## LAB Assignment #4 for ECE 522

Assigned: Mon., Oct 12, 2015
Part I: Due: Wed., Oct. 14, 2015
Part II: Due: Mon, Oct. 19, 2015

## Description: Finish the LCTest_Driver.vhd state machine from the starter code. Simulate it.

**Part I:**

1) Important Notes: Many of the problems most of you have been having are related to the recent discovery that Vivado does not support timing simulations of designs described in VHDL (only verilog designs are supported). It is astonishing that Xilinx had made that decision given the size of the user base that codes in VHDL. I spent the entire fall break + weekend trying to find workarounds for this problem and only came up with two:

- Use Vivado 2014.4, which allows you to configure the simulation tool to use other 3rd party simulators. Since we have Cadence, I used IES (Incisive Enterprise Simulation) and everything worked fine. A student version of ModelSim (vsim) is also available for download if you are so inclined.
- Run only behavioral simulation in Vivado 2014.2 (FIRST/Top option in the simulation menu pop-up).

Since I'm giving you the LaunchCaptureEngine.vhd code, you do not need to run timing simulations (Post Implementation) since I've worked out all the timing issues for you using IES. The IES waveform window is shown below along with the behavioral simulation window to illustrate the differences. Note that all FF and LUT delays in behavioral are ZERO. However, the phase shifted clock still works so you should be able to code-up the state machine as given by the pseudocode below.

I've also reverted to asynchronous RESET in the FF process blocks and removed the reset on the MMCM, both of which were causing reset and testbench difficulties.

Pseudo Code for LCTest_Driver.vhd

1) idle:
   Check 'start', de-assert ready, set target_phase initial value and start PhaseAdjust (FPA_start).
2) set_target_phase:
   Check if PhaseAdjust is done, assert Capture_ClkEn to capture initial value on next clock.
3) start_LC:
   De-assert Capture_ClkEn, store 'Capture_vals' in 'init_FU_vals'
4) evaluate_FU_outputs:
   Check LC is done, if so, compare saved initial values with current values ('Capture_vals). Store target_phase in 'resultx' if initial values have changed. Check and store each output value individually since the delays along 'sum' and 'Cout' will be different. Don't allow updates to 'resultx' once a value has been assigned.
5) check_done
   Check if both 'resultx' registers have been assigned values, if not, go to set_target_phase else go to idle.

Timing Simulation

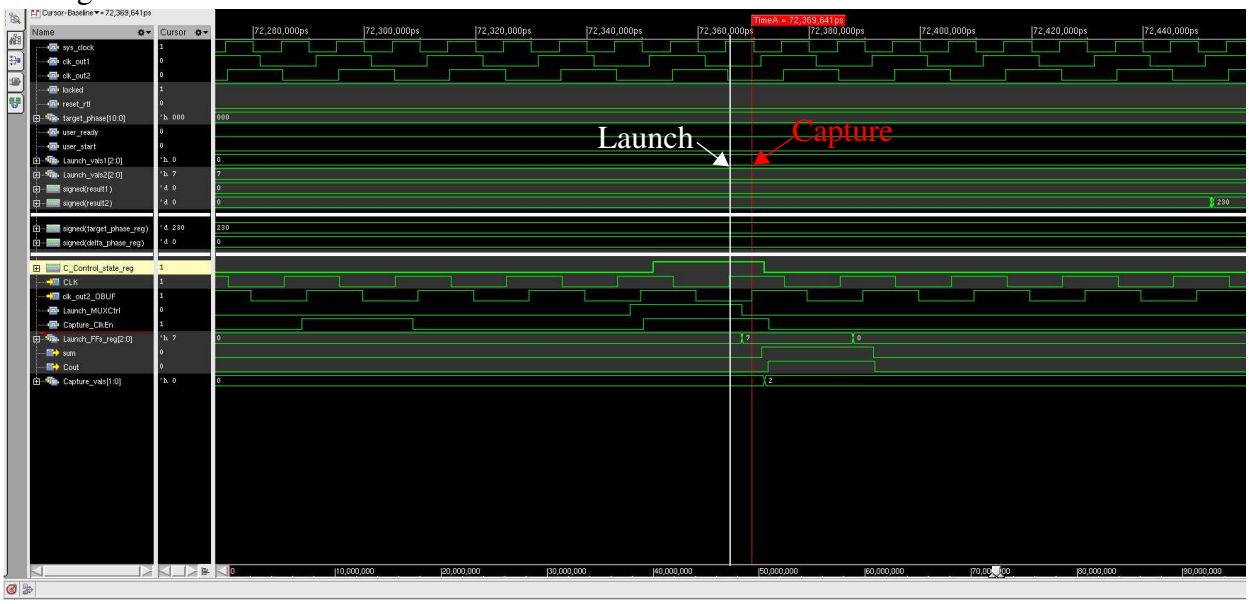


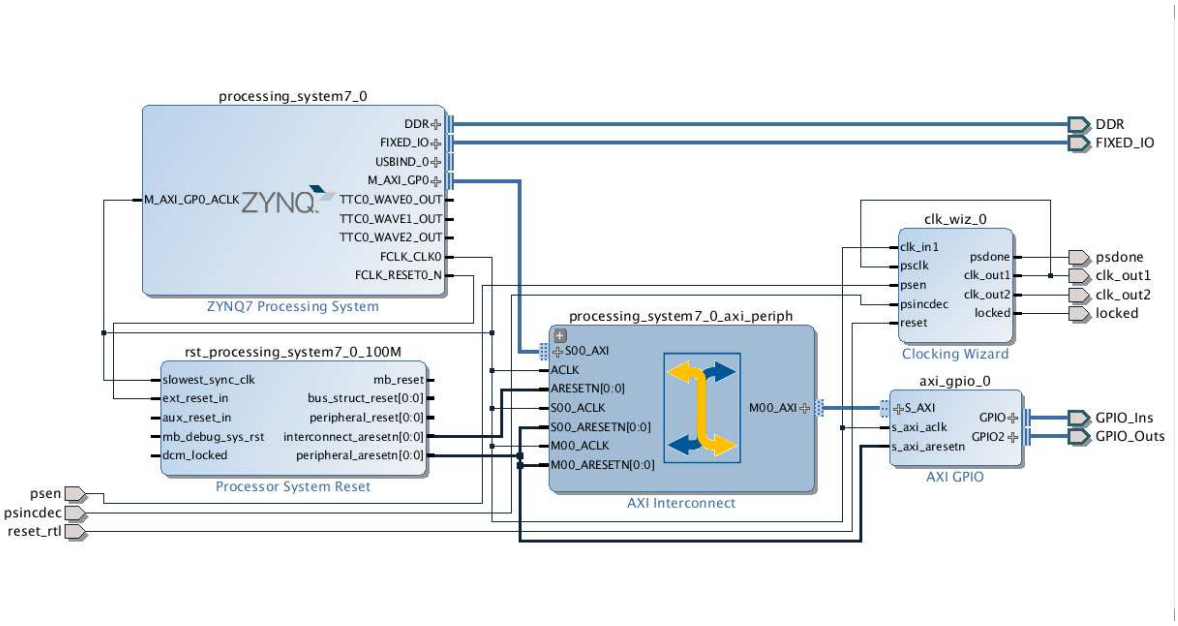

**Fig. 1. IES Timing Simulation showing Launch/Capture event.**

Behavioral Simulation with Vivado 2014.2:

**Fig. 2. Vivado 2014.2 Behavioral Simulation showing Launch/Capture event.**

**Part II: Hardware Demo:**

- Modify your existing Vivado project by adding a Zync processor and GPIO to the existing MMCM from Part I to the block diagram.
- Delete *sys_clock* pin in block diagram and route *FCLK_CLK0* to the *clk_in1* pin on the clk_wiz_0 instance.
- Enable both channels of the GPIO, make them both 'custom' and make the first GPIO channel 'all inputs' and the second channel 'all outputs'.
- Generate wrapper and add an instance of Top.vhd to the design_1_wrapper.vhd file.
  See figure:




- Delete *LCTD_start*, *LCTD_ready*, *Launch_vals1*, *Launch_vals2*, *result1* and *result2* from the Top.vhd entity
- Add the two 32-bit ports from the GPIO, *GPIO_Ins_tri_i*, *GPIO_Out_tri_o*.
  Note that *GPIO_Ins_tri_i* will be an OUT parameter while *GPIO_Outs_tri_o* will be an IN parameter in your Top.vhd module.
- Connect the *LCTD_start*, *Launch_vals1* and *Launch_vals2* to bits of your choice on *GPIO_Outs_tri_o*, and *LCTD_ready*, *result1* and *result2* to bits of your choice on *GPIO_Ins_tri_i*.
- Change *RESET* to *reset_mmcm* and change its direction to OUT.
  Inside your Top.vhd, you should assign 0 to this signal, i.e.,

```
reset_mmcm <= '0';
```

- Make the existing *RESET* a signal within Top.vhd and optionally connect it to a bit of *GPIO_Outs_tri_o* so that you can do a 'software' reset of your PL logic from your C program. Otherwise, connect it to '0', similar to *reset_mmcm* above.

Be sure to delete *clk_out1*, *clk_out2*, *gpio_ins_tri_i*, *gpio_outs_tri_o*, *locked*, *psdone*, *psen*, *psincdec* and *reset_rtl* from the design_1_wrapper entity.

See figure

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.all;

entity Top is
   port (
--       LCTD_start: in std_logic;
--       LCTD_ready: out std_logic;
       Clk_Launch : in std_logic;
       Clk_Capture : in std_logic;
       psdone : in std_logic;
       psen : out std_logic;
       psincdec : out std_logic;
       reset_mmcm : out std_logic
--       Launch_vals1: in std_logic_vector(2 downto 0);
--       Launch_vals2: in std_logic_vector(2 downto 0);
--       result1: out std_logic_vector(10 downto 0);
--       result2: out std_logic_vector(10 downto 0)
       );
end Top;
```

Top entity

```
10 library IEEE;
11 use IEEE.STD_LOGIC_1164.ALL;
12 library UNISIM;
13 use UNISIM.VCOMPONENTS.ALL;
14 entity design_1_wrapper is
15   port (
16     DDR_addr : inout STD_LOGIC_VECTOR ( 14 downto 0 );
17     DDR_ba : inout STD_LOGIC_VECTOR ( 2 downto 0 );
18     DDR_cas_n : inout STD_LOGIC;
19     DDR_ck_n : inout STD_LOGIC;
20     DDR_ck_p : inout STD_LOGIC;
21     DDR_cke : inout STD_LOGIC;
22     DDR_cs_n : inout STD_LOGIC;
23     DDR_dm : inout STD_LOGIC_VECTOR ( 3 downto 0 );
24     DDR_dq : inout STD_LOGIC_VECTOR ( 31 downto 0 );
25     DDR_dqs_n : inout STD_LOGIC_VECTOR ( 3 downto 0 );
26     DDR_dqs_p : inout STD_LOGIC_VECTOR ( 3 downto 0 );
27     DDR_odt : inout STD_LOGIC;
28     DDR_ras_n : inout STD_LOGIC;
29     DDR_reset_n : inout STD_LOGIC;
30     DDR_we_n : inout STD_LOGIC;
31     FIXED_IO_ddr_vrn : inout STD_LOGIC;
32     FIXED_IO_ddr_vrp : inout STD_LOGIC;
33     FIXED_IO_mio : inout STD_LOGIC_VECTOR ( 53 downto 0 );
34     FIXED_IO_ps_clk : inout STD_LOGIC;
35     FIXED_IO_ps_porb : inout STD_LOGIC;
36     FIXED_IO_ps_srstb : inout STD_LOGIC
37 --     clk_out1 : out STD_LOGIC;
38 --     clk_out2 : out STD_LOGIC;
39 --     gpio_ins_tri_i : in STD_LOGIC_VECTOR ( 31 downto 0 );
40 --     gpio_outs_tri_o : out STD_LOGIC_VECTOR ( 31 downto 0 );
41 --     locked : out STD_LOGIC;
42 --     psdone : out STD_LOGIC;
43 --     psen : in STD_LOGIC;
44 --     psincdec : in STD_LOGIC;
45 --     reset_rtl : in STD_LOGIC
46   );
47 end design_1_wrapper;
```

design_1_wrapper entity

- Create a C program that uses ‘mmap’ to access the GPIO registers.
- Write a simple program that sets the *Launch_vals1* and *Launch_vals2* to “000” and “111’ (try other combinations here as well after you get these two values to work).
- After setting the launch vector values, assert and then immediately de-assert (on the next instruction) *LCTD_start*.
- Enter a busy wait loop that loops while *LCTD_ready* is ‘0’.
- Once *LCTD_ready* becomes ‘1’, exit loop, read and print the *result1* and *result2* values.

## 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. Please print out and turn in a copy of your VHDL code.