LAB REPORT

Digital Systems / VHDL [ET-DTV / ESD1]

Traffic Light Controller as Timed State Machine

Date of Execution: 15/05/2019

Group: DTVL – 3

Report By:

Harsha Priya Yapamakula Srinivasa Murthy (37091)

Rama Krishna Reddy Salla (37089)

Under the Guidance of:

Prof. Dr.-Ing. Kai Mueller

Date of Report Submission

05/06/2019

Master of Science in Embedded Systems Design

Hochschule Bremerhaven

June-2019

TABLE OF CONTENTS

INTRODUCTION

AIM: The aim of this program is to implement a simple traffic light controller using a timed state machine. In this system, when the Traffic Light controller is in states red or green, we need the system to stay in these states for a desired amount of time. This is because in the physical world the red and green lights remain for longer durations.

ABSTRACT: In a Finite State Machine only one state can be active at a time. States can be changed whenever there is a trigger or a condition is satisfied. This is called as a state transition. The main feature of a Timed State Machine is that it can stay in a desired state for desired amount of time. The timer value is decided by the user for the particular state.

This program is implemented on an **Arty** board. The program is developed and executed on **Xilinx Vivado 2018.3**.

FEATURES:

Two operations are assigned to a switch and pushbutton on the board.

1. **tstart :** This is assigned to a switch and if the value is high, then the state machine will continue indefinitely without going to **idle** state.
2. **treset :** This is assigned to a push button and whenever this pushbutton is pressed, the state machine will reset to **idle** state.

The following are the states of the state machine and their actions:

1. **idle** **:** In this state, the system does nothing and all LEDs are OFF.
2. **st_red** **:** In this state, **red** LED is turned on and the system waits in this state for 5 pulses of the desired clock.
3. **st_ry** **:** In this state, both the **red** and **yellow** LEDs are turned on and the system waits in this state for 2 pulses of the desired clock.
4. **st_green** **:** In this state, **green** LED is turned on and the system waits in this state for 6 pulses of the desired clock.
5. **st_yel** **:** In this state, the **yellow** LED is turned on and the system waits in this state for 2 pulses of the desired clock.

The Traffic Light Controller state machine operates as follows:

1. After downloading the bitstream onto the board, the state machine is in **idle** state.
2. Whenever the switch is pushed to high position, **tstart** will become **1**.
3. This will trigger the state machine to run and the state transitions are as follows:
   st_red -> st_ry -> st_green -> st_yel -> st_red
4. If during the transitions the switch is pushed to low position, **tstart** will become **0**. This will lead the state machine to reach **idle** state after **st_yel**.

5. During any of the transitions if reset button is pushed, the state will switch to **idle** in the next rising edge of system clock.

NOTE: The 100MHz clock is used to produce a pulse of 4Hz(250ms), **'pulse_250ms'**. This is done by producing a single pulse for every 25000000th rising edge of **tclk.**

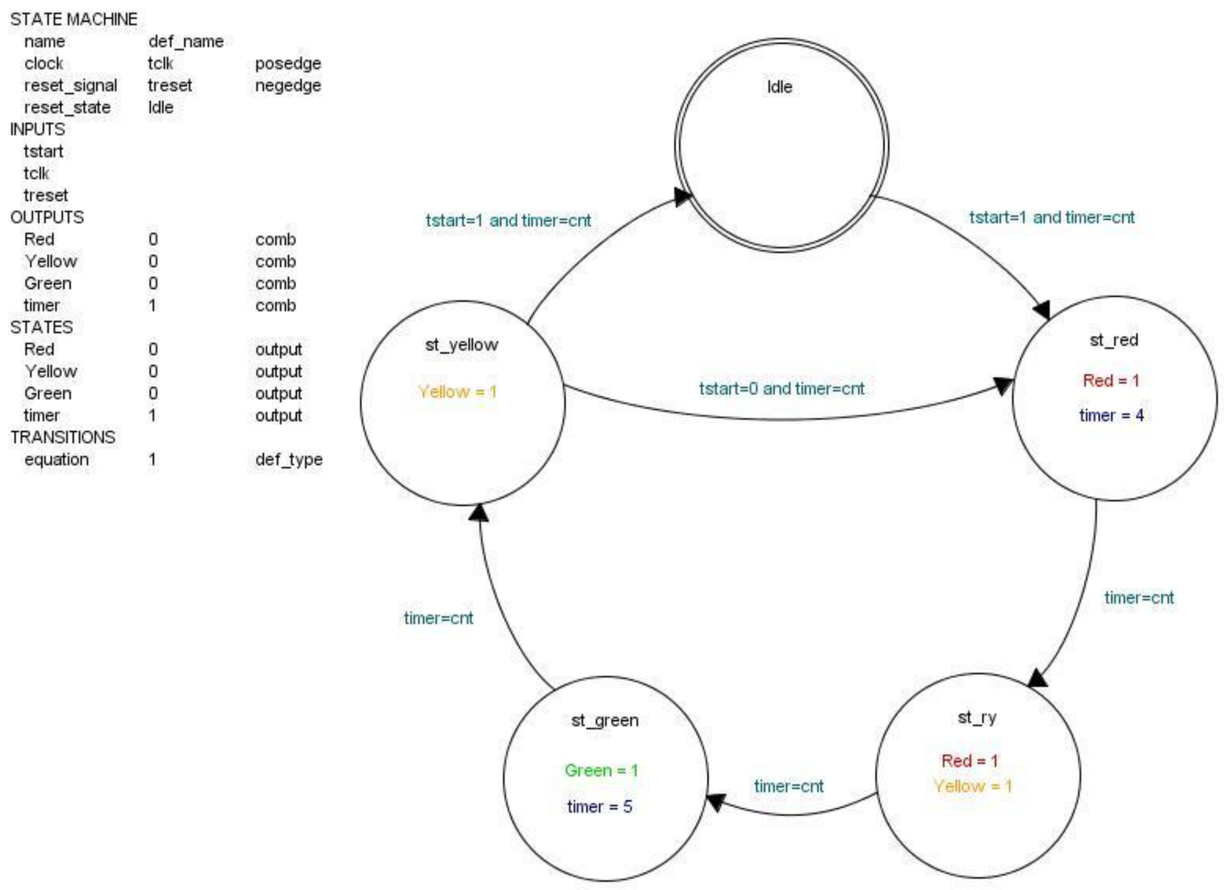

```
STATE MACHINE
  name          def_name
  clock         tclk         posedge
  reset_signal  treset       negedge
  reset_state   Idle
INPUTS
  tstart
  tclk
  treset
OUTPUTS
  Red           0            comb
  Yellow        0            comb
  Green         0            comb
  timer         1            comb
STATES
  Red           0            output
  Yellow        0            output
  Green         0            output
  timer         1            output
TRANSITIONS
  equation      1            def_type
```

Fig 1 : State Diagram of Traffic Controller

VHDL SOURCE CODE

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tlctsm is
    Port ( treset : in STD_LOGIC;
           tstart : in STD_LOGIC;
           tclk : in STD_LOGIC;
           red3 : out STD_LOGIC;
           red2 : out STD_LOGIC;
           red1 : out STD_LOGIC;
           green3 : out STD_LOGIC;
           green2 : out STD_LOGIC;
           green1 : out STD_LOGIC);
end tlctsm;

architecture Behavioral of tlctsm is

CONSTANT MAX_TIMERV : integer := 7;
SUBTYPE Counter_type IS integer RANGE 0 to MAX_TIMERV;
TYPE State_type IS (idle, st_red, st_ry, st_green, st_yel);
SIGNAL state, next_state : State_type;
SIGNAL timerval : Counter_type;
SIGNAL red, yellow, green: STD_LOGIC;

--For simulator
CONSTANT MAXCOUNT : integer := 3;
--For hardware test
--CONSTANT MAXCOUNT : integer := 24999999;
SIGNAL pulse_250ms : std_logic;

begin

    clk_div : PROCESS(tclk)
    VARIABLE k : integer RANGE 0 to MAXCOUNT := MAXCOUNT;
    BEGIN
        IF rising_edge(tclk) THEN
            pulse_250ms <= '0';
            IF k=0 THEN
                pulse_250ms <= '1';
                k := MAXCOUNT;
            ELSE
                k := k-1;
            END IF;
        END IF;
    END PROCESS clk_div;

    red3   <= red;
    green3 <= '0';
    red2   <= yellow;
    green2 <= yellow;
    red1   <= '0';
    green1 <= green;

    seq_tsm : PROCESS(treset, tclk, timerval, next_state, pulse_250ms)
    VARIABLE cnt : Counter_type;
    BEGIN
        IF rising_edge(tclk) THEN
            IF treset='1' THEN
                state <= idle;
                cnt := 0;
            ELSIF pulse_250ms = '1' THEN
                IF cnt = timerval THEN
                    state <= next_state;
                cnt := 0;
                ELSE
                    cnt := cnt+1;
                END IF;
            ELSE
            END IF;
        END IF;
    END PROCESS seq_tsm;

    comb_tsm : PROCESS(state, tstart)
    BEGIN
        next_state <= state;
        red <= '0'; yellow <= '0'; green <= '0';
        timerval <= 1;
        CASE state IS
            WHEN idle =>
                timerval <= 0;
                IF tstart = '1' THEN
                    next_state <= st_red;
                END IF;
            WHEN st_red=>
                timerval <= 4;
                red <= '1';
                next_state <= st_ry;
            WHEN st_ry=>
                red <= '1';
                yellow <= '1';
                next_state <= st_green;
            WHEN st_green=>
                timerval <= 5;
                green <= '1';
                next_state <= st_yel;
            WHEN st_yel=>
                yellow <= '1';
                IF tstart = '1'THEN
                    next_state <= st_red;
                ELSE
                    next_state <= idle;
                END IF;
         END CASE;
    END PROCESS comb_tsm;


end Behavioral;
```

EXPLANATION OF SOURCE CODE

- Library **IEEE** and package **STD_LOGIC_1164** has been added to access data types and functions in the package.
- Define an entity **tlctsm** that has 3 inputs and 6 outputs.
- The inputs are :
  1. **treset** : reset of type **STD_LOGIC**
  2. **tstart** : start indicator of type **STD_LOGIC**
  3. **tclk** : Clock of type **STD_LOGIC**
- The ouputs are :
  1. **red3** : Red component of led3 of type **STD_LOGIC**
  2. **red2** : Red component of led2 of type **STD_LOGIC**
  3. **red1** : Red component of led1 of type **STD_LOGIC**
  4. **green3** : Green component of led3 of type **STD_LOGIC**
  5. **green2** : Green component of led2 of type **STD_LOGIC**
  6. **green1** : Green component of led1 of type **STD_LOGIC**

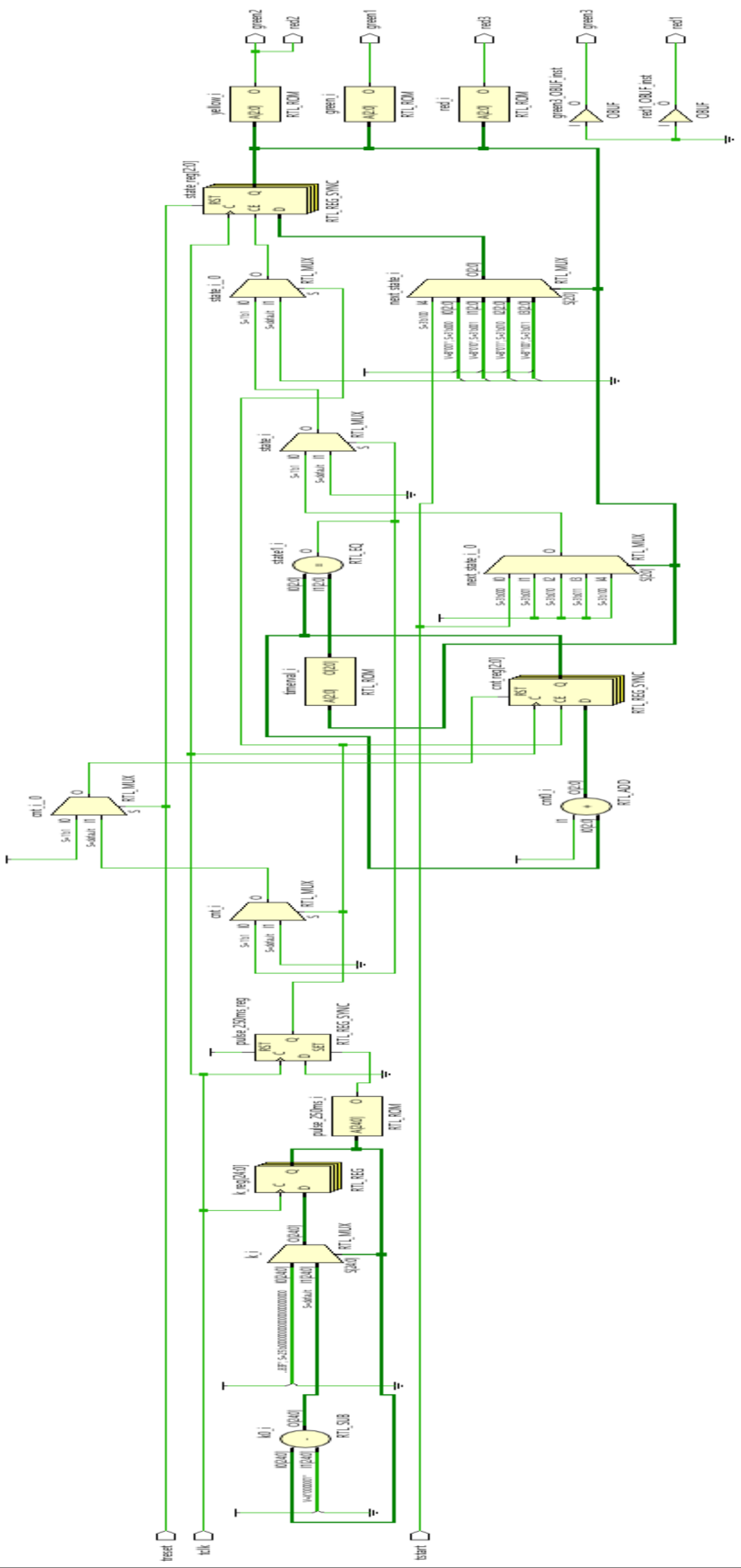
  NOTE: In the Arty board, there are no yellow LEDs. But the colour yellow is produced by mixing red and green components of led2. Led3 is used for Red and led1 is used for Green.
- A Subtype called **Counter_type** is defined to use for all counter/timer operations
- A Type **State_type** is defined and the states are **idle, st_red, st_ry, st_green, st_yel**
- Two Signals **state, next_state** of Type **State_type** are defined.
- Signals **red, yellow, green** are defined so as to assign the **ON/OFF** status of these lights.
- A Signal **pulse_250ms** is defined to produce a pulse that divides the tclk to desired value.
- Three Processes are used in the code:
  1. **clk_div:** This process is used to generate Pulses of 4Hz instead of using a clock of 100MHz. By producing pulses of 4Hz, the transitions will be slowed down to be visible to naked eye. In the current scenario, one pulse is generated for every 25000000 clock cycles. Signal **pulse_250ms** represents this pulse.
  2. **seq_tsm:** This process is used to execute the sequential logic of the timed state machine. If there is a button press for reset, the state changes to **idle**. If the timer value that is set in **comb_tsm** expires, the value of **next_state** is assigned to **state** so the transition takes place.
  3. **comb_tsm:** This process is used to update the parameters when in different states. The parameters are updated as follows:
     - **idle:**
       - **next_state** is set to **st_red** if **tstart** is 1
       - **timerval** is set to 0
       - **red** is set to 0
       - **yellow** is set to 0
       - **green** is set to 0

- **st_red:**
  - **next_state** is set to **st_ry**
  - **timerval** is set to 4
  - **red** is set to 1
  - **yellow** is set to 0
  - **green** is set to 0
- **st_ry**
  - **next_state** is set to **st_green**
  - **timerval** is set to 1
  - **red** is set to 1
  - **yellow** is set to 1
  - **green** is set to 0
- **st_green**
  - **next_state** is set to **st_green**
  - **timerval** is set to 5
  - **red** is set to 0
  - **yellow** is set to 0
  - **green** is set to 1
- **st_yel**
  - **next_state** is set to **st_red** if **tstart** is 1 or set to **idle** if **tstart** is 0
  - **timerval** is set to 0
  - **red** is set to 0
  - **yellow** is set to 1
  - **green** is set to 0

USER CONSTRAINTS FILE

```
## Clock signal
set_property -dict { PACKAGE_PIN E3    IOSTANDARD LVCMOS33 } [get_ports { tclk }]; #IO_L12P_T1_MRCC_35 Sch=gclk[100]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports { tclk }];

## Switches
set_property -dict { PACKAGE_PIN A8    IOSTANDARD LVCMOS33 } [get_ports { tstart }]; #IO_L12N_T1_MRCC_16 Sch=sw[0]

## Buttons
set_property -dict { PACKAGE_PIN D9    IOSTANDARD LVCMOS33 } [get_ports { treset }]; #IO_L6N_T0_VREF_16 Sch=btn[0]

## RGB LEDs
set_property -dict { PACKAGE_PIN J4    IOSTANDARD LVCMOS33 } [get_ports { green1 }]; #IO_L21P_T3_DQS_35 Sch=led1_g
set_property -dict { PACKAGE_PIN G3    IOSTANDARD LVCMOS33 } [get_ports { red1 }]; #IO_L20N_T3_35 Sch=led1_r
set_property -dict { PACKAGE_PIN J2    IOSTANDARD LVCMOS33 } [get_ports { green2 }]; #IO_L22N_T3_35 Sch=led2_g
set_property -dict { PACKAGE_PIN J3    IOSTANDARD LVCMOS33 } [get_ports { red2 }]; #IO_L22P_T3_35 Sch=led2_r
set_property -dict { PACKAGE_PIN H6    IOSTANDARD LVCMOS33 } [get_ports { green3 }]; #IO_L24P_T3_35 Sch=led3_g
set_property -dict { PACKAGE_PIN K1    IOSTANDARD LVCMOS33 } [get_ports { red3 }]; #IO_L23N_T3_35 Sch=led3_r

## Voltage config
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
```

RTL SCHEMATIC

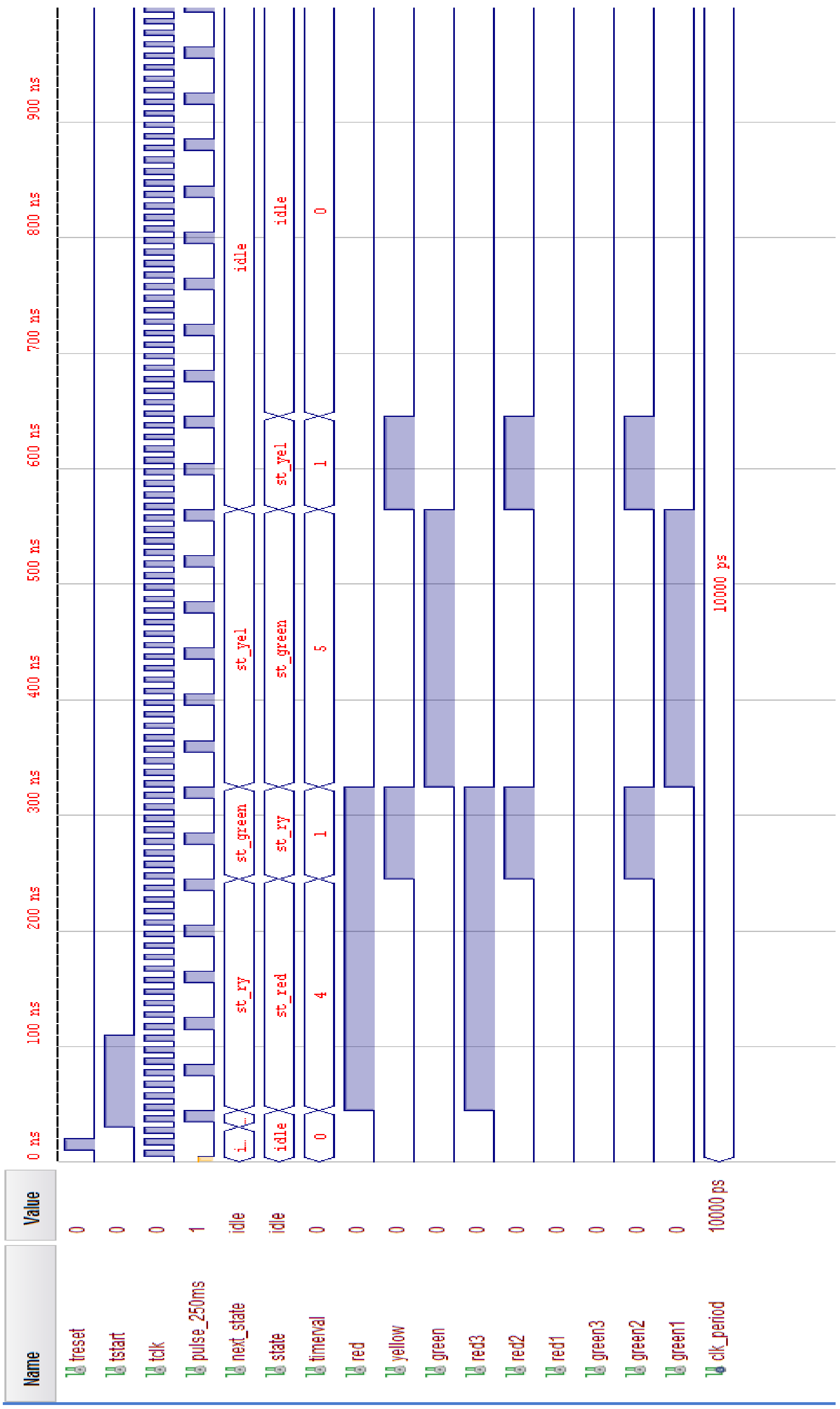


VHDL TESTBENCH

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tlctsm_tb is
end tlctsm_tb;

architecture Behavioral of tlctsm_tb is
COMPONENT tlctsm is
   Port ( treset : in STD_LOGIC;
          tstart : in STD_LOGIC;
          tclk : in STD_LOGIC;
          red3 : out STD_LOGIC;
          red2 : out STD_LOGIC;
          red1 : out STD_LOGIC;
          green3 : out STD_LOGIC;
          green2 : out STD_LOGIC;
          green1 : out STD_LOGIC);
end component;

SIGNAL treset :  STD_LOGIC := '0';
SIGNAL tstart :  STD_LOGIC:= '0';
SIGNAL tclk :  STD_LOGIC:= '0';
SIGNAL red3 :  STD_LOGIC:= '0';
SIGNAL red2 :  STD_LOGIC:= '0';
SIGNAL red1 :  STD_LOGIC:= '0';
SIGNAL green3 :  STD_LOGIC:= '0';
SIGNAL green2 :  STD_LOGIC:= '0';
SIGNAL green1 :  STD_LOGIC:= '0';

  CONSTANT clk_period : time := 10 ns;

begin

   uut : tlctsm
   Port map( treset =>treset,
           tstart =>tstart,
           tclk =>tclk,
           red3 =>red3,
           red2 =>red2,
           red1 =>red1,
           green3 =>green3,
           green2 =>green2,
           green1 =>green1);

          clk_p : PROCESS
          BEGIN
           tclk <= '0';
           wait for clk_period/2;
```

```
         tclk <= '1';
         wait for clk_period/2;
        END PROCESS clk_p;

        stim_p : PROCESS
        BEGIN
         wait for clk_period;
         treset <= '1';
         wait for clk_period;
         treset <= '0';
         wait for clk_period;
         tstart <= '1';
         wait for clk_period*8;
         tstart <= '0';
         wait;
        END PROCESS stim_p;

end Behavioral;
```

SIMULATION WAVEFORMS



SYNTHESIS REPORT

```
---------------------------------------------------------------------------------
Start Writing Synthesis Report
---------------------------------------------------------------------------------

Report BlackBoxes:
+-+--------------+----------+
| |BlackBox name |Instances |
+-+--------------+----------+
+-+--------------+----------+

Report Cell Usage:
+------+-------+------+
|      |Cell   |Count |
+------+-------+------+
|1     |BUFG   |     1|
|2     |CARRY4 |     6|
|3     |LUT1   |    25|
|4     |LUT2   |     3|
|5     |LUT3   |     3|
|6     |LUT4   |     4|
|7     |LUT5   |     5|
|8     |LUT6   |     3|
|9     |FDRE   |    16|
|10    |FDSE   |    18|
|11    |IBUF   |     3|
|12    |OBUF   |     6|
+------+-------+------+

Report Instance Areas:
+------+---------+-------+------+
|      |Instance |Module |Cells |
+------+---------+-------+------+
|1     |top      |       |    93|
+------+---------+-------+------+
---------------------------------------------------------------------------------
Finished Writing Synthesis Report : Time (s): cpu = 00:00:14 ; elapsed = 00:00:16 . Memory (MB): peak = 829.070 ; gain = 494.008
---------------------------------------------------------------------------------
Synthesis finished with 0 errors, 0 critical warnings and 2 warnings.
Synthesis Optimization Runtime : Time (s): cpu = 00:00:09 ; elapsed = 00:00:12 . Memory (MB): peak = 829.070 ; gain = 204.387
Synthesis Optimization Complete : Time (s): cpu = 00:00:14 ; elapsed = 00:00:16 . Memory (MB): peak = 829.070 ; gain = 494.008
INFO: [Project 1-571] Translating synthesized netlist
INFO: [Netlist 29-17] Analyzing 6 Unisim elements for replacement
INFO: [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
```

```
INFO: [Project 1-570] Preparing netlist for logic optimization
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 829.070 ; gain = 0.000
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

INFO: [Common 17-83] Releasing license: Synthesis
18 Infos, 4 Warnings, 0 Critical Warnings and 0 Errors encountered.
synth_design completed successfully
synth_design: Time (s): cpu = 00:00:16 ; elapsed = 00:00:18 . Memory (MB): peak = 829.070 ; gain = 506.590
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 829.070 ; gain = 0.000
WARNING: [Constraints 18-5210] No constraints selected for write.
Resolution: This message can indicate that there are no constraints for the design, or it can indicate that the used_in flags are set such that the constraints are ignored. This later case is used when running synth_design to not write synthesis constraints to the resulting checkpoint. Instead, project constraints are read when the synthesized design is opened.
INFO: [Common 17-1381] The checkpoint 'C:/mueller/DTV/DTVL3/tlctsm/tlctsm.runs/synth_1/tlctsm.dcp' has been generated.
INFO: [runtcl-4] Executing : report_utilization -file tlctsm_utilization_synth.rpt -pb tlctsm_utilization_synth.pb
INFO: [Common 17-206] Exiting Vivado at Wed May 15 14:23:15 2019...
```

CONCLUSION

In this lab, we implemented a Timed State Machine on Arty Board and observed Traffic Light Controller. This is implemented by sequential operation using PROCESS. SIGNALS were used to execute the state machine, implement a timer/counter and to divide the clock so we obtained a pulse of 4Hz.
A testbench file is written and the program is simulated. The correctness of the program is checked by observing the simulated waveform.
A User Constraints File is added and a bitstream is generated. This bitstream is programmed into the Arty Board and the Traffic Light Controller is observed.

The following operations were performed:

1. **tstart** switch is pushed to high after which the state transitions start from idle and continue in the following manner:

   idle->st_red -> st_ry -> st_green -> st_yel -> st_red-> st_ry -> st_green -> st_yel ->…

2. If **tstart** is pushed to low, the transitions would stop after **st_yel** and settle in **idle** state

   idle->st_red -> st_ry -> st_green -> st_yel -> idle

3. If during any time, the reset button is pushed, state will switch back to idle in next rising edge of system clock.

   NOTE: The changes in states are indicated by the LEDs that are **ON/OFF**. The LEDs that are on for respective states are mentioned in the **INTRODUCTION** chapter