University of Tripoli

Faculty of Engineering

Department of Computer Engineering

A graduation project is submitted in partial fulfillment of requirements for the degree of Bachelor in Computer Engineering

Design and Implementation of a Smart Traffic Light System with Libyan License Plate Recognition on FPGA

By: Lamia Atif Ali

Supervised by:

Dr. Mohamed Muftah Eljhani

Spring 2022

Abstract

The main purpose of this project is to design, verify, and implement a real-time density-based traffic light control system with red light violation detection and automatic Libyan license plate recognition. The system measures real-time traffic density using an array of infrared (IR) sensors placed on each three-way intersection roadway. It gives priority to emergency vehicles using radio frequency identification (RFID). It detects red light violations using passive infrared sensors; upon detection, captures the violating vehicle image using a digital camera then the captured image is sent to the automatic license plate recognition system, which processes the image to localize the license plate region and recognize the digits and the Arabic characters.

Acknowledgement

I would like to express my heartfelt gratitude to Allah for bestowing upon us the knowledge, strength, and ability to complete this project. I could not have completed this task without his mercy and guidance.

I am also grateful for the support of our families, friends, and colleagues throughout the journey. I would like to extend a special thank you to those who provided extra help and encouragement.

To my supervisor, Dr. Mohamed Muftah Eljhani I am extremely grateful for your continuous guidance, support, and encouragement. throughout my project.

Thank you to everyone at the Department of Computer Engineering, especially Eng. Yusra Matug, for always listening and encouraging me.

Finally, I would like to thank everyone who helped with this project in any way, and I apologize if I forgot anyone. Thank you very much for your help

Contents

List of Figures

List of Abbreviations

RFID	Radio Frequency Identification
IR	Infrared
SOCP	System on Programmable Chip
LED	Light Emmiting Diod
MATLAB	MAtrix LABoratory
ML	Machine Learning
ANN	Artificial Neural Network
FPGA	Field Programmable Gate Array
OEM	Original Equipment Manufacturer
HSMC	High Speed Mezzanine Card
VHDL	Very High Speed Integrated Circuit (VHSIC) Hardware Description Language
ACF	Aggregate Channel Feature
Verilog HDL	Verifiction and logic Hardware Description Language
SVM	Support Vector Machine

Chapter 1

Introduction

These days, due to the expanding number of vehicles, Vehicle Traffic has ended up a major economic component in both urban and country zones, and it requires appropriate administration and monitoring to guarantee that this mass of vehicles coexists as easily as conceivable [1]. This huge increase in vehicles amount can easily cause a traffic jam. Traffic jams happen when the vehicle is not moving for a long time [2]. The traditional traffic light is no longer suitable for managing traffic jams and congestion, especially since all roads within the intersection, have a fixed period of time, whether the road is blocked with traffic or not, and does not consider emergency vehicles, considering all of these issues a smart traffic system which takes all of the previous issues into consideration is needed. Another problem is raised due to the huge increase in the number of vehicles around the world leading to varying degrees of traffic rule violations, particularly red-light violations [1]. To detain guilty parties and resolve the shortcomings and failures of human traffic administrators who cannot be in different places at the same time, efficient traffic violation detection and license plate recognition criteria are required [1]. that led us to think of designing a smart and more efficient system to overcome those issues.

The suggested system uses radio frequency identification (RFID) readers and infrared (IR) sensors to organize and control traffic, while machine learning and digital image processing are used to determine the offending vehicle's license plate number. As a system on a chip (SOPC), FPGA manages and controls density-based traffic and emergencies automatically. Another feature is proposed in our system, to avoid the blockage at the intersection, the traffic light changes, according to the received information from sensors that are installed in the roads.

Another powerful feature is implemented in our system which is red light violation detection and License plate recognition. The red-light violation detection system is implemented using IR sensors.

license plate recognition is implemented using a combination of machine learning and image processing techniques, the license plate goes through four main steps, these steps are License plate localization, Image pre-processing, license plate number extraction, and license plate number recognition. The license plate localization is done using a machine-learning technique, the image pre-processing step is a combination of digital image processing techniques that prepares the license plate for number extraction and reduces the errors that might occur during extraction, and the license number recognition is done using image classifier to identify digits and word “ليبيا". The key advantage of this license plate system over the other license plate recognition systems is that it takes into account different illumination conditions of the image and the different angles and distances from which the image was captured.

The project is divided into two parts, a smart traffic light system, and a Libyan license plate recognition system.

Chapter 2

Background

2.1 IR sensor

An infrared (IR) sensor is a type of electronic device that measures and detects infrared radiation in its surroundings. So, what exactly is infrared radiation? Infrared radiation, also known as infrared light, is electromagnetic radiation with longer wavelengths than visible light. In fact, it is said to cover wavelengths ranging from about 1 millimeter to about 700 nanometers (the nominal red edge of the visible spectrum). It is thus invisible to the human eye but can be detected as a warm sensation on the skin [3].

2.1.1 Types of IR sensors

There are two types of infrared sensors: active IR sensors and passive IR sensors. **Active infrared sensors** emit and detect infrared radiation. They have two parts: there's a light emitting diode (LED) and a receiver. When an object comes close to the sensor, the infrared light from the LED bounces off of the object and is detected by the receiver. Active IR sensors thus behave as proximity sensors, and are commonly used in obstacle detection systems such as robots. **Passive infrared sensors**, on the other hand, only detect infrared radiation; they do not emit it from an LED. Passive infrared sensors (PIR) are made up of the following components: two strips of pyroelectric material (a pyroelectric sensor); an infrared filter to block out all other wavelengths of light; a Fresnel lens that collects light from several angles into a single point; and a housing unit to protect the sensor from other environmental variables, such as temperature and humidity. PIR sensors are mainly used in PIR-based motion detectors [4].

2.1.2 IR sensors working principle

An IR sensor works the same way an object detection sensor does. The sensor typically has an IR LED & an IR photodiode, and combining these two gives way to a photo-coupler or optocoupler. The IR LED is basically a transmitter emitting IR radiations; it looks similar to a standard LED. And since the radiation it generates is not cannot be seen by the human eye; the radiation is detected by infrared receivers which are available in photodiodes form.

The infrared photodiode responds to the infrared light generated by the infrared LED. The resistance of the photo-diode & the change in output voltage is directly proportional to the infrared light. However, it is imperative to remember that IR photodiodes are not the same as usual photodiodes as they detect only IR radiation. After the infrared transmitter has produced an emission, it arrives at the object & some of the emission bounces or reflects back toward the infrared receiver. Based on the intensity of the response, the sensor output is decided by the IR receiver [4].

2.1.3 IR sensor Usages and benefits

Today, IR sensors are used in many areas, all of which tap into their myriad applications. For instance, the speed sensor is used to synchronize the speed of multiple motors. The temperature sensor is used for industrial temperature control. Passive infrared sensors, aka PIR sensors, are used in automatic door-opening systems, while ultrasonic sensors are used for measuring distances [4].

In this project, IR sensors are used to measure vehicles' traffic density levels. The main benefits of IR sensors are low power usage, their simple design & their convenient features. IR signals are not noticeable to the human eye.

2.2 What is RFID

RFID is a form of wireless communication that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency portion of the electromagnetic spectrum to uniquely identify an object, animal or person " [5].

2.2.1 RFID working principle

Every RFID system consists of three components: a scanning antenna, a transceiver, and a transponder. When the scanning antenna and transceiver are combined, they are referred to as an RFID reader or interrogator. There are two types of RFID readers -- fixed readers and mobile readers. The RFID reader is a network-connected device that can be portable or permanently attached. It uses radio waves to transmit signals that activate the tag. Once activated, the tag sends a wave back to the antenna, where it is translated into data.

The transponder is in the RFID tag itself. The read range for RFID tags varies based on factors including the type of tag, type of reader, RFID frequency, and interference in the surrounding environment or from other RFID tags and readers. Tags can be passive or active. Passive RFID tags are powered by the reader and do not have a battery. Active RFID tags are powered by batteries. Tags that have a stronger power source also have a longer read range [5].

2.2.2 Types of RFID systems

"There are three main types of RFID systems: low frequency (LF), high frequency (HF), and ultra-high frequency (UHF). Microwave RFID is also available. Frequencies vary greatly by country and region.

- Low-frequency RFID systems. These range from 30 KHz to 500 KHz, though the typical frequency is 125 KHz. LF RFID has short transmission ranges, generally anywhere from a few inches to less than six feet.
- High-frequency RFID system These range from 3 MHz to 30 MHz, with the typical HF frequency being 13.56 MHz. The standard range is anywhere from a few inches to several feet.
- UHF RFID systems. These range from 300 MHz to 960 MHz, with a typical frequency of 433 MHz, and can generally be read from 25-plus feet away.
- Microwave RFID systems. These run at 2.45 Ghz and can be read from 30-plus feet away" [5].

2.2.3 uses and benefits of RFID

"RFID systems use radio waves at several different frequencies to transfer data depending on the application and with actual distances. In health care and hospital settings, RFID technologies include the following applications:

- Inventory control
- Equipment tracking
- Out-of-bed detection and fall detection
- Personnel tracking

- Ensuring that patients receive the correct medications and medical devices
- Preventing the distribution of counterfeit drugs and medical devices
- Monitoring patients
- Providing data for electronic medical records systems” [6].

In this project, the RFID system is used to track emergency vehicles, where an RFID reader is installed on each road, and tag is set up on the emergency vehicle and the frequency range is tuned to detect emergency vehicles on the same road that the RFID reader is installed on.

Benefits of RFID systems:

- “Can identify individual objects without direct line of sight.
- Can identify individual objects without direct line of sight.
- Can identify individual objects without direct line of sight.
- Read time is less than 100 milliseconds per tag” [5].

2. 3 Verilog HARDWARE DESCRIPTION LANGUAGE

“Verilog is a HARDWARE DESCRIPTION LANGUAGE (HDL). It is a language used for describing a digital system like a network switch or a microprocessor or a memory or a flip−flop. It means, by using an HDL we can describe any digital hardware at any level. Designs, which are described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits.

Verilog supports a design at many levels of abstraction. The major three are:

- Behavioral level
- Register-transfer level
- Gate level” [7].

2.4 MATALB

“MATLAB (MATrix LABoratory) is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages” [8].

Although MATLAB is intended primarily for numeric computing, an optional toolbox uses the MuPAD symbolic engine allowing access to symbolic computing abilities. An additional package, Simulink, adds graphical multi-domain simulation and model-based design for dynamic and embedded systems.

2.4.1 MATLAB applications

“The basic data element of MATLAB as the name suggests is the Matrix. MATLAB toolboxes are professionally built and enable you to turn your imagination into reality. A toolbox is a package of functions and/or classes. They provide you with tools, generally for a specific topic.

- Statistics and machine learning (ML)

This toolbox in MATLAB can be very handy for programmers. Statistical methods such as descriptive or inferential can be easily implemented. So is the case with machine learning. Various models can be employed to solve modern-day problems. The algorithms used can also be used for big data applications.

- Curve fitting

The curve fitting toolbox helps to analyze the pattern of occurrence of data. After a particular trend which can be a curve or surface is obtained, its future trends can be predicted. Further plotting, calculating integrals, derivatives, interpolation, etc can be done.

- Control systems

Systems nature can be obtained. Factors such as closed-loop, open-loop, controllability and observability, Bode plot, Nyquist plot, etc can be obtained. Various controlling techniques such as PD, PI, and PID can be visualized. Analysis can be done in the time domain or frequency domain.

- Signal Processing

Signals and systems and digital signal processing are taught in various engineering streams. But MATLAB provides the opportunity for proper visualization of this. Various transforms such as Laplace, Z, etc can be done on any given signal. Theorems can be validated. Analysis can be done in the time domain or frequency domain. There are multiple built-in functions that can be used.

- Mapping

Mapping has multiple applications in various domains. For example, in Big data, the MapReduce tool is quite important and has multiple applications in the real world. Theft analysis or financial fraud detection, regression models, contingency analysis, predicting techniques in social media, data monitoring, etc can be done by data mapping.

- Deep learning

It's a subclass of machine learning which can be used for speech recognition, financial fraud detection, and medical image analysis. Tools such as time series, Artificial neural network (ANN), Fuzzy logic, or a combination of such tools can be employed.

Financial analysis

An entrepreneur before starting any endeavor needs to do a proper survey and financial analysis in order to plan the course of action. The tools needed for this are all available in MATLAB. Elements such as profitability, solvency, liquidity, and stability can be identified. Business valuation, capital budgeting, cost of capital, etc can be evaluated.

- Image processing

The most common application that we observe almost every day are barcode scanners, selfies (face beauty, blurring the background, face detection), image enhancement, etc. The digital image processing also plays quite an important role in transmitting data from far-off satellites and receiving and decoding it in the same way. Algorithms to support all such applications are available.

- Text analysis

Based on the text, sentiment analysis can be done. Google gives millions of search results for any text entered within a few milliseconds. All this is possible because of text analysis. Handwriting comparison in forensics can be done. No limit to the application and just one software that can do this all.

- Electric vehicle designing

Used for modeling electric vehicles and analyzing their performance with a change in system inputs. Speed torque comparison, designing, and simulating of a vehicle, and whatnot.

- Aerospace

This toolbox in MATLAB is used for analyzing the navigation and visualizing flight simulator.

- Audio toolbox

Provides tools for audio processing, speech analysis, and acoustic measurement. It also provides audio and speech feature extraction and signal transformation algorithms.

- Computer Vision Toolbox

provides algorithms, functions, and apps for designing and testing computer vision, 3D vision, and video processing systems. You can perform object detection, tracking, feature detection, extraction, and matching. You can automate calibration workflows for single, stereo, and fisheye cameras" [9].

Note. In this project, MATLAB is used to implement an automatic Libya license plate system. The image processing toolbox is used for processing and enhancing license plate images. Computer vision toolbox for labeling data set and training the license plate detector and the license number classifier.

2.5 DE2i-150 FPGA board

A field programming gate array (FPGA) is an integrated circuit made of semiconductor material that can be reprogrammed or configured by the user after purchasing it instead of the original equipment manufacturer (OEM) alone [10]. Int this project, intel altera FPGA board terasic DE2i-150 was used.

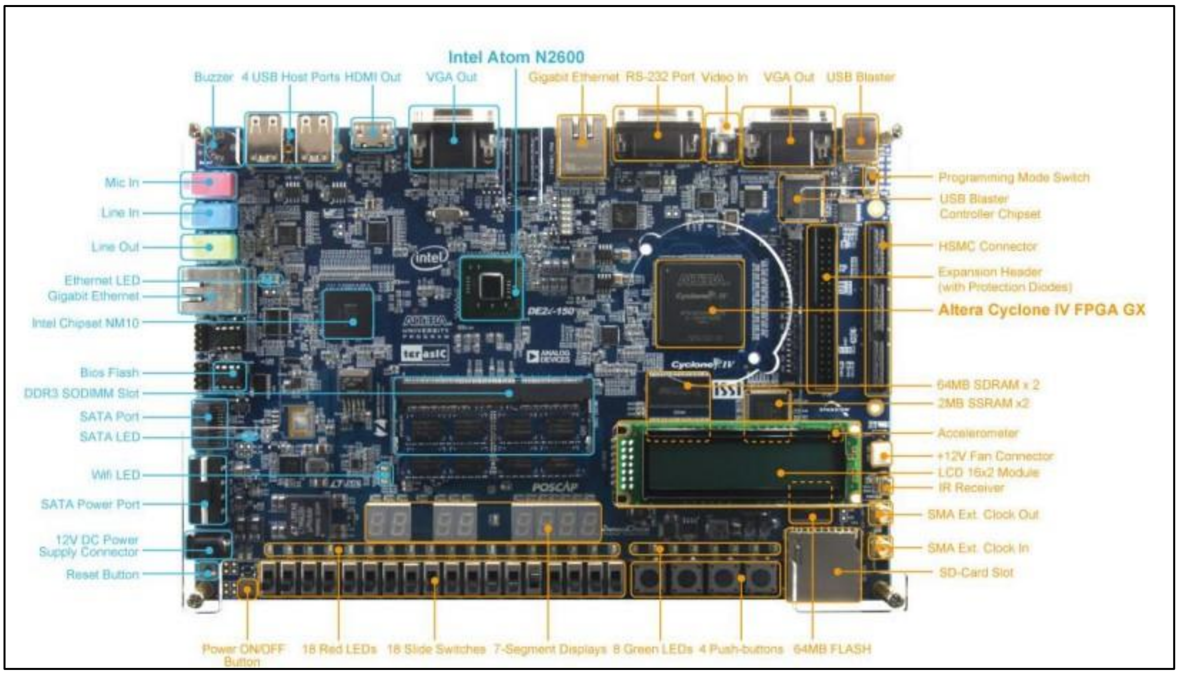


Fig. 1. The DE2i-150 board top view.

“The DE2i-150 board has many features that allow users to implement a wide range of designed circuits, from simple circuits to various multimedia projects.

The following hardware (FPGA System) is provided on the DE2i-150 board:

- Altera Cyclone® IV 4CX150 FPGA device
- Altera Serial Configuration device – EPCS64
- USB Blaster (on board) for programming; both JTAG and Active Serial (AS) programming
- modes are supported
- Two 2MB SSRAM
- Two 64MB SDRAM
- 64MB Flash memory
- SD Card socket
- 4 Push-buttons

- 18 Slide switches
- 18 Red user LEDs
- 9 Green user LEDs
- 50MHz oscillator for clock sources
- VGA DAC (8-bit high-speed triple DACs) with VGA-out connector
- TV Decoder (NTSC/PAL/SECAM) and TV-in connector
- Gigabit Ethernet PHY with RJ45 connectors
- RS-232 transceiver and 9-pin connector
- IR Receiver
- 2 SMA connectors for external clock input/output
- One 40-pin Expansion Header with diode protection
- One High-Speed Mezzanine Card (HSMC) connector
- 16x2 LCD module" [11]

2.6 ModelSim-Altera

"The ModelSim-Altera software is Altera specific and supports behavioral and gate-level timing simulations and either VHDL or Verilog HDL simulations and test benches for Altera PLDs. ModelSim-Altera was used to design and simulate the traffic system" [12].

2.7 Quartus II

"The Quartus II development software provides a complete design environment for system-on-a-programmable-chip (SOPC) design. Regardless of whether you use a personal computer or a Linux workstation, the Quartus II software ensures easy design entry, fast processing, and straightforward device programming. Quartus II was used to program the FPGA board "[13].

Chapter 3

Related Work

The proposed real-time density-based traffic light control system is designed and simulated with Verilog Hardware Description Language (HDL) and implemented on Cyclone IV GX field-programmable gate arrays (FPGA). Automatic license plate recognition is implemented using machine learning (ML) and image processing algorithms; these algorithms were programmed and tested using MATLAB software.

A project is designed to develop a density-based traffic signal system where its signal timing changes automatically after sensing the traffic density at the intersection. Traffic density is defined as the number of vehicles congested at a certain place, in this case, an intersection. The more vehicles available, the higher the traffic density is. This traffic light uses an Arduino UNO microcontroller to create an automation function together with an Infrared sensor (IR sensor) to detect the density of the traffic [14].

Another Design, Simulation, and Implementation of Intelligent Traffic Lights Based on Traffic Density is accomplished, this project is to implement a Density based traffic controlling system using InfraRed (IR) technology and ATMEGA328P [15].

The Libyan license plate recognition project is designed using a support vector machine, there are mainly three stages in this project: plate detection using vertical and horizontal histograms, character segmentation is performed through a connected-component labeling algorithm, and finally, optical character recognition (OCR) by using support vector machines (SVMs) [16].

Chapter 4

Traffic Light System

4.1 System overview

This system manages three-way intersection roadways. Five IR sensors are installed on the intersection roadways to observe the density of the vehicles. RFID readers are set up on the three roads to detect and identify emergency vehicles. The RFID reader emits radio waves and receives signals back from the RFID tag installed in the emergency vehicles. Tags, utilize radio waves to communicate their identity and other information to adjacent readers. Interference between readers and tags can prevent the system from working properly, correct setup and auditing of the frequency spectrums used by devices within the same environment can prevent potential interference issues.

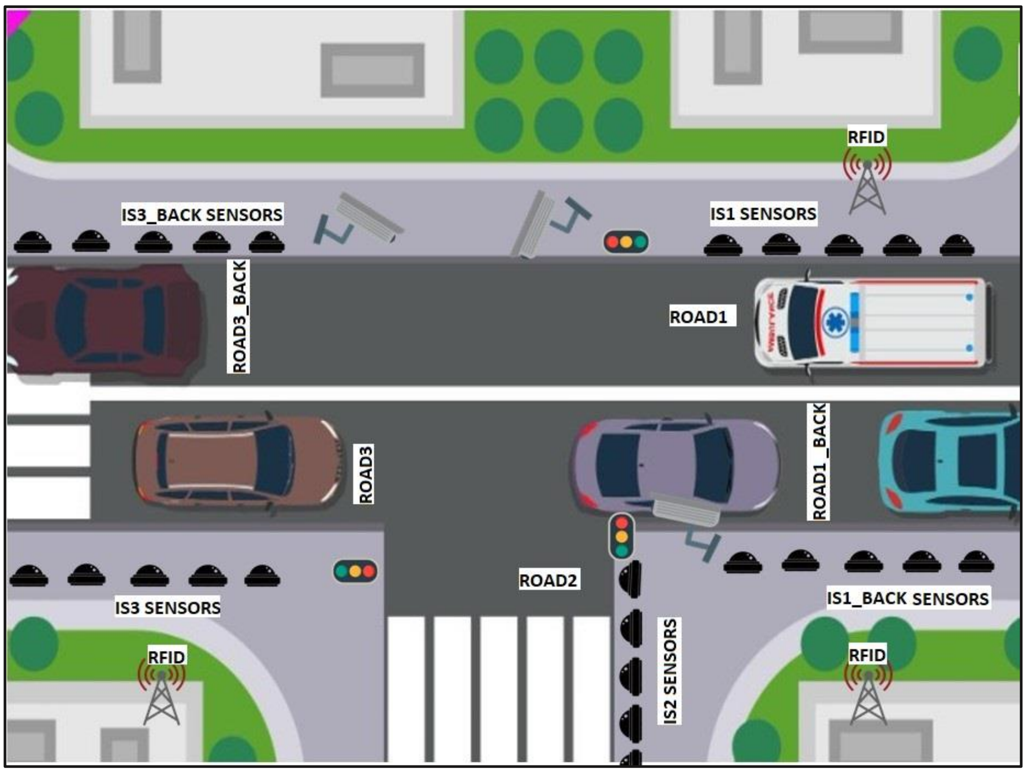


Fig. 2. System structure overview.

4.2 System Algorithm

each road has two states green and yellow and one common state where all roads are red. each state is illustrated in the TABLE. 1 Also, the system has three different states' transition flow based on the emergency signals and the traffic density of each road. The traffic system manages three-way intersection roadways. It always starts by grunting road1 the yellow light for 3 seconds, and while counting up to 3 seconds it checks for any emergency or road blockage signals. If not found it determines the green light period of time for road2 by continuously monitoring the active sensors count on road2, which indicate the traffic density level. After the 3 seconds are done, the green light is grunted to road2 and the system flow continues as shown in Fig. 3. If an emergency signal is received the normal flow is not followed and the system states flow changes based on the emergency signal value as shown in Fig. 4 the system is in the road1 green state "R1G" then it receives an emergency signal from road3 "EM=110" the system immediately transits to "R3G" state and stays in it until the emergency signal is gone "EM=000".If an emergency signal is not present but a road blockage signal is received the system moves immediately to the "ALLRED" state and stays in it until the blockage signal is gone as shown in Fig. 5 the flow chart of road2 green state is shown in Fig. 6 The remaining roads have the same idea as the road2 green state flow chart. Fig. 7 shows the flow chart of road1 yellow state. Fig. 8 shows the flow chart of the all-red state.

TABLE. 1. THE SYSTEM'S STATES

State	**Meaning**
R1G	Road1 is green and the other roads are red.
R1Y	Road1 is yellow and the other roads are red.
R2G	Road2 is green and the other roads are red.
R2Y	Road2 is yellow and the other roads are red.
R3G	Road3 three is green and the other roads are red.
R3Y	Road3 three is yellow and the other roads are red.
ALLRED	The traffic light is red for All roads.

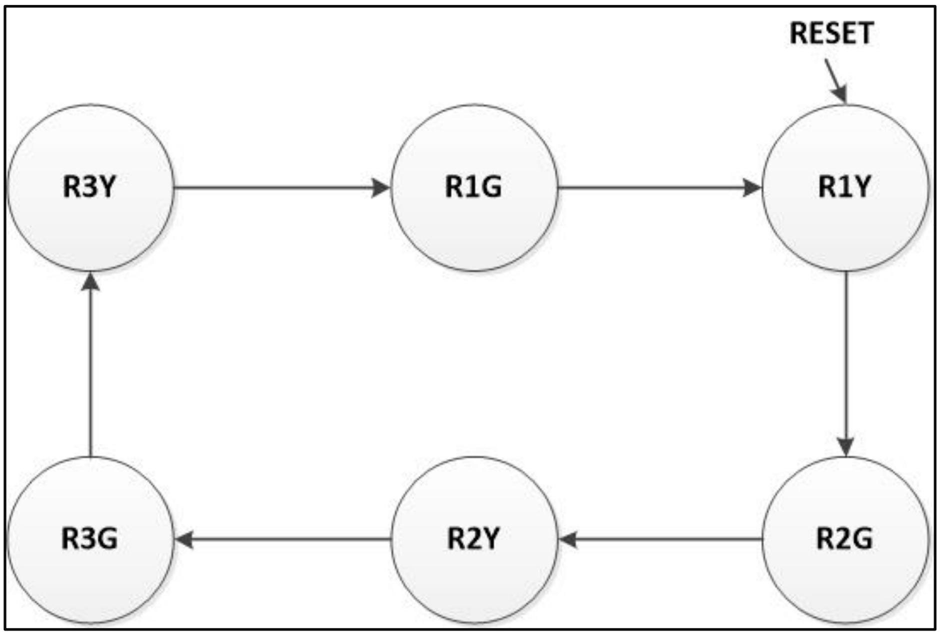


Fig. 3. The normal flow of the system's states.

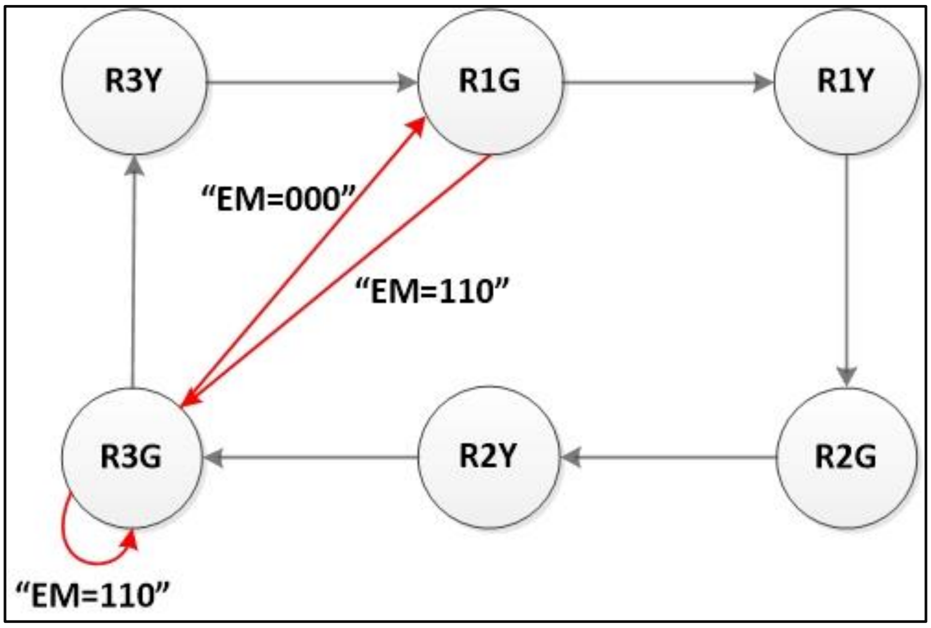


Fig. 4. The system's states flow with an emergency vehicle on road 3.

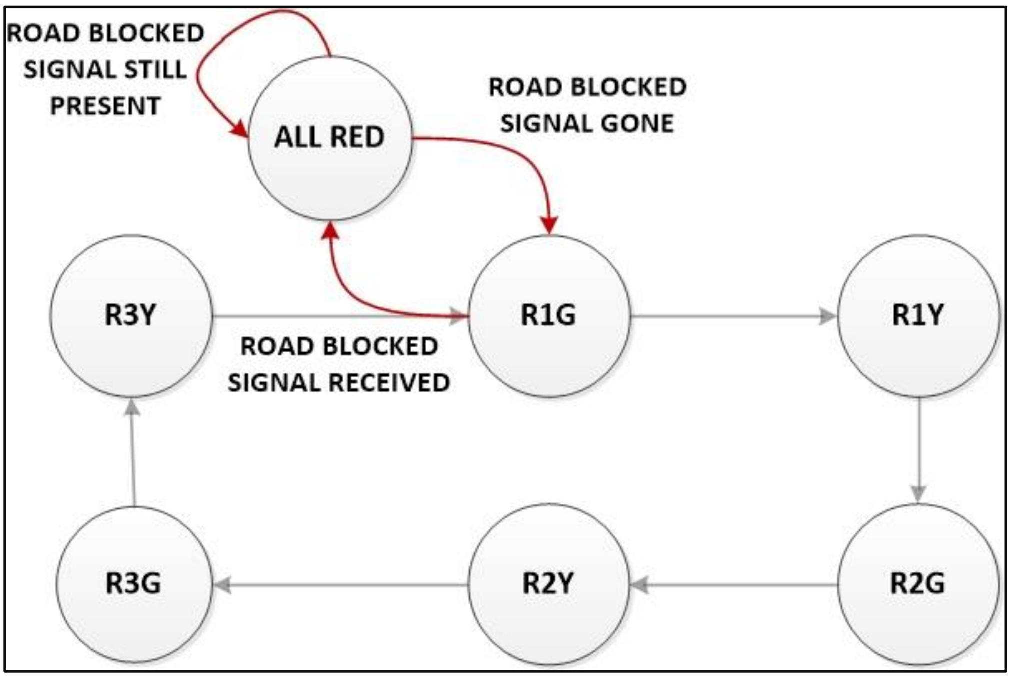

ROAD BLOCKED SIGNAL STILL PRESENT
ALL RED
ROAD BLOCKED SIGNAL GONE
R3Y
R1G
R1Y
ROAD BLOCKED SIGNAL RECEIVED
R3G
R2Y
R2G

Fig. 5. The state transition with the road-blocked signal.

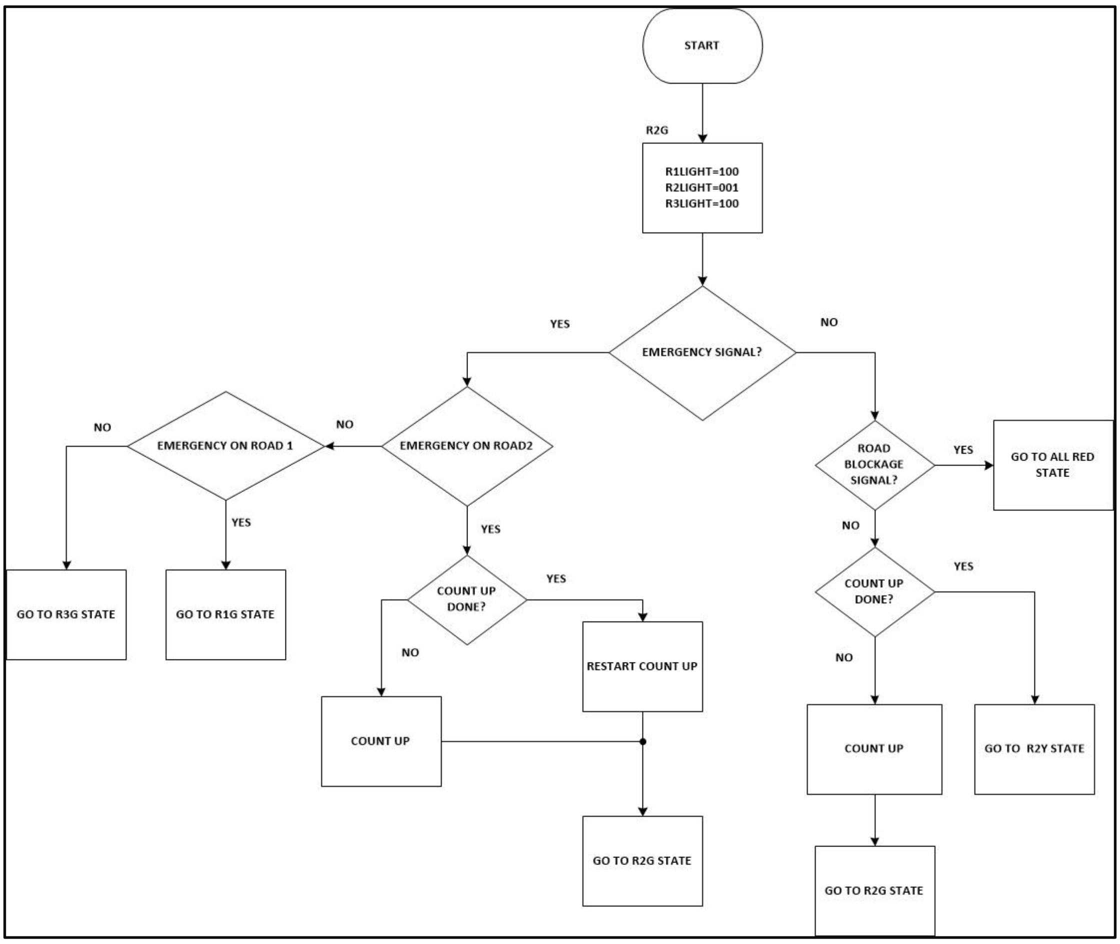


Fig. 6. Flow chart of road2 green state.

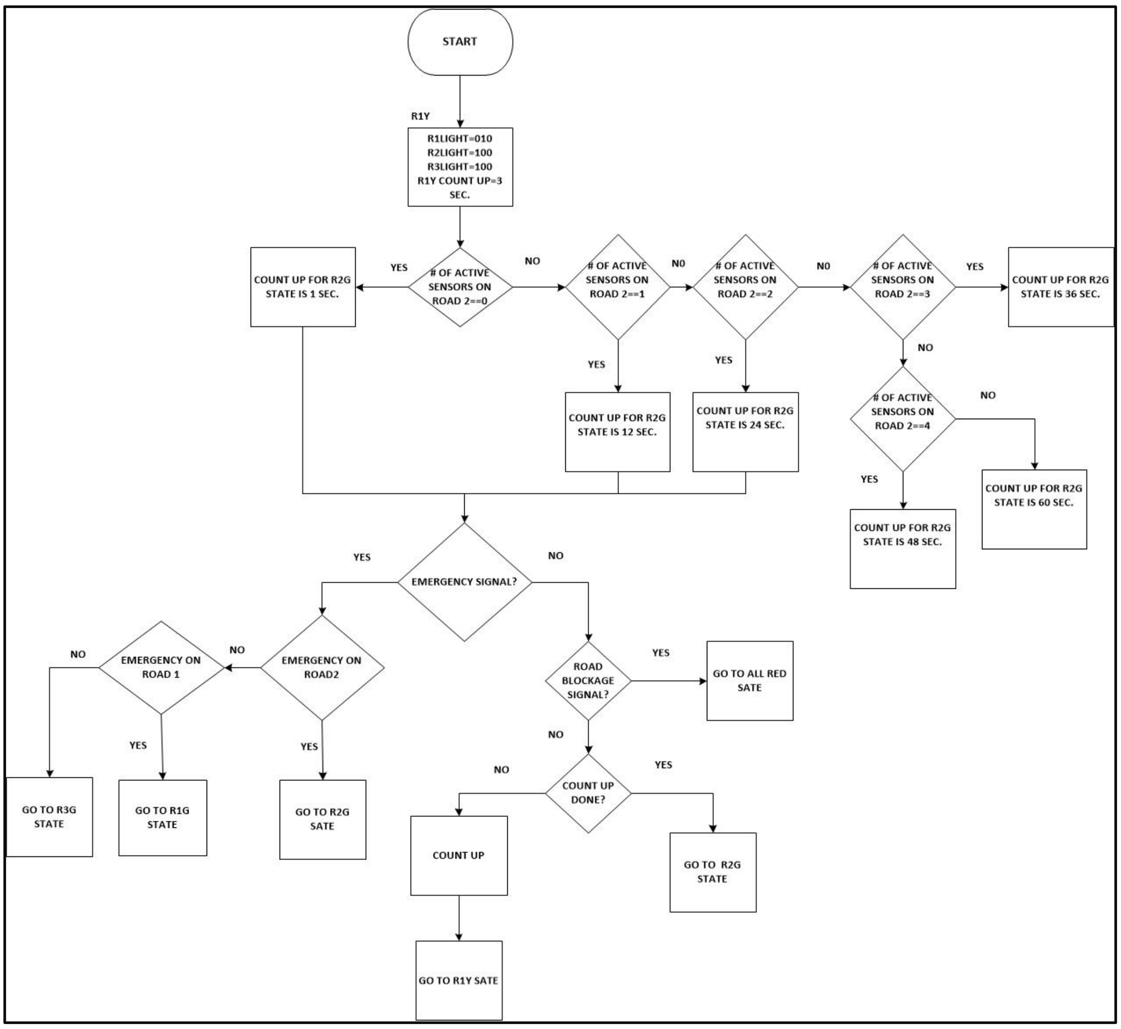

START
R1Y
R1LIGHT=010
R2LIGHT=100
R3LIGHT=100
R1Y COUNT UP=3 SEC.
# OF ACTIVE SENSORS ON ROAD 2==0
YES
COUNT UP FOR R2G STATE IS 1 SEC.
NO
# OF ACTIVE SENSORS ON ROAD 2==1
YES
COUNT UP FOR R2G STATE IS 12 SEC.
N0
# OF ACTIVE SENSORS ON ROAD 2==2
YES
COUNT UP FOR R2G STATE IS 24 SEC.
N0
# OF ACTIVE SENSORS ON ROAD 2==3
YES
COUNT UP FOR R2G STATE IS 36 SEC.
NO
# OF ACTIVE SENSORS ON ROAD 2==4
YES
COUNT UP FOR R2G STATE IS 48 SEC.
NO
COUNT UP FOR R2G STATE IS 60 SEC.
EMERGENCY SIGNAL?
YES
EMERGENCY ON ROAD2
YES
GO TO R2G SATE
NO
EMERGENCY ON ROAD 1
YES
GO TO R1G STATE
NO
GO TO R3G STATE
NO
ROAD BLOCKAGE SIGNAL?
YES
GO TO ALL RED SATE
NO
COUNT UP DONE?
YES
GO TO R2G STATE
NO
COUNT UP
GO TO R1Y SATE

Fig. 7. Flow chart of road1 yellow state.

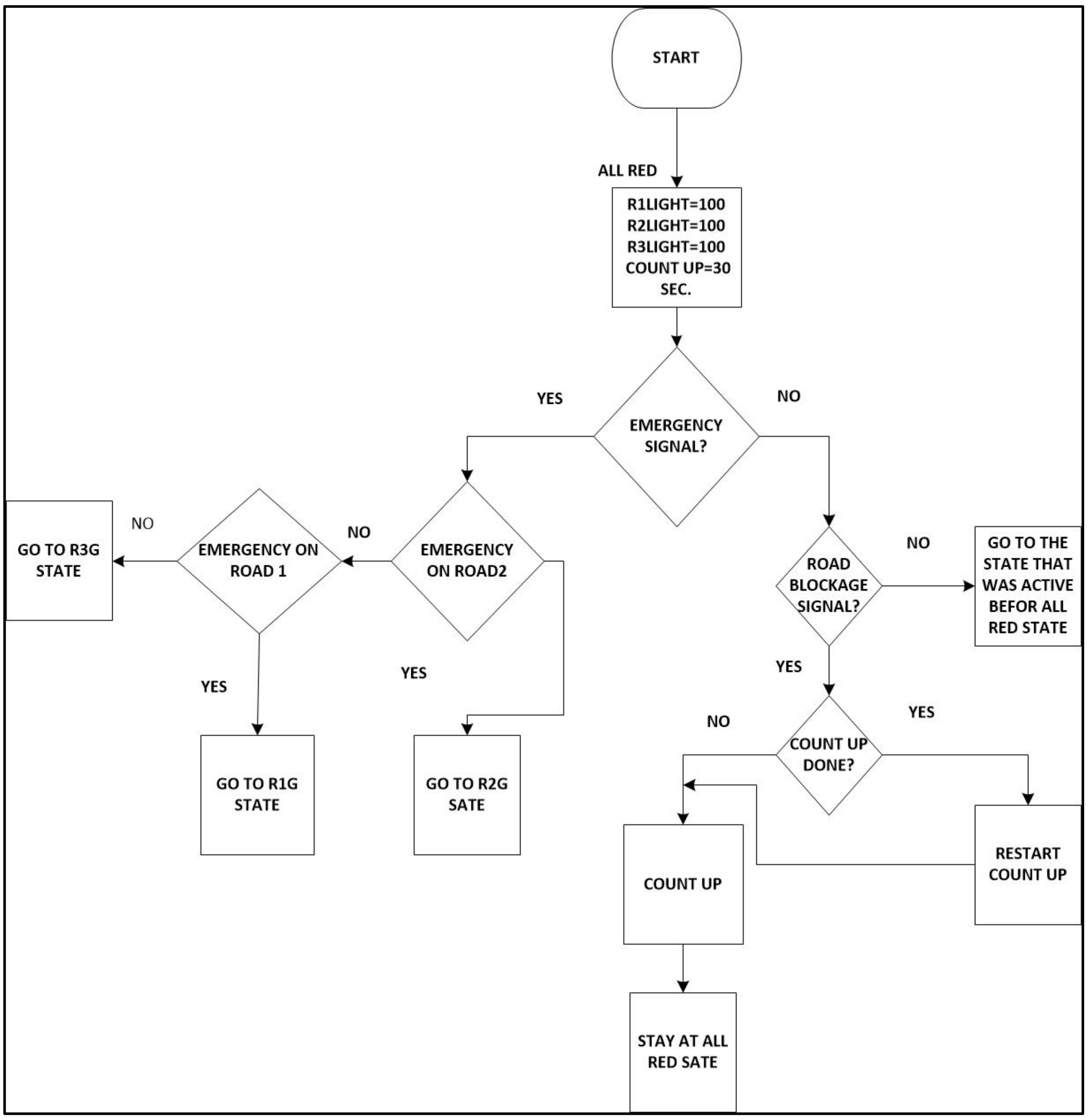


Fig. 8. Flow chart of ALLRED state.

4.3 Features offered by the system

1. Real-time density-based dynamic traffic signal time: This smart traffic system is integrated with a real-time density-based system. Assume at any minute, it is road1 turn to get the green light but the road is not crowded by vehicles at all at the same time road2 is very crowded with vehicles, it is not a good idea to give a fixed amount of time for different traffic densities. This Traffic system is capable of realizing this and gives a green light to road1 for time period based on the traffic density on that particular road, guaranteeing adequate management of traffic, relying on the density of the roads. Fig. 7 shows the flow chart of road1 yellow state illustrating different time periods based on traffic density levels.

2. Giving priority to emergency vehicles at any time: The traffic system permits emergency vehicles to pass. Whenever it is detected on any road. Fig. 3 shows a state transition of a case of an emergency vehicle on road3. while it is a green signal on road1. The traffic system turns road1's signal to red and road3 is granted the green light allowing the emergency vehicle to pass after that the traffic system returns to road1.

3. Preventing three-intersection getting blocked: The traffic system prevents the intersection from getting blocked when one or both of the main roads in opposite directions (roads that the vehicles on exit the intersection) are crowded with vehicles. If any road is given the green light the intersection will be blocked even if the other roads are not crowded. Fig. 5 shows the state transition when a road blockage signal is received. The system enters an all-red state where all roads are given the red light to make some time for the crowded roads to get less crowded. Fig. 9 shows the problem when road1 and road three opposite directions are crowded with vehicles and the green light is given to road2.

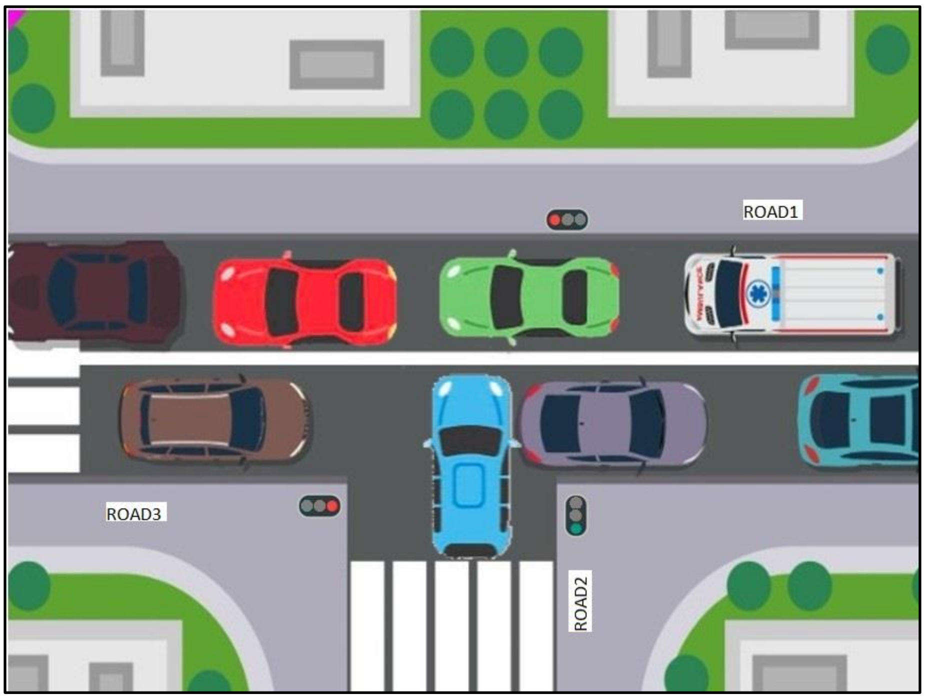


Fig. 9. Three-way intersection blockage view.

4.4 Results and discussions

To verify the working of the system, we have used ModelSim-Intel-ALTER to obtain the timing diagram. In the timing diagram R1light, R2light, and R3light represent the traffic lights for the three roads where 100, 010, and 001 represent green, yellow, and red, respectively. The inputs IS1, IS2, IS3, IS1_BACK, and IS3_BACK are the IR sensors for road1, road2, road3, road1 opposite direction, and road3 opposite direction, respectively their values range from 00000 to 11111 which represent the different traffic densities on each road. EM input which represents the passing emergency vehicle on a particular road, for simulation purposes EM input values are 0xx,100,101, and 110 which represent no emergency vehicles, emergency vehicle on road one, emergency vehicle on road two, and emergency vehicle on road three, respectively. State register represents the current state of the traffic system; the delayR1G, delayR2G, and delayR3G, represent the time assigned to road1 ,road2, and road3 for the green signal , delayRY represents the time for all roads for the yellow traffic signal. The different green light time periods values are 60, 48, 36, 24, 12, and 1 sec. these values depends on the active IR sensors. Fig. 10 presents different traffic densities and different traffic light times based on the IR sensors on each road. Fig. 11 presents a case when the road one opposite direction is full of vehicles if we grant green light to any road that will cause the problem referred to in Fig. 9 This issue is solved by giving all roads red lights for enough amount of time for the crowded road to get less crowded. Fig. 12 present represents passing an emergency vehicle on road three while road the yellow light on road one is on the TLC gives the priority to road three to enable the emergency vehicle to pass, after that the TLC returns to the previous state when road one yellow light was on and continues on the normal state transitions. Fig. 13 show the system implementation on FPGA.

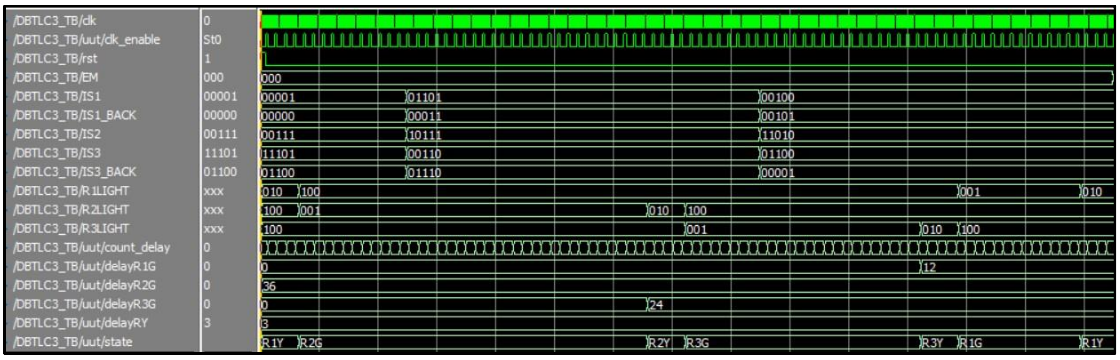


Fig. 10. The timing diagram for the real-time density-based traffic system.

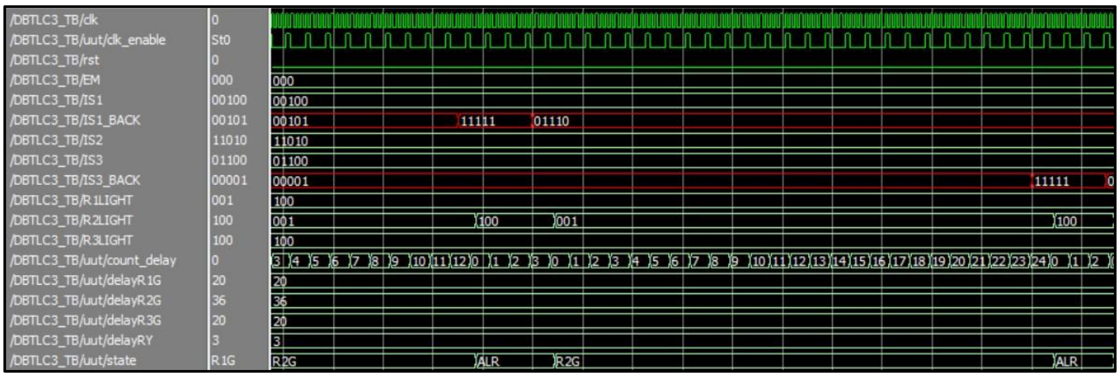


Fig. 11.The timing diagram for the real-time density-based traffic system with road blockage signal.

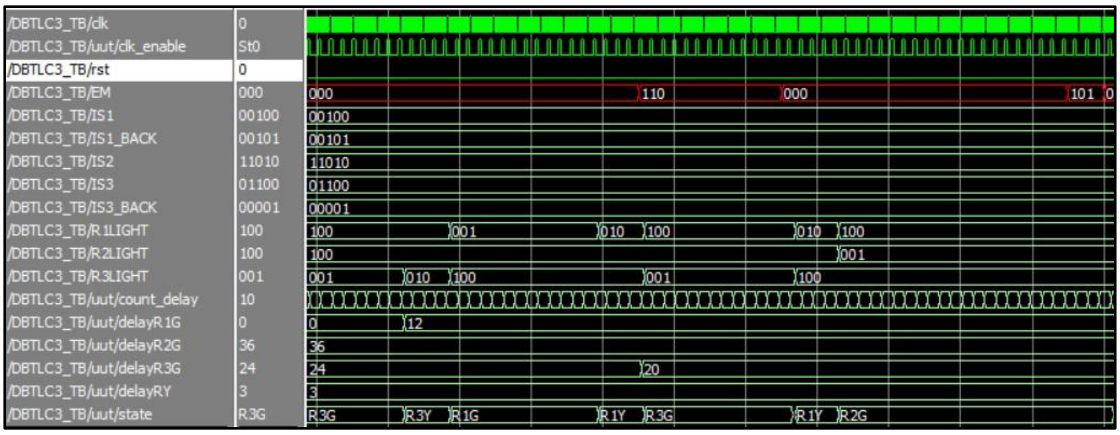

/DBTLC3_TB/clk 0
/DBTLC3_TB/uut/clk_enable St0
/DBTLC3_TB/rst 0
/DBTLC3_TB/EM 000 000 110 000 101 0
/DBTLC3_TB/IS1 00100 00100
/DBTLC3_TB/IS1_BACK 00101 00101
/DBTLC3_TB/IS2 11010 11010
/DBTLC3_TB/IS3 01100 01100
/DBTLC3_TB/IS3_BACK 00001 00001
/DBTLC3_TB/R1LIGHT 100 100 001 010 100 010 100
/DBTLC3_TB/R2LIGHT 100 100 001
/DBTLC3_TB/R3LIGHT 001 001 010 100 001 100
/DBTLC3_TB/uut/count_delay 10
/DBTLC3_TB/uut/delayR1G 0 0 12
/DBTLC3_TB/uut/delayR2G 36 36
/DBTLC3_TB/uut/delayR3G 24 24 20
/DBTLC3_TB/uut/delayRY 3 3
/DBTLC3_TB/uut/state R3G R3G R3Y R1G R1Y R3G R1Y R2G

Fig. 12. The timing diagram for the real-time density-based traffic system with an emergency vehicle on road3.

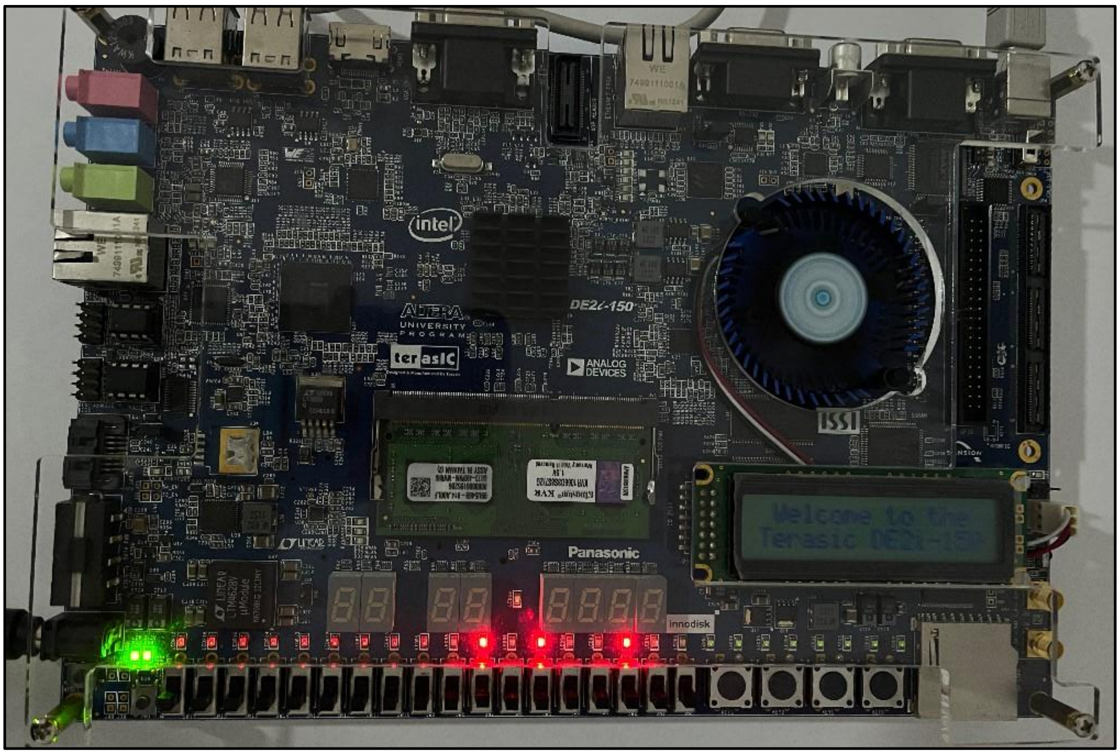


Fig. 13. The Traffic system on FPGA.

Chapter 5

License Plate Recognition

The proposed automatic license plate system consists of five parts:

1. Detection of red-light violation detection and image acquisition.
2. License plate localization.
3. Image pre-processing.
4. Extract the license number.
5. License plate number recognition.

5.1 Detection of red-light violation and image acquisition.

Passive infrared sensors are placed at a distance from each of the intersecting roads' entrance points at the intersection of a three-way road. These sensors are coordinated with the red-light traffic signal so that each road's sensors are enabled and disabled simultaneously. Any car that enters the junction while the light is red is picked up by the infrared sensor. The camera assigned to that road section is triggered upon detection of any violating vehicles and takes a photo of the offending car. The FPGA receives the vehicle image at that point. The camera was positioned so that it could take a picture of the front of a car that was breaking the law, including the license plate.

5.2 License plate localization

A machine-learning algorithm is used to determine the license plate region in the image. The aggregate channel features (ACF) machine learning algorithm is used to train the license plate detector. The detector was trained on a custom-made dataset of Libyan license plates. The detector detects license plates from different angles and different distances and also can detect skewed license plates. The license plate detector is trained on 3069 images and tested on 746 images of cars with license plates from different angles and distances. The data set was collected from streets with different illumination conditions and was labeled manually. The machine learning approach is used for localizing the license plate because it gives us the best results that image processing techniques can't give us in many conditions such as differences in illumination, distances, and angles of the captured image. The evaluation of the license plate detector on the test data produces average precision of 0.9.

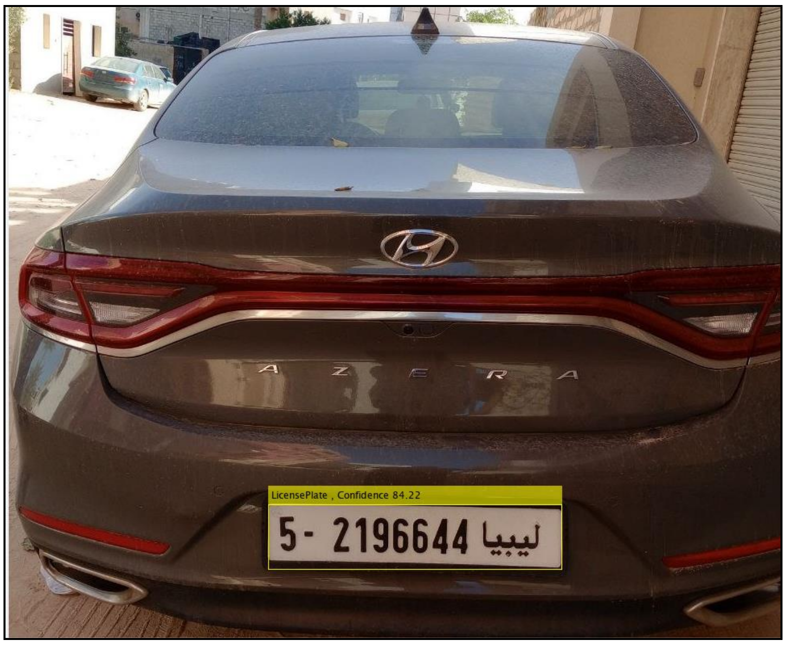


Fig. 14. Detected license plate.

5.3 Image pre-processing

After the license plate region is determined the license region is cropped and goes through image processing steps to prepare it for digit extraction and recognition.

1. Extract the license plate region.

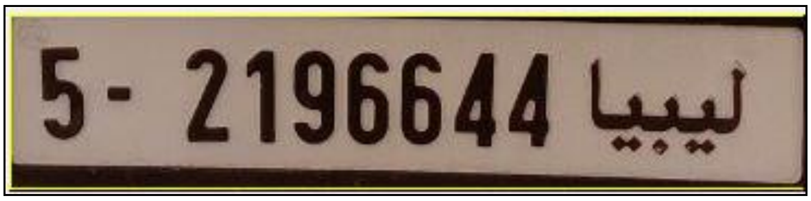


Fig. 15. Cropped license plate.

2. Convert the license plate to a binary image.

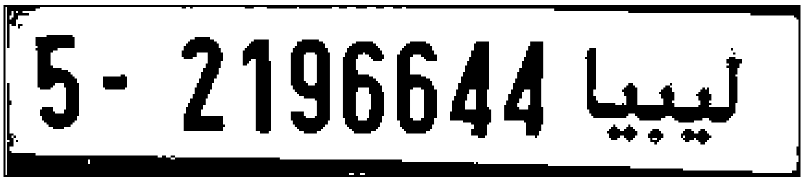


Fig. 16. Binary image of license plate.

3. Skew Detection and Correction.

Skewed license plates cause errors in recognizing license numbers, therefore, correcting skewed license plates is extremely important. Hough transformation is used in this process. The following steps describe the skew detection and correction process:

a) Convert the image to grayscale.

b) Apple Canny or Sobel filter.

c) Find Hough lines between 0.1 to 180-degree angle.

d) Round the angles from line peaks to 2 decimal places.

e) Find the angle with the highest occurrence.

f) Rotate the image with that angle.

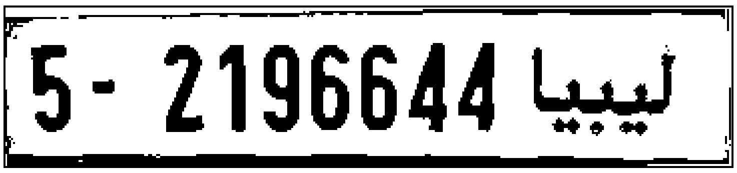


Fig. 17. Skew corrected license plate.

4. Remove the black border.
The black border could cause errors in extracting the license's number, therefore, the black border is removed as follows:

a) Perform two-dimensional median filtering on plate images.
b) Perform hole Filling.
c) NOT(XOR(result of median filtering, result of hole filling)).

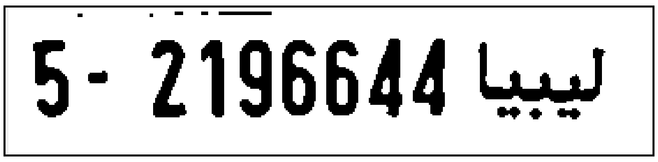


Fig. 18. License image after removing black border.

5. Invert the license image color.

Inverting license plate color is important because the numbers and characters need to be white to get extracted using the connected-component labeling.

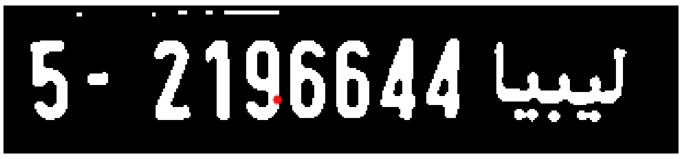


Fig. 19. Colour inverted license plate image.

6. Remove small objects.

The license plate is resized to a fixed size, which in our case [136 × 513], and then the objects with a size less than 400 pixels, which can’t be digits or Libya words, are removed.

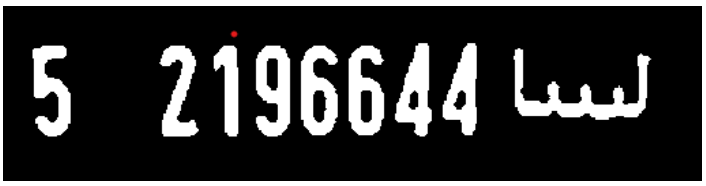


Fig. 20. License plate image after removing small objects.

5.4 Extraction of the license numbers

The license plate number is extracted via connected-component labeling. From top to bottom and left to right, the connected-component labeling begins extracting objects from an image. The connected-component extracting method is appropriate given that Libyan license plates are rectangular in shape and include all of their numbers in a single line.

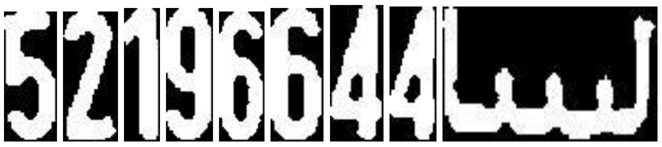
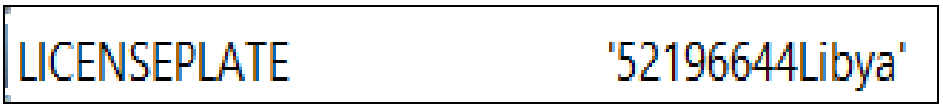


Fig. 21. Extracted license number

5.5 License plate recognition

After extracting the license plate number, the digits, and words are classified using Image category classification by generating a histogram of visual word occurrences that represent the images. this histogram, called a bag of visual words or bag of features, is used to train an image category classifier. The classifier is trained with a custom-made dataset of digits and the word "ليبيا" Libya. 550 images are used in total to train and evaluate the classifier. The dataset is divided into 323 images for training and 137 images for evaluating the classifier. The evaluation set produces average accuracy of 0.966.



LICENSEPLATE '52196644Libya'

Fig. 22. Result of license plate recognition.

5.6 Results and discussions

The Libyan license plate recognition system is implemented using MATLAB. The license plate localization step was implemented using an ACF license plate detector, the detector was trained on 3069 images and tested on 746 images of license plates. The evaluation of the license plate detector on the test data produces average precision of 0.9 as shown in Fig. 22. The license plate number Extraction step worked correctly on 70 test images due to the pre-processing step which helped a lot in improving the extracting step. In the license plate number recognition step, the classifier was trained on a custom dataset of digits and the word "ليبيا" Libya. 550 images were used in total to train and evaluate the classifier. The dataset was divided into 323 images for training and 137 images for evaluating the classifier. The evaluation produces average accuracy of 0.966.

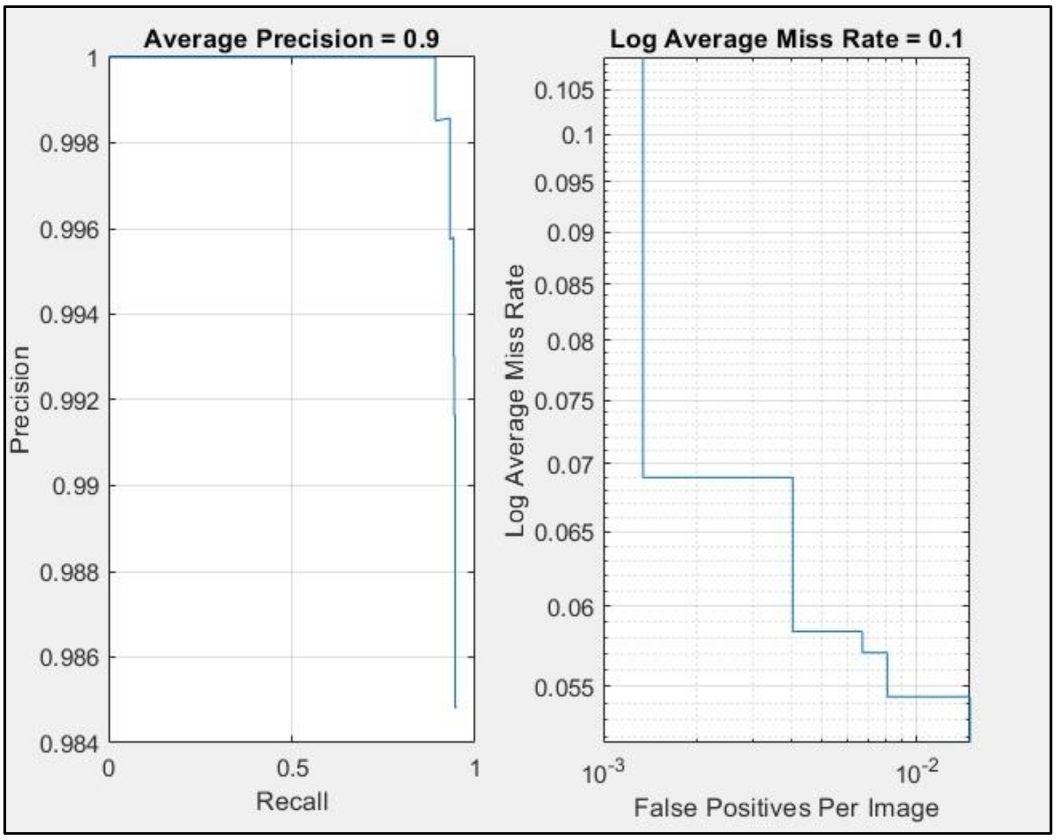


Fig. 23. Evaluation of license plate detector.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This project's main contribution is to design, simulate, and implement a real-time density-based traffic light system and Libyan license plate recognition system. The traffic system is specifically designed to reduce congestion and traffic on intersecting roadways. Based on the volume of traffic, the system can make decisions in real-time to select the best traffic signal time period for each road. The system's key benefits are its ability to prioritize emergency vehicles and make wise decisions to avoid blocking intersections. After the testing of the traffic light system module, the design was simulated and verified using software tools from Intel-Altera. The system was then developed and tested using the Cyclone IV GX: EP4CGX150DF31C8 FPGA evaluation platform to ensure that it worked as intended in hardware. The system for recognizing license plates in Libya is designed to track traffic violations using machine learning and image processing methods. The license plate software is programmed and tested using MATLAB software tool.

6.2 Future Work

Future work can be extended to replace infrared sensors with vehicle detection in a video sequence using image processing techniques. the system records the car movement and calculates the density of vehicles to decide the time delay of the traffic light for each road. And the license plate recognition algorithm can be written from scratch in Verilog HDL and implemented on an FPGA board.

Reference

[1] B. U. Umar, O. M. Olaniyi, A. James, and O. R. Isah, "Traffic violation detection system using image processing," *Computer Engineering and Applications*, Vol. 10, No. 2, June 2021.

[2] J. R. Aworemi, I. A. Abdul-Azeez, A. J. Oyedokun, and J.O.Adewoye, "A study of the causes, effects and ameliorative measures of road traffic congestion in Lagos Metropolis," *European Journal of Social Sciences*, vol. 11, no. 1, pp. 119-128, 2009.

[3] https://robocraze.com/blogs/post/ir-sensor-working.

[4] https://www.youngwonks.com/blog/What-is-an-infrared-sensor

[5] https://www.techtarget.com/iotagenda/definition/RFID-radio-frequency-identification

[6]https://www.fda.gov/radiation-emitting-products/electromagnetic-compatibility-emc/radio-frequency-identification-rfid

[7] https://www.tutorialspoint.com/vlsi_design/vlsi_design_verilog_introduction.htm

[8] https://en.wikipedia.org/wiki/MATLAB

[9] https://www.geeksforgeeks.org/applications-of-matlab/

[10]https://www.arrow.com/en/research-and-events/articles/fpga-basics-architecture-applications-and-uses

[11] DE2i-150_FPGA_System_manual

[12] https://home.engineering.iastate.edu/~zzhang/courses/cpre581-f05/resources/modelsim.pdf

[13]https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/quartus/gl_quartus_welcome.htm

[14] N. N. S. N. Dzulkefli, S. Rohafauzi, A. N. Jaafar, R. Abdullah, R. Shafie, M. S. Selamat,... & M. Z. Z. Muhammad,"Density Based Traffic System via IR Sensor," *In Journal of Physics: Conference Series*, Vol. 1529, No. 2, 2020.

[15] K. J. Ukagwu, S.Nalweyiso,O. M. Awodu, and K. E. Ukhurebor, "A Design, Simulation and Implementation of Intelligent Traffic Lights Based on Traffic Density" *Islamic University Multidisciplinary Journal*, Vol. 7 No. 1, 2020.

[16] A. M. Hassan, S. A.Ghoul, and A. A.Alkabir. "Libyan Vehicle License Plate Recognition with Support Vector Machine," *Al-Mukhtar Journal of Sciences*, vol. 37, pp. 1--13, 2022.

Appendix A: Traffic light system

A.1 Traffic light source code

```
module DBTLC3(R1LIGHT, R2LIGHT,R3LIGHT,
IS1,IS1_BACK,IS2,IS3,IS3_BACK,EM,clk,rst);
parameter R1G="R1G", // road1 green state
   R1Y = "R1Y",// road1 yellow state
   R2G="R2G", // road2 green state
   R2Y = "R2Y",//road2 yellow state
   R3G="R3G", //road3 green state
   R3Y="R3Y", // road3 yellow state
   ALLRED="ALR" ;//all roads red state
```

```
         reg[27:0]
count=0,count_delay=0,delayR1G=20,delayR2G=20,delayR3G=20,delayRY=3,d
elayAllRed=30;//time delays
   input wire [4:0] IS1,IS1_BACK,IS2,IS3,IS3_BACK; // sensorS
   input  wire clk,
   rst; // reset active high
        input wire [2:0]EM;//emergency signal
        output reg[2:0] R1LIGHT, R2LIGHT,R3LIGHT;//traffic lights
        wire clk_enable;
        reg
R2G_count_en=0,R2G_count_done=0,R1G_count_en=0,R1G_count_done=0,R3G_c
ount_en=0,R3G_count_done=0,RY_count_done=0,R1Y_count_en=0,R3Y_count_e
n=0,R2Y_count_en=0,ALREd_count_en,ALR_count_done,R1G_EM=0,R2G_EM=0,R3
G_EM=0;
   wire [2:0]
sensorcount1,sensorcount1_1,sensorcount2,sensorcount3,sensorcount3_3;
//sensors count
reg[63:0] state,next_state;
reg [63:0] next_state2;
countones u1(IS1,sensorcount1);//counting the active sensors model
countones u1_1(IS1_BACK,sensorcount1_1);
countones u2(IS2,sensorcount2);
countones u3(IS3,sensorcount3);
countones u3_3(IS3_BACK,sensorcount3_3);
always @(posedge clk)
begin
if(rst)
state<=R1Y;
else
state<=next_state;
end
////////////////////////
//////////////////////////////////////////////////////////////////////
///////

always @ (*)
begin
case(state)
R1G:begin // Green on ROAD1 and red on ROAD2 and ROAD3
 R1LIGHT=3'b001;//road1 green
 R2LIGHT=3'b100;//road2 red
 R3LIGHT=3'b100;//road3 red
 // enable counting up
 R1G_count_en<=1;
 R1Y_count_en<=0;
 R2G_count_en<=0;
 R3G_count_en<=0;
 R2Y_count_en<=0;
 R3Y_count_en<=0;
 ALREd_count_en<=0;
 R2G_EM<=0;
 R3G_EM<=0;
 //next state logic
 if(EM==3'b100)begin//if there emergency on the same road
   R1G_EM<=1;//enable emergency on road1
 end
 if(R1G_count_done)begin//if count up is done
 if(~EM[2])begin//no emergency
 if(~(sensorcount1_1==5 || sensorcount3_3==5))begin//no road blockage
  if(R1G_EM)//if emergency is enabled
 next_state=next_state2;//save the current state
 else
   next_state=R1Y;//go to the next state in sequence
 end
  else if((sensorcount1_1==5 || sensorcount3_3==5))begin//road
blockage signal
    next_state<=ALLRED;//go to all road red state
  next_state2<=R1G;//save the current state
 end
 end
 else if(EM[2])begin
 case({EM[1],EM[0]})
2'b00:begin//if emergency on road 1
if(R1G_count_done)begin//count up is done
delayR1G=20;//give 20 sec more
next_state2<=R1Y;//save the current state
next_state<=R1G;// go to the emergency state
end
else begin
next_state2<=R1Y;
next_state<=R1G;
end
end
2'b01:begin
delayR2G=20;
next_state2<=R1G;
R1G_EM<=0;
next_state<=R2G;
end
2'b10:begin
delayR3G=20;
next_state2<=R1G;
R1G_EM<=0;
next_state<=R3G;
end
default:
next_state<=R1Y;
endcase
end

 end
 end

 /////////
 R1Y: begin // Yellow on ROAD1 state
 ////////////////////////////
 //determine road 2 green light time period
 if(sensorcount2==0)
delayR2G=1;
else if(sensorcount2==1)
delayR2G=12;
else if(sensorcount2==2)
delayR2G=24;
else if(sensorcount2==3)
```

```
delayR2G=36;
else if(sensorcount2==4)
delayR2G=48;
else if(sensorcount2==5)
delayR2G=60;
//////////////////////////////
 R1LIGHT=3'b010;//road1 yellow
 R2LIGHT=3'b100;//road2 red
 R3LIGHT=3'b100;//road3 red
 //enable count up of  road1 yellow state
 R1G_EM<=0;
 R2G_EM<=0;
 R3G_EM<=0;
R1Y_count_en<=1;
R1G_count_en<=0;
 R2G_count_en<=0;
 R3G_count_en<=0;
 R2Y_count_en<=0;
 R3Y_count_en<=0;
  ALREd_count_en<=0;
  ///next state logic
 if(RY_count_done  )begin
 if(EM[2]==0)begin
 if(~(sensorcount1_1==5 || sensorcount3_3==5))begin
 next_state<=R2G;
 end
 end
  else if(EM[2]==1)begin
 case({EM[1],EM[0]})
2'b00:begin
delayR1G=20;
next_state<=R1G;
end
2'b01:begin
delayR2G=20;
next_state<=R2G;
end
2'b10:begin
delayR3G=20;
next_state<=R3G;
end
default:
next_state<=R2G;
endcase
 end
  else if((EM[2]==0) && (sensorcount1_1==5 ||
sensorcount3_3==5))begin
   next_state2<=R1Y;
 next_state<=ALLRED;
 end
 end

end
//////////////
 R2G: begin // Green on ROAD2 and red on ROAD and ROAD3
 R1LIGHT=3'b100;//road1 red
 R2LIGHT=3'b001;//road2 green
 R3LIGHT=3'b100;//road3 red
 //enable count up of road2 green state
 R2G_count_en<=1;
 R1Y_count_en<=0;
R1G_count_en<=0;
 R3G_count_en<=0;
 R3Y_count_en<=0;
  R2Y_count_en<=0;
   ALREd_count_en<=0;
    R1G_EM<=0;
 R3G_EM<=0;
 //next state logic
    if(EM==3'b101)begin
   R2G_EM<=1;
 end
 if(R2G_count_done )begin//count up is done
 if(EM[2]==0 )begin//no emergency
  if( ~(sensorcount1_1==5 || sensorcount3_3==5))begin//no blockage
signal
   if(R2G_EM)begin//if count up was for emergency
 next_state<=next_state2;//return to last state before the emergency
 end
 else
  next_state<=R2Y;//go to next state of road2 yellow
 end
  else if((EM[2]==0 ) && (sensorcount1_1==5 || sensorcount3_3==5)
)begin//road blockage signal is there
 next_state2<=R2G;//save the current state
 next_state<=ALLRED;//go to all roads are red state
 end
 end
 else if(EM[2]==1 )begin//if count up is done and there emergency
signal
 case({EM[1],EM[0]})
2'b00:begin//if emergency on road1
delayR1G=20;//determine time period for the emergency
next_state2<=R2G;//save the current state
 R2G_EM<=0;
next_state<=R1G;//go to the emergency road state
end
2'b01:begin// if emergency on road2
  if(R2G_count_done )begin
delayR2G=20;
next_state<=R2G;
end
else
 next_state<=R2G;
end
2'b10:begin //if emergency on road3
delayR3G=20;
next_state2<=R2G;
 R2G_EM<=0;
next_state<=R3G;
end
default:
next_state<=R2Y;
endcase
```

```
end

 end

end
/////////////////////////
R2Y: begin // YELLOW on ROAD2 and RED on ROAD3 and ROAD1
if(sensorcount3==0)
delayR3G=1;
else if(sensorcount3==1)
delayR3G=12;
else if(sensorcount3==2)
delayR3G=24;
else if(sensorcount3==3)
delayR3G=36;
else if(sensorcount3==4)
delayR3G=48;
else if(sensorcount3==5)
delayR3G=60;
 R1LIGHT=3'b100;//road1 red
 R2LIGHT=3'b010;//road2 YELLOW
 R3LIGHT=3'b100;//road3 RED
 R1G_EM<=0;
 R2G_EM<=0;
 R3G_EM<=0;
 R2G_count_en<=0;
 R1Y_count_en<=0;
R1G_count_en<=0;
 R3G_count_en<=0;
 R3Y_count_en<=0;
  R2Y_count_en<=1;
   ALREd_count_en<=0;
 if(RY_count_done )begin
 if(EM[2]==0 )begin
 if(~(sensorcount1_1==5 || sensorcount3_3==5))begin
 next_state<=R3G;
 end
 end
 end
 else if(EM[2])begin
 case({EM[1],EM[0]})
2'b00:begin
delayR1G=20;
next_state<=R1G;
end
2'b01:begin
delayR2G=20;
next_state<=R2G;
end
2'b10:begin
delayR3G=20;
next_state<=R3G;
end
default:
next_state<=R3G;
endcase
end
```

```
 else if((EM[2]==0) && (sensorcount1_1==5 || sensorcount3_3==5))begin
  next_state2<=R2Y;
 next_state<=ALLRED;
 end
end
/////////////////////////
R3G: begin // Green on ROAD3 and red on ROAD2,ROAD1
 R1LIGHT=3'b100;//road1 red
 R2LIGHT=3'b100;//road2 red
 R3LIGHT=3'b001;//road3 green
 R2G_EM<=0;
 R1G_EM<=0;
 R3G_count_en<=1;
R2G_count_en<=0;
 R1Y_count_en<=0;
R1G_count_en<=0;
 R3Y_count_en<=0;
  R2Y_count_en<=0;
   ALREd_count_en<=0;
   if(EM==3'b110)begin
   R3G_EM<=1;
 end
 if(R3G_count_done)begin
 if( EM[2]==0 )begin
 if( ~(sensorcount1_1==5|| sensorcount3_3==5))begin
   if(R3G_EM)
 next_state<=next_state2;
 else
  next_state<=R3Y;
 end
  else if((EM[2]==0 ) && (sensorcount1_1==5 ||
sensorcount3_3==5))begin
  next_state2<=R3G;
 next_state<=ALLRED;
 end
 end
  else if(EM[2]==1)begin
case({EM[1],EM[0]})
2'b00:begin
delayR1G=20;
next_state2<=R3G;
R3G_EM<=0;
next_state<=R1G;
end
2'b01:begin
delayR2G=20;
next_state2<=R3G;
R3G_EM<=0;
next_state<=R2G;
end
2'b10:begin
if(R3G_count_done)begin
delayR3G=20;
next_state<=R3G;
end
else
 next_state<=R3G;
```

```
end
default:
next_state<=R3Y;
endcase
end

 end

end
///////////////////////////////////////
R3Y: begin // WELLOW on ROAD3 and READ on ROAD4,ROAD2,ROAD2
if(sensorcount1==0)
delayR1G=1;
else if(sensorcount1==1)
delayR1G=12;
else if(sensorcount1==2)
delayR1G=24;
else if(sensorcount1==3)
delayR1G=36;
else if(sensorcount1==4)
delayR1G=48;
else if(sensorcount1==5)
delayR1G=60;
 R1LIGHT=3'b100;//road1 red
 R2LIGHT=3'b100;//road2 red
 R3LIGHT=3'b010;//road3 YELLOW
  R1G_EM<=0;
 R2G_EM<=0;
 R3G_EM<=0;
  R2G_count_en<=0;
 R1Y_count_en<=0;
R1G_count_en<=0;
 R3G_count_en<=0;
 R3Y_count_en<=1;
  R2Y_count_en<=0;
   ALREd_count_en<=0;
 if(RY_count_done)begin
 if(EM[2]==0)begin
 if(~(sensorcount1_1==5 || sensorcount3_3==5))begin
 next_state<=R1G;
  end
  end
   else if((EM[2]==1 )) begin
 case({EM[1],EM[0]})
2'b00:begin
delayR1G=20;
next_state<=R1G;
end
2'b01:begin
delayR2G=20;
next_state<=R2G;
end
2'b10:begin
delayR3G=20;
next_state<=R3G;
end
default:
```

```
next_state<=R1G;
endcase
end
 else if((EM[2]==0 ) && (sensorcount1_1==5 ||
sensorcount3_3==5))begin
  next_state2<=R3Y;
 next_state<=ALLRED;
 end
  end

end
/////////////////////////
ALLRED: begin // ALL ROADS ARE RED
 R1LIGHT=3'b100;//road1 RED
 R2LIGHT=3'b100;//road2 red
 R3LIGHT=3'b100;//road3 red
  R1G_EM<=0;
  R2G_EM<=0;
  R3G_EM<=0;
  R2G_count_en<=0;
 R1Y_count_en<=0;
R1G_count_en<=0;
 R3G_count_en<=0;
 R3Y_count_en<=0;
  R2Y_count_en<=0;
ALREd_count_en<=1;
 if(ALR_count_done)begin
 if(EM[2]==0)begin
 if( ~(sensorcount1_1==5 || sensorcount3_3==5))begin
 next_state<=next_state2;
 end
 end
 else if(EM[2]==1)begin
 case({EM[1],EM[0]})
2'b00:begin
delayR1G=20;
next_state<=R1G;
end
2'b01:begin
delayR2G=20;
next_state<=R2G;
end
2'b10:begin
delayR3G=20;
next_state<=R3G;
end
default:
next_state<=next_state2;
endcase
 end
  else if( (EM[2]==0 ) && (sensorcount1_1==5 || sensorcount3_3==5))
 next_state<=ALLRED;
 end
 end
 default:
 next_state<=R1Y;
endcase
```

```
end
//counting up logic
always @(posedge clk )
begin
if(clk_enable==1) begin

if(R1G_count_en||R2G_count_en||R3G_count_en||R1Y_count_en||R2Y_count_
en||R3Y_count_en|| ALREd_count_en)
  count_delay <=count_delay + 1;//count up
  //if count up is reached or there emergency or road blockage signal
rise the count don flag and reset the counting
  if(((count_delay == delayR1G)&&R1G_count_en) ||
R1G_count_en&&(EM[2] && ~(EM==3'b100))||R1G_count_en && ~EM[2]
&&((sensorcount1_1==5 || sensorcount3_3==5))||R1G_count_en&&
~(EM==3'b100) && R1G_EM)
  begin

  R1G_count_done<=1;
   R2G_count_done<=0;
   R3G_count_done<=0;
        RY_count_done<=0;
        ALR_count_done<=0;
   count_delay<=0;
  end
  else if((count_delay == delayR2G)&&R2G_count_en || R2G_count_en &&
(EM[2] && ~(EM==3'b101)) ||R2G_count_en  &&(sensorcount1_1==5 ||
sensorcount3_3==5)&& (~EM[2])||R2G_count_en&& ~(EM==3'b101) &&
R2G_EM)
  begin
   R2G_count_done<=1;
   R1G_count_done<=0;
   R3G_count_done<=0;
        RY_count_done<=0;
        ALR_count_done<=0;
   count_delay<=0;
  end
  else if((count_delay == delayR3G)&&R3G_count_en || R3G_count_en &&
(EM[2] && ~(EM==3'b110))||R3G_count_en  &&(sensorcount1_1==5 ||
sensorcount3_3==5)||R3G_count_en&& ~(EM==3'b110) && R3G_EM)
  begin
  R1G_count_done<=0;
   R2G_count_done<=0;
   R3G_count_done=1;
        RY_count_done<=0;
        ALR_count_done<=0;
   count_delay<=0;
  end
  else if((count_delay ==
delayRY)&&(R3Y_count_en||R2Y_count_en||R1Y_count_en)||(R3Y_count_en||
R2Y_count_en||R1Y_count_en) &&((sensorcount1_1==2'b11 ||
sensorcount3_3==5)&&~EM[2]|| EM[2]))
  begin
  R1G_count_done<=0;
   R2G_count_done<=0;
   R3G_count_done<=0;
        RY_count_done<=1;
        ALR_count_done<=0;
   count_delay<=0;
  end
  else if((count_delay == delayAllRed)&& ALREd_count_en ||
ALREd_count_en && EM[2] || ALREd_count_en && (~(sensorcount1_1==5 ||
sensorcount3_3==5))&&~EM[2] )
  begin
  R1G_count_done<=0;
   R2G_count_done<=0;
   R3G_count_done<=0;
        RY_count_done<=0;
        ALR_count_done<=1;
   count_delay<=0;
  end
  else
  begin
   R1G_count_done<=0;
   R2G_count_done<=0;
   R3G_count_done<=0;
        RY_count_done<=0;
        ALR_count_done<=0;
end
 end
end
////////
always @(posedge clk)
begin
 count <=count + 1;
 //if(count == 50000000) // 50,000,000 for 50 MHz clock running on
real FPGA
 if(count == 3)
  count <= 0;
end
 assign clk_enable = count==3 ? 1: 0; // 50,000,000 for 50MHz running
on FPGA

endmodule
```

A.2 Active sensors count module code

```
module countones(A,ones);
    input [4:0] A;
    output reg [4:0] ones;

integer i;

always@(A)
begin
    ones = 0;  //initialize count variable.
    for(i=0;i<5;i=i+1)   //for all the bits.
        ones = ones + A[i]; //Add the bit to the count.
end

endmodule
```

A.3 Test bench code

```
`timescale 1s/ 1s
module DBTLC3_TB;
```

```
  reg [4:0] IS1,IS1_BACK,IS2,IS3,IS3_BACK; // sensorS
   reg clk, // clock
   rst; // reset active high
   reg [2:0] EM;
   wire[2:0] R1LIGHT, R2LIGHT,R3LIGHT; // output of lights
    DBTLC3 uut(R1LIGHT,
R2LIGHT,R3LIGHT,IS1,IS1_BACK,IS2,IS3,IS3_BACK,EM, clk, rst);
always # 0.5 clk=~clk;
initial
begin
  clk=0;
  rst=1;
  EM=3'B000;
  IS1=5'b00001;
  IS1_BACK=5'b00000;
  IS2=5'b00111;
  IS3=5'b11101;
  IS3_BACK=5'b01100;


  #5 rst=1'b0;
  #120 IS1=5'b01101;
  IS1_BACK=5'b00011;
  IS2=5'b10111;
  IS3=5'b00110;
  IS3_BACK=5'b01110;


   #100 EM=3'B000;
 #200 IS1=5'b00100;
  IS1_BACK=5'b00101;
  IS2=5'b11010;
  IS3=5'b01100;
  IS3_BACK=5'b00001;
  #300 EM=3'b110;
  #100 EM=3'B000;

 #200 EM=3'b101;
 #25 EM=3'b000;
 #50 EM=3'b100;
#50 EM=3'b000;

 #100 IS1_BACK=5'b11111;
 #30 IS1_BACK=5'b01110;
 #200 IS3_BACK=5'b11111;
#30 IS3_BACK=5'b01111;
#50 IS1_BACK=5'b01100;
 #30 EM=3'b101;
 #25 EM=3'b000;
 #50 EM=3'b100;
#50 EM=3'b000;

 #30 EM=3'b110;
 #25 EM=3'b000;
 #50 EM=3'b110;
#50 EM=3'b000;
#3000 $finish;
```

```
end
endmodule
```

Appendix B: License Plate Recognition

B.1 License plate recognition MATLAB code

```
close all
clear
clc
%% load the license plate detector
load('DetectorL3.mat');
%% License Plate detection and processing
% load test image
I=imread("cars (1).jpg");
% detect the license plate
[bboxes1, scores1] = detect(detectorS,I,'Threshold',1);
[~,idx] = max(scores1);
  annotation = sprintf('%s , Confidence %4.2f',detectorS.ModelName,scores1(idx));
   I = insertObjectAnnotation(I,'rectangle',bboxes1(idx,:),annotation);
   %show the detected license plate image
   imshow(I);
   %%
    BB= bboxes1(idx,:)
   %crop the boundingbox
   subImage = imcrop(I,BB);
   imshow(subImage);
  % convert the license plate image to gray level
   I=rgb2gray(subImage);
   imshow(I)
 % convert the License plate to BW image
 subImagegray=I;

subImageBW=imbinarize((subImagegray),'adaptive','ForegroundPolarity','dark','Sensi
tivity',0.4);
```

```
 imshow((subImageBW))
  %%
%skew detection and correction
g=HoughTransformation(subImageBW);
imshow(g)
%remove black boarder of plate
result=remove_boarder(g);
%%
imshow(result)

 %% rezise the license plate image
result=imresize(result,[136 513]);
imshow(result)
%%
%% remove small objects <400 pixels
[result2,num]= bwlabel(result,4);
result2 = bwareaopen(result2,400)
imshow(result2)
%% remove objects not at the same line of the liense plate
 [r c]=size(result2);
se1=strel('line',c,0);
e1=imclose(result2,se1);
figure;
imshow(e1);
result3 = bwareafilt(e1,1);
figure;
imshow(result3);
mask=result3;
result4=and(mask,result2)
figure;
imshow(result4);

%% connected component labelling to get extract license number
[result6,num]= bwlabel(result4,4);
```

```
s1=regionprops(result6,"BoundingBox");
%% empty list
LICENSEPLATE=[];
%% to get numbers of license Plate and recognize it
for i=1:num
s2=s1(i).BoundingBox;
object=imcrop(result6,s2);
 imwrite(object,['image' num2str(i) '.jpg'],'jpg');
 img = imread(['image' num2str(i) '.jpg'],'jpg');
 [labelIdx, score] = predict(categoryClassifier,img);
 y=categoryClassifier.Labels(labelIdx)
 y1=cell2mat(y)
 LICENSEPLATE=[LICENSEPLATE,y1];
imshow(object)
end
```

B.2 Skew detection and correction function

```
function Image=HoughTransformation(Image1)
k=Image1;
sam=k;
m=k;
[r c]=size(k);
BW = edge(m,'canny');
[H,T,R] = hough(BW);
P  = houghpeaks(H,1,'threshold',ceil(0.9*max(H(:))));
lines = houghlines(BW,T,R,P,'FillGap',0.8*c,'MinLength',40);
A=zeros(1, size(lines,2));
for i=1:size(lines,2)
    A(i)=lines(i).theta;
end
for i=1:size(lines,2)
if(A(i)<0)
g=imrotate_white(m,(90-abs(A(i))));
else
```

```
    g=(imrotate_white(m,A(i)-90));
end
end
Image=g;
end
```

B.3 Remove black border function

```
function no_border_img=remove_boarder(border_img)
temp1= medfilt2(border_img);
temp2 = imfill(temp1 ,"holes");
temp3=not(xor(temp1,temp2));
imshow(temp3)
temp4=not(temp3);
no_border_img=temp4;
end
```

B.4 License plate detector training

```
%% ground truth
load('traingtF4.mat')
%%
%% store labels in table
vehicleGTruth1= selectLabels(gTruth,'LicensePlate');
%%

%%
%%
if isfolder(fullfile('TrainingData'))
    cd TrainingData
else
    mkdir TrainingData
end
addpath('TrainingData');
%%
trainingData= objectDetectorTrainingData(vehicleGTruth1,'SamplingFactor',1,...
'WriteLocation','TrainingData');
%%
%rain the ACF detector. You can turn off the training progress output by specifying
'Verbose',false as a Name,Value pair.
detectorS = trainACFObjectDetector(trainingData,'NumStages',5);
%%
%Save the detector to a MAT file
save('DetectorL3.mat','detectorS');
rmpath('TrainingData');
```

B.5 Number classifier training

```
%% train number classifier
setDir  = fullfile("numbers");
imds = imageDatastore(setDir,'IncludeSubfolders',true,'LabelSource',...
    'foldernames');
[trainingSet,testSet] = splitEachLabel(imds,0.8,'randomize');
bag = bagOfFeatures(trainingSet);
categoryClassifier = trainImageCategoryClassifier(trainingSet,bag);
confMatrix = evaluate(categoryClassifier,testSet)
mean(diag(confMatrix))
```