TRAFFIC LIGHT CONTROLLER- Using Spartan 3E FPGA Board

PROJECT REPORT

Submitted for the course:

EEE4019: Advanced Digital Design with FPGAs

PRERNA SARKAR	16BEE0132
PAVAN KUMAR G.N.	16BEI0076

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2019

SCHOOL OF ELECTRICAL ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY

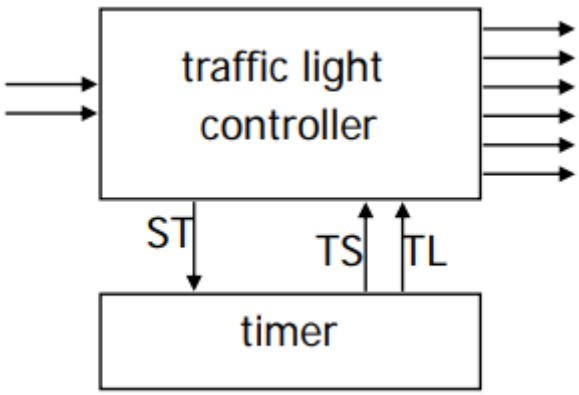
VELLORE 632014

OBJECTIVE: To implement a Traffic Light Controller for a junction using Xilinx software for Verilog coding and FPGA Board

ABSTRACT

Traffic light controller (TLC) is used to lessen or eliminate conflicts at area shared among multiple traffic streams called intersections; by controlling the access to the intersections and apportioning effective period of time between various users. The main goal of this project is to manage the traffic movement of four intersecting roads and to achieve optimum use of the traffic. In general, traffic lights of all main roads are controlled with a fix-time system while the smaller roads are controlled autonomously by sensors.

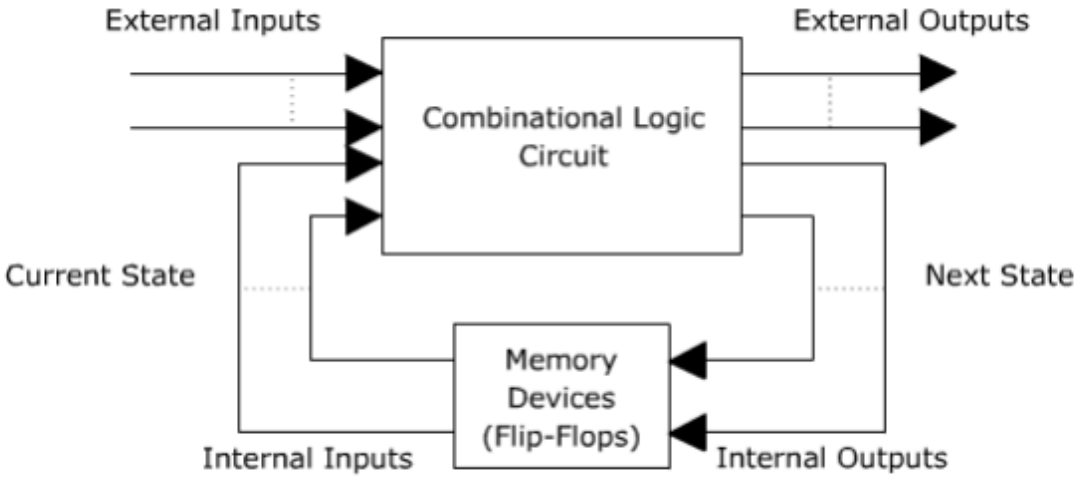
The Traffic Light Controller is designed to generate a sequence of digital data called switching sequences that can be used to control the traffic lights of a main road and cross road junction in a fixed sequence. It plays a very important role in modern management and control of urban traffic when it comes to reducing the accident and traffic jam on roads. It is a sequential machine which needs to be analysed and programmed through a multistep process. The device involves itself in an analysis of existing sequential machines in traffic lights controllers, timing, synchronization and overview of operation and flashing light synthesis sequence. The procedure that is used in this project is designing the circuit, writing a code, simulation, synthesis and implementation in hardware. In this project, XILINX Software has been chosen to write a code using Verilog HDL (Hardware Description Language) text editor and implementation is done using Spartan 3E FPGA Board



THEORY

Sequential Circuits:

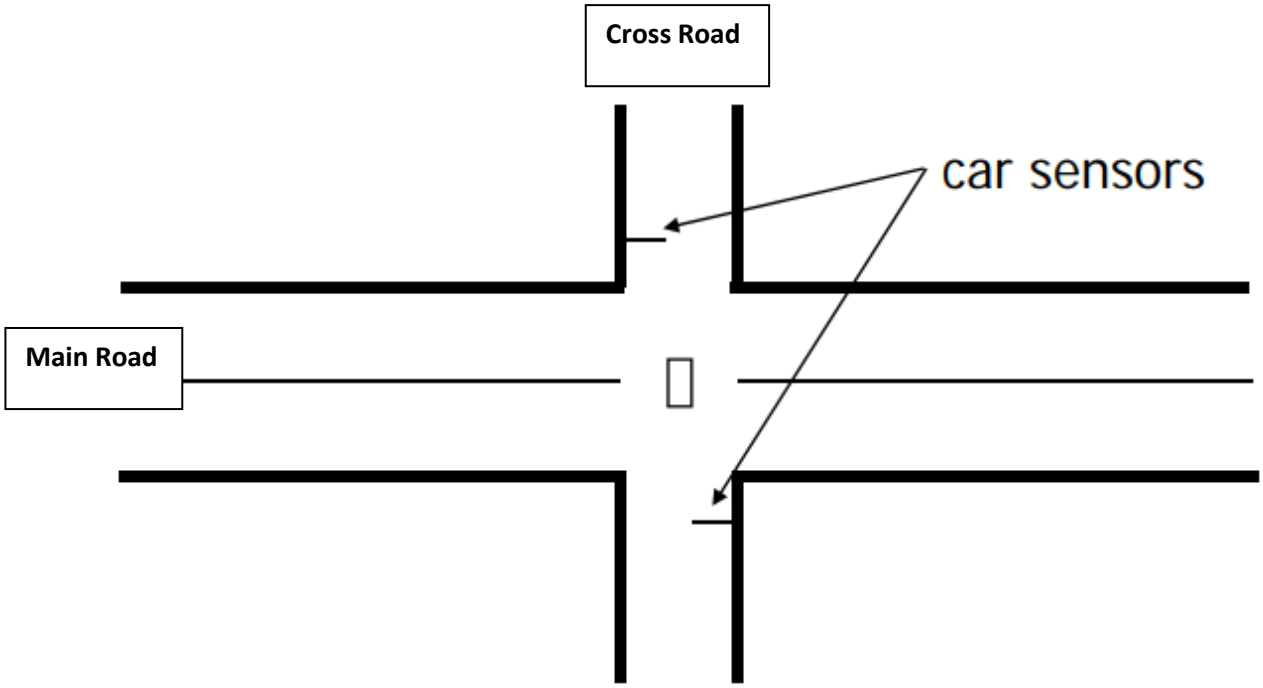
Sequential circuits work on a clock cycle which may be synchronous or asynchronous. The figure shows a basic diagram of sequential circuits. Sequential circuits use current inputs and previous inputs by storing the information and putting back into the circuit on the next clock cycle.



Finite State Machine (FSM):

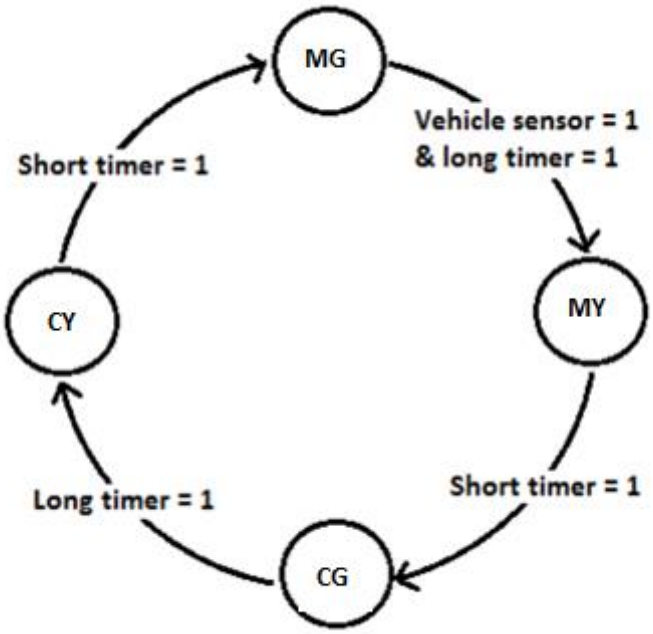
A FSM is a model used to design sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition, this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition. This can be shown with the help of a state diagram.

ALGORITHM AND STATE DIAGRAM



There are two timers a long timer for green signal and a short timer for yellow signal.

State Diagram:



State 1: In this state, Main Road signal is green, long timer is on and Cross road signal is red. Let this state be called “MG”. When the vehicle sensor is high and the long timer is out, it goes into state 2 and long timer is reset otherwise the system remains in this state.

State 2: In this state, Main Road signal is yellow, short timer is on and Cross Road signal is red. Let this state be called “MY”. When the short timer is out, it goes into state 3 and short timer is reset, otherwise the system remains in this state.

State 3: In this state, Cross Road signal is green, long timer is on and Main Road signal is red. Let this state be called “CG”. When the long timer is out, it goes into state 4 and long timer is reset otherwise the system remains in this state.

State 4: In this state, Cross Road signal is yellow, short timer is on and Main Road signal is red. Let this state be called “CY”. When the short timer is out, it goes into state 1 and short timer is reset, otherwise the system remains in this state.

State Table:

Inputs			**Present State**	**Next State**	**Outputs**		
C (sensor/detector)	**TL (long time pulse)**	**TS (short time pulse)**			**ST (set signal)**	**Main Road**	**Cross Road**
0	-	-	MG	MG	0	Green	Red
-	0	-	MG	MG	0	Green	Red
1	1	-	MG	MY	1	Green	Red
-	-	0	MY	MY	0	Yellow	Red
-	-	1	MY	CG	1	Yellow	Red
1	0	-	CG	CG	0	Red	Green
0	-	-	CG	CY	1	Red	Green
-	1	-	CG	CY	1	Red	Green
-	-	0	CY	CY	0	Red	Yellow
-	-	1	CY	MG	1	Red	Yellow

SA1:	MG = 00	MY = 01	CG = 11	CY = 10
SA2:	MG = 00	MY = 10	CG = 01	CY = 11
SA3:	MG = 0001	MY = 0010	CG = 0100	CY = 1000

Detectors C sense the presence of cars waiting on the cross road – with no car on cross road, light remains green in main road direction – if vehicle on cross road, main road lights go from Green to Yellow to Red, allowing the cross road lights to become green – these stay green only as long as a cross road car is detected but never longer than a set interval; after the interval expires, cross lights transition from Green to Yellow to Red, allowing main road to return to green – even if cross road vehicles are waiting, main road gets at least a set interval of green

Assume you have an interval timer that generates: – a short time pulse (TS) and – a long time pulse (TL), – in response to a set (ST) signal. – TS is to be used for timing yellow lights and TL for green lights

CODE

- **Main Code: traffic_light.v**

```
`timescale 1ns / 1ps
module traffic_light(light_main, light_cross, C, clk, rst_n, mr, my, mg, cr, cy, cg);
parameter MGRE_CRED=2'b00, // main green and cross red
  MYEL_CRED = 2'b01,// main yellow and cross red
  MRED_CGRE=2'b10,// main red and cross green
  MRED_CYEL=2'b11;// main red and cross yellow
input C, // sensor
  clk, // clock = 50 MHz
  rst_n; // reset active low
output reg[2:0] light_main, light_cross; // output of lights
output reg mr, my, mg, cr, cy, cg;

reg[27:0] count=0,count_delay=0;
reg delay10s=0,
delay3s1=0,delay3s2=0,RED_count_en=0,YELLOW_count_en1=0,YELLOW_count_en2=0;
wire clk_enable; // clock enable signal for 1s
reg[1:0] state, next_state;
// next state
always @(posedge clk or negedge rst_n)
begin
if(~rst_n)
 state <= 2'b00;
```

```
else
 state <= next_state;
end
// FSM
always @(*)
begin
case(state)
MGRE_CRED: begin // Green on main and red on cross way
 RED_count_en=0;
 YELLOW_count_en1=0;
 YELLOW_count_en2=0;
 light_main = 3'b001;
 light_cross = 3'b100;
 mr = 1'b0;
 my = 1'b0;
 mg = 1'b1;
 cr = 1'b1;
 cy = 1'b0;
 cg = 1'b0;
 if(C) next_state = MYEL_CRED;
 // if sensor detects vehicles on cross road,
 // turn main to yellow -> green
 else next_state =MGRE_CRED;
end
MYEL_CRED: begin// yellow on main and red on cross way
  light_main = 3'b010;
```

```
 light_cross = 3'b100;
 RED_count_en=0;
YELLOW_count_en1=1;
YELLOW_count_en2=0;
mr = 1'b0;
my = 1'b1;
mg = 1'b0;
cr = 1'b1;
cy = 1'b0;
cg = 1'b0;
 if(delay3s1) next_state = MRED_CGRE;
 // yellow for 3s, then red
 else next_state = MYEL_CRED;
end
MRED_CGRE: begin// red on main and green on cross way
 light_main = 3'b100;
 light_cross = 3'b001;
 RED_count_en=1;
 YELLOW_count_en1=0;
 YELLOW_count_en2=0;
 mr = 1'b1;
 my = 1'b0;
 mg = 1'b0;
 cr = 1'b0;
 cy = 1'b0;
 cg = 1'b1;
```

```
 if(delay10s) next_state = MRED_CYEL;
 // red in 10s then turn to yello -> green again for high way
 else next_state =MRED_CGRE;
end
MRED_CYEL:begin// red on main and yellow on cross way
 light_main = 3'b100;
 light_cross = 3'b010;
 RED_count_en=0;
 YELLOW_count_en1=0;
 YELLOW_count_en2=1;
  mr = 1'b1;
  my = 1'b0;
  mg = 1'b0;
  cr = 1'b0;
  cy = 1'b1;
  cg = 1'b0;
 if(delay3s2) next_state = MGRE_CRED;
 // turn green for main, red for cross road
 else next_state =MRED_CYEL;
end
default: next_state = MGRE_CRED;
endcase
end

// create red and yellow delay counts
always @(posedge clk)
```

```
begin
if(clk_enable==1) begin
 if(RED_count_en||YELLOW_count_en1||YELLOW_count_en2)
  count_delay <=count_delay + 1;
  if((count_delay == 9)&&RED_count_en)
  begin
   delay10s=1;
   delay3s1=0;
   delay3s2=0;
   count_delay<=0;
  end
  else if((count_delay == 2)&&YELLOW_count_en1)
  begin
   delay10s=0;
   delay3s1=1;
   delay3s2=0;
   count_delay<=0;
  end
  else if((count_delay == 2)&&YELLOW_count_en2)
  begin
   delay10s=0;
   delay3s1=0;
   delay3s2=1;
   count_delay<=0;
  end
  else
```

```
 begin
  delay10s=0;
  delay3s1=0;
  delay3s2=0;
 end
end
end
// create 1s clock enable
always @(posedge clk)
begin
 count <=count + 1;
 //if(count == 50000000) // 50,000,000 for 50 MHz clock running on real FPGA
 if(count == 3) // for testbench
  count <= 0;
end
 assign clk_enable = count==3 ? 1: 0; // 50,000,000 for 50MHz running on FPGA
endmodule
```

- **Test Bench: tb_traffic.v**

```
timescale 1ns / 1ps
//`timescale 10 ns/ 1 ps
`define DELAY 1
//`include "counter_define.h"
module tb_traffic;
parameter ENDTIME  = 400000;
//integer count, count1, a;
```

```
reg clk;
reg rst_n;
reg C;
wire [2:0] light_cross;
wire [2:0] light_main;

traffic_light tb(light_main, light_cross, C, clk, rst_n);

initial
 begin
 clk = 1'b0;
 rst_n = 1'b0;
 C = 1'b0;
 //count = 0;
 //count1=0;
 //a=0;
 end
initial
 begin
 main;
 end
task main;
 fork
 clock_gen;
 reset_gen;
 operation_flow;
```

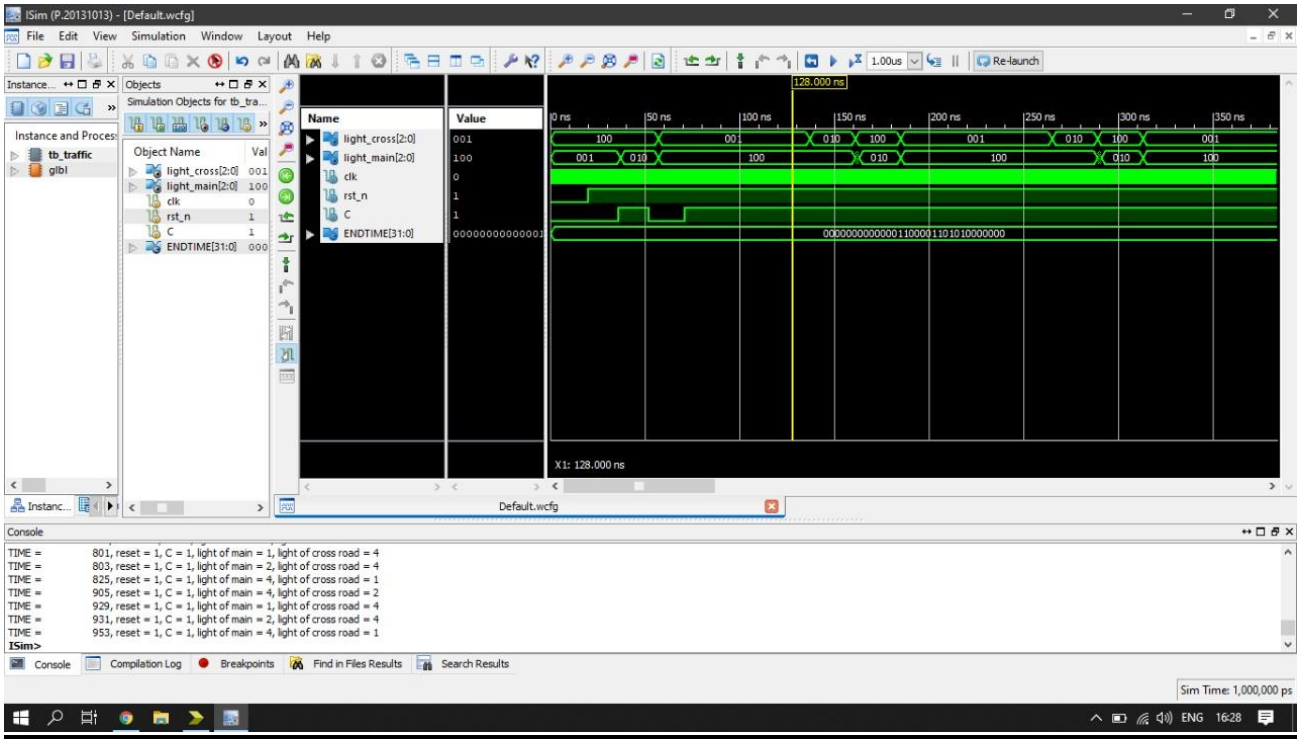
```
 debug_output;
 endsimulation;
 join
endtask
task clock_gen;
 begin
 forever #`DELAY clk = !clk;
 end
endtask

task reset_gen;
 begin
 rst_n = 0;
 # 20
 rst_n = 1;
 end
endtask

task operation_flow;
 begin
 # 19
 C = 0;
 # 17
 C = 1;
 # 16
 C = 0;
```

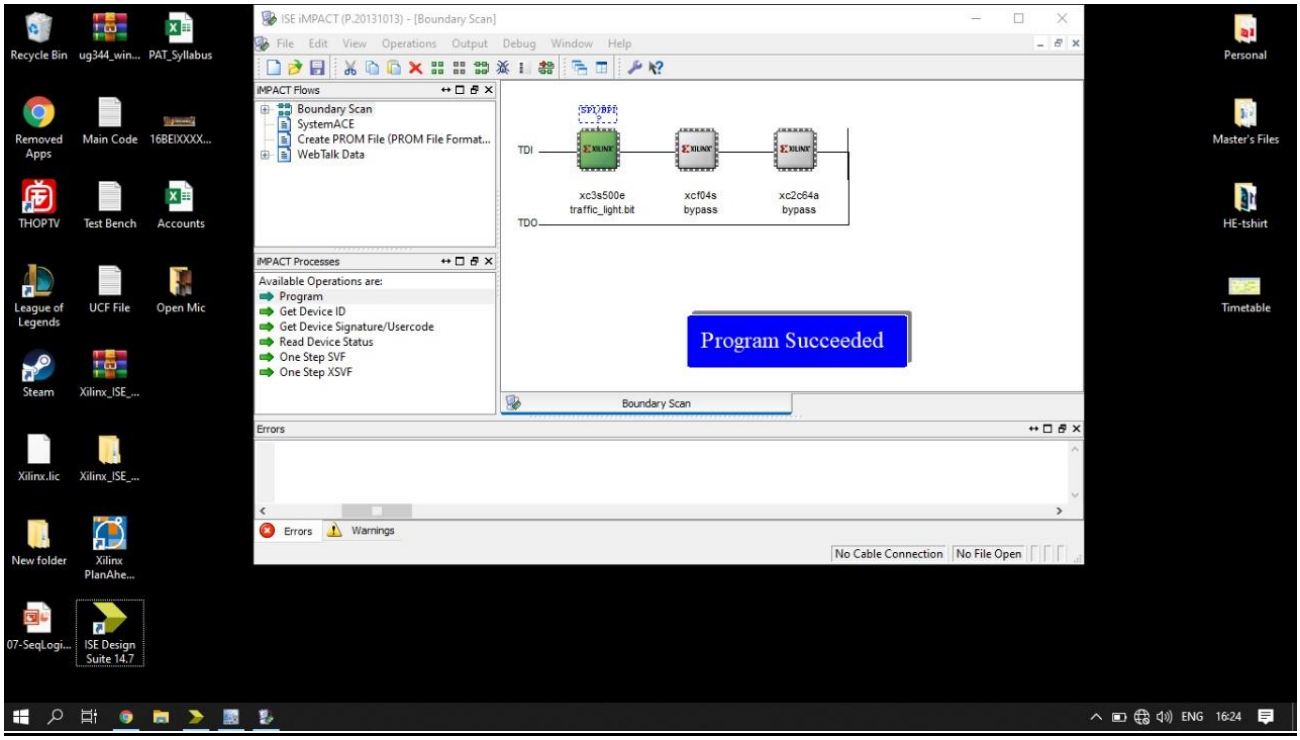
```
 # 19
 C = 1;
 end
endtask

task debug_output;
 begin
 $monitor("TIME = %d, reset = %b, C = %b, light of main = %h, light of cross road =
%h",$time,rst_n ,C,light_main,light_cross );
 end
endtask

task endsimulation;
 begin
 #ENDTIME
 $display("---------- THE SIMUALTION END ----------");
 $finish;
 end
endtask

endmodule
```

- **UCF File: traffic_light.ucf**

```
# clock signal for 50 MHz

NET "clk" LOC = C9 | IOSTANDARD = LVCMOS33;

NET "clk" PERIOD = 20.0ns HIGH 50%;


#input for switch

NET "C" LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;


# output for LED

 NET "mr" LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // Main road red

 NET "my" LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // main road yellow

 NET "mg" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // main road green

 NET "cr" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // cross road red

 NET "cy" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // cross road yellow

 NET "cg" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ; // cross road green
```

OUTPUT


ISim (P.20131013) - [Default.wcfg]
File Edit View Simulation Window Layout Help
Re-launch
Name
Value
light_cross[2:0]
light_main[2:0]
clk
rst_n
C
ENDTIME[31:0]
128.000 ns
X1: 128.000 ns
Default.wcfg
Console
TIME = 801, reset = 1, C = 1, light of main = 1, light of cross road = 4
TIME = 803, reset = 1, C = 1, light of main = 2, light of cross road = 4
TIME = 825, reset = 1, C = 1, light of main = 4, light of cross road = 1
TIME = 905, reset = 1, C = 1, light of main = 4, light of cross road = 2
TIME = 929, reset = 1, C = 1, light of main = 1, light of cross road = 4
TIME = 931, reset = 1, C = 1, light of main = 2, light of cross road = 4
TIME = 953, reset = 1, C = 1, light of main = 4, light of cross road = 1
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps


ISE iMPACT (P.20131013) - [Boundary Scan]
File Edit View Operations Output Debug Window Help
iMPACT Flows
Boundary Scan
SystemACE
Create PROM File (PROM File Format...
WebTalk Data
TDI
TDO
xc3s500e
traffic_light.bit
xcf04s
bypass
xc2c64a
bypass
iMPACT Processes
Available Operations are:
Program
Get Device ID
Get Device Signature/Usercode
Read Device Status
One Step SVF
One Step XSVF
Program Succeeded
Boundary Scan
Errors
Errors
Warnings
No Cable Connection
No File Open
Recycle Bin
ug344_win...
PAT_Syllabus
Removed Apps
Main Code
16BEIXXXX...
THOPTV
Test Bench
Accounts
League of Legends
UCF File
Open Mic
Steam
Xilinx_ISE_...
Xilinx.lic
Xilinx_ISE_...
New folder
Xilinx PlanAhe...
07-SeqLogi...
ISE Design Suite 14.7
Personal
Master's Files
HE-tshirt
Timetable

REFERENCES

- El-Medany, W. M., & Hussain, M. R. (2007, September). FPGA-based advanced real traffic light controller system design. In *2007 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications* (pp. 100-105). IEEE.
- Chavan, S. S., Deshpande, R. S., & Rana, J. G. (2009, December). Design of intelligent traffic light controller using embedded system. In *2009 Second International Conference on Emerging Trends in Engineering & Technology* (pp. 1086-1091). IEEE.
- Han, T., & Lin, C. (2002, October). Design of an intelligence traffic light controller (ITLC) with VHDL. In *2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. TENCOM'02. Proceedings.* (Vol. 3, pp. 1749-1752). IEEE.