

TABLE OF CONTENTS

1 INTRODUCTION	Error! Bookmark not defined.
1.1 Objective.....	2
1.2 Finite State Machines	2
2 DESIGN PROCEDURE.....	2
2.1 Entity.....	2
2.2 Architecture	3
3 VHDL SOURCE CODE	3
4 TEST BENCH DESIGN	6
5 VHDL TEST BENCH SOURCE CODE.....	6
6 SIMULATION PROCEDURE.....	8
7 WAVEFORMS	8
7.1 Command Prompt Output.....	8
7.2 Waveform	9
8 CONCLUSIONS	9

1 INTRODUCTION

1.1 Objective

The Objective of this project is to design and simulate rotating LEDs state machine that performs left and right rotation operations of LEDs. The sequence and direction of rotation can be updated using push buttons. Design is functionally verified by simulation of VHDL program using GHDL and gtkwave.

1.2 Finite State Machines

Finite State Machine is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time. A finite state machine can change from one state to another in response to some inputs. The change from one state to another is called a transition. A finite state machine is defined by a list of its states, its initial state, and the inputs that trigger each transition.

The following VHDL program is written in jEdit and is executed using GHDL tool and waveforms are simulated using gtkwave.

2 DESIGN PROCEDURE

- Library IEEE and package STD_LOGIC_1164 has been included to access data types and functions in the package.

2.1 Entity

- Entity defines names, input, output signals and modes of hardware module.
- It describes the interface between the design entity to the external hardware.
- To implement moving light application, I have used 6 inputs and 1 output.
- The name of the entity here is 'movelight'.
- Inputs:

Name	Type	Description
clk	STD_LOGIC	A Clock Signal of 10ns
btnd	STD_LOGIC	A push button to load pattern from switches.
btnl	STD_LOGIC	A push button to rotate left
btnr	STD_LOGIC	A push button to rotate right
btnc	STD_LOGIC	A push button to stop rotation
switches	STD_LOGIC_VECTOR (7 DOWNT0 0)	A fixed load pattern for LEDs

- Output:

Name	Type	Description
leds	STD_LOGIC_VECTOR (7 DOWNT0 0)	LED light status

2.2 Architecture

- It describes the behavior of your inner design and operation.
- A Signal 'led_reg' of type STD_LOGIC_VECTOR and of size 8 is used to rotate the byte value and assign to output.
- A Signal 'pulse' of type STD_LOGIC is used to provide the time for rotation set by counter. The counter value is set with MAX_COUNT variable which is 3.
- A Signal 'mv_left' and 'mv_right' is used to assign values when left button(btnl) and right button(btnr) is pressed respectively.
- There are 3 processes defined to execute moving light operation.
 - i. **count_p**: This process is used to generate the 'pulse' for every 3 cycles of clk. At rising edge of clk and when pulse is '1', the LEDs move or stop as per given operation.
 - ii. **mv_logic**: This process is used to provide the logic for movement of LEDs by assigning the signals mv_left and mv_right with either '0' or '1' as below and deciding the direction of rotation or to stop the LED movement.
 - **Left Rotate**: btnl = '1'; mv_left = '1' and mv_right = '0'.
 - **Right Rotate**: btnr = '1'; mv_left = '0' and mv_right = '1'.
 - **Stop Rotation**: btnc = '1'; mv_left = '0' and mv_right = '0'.
 - iii. **lr_rot**: This process is used to update the value of led_reg based on the values of signals mv_left and mv_right when pulse = '1'.
 - If btnd is pressed i.e., '1', the led movement sequence is updated with the input given by switches.
 - If mv_left is 1, then left rotate operation is performed by using concatenation of bits 6 to 0 with bit 7.
led_reg <= led_reg (6 downto 0) & led_reg (7).
 - If mv_right is 1, then right rotate operation is performed by using concatenation of bit 0 with bits 7 to 1.
led_reg <= led_reg (0) & led_reg (7 downto 1).

3 VHDL SOURCE CODE

```

LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;

--entity declarations--
ENTITY movelight IS
    PORT ( clk : IN STD_LOGIC ; --clock input
          btnl : IN STD_LOGIC ; --rotate to left
          btnr : IN STD_LOGIC ; --rotate to right
          btnc : IN STD_LOGIC ; --stop rotation
          btnd : IN STD_LOGIC ; --load pattern
          switches : IN STD_LOGIC_VECTOR (7 DOWNTO 0) ;
          leds : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
        );
END movelight ;
    
```

ARCHITECTURE Behavioral OF movelight IS

```
SIGNAL led_reg : STD_LOGIC_VECTOR(7 downto 0) := x"05";  
SIGNAL mv_left : STD_LOGIC := '0';  
SIGNAL mv_right : STD_LOGIC := '0';  
SIGNAL pulse : STD_LOGIC := '0';  
  
CONSTANT MAX_COUNT : INTEGER := 3;  
SUBTYPE Count_type IS INTEGER RANGE 0 TO MAX_COUNT-1;  
BEGIN  
    leds <= led_reg;  
    --generation of pulse--  
    -- determines the speed of rotation--  
    -- It is slowed down by factor of 3 from clock pulse to visualize rotation.  
    count_p : PROCESS(clk)  
    VARIABLE cnt : Count_type := MAX_COUNT-1;  
    BEGIN  
        IF rising_edge(clk) THEN  
            pulse <= '0';  
            IF cnt = 0 THEN  
                pulse <= '1';  
                cnt := MAX_COUNT-1;  
            ELSE  
                cnt := cnt - 1;  
            END IF;  
        END IF;  
    END PROCESS count_p;
```

```

--move logic--
--mode of operation [ left - right - stop ]
mvlogic : PROCESS(clk)
BEGIN
    IF rising_edge(clk) THEN
        IF pulse = '1' THEN
            IF btnl = '1' THEN --move left
                mv_left <= '1';
                mv_right <= '0';
            ELSIF btnr = '1' THEN --move right
                mv_right <= '1';
                mv_left <= '0';
            ELSIF btnc = '1' THEN -- stop rotation
                mv_left <= '0';
                mv_right <= '0';
            ELSE
                mv_left <= mv_left;
                mv_right <= mv_right;
            END IF ;
        END IF ;
    END IF ;
END PROCESS mvlogic ;

--rotate operation--
lr_rot : PROCESS (clk)
BEGIN
    IF rising_edge(clk) THEN
        IF pulse = '1' THEN
            IF btnd = '1' THEN -- switches
                led_reg <= switches ;
            ELSIF mv_left = '1' THEN -- rotate left
                led_reg <= led_reg(6 DOWNTO 0) & led_reg(7) ;
            ELSIF mv_right = '1' THEN -- rotate right
                led_reg <= led_reg(0) & led_reg(7 DOWNTO 1) ;
            END IF ;
        END IF ;
    END IF ;
END PROCESS lr_rot ;

END Behavioral;

```

4 TEST BENCH DESIGN

- A Test bench file is used to design VHDL code required to simulate a waveform.
- An entity 'movelights_tb' is defined for test bench code.
- Signal inputs in source code are set to '0' as default for clk, btnl, btnr, btnc and btnd.
- Signal input switches is set to hex value of "15" and signal output led is set to "05" as initial value.
- A constant 'clock_period' is set to 10 ns.
- Unit Under Test (uut) are defined for movelight source code.
- Two processes are defined for test bench file which provide all the test cases for the moving lights operation.
 - i. **clk_p**: This process is used to generate the clock pulses with '0' and '1' for half of clock_period time i.e., 5ns repeatedly until the stop time declared.
 - ii. **sim_p**: This is simulation process which is used to assign '0' and '1' values for btnl, btnr, btnc and btnd inputs to decide which operation is to be operated in order.

5 VHDL TEST BENCH SOURCE CODE

```

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 ; --clock input
          btnl : IN STD_LOGIC ; --rotate to left
          btnr : IN STD_LOGIC ; --rotate to right
          btnc : IN STD_LOGIC ; --stop rotation
          btnd : IN STD_LOGIC ; --load pattern from switches
          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 btnl : STD_LOGIC := '0' ;
SIGNAL btnr : STD_LOGIC := '0' ;
SIGNAL btnc : STD_LOGIC := '0' ;
SIGNAL btnd : STD_LOGIC := '0' ;
SIGNAL switches : STD_LOGIC_VECTOR(7 DOWNTO 0) := x"15" ;
SIGNAL leds : STD_LOGIC_VECTOR(7 DOWNTO 0) := x"05" ;
    
```

```

CONSTANT clock_period : TIME := 10 ns ;
BEGIN
    uut : movelight
    PORT MAP (      clk => clk ,
                    btnl => btnl ,
                    btnr => btnr ,
                    btnc => btnc ,
                    btnd => btnd ,
                    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;

    sim_p: PROCESS
    BEGIN
        WAIT FOR clock_period * 15 ;
        btnd <= '1';
        WAIT FOR clock_period * 15 ;
        btnd <= '0';
        WAIT FOR clock_period * 15 ;
        btnl <= '1';
        WAIT FOR clock_period * 15 ;
        btnl <= '0';

        WAIT FOR clock_period * 15 ;
        btnr <= '1';
        WAIT FOR clock_period * 15 ;
        btnr <= '0';
        WAIT FOR clock_period * 15 ;
        btnc <= '1';
        WAIT FOR clock_period * 15 ;
        btnc <= '0';
        switches<= x"F0";
        WAIT FOR clock_period ;
        btnd <= '1';
        WAIT FOR clock_period ;
        REPORT " movelight simulation done. " ;
        WAIT;
    END PROCESS sim_p;
END Behavioral;

```

6 SIMULATION PROCEDURE

Step 1: This simulation is done for a stop time of 1500ns.

Step 2: Initially leds will have the hex value “05” as given in test bench code.

Step 3: At 150ns, btnd is made ‘1’ and hence led is loaded from switches when pulse = ‘1’. switches pattern is initialized with hex value “15” from test bench code.

Step 4: At 450ns, btnl is made ‘1’, So mv_left = ‘1’ and hence leds starts rotating left when pulse = ‘1’. Left rotated hex value of leds:

15->2A->54->A8->51->A2->45->8A->15->2A->54

Step 5: At 750ns, btrr is made ‘1’, So mv_right = ‘1’ and hence leds starts rotating right when pulse = ‘1’. Right rotated hex value of leds:

54->2A->15->8A->45->A2->51->A8->54->2A->15

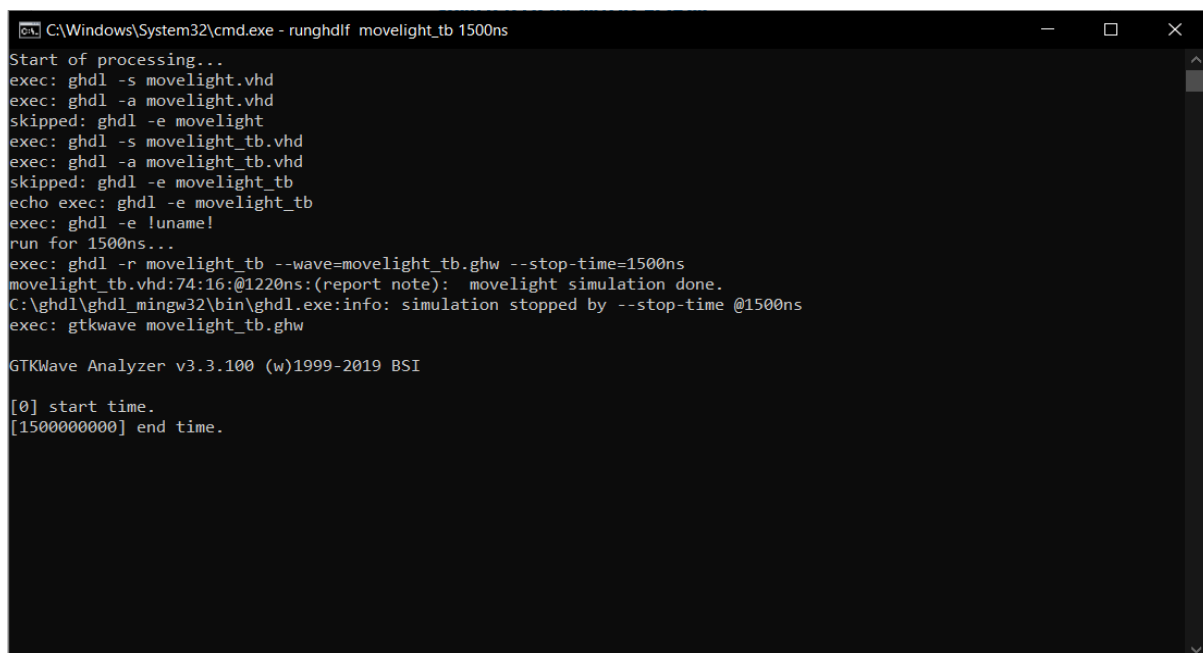
Step 6: At 1050ns, btnc is made ‘1’, So rotation stops and hex value “15” is loaded onto leds when pulse = ‘1’.

Step 7: Now switches pattern is loaded with new hex value “F0”.

Step 8: At 1210ns, btnd is made ‘1’ again and hence led is loaded from switches (i.e., hex value “F0”) when pulse = ‘1’.

7 WAVEFORMS

7.1 Command Prompt Output



```
C:\Windows\System32\cmd.exe - runghdlf movelight_tb 1500ns
Start of processing...
exec: ghdl -s movelight.vhd
exec: ghdl -a movelight.vhd
skipped: ghdl -e movelight
exec: ghdl -s movelight_tb.vhd
exec: ghdl -a movelight_tb.vhd
skipped: ghdl -e movelight_tb
echo exec: ghdl -e movelight_tb
exec: ghdl -e !uname!
run for 1500ns...
exec: ghdl -r movelight_tb --wave=movelight_tb.ghw --stop-time=1500ns
movelight_tb.vhd:74:16:@1220ns:(report note): movelight simulation done.
C:\ghdl\ghdl_mingw32\bin\ghdl.exe:info: simulation stopped by --stop-time @1500ns
exec: gtkwave movelight_tb.ghw

GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

[0] start time.
[1500000000] end time.
```

Figure 1.1 Command Prompt Output

7.2 Waveform

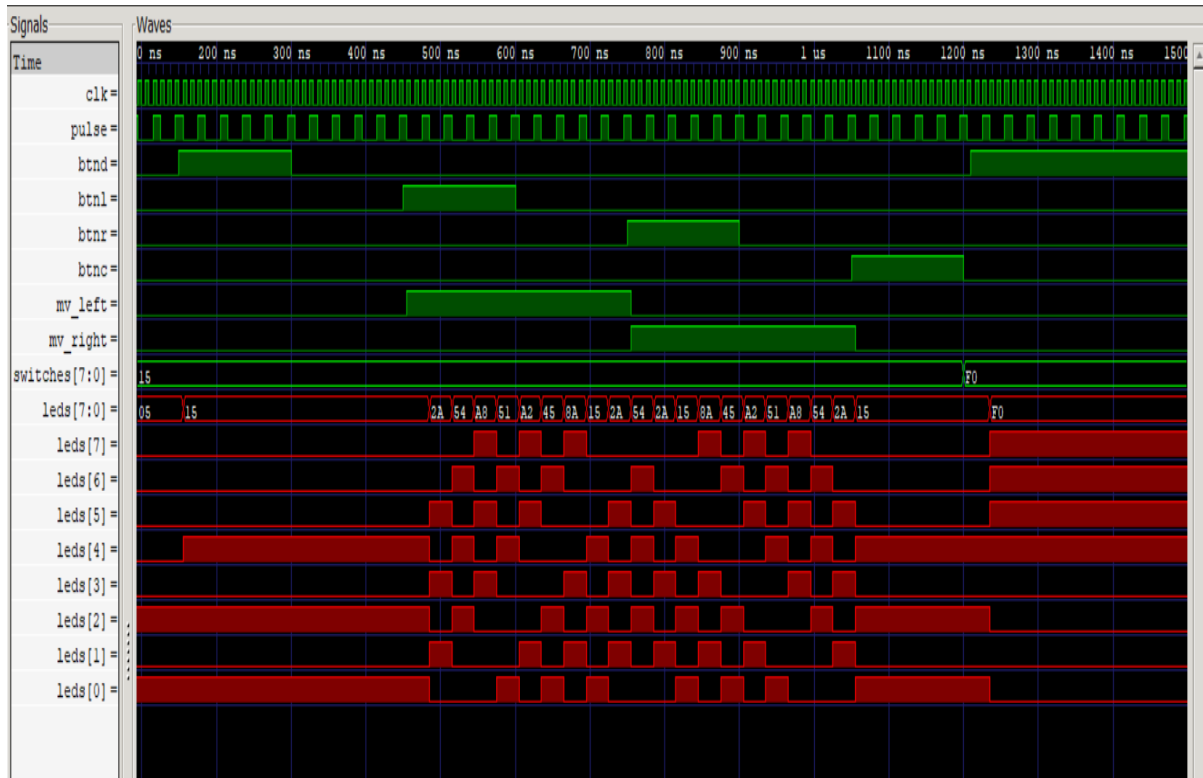


Figure 1.2 Simulation Waveform

8 CONCLUSIONS

I have implemented a Finite State Machine with Moving light LED and observed the simulation waveform for movement of LEDs. The testbench file was written and the waveform was simulated. The waveform generated was verified as per the testbench file and given operation.

The following test cases were performed:

- When btnl is '1', Left Rotate operation of LEDs was performed.
- When btnr is '1', Right Rotate operation of LEDs was performed.
- When btnd is '1', Switches Pattern were loaded to LEDs.
- When btnc is '1', The LED movement was stopped.