A Verilog Model of Adaptable Traffic Control System Using Mealy State Machines

M. Ali Qureshi, Abdul Aziz, and S. Hammad Raza

Abstract—**In this paper an efficient traffic control system is designed using Mealy finite state machines. The effects of state encoding schemes like sequential encoding, gray encoding and One-Hot encoding are compared on the basis of size of synthesized circuit. Coding of the design is done in Verilog HDL and the design is tested and simulated on Spartan-3 xc3s400 FPGA development kit. The sensors are added as input to the controller for emergency conditions like ambulance etc. This system is also capable to change the timings of traffic signals according to the density of vehicles on the roads. The design has several benefits over ordinary traffic light controllers built with microcontrollers or Programmable logic controller such as simple structure, high reliability, low costs, ease in installation and maintenance. The design of adaptable traffic control system is carried out for a chowk consisting of five roads. Each road is divided into main road (for straight movement) and cross road (for crossing). To some extent, traffic jam, unreasonably latency time of stoppage of vehicles, emergency vehicles like ambulances, VIP movement or forcibly passing is solved. So, this system carries a broader application prospect.**

Index Terms—**FPGA, DSP, FSM, HDL, Xilinx, ASIC, LUT**

I. Introduction

Conventional traffic control systems has two major drawbacks: First, [1] due to lack of adjustments in timings of traffic signals, the traffic has to wait a long on the lane with few vehicles while on same lane, the traffic cannot pass through in short time due to rush on lane. Second, [2] there is no provision of movement of emergency vehicles like ambulance and fire brigades etc. In rush hours these emergency vehicles have to wait a long and results in human and financial loss. So, there is a need to develop a secure, fast and reliable traffic control system capable to control the vehicular traffic in rush hours without a need of traffic sergeant.

In this paper, we have developed a real traffic control system using Mealy state machines. The design is implemented in Verilog HDL Hardware Description language. We use different modeling styles to implement state machines to improve the readability of code and to increase the speed. The effect of state encoding on the size of synthesized circuit is also realized. The implemented architecture is then tested for the validation of the design on Spartan-3 xc3s400 FPGA Development Kit.

A. Field Programmable Gate Array (FPGA)

Field Programmable Gate Array (FPGA) is a reconfigurable hardware platform useful for the implementation of high digital functions. Using fixed point, parallel computational structures, FPGA provides computational speeds as much as 100 times greater than those possible with Digital Signal Processors (DSP). The extremely fast computational capability of FPGAs allows a few microseconds for real-time computation of algorithms in spite of their complexities. Furthermore, as DSPs, FPGAs are very low cost components [3][4].

Xilinx Spartan-3 FPGAs are ideal for low-cost, high-volume applications and are targeted as replacements for fixed-logic gate arrays. The Spartan-3 FPGA is not only available for a very low cost, but it integrates many architectural features associated with high-end programmable logic. This combination of low cost and integrated features has made it an ideal replacement for ASICs (Application Specific Integrated Circuits) [5].

B. Finite State Machines (FSM)

Finite State machines are used to generate sequence of control signals. There are two types of state machines: Mealy machines and Moore machines. The difference between Mealy and Moore machines relies in the methods of output generation. In Moore machines, outputs are function of current state. This means that whenever state changes, the output also changes. In Mealy machines, outputs are function of both input and current state. A state machine can be divided into three parts: State register, Next-State Logic and Outputs. It is necessary to model each part of the state machine in Verilog [6].

C. State Machine Modeling Style

We can model each part of the state machine independently or by combining parts into one section. Different modeling styles of state machines are shown in Table I [7].

TABLE I: STATE MACHINE MODELLING STYLE

Style	State Register	Next-State Logic	Output Logic
1	Separate	Separate	Separate
2	Combined	Combined	Separate
3	Separate	Combined	Combined
4	Combined	Combined	Combined
5	Combined	Separate	Combined

In the first style, each of the part of FSM is modeled in a separate always block. This style is modular in nature and easiest to maintain. The size of code increases in this modeling style.

In second style, next-state logic and state register are combined. As Next-state logic and state register are strongly

Manuscript received April 9, 2012; revised May 25, 2012.

M. A. Qureshi and A. Aziz Qureshi are with Islamia University of Bahawalpur, Pakistan (e-mail: ali.qureshi@ iub.edu.pk, abdul.aziz@ iub.edu.pk).

related, there is no need for the next_state temporary variable. This style is more compact than the first and even more efficient because the next-state logic is only evaluated on clock edges. Style three combines the next-state logic and output logic. But the code becomes complex and difficult to read.

D. *Encoding*

State encoding scheme have a great influence on circuit size as well as performance. Different types of state encoding schemes can be used like Sequential State Encoding, Gray State Encoding, and One-Hot State Encoding. Out of these encoding schemes, One-Hot encoding is more efficient in terms of speed and size. The choice of state encoding scheme depends on the design application [7].

State machines are widely used in applications that require prescribed sequential activity e.g. sequence detector, digital combinational lock, elevator control, traffic light controller.

II. System Implementation

A. *Roads Structure*

Generally, a traffic signal system has three lights. A green light on the bottom of the signal indicates the traffic to proceed, a yellow light in the middle warns the traffic to slow and prepare to stop, and red light on the top indicates the traffic to stop. Fig. 1 shows structure of any chowk consisting of five main roads and each road is divided into two main roads (straight and cross). We are using ten traffic signals L1, L2,….L10. The signals on straight roads are L1, L3, L5, L7 and L9 while L2, L4, L6, L8 and L10 are traffic signals on cross roads. Traffic signals on straight roads have three lights red, yellow and green while signals on cross roads have two lights red and green.

There are four sensors on roads SW1, SW2, SW3 and SW4 for emergency conditions. SW1, SW2, SW3 and SW4 switches are linked with traffic signals (L1, L5), (L2, L6), (L3, L7) and (L4, L8) respectively. Whenever any one of the sensors output is enabled, appropriate traffic starts to continue on the roads according to the position and priority of the switches and rest of the signals are off.

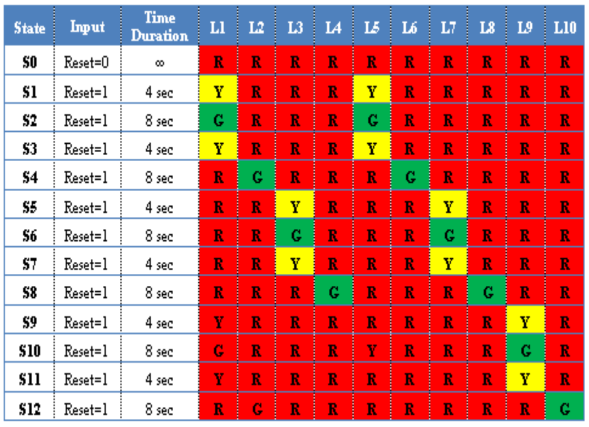
TL and TS are two inputs for controlling the green signal ON timings. When TS is enabled, ON timing of green light will reduce to halve. When TL is enabled, ON timing of green light will be doubled.

B. *State Table*

The timing states of traffic lights are shown in Table II. The timing of signal lights can be increased or decreased according to the traffic load on the roads.

- Initially before resetting the TLC, red lights of all traffic signals L1-L10 are ON.
- After resetting the TLC, yellow light of signal L1 & L5 are ON and red lights on the remaining signals are ON.
- After a delay of four seconds, green lights of traffic signals L1 & L5 are ON while red light on the remaining signals are ON.
- After eight seconds, yellow light of signals L1 & L5 are again ON while red light on the remaining signals are ON.
- After four seconds, red light of signals L1 & L5 are ON and green light on L2 and L6 are ON.
- After eight seconds, red light of signals L2 & L6 are ON and yellow light on L3 and L7 are ON.
- After four seconds, green light of signals L3 & L7 are ON while red lights on the remaining signals are ON.
- After eight seconds, yellow light of the signals L3 & L7 are ON and red lights on the remaining signals are ON.
- After four seconds, green light on the signals L4 and L8 are ON and red lights on the remaining signals are ON.
- After eight seconds, yellow light of signal L9 is ON and red lights on the remaining signals are ON.
- After four seconds, green light of signal L9 is ON and red lights on the remaining signals are ON.
- After eight seconds, yellow light of signal L9 is ON and red lights of the remaining signals is ON.
- After four seconds, green light of signal L10 is ON and remains ON for eight seconds with all red lights are ON.
- The sequence repeats until the reset of TLC or any one of the emergency switches are enabled. The graphical representation of the sequences of traffic lights with the help of state diagram is shown in Fig. 2.
- Whenever there is emergency vehicle on the road, corresponding emergency switch installed on that road is enabled, the traffic on that road is allowed to pass and the traffic on all the roads is forcibly stopped. After passing of emergency vehicle, the emergency switch will be disabled. And the sequence is allowed to continue from a position where it was stopped. There is priority queue for the emergency switches. The switch which is enabled first will has highest priority. The sequence continues to start in order of switches priority.

TABLE II: Timing States of Traffic Lights



State	Input	Time Duration	L1	L2	L3	L4	L5	L6	L7	L8	L9	L10
S0	Reset=0	∞	R	R	R	R	R	R	R	R	R	R
S1	Reset=1	4 sec	Y	R	R	R	Y	R	R	R	R	R
S2	Reset=1	8 sec	G	R	R	R	G	R	R	R	R	R
S3	Reset=1	4 sec	Y	R	R	R	Y	R	R	R	R	R
S4	Reset=1	8 sec	R	G	R	R	R	G	R	R	R	R
S5	Reset=1	4 sec	R	R	Y	R	R	R	Y	R	R	R
S6	Reset=1	8 sec	R	R	G	R	R	R	G	R	R	R
S7	Reset=1	4 sec	R	R	Y	R	R	R	Y	R	R	R
S8	Reset=1	8 sec	R	R	R	G	R	R	R	G	R	R
S9	Reset=1	4 sec	Y	R	R	R	R	R	R	R	Y	R
S10	Reset=1	8 sec	G	R	R	R	Y	R	R	R	G	R
S11	Reset=1	4 sec	Y	R	R	R	R	R	R	R	Y	R
S12	Reset=1	8 sec	R	G	R	R	R	R	R	R	R	G

TABLE III: Comparison of Encoding Schemes Based on Device Utilization

Encoding Scheme	No. of Slices	No. of Slice Flip-Flops	No. of 4-input LUTs
Sequence Encoding	84	66	128
Gray Encoding	86	65	132
One-Hot Encoding	77	73	114

III. Simulation Results

The design is simulated with Xilinx ISE Simulator to verify that the design behaves as expected in the target device both in terms of functionality and timing. Fig 3 shows the timing waveform of the design obtained with Xilinx ISE Simulator.

The design is tested for different state encoding schemes. The percentage utilization of FPGA in terms of number of slices, number of flip-flops & number of (LUTs) is compared for different encoding schemes as shown in Table III. The Top-Level block diagram of custom system is shown in Fig. 4. The inputs are shown on the left side and outputs are shown on the right side of the diagram. The descriptions of different I/O pins of the IC are given in Table IV.

TABLE IV: Pins Description

Input Signal	**Description**
CLK	System Clock
RESET	Reset Input
SW1	Embergency switch on Road 1 and 5
SW2	Embergency switch on Road 2 and 6
SW3	Embergency switch on Road 3 and 7
SW4	Embergency switch on Road 4 and 8
TS	When enabled, Green signal ON timing halved
TL	When enabled, Green signal ON timing doubled
Output Signal	**Description**
L15 <2:0>	Traffic Lights on Road 1 and 5 (Straight)
L26 <1:0>	Traffic Lights on Road 2 and 6 (Cross)
L37 <2:0>	Traffic Lights on Road 3 and 7 (Straight)
L48 <1:0>	Traffic Lights on Road 4 and 8 (Cross)
L9 <2:0>	Traffic Lights on Road 9(Straight)
L10 <1:0>	Traffic Lights on Road 10 (Cross)
SEG_a-SEG_g	Seven segment display for timing representations

IV. Conclusion

In this paper, we have used finite state machines (FSM) to design an efficient and intelligent traffic light controller. Mealy machines are used to implement the system because outputs are also controlled by the inputs. The language used for the implementation is Verilog HDL with modeling style 2. Using Style 2, the code is compact and more efficient in terms of speed. The design is tested on Spartan-3 xc3s400 FPGA development kit.

The design is tested for different types of state encoding schemes like sequential encoding, gray encoding and One-Hot encoding. The percentage utilization of FPGA in terms of number of slices, number of flip-flops and number of Look up tables (LUTs) for these schemes are evaluated. From the results, it is evident that one-Hot encoding scheme is the best choice in terms of device utilization.

This model takes care of traffic on any chowk consisting of five roads. The implementation of complex traffic light for ten crossing roads in real time are considered for any chowk consist of five main roads and each main road is divided into two roads one for straight and other for cross moving vehicles. The system has several benefits over ordinary traffic light controllers such as simple structure, high reliability, low costs, easy installation and maintenance and so on. So, the system owns a broader application prospect.

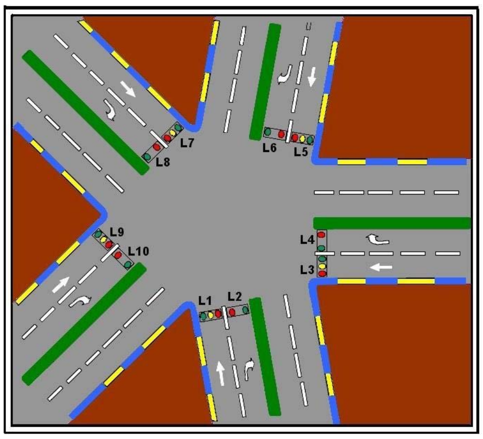


Fig. 1. Structure of chowk consisting of five main roads

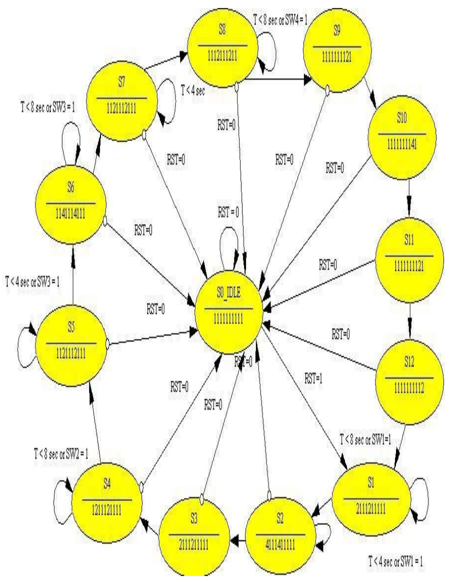


Fig. 2. State diagram for controlling traffic lights

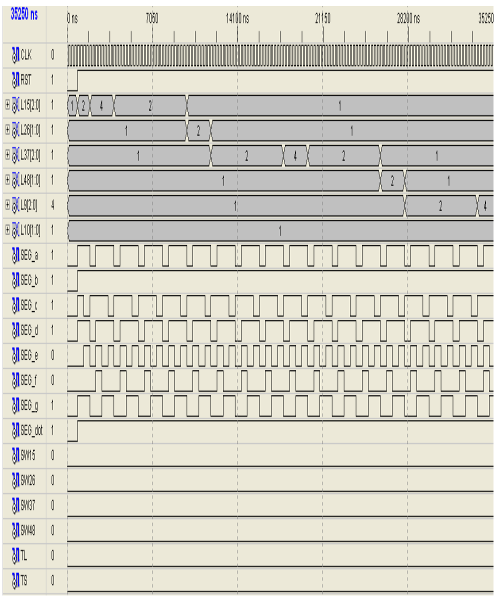


Fig. 3. Simulation results obtained with Xilinx ISE simulator.

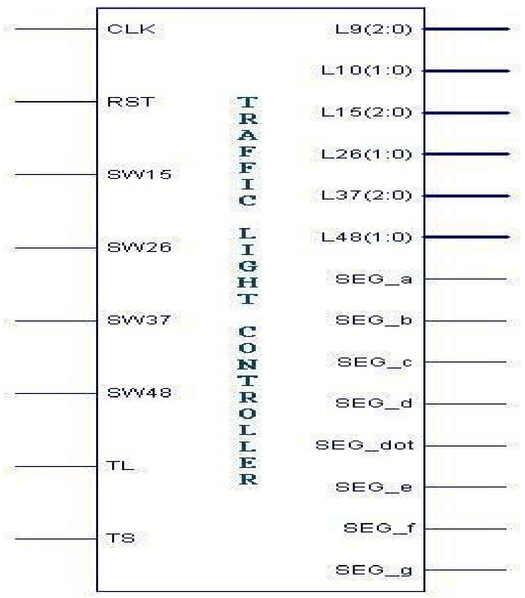


Fig. 4. Top Level block diagram of traffic light controller

ACKNOWLEDGEMENTS

We would like to thank our respected teachers and colleagues especially to Engr. Qaiser Ijaz, Engr. Anees-ul-Hassan for their guidance and ever present support. We would also pay special thanks to Prof. Jan M. Keerio for development of VLSI Design Lab at UCE&T and providing an environment suitable for research and development.

REFERENCES

[1] Z. Yuye and Y. Weisheng, “Research of Traffic Signal Light Intelligent Control System Based On Microcontroller,” 2009 First International Workshop on Education Technology and Computer Science.

[2] W. M. E. Medany and Mr Hussain, “FPGA-based Advanced Real Traffic Light Controller System Desigh,” *IEEE International workshop on Intelligent Data Acquisition Computing Systems: Technology and Applications*, 6-8 September, Dortmund, Germany, 2007.

[3] FPGA Design Tutorial, Copyright © 1-CORE Technologies, 2004-2009. [Online]. Available: http://www.1-core.com/library/digital/fpga-design-tutorial/quick-start-guide.shtml

[4] P. P. Chu, “FPGA Prototyping by VHDL Examples-Xilinx Spartan-3 Version,” *John Wiley and Sons*, 2008

[5] ISE 9.1 In-Depth Tutorial, Xilinx Inc. Copyright 1995-2007. [Online]. Available: http://download.xilinx.com/direct/ise9_tutorials/ise9tut.pdf

[6] Programmable Logic Design (Quick Start Handbook), Xilinx Copyright 2002-2006, Online]. Available: http://www.xilinx.com/ publications/products/ cpld/logic_handbook.pdf

[7] J. M. Lee, “Verilog Quick Start,” *2nd Edition*, 1998.