UNIVERSITI KUALA LUMPUR
MALAYSIAN SPANISH INSTITUTE

SAB36603
ADVANCE DIGITAL DESIGN AND FPGA

MINI PROJECT
TRAFFIC LIGHT CONTROLLER

PREPARED BY:

STUDENT'S NAME	ID NUMBER
MUHAMMAD ADIEL ADAM BIN MOHD NOOR	54222121212
KHAIRUL ANWAR BIN KHAIRUL SALLEH	54215121186

LECTURER'S NAME: ENCIK MOHD REZAL BIN MOHAMED

1.1 Introduction

Traffic congestion and intersection accidents are common problems in both urban and rural settings. These issues are caused by the large number of cars merging at intersections, combined with the variable speeds and directions of traffic flow. Inadequate management of these crossings can result in extended wait times, greater fuel usage, high levels of air pollution, and, most importantly, an increased risk of accidents. Effective traffic signal management systems can help to reduce these concerns by guaranteeing orderly traffic flow, reducing wait times, and lowering the danger of collisions.

One of the key goals for this project is to investigate the capabilities of digital design with Verilog to create a practical, responsive, and efficient traffic light controller. Verilog, a hardware description language (HDL), is commonly used in the creation of digital systems, particularly for FPGA (Field Programmable Gate Array) and ASIC (Application-Specific Integrated Circuit) implementations. Using Verilog, this project attempts to construct a traffic signal controller that can adjust to changing traffic circumstances in real time, optimizing traffic flow and improving intersection safety.

This project also serves as an educational tool, illustrating the actual use of HDLs in solving real-world challenges. It demonstrates how digital design concepts may be used to build effective solutions to daily difficulties. This project provides learners with vital insights into the design, development, and testing of digital systems, preparing them for more complicated projects in the future.

2.1 Objectives

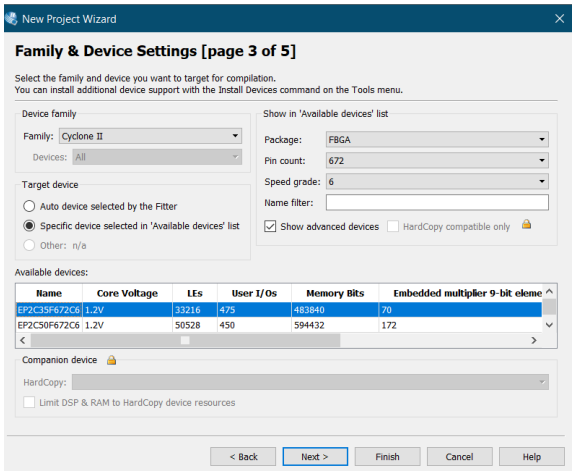
The objectives of this project are as follows:

- Create a traffic light controller using Verilog that effectively manages the flow of traffic at an intersection between a highway and a farm road.
- Develop accurate timing controls to manage the duration of each traffic light state using counters.
- Ensure that the traffic light controller transitions correctly between states (green, yellow, red) for both the highway and farm road.

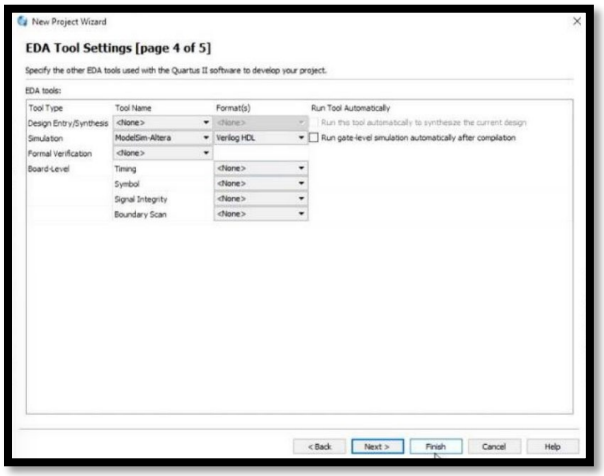
3.1 Procedure

3.1.1 Creating New Project Window.

- Create a new project with Quartus Software's New Project Wizard. Then, name the project and top-level design entity as traffic lights.
- On page 3 of the **New Project Wizard**, set the FPGA as **Cyclone ii EP2C35F672C6**.

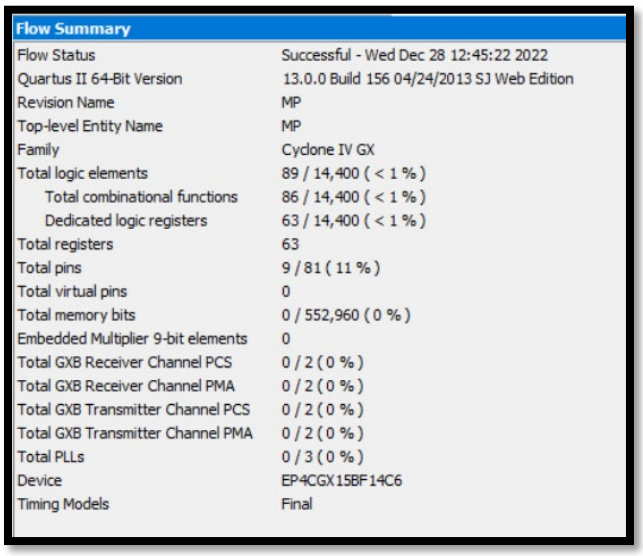


- On page 4 of the New Project Wizard, specify the tool name for the simulation. Type: **ModelSim-Altera**, format(s): **Verilog HDL**.



- Click on the **Finish** button to complete the new project creation.
- Open file and create a new **Verilog HDL File**.
- In the blank workspace, paste the provided Verilog code into the editor and save it as ***trafficlight.v***.
- Create a new Verilog HDL file and apply the specified testbench code. Save the file as "***tb_trafficlight.v***."

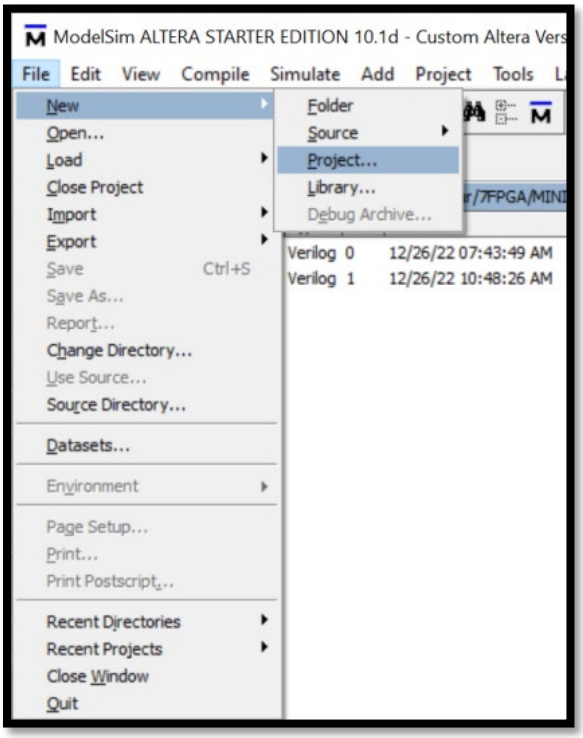
- The transferred code is reviewed and changed to get the desired output before compilation.



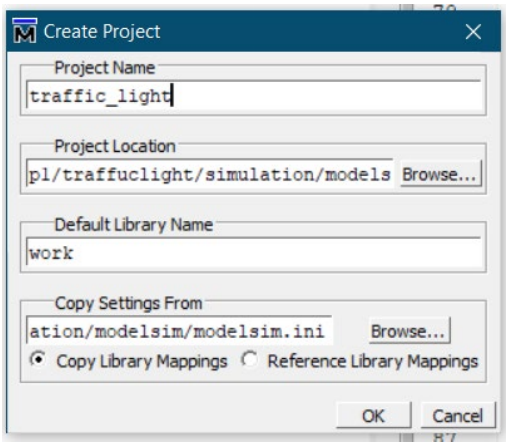
Flow Summary	
Flow Status	Successful - Wed Dec 28 12:45:22 2022
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	MP
Top-level Entity Name	MP
Family	Cyclone IV GX
Total logic elements	89 / 14,400 (< 1 %)
Total combinational functions	86 / 14,400 (< 1 %)
Dedicated logic registers	63 / 14,400 (< 1 %)
Total registers	63
Total pins	9 / 81 (11 %)
Total virtual pins	0
Total memory bits	0 / 552,960 (0 %)
Embedded Multiplier 9-bit elements	0
Total GXB Receiver Channel PCS	0 / 2 (0 %)
Total GXB Receiver Channel PMA	0 / 2 (0 %)
Total GXB Transmitter Channel PCS	0 / 2 (0 %)
Total GXB Transmitter Channel PMA	0 / 2 (0 %)
Total PLLs	0 / 3 (0 %)
Device	EP4CGX15BF14C6
Timing Models	Final

3.1.2 ModelSim Altera

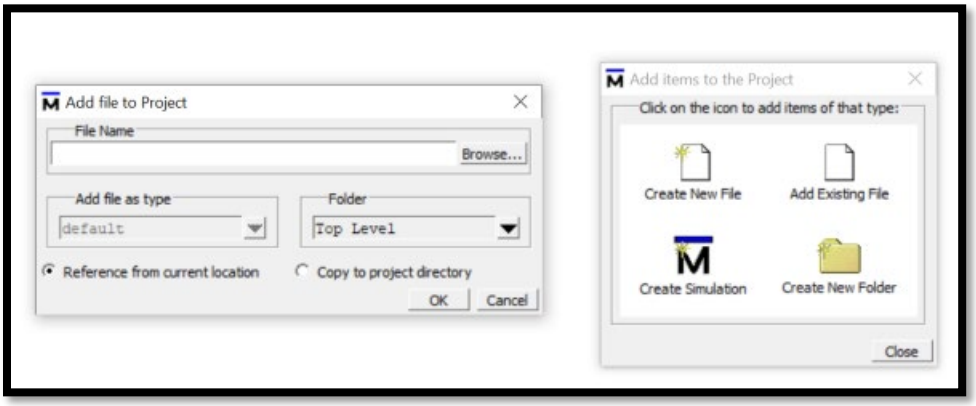
- After compiling the programs, launch the **ModelSim Altera** software and create a new project using the startup wizard.



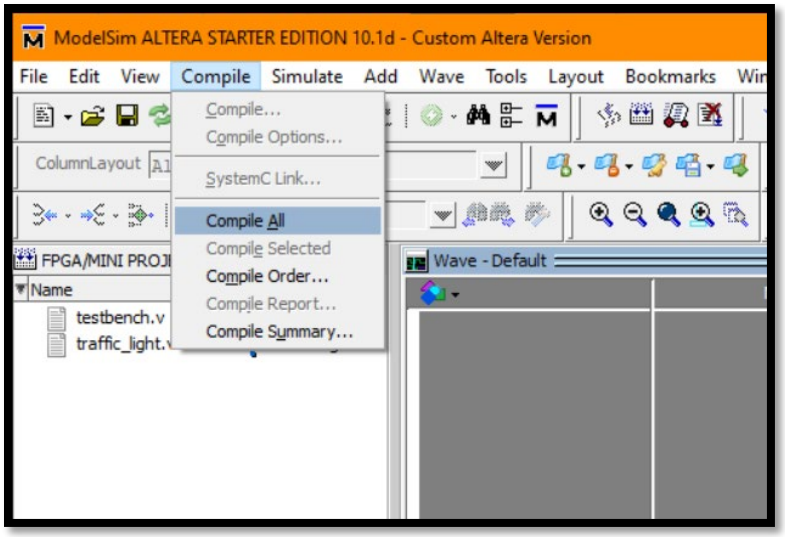
- Set the Project Name and Project Location to the same directory as the previous Quartus projects.

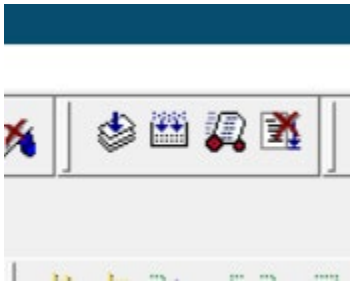


- In the Add Items to the Project window, select the Add Existing File option and look for previously created Verilog files. Click OK to close the window.

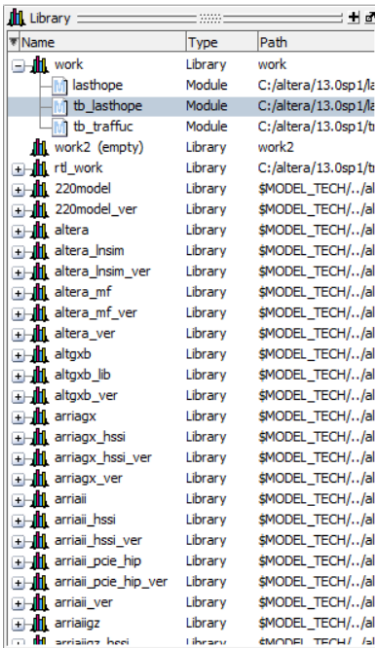


- The workspace's Project tab displays newly added files with a question mark as their status indicator. To compile the files, click the Compile All button in the toolbar.





- After compiling, the question marks will convert into a checkmark.
- Navigate to the Library tab of the workspace. Expand the work2 name by clicking the 'plus' button, then right-click the testbench Verilog file. Click Simulate



- New windows will appear, including the Sim tab, Objects, Processes (Active), and Wave windows. If the Object or Wave window does not appear, open it manually using the View options on the toolbar.
- To monitor signals, drag & drop them from the Objects to the Wave window.
- In the Wave box, click the Run -All button to start the testbench simulation.

3.1.3 Verilog Code for Traffic Light Controller

```
// fpga4student.com FPGA projects, VHDL projects, Verilog projects
// Verilog project: Verilog code for traffic light controller
module traffic_light(light_highway, light_farm, C, clk, rst_n);
parameter HGRE_FRED=2'b00, // Highway green and farm red
  HYEL_FRED = 2'b01,// Highway yellow and farm red
  HRED_FGRE=2'b10,// Highway red and farm green
  HRED_FYEL=2'b11;// Highway red and farm yellow
input C, // sensor
  clk, // clock = 50 MHz
  rst_n; // reset active low
output reg[2:0] light_highway, light_farm; // output of lights
// fpga4student.com FPGA projects, VHDL projects, Verilog projects
reg[27:0] count=0,count_delay=0;
reg delay10s=0,
delay3s1=0,delay3s2=0,RED_count_en=0,YELLOW_count_en1=0,YELL
OW_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)
HGRE_FRED: begin // Green on highway and red on farm way
 RED_count_en=0;
 YELLOW_count_en1=0;
 YELLOW_count_en2=0;
 light_highway = 3'b001;
 light_farm = 3'b100;
 if(C) next_state = HYEL_FRED;
 // if sensor detects vehicles on farm road,
```

```
 // turn highway to yellow -> green
 else next_state =HGRE_FRED;
end
HYEL_FRED: begin// yellow on highway and red on farm way
  light_highway = 3'b010;
  light_farm = 3'b100;
  RED_count_en=0;
 YELLOW_count_en1=1;
 YELLOW_count_en2=0;
  if(delay3s1) next_state = HRED_FGRE;
  // yellow for 3s, then red
  else next_state = HYEL_FRED;
end
HRED_FGRE: begin// red on highway and green on farm way
 light_highway = 3'b100;
 light_farm = 3'b001;
 RED_count_en=1;
 YELLOW_count_en1=0;
 YELLOW_count_en2=0;
 if(delay10s) next_state = HRED_FYEL;
 // red in 10s then turn to yello -> green again for high way
 else next_state =HRED_FGRE;
end
HRED_FYEL:begin// red on highway and yellow on farm way
 light_highway = 3'b100;
 light_farm = 3'b010;
 RED_count_en=0;
 YELLOW_count_en1=0;
 YELLOW_count_en2=1;
 if(delay3s2) next_state = HGRE_FRED;
 // turn green for highway, red for farm road
 else next_state =HRED_FYEL;
end
default: next_state = HGRE_FRED;
endcase
end
// fpga4student.com FPGA projects, VHDL projects, Verilog projects
// 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
```

3.1.4 Testbench Verilog Code for Functional Simulation

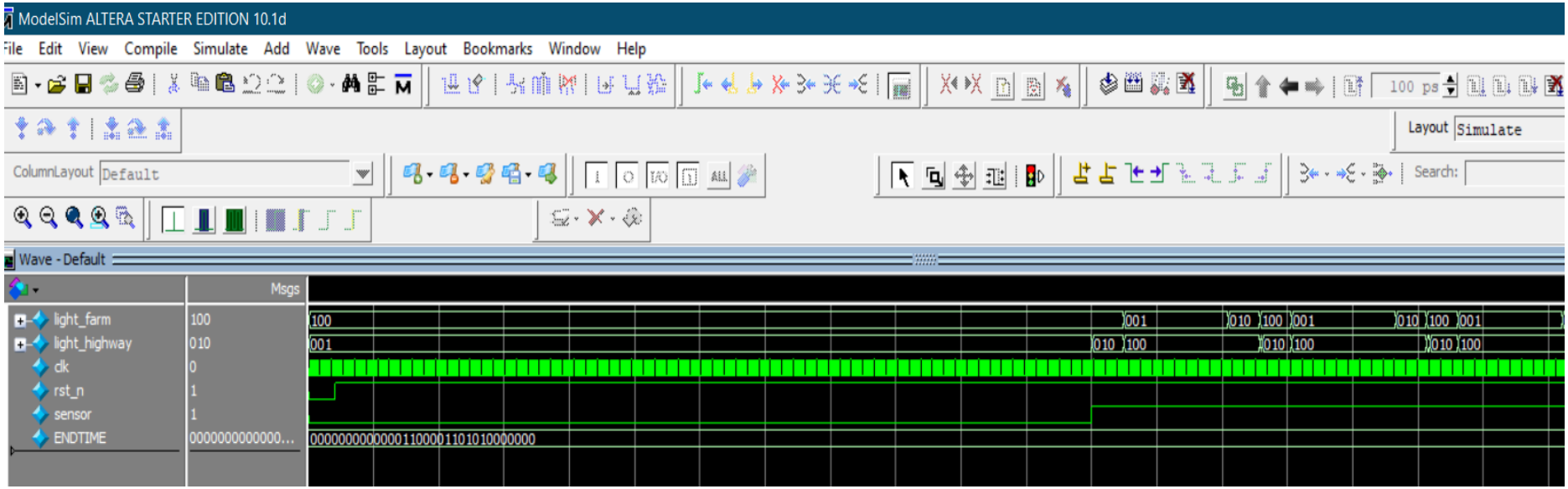
```
// fpga4student.com FPGA projects, VHDL projects, Verilog project
// Verilog project: Verilog code for traffic light controller
`timescale 10 ns/ 1 ps
// 2. Preprocessor Directives
`define DELAY 1
// 3. Include Statements
//`include "counter_define.h"
module tb_traffic;
// 4. Parameter definitions
parameter ENDTIME  = 400000;
// 5. DUT Input regs
//integer count, count1, a;
reg clk;
reg rst_n;
reg sensor;
wire [2:0] light_farm;
// 6. DUT Output wires
wire [2:0] light_highway;

// fpga4student.com FPGA projects, VHDL projects, Verilog projects
// 7. DUT Instantiation
traffic_light tb(light_highway, light_farm, sensor, clk, rst_n);

// 8. Initial Conditions
initial
 begin
 clk = 1'b0;
 rst_n = 1'b0;
 sensor = 1'b0;
 // count = 0;
//// count1=0;
// a=0;
 end
// 9. Generating Test Vectors
initial
 begin
 main;
 end
task main;
```

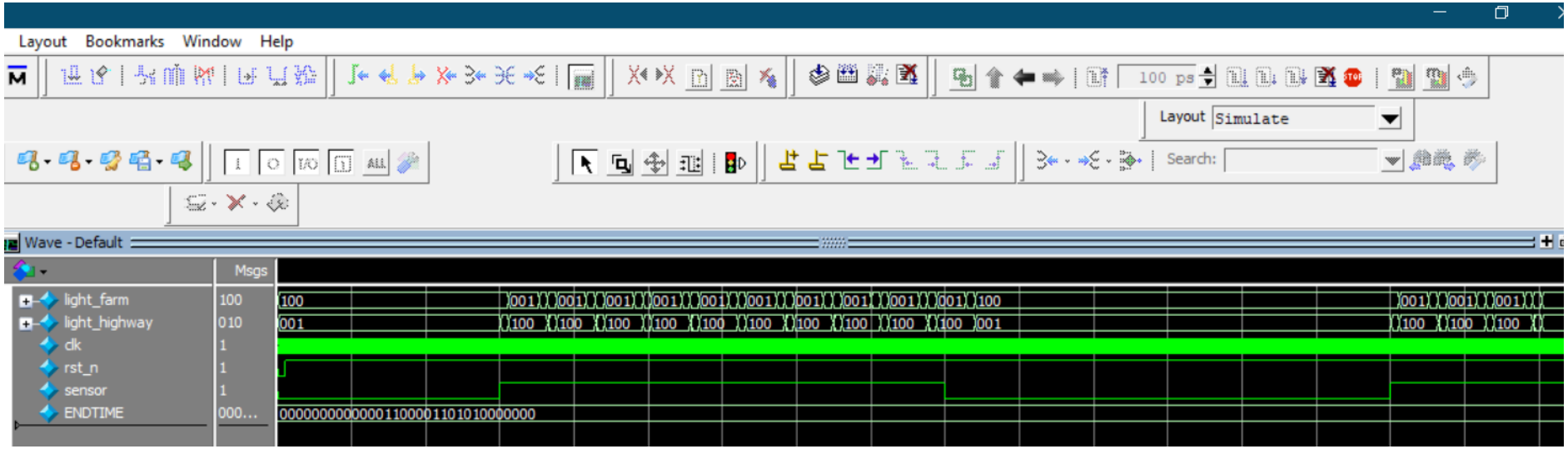
```
 fork
 clock_gen;
 reset_gen;
 operation_flow;
 debug_output;
 endsimulation;
 join
endtask
task clock_gen;
 begin
 forever #`DELAY clk = !clk;
 end
endtask

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

// fpga4student.com FPGA projects, VHDL projects, Verilog projects
task operation_flow;
 begin
 sensor = 0;
 # 600
 sensor = 1;
 # 1200
 sensor = 0;
 # 1200
 sensor = 1;
 end
endtask
// 10. Debug output
task debug_output;
 begin
 $display("-----------------------------------------------");
 $display("------------------     ----------------------");
 $display("----------- SIMULATION RESULT ----------------");
 $display("--------------            --------------------");
 $display("----------------        ----------------------");
 $display("-----------------------------------------------");
 $monitor("TIME = %d, reset = %b, sensor = %b, light of highway = %h,
light of farm road = %h",$time,rst_n ,sensor,light_highway,light_farm );
 end
```

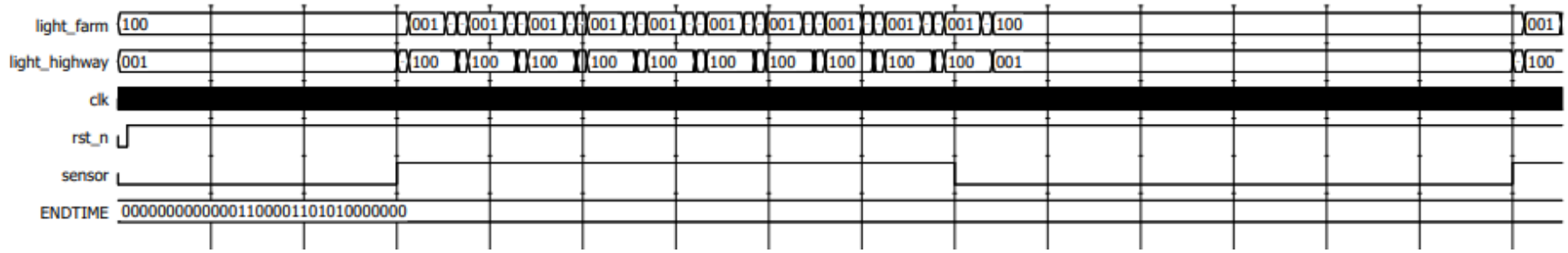
```
endtask

// fpga4student.com FPGA projects, VHDL projects, Verilog projects
//12. Determines the simulation limit
task endsimulation;
 begin
 #ENDTIME
 $display("---------- THE SIMUALTION END ----------");
 $finish;
 end
endtask

endmodule
```

4.1 Result & Analysis

4.2 Simulation Result




Layout Bookmarks Window Help
100 ps
Layout Simulate
Search:
Wave - Default
Msgs
light_farm 100
light_highway 010
clk 1
rst_n 1
sensor 1
ENDTIME 000...
00000000000001100001101010000000


light_farm
light_highway
clk
rst_n
sensor
ENDTIME
00000000000001100001101010000000

Transcript

```
# TIME =              1604897, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1604921, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1604923, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1604945, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605025, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605049, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605051, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1605073, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605153, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605177, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605179, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1605201, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605281, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605305, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605307, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1605329, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605409, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605433, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605435, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1605457, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605537, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605561, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605563, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
# TIME =              1605585, reset = 1, sensor = 1, light of highway = 4, light of farm road = 1
# TIME =              1605665, reset = 1, sensor = 1, light of highway = 4, light of farm road = 2
# TIME =              1605689, reset = 1, sensor = 1, light of highway = 1, light of farm road = 4
# TIME =              1605691, reset = 1, sensor = 1, light of highway = 2, light of farm road = 4
```

4.3 Analysis

Overview

The aim of this project is to design and simulate a traffic light controller for an intersection using Verilog and ModelSim. The traffic light controller manages the traffic flow for a highway and a farm road, transitioning through a series of states based on a sensor input and clock signals. The project implemented has four primary states: highway green and farm red, highway yellow and farm red, highway red and farm green, and highway red and farm yellow.

The traffic light controller uses the following states and transitions:

- **HGRE_FRED**: Highway green, farm red.
- **HYEL_FRED**: Highway yellow, farm red.
- **HRED_FGRE**: Highway red, farm green.
- **HRED_FYEL**: Highway red, farm yellow.

Simulation Setup

The simulation was performed using Model Sim with two Verilog files:

1. **traffic_light.v**: Contains the main traffic light controller module.
2. **tb_traffic.v**: Contains the testbench to simulate and verify the controller's functionality.

Observations

1. **Transition Sequence**: The traffic light transitions follow the expected state sequence:
    - **HGRE_FRED**: Highway is green, and farm road is red.
    - **HYEL_FRED**: Highway is yellow, and farm road is red.
    - **HRED_FGRE**: Highway is red, and farm road is green.
    - **HRED_FYEL**: Highway is red, and farm road is yellow.
2. **Timing**: The delays between transitions (3s and 10s) are correctly implemented and reflected in the simulation timestamps. The clock cycles are effectively used to generate the required delays.
3. **Sensor Input**: The sensor signal is high (1) throughout these logs, influencing the state transitions from HGRE_FRED to HYEL_FRED and onwards.
4. **Reset**: The reset (rst_n) is active high (1), ensuring that the FSM operates correctly during the simulation.

5.1 Conclusion

The traffic light controller project successfully implements a finite state machine to manage an intersection's traffic flow. The simulation results align with the design specifications, confirming that the Verilog code operates as intended. The transitions between states, controlled by the sensor input and clock signals, ensure that the traffic lights switch appropriately, providing a safe and efficient flow of traffic for both the highway and the farm road. The design is robust and ready for further validation or deployment on hardware.