Arsal Abbas
Matriculation No.: 38707

VHDL Laboratory Report

MOVING LIGHT

Course Instructor

Prof. Dr.-Ing. Kai Müller

Contents

1. Introduction:

The objective of this lab is to develop a VHDL program that can move or rotates the series of 8 LEDs in the left and right direction for a given pattern defined by the user through toggle switches. LEDs value is loading from Switches and also holding the previous input. Switches are connected to LEDs directly. Clock speed is needed to be reduced, so that physical blinking LEDs can be observed, to achieve this define a new counter which acts clock with reduced speed. All modes have to be verified by simulation.

2. Task Description:

The user defines the pattern by toggling any of the switches numbered from 0 to 7, connected parallel to their paired LEDs. This design also contains 4 push buttons to perform 4 different operations on these LEDs. Upon pressing btnd the pattern is loaded. If btnl is pressed the LEDs indicate the pattern moving to the left. If btnr is pressed the LEDs indicate the pattern moving to the right. Btnc is used to stop the rotation.

3. Algorithm Description:

3.1. Library:

- Using the "use" statement, all components of the package "STD_LOGIC_1164" part of library IEEE are visible for later use in the VHDL code. "Library" statement is included above the "use" statement so that compiler would know that "IEEE" is a library.
- VHDL datatype "STD_LOGIC" and rising_edge () is declared in IEEE.STD_LOGIC_1164.

3.2. Entity:

Entity declaration represents the external interface to the design entity. Entity statement declares the design name; here it is "movelight". This interface consists of the following input and output ports:

3.2.1. System Inputs:

- **clk** - system clock signal
- **switches** - 8 slide switches [will be a std_logic_vector (7 downto 0)]
- **btnd** - pushbutton (down): Load a new pattern from the switches
- **btnl** - pushbutton (left): rotate to the left
- **btnc** - pushbutton (center): stop rotation
- **btnr** - pushbutton (right): rotate to the right

3.2.2. System output:

- **leds** - 8 leds [will be a std_logic_vector (7 downto 0)]

3.3. Architecture:

- Architecture is a description of the inner design operation.
- The Architecture "Behavioral" is assigned to the entity "movelight" using the keyword "of", such that the port definitions are visible to the architecture.

- Constant named **"Max_count"** is defined and assigned a value of 3 ", **sub_type** of integer named **"Count_type"** is defined and assigned a range of **"0 to Max_count**-1".
- **Max_count** is used to define the speed of the **r_pulse**.
- 3 signals named **mv_left, mv_right,** and **r_pulse** are defined of the type standard logic, Signal **led_reg** is defined as a standard logic vector with 8 bits. Signal **led_reg** acts as a memory device (register) to load the pattern from the switches when the appropriate button is pressed **(btnd)** and outputted to the LEDs.
- Furthermore, 3 processes **named count_p, mvlogic,** and **lr_rot** are defined.
- All processes take **(clk)** as a sensitivity list because all processes are time-dependent there is a need that all processes to be synchronized.
- In the process **"count_p"** the objective is to generate a repeating pulse named **r_pulse**, this is done to set the speed of the rotating LEDs. The process count_p is defined such that it sets a HIGH bit for **r_pulse** for after every 3 clock cycles and the HIGH BIT lasts 1 clock cycle.
- In the process **"mvlogic"** on every clock cycle, three conditions check the state of **btnl, btnr** and **btnc**, if **btnl** == 1 then **mv_left** == 1, if **btnr** = 1 then **mv_right** = 1, if **btnc** is pressed then both **mv_left** and **mv_left** are assigned 0.
- The process **"lr_rot"** is defined to implement the left and right rotation.
  - **Left Rotation**: The led register is assigned by concatenating bits from 6 to 0 and MSB bit 7. So, at each iteration due to this assignment provided that mv_left = '1', the pattern is rotated to the left direction.

    led_reg <= led_reg(6 downto 0) & led_reg(7);

  - **Right Rotation:** The led register is assigned by concatenating LSB bit 0 and bits 7 to 1. So, at each iteration due to this assignment provided that mv_right = '1', the pattern is rotated to the right direction.

    led_reg <= led_reg(0) & led_reg(7 downto 1);

4. VHDL Coding:

4.1. Source Code:

```
----------------------------------------------------------------------------
-- Company:      Univ. Bremerhaven
-- Engineer:     Arsal Abbas
-- Create Date:  25-MAY-2021
-- Description:  movelight - Behavioral
----------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity movelight is
    PORT ( clk : in std_logic;
           btnd : in std_logic;          -- load pattern from switches
```

```
                btnl : in std_logic;                -- left rotation
                btnc : in std_logic;                -- stop rotation
                btnr : in std_logic;                -- right rotation
                switches : in std_logic_vector (7 downto 0);
                leds : out std_logic_vector (7 downto 0) );
end movelight;

architecture Behavioral of movelight is
CONSTANT MAX_COUNT : INTEGER := 3;
SUBTYPE Count_type IS INTEGER RANGE 0 to MAX_COUNT-1;

SIGNAL led_reg : std_logic_vector (7 downto 0);
SIGNAL mv_left : std_logic := '0';
SIGNAL mv_right : std_logic := '0';
SIGNAL r_pulse : std_logic;

begin
        leds <= led_reg;

        -- speed of rotation
        count_p: PROCESS(clk)
        VARIABLE cnt : Count_type := MAX_COUNT-1;
        BEGIN
                IF rising_edge(clk) THEN
                        r_pulse <= '0';
                        IF cnt = 0 THEN
                                r_pulse <= '1';
                                cnt := MAX_COUNT-1;
                        ELSE
                                cnt := cnt -1;
                        END IF;
                END IF;
        END PROCESS count_p;

        -- mode of operation (left - right -stop)
        mvlogic: PROCESS(clk)
        BEGIN
                IF rising_edge(clk) THEN
                        IF btnl='1' THEN
                                mv_left <= '1';
                                mv_right <= '0';
                        ELSIF btnr='1' THEN
                                mv_left <= '0';
                                mv_right <= '1';
                        ELSIF btnc='1' THEN
                                mv_left <= '0';
                                mv_right <= '0';
```

```
                END IF;
            END IF;
        END PROCESS mvlogic;

        -- push button storage and rotation handling
        lr_rot: PROCESS(clk)
        BEGIN
            IF rising_edge (clk) THEN
                IF r_pulse='1' THEN
                    IF btnd='1' THEN
                        led_reg <= switches;
                        ELSIF mv_left='1' THEN
                        led_reg <= led_reg(6 downto 0) & led_reg(7);
                        ELSIF mv_right='1' THEN
                        led_reg <= led_reg(0) & led_reg(7 downto 1);
                        END IF;
                    END IF;
                END IF;
        END PROCESS lr_rot;
end Behavioral;
```

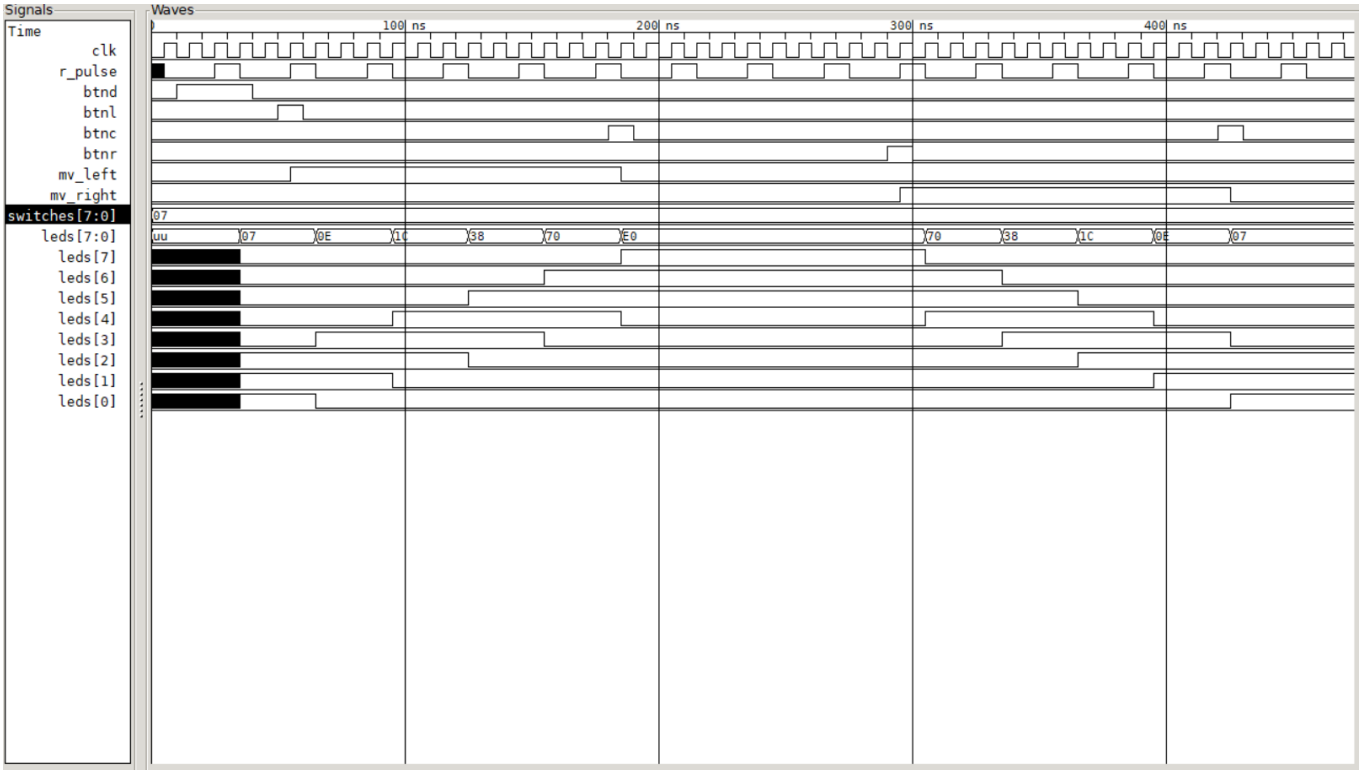
4.2. Test Bench:

```
----------------------------------------------------------------------
-- Company:        Univ. Bremerhaven
-- Engineer:       Arsal Abbas
-- Description:    movelight test bench
----------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity movelight_tb is
end movelight_tb;

architecture Behavioral of movelight_tb is
COMPONENT movelight IS
        PORT ( clk : in std_logic;
                btnd : in std_logic;            -- load pattern from switches
                btnl : in std_logic;    -- left rotation
                btnc : in std_logic;            -- stop rotation
                btnr : in std_logic;            -- right rotaticmdon
                switches : in std_logic_vector (7 downto 0);
                leds : out std_logic_vector (7 downto 0) );
END COMPONENT movelight;

SIGNAL clk : std_logic := '0';
```

```
SIGNAL btnd : std_logic := '0';
SIGNAL btnl : std_logic := '0';
SIGNAL btnc : std_logic := '0';
SIGNAL btnr : std_logic := '0';
SIGNAL switches : std_logic_vector (7 downto 0) := x"07";
SIGNAL leds : std_logic_vector (7 downto 0) := x"00";

CONSTANT clock_period : time := 10 ns;

begin

	uut : movelight
	PORT MAP ( clk => clk,
			btnd => btnd,
			btnl => btnl,
			btnc => btnc,
			btnr => btnr,
			switches => switches,
			leds => leds );

	clk_p: PROCESS
	BEGIN
		clk <= '0';
		wait for clock_period / 2;
		clk <= '1';
		wait for clock_period / 2;
	END PROCESS clk_p;

	stim_p: PROCESS
	BEGIN
		wait for clock_period;
		btnd <= '1';
		wait for clock_period * 3;
		btnd <= '0';
		wait for clock_period;
		btnl <= '1';
		wait for clock_period;
		btnl <= '0';
		wait for clock_period * 12;
		btnc <= '1';
		wait for clock_period;
		btnc <= '0';
		wait for clock_period * 10;
		btnr <= '1';
		wait for clock_period;
		btnr <= '0';
		wait for clock_period *12;
```

```
                btnc <= '1';
                wait for clock_period;
                btnc <= '0';
                wait for clock_period * 10;
                report "movelight simulation done.";
                wait;
        END PROCESS stim_p;
end Behavioral;
```

5. Simulation Results:



6. Conclusion:

- On pressing btnd the toggle switches pattern load into the LED's by the mean of led_reg register.
- Btnl the light moves towards the left (LSB to MSB).
- Similarly in the case of btnr the light move towards the right (MSB to LSB).
- Btnc stops the rotation action.
- R_pulse signal is used to define the speed of the rotation in this case on every 3 third rising edge of clock r-pulse value is HIGH, means LEDs move their position on every third rising edge of clock.