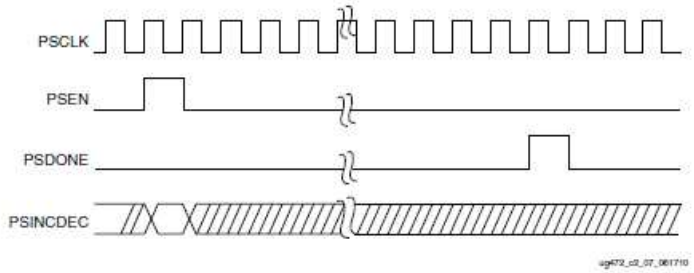
(see MMCM_lab3/MMCM_lab3.srcs/sources_1/bd/design_1/ip/design_1_clk_wiz_0_0/design_1_clk_wiz_0_0_clk_wiz.vhd for constants) -- use the name of your lab for MMCM_lab3.

Fine phase can be adjusted in increments of:

$$\text{phase adjust resolution} = \frac{1}{56F_{VCO}} = \frac{1}{56(1\text{GHz})} = 17.9\text{ ps}$$

Your testbench should control the MMCM in the following manner (from page 73 above):

- The variable phase shift is controlled by the PSEN, PSINCDEC, PSCLK, and PSDONE ports. The following timing diagram illustrates the I/O behavior of these signals.



- Note that PSCLK, PSEN and PSINCDEC are signals your testbench needs to control. PSCLK should be connected to your testbench clk signal that you need to generate with an 'always' block at 50 MHz (see starter code), i.e., do NOT use 'clk_out1' or 'clk_out2' as the source of clk in your test bench -- you will be 'monitoring' these clocks in this lab. Similarly, you will use an always block to generate the 'sys_clock' signal at 100 MHz that is synchronized to the 50 MHz PSCLK.

- Pulsing PSEN starts a phase shift operation, which is carried out within the MMCM. PSINCDEC controls the direction of the phase shift, with '1' causing the phase to increase by 17.9 ps and '0' causing it to decrease by 17.9 ps. With a 50 MHz

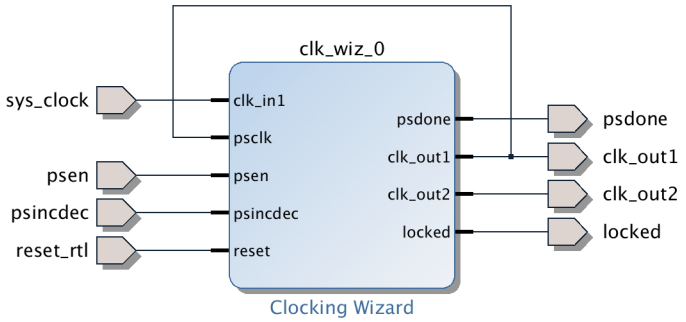
NOTE: You MUST wait for 100 ns (#100) in the test bench before changing any signals. Xilinx requires this time to reset the hardware. You should pulse 'reset_rtl' for one 'PSCLK' period after the 100 ns wait and then wait for the 'locked' signal to assert before setting PSINCDEC to '0' or '1' and pulsing PSEN (for 20 ns) to demonstrate the phase shift on the clk_out2 output.

You need to print out your waveform window showing that portion of the simulation that shows the fine phase shift feature working (zoom in as needed since the shifts are small, e.g., 18 ps). Include plots of both 'clk_out1' and 'clk_out2'.

Part II:

1) Create a project with the following components:

- Modify the block diagram from Part I as follows and re-generate the wrapper:



- Modify the generated with the vhd code on my website labeled "design_1_wrapper.vhd"
- Add the following VHDL files (also on my website) to your project: LCTest_Driver.vhd, PhaseAdjust_starter_code.vhd and Top.vhd

2) Add the VHDL state machine to the PhaseAdjust.vhd file that implements the following functionality:

- Has 3 states, ‘idle’, ‘adjust_phase’ and ‘adjust_done’
- Maintains an internal register called ‘current_phase’ that tracks the fine phase adjust currently programming into the MMCM (assume it is 0 after doing a reset)
- Remains in idle until the ‘LC_start’ is asserted (see starter code for PhaseAdjust.vhd)
- De-asserts ‘ready’, computes a ‘delta_phase’, which is the signed difference between the ‘current_phase’ and the ‘target_phase’
- Adjusts the fine phase of the MMCM until the ‘target_phase’ is reached
- Once ‘current_phase’ becomes equal to ‘target_phase’, return to idle and re-assert ‘ready’

3) Synthesize and Implement the design (do **NOT** generate a bitstream)

****

DO NOT PROGRAM THE FPGA WITH THIS PROJECT. YOU HAVE NOT SETUP CONSTRAINTS for the entity port signals in ‘design_1_wrapper’ and therefore, Vivado will map them to arbitrary pins.

****

4) Simulate the design using Post-Implementation Timing Simulation by instantiating the design_1_wrapper.vhd in your test bench. Set a ‘target_phase’ in the testbench, assert the ‘user_start’ signal and show wavefroms that illustrate the automatic adjustment of the ‘clk_out2’ phase. Note that you need to drive ‘sys_clock’ at 100 MHz, as you did in Part I.

Laboratory Report Requirements:

Grading:
The grading from this lab will be based entirely on your in-class demo. Bonus points will be given to any implementation feature that goes above and beyond the requirements.