**CODE**

/ fpga4student.com [FPGA projects](https://www.fpga4student.com/p/fpga-projects.html), [VHDL projects](https://www.fpga4student.com/p/vhdl-project.html), [Verilog project](https://www.fpga4student.com/p/verilog-project.html)s

// 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**,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)

**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**

**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**