Rotating LED Circuit

Signature and Grading Sheet

Group #:_____ Name(s):_____ .

Grading

- Section 4.1(b): VHDL code (25 points):_____.
  Attach code
- Section 4.2(b): VHDL code (25 points):_____.
  Attach code
- Section 4.3(a): VHDL code (10 points):_____.
  Attach code
- Section 4.4(b) RT-level simulation waveform (20 points):_____.
  Attach simulation waveform screen capture
- Section 4.5(e): post-synthesis simulation waveform (10 points): _____.
  Attach simulation waveform screen capture
- VHDL code format and comments (10 points):

Total points: .

Experiment
Rotating LED Circuit

1 Purpose

To design an intermediate-sized sequential circuit

2 Reading

- Chapter 4 of *FPGA Prototyping by VHDL Examples*

3 Project specification

We want to design a circuit that circulates 5 LEDs as follows:

- One out of 5 LEDs is asserted
- The asserted LED can rotate in either direction:
  - rotate right and wrap around: 00001→ 10000 → 01000→ 00100 → 00010 → ⋯
  - rotate left and wrap around: 00001 → 00010 → 00100→ 01000 → 10000 → ⋯

The control signals of the circuit can specify the rotation speed, the direction of rotation (i.e., right or left), and pause the operation. The input and output are

- input:
  - `clk`: clock signal.
  - `reset`: reset signal.
  - `pause`: 1-bit enable signal. The rotation pauses when it is 1.
  - `rt`: 1-bit direction signal. The pattern rotates right when it is 1 and rotates left when it is 0.
  - `fast`: 1-bit speed control:
    - 1: each pattern stays 2 clock cycles (i.e., 5*2 clocks to complete one rotation)
    - 0: each pattern stays 4 clock cycles (i.e., 5*4 clocks to complete one rotation)
- output
  - `dout`: 4-bit output connected to 4 LEDs

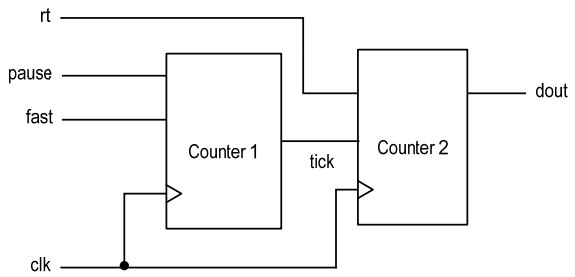
The entity declare of this circuit is

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity rotate_led is
   port(
      clk, reset: in std_logic;
      pause, rt, fast: in std_logic;
      dout: out std_logic_vector(4 downto 0)
    );
end rotate_led;
```

The design must synchronous, or 50% will be deducted.

4 Design Procedures

We can divide this circuit into two segments:



- Counter 1: a pulse generator that generates a one-clock pulse (tick) every 40 ns or 80 ns when it is not paused.
- Counter 2: a mod-5 counter that counts up or counts down and decodes the counter values to 5 output patterns.

4.1 Counter 1

(a) Design counter 1 as an independent VHDL module. The entity declaration of this design is

```
entity counter1 is
    port(
       clk, reset: in std_logic;
       pause, fast: in std_logic;
       tick: out std_logic
    );
end counter1
```

(b) Derive the architecture body.

4.2 Counter 2

(a) Design counter 2 as an independent VHDL module. The entity declaration of this design is

```
entity counter2 is
   port(
      clk, reset: in std_logic;
      tick, rt: in std_logic;
      dout: out std_logic_vector(4 downto 0)
    );
end counter2;
```

(b) Derive the architecture body.

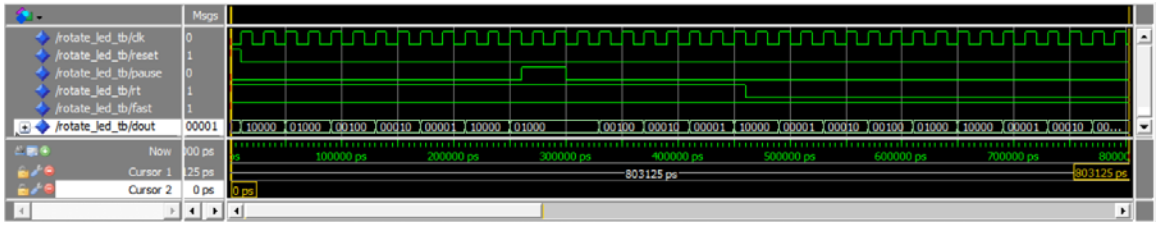
4.3 Top-level circuit

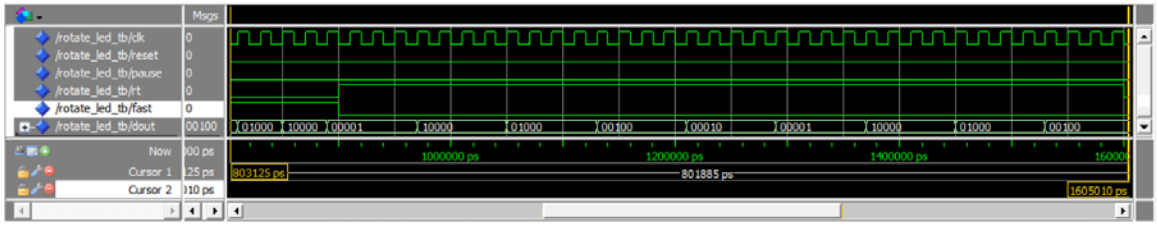
(a) Derive the VHDL code for the top-level system using the component instantiations of the previous modules.

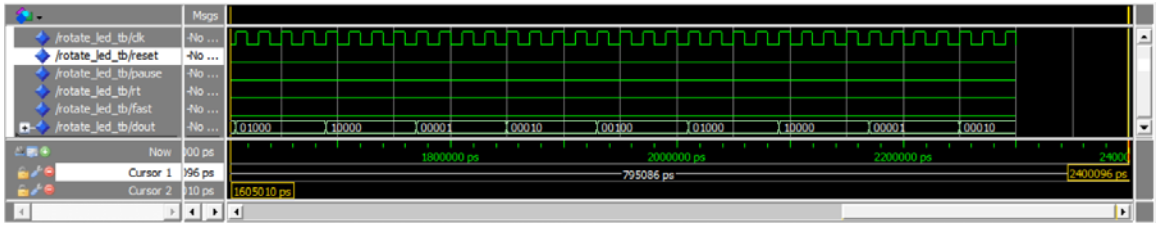
4.4 Simulation

(a) Use the testbench (`rotate_led_tb.vhd`) to simulate your VHDL code.

(b) Develop a proper "layout" and "format" for the simulated waveform. Use multiple screen captures to make the 5-bit output rotation patterns visible (for example, the timing diagrams below breaks the 2400-ns simulation into three 800-ns captures). To get full credits, the input and output signals should be properly arranged and represented in a proper format so that the simulation result can be easily understood.







4.5 Post-synthesis ModelSim simulation

(a) Perform compiling (synthesis/placement and routing) and obtain the .vho file.

(b) Follow the format guideline to add a header to the file and verify the "structure" architecture body is generated.

(c) Revise the testbench to use the "structure" architecture for uut.

(d) Perform post-synthesis simulation

(e) Do screen capture(s) of the simulated result.

(f) Since the waveform is the same as the RT-level simulation, the snapshot should include the expanded uut on the left panel. No point will be given if the uut unit is not expanded in the snapshot.