Lab Assignment 6

Title: Brightness control of LEDs

Learning Objective:
Learn how to control brightness of LED displays using pulse width modulation

Specification:
Design a circuit that controls brightness of LED displays based on a 4-bit input. Use pulse width modulation to control LED brightness.

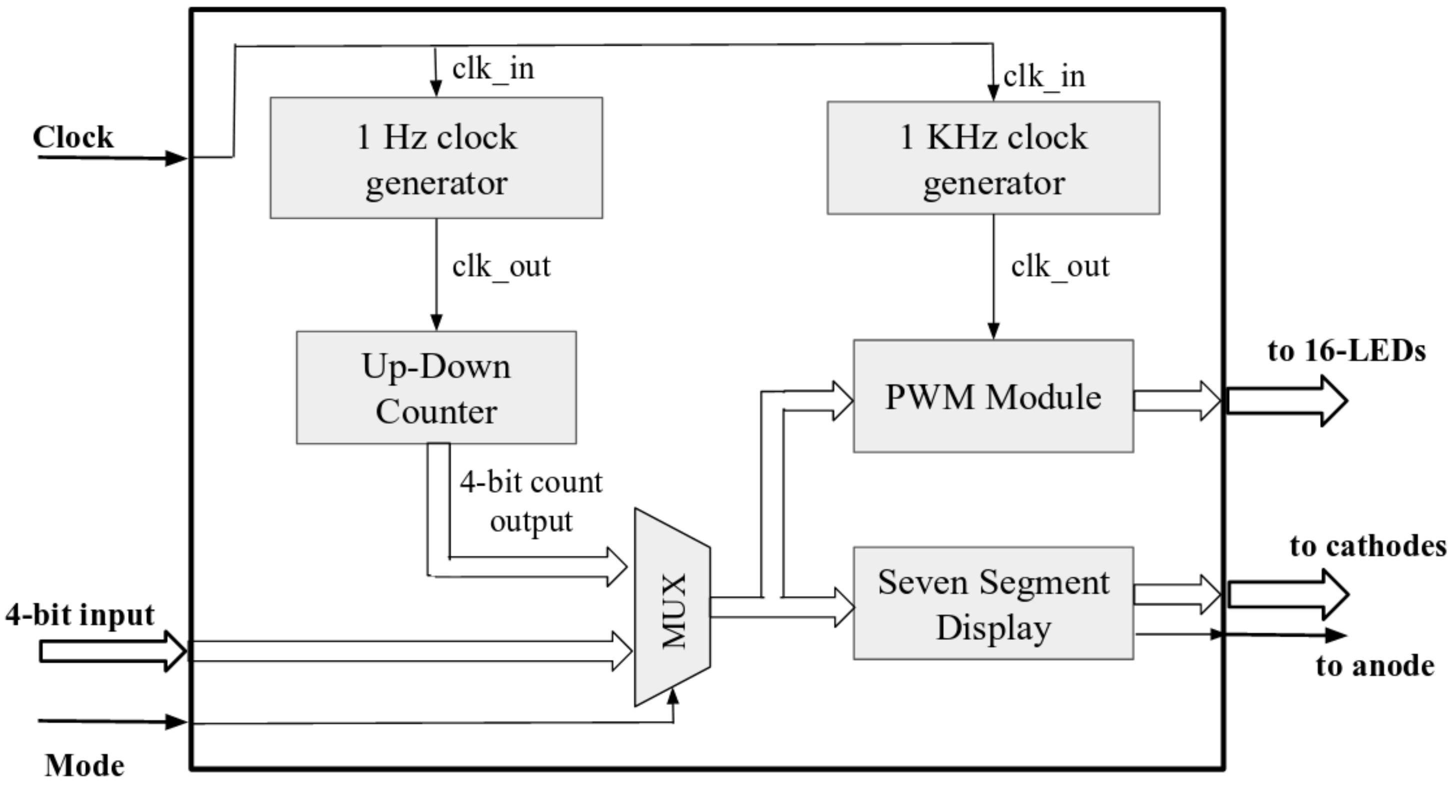
Details:
Design and implement a circuit to vary brightness of 16 LEDs using PWM. Let there be 16 levels of brightness, corresponding to duty cycle varying from 0 to 15/16, in steps of 1/16. Display brightness level (numeric value) using 7-segment display.

For specification of the brightness level, define two modes – (i) switch mode and (ii) continuous mode. In “switch mode”, 4 slide switches specify the brightness level. In “continuous mode”, a 4-bit up/down counter specifies the brightness level. This counter repeatedly counts from 0 to 15 and then down to 0, at a rate that is perceptible to eye. Use a fifth slide switch for selecting the mode.

Express your design in VHDL, verify the logic using simulator and then synthesize and test on the FPGA board.

Use 1 Hz clock for Up-Down counter and clock of 1 KHz to be used for PWM module. PWM module has the functionality of varying brightness of LEDs corresponding to duty cycle. If the number to be displayed is 3, then all LEDs should be ON for 3 out of 16 cycles.

Block Diagram:



Top Module

The sequential blocks in the block diagram should be implemented using process(..) in VHDL. Combinational blocks can be implemented using concurrent statements in VHDL.

Some Programming hints :

- Where ever there is assignment statement as shown
  a <= "0000";
  use ***others***
  a <= (others => '0');
  others is used assign all numbers in the asignment to 0/1.
  a <= "1000"; can be replaced with
  a <= (3 => '1', others => '0');
  a <= "0011"; can be replaced with
  a <= (1 downto 0 => '1', others => '0');

- For clock frequency conversion :
  N = clk_input/(2*clk_output)
  Use a signal which increments count from 0 to N, then invert the clock output (Initial clock output is 0) and re-initialize count to 0.

- For std_logic_vector to integer or any other conversions, refer link given
  https://www.nandland.com/vhdl/tips/tip-convert-numeric-std-logic-vector-to-integer.html

Code Snippet :

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity <lab_6> is

port( clk : in std_logic;

        input : in std_logic_vector(3 downto 0);

       -------);

end <lab_6>;

architecture Behavioral of <lab_6> is

<signal declarations>
begin

-- 1 Hz Clock generation

   process(----,--, )
   begin
   ----
   end process;
  -- 1 KHz Clock generation
    process( ---,---,)
    begin
   ---
   end process;
-- Up-Down Counter
    process(---,---, )
     begin
   ---
   end process;
-- PWM
  process( ---,---, )
  begin
   ---
   end process;

  •

  •

  •

  •
end Behavioral;
```