Verilog Implementation of Digital Circuit Designs on FPGA using Vivado

By : ROHIT BAGDI (POWER ENGINNERING 214102113)

ABSTRACT:

Verilog designing is hardware descriptive language, the name itself suggest that it deals with the hardware designing and simulation. Basically, it becomes very difficult to mount the various electronic component on breadboard or PCB circuit. It also takes too much time for the simulation and sometimes many errors occur because of improper connection of components onto the circuit. And thus, to overcome this factor hardware descriptive language comes into conclusion. we can code the process using Verilog and we can mount it on a circuit or just upload it to the circuit accordingly so that particular circuit will work as according to the code we have written. HDL language is often used for sequential circuits like shift register, combinational logic circuit like adder, subtractor etc. basically it describes the digital systems like microprocessor or a memory. Whatever design that is describe in HDL are independent, it has its unique state of work, very much easy to simulate, designing and debugging, and very useful than schematics, especially for large circuits thus, to overcome difficulties or problems to design the circuits manually with breadboard and PCB, use of Verilog designing in this complex world is increasing a way better. Keywords: HDL, Verilog, PCB, Combinational logic circuit, microprocessor, simulation, register.

I. INTRODUCTION

Traffic light signal controlling is most important and essential thing for any country to protect the people from heavy load of traffic. before this kind of invention there was much difficult for traffic police to handle the heavy traffic (particular direction given manually). thus, traffic signal controlling technology made much easier to handle the heavy loads of traffic. Safe movement of vehicles without any type of collision, accidents. Apart from the traffic it is very necessary for the people to cross the roads at particular time interval. And this is only possible by controlling the traffic by giving some kind of signal. Analysing the traffic, estimating the delays to the areas is crucial part.

States for Three intersection way.

The aim of the project is to design a traffic controller for a T-intersection. Let's understand the problem statement through the image given below.

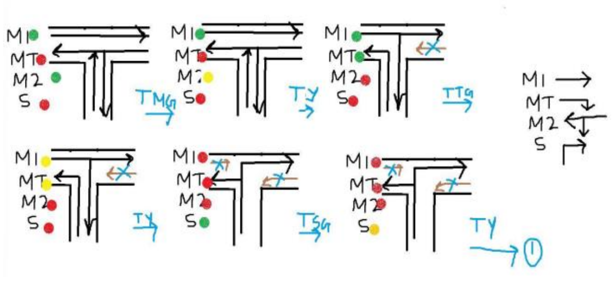


Fig. 1 Traffic signals for T-intersection way

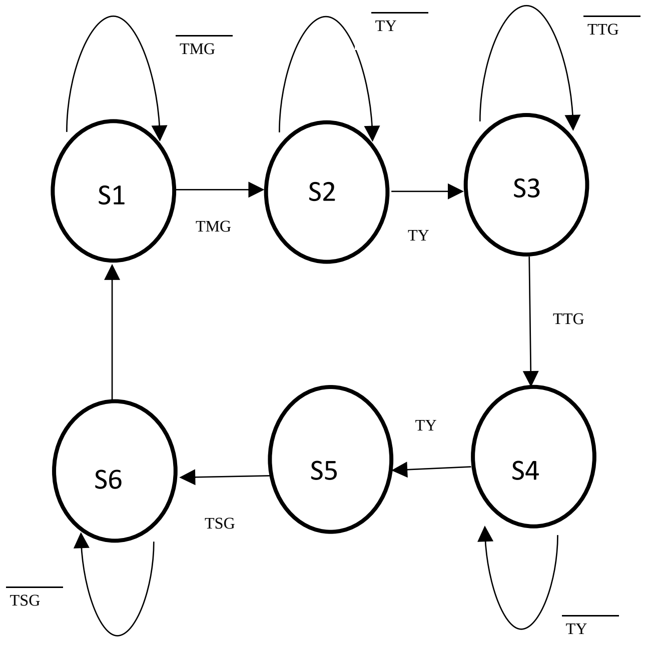


Fig 2. State diagram for the project.

Present state ABC	**Input**	**Next state**	**M1 RYG**	**M2 RYB**	**T RYB**	**S RYB**
001	$\overline{TNG}$	001	001	001	100	100
001	TMG	001	001	001	100	100
010	$\overline{TY}$	001	001	010	100	100
010	TY	001	001	010	100	100
011	$\overline{TTG}$	001	001	100	001	100
011	TTG	001	001	100	001	100
100	$\overline{TY}$	010	010	100	010	100
100	TY	010	010	100	010	100
101	$\overline{TSG}$	100	100	100	100	001
101	TSG	100	100	100	100	001
110	$\overline{TY}$	100	100	100	100	010
110	TY	100	100	100	100	010

State table

Verilog code for traffic light controller:

```
//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
//  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
// , 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
```

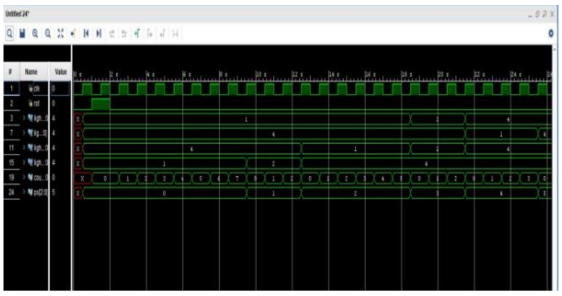
```
   delay10s=0;
   delay3s1=0;
   delay3s2=1;
   count_delay<=0;
  end
  else
  begin
   delay10s=0;
   delay3s1=0;
   delay3s2=0;
  end
 end
end
// create 1s clock enable
always @(posedge clk)
begin
 count <=count + 1;
 //if(count == 50000000) // 50,000,000 for 50 MHz clock running on real FPGA
 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

```
// 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;

//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

//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,li
ght_farm );
 end
endtask

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

endmodule
```

Output waveform:



After synthesis