Digital System Design

Homework #2

(Due on 03/18 PM 8:00)

Note: Please upload your codes to eeClass and hand in the **hardcopy** of this experiment including

a. Verilog codes
b. Test bench
c. Behavior simulation (text output and waveforms)
d. Synthesis timing report.
e. Post-route simulation (Note that a global reset will be asserted by VIVADO during post-route simulation. Please simply define the test pattern after 100ns.)

In this homework, we will implement sequential logic.

For even-numbered students,

1. In this homework, we learn an alternative method to implement incrementer. Fig. 1 shows the block diagram.

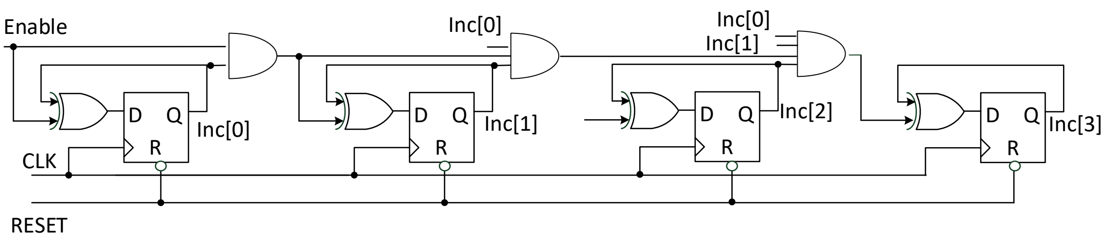

Fig. 1 Incrementer

You must consider the following functions:

(a) A clock signal "**CLK**", which has a period of 10ns.
(b) An active-low asynchronous "**RESET**" signal can clear all the registers to a pre-defined initial state, which corresponding to 4-bit binary representation of the last digit of your student ID. For example, if my last digit is 5, then "**Inc**=4'b0101" is used as the initial state of the register.
(c) An input signal "**Enable**" to activate the incrementer. If it is 0, the incrementer stops counting.
(d) An output signal "**Inc**[3:0]" that can count between 0 to 15.

Write Verilog codes to correctly implement these functions. Note that you have to write "XOR" and "AND" gates in your codes. (50%)

2. Write a testbench to test your sequential logic with the following signals: (15%)
Note that you have to repeat your clock signal until you can see the output "**Inc**[3:0]"

goes all possible <u>16 states</u>. Then, de-assert "**Enable**" signal to stop the incrementer.

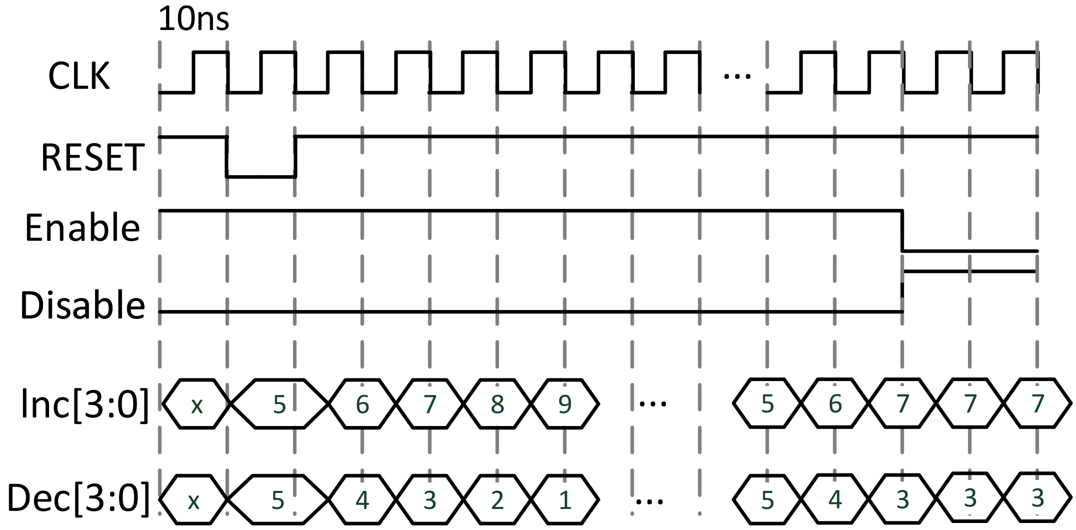

Fig. 2 Test pattern (example of a student whose ID is 5)

3. Show the text mode results of behavior simulation by the command "monitor". (5%)
4. Show the waveform results of behavior simulation and post-route simulation. (20%)
5. List the timing report. (10%)

For odd-numbered students. (Replace 1 and 2 by 6 and 7)

6. Please use the circuits in Fig. 3. Note that it is a decrementer.

   (a) A clock signal "**CLK**", which has a period of 10ns.

   (b) An active-low asynchronous "**RESET**" signal can clear all the registers to a pre-defined initial state, which corresponding to 4-bit binary representation of the last digit of your student ID. For example, if my last digit is 5, then "**Inc**=4'b0101" is used as the initial state of the register.

   (c) An input signal "**Disable**" to stop the decrementer. If it is 1, the decrementer stops counting.

   (d) An output signal "**Dec**[3:0]" that can count between 15 to 0.

   Write Verilog codes to correctly implement these functions. Note that you have to write "XNOR" gates and "OR" gates in your codes. (50%)

7. Write a testbench to test your sequential logic with the following signals: (15%)

   Note that use your last digit and transfer it to binary representation as the initial state and repeat your clock signal until you can see the output "**Dec**[3:0]" goes all possible <u>16 states</u>. Then, assert "**Disable**" signal to stop the incrementer.

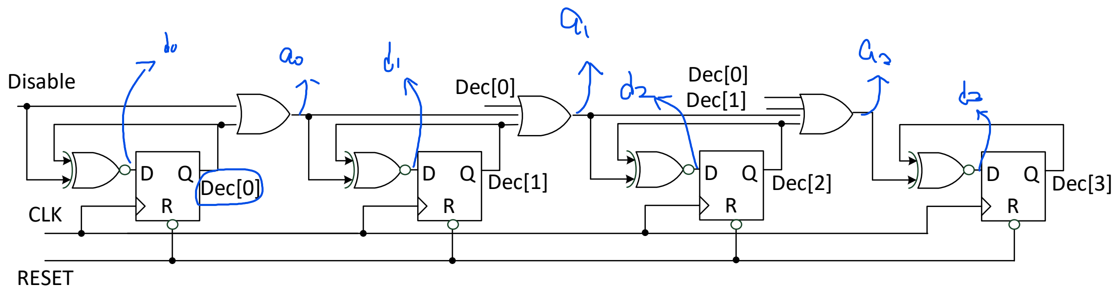

Fig. 3 Decrementer