Traffic Light Controller using Verilog HDL

A.Apurva Reddy

Electronics and Communication Engineering, Chaitanya Bharathi Institute of Technology

Gandipet, Hyderabad

apurvareddy212@gmail.com

Telangana, India

***Abstract*— The objective of this project is to design a traffic light controller using hardware description language Verilog. The controller is designed using a state machine approach.**

***Keywords*— Finite state machine, behavioural modelling, Xilinx , iSim simulator**

I. Introduction

Traffic lights are an essential part of human civilization today. Without them road transport would be chaos. The traffic light controller controls and monitors the state of the traffic lights at a given time. Thereby it is a helpful tool in regulating traffic. The level of complexity of the design depends on the expected volume of traffic in a certain area. Areas like metropolitan cities which tend to have heavy traffic will require a more sophisticated design. Simpler traffic light controller designs like the one implemented here can be used in remote areas which have less traffic.

II. Design

The design required only a single top level model. The software used was Xilinx ISE Design Suite 14.7. Finite state machine approach was taken. Code was written in verilog language.

A. Concept

The traffic light controller controls four traffic lights located at an intersection. The lights were considered to be located exactly at north, south ,east, and west positions. The three lights that are displayed are red for stop, green for go and yellow for slow down. Each of the four lights stays at red, green and yellow states for a certain amount of time before transferring to the next state.While one of the fours lights are in green or yellow states the remaining 3 lights would be in red state.

B. Code

Clock and reset inputs were given and outputs are four 3-bit registers. Parameters were defined for each of the traffic lights in green and yellow states. An always block is initiated to ensure the process is continuous. At reset the traffic light at north will be green and clock which is named as tick in the program will start counting from zero. The controller will remain in this state for 8 clock cycles. Afterwards it will go to the next state. After defining the reset condition we go on defining the remaining state by taking case statement. In each case the clock tick is monitored to see if it reaches 8 counts. If 8 cycles have not been completed tick is incremented. After all the states have been covered the outputs for the given states are defined using case statements. For green light the binary value is 3'b001, for yellow 3'010 and red 3'b100. The output value for each parameter are specified. Hence the traffic light controller has been designed in the same way as a finite state machine.

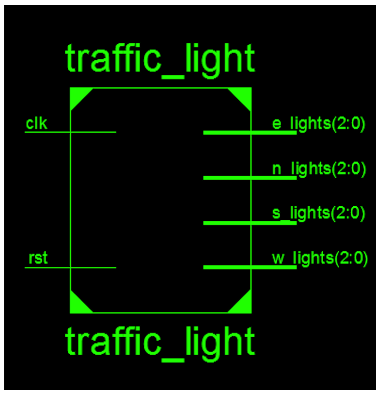


Fig. 1 Schematic of top level module for traffic light controller.

III. Functional Verification

The design was tested using a Verilog testbench and simulated in iSim simulator. The design was confirmed only after the simulation results showed that it was functioning as expected.

A. Testbench

The testbench required only a single module. A unit under test(uut) was instantiated to connect the ports in the design and testbench. Two procedural blocks were initialized. The first block is a clock generator and the second block contains the stimulus. The simulation was coded to run for 1000 ns.

B. Simulation Results

After checking the behavioral syntax of the testbench the simulation results were run. The simulation tool used was iSim simulator. Initially, there were a few unexpected results that deviated from the desired results. A few minor modifications were made in the design. The second time expected results were achieved and the functioning of the design was verified.

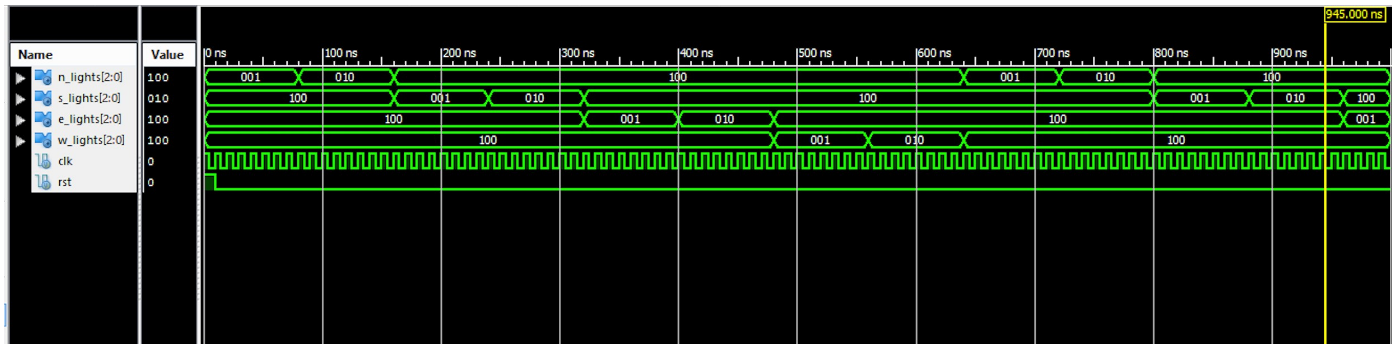
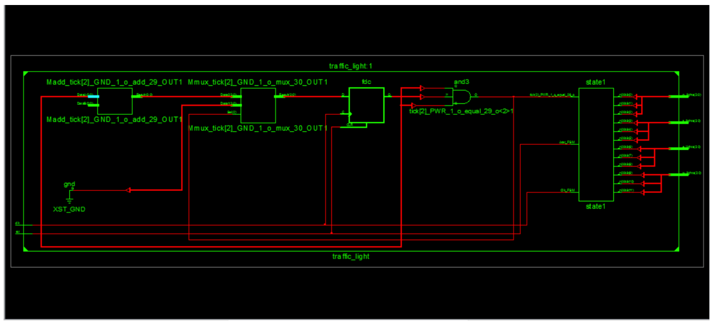


Fig. 2 Simulation result waveform

IV. Conclusions

The concept of finite state machines has been used to design a practical application that is widely used in everyday life. There are many similar real life applications that use state machines and can be modelled easily using hardware description languages like Verilog. The final RTL schematic of the design is presented below.



traffic_light 1
Madd_tick[2]_GND_1_o_add_29_OUT1
Mmux_tick[2]_GND_1_o_mux_30_OUT1
fdc
and3
state1
tick[2]_PWR_1_o_equal_29_o<2>1
gnd
XST_GND
traffic_light

Fig. 3 RTL schematic of the final design.

References

1. https://veriloggguide.readthedocs.io/en/latest/verilog/fsm.html
2. https://www.xilinx.com/support/documentation/university/ISE-Teaching/HDL-Design/14x/Nexys3/Verilog/docs-pdf/lab10.pdf
3. Samir Palnitkar, “Verilog HDL: A guide to Digital design and synthesis”, 2/e, Pearson Education, 2008.
4. https://www.fpga4student.com/2016/11/verilog-code-for-traffic-light-system.html
5. https://vlsicoding.blogspot.com/2013/11/verilog-code-for-traffic-light-control.html