CSIE ADIC Lab 2023

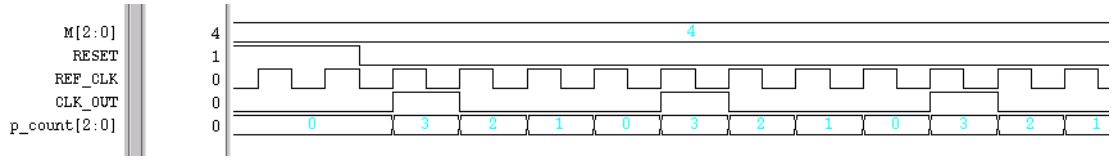
Lab04C. Design Programmable Frequency Divider and PLL Controller with NC-Verilog

1. Data Preparation

1. Extract LAB data from TA's directory:
   % tar xzvf ~adicta/lab04c.tar.gz
2. The extracted LAB directory (**lab04c**) contains:
   a. **Exercise1/** : Programmable Frequency Divider
   b. **Exercise2/** :All-Digital PLL Controller

2. Design a Programmable Frequency Divider

1. Change to the directory: **Exercise1**
2. Open and read the exercise1 Verilog codes.
   % gedit *.v &
   At the top level of this circuit (test_FREQ_DIV.v), a frequency divider (FREQ_DIV.v) is instantiated. In addition, a reference clock is provided to test the function of this circuit.
3. In this exercise, you need to design a counter-based frequency divider for all-digital PLL which meets the following requirements:
   a. Please use the counter-based architecture shown in **02_All-Digital_PLL.pdf, page 54** to build up your frequency divider.
   b. The input ports of FREQ_DIV are reset, clk, and M[2:0].
   c. The output port of FREQ_DIV is out_clk.
   d. The programmable M[2:0] ranges from 3'd1 to 3'd7.
4. After you finish the design for the frequency divider (**FREQ_DIV.v**). Execute NC-Verilog to simulate **test_FREQ_DIV.v**
   % ncverilog test_FREQ_DIV.v
   A log file named ncverilog.log will be generated. It records error or warning messages during Verilog simulation.
5. Start Verdi to debug the output waveform:
   % nWave &
6. The sample output waveform of this lab is shown in the following figure:



7. In this exercise, you also need to test all possible M[2:0] in your frequency divider to verify the correctness of FREQ_DIV.v.

3. Design a PLL Controller

1. Change to the directory: **Exercise2**
2. Open and read the exercise2 Verilog codes.
   % gedit *.v &
   In top-level of this circuit (test_CONTROLLER.v), a PLL Controller (CONTROLLER.v) is instantiated. A phase clock generated from the PFD's output up (P_UP) and down (P_DOWN) is provided to test the function of this circuit. Whenever P_UP=1'b0 is detected at the negative edge of the phase clock (phase_clk), the PLL Controller will increase the DCO control code(dco_code). When P_DOWN=1'b0 is detected at the negative edge of the phase clock (phase_clk), the PLL Controller will decrease the DCO control code (dco_code).
3. In this exercise, you need to design a PLL Controller for all-digital PLL which meets the following requirements:
   a. Please use the binary search architecture shown in **02_All-Digital_PLL.pdf, page 61** to build up your PLL Controller.
   b. You should use the binary search in the PLL controller. That means a variable search step should be designed. Whenever a low pulse in p_up or p_down is detected at the negative edge of phase_clk, a dco_code update is performed by adding or subtracting the previous dco_code with the search step. Whenever a p_up=1'b0 condition changes to p_down=1'b0 condition or vice versa, a polarity signal should be generated to indicate that the search step should be reduced to half of the previous step size. Finally, when the search step is reduced to 1, the freq_lock=1'b1 indicates the PLL is locked.
4. After you finish the design for PLL Controller (**CONTROLLER.v**). Execute NC-Verilog to simulate **test_CONTROLLER.v**
   % ncverilog test_CONTROLLER.v
   A log file named ncverilog.log will be generated. It records error or warning messages during Verilog simulation.
5. Start Verdi to debug the output waveform:
   % nWave &
6. The sample output waveform of this lab is shown in the following figure: