Embedded Systems (CSCE 4114)

Lab 2

Zack Fravel

9/15/16

zpfravel@uark.edu

Abstract

Laboratory 2 consisted of designing and implementing a pulse generator as well as changing the behavior of the original tutorial circuit to shift the LED's. By the end of the project, we have the 1 LED shifting to the left across the 8 spaces at variable rates based on the switch inputs.

Introduction

This lab is an extension of the first lab. We come at it with our previous test-bench, which will later be slightly modified, as well as our same tutorial.vhd file we implemented in the previous lab. We're looking to add a shift functionality to this module, allowing the positions of the lights to change with respect to time. The problem with our current design is there's no way to translate it into a timeframe that is observable by humans. Everything is happening at the nanosecond ($1x10^{-9}$ sec) level.

It's at this point where we need to implement what we'll refer to as a pulse generator module. The pulse generator will output a signal, or pulse, at a rate that we can determine in order to be able to observe the shift. Once we have this pulse generator implemented, we should then be able to modify the functionality of our system so that, instead of the switches controlling configurations with one option to scroll, we have an array of LED's that scroll at various rates based on the switch orientation.

Design and Implementation

The first order of business in this lab was to modify our original design to be capable of a left shift when the switch (3) input is high. The figure at the top of the next page describes exactly what functionally we're looking for.

Shift module

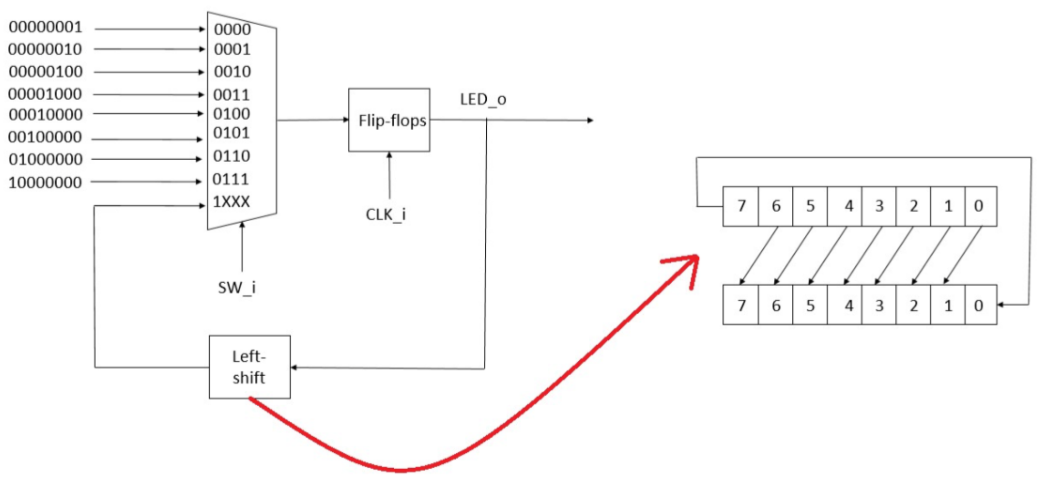


Figure 1

This is implemented in design fairly simply. On my last "when others" case statement, I added a condition so that on every rising clock edge the LED output signal is assigned as follows:

LED_s (7 downto 1) <= LED_s (6 downto 0);

LED_s (0) <= LED_s (7);

and outside the process that LED_s signal is assigned to the LED output. After that, when run in the simulator it was working, however when downloaded to the spider, all the 4th switch seemed to do was light up all the LED's. This is because you cannot actually see the LED's shifting, they're moving so fast because they're shifting every clock cycle. With 1,000,000,000 ns in 1 sec and a 20 ns clock cycle, this means the LED is shifting 50,000,000 times every second; obviously, this is far beyond the speed any human could possibly discern.

This is where the pulse generator module comes in! The pulse generator outputs a pulse at a programmable rate. For our implementation, we want to pick a speed that we can observe in real time on the board. For example, the figure below shows the pulse generator operating at a rate of 16 Hz, or 16 pulses per second. The way this module is implemented in the design is we add an integer signal (called counter) and create a process that checks whether or not this integer has reached a value specified by the engineer every clock cyclce. If the signal hasn't reached the value, it is incremented once and the process repeats itself. So for 16 Hz we need our counter to go to 3,125,000.

1,000,000,000 ns per second

/ 20 ns clock cycle = 50,000,000 clock cycles per second

/ 16 Hz = **3,125,000 Clock Cycles for 16 Hz rate**

Pulse Generator (Before, During, After)

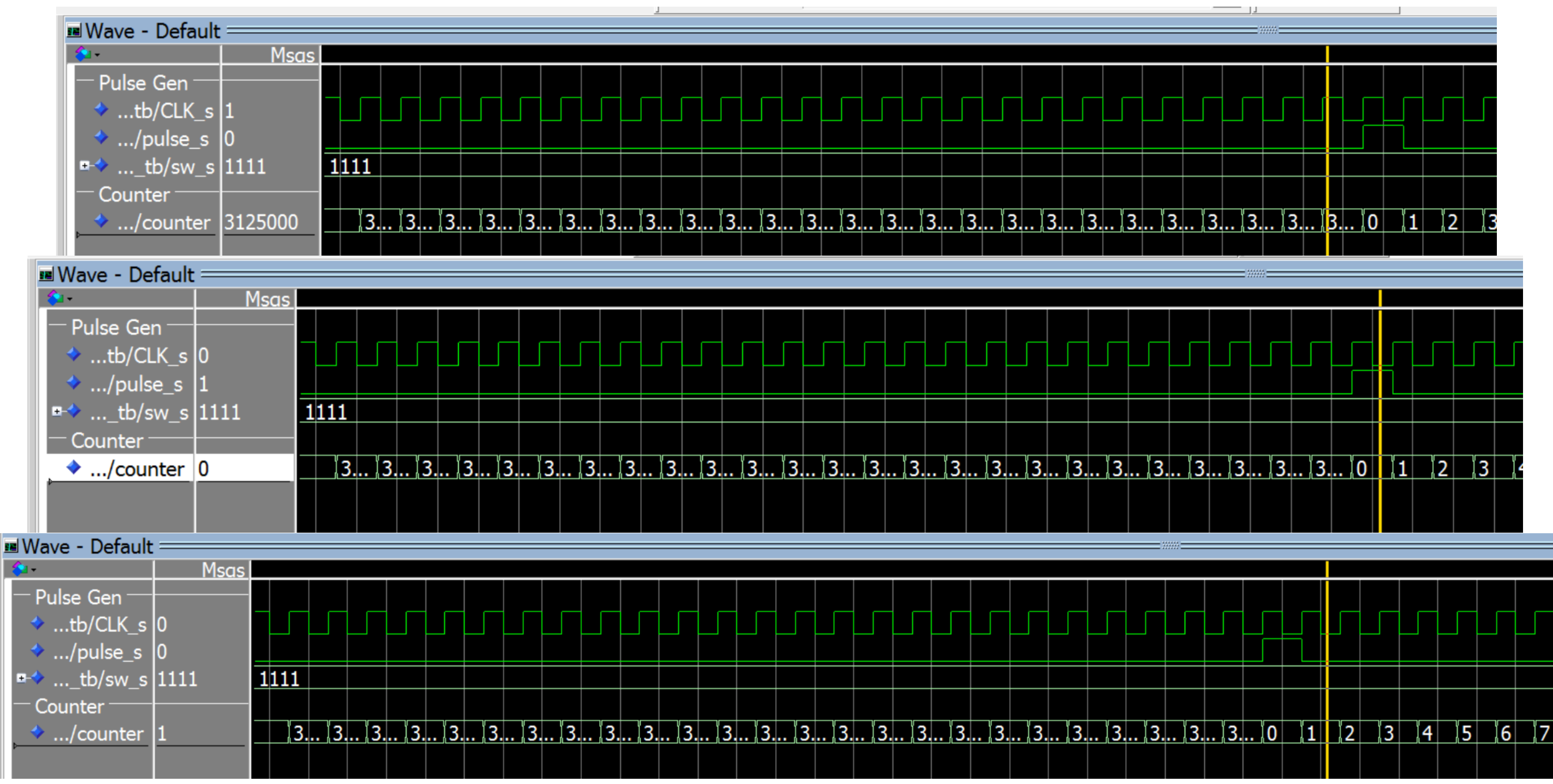


Figure 2

With the pulse generator implemented and compiled, the LEDs can be seen shifting in real time.

The final step in the lab was to modify the system so that instead of the top level design being driven by the pulse generator but performing the logic; we want to modify the top level design so that all it is responsible for is shifting the LED's when it receives a pulse. We will then add a new input to our pulse generator, connect the switches to that, and remove the excess logic from the top level entity. Now, all that is left to do is to add the rest of the logic in the pulse generator module for the different pulse rates in as a case statement dependent on the switches. The implementation is as follows: for each case with the four-bit switches (0000, 0001, 0010, etc.) we add a separate if statement that checks if the counter has reached a value specific to the rate corresponding to the orientation of the switches.

Pulse Generator Logic

```
If (SW_i="0000") then frequency should be 1 Hz
If (SW_i="0001") then frequency should be 2 Hz
...
If (SW_i="1111") then frequency should be 16 Hz
```

Figure 3

Final Design

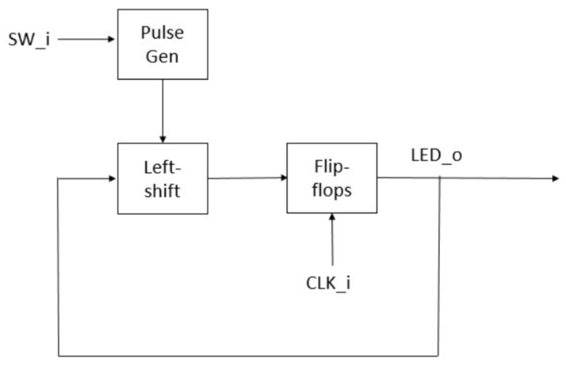


Figure 4

To find the corresponding values, I used the same division process I laid out on the previous page for each different rate (using 50,000,000 CC per second as the scale). The source code for the final implementation is attached.

Results

Once all is said and done and everything is compiled, we are left with what we were wanting to accomplish! Below is a waveform showing three different rates of LED shift on the simulation. The simulation takes place over a course of 4 sec, which takes a long time to simulate!

Final Waveform

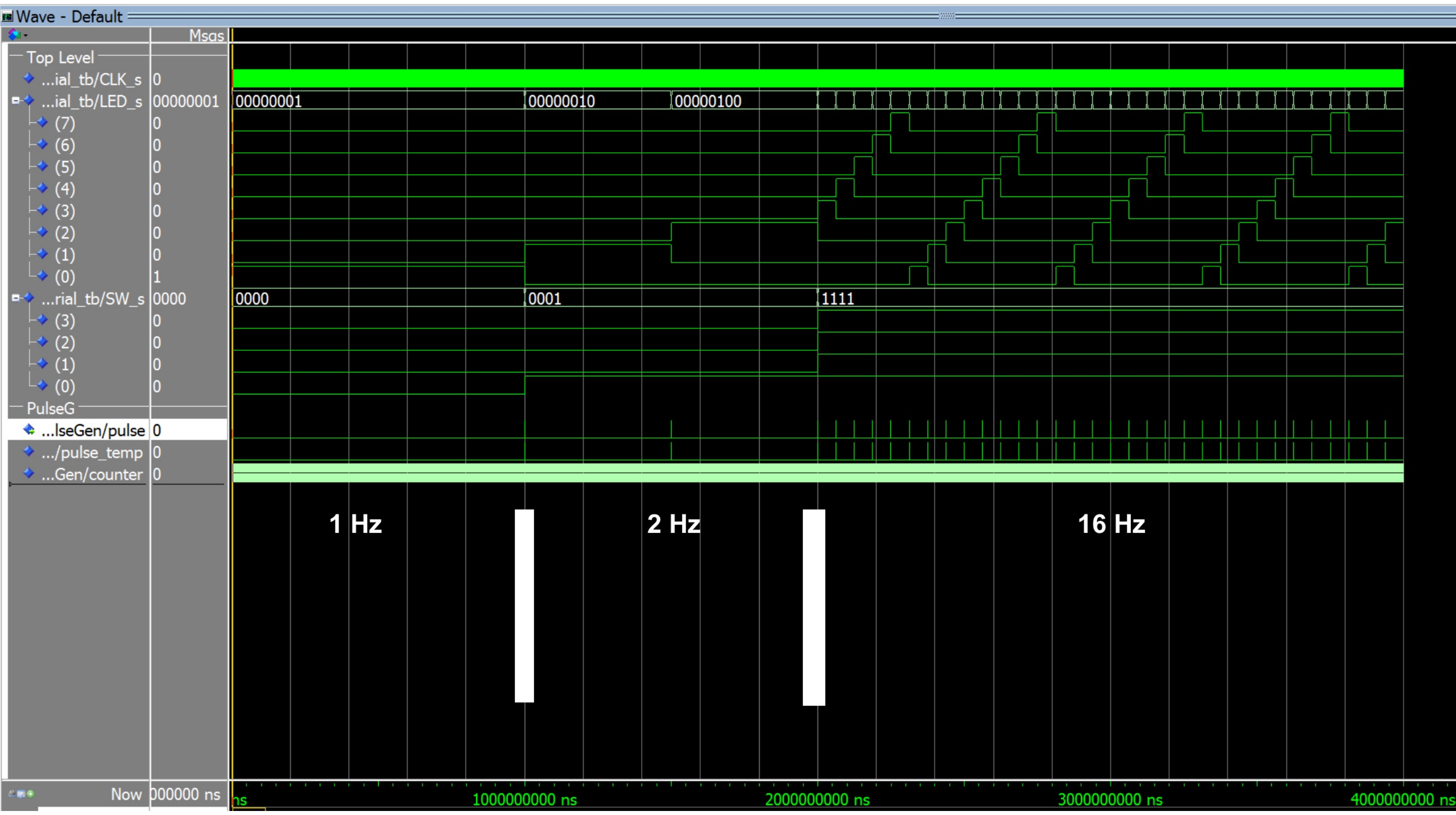
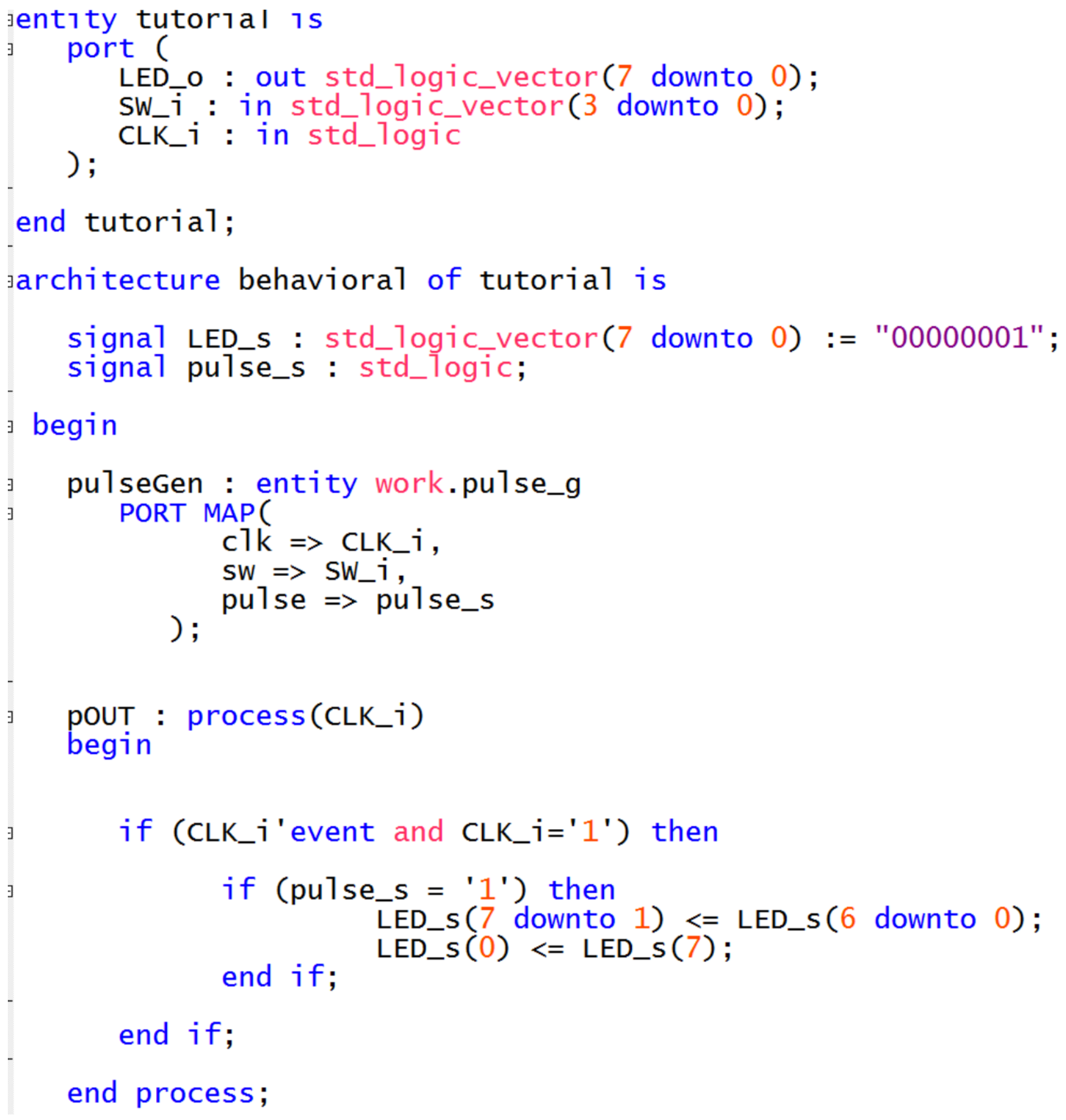


Figure 5

It can be observed above that the “resting state” of the functioning circuit is the LED’s shifting once per second. After 1 second, the first switch is flipped for a rate of 2 Hz and the corresponding LEDs are lit accordingly. Finally, the last 2 seconds of the simulation show the circuit running at 16 Hz, which allows the LEDs to cycle twice per second. The only main issue I encountered was in my pulse generator at first it would simulate but not act properly when downloaded to the Spider board. After making a few modifications to the code and when the counter variable was being reset, everything went smoothly.

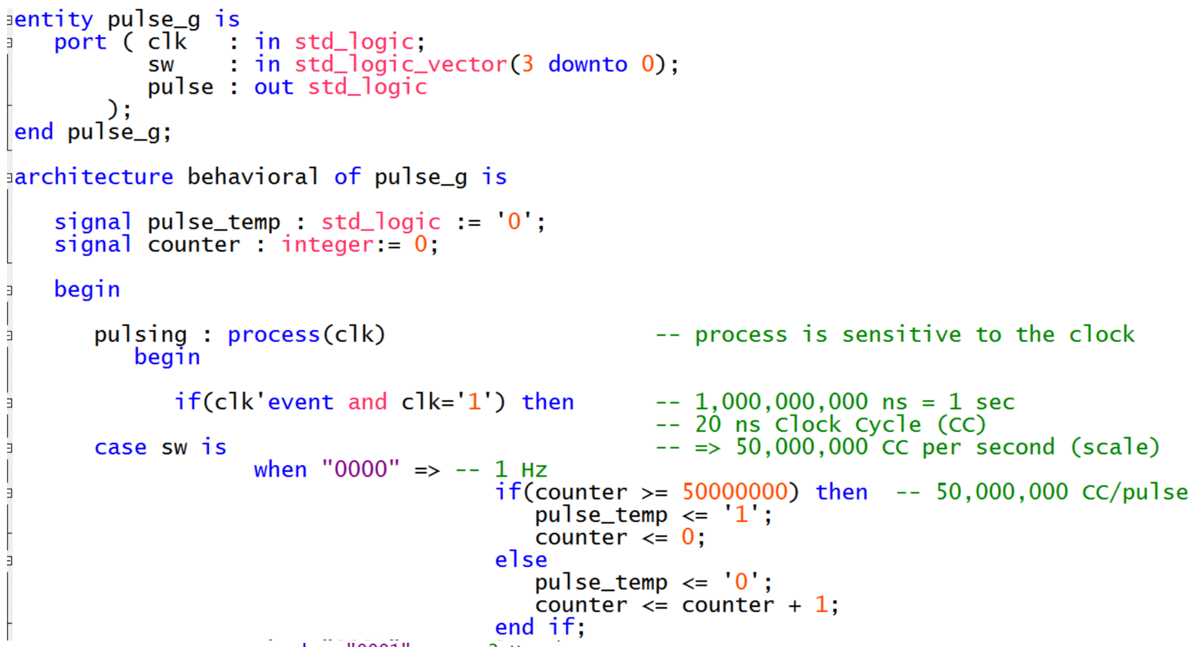
Our original problem was to modify our existing LED/switch structure to allow left shift capability, as well as make the switches correspond to the rate at which the LED’s shift. Using a large case statement and correctly instantiating new signals in the pulse generator and connecting them to the original module, I came up with a design that does just that.

Tutorial.VHD

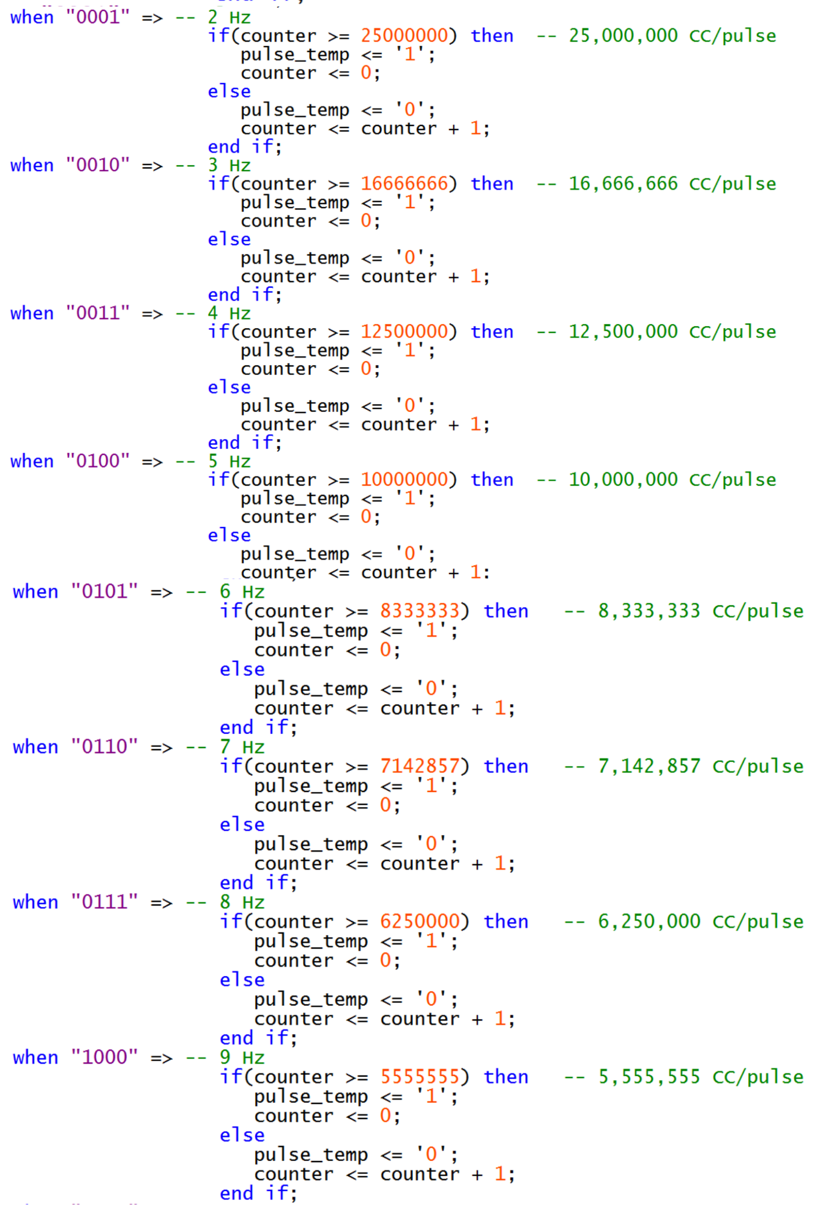


```vhdl
entity tutorial is
   port (
      LED_o : out std_logic_vector(7 downto 0);
      SW_i : in std_logic_vector(3 downto 0);
      CLK_i : in std_logic
   );

end tutorial;

architecture behavioral of tutorial is

   signal LED_s : std_logic_vector(7 downto 0) := "00000001";
   signal pulse_s : std_logic;

 begin

   pulseGen : entity work.pulse_g
      PORT MAP(
            clk => CLK_i,
            sw => SW_i,
            pulse => pulse_s
         );


   pOUT : process(CLK_i)
   begin


      if (CLK_i'event and CLK_i='1') then

            if (pulse_s = '1') then
                     LED_s(7 downto 1) <= LED_s(6 downto 0);
                     LED_s(0) <= LED_s(7);
            end if;

      end if;

   end process;
```

Pulse_G.VHD



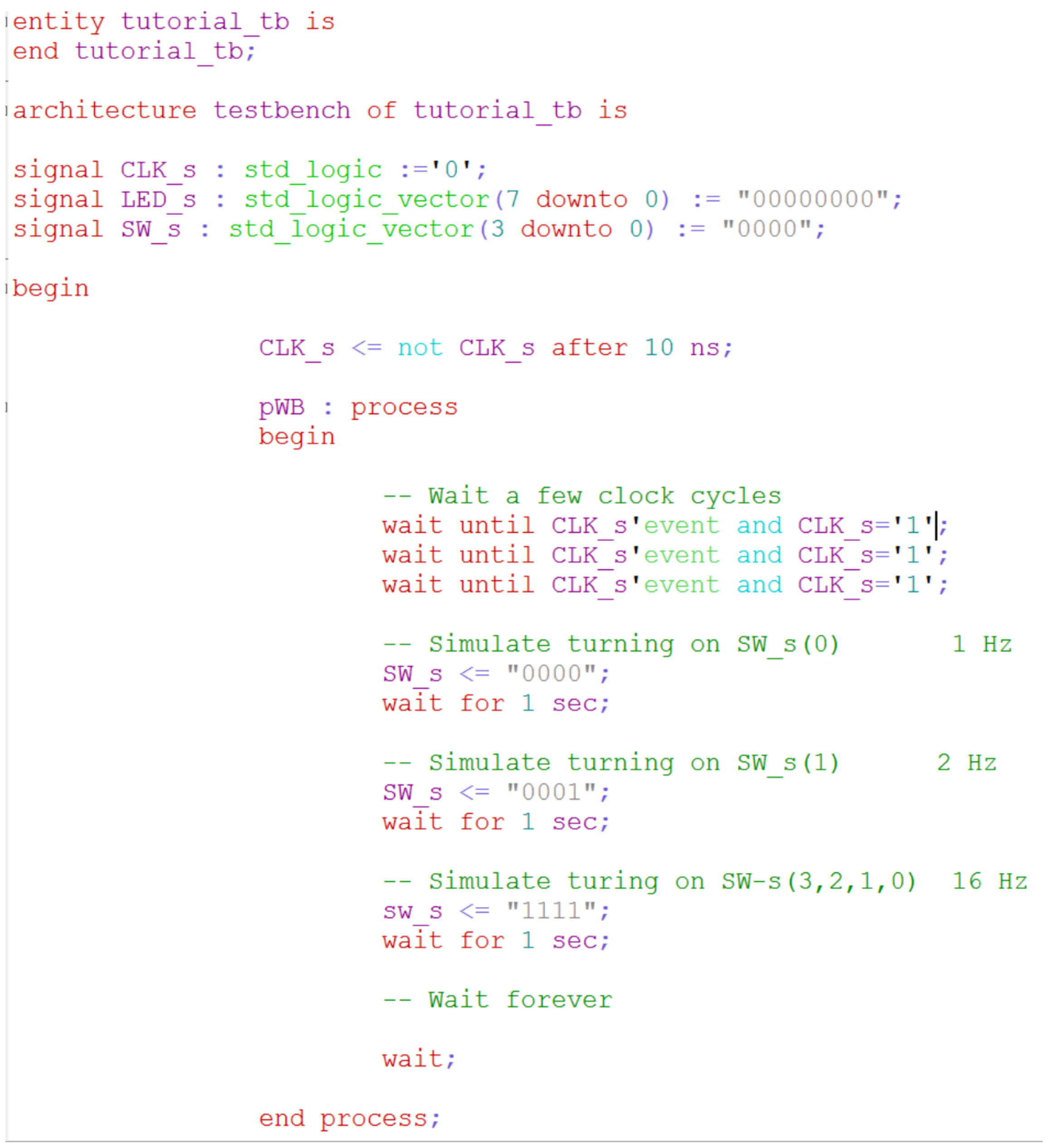
```
entity pulse_g is
   port ( clk   : in std_logic;
          sw    : in std_logic_vector(3 downto 0);
          pulse : out std_logic
      );
end pulse_g;

architecture behavioral of pulse_g is

   signal pulse_temp : std_logic := '0';
   signal counter : integer:= 0;

   begin

      pulsing : process(clk)                     -- process is sensitive to the clock
         begin

            if(clk'event and clk='1') then       -- 1,000,000,000 ns = 1 sec
                                                 -- 20 ns Clock Cycle (CC)
      case sw is                                 -- => 50,000,000 CC per second (scale)
                  when "0000" => -- 1 Hz
                                    if(counter >= 50000000) then  -- 50,000,000 CC/pulse
                                       pulse_temp <= '1';
                                       counter <= 0;
                                    else
                                       pulse_temp <= '0';
                                       counter <= counter + 1;
                                    end if;
```



```
      when "0001" => -- 2 Hz
                     if(counter >= 25000000) then  -- 25,000,000 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "0010" => -- 3 Hz
                     if(counter >= 16666666) then  -- 16,666,666 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "0011" => -- 4 Hz
                     if(counter >= 12500000) then  -- 12,500,000 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "0100" => -- 5 Hz
                     if(counter >= 10000000) then  -- 10,000,000 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1:
      when "0101" => -- 6 Hz
                     if(counter >= 8333333) then   -- 8,333,333 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "0110" => -- 7 Hz
                     if(counter >= 7142857) then   -- 7,142,857 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "0111" => -- 8 Hz
                     if(counter >= 6250000) then   -- 6,250,000 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
      when "1000" => -- 9 Hz
                     if(counter >= 5555555) then   -- 5,555,555 CC/pulse
                        pulse_temp <= '1';
                        counter <= 0;
                     else
                        pulse_temp <= '0';
                        counter <= counter + 1;
                     end if;
```

```vhdl
               when "1001" => -- 10 Hz
                                 if(counter >= 5000000) then   -- 5,000,000 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when "1010" => -- 11 Hz
                                 if(counter >= 4545454) then   -- 4,545,454 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when "1011" => -- 12 Hz
                                 if(counter >= 4166666) then   -- 4,166,666 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when "1100" => -- 13 Hz
                                 if(counter >= 3846153) then   -- 3,846,153 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when "1101" => -- 14 Hz
                                 if(counter >= 3571428) then   -- 3,571,428 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when "1110" => -- 15 Hz
                                 if(counter >= 3333333) then   -- 3,333,333 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;
               when others => -- 16 Hz
                                 if(counter >= 3125000) then   -- 3,125,000 CC/pulse
                                    pulse_temp <= '1';
                                    counter <= 0;
                                 else
                                    pulse_temp <= '0';
                                    counter <= counter + 1;
                                 end if;

            end case;
         end if;
      end process;

   pulse <= pulse_temp;

end behavioral;
```

Tutorial_tb.VHD



```
entity tutorial_tb is
end tutorial_tb;

architecture testbench of tutorial_tb is

signal CLK_s : std_logic :='0';
signal LED_s : std_logic_vector(7 downto 0) := "00000000";
signal SW_s : std_logic_vector(3 downto 0) := "0000";

begin

                CLK_s <= not CLK_s after 10 ns;

                pWB : process
                begin

                        -- Wait a few clock cycles
                        wait until CLK_s'event and CLK_s='1';
                        wait until CLK_s'event and CLK_s='1';
                        wait until CLK_s'event and CLK_s='1';

                        -- Simulate turning on SW_s(0)       1 Hz
                        SW_s <= "0000";
                        wait for 1 sec;

                        -- Simulate turning on SW_s(1)      2 Hz
                        SW_s <= "0001";
                        wait for 1 sec;

                        -- Simulate turing on SW-s(3,2,1,0)  16 Hz
                        sw_s <= "1111";
                        wait for 1 sec;

                        -- Wait forever

                        wait;

                end process;
```