Exercises

EE332: Digital System Design

Lab2

Objectives

The objectives of this laboratory are the following:
1. To understand delay in VHDL codes.
2. To understand the differences between the signal assignment and variable assignment.
3. To practice on the developing of test bench for both combination and sequential circuits.
4. To become familiar with the structural coding style.
5. To practice on the sequential circuits, FSM, and FSMD design.
6. To understand and practice on pipeline design and parameterized design.

Complete the following exercises:

1. Do the Simulation of the VHDL code of Full adder on page 76.
   Design a test bench with stimulus of A, B and Cin as shown on page 77. Simulate the VHDL code in behavior simulation, post synthesis simulation and post place and route simulation. You may think and analyze your simulation results from, but limited to, the following aspects: 后两个simulation又分为function和time仿真
   a. If the simulation result is consistent with those shown in lecture notes.
   b. If the addition result of the full adder is consistent with common sense.
   c. What cause the simulation results different from that of common sense?
   d. Any difference between the 3 simulation results? Why there are such differences?
   e. What cause the spikes if any in the output signal? 毛刺
   Report is required for this exercise.

2. Derive the output res1 and res2 manually in the example in page 119 for the following stimulus:
   Apply signal x with a period of 80 ns, signal y with a period of 40 ns, and signal z with a period of 20ns. All x, y, and z are 0 initially with duty cycle of 50%.
   Design a test bench and simulate the result in the VHDL simulator to verify your derivation.

3. Simulate the three-digit decimal counter (pages 199 – 203). For the test bench design, refer to the example in pages 206 – 211.
   Modify the counter to have the following function and implement the counter onto the Nexys4 DDR board:
   1) A reset function to reset the counter to 000; use a push button to reset.

2) A load function to load an external number as the starting point of the counting; use a switch to turn on or turn off the load mode, and use 12 switches to set the number.
3) The counter increases by 1 for every 1 second.
4) Display the counting results on the 12 LEDs (4 LEDs represent 1 decimal digit).

In this exercise, the counter increases 1 for every 1 second. Meanwhile, the FPGA board provides a 100MHz clock source, which is 10^8 times faster than the required clock. In FPGA design, it should be noted that we should avoid using "combinational logic with registers" to generate clock signal (that is to be connected to the clock terminal of registers of counters), especially when the frequency and/or fan-out of the resultant clock are high.

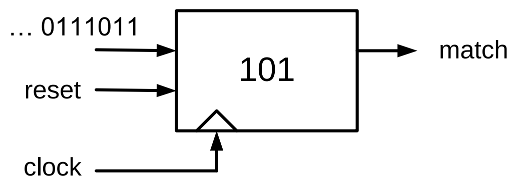
Instead, a typical way to generate a clock that is slower than the clock source is to use Clock Wizard of IP core. However, the slowest clock that can be generated by the Clock Wizard is 5MHz. Thus, in this exercise, you are required to **use Clock Wizard to generate a 10MHz clock** to be used by the decimal counter. In the decimal counter, **use an additional register to control the lowest decimal digit** to increase 1 for every 10^7 clock tics, while the others are the same as the decimal counter described in the lecture notes. In such a way, inside the decimal counter, you still use "combinational logic with registers" to counter 10^6 times, but the generated signal is not used as a clock signal; instead it is used to control combinational logic of the decimal counter. You should understand the difference between these two cases.

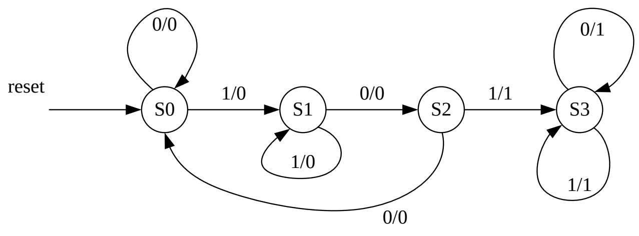
Requirements:

(1) Make use of Clock Wizard IP core to generate a 10MHz clock IP module.
(2) Develop a decimal counter which increases 1 for every 10^7 clock tics.
(3) In a top module, instantiate both the components of decimal counter and the clock module, and use the 10MHz clock signal generated from the clock module to control the decimal counter, such that the counter increases in a rate of 1 Hz.

Report is required for this exercise.

4. Consider the state machine shown below. The state machine has three inputs. The first is the reset signal, which initializes the machine to state 0. The second is a bit-serial input. The third is the clock input. The state machine recognizes the sequence 101 in the input sequence and sets the value of the output to 1. The value of the output remains at 1 until the state machine is reset.





(a) Derive the VHDL code for the FSM.
(b) Simulate the VHDL model.

5. The Fibonacci function is defined as

$$fib(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ fib(n-1) + fib(n-2) & \text{if } n > 1 \end{cases}$$

Implement this function in hardware. Assume that n is a 6-bit input and interpreted as an unsigned integer. Note that $fib(63)$ is 6557470319842.
(a) Derive an algorithm state machine chart.
(b) Derive the VHDL code in two-segment style.
(c) Simulate the VHDL model.

Exercises 4 and 5 are combined to submit one report.

6. Simulate and synthesize the two designs of an 8-bit adder-based multiplier discussed in the lecture (the combinational design P175-181, the repetitive-addition design P231-249). Examine the synthesized RTL schematics, and compare the performance of the different designs in terms of the device utilization, delay, etc.

7. Consider the pipelined designs of the 8 bit multiplier on P312-320. Develop an *n*-bit pipeline design of multiplier where *n* may be an integer from 1 to 9. Perform the post-layout simulation of the design, and find out the delay and throughput, and compare them with the design in Exercise 6.

Pre-Lab Preparation

1) Manually derive the output waveforms in Exercises 1 and 2.
2) Develop block diagrams for Exercise 3 for the decimal counter component, as well as the hierarchical structure of the top module.

3) Derive an algorithm state machine chart for Exercises 5.
These will be checked at the beginning of the lab.

During the Lab

For checkoff, you will demonstrate the following:

1. The simulation results (including the behavior simulation, post synthesis simulation and post place and route simulation) of the output waveforms of Exercises 1 and 2.
2. The implementation of Exercise 3 on the Nexys4 DDR board.
3. The post place and route simulation results of Exercise 4, 5, 6 and 7 for different inputs.

Lab Report

The report should include the following for each exercise:

1. Title
2. Objective of the laboratory. 10%
3. Pre-lab preparation results. 10%
4. Complete VHDL code and test bench code, and remarks for your codes. 20%
5. Results, including schematics, waveform images, and data, tables if any, and remarks for the results. 20%
6. Your thinking and analysis to the results. 30%
7. Conclusions. 10%

If a report contains only a title, a VHDL code with resultant waveform images, the report will have not than 60 marks over 100 marks. Pre-lab preparation results and analysis to your results take heavy weight for high marks.