Real Time Traffic Light Controller Using VHDL

By Angelica Reyes

Abstract:
This project proposes the VHDL design of a FPGA based Traffic Light controller that manages the traffic on a common intersection in a road. The tools needed will be the Xilinx software, a keyboard, mouse, and laptop. The implementation will be based on the laws and behaviors of the state of Texas more specifically the city of Dallas. The focus will be around allocating appropriate light switching of an intersection to control traffic. The reason I am choosing to do this project is to bring understanding of how VLSI and FPGA technology plays a great role in improving our safety and managing our time during daily life. One of the most common issues of major cities is controlling the traffic. Traffic is not just frustrating to the public, but it also impacts the environment as well. All the vehicles being kept in an area at the same time impacts the air quality. Therefore, it is an important task for a city to have efficient traffic light controllers. Over time traffic light controllers have improved significantly. Unlike before, modern traffic lights now use several sensors to better the flow of traffic and increase the number of cars per road. A vehicle density sensor is used to count the traffic at one end of a traffic light. This sensor will allow the traffic light to remain green for longer on the side where there is more traffic therefore balancing the traffic flow. Another innovation of traffic light is the emergency vehicle sensor. This new technology allows first responders to control traffic lights during an emergency via GPS and frequency sensors. One of the most fundamental tasks to control traffic is controlling intersections where multiple streams of traffic need to be safely and efficiently crossed. Taking a step into it, we introduce the best way to control the right of way at intersections, traffic light signaling. A traffic light controller consists of three LEDs (green, yellow, and red), and a clock/timer. For the purpose of this project we will design a one-way intersection with two traffic lights. Traffic light A will control traffic between North to South and Traffic light B will control traffic going between West to East. Average time in Texas a light is green is 30 seconds between time it takes to turn red. Average time it takes for the light to go from yellow to red is 5 seconds. We will limit the seconds of this project demo design to 1 second per 5 seconds of this timing clock sequence. From Table 1, The traffic light A will stay green for 4 seconds, then yellow for 1 seconds, then will remain red for 5 seconds. The clock will then reset and start from Traffic light B where the same count will apply. Both lights will be red for 1 second allowing a delay for traffic lights to switch from A to B. At no point can both traffic lights be either yellow or green. As a result, indicating that a traffic light control will efficiently and safely process the traffic at an intersection.

Keywords: FPGA, VHDL, Xilinx.

Outputs	Output	Delay (seconds)	Light State Details
0	NS_Green	7	Green light on the North South traffic light.
1	NS_Yellow	3	Yellow light on the North South traffic light.
2	NS_Red	10	Red light on the North South traffic light.
3	WE_Green	7	Green light on the West East traffic light.
4	WE_Yellow	3	Yellow light on the West East traffic light.
5	WE_Red	10	Red light on the West East traffic light.

Table 1: Traffic Light Outputs

1. Traffic Light Controller: Design and Purpose

Traffic lights were first manually operated gas lit signals. The very first traffic light was invented in London in 1968. The development of modern traffic lights started in London but grew in the United states with automated stop and go systems. Now, a traffic light requires controller to coordinate traffic and ensure it is maneuvered smoothly and safely. Thus, the invention of the Traffic light Controller. The controller manages traffic signals inside a box called the control box. Most recently, controllers have been updated to solid state. The traffic light must be instructed exactly when it needs to change in color and signal. These changes of lights are usually coordinated in relationships so that there is only one green light at a time to avoid accidents.

The traffic light controller for this project consists of two inputs for sensor and clock and four FSM state outputs for the Green, Yellow, and Red lights in the direction North South and West East. The project is implemented using Vivado software. Traffic lights operate in relationships that are called be converted to an FSM in FPA. The finite state machine and timers design what we will have as a traffic light controller. FSM are sequential logic systems that consist of states, inputs, and outputs and rules for when to move between the states. For this design we will use four state FSM. For state 1 we have West East set to green and South North light set to Red while the enable for the red, green, and yellow lights are set to 0 and the West East light is set to green with binary 001 and South North light is set to binary 100 for a result of a red light. The light on the west east will be the primary light and will stay green while the clock is active and until the sensor is high on the North South side of the two-way intersection. The sensor in this design will be set to high as an implementation of the switch between Green right of way for the West east and north south lights. This sensor will trigger the next state to be initiated and it will now become the current state. The second state is WE_Y_NS_R for when the sensor is high, and the West East light is set to Yellow for three seconds and North South light will stay Red for this state as well. The timer will count for 3 second until it is through and the next state is called, WE_R_NS_G and the timer for this state is started. The third state is created to implement the west east light to be set to red and the North south to be set to green. Similarly, the timer for this state is stated continues for 10 seconds before going to the next and fourth state of WE_R_NS_Y. At this fourth state the west east light is set to red and north south light is set to yellow for 3 seconds. Again, the timer for this state counts through 3 seconds and then the first state becomes the next state, thus the current state and primary state.

2. Clock and Timing

This project implementation of a Traffic light controller works just as a real life one would. The most common traffic lights work with simple timers. Depending on the level of traffic there are sensors that can adjust the timers to help elevate traffic during busy road times. The traffic light will cycle thorough red, yellow, and green intervals to make sure there is a consisted flow of traffic in all directions of the intersection. Timer based software is imperative to make sure busy areas that have consistent heavy traffic move to elevate that traffic.

The traffic sensors work together with the clock of the system in order to move traffic in areas where the traffic can be chaotic and even unpredicted. Timer based system will adjust the clock to give more time to the intersection side with the heaviest traffic. These system timers are very dependent on the smart sensors that signals the timer of the signal system and detects when a vehicle is there. These types of sensors are usually used when there is one area of an intersection that doesn't get used as

much as the others. Most often you will see these sensors on turns that have to cross major highways or farm roads. The sensor will sense that a car is preset and then adjust the timer in the system to allow the turn signal to activate and all other lights to go low, or red. When the timer triggers red, it starts counting down the most common time for these is 10seconds. Once the timer reaches 10 seconds it will turn the Yellow light on high and the green light will now be low. Similarly, the Yellow light will reach 3 and then reset itself to low and the red light will go high. This is when the system kicks back into the original timing program that it has already been set to.

3. System Architecture: The Structure of the Controller

The traffic light controller for this project requires VHDL capable software. Here we use 2019 Vivado Xilinx software. The design consists of one main source code and a test bench. An FSM diagram was constructed before beginning the code so that all states of possible outcomes are included in the output of the project. The project FSM is the implementation of a traffic light controller. Traffic light controllers can be used for a range of intersections but for the purpose of simplicity and demonstration this design will use the idea of a two-road traffic light controller. A two-way traffic light controller consists of two traffic light systems. One traffic light controls traffic going North and South and the second traffic light controls traffic going East and west. Figure1 shows this design in a block diagram. The clock and reset control the timer. The sensor will be connected to the traffic light controller. The Clock divider will set the appropriate duration for the timer of each light. The traffic light controller controls the two-way lights at the intersection and with the state machine controls which light is Red, Green and Yellow and for how long. FSM is typically used in programming designs using VHDL for sequential logic implementations. The traffic light controller being designed has two intersection that will be demonstrated in the project simulation and testing phase. To demonstrate the project a test bench will be essential and a Truth Table of all possible outcomes. The Truth Table and simulation should match in results verifying an error free traffic light controller design.

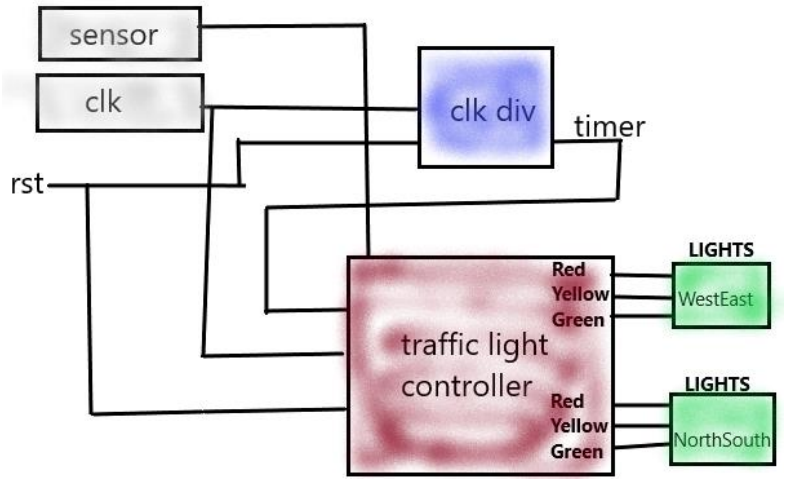

Figure1:

4. Software Description and State Machine

This Traffic light controller design implements a finite state machine in VHDL. Optimal timing on the controller is designed to minimize vehicle delay time and wait time. For this design the very core is the

Finite state machine that directs the traffic to light green on the road that is causing the most traffic. This FSM is designed to give priority to the North and South light when there is a car. The vehicle sensor senses the car and sets the sensor to high. This triggers the state for West East light to go yellow and subsequentially red. This will allow traffic on the North and South to go Green and therefore elevate the traffic on that road. Take a look at Figure2b where the Graphic will show you how the intersection is set up. You can apply this to real life intersection as this design was grounded form a standard two-way intersection in the United States.

VHDL is a language strong and rich in typing. In it is derived from ada programming language and the requirement is what makes is verbose compared to Verilog. VHDL was ideal for this project because of its ability to emphasize semantics in the design. In VHDL it is easy to implement a test bench for the purpose of demonstrating and testing the design of the Traffic Light Controller. Further, these are many resources available for programming FSM in VHDL.

The FSM is designed so that it will give primary Green light to the West East light. Only when the sensor is high will it trigger the North South light to go Green. After finalizing the FSM, I constructed a written VHDL code for the traffic light controller. The FSM works with 4 states in VHDL. The first state is when West East is Green, North South is Red. This state will turn the enable for Red, Green, and Yellow to low for north south allowing the West east to go green. The second state is when West East is Yellow, North South is Red. This state will turn the enable for Red, Green to go low and yellow to go high for north south allowing the West east to go green. The third state is when West East is Red, North South is Yellow. This state will turn the enable for Red, Green, and Yellow low for west eat allowing the north south to go green. The fourth state is when West East is Red, North South is Green. This state will turn the enable for Red, Green, and Yellow high for north south allowing the West east to go red. The FSM is coded in VHDL with if else when statements for each state.

This design also uses a reset, clock, sensor, and timer. The clock and reset control the time of when each light goes high or low. The timer controls the duration of each light and how long it will be high or low. The sensor is a vehicle smart sensor that is very widely used now in days in modern traffic light controllers. This sensor detects the amount of traffic in a road in order to adjust the traffic light controller to elevate traffic where it is mostly needed.

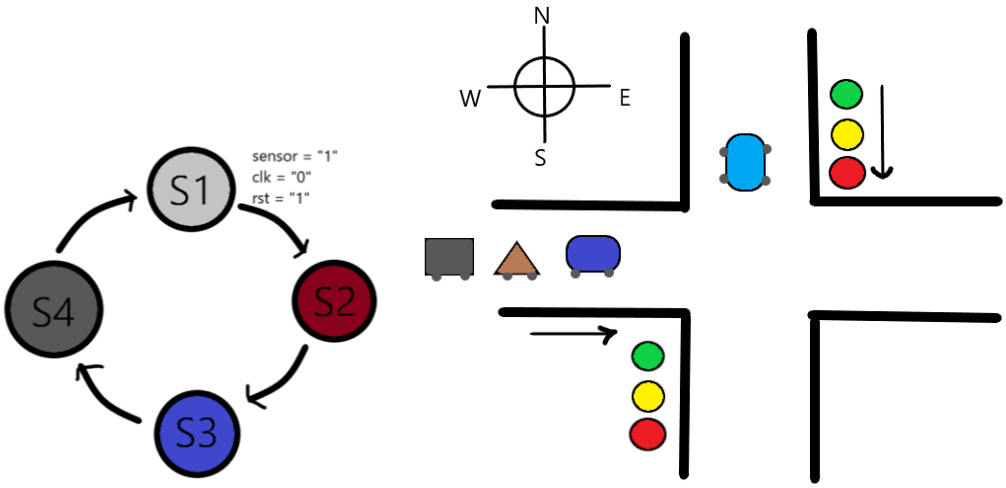

Figure2: (a) FSM Diagram (b) Graphic representation of the Traffic light controller with two directions North South and East West.

5. Sensors

Traffic lights have timers and sensors which help direct traffic flow at an intersection. When a vehicle is detected at a light, the sensor is triggered, and the signal is sent to the traffic light controller. A sensor is very useful on roads where there is inconsistent traffic flow. They use a variety of technology but the one we will use for this project will be using VHDL in a FPGA technology. When a car arrives at an intersection the sensor can detect if there are too many cars stacked up or when a car has arrived at the stop light. The most common technology for these sensors is inductive loops. Inductive loops are coils of wire embedded in the roads surface that are triggered when a vehicle is on one. The installation of the sensor is to lay the asphalt and then cut a groove with a saw for the wire. The wire is placed in these sawed grooves and sealed with rubber compound. You can often see this in daylight as a thick line at the stop light. The inductive loops work by detecting a change of inductance. For this project, we set the sensor to active manually for demonstration. This high state of the sensor will implement a change of inductance and thus trigger the sensor to high. The sensor will trigger the state when North and South light has sensed a vehicle. This will then send a signal to the controller and give priority to the North and South road of the two-way intersection. The primary light in this design is the West East light that is always green until the sensor has detected a vehicle on the North South road of the intersection. In order to elevate the traffic from north and south the WE_R_NS_G is the goal. As you on Figure 3 can see the state will turn the red light on the West East and the Green light on the North South.

(a)

```
when WE_R_NS_G =>
    WestEast <= "100";-- Red light on WestEast
    NorthSouth <= "001";-- Green light on NorthSout
    enableR <= '1';
    enableY <= '0';
    enableG <= '0';
```

(b)

```
if((delay_count = x"9") and enableR ='1') then
 timerC <= '1';
 timerB <= '0';
 timerA <= '0';
 delay_count <= x"0";
elsif((delay_count = x"2") and enableY= '1') then
 timerC <= '0';
 timerB <= '1';
 timerA <= '0';
 delay_count <= x"0";
elsif((delay_count = x"2") and enableG= '1') then
 timerC <= '0';
 timerB <= '0';
 timerA <= '1';
 delay_count <= x"0";
else
 timerC <= '0';
 timerB <= '0';
 timerA <= '0';
```

Figure 3: (a) Example of State 4. When a sensor is detected, the traffic light controller will adjust. (b) the loop that will assign the color of the lights to high or low for each state.

6. Capabilities of Traffic Light Controller

The increase of population is causing for an increase demand in the handling of traffic capacity of roads. Reducing collisions and waiting time for both vehicles and the safety of civilians. Traffic light controllers help ensure safe travel within the intersections of roads. These are necessary even in areas where traffic is not constant. They not only make traffic a lot safer but also control traffic for pedestrian. Helping to

reduce the number of accidents and make collisions at intersections less frequent is just the basics of the capabilities of traffic light controllers.

Traffic light controllers are very important and offer maximum control at intersections. For this a lot of them include sensors. Some sensors help make a turn lane light alleviate traffic. Others help pedestrians cross an intersection. The common function of a sensor at Traffic light controller is to give the right of way to conflicting movements of traffic at any intersection. This is done by allowing the conflicted side of the intersection to alternate assigning timers in order to make more time for the intersection with the traffic. These sensors may also interrupt extremely packed crossings that would not be able to move at a safe pace otherwise. Timing is the most important part of the traffic controller. Without proper timing the traffic signals will not work efficiently, and their purpose and capabilities are diminished. Properly timed traffic light controllers handle the capacity at an intersection. Traffic engineers are the force in charge of keeping up with the traffic light controllers to ensure they are working properly.

Modern traffic light controllers are solid state and can be controlled and troubleshooted remotely. With vehicle travel increasing across the world it is much faster for engineers to fix a troubled traffic controller remotely when before they had to find a way through traffic sometimes making it impossible to get it fixed on time. As the increase in road usage increases cities are finding the best way for the control off the traffic and upgrading their traffic controllers. Just to clarify the basics of a traffic light system we look to the design. When the light is green this means cars facing the traffic light that is green are allowed to proceed in the direction denoted by the street sign. When the light is illuminated Yellow it is an indication to all drivers to slow down and stop for the switch to red light. When the light is illuminated red, it indicates all the vehicles facing this light to stop and take caution for oncoming traffic.

Traffic lights also have signals that are implemented depending on the type of intersection. Some signals are for turn left or right. Other signals are to caution a change of speed. In the system for this project design signals are not implemented but it is just one way that the project can be improved. Emergency mode is another feature of modern traffic light controller. This mode is used when important vehicles are crossing such as ambulance and fire trucks. These emergency vehicles often have a remote that sends a signal to the traffic light that they are approaching. The traffic light will then prioritize the emergency vehicle and allow the light to turn Green for them.

Hardware of traffic lights is also another important part of traffic light controller. Modern traffic light systems use computer-controlled systems. These systems began to evolve with computers in the early 1960s. The computers that are used to control these systems are programmed with city-controlled software managed by traffic engineers. Although Traffic lights are manufactured in big factories, they are installed and wired at the site of the intersection. Initially there simply existed no traffic control. People negotiated their own way through traffic which causes a lot of accidents. This is why traffic lights controllers is very necessary and is a technology that is a demand and continues to grow throughout the years.

7. Testbench: Testing and Simulation

A test bench is used in this design to test the Traffic light controller. The design of the test bench begins with adding a component for the inputs and outputs of the traffic light controller. Here we have three inputs for sensor, clock and reset. The outputs will be the results of the design which in this case are the lights for the two-way intersection of North South and West East. Once the component for the

controller is added we will end the component and begin to set the inputs and outputs for the test bench code. Similarly, we will need to declare a signal for each of these inputs and outputs. We have a signal for sensor, clock, reset, West East, and North South. An additional signal is added for the pulse of the controller. This will be the clock period that will start at 10 nano seconds. Set each port map too for the inputs and outputs. If you forget one there will be a syntax error. Simply look over and make sure all the inputs and outputs are being included. Begin the process by naming a new process and the code after beginning will commence. We will be using a clock divider for the test bench. The clock will begin at 0 then go high for half a clock period which is 5 nano seconds. To demonstrate the sensor and how it works with the North and South light we will include in the test bench a section where the sensor is set to high. This will also set the reset to high. The sensor will cycle through three clock periods before going back to low.

Now that the test bench is written what's left is to verify that the project design is working as it should for the two-way intersection traffic light controller. For this we will use the truth table shown on Figure3. The truth table is constructed after and from the FSM State diagram on Figure2. This shows all the possible outcomes of the traffic light controller and is important to verify that the design is working as intended. The FSM is intended to implement the sequential logic system of the traffic light controller and will be demonstrating the binary outputs of the Red, Green, and Yellow lights. This will be for both North South and West East lights. Furthermore, this two-way intersection will be implemented and tested via the test bench.

States				
	WestEast	NorthSouth	WestEast	NorthSouth
S1	WE_Green	NS_Red	001	100
S2	WE_Yellow	NS_Red	010	100
S3	WE_Red	NS_Yellow	100	010
S4	WE_Red	NS_Green	100	001

Figure 4: Truth table for states to be used during simulation and testing.

To begin the simulation, go first make sure there are no syntax error left on the traffic light controller or test bench. Once you have made sure all errors are cleared it is time to run the simulation. By clicking on the simulation and selecting Run Behavioral simulation we will now see a clock diagram of our traffic light controller design. Test all possible outcomes by doing a side by side testing with the truth table. Before doing so setup the wave file to run for 10 seconds. Over to the scope, drag in the North South and West East outputs. Convert these to Binary in order to be able to easily compare with the truth table. Make sure that the Sensor, clock, and reset are included. These are all very important in checking if the Traffic light controller design will be working as it should. Run all and a simulation will be produce as shown on Figure4.

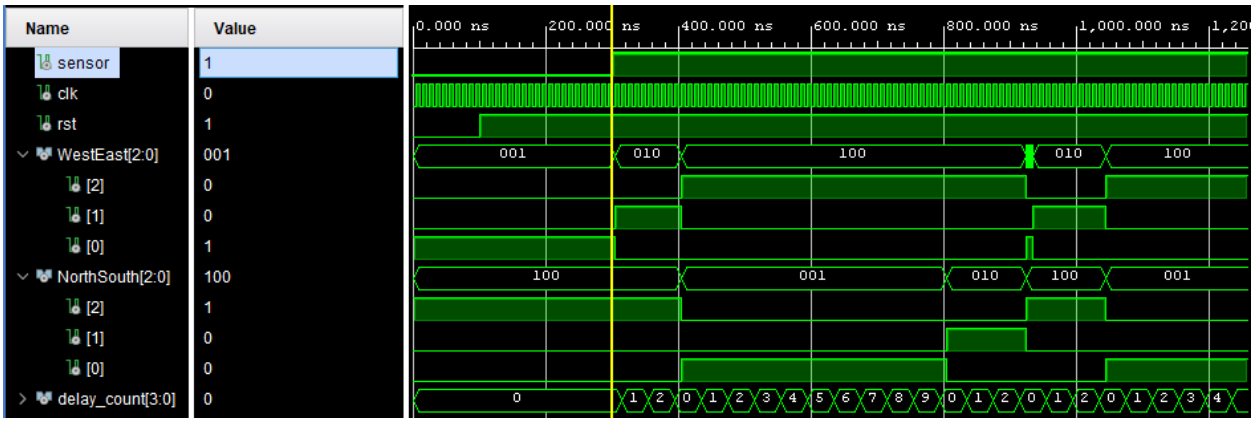

Figure5: Simulation of the traffic light controller in VHDL.

Checking the simulation, we will look at the first state when the first state when West East is green the North South is Red. The clock will pulse high and low for as long as the simulation is being ran. The sensor will be off set to 0 when the WestEast light is green. The binary 001 is for green, 010 is for yellow, and 100 is for red. To verify the second state, click on to make the sensor high set to 1. This will take us to the next state of WE_Yellow and NS_Red. The WE light will be yellow for 3 seconds before turning red. The next state is then the WE_Red and NS_Green where we now have a green light at the light where the sensor was triggered. The light will remain green for 10 seconds before going to the next state. The next state will be NS_Yellow and WE_Red. This is the in between moment before the West East light goes back to green. Finally, the FSM is reset to the first state of WE_Green and NS_Red and the sensor reset, and clock are all reset to their opposite values. At this time the cycle will start again and can be tested by toggling the values of the inputs and outputs using the arrow keys on your keyboard. Throughout the running of simulation, the 1nano second delay that we made on the controller code is being ran for 10seconds. It resets every 10seconds to keep pace of the controller's duration when in the middle of simulation.

8. Conclusion

In this design of a Traffic light controller there are five key sections: timing/clock, FSM states, controller for lights, a sensor, and testbench. The libraries required are library IEEE ,use IEEE.STD_LOGIC_1164.ALL, and use IEEE.STD_LOGIC_UNSIGNED.ALL. The timing is the heart of a traffic light controller. Without appropriate distribution of time the traffic light controller will not function. In this project a 50MHz timer is used to initiate the pulse of the traffic light controller. A process for clock is designed to include a delay of one nano second between pulses. The clock works by triggering the enable clock to high when the counter for the timer starts. The clock also initiates the process for the control of the lights and when they go Red, Yellow, and Green. This section of the code consists of if else statements that will store binary for all four FSM states. Each light color has an enable that will have to be turned on or off depending on the FSM State. When the clock is on the rising edge the enable for red, yellow and green are all high. The delay counter starts, and the four possibilities are coded. The first binary for 100 denotes when the light is red. This is distributed through the traffic light controller depending on the state. The binary of 010 denotes a yellow light which again is used throughout the FSM states of this traffic controller design. The third binary is for 001 which turns the light to green. Finally, if the system is not in use then all lights will be disabled thus the binary of 000.

The third part of the traffic light controller source code is for the states. Implementing FSM in VHDL is a way to use sequential circuits with Vivado. FSMs consist of a set of states that have inputs and outputs and set of rules per state and rules of when to move from one state to the next, and back if necessary. In this design we use a clock and reset to initiate the inputs of the FSM sequential logic. Beginning the process with the reset set to low, the primary state is WE_G_NS_R. In this state WestE ast light is the primary light that is green while North South is Red. Every time the clock pulses the program design goes to the next state. The next state then becomes the current state. The current state is processed with a set of timers that will be used in the enables of the red, yellow, and green lights.

When West East is green and South North light is Red the enable for the lights are off and the West East light is loaded with binary 001 for green and South North light is loaded with binary 100 for red light. This light on the west east will remain green until a sensor detects a vehicle on the North South light. The sensor then goes to a high value 1 and the next state is called. The next state is WE_Y_NS_R where West East light is Yellow for 3 seconds and North South light remains Red for the same time. When the 3 second timer is up the next state is called for, WE_R_NS_G and the timer for this clock is activated. This state is for when West east is red and the light at North south is green. Once the timer is up after 10seconds it goes to the next state of WE_R_NS_Y. At this state West east is red and North south is yellow for 3 seconds. At the end of the timer the initial state becomes the next state.

Finally, the last step and code for this project is the test bench. The test bench for this project is simple. The libraries required are library IEEE and use IEEE.STD_LOGIC_1164.ALL. The test bench begins by including a component for the inputs and outputs of the traffic light controller code. Here we have three inputs for sensor, clock, and reset. All three inputs are set to be std logic inputs. Two outputs are included in the component for West East and North South. These outputs are std logic vectors from 2 down to 0. Next the test bench declares the signals that will be used for simulation. Similarly, the signals are three inputs for sensor, clock, and reset. All three inputs are set to be std logic inputs set to zero. Outputs are also declared as signals for West East and North South. These outputs are std logic vectors from 2 down to 0. A new constant for clock period is used to control the wait period between simulation results. The delay of 10 seconds is set as the constant for the clock delay period. The test bench process begins with the clock set to 0 then to high after a wait period. The process than begins and resets to 0 as well as sets the sensor to 0. In order to be able to test the sensor, we add a sensor set to high and the reset to high. The sensor is then set to low and the process continues until the time is up.

Questions for the conclusion:

1. How much have you implemented in this project?

This project is the implementation of a complete Traffic Light Controller. It includes two separate files required to get the project to process successfully.

2. What is left to be implemented that you could not do?

Some things that can be implemented that I did not do add more sensors, signals, or roads to this Traffic Light controller. If anyone wants would want to continue the project, they can do so my adding to the FSM design and then code the additional states on to the traffic light controller.

3. What are some difficulties and challenges faced for this project?

Some difficulties that I faced while creating the project was finding ways to successfully time the traffic light controller. Timing is the heart of a traffic light controller and without proper timing the traffic light controller is not successful. Another difficulty I had was with the test bench. It took me hours to program it in order to be able to test the results of the traffic light controller code against the truth table and simulation.

9. About the Author

Angelica Reyes is a student at the University of North Texas in the college of Computer Engineering. She is currently in her last year as a senior in the university and will be getting her second degree for the focus in computer engineering and science. She will be receiving her bachelor's degree in Computer Engineering in December Fall of 2020. Her goal is to be able to get a toe into the FPGA industry and combine her 5-year knowledge of working as a Data Analyst in software company for future career possibilities.

REFERENCES

1. Popa, Bogdan. "How Traffic Light Control Systems Work." *Autoevolution*, 15 Jan. 2012, www.autoevolution.com/news/how-do-traffic-light-control-systems-work-41839.html.
2. Ranea, Constantin and Popsecu, Marius. "Solutions for Traffic Lights Intersections Control." *University of Craiova*, April 2010,www.researchgate.net/publication/262392270_Solutions_for_traffic_lights_intersections_control
3. "The Texas Highway Man - Texas Traffic Laws (and Good Driving Habits)." *Texas Gov*, 2020, www.texashighwayman.com/laws.shtml.
4. "Galak Electronics." *Galka*, 2020, www.galakelectronics.com/VG-305B.htm?gclid=Cj0KCQiAwMP9BRCzARIsAPWTJ_E2tubqmTOzmBV54cuZ9oFWG1gZY5Yd47BRH9nK5FNQy98XnzEdKL8aAmDeEALw_wcB.
5. "Traffic Light Controller." *Traffic-Lite-Control*, 2020, www.lightcontroller.net.
6. "VHDL Code for Traffic Light Controller." *FPGA4student.Com*, 2020, www.fpga4student.com/2017/08/vhdl-code-for-traffic-light-controller.html.
7. Gadawe, Nour. "Design and Implementation of Smart Traffic Light Controller Using VHDL Language | T. Gadawe | International Journal of Engineering & Technology." *Sciencepubco*, 15 Dec. 2019, www.sciencepubco.com/index.php/ijet/article/view/29478.
8. "Digital Circuits and Systems - Circuits i Sistemes Digitals (CSD) - EETAC - UPC." *Circuits i Sistemes Digitals*, 2020, digsys.upc.es/csd/P06/P6_T/TLC/P6.html.
9. "How Traffic Signal Controllers Work, Part 1: An Overview of Controllers." *North Star Highways*, 19 Oct. 2016, northstarhighways.wordpress.com/2016/09/10/how-traffic-signal-controllers-work-part-1-an-overview-of-controllers/#:%7E:text=1)%20A%20dial%20with%20keys,gears%20that%20can%20be%20installed.&text=2)%20As%20the%20keys%20pass,activates%20a%20solenoid%20or%20motor.
10. Popa, Bogdan. "How Traffic Light Control Systems Work." *Autoevolution*, 15 Jan. 2012, www.autoevolution.com/news/how-do-traffic-light-control-systems-work-41839.html.
11. Hillhouse, Grady. "How Do Traffic Signals Work?" *Practical Engineering*, 14 May 2019, practical.engineering/blog/2019/5/11/how-do-traffic-signals-work.
12. "FSM Design Using Verilog :: Electrosofts.Com." *Electrosofts*, 2020, electrosofts.com/verilog/fsm.html.
13. "Implementing a Finite State Machine in VHDL - Technical Articles." *Technical Articles*, 24 Feb. 2018, www.allaboutcircuits.com/technical-articles/implementing-a-finite-state-machine-in-vhdl.
14. "State Machines." *Xilinx*, 2020, www.csit-sun.pub.ro/courses/Masterat/Xilinx%20Synthesis%20Technology/toolbox.xilinx.com/docsan/xilinx4/data/docs/xst/hdlcode15.html.

15. “Possible Issue with Vivado Synthesis Encoding State Machines.” *Electrical Engineering Stack Exchange*, 23 July 2018, electronics.stackexchange.com/questions/387156/possible-issue-with-vivado-synthesis-encoding-state-machines.