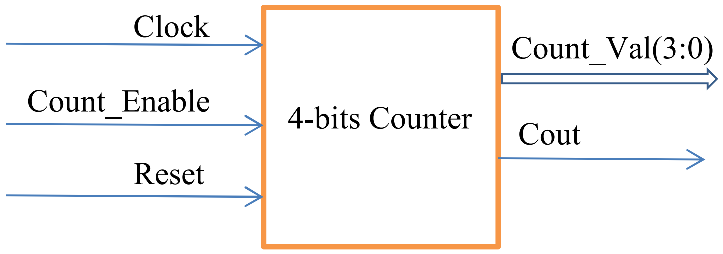
Lab #9: Counter Design

I. 4-bit Binary Counter Design

1. Requirement

Design a 4-bit binary counter with asynchronous reset using ***behavioral implementation style***. The component starts counting if Count_Enable signal is 1, otherwise it will stop counting. The counter will reset to "0000" when Reset signal is 1. The carry out happens when the count value exceeds 15 (overflow occurs). The block diagram of this component is as below:



The interface can be as below:

```
entity CounterUnit_4Bits_Design is
      port(
            Clock        :     in  std_logic;
            Count_Enable:     in  std_logic;
            Reset        :     in  std_logic;
            Cout         :     out std_logic;
            Count_Val    :     out std_logic_vector(3 downto 0)
      );
end ShifterUnit_32Bits_Design;
```

2. Pre-lab

- Study the working of binary counters
- Study conversion functions in library **numeric_std** to convert integer to std_logic_vector and vice versa (look at the guidelines in the memory design lab for your reference)

3. Lab

- Write the VHDL code for this unit
- Simulate using Xilinx ISE simulator

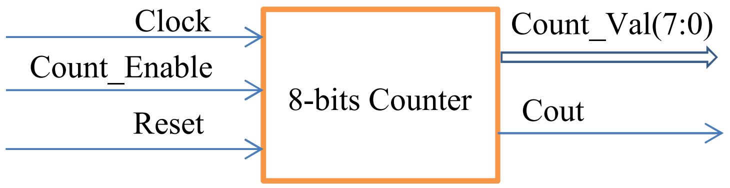
4. Deliverables

- VHDL program
- VHDL test bench with enough test cases
- Waveform with comments

II. 8-bit binary counter design

1. Requirement

Design an 8-bit binary counter with asynchronous reset from above 4-bit binary counter using structural implementation style. Its behavior is similar to 4-bit binary counter. The carry out happens when the value exceeds 255 (overflow occurs). The block diagram of this component is as below:



The interface can be as below:

```
entity CounterUnit_8Bits_Design is
        port(
                Clock           :     in   std_logic;
                Count_Enable:     in   std_logic;
                Reset           :     in   std_logic;
                Cout            :     out  std_logic;
                Count_Val       :     out  std_logic_vector(7 downto 0)
        );
end CounterUnit_8Bits_Design;
```

2. Pre-lab

- Block diagram of 8-bits counter from **two** 4-bits counters (How are 2 4-bits counters connected to create a 8-bits counter)

3. Lab

- Write the VHDL code for this unit
- Simulate using Xilinx ISE simulator

4. Deliverables

- VHDL program
- VHDL test bench with enough test cases
- Waveform with comments

III. Bonus Lab (Optional): Mod-12 counter (50 points)

1. Requirement

Design a mod-12 counter using the 4-bit binary counter designed above. Designing “mod” counters involves “clearing” the counter at the specified count. It may involve adding a new NAND gate entity to clears the circuit.

2. Lab/Deliverable

- VHDL program
- VHDL test bench with enough test cases
- Waveform with comments

Note: mod-12 counter means that the counter will automatically reset to 0 after counting to 11

For example: 0 1 2 3 4 5 6 7 8 9 10 11 0 1 2 3 4 5 6 7 8 9 10 11 0 1 2 . . . .