International Journal of Research, Science, Technology & Management
ISSN Online: 2455-2240, Volume 18 Issue 2, February 2020

TRAFFIC LIGHT CONTROLLER

Anjali Pratap Singh, Pallavi Kesharwani

ABSTRACT

Traffic congestion is a severe problem in many modern cities around the world. To solve the problem, we have proposed a framework for a dynamic and automatic traffic light control expert system combined with a VHDL code. This adopts inter arrival time and inters departure time to simulate the arrival and leaving number of cars on roads. Each sub model represents a road that has three intersections. The paper aims to design a Traffic Light Controller using VHDL and implement the Traffic Light Controller in FPGA. The traffic in road crossings/junctions are controlled by switching ON/OFF Red, Green & Amber lights in a particular sequence. The Traffic Light Controller is designed to generate a sequence of digital data called switching sequences that can be used to control the traffic lights of a junction in a fixed sequence. The simulation results physically prove the efficiency of the traffic system in an urban area, because the average waiting time of cars at every intersection is sharply dropped when the red light duration is 65 s and the green light time duration is 125 s. Meanwhile, further analysis also shows if we keep the inter arrival time of roads A, B, and C, and change that of roads D, E, and F from 1.7 to 3.4 s and the inter departure times at the three intersections on roads A, B, and C are equal to 0.6 s, the total performance of this is the best. Finally, according to the data collected from RFID readers and the best, second and third best traffic light durations generated from the simulation model, the automatic and dynamic traffic light control expert system can control how long traffic signals should be for traffic improvement.

1. INTRODUCTION

Traffic congestion has been causing many critical problems and challenges in most cities of modern countries. To a commuter or traveler, congestion means lost time, missed opportunities, and frustration. To an employer, congestion means lost worker productivity; trade opportunities, delivery delays, and increased costs. To solve congestion problems is feasible not only by physically constructing new facilities and policies but also by building information technology transportation management systems. Their proper operation can spell the difference between smooth flowing traffic and four-lane gridlock. Proper operation entails precise timing, cycling through the states correctly, and responding to outside inputs, like walk signals. The traffic light controller is designed to meet a complex specification. That specification documents the requirements that a successful traffic light controller must meet. It consists of an operation specification that describes the different functions the controller must perform, a user interface description specifying what kind of interface the system must present to users, and a detailed protocol for running the traffic lights. Each of these requirements sets imposed new constraints on the design and introduced new problems to solve. The controller to be designed controls the traffic lights of a busy highway (HWY) intersecting a side road (SRD) that has relatively lighter traffic load. Sensors at the intersection detect the presence of cars on the highway and side road. A traffic light system is an electronic device that assigns right of way at an intersection or crossing or street crossing by means of displaying the standard red, yellow and green colored indications. A traffic light, also known as traffic signal, stop light, stop-and-go lights, is a signaling device positioned at a road intersection, pedestrian crossing, or other location in order to indicate when it is safe to drive, ride, or walk using a universal color code. Nowadays, a red light meant traffic in all directions had to stop. A yellow light meant cross-town traffic would have to slow and a green light would to go or proceed. The following project will demonstrate the use of digital combinational logic to achieve an efficient traffic light control system that may be implemented throughout the country.

2. ELECTRONIC DESIGN WITH FPGAs

In most digital designs, the circuitry can be classified by the following categories:

•**Standard products**-These products provide a functionality which is not associated with a specific application area but common to a broad range of devices. Typical parts in this category are processors and memories.

•**Application Specific Standard Products or ASSPs**-These products provide functionality which is not associated with a specific implementation, but common to an application area. Typical parts in this category are MPEG decoders.

• **Custom Logic**- This logic is associated with a specific application and is the essence of what distinct one product from another. Often this is glue logic, connecting standard products or ASSPs with each other.

There are several options on how to implement custom logic, FPGAs being one amongst them.

2.1 FPGAs [Field Programmable Gate Array]

FPGA is the abbreviation of Field Programmable Gate Array. This denotes an integrated circuit which is programmed in the field, i.e. by the system manufacturer. FPGAs can be characterized by the following items:

-High production cost

-Low design density

-Programmable fabric adds significant overhead

-Low development effort

-Low dead-time

-simplified timing

-No test vectors

-Relaxed verification

-Physical design is "hands-off"

3. FPGA TECHNOLOGY IN DETAIL

FPGAs are chips, which are programmed by the customer to perform the desired functionality.The chips may be programmed either

• **Once**: - Anti fuse technology, e.g. devices manufactured by Quick logic

• **Several times**: - Flash based e.g. Devices manufactures by Actel.

• **Dynamically**: - SRAM based e.g. Devices manufactured by Actel, Altera, Atmel, Cypress, Lucent, Xilinx.

Each technology has its own advantages, which shall be discussed only very briefly:

• Anti fuse FPGAs:

Devices are configured by burning a set of fuses. Once the chip is configured, it can- not be altered any more.

- Bug fixes and updates possible for new PCBs, but hardly for already manufactured boards.

- ASIC replacement for small volumes.

• Flash FPGAs

- Devices may be re-programmed several thousand times and are non-volatile, i.e. keep their configuration after power-off

- With marginal additional effort, the chips may be updated in the field

- Expensive

- Re-configuration takes several seconds

The CLBs form the central logic structure with easy access to all support and routing structures. The IOBs are located around all the logic and memory elements for easy and quick routing of signals on and off the chip.

• SRAM FPGAs

- Currently the dominating technology

- Unlimited re-programming

- Additional circuitry is required to load the configuration into the FPGA after power- on

- Re-configuration is very fast, some devices allow even partial reconfiguration during operation.

- Allows new approaches and applications

4. GENERAL OVERVIEW

There are several families of FPGAs available from different semiconductor companies. These device families slightly differ in their architecture and feature set, however most of them follow a common approach: A regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), surrounded by a perimeter of program. Values stored in static memory cells control all the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can reload if necessary to change the function of the device.

4.1 Traffic light constraints

The control input matrix cannot be every possible binary matrix, because there are also some constraints for the control of the traffic lights. The following traffic light constraints decrease the number of possible solutions. All lights in one group are green/red on the same time, - If the lights in one group are green, the lights in the other groups are red, - The sum of green time of the four groups must be equal to the length of the cycle time of an intersection, - Each light has a minimum green time, this minimum is equal for all lights, - Each light has a maximum green time, this maximum is equal for all lights.

5. INTRODUCTION TO VHDL

VHDL is a language for describing digital electronic systems. It arose out of the United States Government's Very High Speed Integrated Circuits (VHSIC) program, initiated in 1980. In the course of this program, it became clear that there was a need for a standard language for describing the structure and function of integrated circuits (ICs). Hence the VHSIC Hardware Description Language (VHDL) was developed, and subsequently adopted as a standard by the Institute of Electrical and Electronic Engineers (IEEE) in the US.

5.1. Describing Structure

A digital electronic system can be described as a module with inputs and/or outputs. The electrical values on the outputs are some function of the values on the inputs. A module F has two inputs, A and B, and an output Y. Using VHDL terminology, we call the module F a design entity, and the inputs and outputs are called ports.

One way of describing the function of a module is to describe how it is composed of sub-modules. Each of the sub-modules is an instance of some entity, and the ports of the instances are connected using signals. This kind of description is called a structural description.

5.2. Discrete Event Time Model

Once the structure and behavior of a module have been specified, it is possible to simulate the module by executing its behavioral design.

5.2.1. Describing Behavior

In many cases, it is not appropriate to describe a module structurally. One such case is a module which is at the bottom of the hierarchy of some other structural description. For example, if you are designing a system using IC packages bought from an IC shop, you do not need to describe the internal structure of an IC. In such cases, a description of the function performed by the module is required, without reference to its actual internal structure. Such a description is called a functional or behavioral description.

5.2.2. Encryption

This is done by simulating the passage of time in discrete steps. At some simulation time, a module input may be stimulated by changing the value on an input port. The module reacts by running the code of its behavioral description and scheduling new values to be placed on the signals connected to its output ports at some later simulated time. This is called scheduling a transaction on that signal. If the new value is different from the previous value on the signal, an event occurs, and other modules with input ports connected to the signal may be activated.

The simulation starts with an initialization phase, and then proceeds by repeating a two-stage simulation cycle. In the initialization phase, all signals are given initial values, the simulation time is set to zero, and each module's behavior program is executed. This usually results in transactions being scheduled on output signals for some later time.

In the first stage of a simulation cycle, the simulated time is advanced to the earliest time at which a transaction has been scheduled. All transactions scheduled for that time are executed, and this may cause events to occur on some signals. In the second stage, all modules which react to events occurring in the first stage have their behavior program executed. These programs will usually schedule further transactions on their output signals. When all of the behavior programs have finished executing, the simulation cycle repeats.

If there are no more scheduled transactions, the whole simulation is completed. The purpose of the simulation is to gather information about the changes in system state over time. This can be done by running the simulation under the control of a simulation monitor. The monitor allows signals and other state information to be viewed or stored in a trace file for later analysis. It may also allow interactive stepping of the simulation process, much like an interactive program debugger.

5.2.2.1. VHDL/VERILOG is Like a Programming Language

The behavior of a module may be described in programming language form. If you are familiar with the Ada programming language, you will notice the similarity with that language. This is both a convenience and a nuisance. The convenience is that you don't have much to learn to use these VHDL/VERILOG facilities. The problem is that the facilities are not as comprehensive as those of Ada, though they are certainly adequate for most modeling purposes.

5.2.2.2. Entity Declarations

A digital system is usually designed as a hierarchical collection of modules. Each module has a set of ports which constitute its interface to the outside world. In VHDL, an entity is such a module which may be used as a component in a design, or which may be the top level module of the design. The entity declarative part may be used to declare items which are to be used in the implementation of the entity. Usually such declarations will be included in the implementation itself, so they are only mentioned here for completeness.

Also, the optional statements in the entity declaration may be used to define some special behavior for monitoring operation of the entity. The entity header is the most important part of the entity declaration. It may include specification of generic constants, which can be used to control the structure and behavior of the entity, and ports, which channel information into and out of the entity. The actual value for each generic constant is passed in when the entity is used as a component in a design. The entity ports are also specified using an interface list, but the items in the list must all be of class signal.

5.2.2.3. Architecture Declarations

Once an entity has had its interface specified in an entity declaration, one or more implementations of the entity can be described in architecture bodies. Each architecture body can describe a different view of the entity. The declarations in the architecture body define items that will be used to construct the design description. In particular, signals and components may be declared here and used to construct a structural description in terms of component instances.

5.2.2.4. Signal Declarations

Signals are used to connect sub modules in a design.

5.2.2.5. Component Declarations

An architecture body can also make use of other entities described separately and placed in design libraries. In order to do this, the architecture must declare a component, which can be thought of as a template defining a virtual design entity, to be instantiated within the architecture. Later, a configuration specification can be used to specify a matching library entity to use.

6. VLSI BASED DESIGN AND IMPLEMENTATION OF TRAFFIC LIGHT CONTROLLER

6.1. AIM

The aim of the project is to design a Traffic Light Controller using VHDL and implement the Traffic Light Controller in FPGA.

6.2. THEORY

The traffic in road crossings/junctions are controlled by switching ON/OFF Red, Green & Amber lights in a particular sequence. The Traffic Light Controller is designed to generate a sequence of digital data called switching sequences that can be used to control the traffic lights of a junction in a fixed sequence.

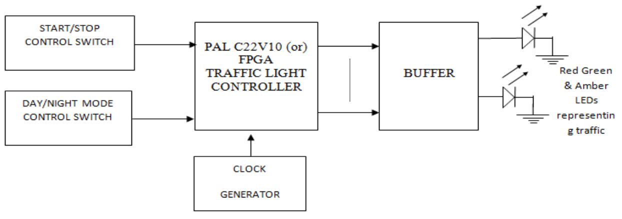

START/STOP CONTROL SWITCH
DAY/NIGHT MODE CONTROL SWITCH
PAL C22V10 (or) FPGA TRAFFIC LIGHT CONTROLLER
CLOCK GENERATOR
BUFFER
Red Green & Amber LEDs representin g traffic

Figure 1: Block diagram of Traffic Light Controller

7. DESIGN

Our task is to implement a controller for a traffic light. The sequence of lights is to be as follows:

N-S Light	E-W Light
R	R
G	R
Y	R
R	R
R	G
R	Y

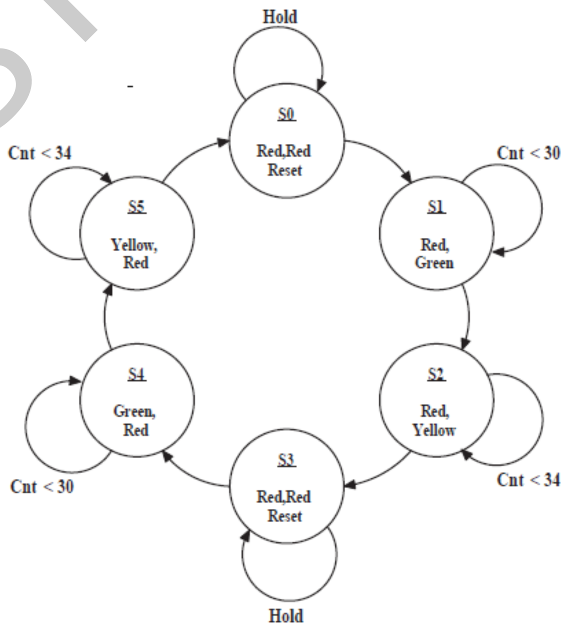
Where the N-S light controls the flow of traffic arriving at the intersection from the north and the south and the E-W light controls traffic arriving at the intersection from the east and west. The red-red condition provides a bit of a safety margin.

However, traffic controllers also have timing requirements. For example we can specify that a Green light is required to stay on for 30 seconds and the Yellow light for 4 seconds. If we add the requirement that both lights are Red for only one second, the timing for the traffic light controller is completely specified.

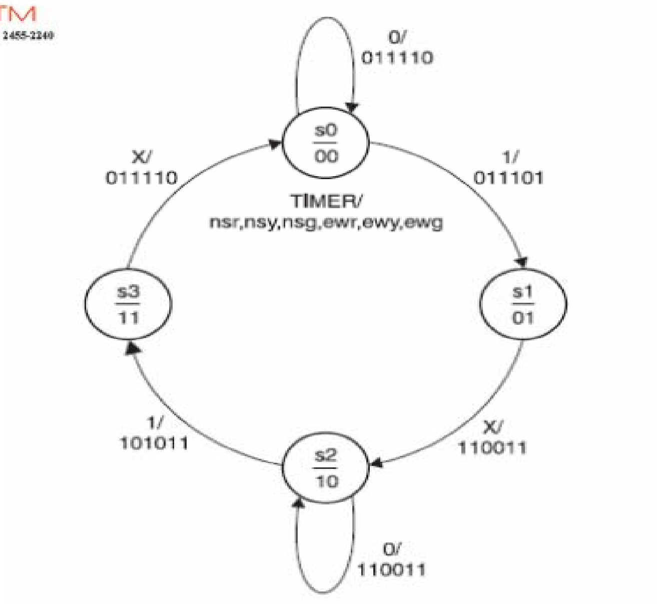
Ordinarily, the above sequence is to be repeated without alteration, but to allow a traffic officer to manually direct traffic, we provide a HOLD control. If the HOLD signal is asserted, the normal sequence of lights proceeds until it arrives at either of the two points in the sequence where both lights are red, and then both lights are held red until the HOLD signal is released.

7.1. Implementing the Timing

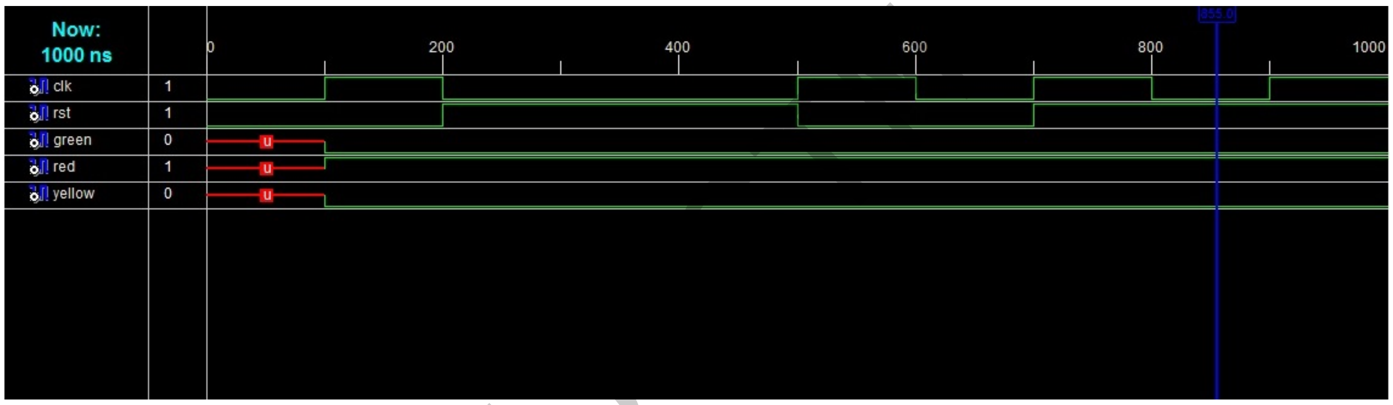
There are a number of different ways to implement such a controller. Since the clock period for the controller is one second, one way to keep the green light on for 30 seconds would be to replace each of the two states that set the green light on with a linear sequence of 30 states, each of which has the same outputs as the state that was replaced. Similarly, we would replace each of the states that set a yellow light on with 4 equivalent states. Thus, with the two states that are responsible for Red-Red outputs, our controller would require a total of 70 states. While this only requires 7 flip-flops, it would be very difficult to design since it is next to impossible to use K-Map techniques to minimize functions of more than 6 variables. Another way to implement the controller is to use a separate counter (incremented once each clock cycle) to keep track of the number of seconds that have elapsed. The counter is enabled when a state that produces a green or a yellow light is entered, and the machine stays in the state until the counter reaches the appropriate value. With this, our state diagram can be drawn as in Figure 1. Since the state in which a yellow light is turned on always follows a state in which the green light was on, we don't have to reset the counter between these two states. We merely stay in the state that produces a green light until the counter reaches 30, and then we move to the state that enables the yellow light and stay in this state until the counter reaches 34.



State Diagram for the Traffic Light Controller



8. OUTPUT



9. CONCLUSION

This paper named "TRAFFIC LIGHT CONTROLLER" undertook by the students of M.Tech (VERI LARGE SCALE INTEGERATION) under the guidance and support of our teachers.

The reason behind undertaking this very project simply lies with the fact that there are so many places where traffic congestion is very common problem.

To resolve this purpose we have made this very project, so that if such a kind of system is used then at least it may be able to sense the traffic conditions and accordingly necessary conditions can be undertaken to cope up with the adverse situation.