DIGITAL SYSTEM FOR TRAFFIC LIGHT CONTROLLER

ANIL KUMAR C
[1]Research Scholar, ECE Dept, St.Mary's Group of Institutions, Hyderabad, AP-India
E-mail id: - ac58248@gmail.com

Abstract: *- The simple traffic light controller design project was introduced to alleviate this shortcoming and gain experience in solving implementation and interfacing problems of a modern digital system. we implement a fully functional traffic signal controller for a four-way intersection. Intersection is complete with sensors to detect the presence of vehicles waiting at or approaching the intersection. This project incorporates many concepts and components that are discussed in detail throughout the theory. These include FPGAs, VHDL for modeling and finite state machines, serial communication, synthesis, embedded microprocessors, memory interfaces, and signal synchronization. A bottom-up with a partially specified design methodology is used to encourage students to use their breadth of knowledge and creativity.*

Key words: *Spartan 3e fpga, vhdl, seven segment display, Traffic lights*

1. INTRODUCTION

Traffic lights, also known as traffic lamps, traffic signals, stoplight, stop-and-go lights semaphore or robots, are signaling devices positioned at pedestrian crossings, road intersections, and other locations to control competing flows of traffic. Traffic lights have installed in most cities around the world to control the flow of traffic. It assign the right of way to road users by the use of lights in standard colors (Red - Amber - Green), using a universal color code (and a precise sequence, for color blind). Traffic lights are used at busy intersections to more evenly apportion delay to the various users.

The increasing amount of traffic in the cities has a large impact on the congestion and the time it takes to reach a certain destination. But not only the amount of traffic but also how you deal with this traffic has a large impact. Adding roads is not sufficient by itself, since they will always reach an end point, like junction or bottlenecks. Bottlenecks cannot be prevented. However the way junctions are controlled has a lot of room for improvement. Junctions are controlled; it is mostly done by traffic lights. Traffic lights though, are most of the time not adaptive.

The classic traffic light controller has a 'fixed-cycle' which does not take in account how much traffic comes from any direction; it just switches configurations of lights on a timer interval. It often causes road users to wait at a completely empty junction with only one road user waiting for a red sign.

Improvements already have been made, by putting sensors in the lanes in front of the traffic lights to let the controller only cycle between occupied lanes, thus disabling the chance of having to wait at a red light at an empty junction. An extension to this is "green waves", which are multiple traffic light controllers linked together. Traffic lights in the "green wave" are programmed to switch to green signs in a wave like fashion. So when road users are detected at the first traffic light, they will be able to get green signs for all the next junction participating in the "green wave".

More theoretical approaches to improve the traffic light control include machine learning algorithms. Machine Learning algorithms store the sensor information the sensors gather about the road users crossing the junction. This stored sensor information samples provide a way to predict the future driving behavior of road users and therefore enable the traffic light controller to calculate future waiting times for those road users for each action the traffic light controller can make. When the controller has the actions combined with waiting times, the optimal action would be to do one of these actions where the expected waiting times are the lowest.

In this a Reinforcement Learning implementation is used on a simulation of a simplified city with roads and junctions. The simulation has the possibility to get all the information that is needed to make a decision and is therefore completely observable. Viability of these machine learning techniques is shown. These theoretically proven methods do not take some issues, like partial observability. In the real world it is not realistic to assume that all the information can be gathered or that the gathered information is 100% accurate. Thus decisions have to be made based on incomplete or erroneous information. This implies working with partial observability. The used simulator has a discreet grid-like state space representation of the whereabouts of all road users. Running the simulator generates data this data exists of road users moving from one grid point to another.

Adding the Multi Agent approach increases the computation complexity. Instead having to compute a best action based on one agent, the combined best action needs to be chosen from all the combinations of best actions of the single agents. To diminish the computational complexity several heuristic approximations are possible in solving partial observability. Examples of heuristic approximations are Most Likely State (MLS) and Q-MDP.

2. TRAFFIC LIGHT CONTROLLER DESIGN

This thesis describes a simple traffic light controller design project for a junior level digital systems class at the Virginia Commonwealth University. It is developed because there was a need for laboratory exercises that incorporated microprocessors, simulation, VHDL modeling, serial communications, memory interfaces, and a variety of related topics into a complete digital system. It requires students to develop a state machine based controller for traffic signals at a four-way intersection. This intersection has two travel lanes in each direction; east, west, north and south. In addition, each direction has a dedicated left turn lane. Each lane has a sensor to indicate if a car is waiting at a red light. Travel lanes contain additional sensors to indicate if cars are approaching the intersection.

This project stresses the difference of writing VHDL for modeling and synthesis and that VHDL should not be thought of as a programming language. It gives proper design of combinational and sequential circuits. It requires proper definition of pin constraints for interfacing peripherals external to the FPGA.

Another drawback is that the study of components often occurred only in simulation. This led to confusion on how to write synthesizable VHDL.

The following sections describe the traffic signal phase requirements, the system architecture, and hardware test bench. Then, the progression of the signal controller design project is presented as a series of six stages, each building upon the previous.

2.1 Traffic Signal Phase Sequences

The traffic light controller must handle a four-phase signal intersection. Students have the option to implement two 4-phase sequence options. In the first option, through traffic for the east and west directions proceed at the same time; then north and south through traffic, followed by the north and south turning lanes. Finally, the east and west turning lanes can proceed. In the second option, through traffic and the turning lane for a single travel direction access the intersection at the same time. Students may implement an alternate phase sequence, but each phase must be as described. See fig).2.1 for phase details.

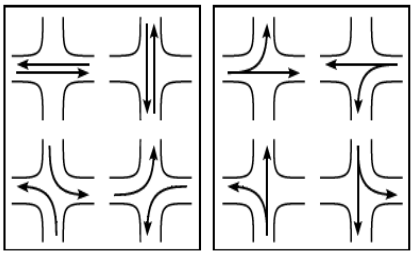


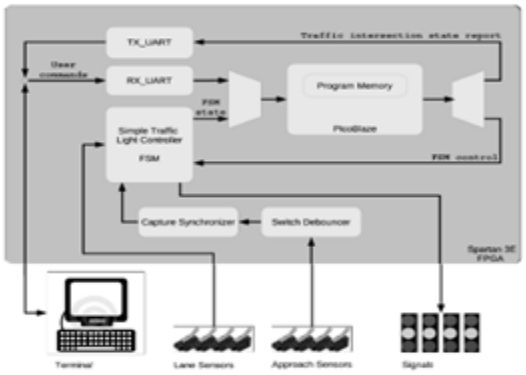
Fig: 2.1. 4-Phase options: (left) option I, (right) option II.

A six-phase signal sequence can also be implemented. The six-phase intersection is commonly referred to as a split green intersection. It is a combination of leading and lagging left turns. Left turns are implemented at the beginning and the end of each north-south and east-west sequence.

2.3 System Architecture

The complete architecture, shown in figure 2.2, consists of the traffic light controller FSM, an embedded processor, embedded program memory, multiplexers for processor input and output, a bidirectional UART, digital switch de bouncers, and capture synchronizers. Lane sensor inputs are directly monitored by the FSM. Approach signals are first de-bounced and synchronized. The light controller state information is passed to the processor for monitoring and reporting purposes. The processor reports the FSM state in a human readable format to a PC terminal through the embedded UART. Commands from the terminal can also be received and processed to generate control signals to the FSM. For example, the ASCII command "AR" may instruct the processor to set the traffic light controller to an, all-red, failsafe mode.

Implementation of the traffic light controller is targeted to the Diligent Spartan 3e 2 FPGA development board. This board contains a Xilinx Spartan 3E FPGA, flash memory, and various user IO devices and interfaces. Using an FPGA allows students to add hardware without having to worry about the physical connections between them. Instead, they can concentrate on the operation and interfaces of the various devices used. At the beginning of the project, a Moore finite state machine is the only device on the FPGA. Later, the signal controller will monitor sensors to detect whether a car is present or approaching the intersection. Sampling the lane sensors is fairly trivial and does not require the use of any additional hardware.



Fig; 2.2 System Architecture

Things become a slightly more complicated when approach sensors are introduced. These asynchronous signals must be debounced and captured. To implement this feature, a digital switch debouncer is implemented using a sample and count technique. A simple capture synchronizer, shown in figure 2.3, is then used to synchronize the asynchronous approach sensor signals to the FSM clock.

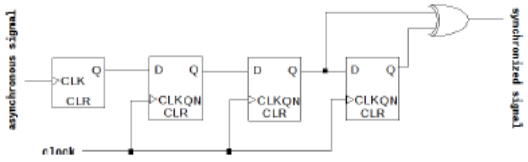


Fig: 2.3. Capture Synchronizer.

3. IMPLEMENTATION

The traffic light controller can be implemented by using the state machines in the four directions. Initial state can be considered as the 'north' in which the yellow lights and green lights of remaining directions are 0's and red lights of remaining directions are 1's (where '1' indicates ON states and '0' indicates OFF). When the time is less than 10 seconds to the initial point the red and green lights are made OFF i.e. made low. the yellow light of north direction is ON i.e. made high and After 10 seconds the yellow light is made low and the green light is made high. This goes until 39 seconds.

Once the time reaches 40 seconds the red light is made high with remaining two to be low. The state is then initialized to 'west' and the timer is once again set to zero. The state can be considered as the 'west' in which the yellow lights and green lights of remaining directions are 0's and red lights of remaining directions are 1's. When the time is less than 10 seconds to the initial point the yellow light of west direction is made high i.e. ON and the red and green lights are made low i.e. made OFF. After 10 seconds the yellow light is made low and the green light is made high.

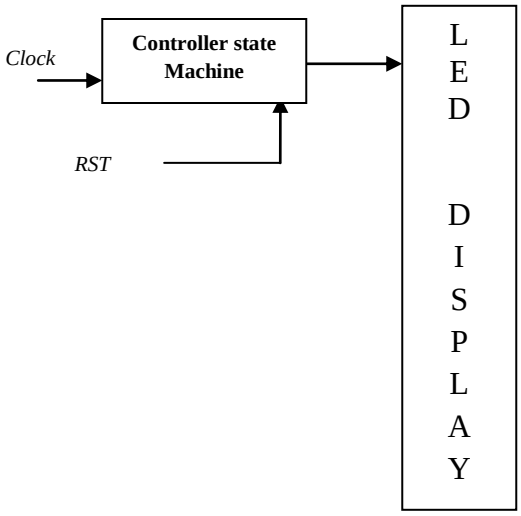


Fig: 3.1 Block Diagram of "TRAFFIC LIGHT CONTROLLER"

It goes until 39 seconds. Once the time reaches 40 seconds the red light is made high with remaining two to be low. The state is then initialized to 'south' and the timer is once again set to zero. The state can be considered as the 'south' in which the yellow lights and green lights of remaining directions are 0's and red lights of remaining directions are. When the time is less than 10 seconds to the initial point the yellow light of south direction is ON i.e. made high and the red and green lights are made low i.e. made OFF. After 10 seconds the yellow light is made low and the green light is made ON. It is for 39 seconds. Once the time reaches 40 seconds the red light is made high with remaining two to be low. The state is then initialized to 'east' and the timer is once again set to zero.

The initial state can be considered as the 'east' in which the yellow lights and green lights of remaining directions are 0's and red lights of remaining directions are 1's. When the time is less than 10 seconds to the initial point the yellow light of east direction is made high i.e. ON and the red and green lights are made low i.e. made OFF. After 10 seconds the yellow light is made low and the green light is made high. This for 39 seconds. Once the time reaches 40 seconds the red light is made high with remaining two to be low. The state is then initialized to 'north' and the timer is once again set to zero. The output can be shown on the LED Display provided on the FPGA board used for dumping the code. The logic '1' can be considered when the LED is ON and '0' when the LED is OFF.

3.1 Intersection Model

In addition to the FPGA board, a model of a four-way intersection was constructed. It contains two travel lanes and a left turn lane for each direction. To simulate sensors, momentary switches are used to detect cars waiting at a red light and cars approaching the intersection. Because of IO constraints on the Spartan 3e board, the turn lane sensor is tied to the through traffic lane sensor. This sensor can be untied and connected via the Spartan 3e' FX2 connector if the student design requires independent monitoring of the left turn sensor signal.

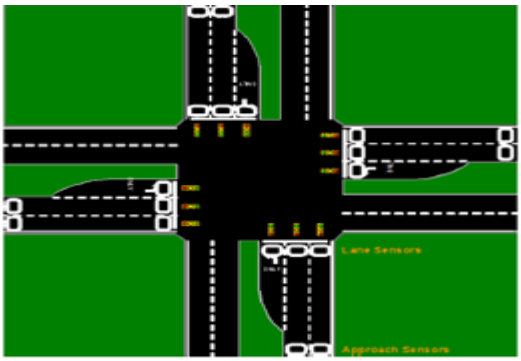


Fig: 3.2. Intersection configuration.

SPST-NO switches are embedded into the base of the model intersection at the locations marked in figure 4. These are connected to FPGA IO pins via the Peripheral Module (PMOD) connectors on the Spartan 3e 2 board. Pull-down resistors are enabled for these pins. Traffic can be simulated simply by rolling Matchbox cars over the sensor switches. The signal lights are LEDs mounted on a PCB in the shape of a light post with three signals. The posts were fabricated in house using VCU's PCB milling facilities. This reduced the cost of manufacturing the model test bench and allowed for the complex shape of the PCB. LEDs with wavelengths of 630 nm, 590 nm, and 528 nm for the red, yellow, and green signals where used because they closely match the actual colors used on a real traffic signal light. Because the FPGA IO pins cannot source sufficient current to light an LED, NPN transistors are used as simple LED drivers. Each signal post is fastened to the model base using an IDC connector. Three pins on this connector are used to control the through traffic signals. Another three are used for the left turn signal.

Fig: 3.3. Completed intersection model test bench.

Power and ground are provided with an additional two pins. These pins along with the lane and approach sensor signals are connected to a PMOD connector on the Spartan 3e board. Each of the four available PMOD connectors have enough IO pins to service a single travel direction. The base of the test bench is a 1.5 inch thick Styrofoam sheet covered with poster board. The road markings were created by adhering a large piece of paper with the plot of a four-way intersection. A wood frame surrounds the model to provide structural support and protect the edges. All materials for the base can be inexpensively purchased at any home improvement center.

Top level Design of Traffic light controller

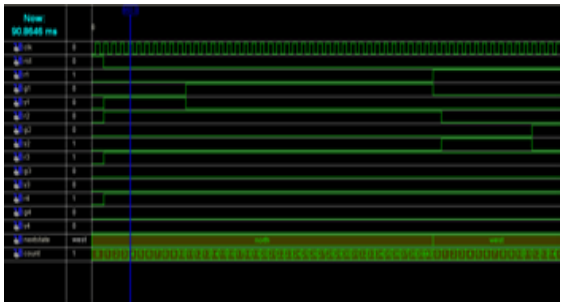


Fig 3.4: Simulation results of Top level design 1

4. CONCLUSION

The Simple Traffic Light Controller digital design project turned out to be a great success. A high rate of successful completion was observed. This can be attributed to the implementation of the controller as a series of milestones, the use of a realistic hardware test bench and allowing students to innovate the simple controller by adding functionality of their own design.

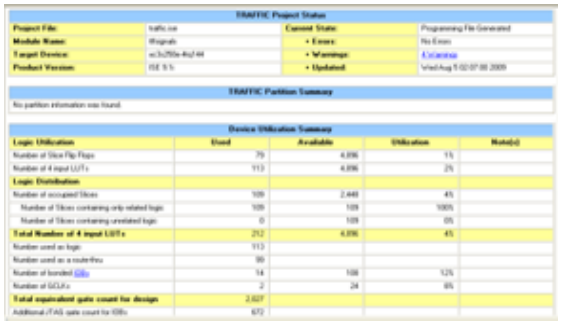


Fig 3.5: Design summary of Top level deign

More importantly, the project provided hand son experience with most of the course's learning objectives. These include the use of VHDL to model, simulate and synthesize digital systems, understanding the function, design and implementation of components of modern digital systems, understanding the fundamental concepts and applications of FPGAs, and to develop expertise in microprocessor interfaces.

The traffic light controller implemented on Xilinx Spartan 3e FPGA. The IP Core is developed using the Xilinx 9.1i. The simulation is done using ISE Simulator.

FUTURE SCOPE

Depending on the Traffic density at busy hours the chip can be designed respectively.

5. REFERENCES

[1] Simple Traffic Light Controller: A Digital Systems Design Project Jose E. Ortiz, Robert H. Klenke Electrical and Computer Engineering Department Virginia Commonwealth University Richmond, VA 23284 Email: {ortizje, rhklenke}@vcu.edu 978-1-4244-5853-0/10/$26.00 ©2010 IEEE

[2] R. L. Gordon and W. Tighe, "Traffic control systems handbook," Federal Highway Administration, Washington, DC, Tech. Rep. FHWA-HOP-06- 006, Oct. 2005.

[3] Digilent Spartan 3e2 Board Reference Manual, Digilent Inc.

[4] A. Greensted. (2009, ‚Aug.) Switch debouncing. [Online]. Available: http://www.labbookpages.co.uk/electronics/debounce.html

[5] K. Chapman, 200 MHz UART with Internal 16-Byte Buffer, Xilinx, Apr.2008.

[6] PicoBlaze 8-bit Embedded Microcontroller User Guide, Xilinx, Nov.2009.

[7] F. Poderico, PicoBlaze C Compiler, Jul. 2005.

[8] J. Cerda Boluda, M. A. Martinez Peiro, M. A. Larrea Torres, R. Gadea Girones, and R. J. Colom Palero, "An Active Methodology for Teaching Electronic Systems Design," IEEE Transactions on Education, vol. 49, pp. 355–359, Aug. 2006.

[9] R. Duckworth, "Embedded system design with fpga using hdl (lessons learned and pitfalls to be avoided)," in Microelectronic Systems Education, 2005. (MSE '05). Proceedings. 2005 IEEE International Conference on, June 2005, pp. 35–36.