PROJECT REPORT

ON

AUTOMATED PARKING TICKET GENERATOR USING FPGA

Submitted by

Deepak Shivani (012560340)

Navyashree Chandraiah (012555192)

Shrusti Shashidhar (012418523)

Under the guidance of

Prof. John (JeongHee) Kim

ABSTRACT

Due to immense rise of vehicles there are less parking spots available in urban areas, this leads to the necessary implementation of the parking laws by the authorities to keep a proper conduct. There are certain parking spots where if the vehicle parked violates the parking law may lead to a parking ticket and using labor force to supervise the illegal parking of vehicles can be tedious task. We are proposing a device which can automatically generate parking tickets for illegally parked vehicles. The design is based on a FPGA board integrated with couple of detectors such as a RFID or an IR sensor to detect the presence of the car which are further connected to Wi-Fi or GSM based modules to send the information to the Office of Parking Violation. If any of the sensors triggers a signal or detects the presence of car in the illegal parking spot then a notification will be sent using the internet application.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Each and every state has its own parking violations but some of the laws can be generalized and can be kept mind to avoid illegal parking, let us consider some of those points:

Never park or leave your vehicle:

- Where a “No Parking” sign is posted.
- On a marked or unmarked crosswalk, sidewalk, partially blocking a sidewalk, or in front of a driveway.
- Within 3 feet of a sidewalk ramp for disabled persons or in front of or on a curb that provides wheelchair access to a sidewalk.
- In a disabled person parking space, unless you are disabled and display a placard or disabled person license plates.
- In the space next to a disabled person parking space, if it is painted in a crosshatched (diagonal) pattern. (Fig 1.1)

Fig 1.1. Example of crosshatched (diagonal lines) area

- In a space designated for parking or fueling zero-emission vehicles which display an identifying decal.
- In a tunnel or on a bridge, except where permitted by signs.
    - Within 15 feet of a fire hydrant or a fire station driveway.

- On or within 7½ feet of a railroad track.
- Between a safety zone and the curb.
- “Double parked.” (Parking in the street when all legal parking places at the curb are taken.)
- On the wrong side of the street.
- At a red curb.
- On a freeway, except:
    - In an emergency.
    - When a peace officer or device requires a stop.
    - Where a stop is specifically permitted. A vehicle (even if disabled) that is stopped, parked, or left standing on a freeway for more than 4 hours may be removed.

Painted colored curbs have the following special parking rules:

White–Stop only long enough to pick up or drop off passengers or mail.

Green–Park for a limited time. Look for a posted sign next to the green zone for time limits, or locate the time limit painted on the curb.

Yellow–Stop no longer than the time posted to load or unload passengers or freight. Drivers of noncommercial vehicles are usually required to stay with the vehicle.

Red–No stopping, standing, or parking. (Buses may stop at a red zone marked for buses.)

Blue–Parking is permitted only for a disabled person or driver of a disabled person who displays a placard or special license plate for disabled persons or disabled veterans. Disabled people with a placard or special plates may park in special areas for unlimited periods of time, regardless of time restrictions.

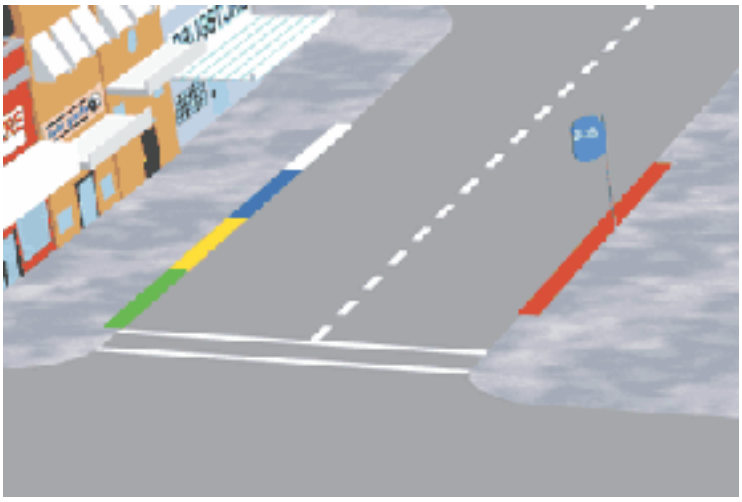

Fig 2. Parking at Colored Curb

An FPGA can be considered a programmable special-purpose processor as it can handle signals at its input pins, process these and drive signals on its output pins. Using Field Programmable Gate Array, has various benefits such as speed, hardware acceleration and parallelism, increasing reliability by having fewer on-board devices, long-term maintenance. FPGA uses fewer clock cycles to process larger data, and the update on FPGA is easier as no change in circuit layout is required. Altera Cyclone II EP2C20F484C7N FPGA is used to implement the proposed design.

IoT uses Wi-Fi to exchange data wirelessly for large distances using Internet. IoT module (ESP8266) is used to communicate the signal from the sensors to the desired output device. IoT will soon be driven by field-programmable gate array (FPGA)-like devices, because these devices can interface with the outside world very easily and provide lowest power, lowest latency and best determinism.

CHAPTER 2

HARDWARE AND SOFTWARE REQUIREMENTS

Altera Cyclone II EP2C20F484C7N FPGA

Cyclone series FPGAs and SoC FPGAs are the company's lowest cost, lowest power FPGAs, with variants offering integrated transceivers up to 5 Gbit/s. Cyclone II FPGAs are perfectly suited as an embedded processor or microcontroller when combined with Intel's 32-bit Nios II embedded processor IP cores. The Cyclone II FPGA Starter Development Kit features:

- Cyclone II Starter Development Board
- Cyclone II EP2C20F484C7N device
- Configuration
    - USB-Blaster™ download cable (embedded)
    - EPCS4 serial configuration device
- Memory
    - 8-Mbyte SDRAM
    - 512-Kb SRAM
    - 1- to 4-Mbyte flash
- Clocking
    - SMA connector (external clock input)
- Audio
    - 24-bit coder/decoder (CODEC)

- Switches and indicators
    - Ten switch and four push buttons
    - Four, 7-segment displays
    - Ten red and eight green LEDs

- Connectors
    - VGA, RS-232, and PS/2 ports
    - Two 40-pin expansion ports
    - SD/MMC socket

Quartus II 13.0sp

The Altera Quartus II design software is a multiplatform design environment that easily adapts to your specific needs in all phases of FPGA and CPLD design. Quartus II software delivers the highest productivity and performance for Altera FPGAs, CPLDs, and HardCopy ASICs. Quartus II software delivers superior synthesis and placement and routing, resulting in compilation time advantages. Compilation time reduction features include:

• Multiprocessor support

• Rapid Recompile

• Incremental compilation

The Altera Quartus II design software provides a complete design environment that easily adapts to your specific design requirements.

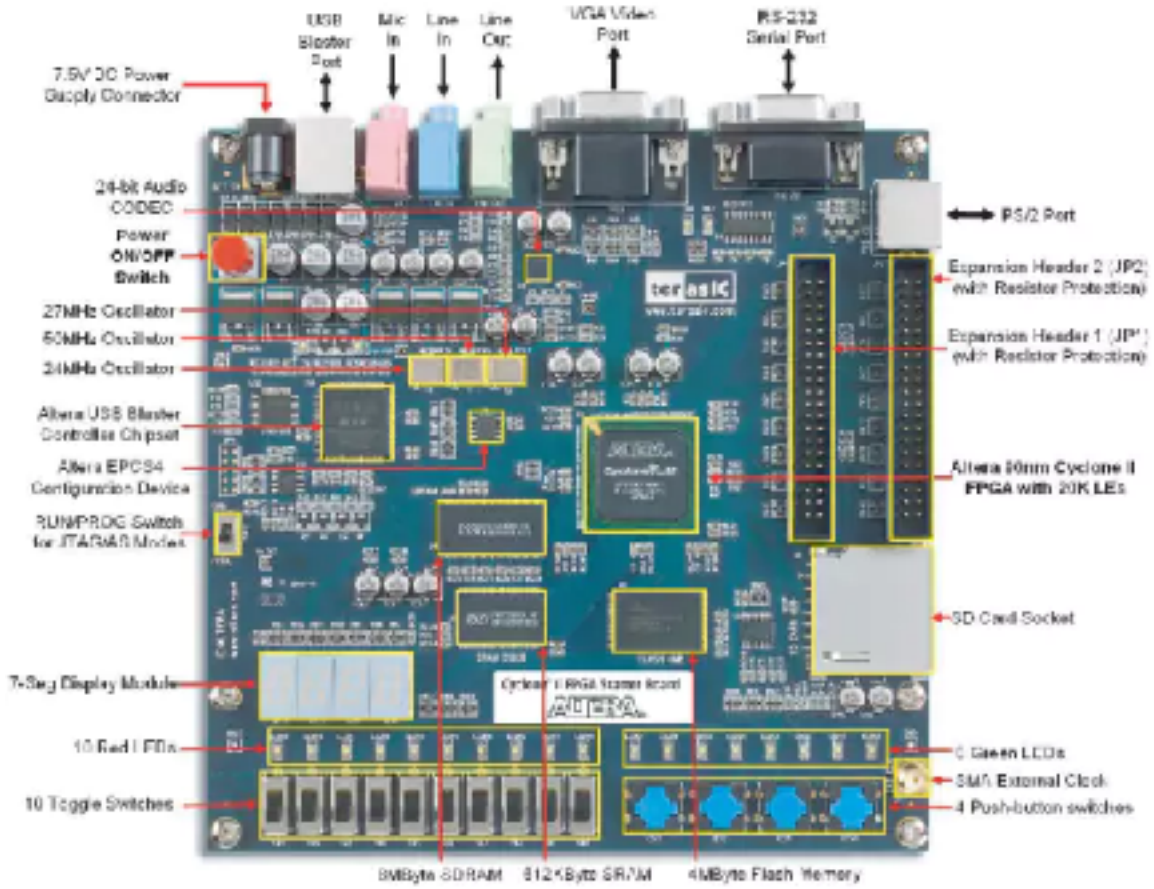

Fig 2.1. Cyclone II FPGA Starter Development Kit

Wi-Fi Module (ESP8266)

The ESP8266 Wi-Fi Module is a self-contained SOC with integrated TCP/IP protocol stack that can give any FPGA or microcontroller access to your Wi-Fi network. The ESP8266 is capable of either hosting an application or offloading all Wi-Fi networking functions from another application processor. Each ESP8266 module comes pre-programmed with an AT command set firmware.

1. This module has a powerful enough on-board processing and storage capability that allows it to be integrated with the sensors and other application specific devices through its GPIOs with minimal development up-front and minimal loading during runtime. Its high CH_PD, Chip Power Down

degree of on-chip integration allows for minimal external circuitry, including the front-end module, is designed to occupy minimal PCB area.

The Pinout is as follows for the basic module,

2. VCC, Voltage (+ 3.3 V (upto 3.6 V it can handle))
3. GND, Ground (0 V)
4. RX, Receive data bit X
5. TX, Transmit data bit X
6. RST, Reset
7. GPIO 0, General Purpose Input-Output No. 0
8. GPIO 2, General Purpose Input-Output No. 2

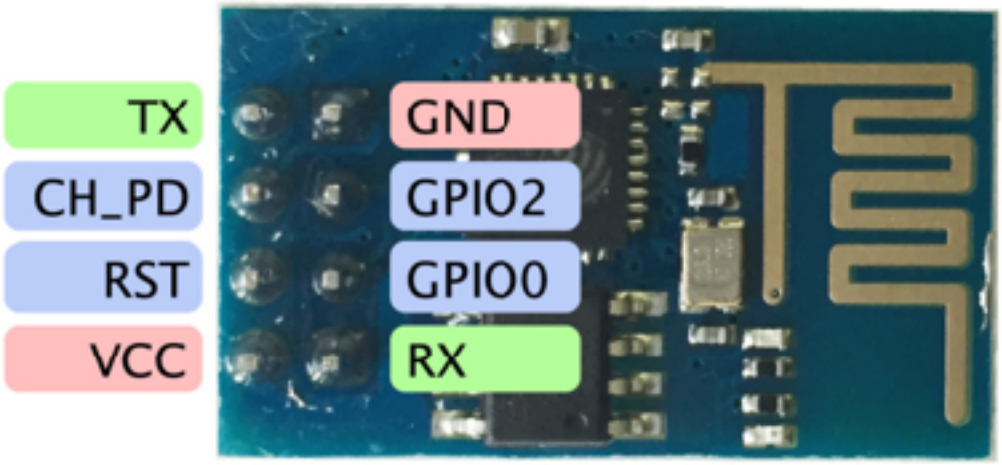

Fig 2.2. ESP8266

IFTTT Applet

If This Then That, also known as IFTTT, is a free web-based service to create chains of simple conditional statements, called *applets*. We know that the ESP8266 module can be configured both as AP or STA. Here we have configured it to work as station and have connected it to our Wi-Fi Router. Once the connection is establish we have to send SMS online.

This must also be easily accessible by our ESP8266 module. Here is where we leverage the power of IFTTT (If This Then That) website. Using this website we can send SMS, E-mail, WhatsApp messages, Facebook updates, Twitter tweets. They have a lot of tools that can be used with little knowledge, but in this project we are going to use the send message feature.

After few tweaks in the IFTTT website we will able to get a HTTPS URL which when triggered will send a predefined Text message to a specific mobile number. Now, this URL has to be called by our ESP8266 module when needed and the SMS will be fired to your mobile number.

Sensors/ Switch

A sensor is a device that detects and responds to some type of input from the physical environment. The specific input could be light, heat, motion, moisture, pressure, or any one of a great number of other environmental phenomena. The output is generally a signal that is converted to human-readable display at the sensor location or transmitted electronically over a network for reading or further processing.

Here our aim is to detect the presence of car and this can be attained by the use of various sensors available in the market. In our project we just have implemented the output of a basic switch from our FPGA as the sensor output i.e. if the switch is HIGH it corresponds that the sensor output is HIGH else the output signal is low. This particular signal is used to trigger our Wi-Fi Module.

CHAPTER 3

DESIGN AND IMPLEMENTATION

The Automated Parking Ticket Generator using FPGA is designed for detecting if any vehicle is parked at any of the no permitted parking spots and then sending a text message using the ESP8266 Wi-Fi, which is integrated with the IFTTT applet. The ESP8266 was preprogrammed using the AT commands using the Arduino IDE. The basic working of the device is shown using a block diagram in Fig.

The sensor/switch if is in HIGH state, then a signal is sent to the FPGA corresponding that illegal parking has been detected, the FPGA further triggers a signal to the ESP8266 which is then activated and performs its function of sending a text message using a IFTTT applet which was initially created.

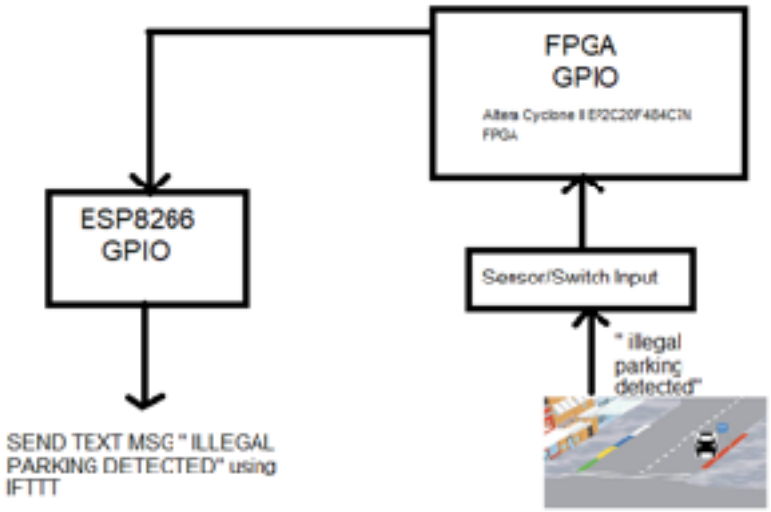

Fig 3.1: Block Diagram of Automated Parking Ticket Generator

3.1 STATE TABLE

Reset	Switch/sensor State	FPGA I/O	ESP8266 I/O	c_mode	OPERATION
1	1	1	1	0	" Send Text Message"
1	0	0	0	0	" Don't send Text Message"
0	X	X	X	1	" reset mode for ESP8266 or AT cmd mode"

Table 3.1: State Table

The switch/sensor signal is connected to one of the pins on the FPGA, and the GPIO of ESP8266 is also connected to one of the I/O ports of the FPGA. When the switch/ sensor output is in HIGH state then the FPGA pin connected to it is turned to logic 1 which further turns the pin connected to ESP8266 GPIO to logic 1. The ESP8266 Wi-Fi module then triggers a signal through which the message is sent.

3.2 FLOW CHART

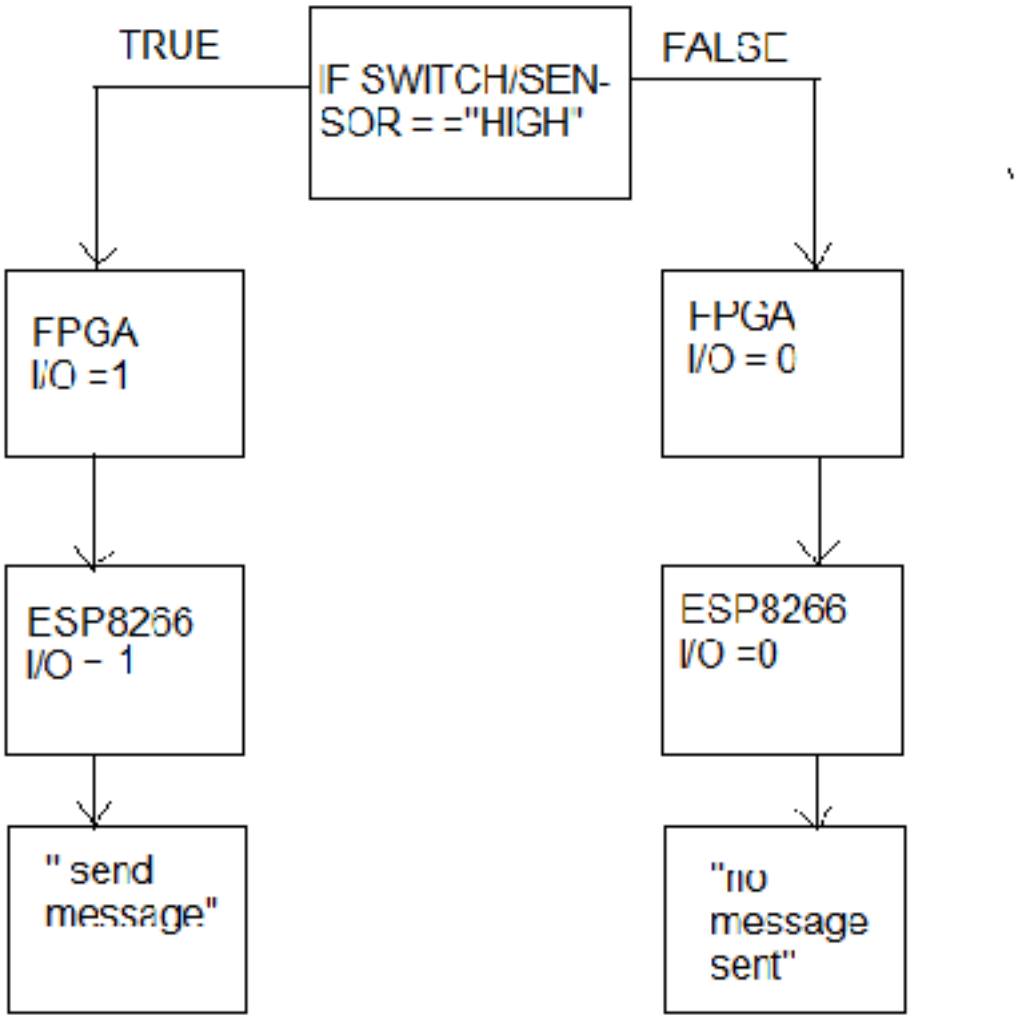

Fig 3.2.1: Flow Chart

There are two cases in the flow chart, if the condition about the sensor/switch is true then the GPIO pins of the FPGA are turned to logic 1 and the message is triggered but if the condition is false then the GPIO pins of the FPGA are turned to logic 0 and no message is sent.

3.3 ESP8266 AT Commands

The AT Commands of the ESP8266 Wi-Fi Module are responsible for controlling all the operations of the module like restart, connect to Wi Fi, change mode of operation and so forth. So, it is important to understand the ESP8266 AT Commands

Basically, the ESP8266 AT Commands can be classified into four types:

Test Commands: The Test AT Commands of ESP8266 Wi-Fi Module are used to get the parameters of a command and their range.

Query Commands: The Query Commands returns the present value of the parameters of a command.

Set Commands: The Set Commands are used set the values of the parameters in the commands and also runs the commands.

Execute Commands: The Execute Commands will run the commands without parameters.

The ESP8266 AT Commands Set is divided into three categories. They are:

- Basic AT Commands
- Wi Fi AT Commands
- TCP/IP AT Commands

A few important ones of AT commands, which are utilized, are shown in the table below with their descriptions,

Commands	Description	Set/Execute	Parameters
AT+RST	restart the module	–	–
AT+CWMODE	wifi mode	AT+CWMODE=<mode>	1= Sta, 2= AP, 3=both
AT+CWJAP	join the AP	AT+ CWJAP =<ssid>,< pwd >	ssid = ssid, pwd = wifi password
AT+CWLAP	list the AP	AT+CWLAP	
AT+CWQAP	quit the AP	AT+CWQAP	
AT+ CWSAP	set the parameters of AP	AT+ CWSAP= <ssid>,<pwd>,<chl>, <ecn>	ssid, pwd, chl = channel, ecn = encryption
AT+ CIPSTATUS	get the connection status	AT+ CIPSTATUS	
AT+CIPSTART	set up TCP or UDP connection	1)single connection (+CIPMUX=0) AT+CIPSTART= <type>,<addr>,<port>; 2) multiple connection (+CIPMUX=1) AT+CIPSTART= <id><type>,<addr>, <port>	id = 0-4, type = TCP/UDP, addr = IP address, port= port
AT+CIPSEND	send data	1)single connection(+CIPMUX=0) AT+CIPSEND=<length>; 2) multiple connection (+CIPMUX=1) AT+CIPSEND= <id>,<length>	
AT+CIPCLOSE	close TCP or UDP connection	AT+CIPCLOSE=<id> or AT+CIPCLOSE	
AT+CIFSR	Get IP address	AT+CIFSR	
AT+ CIPMUX	set mutiple connection	AT+ CIPMUX=<mode>	0 for single connection 1 for mutiple connection
AT+ CIPSERVER	set as server	AT+ CIPSERVER= <mode>[,<port>]	mode 0 to close server mode, mode 1 to open; port = port
+IPD	received data		

One of the ways to programming the ESP8266 is throughout the Arduino micro-controller board. The ESP8266 is programmed using AT commands; when received, it replies with an acknowledgment. The ESP8266 is powered up using a 3.3 Volts power source, An Arduino Uno will be able to power up the ESP through its regulated 3.3 V power pin.The ESP8266 is connected to the Arduino through TXD, RXD, GND, and the VCC pins to the RX, TX, GND and 3.3 V pins, respectively.

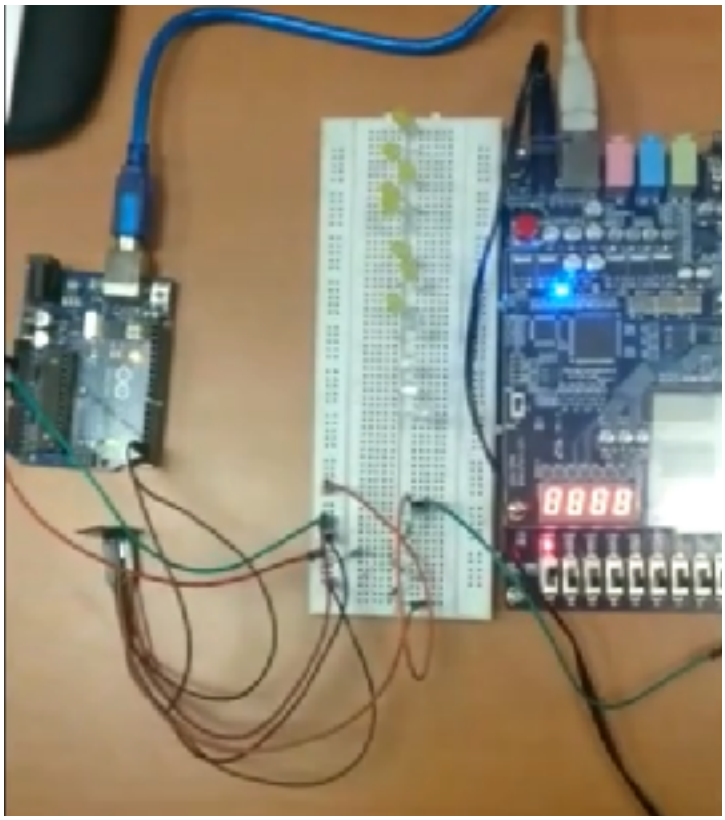

Fig 3.2: Interfacing

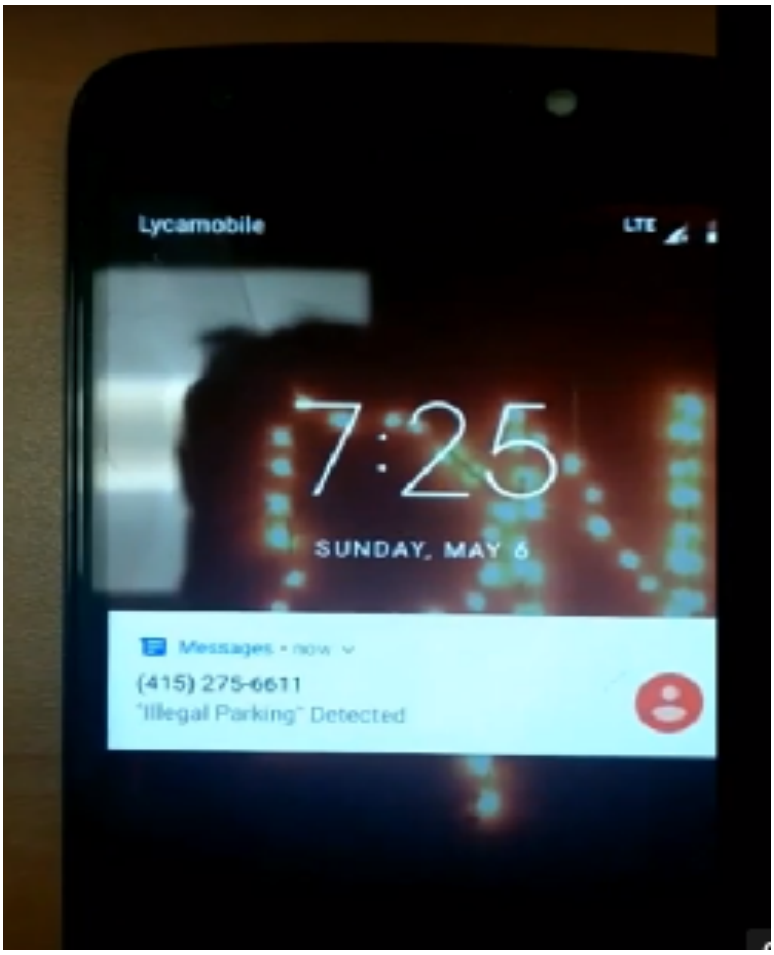

Fig 3.3: Text message received

3.4 Pin Planner

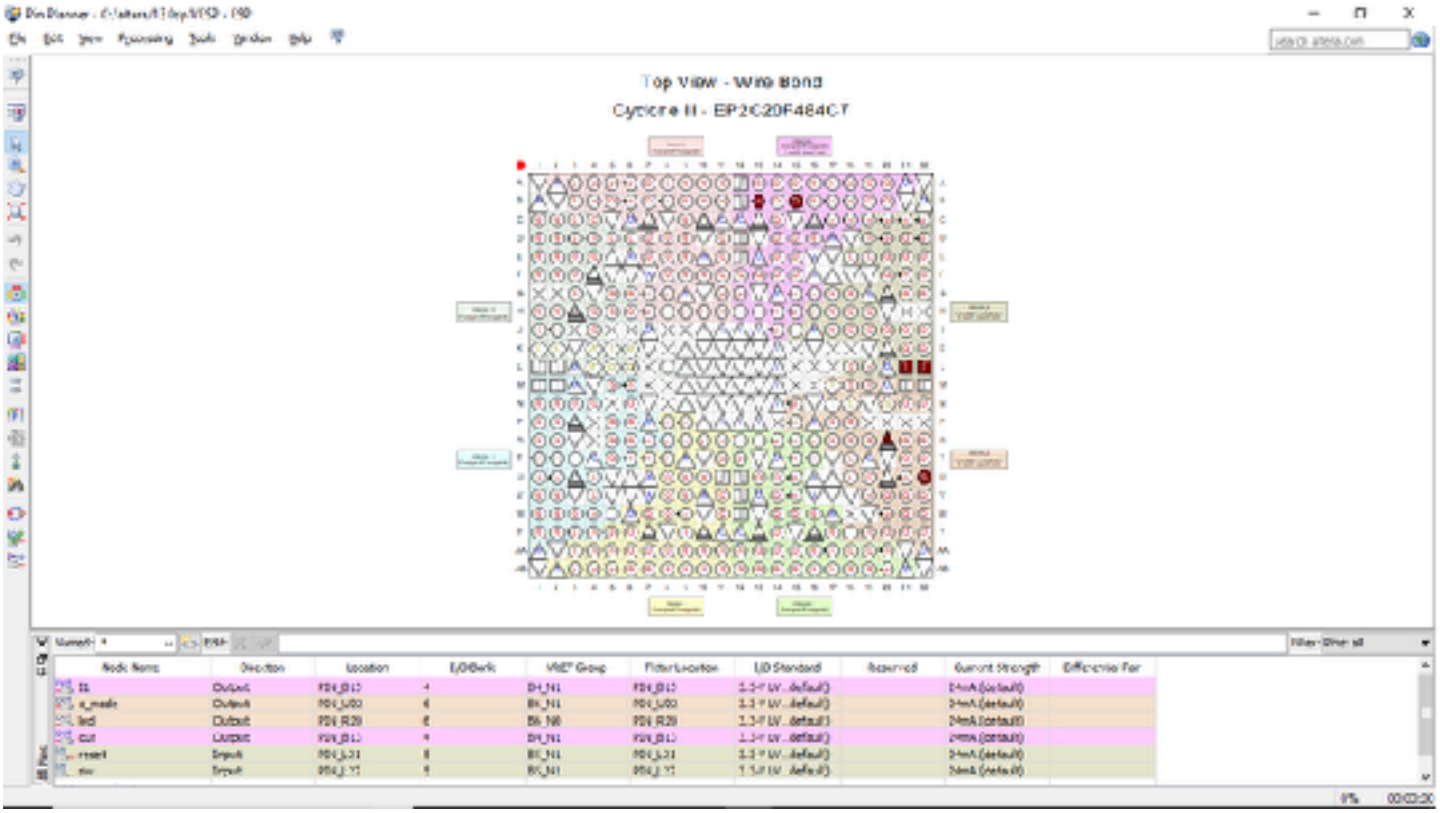

Fig 3.4 Pin Planner

CHAPTER 3

SIMULATION RESULTS

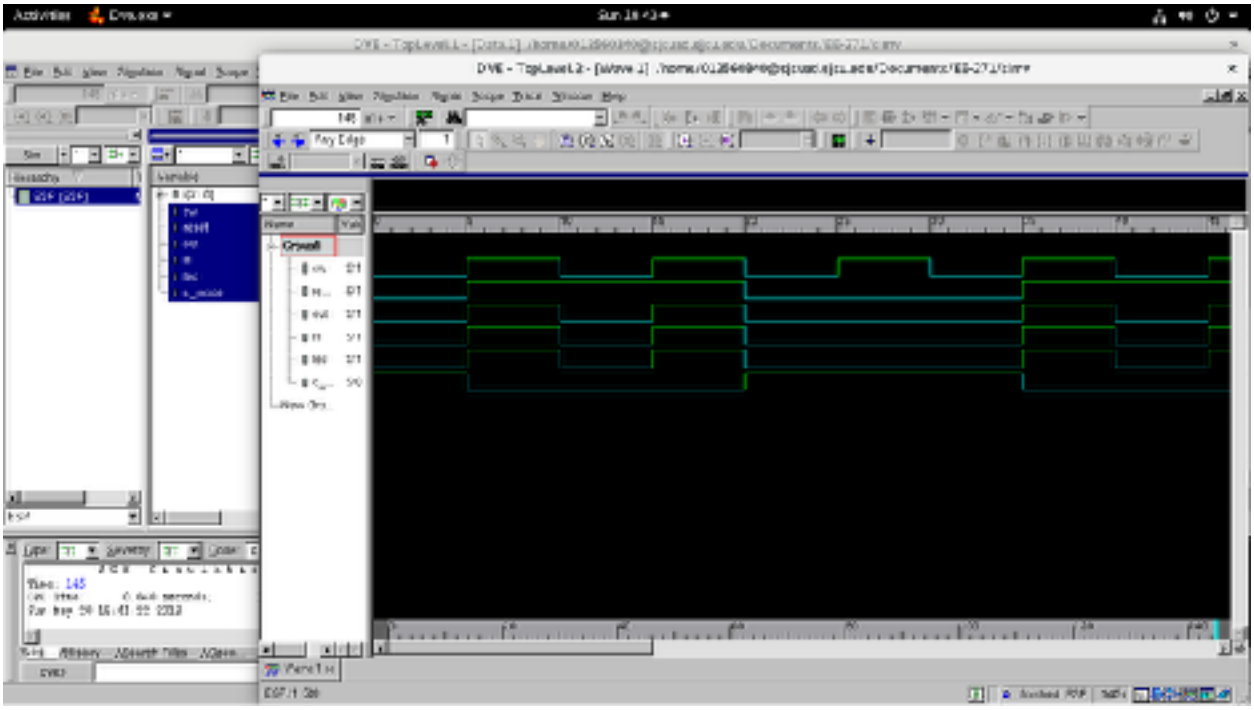

Fig3.1: Waveform

The simulation is performed using Synopsys VCS based on the design discussed above.

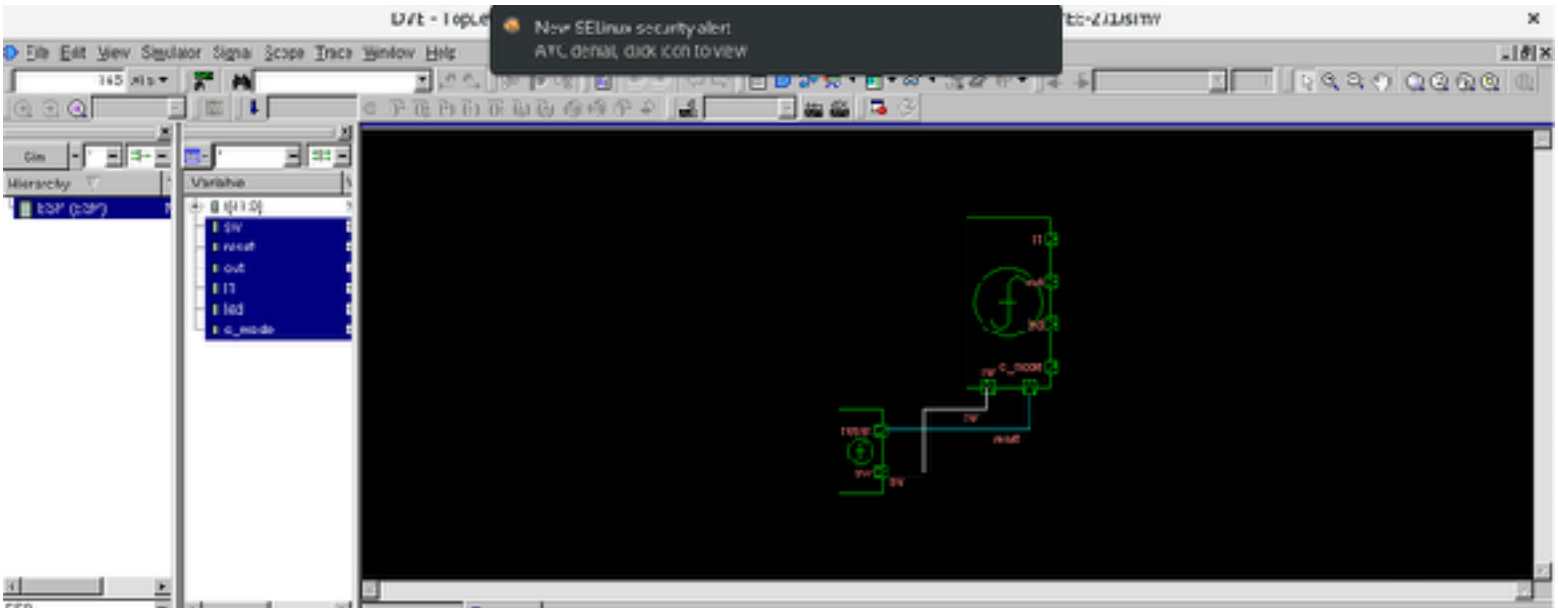

Fig 3.2 RTL View

CHAPTER 4

CONCLUSION

- With our project, we have tried to implement a prototype of a system, which can automatically detect an illegal parked vehicle and notify the Office of Parking Violations.
- The Wi Fi module used in our project makes our device based on IoT application which is one of the emerging technology of this era.
- Our implementation of a switch instead of a specified sensor was a counter measure to demonstrate the universality of all the sensors which can be integrated with the FPGA.
- We have used a FPGA because a lot of other sensor I/Os can be linked to it and a complete parking ticket management system can be implemented based on this prototype.

CHAPTER 5

REFERENCES

- California Driver handbook – Parking https://www.dmv.ca.gov/portal/dmv/detail/pubs/hdbk/parking
- Do you know your curb colors?

  http://blog.esurance.com/do-you-know-your-curb-colors/
- ESP8266 datasheet

  https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
- ESP8266 AT instruction set

  https://www.espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf
- IFTTT https://ifttt.com/