**A**

**MAJOR PROJECT REPORT ON**

# DESIGN AND IMPLEMENTATION OF CAR PARKING MANAGEMENT SYSTEM USING VERILOG

**Submitted in partial fulfilment of the requirement for the award of degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

## ELECTRONICS AND COMMUNICATION ENGINEERING

SUBMITTED BY


| | |
|---|---|
| **ABHISHEK KUMAR** | **218R1A0466** |
| **A.HARI PRIYA** | **218R1A0467** |
| **B.SANTHOSH VARMA** | **218R1A0469** |

Under the Esteemed Guidance of

**G.KALPANA**
Associate professor




**DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING**

# CMR ENGINEERING COLLEGE

## UGC AUTONOMOUS

**(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA)**

**Kandlakoya(V), Medchal(M), Telangana – 501401**


**(2024-2025)**

# CMR ENGINEERING COLLEGE

## UGC AUTONOMOUS

**(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA)**

**Kandlakoya(V), Medchal Road, Hyderabad - 501 401**

### DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

## CERTIFICATE

This is to certify that the major-project work entitled **"DESIGN AND IMPLEMENTATION OF CAR PARKING MANAGEMENT SYSTEM USING VERILOG"**is being submitted by **ABHISHEK KUMAR** bearing Roll No **218R1A0466, A.HARI PRIYA** bearing Roll No **218R1A0467, B.SANTHOSH VARMA** bearing Roll No **218R1A0469** in B.Tech IV-II semester, Electronics and Communication Engineering is a record Bonafide work carried out during the academic year 2024-25. The results embodied in this report have not been submitted to any other University for the award of any degree.

INTERNAL GUIDE
**Mrs.G.KALPANA**

HEAD OF THE DEPARTMENT
**Dr. SUMAN MISHRA**

**EXTERNAL EXAMINER**

# ACKNOWLEDGEMENTS

We sincerely thank the management of our college **CMR Engineering College** for providing required facilities during our project work. We derive great pleasure in expressing our sincere gratitude to our Principal **DR. A. S. REDDY** for his timely suggestions, which helped us to complete the project work successfully. It is the very auspicious moment we would like to express our gratitude to **Dr. SUMAN MISHRA,** Head of the Department, ECE for his consistent encouragement during the progress of this project.

We take it as a privilege to thank our project coordinator **Dr. T. SATYANARAYANA**, Associate Professor, Department of ECE for the ideas that led to complete the project work and we also thank him for his continuous guidance, support and unfailing patience, throughout the course of this work. We sincerely thank our project internal guide **Mrs.G.KALPANA,** Associate Professor of ECE for guidance and encouragement in carrying out this project work.

# DECLARATION

We hereby declare that the major project entitled **"DESIGN AND IMPLEMENTATION OF CAR PARKING MANAGEMENT SYSTEM USING VERILOG"** is the work done by us in campus at **CMR ENGINEERING COLLEGE,** Kandlakoya during the academic year 2024-2025 and is submitted as major project in partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING FROM JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD.**


**ABHISHEK KUMAR (218R1A0466)**

**A.HARI PRIYA (218R1A0467)**

**B.SANTHOSH VARMA (218R1A0469)**

# ABSTRACT

The increasing number of vehicles on the road has led to a significant challenge in urban areas: efficient car parking management. This project presents the design and implementation of a Car Parking Management System (CPMS) using Verilog, a hardware description language widely used for digital circuit design. The primary objective of this system is to automate the parking process, thereby enhancing the efficiency of space utilization and minimizing the time spent by drivers searching for available parking spots.

The proposed CPMS is designed to manage multiple parking slots, providing real-time information about availability and guiding vehicles to open spaces. The system employs a combination of sensors and control logic implemented in Verilog to detect vehicle presence, monitor parking slot occupancy, and facilitate user interaction through a simple interface. The architecture includes a state machine that governs the operation of the parking system, ensuring seamless transitions between different states such as parking, exiting, and slot availability checks.

The main aim of this work is to implement a cellar parking system based on sensor detection. The first part involves the designing of a counter that indicates the count of slots (parking slots) filled or vacant and the second part involves the designing of an indicates filled & vacant using LCD in order to prevent forward collision between vehicles. The led values depend upon the detection of sensor values which indicates the presence of a object. The value of LED indicates the entry and the exit of vehicles from the slots, which in return specifies the presence of vehicles opposite to the other vehicles at the entrance of the slots. xilinx ISE Design Suite 14.7 Version are used for simulation and implementation of design.

# CONTENTS

**CHAPTERS** **PAGE**

# LIST OF FIGURES

# CHAPTER-1
# INTRODUCTION

We know what happens when vehicle drivers cannot find a parking space near their destination: they circle the block until their luck changes. Plenty of research has documented the great role this takes on traffic congestion and air pollution.

The field's leading scholar, UCLA parking guru Donald Shoup, has found that cruising for parking can account for some 15% of downtown traffic at the lower end of the spectrum, and up to 74% at the extremes**.** So much time is spent on the streets looking for a parking space – it's an awful waste of time and productivity, coming at the cost of peace of mind, and essentially pushing our roads off the edge with the heightened congestion.

## 1.1 OVERVIEW OF PARKING ISSUES

The increasing number of vehicles in urban areas has led to a significant challenge in finding adequate parking spaces. While traffic congestion remains a common issue, the lack of proper parking arrangements has emerged as a more pressing concern. Many office spaces, despite employing a considerable workforce, do not have dedicated parking facilities. As a result, employees are forced to park their vehicles on the streets, contributing to road congestion, especially during peak hours. Additionally, unauthorized parking increases the risk of vehicle damage and security concerns.

This issue is not confined to a single city but is prevalent across both Indian and international metropolitan areas. Rapid economic growth and urban migration have intensified the pressure on infrastructure, leading to inadequate parking solutions. Studies indicate that urban drivers spend an average of 18 to 20 minutes searching for parking, which results in stress, fuel wastage, increased emissions, and reduced productivity. In several global cities, commuters report spending even longer periods searching for a parking spot, further worsening traffic congestion.

Research also highlights that in densely populated cities, commuters spend several additional hours annually driving extra miles in search of parking. The rise in the number of vehicles has not been matched by an increase in available parking spaces, as urban land remains limited and expensive. This imbalance has severely impacted mobility and accessibility, particularly in metropolitan regions where land scarcity is a critical concern. As the demand for parking continues to rise, innovative solutions and better urban

planning are necessary to address the growing challenges of urban mobility.

A study indicates that commuters in major cities spend over 80 hours annually, driving more than 150 miles extra in search of parking spaces. While the number of vehicles on the roads has increased rapidly, the available space in urban areas has either remained constant or decreased due to growing population pressure. This imbalance is a key factor affecting mobility and accessibility in cities. The issue is particularly severe in metropolitan regions, where land is both scarce and expensive, and the rising demand for parking has further strained existing land resources.

**Fig 1.1: Absence of parking space.**

## 1.2 PARKING CRISIS IN URBAN AREAS

In the absence of parking spaces, most of the cars are parked on streets in residential areas. This is the condition of a colony in Malviya Nagar area in Delhi. Another study shows how public transport plays an important role in relieving pressure off existing roads and spaces. The Center for Science and Environment found that in Connaught Place, Delhi, parking demand declined by 10 percent after the metro came up. By developing a stronger mass transit network, valuable space given to parking lots may be reduced to a great extent. While city governments and planners have to work towards discouraging the use of

private vehicles in favor of public transport in the long term, providing parking spaces for existing vehicles is a challenge today. Several solutions have been suggested by planners to tackle the problem. Smart sensor-based parking is a way that is being adopted increasingly by cities to relieve pressure and optimize the use of available spaces.

In the above system, whenever a car gets parked in the parking slot, it is detected by the sensor. Real-time information from the sensors is stored on a server and then sent on a mobile app. This provides the driver with a real-time map of available spaces. This would also allow drivers to reserve a parking spot at a convenient parking lot based on their preference, the traffic condition, and availability of public transport. This significantly cuts down the amount of time spent searching for parking spaces and, crucially, reduces the environmental impact resulting from congestion and emission in urban areas.


**Fig 1.2: Working of sensor-based parking**

The image illustrates a Smart Car Parking System that utilizes wireless technology, real-time data processing, and digital communication to enhance parking efficiency. The system consists of wireless sensors embedded in parking spots, which detect vehicle presence and transmit real-time availability data to a central platform. This information is

then displayed on Dynamic Message Signs, helping drivers quickly identify vacant parking spaces. Additionally, a Parking App enables users to check space availability, navigate to an open spot, and make cashless payments, reducing the need for physical transactions. By integrating these technologies, the system offers several benefits. It significantly reduces driving time spent searching for parking, which in turn lowers fuel wastage and minimizes traffic congestion. The optimized use of parking spaces ensures efficient resource allocation and better urban mobility. Furthermore, the system provides real-time space availability updates and facilitates parking pattern analysis, allowing city planners to make data-driven decisions for improved infrastructure management. Ultimately, this smart parking solution enhances convenience for drivers, promotes sustainability, and contributes to the development of smarter and more efficient cities.

# CHAPTER-2

# LITERATURE SURVEY

## 2.1 AUTOMATIC PARKING LED SYSTEMS

In “Automatic Car Parking System with Visual Indicator along with IoT”, it focuses on the concept of car parking detection mechanism using the ultrasonic sensor, in combination with the usage of Internet of Things i.e., sending the status of the parking slot to the Internet. Through which the user at any place in the world can see which parking slot is empty and where to park. This is done by sending the data of the ultrasonic sensor through the Wi-Fi module that is ESP8266 to any open source easy to use IOT platform that uses HTTP to display the data. The status of the slot is shown with the help of the indicators and through IoT. The table 1 below shows the observation of the particular slot during the time of the day. In the table, various car models are shown, and the output is checked, thus confirming that the research was done up to the mark and showing the output correctly both in hardware and in thingspeak, irrespective of the car model.

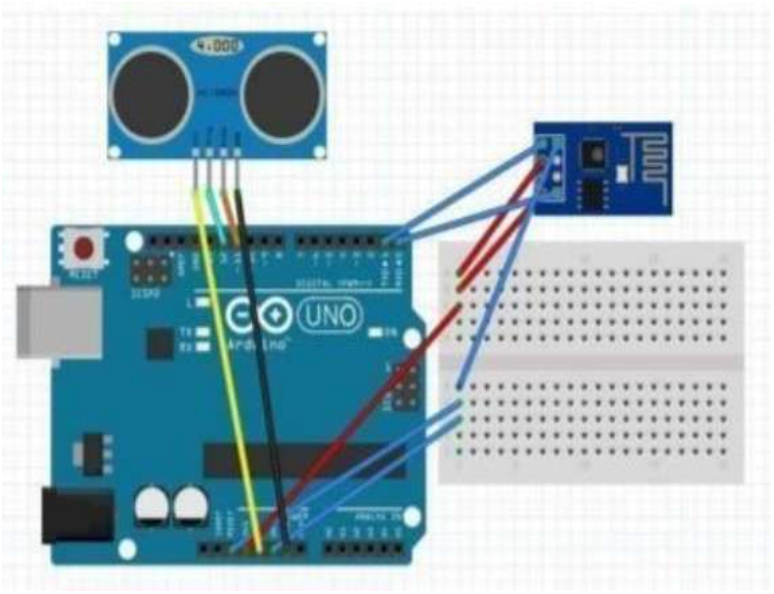



**Fig 2.1: Circuit connections**

In “Real Time Car Parking System using Image Processing” Car parking lots are an important object class in many traffic and civilian applications. With the problems of increasing urban traffic congestion and the increasing shortage of space. The vision-based method makes it possible to manage a large area by just using several cameras. It is consistent in detecting incoming cars because it uses actual car images. Goals of intelligent parking lot management include counting the number of parked cars, and identifying the available location. This work proposes a new system for providing parking information

and guidance using image processing. The proposed system includes counting the number of parked vehicles, and identifying the stalls available. The system detects cars through images instead of using electronic sensors embedded on the floor. A camera is installed at the entry point of the parking lot. It will capture image sequences.


**Fig 2.2: Experimental output**

In "Automatic Car Parking Indicator System" The basic concept is, when a car enters the car park through the entry point, if there is an empty parking slot, the barrier will allow access to the car park. One big display is situated after entry which shows the remaining parking slots. As soon as the car enters through entry, the sensor situated at entry gets activated and shows the empty parking slot. If there are no slots available for parking It shows the message that all lots are full and drivers have to wait for some time. The parking space will be monitored by an AVM camera. Ultrasonic sensors are mounted in the road, to provide exact direction to the car on the lane. As soon as the car enters, the driver gets information of the filled slots and empty slots on the big display board. As the driver moves further, he will get a message of an allotted parking slot and navigation for that slot on a small display board situated in the inner lane. When all slots are full, No slots available message displays on the display board. The above principle is illustrated by the following diagram. As soon as the car enters, the driver gets information of the filled slots and empty

slots on the big display board. As the driver moves further.

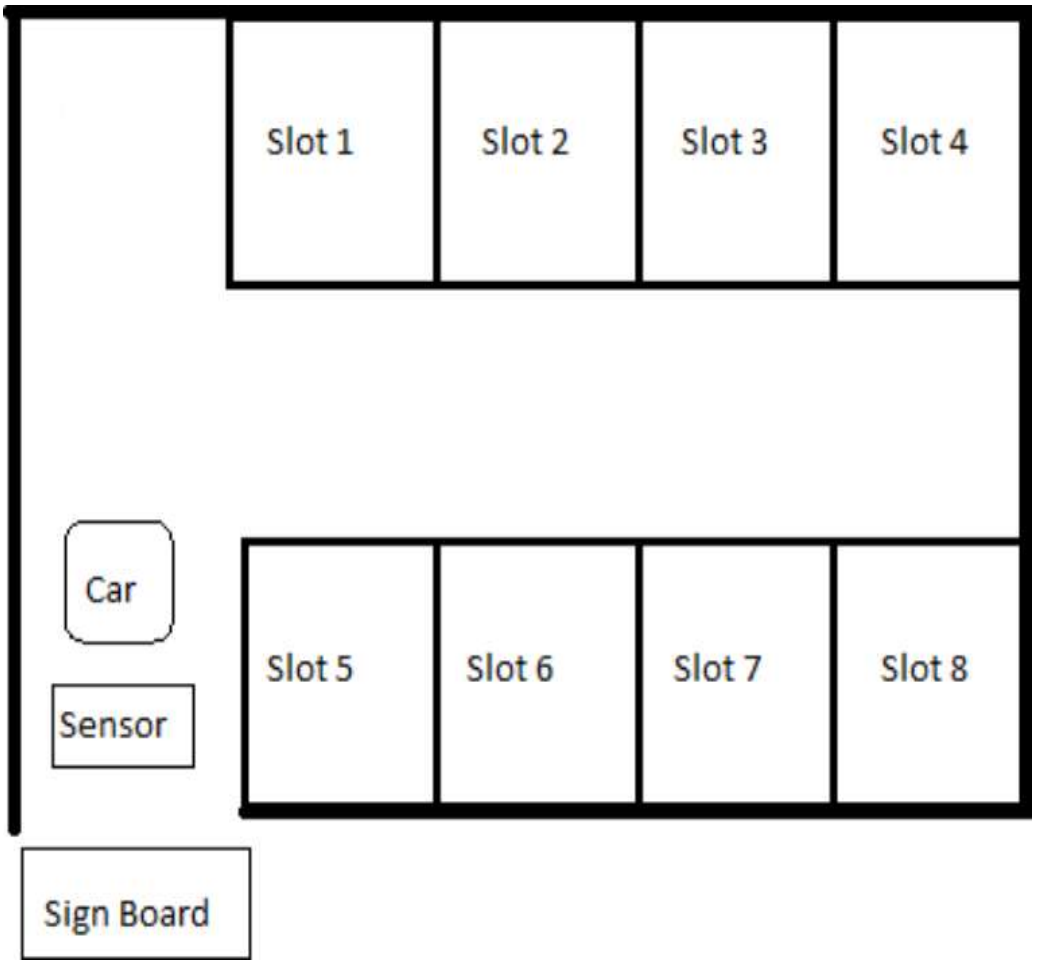



**Fig 2.3: Car Parking Operation Principle**

"Towards a Fog Enabled Efficient Car Parking Architecture" Fog computing, bringing the cloud computing resources in proximate vicinity to the network edge, overcomes not only the latency issue but also provides significant improvements, such as on-demand scaling, resource mobility, and security. The primary motivation to employ fog computing in the proposed approach is to minimize the latency as well as network usage in the overall smart car parking system. To demonstrate the effectiveness of the proposed approach for reducing the lag and network usage, simulations have been performed in iFogSim and the results have been compared with that of the cloud-based deployment of the smart car parking system. Experimental results exhibit that the proposed fog-based implementation of the efficient parking system minimizes latency significantly. It is also observed that the proposed fog-based implementation reduces the overall network usage in contrast to the cloud-based deployment of smart car parking. A limitation of the proposed research is the use of cameras for parking space detection. This may give rise to the privacy issues for the car owners as the images are stored in the cloud. Considering the fact that most of the

storage and processing is handled on the nearby fog nodes. Though, there is a need to preserve the privacy of data in cloud storage by applying suitable encryption techniques and can be an important direction for future work.

“An Autonomous Parking System of Optimally Integrating Bidirectional Rapidly-Exploring Random Trees* and Parking-Oriented Model Predictive Control” Autonomous parking techniques can be used to tackle the lacking problem of parking spaces. In this paper, a sampling-based motion planner consisting of optimizing bidirectional rapidly-exploring random trees* (Bi-RRT*) and parking-oriented model predictive control (MPC) is proposed to properly deal with various parking scenarios. The optimal Bi-RRT* approach aims to improve the common defects of traditional sampling- based motion planners, such as uncertainties of path quality and consistency, and exploring inefficiency in narrow spaces. For this reason, the proposed motion planner is able to overcome strict environments with obstacles and narrow spaces. The parking- oriented MPC is then designed for steering and speed controls simultaneously for accurately and smoothly tracking parking paths. Furthermore, the proposed controller is dedicated to work under the practical scenarios, such as vehicle considerations, real-time control, and signal delay. To verify the effects of the proposed autonomous parking system, extensive simulations and experiments are conducted in common and strict parking scenarios, such as perpendicular parking, parallel parking. The simulation results not only verify the effects of each technical element, but also show the capability to deal with the various parking scenarios. Furthermore, various on-car experiments sufficiently demonstrate that the proposed system can be actually implemented in everyday life.

“Path Loss Models for Low-Power, Low-Data Rate Sensor Nodes for Smart Car Parking Systems” Smart parking management systems need to keep up with the state of parking spots at all times. This is efficiently accomplished via sensor nodes that detect vehicles and update the system’s state as it changes in real time. When sensor nodes use wireless communication as their primary communication link, the deployment approach becomes important, since it directly affects network connectivity, cost, and lifetime. The main factor to consider when deploying wireless sensor networks (WSN) in parking environments is the prediction of radio frequency (RF) signal propagation. Inaccurate propagation models lead to systems that under- or over-perform, both of which negatively affect WSN performance. Most of the existing RF propagation models are created to support cellular systems environments, which drastically differ from indoor/outdoor parking environments; few or no models exist that accurately predict RF signal propagation in parking

environments. Therefore, there is a need for models that accurately characterize RF signal propagation in these environments. This paper proposes empirical path loss models for WSN deployment in indoor and outdoor car parking lots. The proposed models are compared with theoretical models. Theoretical models deviate from the proposed models and the measured values by 10% to 46%. The provided models, as well as the measured data, can be used for efficient planning and deployment of WSN in various proposed smart cities, intelligent transportation, and parking lot systems.

"Probabilistic Occupancy Filter for Parking Slot Marker Detection in an Autonomous Parking System Using AVM" Many car makers have examined parking assistance systems and auto parking systems that automatically find free parking spaces has demonstrated that a Bayesian regularized neural network exploiting historical data, weather condition, and traffic flow data can offer a robust approach for the implementation of reliable and fast predictions of available slots in terms of flexibility and robustness to critical cases. The solution adopted in Smart City Apps in the Florence area for sustainable mobility has been welcomed with broad appreciation or has been praised as successful.

"Deep Learning-Based Mobile Application Design for Smart Parking" In the era of Internet of Things (IoT) and smart city ecosystems, there is a need for innovative smart and can park a car. The around view monitoring (AVM) can compensate for the disadvantages of distance-sensor-based detection because it can recognize parking spaces based on parking slot markers instead of empty spaces.

However, in the case of AVM-based parking marker recognition, false-positive (FP) features can be detected from 3-D objects and shadows, and the parking slot marker is occluded by the vehicle on which the AVM camera is attached. In this paper, we propose a probabilistic occupancy filter to detect parking slot markers. This filter uses a series of AVM images and onboard sensors to improve the occlusion problem and reduce the FPs from other objects. Each pixel of an AVM image has an occupancy probability of parking slot marker features. The occupancy probability compensates for using vehicle motion and updates using the error model of the AVM system and the performance indicator of the feature extractor.

"An Autonomous Parking System of Optimally Integrating Bidirectional Rapidly-Exploring Random Trees* and Parking-Oriented Model Predictive Control" Autonomous parking techniques can be used to tackle the lacking problem of parking spaces. In this paper, a sampling-based motion planner consisting of optimizing bidirectional rapidly-exploring random trees* (Bi-RRT*) and parking-oriented model predictive control

(MPC) is proposed to properly deal with various parking scenarios. The optimal Bi-RRT* approach aims to improve the common defects of traditional sampling- based motion planners, such as uncertainties of path quality and consistency, and exploring inefficiency in narrow spaces. For this reason, the proposed motion planner is able to overcome strict environments with obstacles and narrow spaces. The parking- oriented MPC is then designed for steering and speed controls simultaneously for accurately and smoothly tracking parking paths. Furthermore, the proposed controller is dedicated to work under the practical scenarios, such as vehicle considerations, real-time control, and signal delay. To verify the effects of the proposed autonomous parking system, extensive simulations and experiments are conducted in common and strict parking scenarios, such as perpendicular parking, parallel parking. The simulation results not only verify the effects of each technical element, but also show the capability to deal with the various parking scenarios. Furthermore, various on-car experiments sufficiently demonstrate that the proposed system can be actually implemented in everyday life.

## 2.2 PARKING SLOT PREDICTION

In “Predicting Available Parking Slots on Critical and Regular Services by Exploiting a Range of Open Data” The selection of suitable car parks could be influenced by multiple factors-e.g., the walking distance to destination, driving and waiting time, parking prices, availability, and accessibility-while the availability of unused parking slots might depend on parking location, events in the area, traffic flow, and weather conditions. This paper presents a set of metrics and techniques to predict the number of available parking slots in city garages with gates. With this aim, we have considered three different predictive techniques, while comparing different approaches. The comparison has been performed according to the data collected in a dozen of garages in the area of Florence by using Sii-Mobility National Research Project and Km4 City infrastructure. The resulting solution is parking systems for more sustainable cities. With the increasing number of vehicles in the cities every year, it takes more time to find parking spaces. The solution methods developed are no longer sufficient. The time that passes while waiting for a parking space in traffic carries with it problems such as energy, environmental pollution and stress. In this study, a deep learning and cloud-based new mobile smart parking application was developed to minimize the problem of searching for parking spaces. Here, dynamic access is provided to the LSTM-based model previously created through the mobile device of the user, and the process of displaying the occupancy rates of the parks at the desired place is accomplished on the mobile device by entering the relevant parameters. By this means,

both energy and time savings have been achieved. With the real-time car parking data collected in the city of Istanbul in Turkey, high accuracy results were obtained. In order to demonstrate the effectiveness of the model proposed, it was compared with the Support Vector Machine, Random Forest and ARIMA methods. The results have confirmed the high accuracy and reliability that was promised. Soh Chun Khang, et.al (2010) presented work on a parking system in which the number of slots which are available for parking is sent as a message to drivers. Drivers can resend sms demanding for a new position when the earlier allotted slot gets filled. Huachun tan, et.al (2009) proposed a system which is helpful to find the park at places where there is a large parking lot. This is done by capturing images through cameras mounted at each parking slot. Information such as the number plate of the car and colour are recognized and stored in data. This data therefore includes information about all cars parked in the lot and hence it is possible to find any car easily. S.V.Srikant, et.al(2009) came up with a system to detect the free parking slots. Author has used wireless communication technology to make the parking system more efficient. Gongjun Yan, et.al (2011), proposed an intelligent parking system which was based on secured wireless system and sensor communication. Autonomous parking techniques can be used to tackle the lacking problem of parking spaces. In this paper, a sampling-based motion planner consisting of optimizing bidirectional rapidly-exploring random trees* (Bi-RRT*) and parking-oriented model predictive control (MPC) is proposed to properly deal with various parking scenarios. The optimal Bi-RRT* approach aims to improve the common defects of traditional sampling- based motion planners, such as uncertainties of path quality and consistency, and exploring inefficiency in narrow spaces. For this reason, the proposed motion planner is able to overcome strict environments with obstacles and narrow spaces. The parking- oriented MPC is then designed for steering and speed controls simultaneously for accurately and smoothly tracking parking paths. Furthermore, the proposed controller is dedicated to work under the practical scenarios, such as vehicle considerations, real-time control, and signal delay. To verify the effects of the proposed autonomous parking system, extensive simulations and experiments are conducted in common and strict parking scenarios, such as perpendicular parking, parallel parking. The simulation results not only verify the effects of each technical element, but also show the capability to deal with the various parking scenarios. Furthermore, various on-car experiments sufficiently demonstrate that the proposed system can be actually implemented in everyday life.

# CHAPTER-3

# EXISTING SYSTEM

Various parking sensors are already installed in some of the public spaces in developed countries which use infrared sensors (hereinafter called as IR Sensors) in combination with ultrasonic sensors to detect the presence of a car in a particular spot. The motivation that drives the result is the pursuit of an alternative solution for the problem, that is instead of using IR Sensors, it would be more efficient to switch to Ultrasonic Sensor which is not affected by variations in the light intensity in a particular environment. Also, instead of using the Ethernet shield or connecting it through LAN cable, a Wi-Fi module (ESP8266) is used. Thus, reducing the cost of cable, increasing the efficiency and making it more feasible to get implemented.

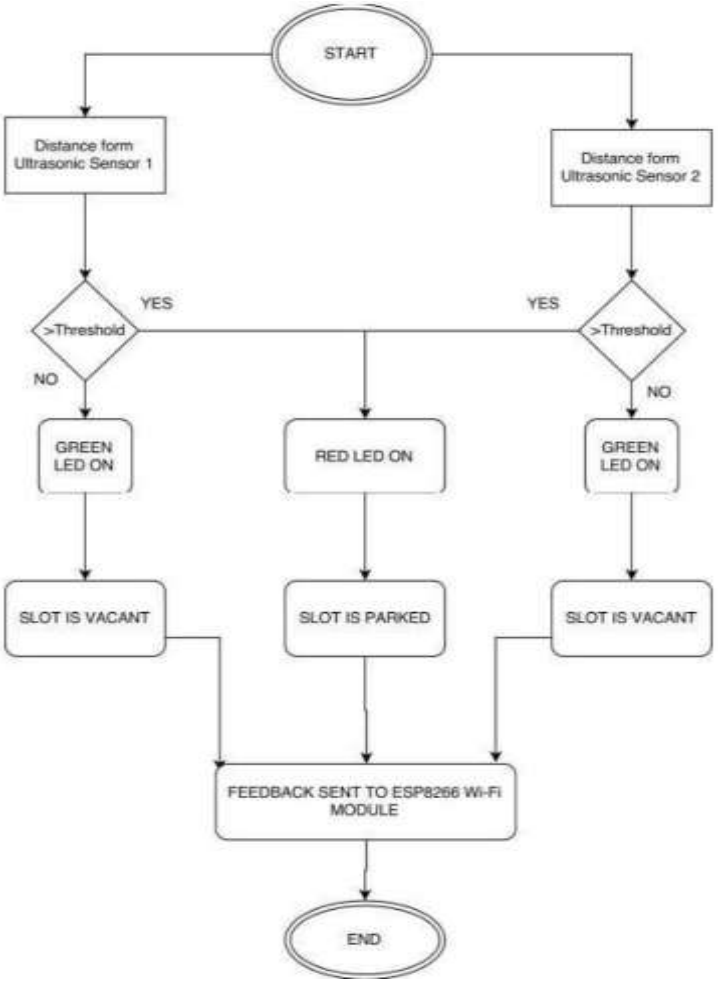



**Fig 3.1: Flowchart of the system**

In the proposed system an LCD screen is placed at the entrance of the parking system and also a sensor which is activated to detect a vehicle coming. When a vehicle enters in, the LCD screen is used to display the availability of vacant slots. If the slots are available to park the vehicle it displays the details of exact vacant slots along with directions. If the slots are not available to park the vehicle, then it displays a message. Hence the person can choose another parking area which saves a lot of time.

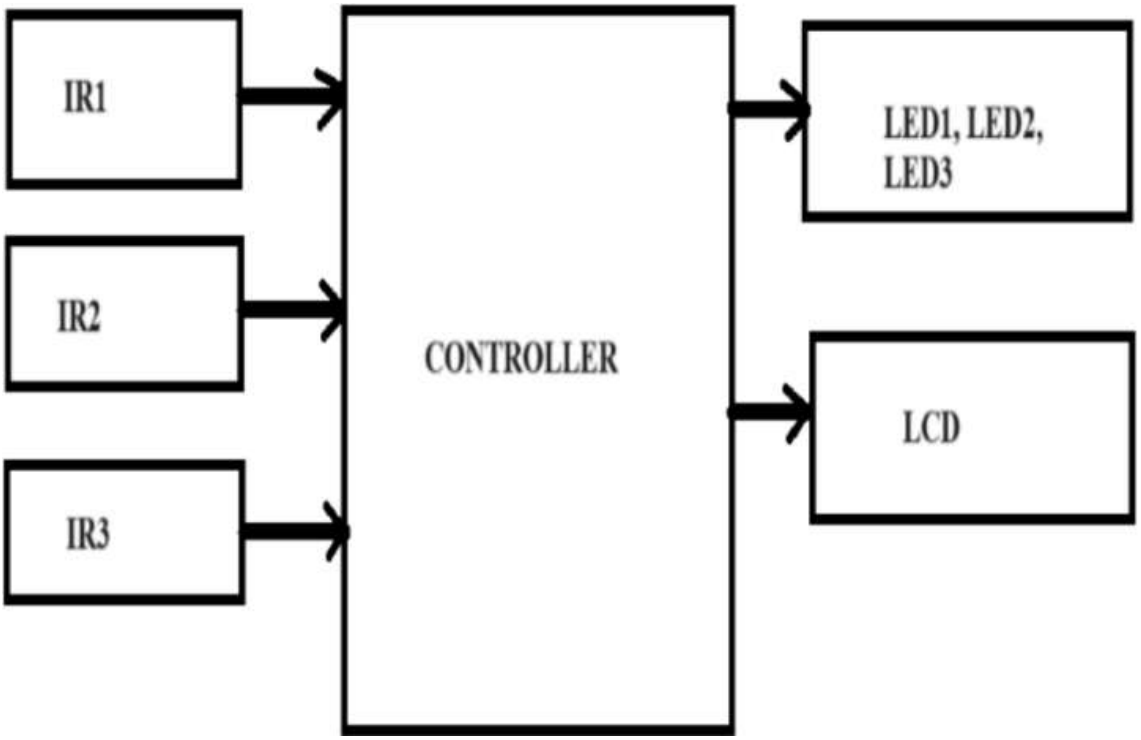



**Fig 3.2: Block diagram**

The basic concept is, when a car enters the car park through the entry point, if there is an empty parking slot, the barrier will allow access to the car park. One big display is situated after entry which shows the remaining parking slots. As soon as the car enters through entry, the sensor situated at entry gets activated and shows the empty parking slot. If there are no slots available for parking It shows the message that all slots are full and drivers have to wait for some time. The original design required the use of sensors mounted in the road. This system is usable for the parking slot seeker to know the exact empty parking slot.

The purpose of the Automatic Car Parking Indicator system is to guide the driver to a suitable parking space if one is available. The system would only allow entry to the car park, when there is space available. The system would also display the amount of space

available in the car park. The car park system will consist of a systematic entry and exit point, which would allow entry depending upon the availability of spaces. The design of the system would also involve the planning of a systematic lane system allowing the optimum amount of the parking slot to be allocated in order to avoid congestion as well as the difficulty in removing one's vehicle.

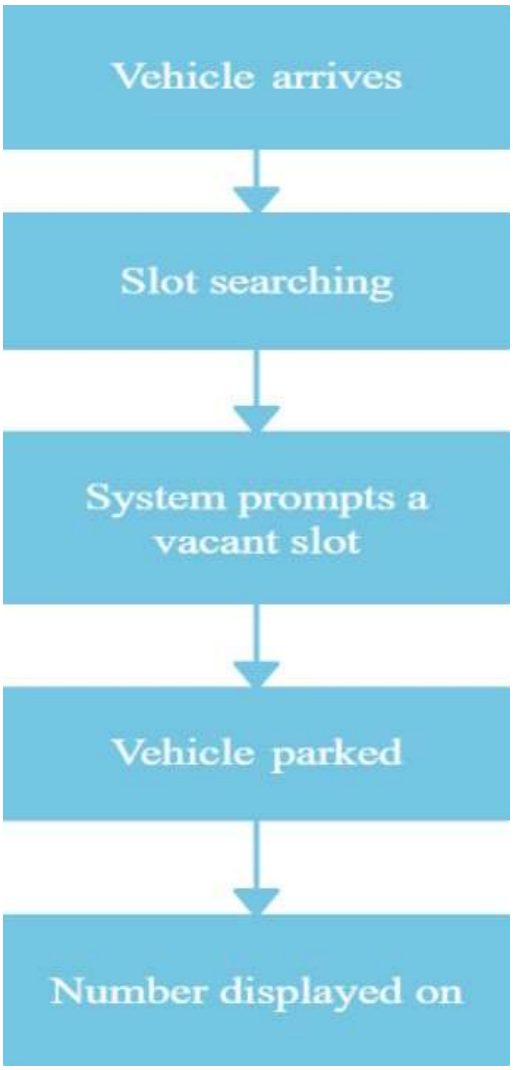



**Fig 3.3: Flowchart of the system**

The purpose of the Automatic Car Parking Indicator system is to guide the driver to a suitable parking space if one is available. The system would only allow entry to the car park, when there is space available. The system would also display the amount of space available in the car park. The car park system will consist of a systematic entry and exit point, which would allow entry depending upon the availability of spaces. The design of the system would also involve the planning of a systematic lane system allowing the optimum amount of the parking slot to be allocated in order to avoid congestion as well as

the difficulty in removing one's vehicle.

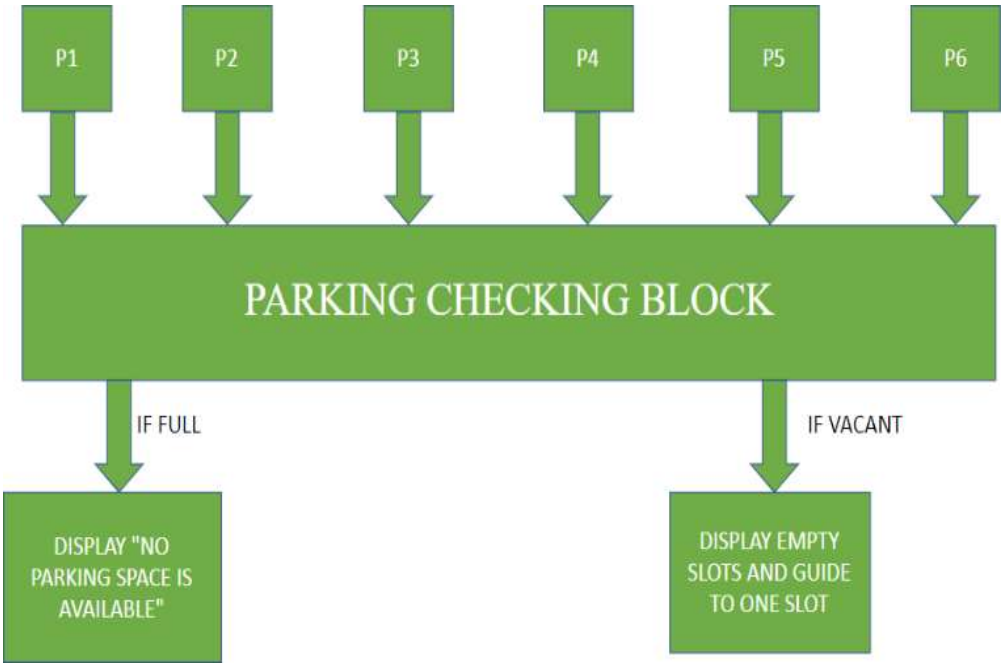



**Fig 3.4: Block diagram**

The above figure represents the block diagram of the proposed methodology. It consists mainly of Nine blocks where six of them are used to indicate the parking slots and other three are the conditional blocks used to check the availability of slots.

**Content related to block diagram:**

- P1, P2, P3, P4, P5, P6:
- It indicates the parking slots that are used in our system.

**Parking checking block:**

- In the parking check block, firstly an IR sensor is used to detect the vehicle and when it detects any vehicle the parking gate gets automatically opened and near the gate an LCD screen is placed to display the availability of free slots.

**Display "no parking space is available":**

- If all the parking slots are filled then a message is displayed on the LCD screen that "No Parking Space is Available". Hence it saves the time for the drivers to find other vacant parking slot without checking all the parking slots one by one.

**Display empty slots and guide to one slot:**

- If any parking slot is empty then a message is displayed on the LCD screen about the details of the vacancy slot. After displaying it also guides the driver with directions to the vacancy slot.
- When all the slots are empty then the system prompts the slot number to the users according to the ascending order i.e., in the order of P1, P2, P3, P4, P5, P6.
- For example, consider a situation where all the slots are empty. Whenever a vehicle enters into the parking lot then the system displays the vacant slot number as P1, even though the remaining slots are empty.

## 3.1 ADVANTAGES

- Space Efficiency.
- Enhanced Security.
- Privacy and Convenience
- Reduced Traffic Congestion.

# CHAPTER-4

# INTRODUCTION TO VLSI

Digital systems are highly complex at their most detailed level. They may consist of millions of elements i.e., transistors or logic gates. For many decades, logic schematics served as the Gur Franca of logic design, but not anymore. Today, hardware complexity has grown to such a degree that a schematic with logic gates is almost useless as it shows only a web of connectivity and not functionality of design. Since the 1970s, computer engineers, electrical engineers and electronics engineers have moved toward Hardware description language (HDLs).

Digital circuit has rapidly evolved over the last twenty five years. The earliest digital circuits were designed with vacuum tubes and transistors. Integrated circuits were then invented where logic gates were placed on a single chip. The first IC chip was small scale integration (SSI) chips where the gate count is small. When technology became sophisticated, designers were able to place circuits with hundreds of gates on a chip. These chips were called MSI chips with the advent of LSI; designers could put thousands of gates on a single chip. At this point, the design process is getting complicated and designers felt the need to automate these processes.

With the advent of VLSI technology, designers could design a single chip with more than a hundred thousand gates. Because of the complexity of these circuits computer aided techniques became critical for verification and for designing these digital circuits.

One way to lead with increasing complexity of electronic systems and the increasing time to market is to design at high levels of abstraction. Traditional paper and pencil and capture and simulate methods have largely given way to the described UN synthesized approach.

For these reasons, hardware description languages have played an important role in describe and synthesis design methodology. They are used for specification, simulation and synthesis of an electronic system. This helps to reduce the complexity in designing and products are made to be available in the market quickly.

The components of a digital system can be classified as being specific to an application or as being standard circuits. Standard components are taken from a set that has been used in other systems. MSI components are standard circuits and their use results in a significant reduction in the total cost as compared to the cost of using SSI Circuits. In

contrast, specific components are particular to the system being implemented and are not commonly found among the standard components.

The implementation of specific circuits with LSI chips can be done by means of IC that can be programmed to provide the required logic.

## 4.1 VLSI DESIGN FLOW

Typical design flow for designing VLSI circuits is shown in the tool flow diagram. This design flow is typically used by designers who use HDLs. In any design, specification is first. Specification describes the functionality, interface and overall architecture of the digital circuit to be designed. At this point, architects need not think about how they will implement their circuit. A behavioral description is then created to analyze the design in terms of functionality, performances and other high level issues. The behavioral description is manually converted to an RTL (Register Transfer Level) description in an HDL. The designer has to describe the data flow that will implement the desired digital circuit. From this point onward the design process is done with assistance of CAD tools.

Logic synthesis tools convert the RTL description to a gate level netlist. A gate level netlist is a description of the circuit in terms of gates and connections between them. The gate level netlist is input to an automatic place and route tool, which creates a layout. The layout is verified and then fabricated on a chip. Thus most digital design activity is concentrated on manually optimizing the RTL description of the circuit. After the RTL description is frozen, CAD tools are available to assist the designer in further process. Designing at RTL level has shrunk design cycle times from years to a few months.

### Emergence of hardware description language

As designs got larger and more complex, logic simulation assumed an important role in the design process. For a long time, programming languages such as fortran, pascal & c were used to describe the computer programs that were used to describe the computer programs that were sequential in nature. Similarly in the digital design field, designers felt the need for a standard language to describe digital circuits. Thus HDL came into existence. HDLs allowed the designers to model the concurrency of processes found in hardware elements. HDLs such as VERILOG HDL & VHDL (Very high speed integrated circuit hardware description language).

## 4.2 HISTORY OF VERILOG

Verilog was started in 1984 by Gateway Design Automation Inc as a proprietary hardware modeling language. It is rumored that the original language was designed by taking features from the most popular HDL language of the time, called HiLo, as well as from traditional computer languages such as C. At that time, Verilog was not standardized and the language modified itself in almost all the revisions that came out within 1984 to 1990.

Verilog simulator first used in 1985 and extended substantially through 1987.The implementation of Verilog simulator sold by Gateway. The first major extension of Verilog is Verilog-XL, which added a few features and implemented the infamous "XL algorithm" which is a very efficient method for doing gate-level simulation. Later 1990, Cadence Design System, whose primary product at that time included thin film process simulator, decided to acquire Gateway Automation System, along with other Gateway products., Cadence now become the owner of the Verilog language, and continued to market Verilog as both a language and a simulator. At the same time, Synopsys was marketing the top-down design methodology, using Verilog. This was a powerful combination.

In 1990, Cadence organized the Open Verilog International (OVI), and in 1991 gave it the documentation for the Verilog Hardware Description Language. This was the event which "opened" the language.

Later 1990, Cadence Design System, whose primary product at that time included thin film process simulator, decided to acquire Gateway Automation System, along with other Gateway products., Cadence now become the owner of the Verilog language, and continued to market Verilog as both a language and a simulator. At the same time, Synopsys was marketing the top-down design methodology, using Verilog. This was a powerful combination.

In 1990, Cadence organized the Open Verilog International (OVI), and in 1991 gave it the documentation for the Verilog Hardware Description Language. This was the event which "opened" the language.

## 4.3 BASIC CONCEPTS

Hardware Description Language

Two things distinguish an HDL from a linear language like “C”: Concurrency:

- The ability to do several things simultaneously i.e. different code-blocks can run

concurrently.

Timing:

- Ability to represent the passing of time and sequence events accordingly.

## Verilog introduction

- Verilog HDL is a Hardware Description Language (HDL).
- A Hardware Description Language is a language used to describe a digital system; one may describe a digital system at several levels.
- An HDL might describe the layout of the wires, resistors and transistors on an Integrated Circuit (IC) chip, i.e., the switch level.
- It might describe the logical gates and flip flops in a digital system, i.e., the gate level.
- An even higher level describes the registers and the transfers of vectors of information between registers. This is called the Register Transfer Level (RTL).
- Verilog supports all of these levels.
- A powerful feature of the Verilog HDL is that you can use the same language for describing, testing and debugging your system.

## Verilog features

- **Strong Background**: Supported by open verilog international and Institute of Electrical and Electronics Engineering standardized.
- **Industrial support**: Simulation is very fast and synthesis is very efficient.
- **Universal**: Entire process is allowed in one design environment.
- **Extensibility**: It also allows Verilog PLI for extension of Verilog capabilities

## 4.4 DESIGN FLOW

The typical design flow is shown in figure,

## Design Specification

- The project Specifications and requirements are written first
- The digital circuit functionality is explained for the architecture to be designed.
- Specification: It uses wave former, test bencher or word for drawing waveform.

## RTL Description

- CAD Tools are used for coding format for the Conversation of Specification.

## Coding Styles:

- Gate Level Modeling
- Data Flow Modeling
- Behavioral Modeling

## Functional Verification Testing

- The method of coding with respective inputs and outputs are going to be tested.
- Check the RTL Description once again if testing fails.
- **Simulation**: Using Xilinx , Verilog-XL ,ModelSim

## Logic Synthesis

- RTL description into Gate level -Net list form conservation.
- The circuit is described as a function of gates and connections.
- **Synthesis**: Synthesis is done by Altera and Xilinx ,Simplify Pro, Leonardo Spectrum Design Compiler, FPGA Compiler.

## Logical Verification and Testing

- Simulation and synthesis are used for functional Checking of HDL coding. Check the RTL description if it fails.

Floor Planning Automatic Place and Route

- Layout is created with the respective gate level Net list.
- The blocks of the net list are arranged on the chip.

- Place & Route: Implement FPGA vendors P&R tool for FPGA. Very costly P&R tools like Apollo required for ASIC tools.

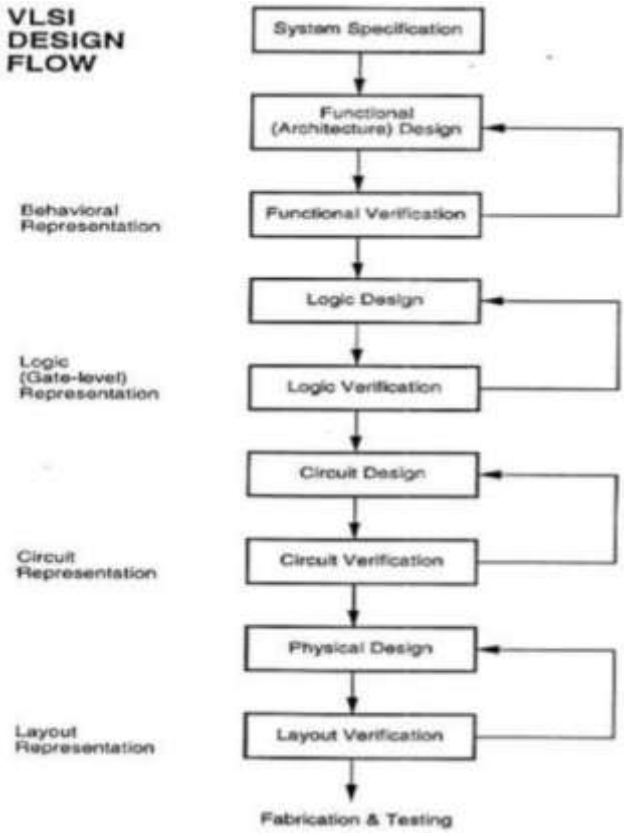



**Fig 4.1: Vlsi Design flow**

## Logical Verification and Testing

- The process of describing a circuit description into the physical layout is called the Physical design, it explains the interconnections between the cells and routes position.

## Layout Verification

- Under layout verification first the physical layout structure has to be verified.
- Floor Planning Automatic Place and Route and RTL Description can be done for any modifications.

## Implementation

- The design process is the final stage in implementation.
- coding and RTL can be Implemented using Integrated circuits.

## 4.5 MODULES

Distinct parts of a Verilog module consists of are as shown in below figure. The keyword module is the beginning of a module definition. In a module definition the module name, port list, port declarations, and optional parameters must come first in its definition. If the module has any ports to interact with the external environment then only Port list and port declarations are present. There are five components within a module.

- Variable declarations
- Dataflow statements
- Behavioral blocks
- Tasks or functions.
- Instantiation of lower modules

components may appear in any order and at any place in a given module definition.The end module statement must always come last in a module definition. All components except module, module name, and end module are optional and can be mixed and matched as per design needs. Multiple module definitions in a single file are allowed by Verilog. In the file the modules can be defined in any order.

Example Module Structure:

Module <module name>(<module_terminals_list>);

….. <module internals> ….

Endmodule

Components may appear in any order and at any place in a given module definition.The end module statement must always come last in a module definition. All components except module, module name, and end module are optional and can be mixed and matched as per design needs. Multiple module definitions in a single file are allowed by Verilog.

In the file the modules can be defined in any order.

Example Module Structure:

Module <module name>(<module_terminals_list>);

….. <module internals> ….

Endmodule

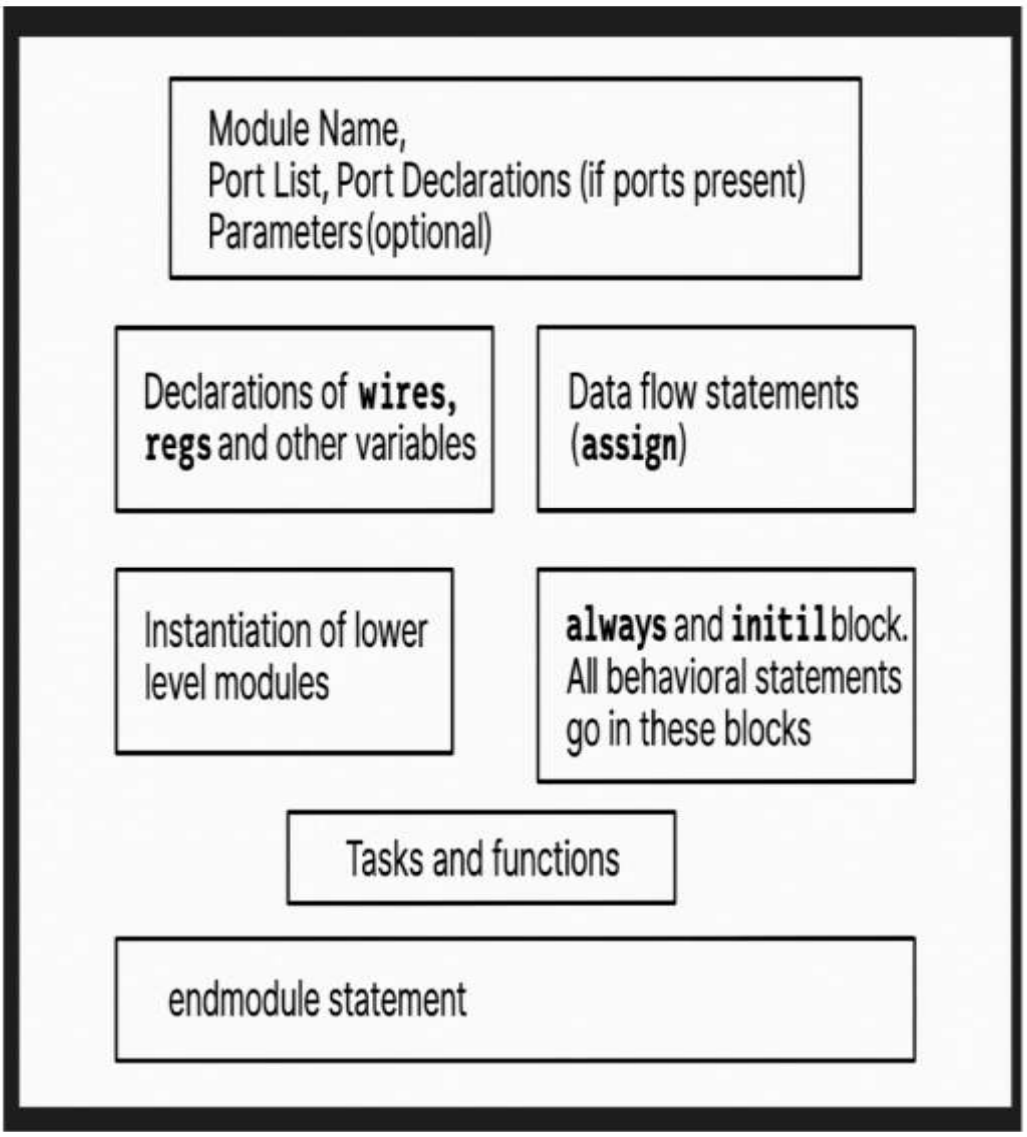



**Fig 4.2: Module Design**

## 4.1 PORTS

The interface between a module and its environment is provided by the ports. The input/output pins of an Integrated Circuit chip are its ports. The environment cannot see the internals of the module. It is a great advantage for the designer. As long as the interface is not modified the internals of the module may be mod without affecting its environment. Terminals are the synonyms for the ports.

### 4.1.1 Port Declaration

The Arduino module may contain the declaration of all ports in the given list of ports . The declaration of ports are explained in detail below.

### 4.1.2 Port Declaration

Input:- Input port

Output:- Output port

inout :-Bidirectional port depending on the direction of the port signal, each port in the port list is given a label as follows input, output, or inout,

### 4.1.3 Port Connection Rules

A port consisting of two units, primary unit is into the module and secondary unit is out of the module. The primary and secondary units are connected. When modules are instantiated within other modules there are rules governing port connections within the module. If any port connection rules are violated then the Verilog simulator complains. The figure 5.6 shows the port connection rules.

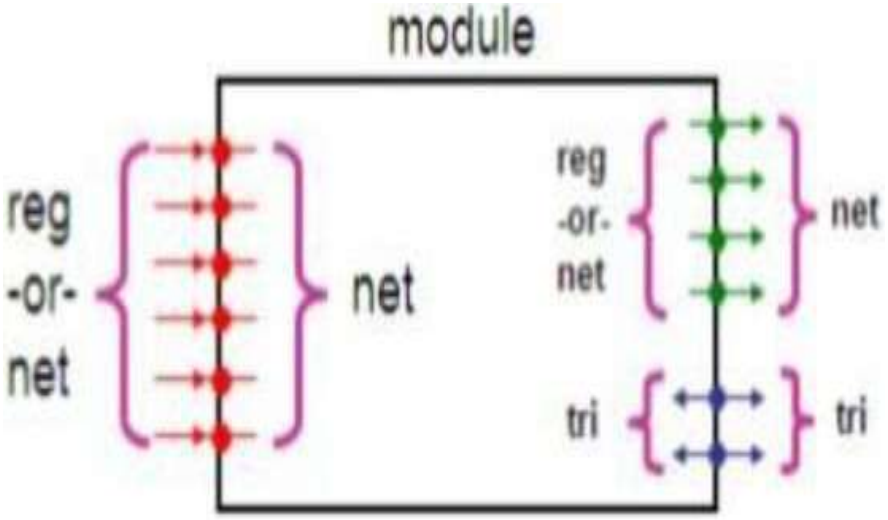



**Fig 4.3 : Module connections**

**Inputs**:

- Internally must be of net data type (e.g. wire)
- Externally the inputs may be connected to a reg or net data type.

**Outputs**:

- Internally may be of net or reg data type

**In outs:**

- Internally must be of net data type (tri recommended)
- Externally must be connected to a net data type (tri recommended) Ports Connection to External Signals. Signals and Ports in a module can be connected in two ways. In the module definition those two methods cannot be mixed.
- Port by order list
- Port by name

**Port by order list:**

Most spontaneous method for learners is the Connecting port by order list. The order in which the ports in the ports list in the module definition must be connected in the same order.

Syntax for instantiation with port order list:

module name instance name (signal, signal...);

External signals a, b, out appear in exactly the same order as the ports a, b, out in the module defined in adder in the below example.

Example

```
module adder( a,b,out);

input[1:0]a;
input[1:0]b;
output[1:0]out;
wire [1:0]out;

assign out=a+b;

endmodule
```

```
module top_example;

reg [1:0]a;
reg [1:0]b;
wire [1:0]out;

adder ex1(a,b,out);

endmodule
```

**Fig 4.4: Module I/O pins**

**Port by name**

For larger designs where the module have say 30 ports ,it is almost impractical and there is a possibility of errors if remembering the order of the ports in the module definition. There is capability to connect external signals to ports by the port names, rather than by position provided by the Verilog. Syntax for instantiation with port name:

Module name instance name (.port name(signal), .port name (signal)… );

The port connects in any order as long as the port name in the module definition correctly matches the external signal.

# CHAPTER -5

# WORKING AND DESIGN SIMULATION

## 5.1 WORKING

A Car Parking Management System using Verilog is an efficient digital system designed to automate vehicle parking operations. The system ensures optimal space utilization, monitors vehicle movements, and provides real-time parking slot availability using hardware sensors, state machine logic, storage elements, and display mechanisms. This document provides a detailed breakdown of how the system works, covering its functional components, input processing, digital logic, state transitions, slot allocation mechanisms, and user interactions.

The system is composed of multiple digital components working together to manage parking in an automated manner. The key components include Parking Slot Sensors, which detect whether a parking space is occupied or vacant, a Control Unit (FPGA or Microcontroller) that executes the Verilog program and processes parking slot information, Storage Elements (Registers & Flip-Flops) that store real-time parking slot availability data, Multiplexers & Decoders that control selection and display of slot information, a 7-Segment Display or LED Indicators for visually displaying available and occupied slots, Entry and Exit Gates to monitor vehicle movements, and a Communication Module for remote parking monitoring if needed.

The functional workflow begins when a vehicle approaches the parking entrance. The system detects an incoming car using IR sensors and checks for the nearest available parking slot. The assigned slot is then displayed to the driver. Once the car is parked, the slot status updates to ‘Occupied’. When the car exits, the slot is marked as ‘Vacant’, and the display refreshes in real-time. The system receives input signals from various sources, including Infrared (IR) or Ultrasonic Sensors that generate digital signals (1 for occupied, 0 for vacant) based on car detection, Entry & Exit Gates that update parking status when a vehicle enters or leaves, and Control Inputs (Admin Panel) that allow manual updates for slot reservations.

The Car Parking Management System uses a Finite State Machine (FSM) to manage parking slot allocation and updates. The FSM operates through several states, including the Idle State where all parking slots are vacant and waiting for input, the Car Detection State

when the entry sensor detects a vehicle, the Slot Assignment State where the system finds the nearest available slot, the Slot Occupied State when the car is parked and the slot status is updated, the Car Exit State when the exit sensor detects a vehicle departure, and finally the Update and Display State where the system refreshes the slot status and displays updated availability.

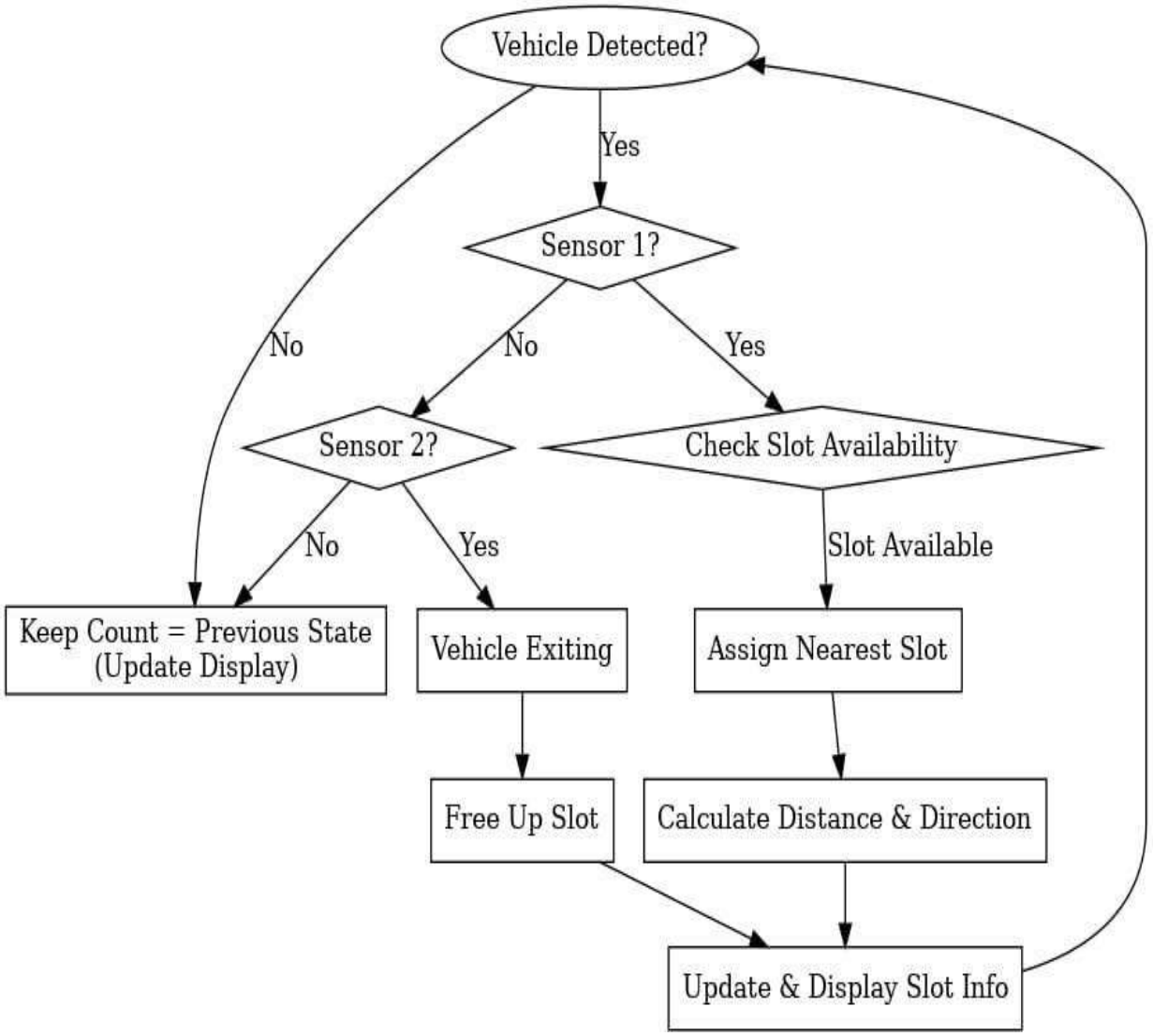



**Fig 5.1: Module Design**

**Verilog Code Implementation**

```
module parking_system(
   input clk, reset,
   input [3:0] sensor,  // 4 sensors detecting parking slots
   output reg [3:0] slot_status  // 1 = Occupied, 0 = Vacant
```

);

- clk – System clock signal for synchronization.
- reset – Resets all slots to vacant state.
- sensor – Takes input from parking slot sensors.
- slot_status – Stores slot occupancy status.

**FSM Logic for Slot Allocation**

```
always @(posedge clk or posedge reset) begin
    if (reset)
        slot_status <= 4'b0000;  // Reset all slots to vacant
    else
        slot_status <= sensor;  // Update slot status dynamically
end
```

The FSM updates the slot occupancy based on real-time sensor inputs. If reset is triggered, all slots become vacant (0000).

**Displaying Available Slots**

```
assign available_slots = ~slot_status;  // Invert status for display purposes
```

Vacant slots are displayed using a simple bitwise NOT operation.

**Assigning the Nearest Available Slot**

```
always @(posedge clk) begin
    case (available_slots)
        4'b0001: assigned_slot = 1;
        4'b0010: assigned_slot = 2;
        4'b0100: assigned_slot = 3;
        4'b1000: assigned_slot = 4;
        default: assigned_slot = 0;  // No slots available
```

```
    endcase
end
```

This logic finds the nearest vacant parking slot for an incoming vehicle.

**Handling Car Exit**

When a car exits, the system updates the parking slot availability.

**Data Storage and Processing**

The data storage and processing mechanisms involve Registers & Flip-Flops to store slot occupancy status dynamically, Shift Registers to manage sequential updates for slot allocation, and Memory Blocks to store historical parking data if required.

**Display System and User Interface**

1. Display System and User Interface
2. Green Light = Vacant Slot
3. Red Light = Occupied Slot
4. Digital Display shows the nearest available slot
5. Remote monitoring can be implemented via a web interface or mobile app for live slot tracking.

**Real-Time Parking Slot Updates**

Real-time parking slot updates occur whenever a vehicle enters or exits, ensuring immediate slot status changes. The FSM dynamically updates the display and processes slot availability efficiently.

1. Error Handling and Security Features
2. Redundant Sensor Validation to avoid false detections.
3. Access Control for the Admin Panel to prevent unauthorized slot assignment.
4. Power Backup to ensure uninterrupted operation.

**Advantages of the System**

1. Automated and Real-Time Updates eliminate manual management.
2. Efficient Space Utilization optimizes slot assignment.

3. User-Friendly Interface ensures ease of parking.
4. Scalable System that can be expanded for larger parking lots.

**Future Enhancements**

Future enhancements could include an Automated Payment System that charges based on parking duration, AI-Powered Slot Prediction to optimize parking recommendations, and IoT Integration for remote parking space monitoring.

### 5.1.1 Xilinx ise:

Xilinx Integrated Software Environment (ISE) is a comprehensive software suite developed by Xilinx for the design, synthesis, simulation, and implementation of digital circuits on Field-Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs). It is widely used by engineers, researchers, and students for developing hardware-based applications using Hardware Description Languages (HDLs) such as Verilog and VHDL. Xilinx ISE provides an intuitive project management interface that simplifies the design process, making it easier to develop complex digital circuits. It integrates multiple tools that assist in different stages of FPGA design, including synthesis, simulation, debugging, and implementation.One of the key features of Xilinx ISE is its Project Navigator, which allows users to organize project files, define target FPGA devices, and access various tools needed for design implementation. The synthesis process, carried out by Xilinx Synthesis Technology (XST), converts HDL descriptions into gate-level representations, optimizing the design for FPGA implementation. The Implementation phase involves translating, mapping, placing, and routing the synthesized logic onto the physical FPGA resources, ensuring efficient utilization of the available logic blocks, memory, and I/O pins. Additionally, Constraints Management enables designers to specify timing, area, and pin assignments, ensuring that the design meets performance requirements.

Create a New Project Create a new ISE project which will target the FPGA device on the Spartan-3 Starter Kit demo board. To create a new project:

1. Select File > New Project... The New Project Wizard appears.

2. Type tutorial in the Project Name field.

3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.

4. Verify that HDL is selected from the Top-Level Source Type list.

5. Click Next to move to the device properties page. 6. Fill in the properties in the table as shown below:

- **Product Category**: All
- **Family**: Spartan3
- **Device**: XC3S200
- **Package**: FT256
- **Speed Grade**: -4
- **Top-Level Source Type**: HDL
- **Synthesis Tool**: XST (VHDL/Verilog) Simulator: ISE Simulator (VHDL/Verilog)
- **Preferred Language**: Verilog (or VHDL)
- Verify that Enable Enhanced Design Summary is selected.

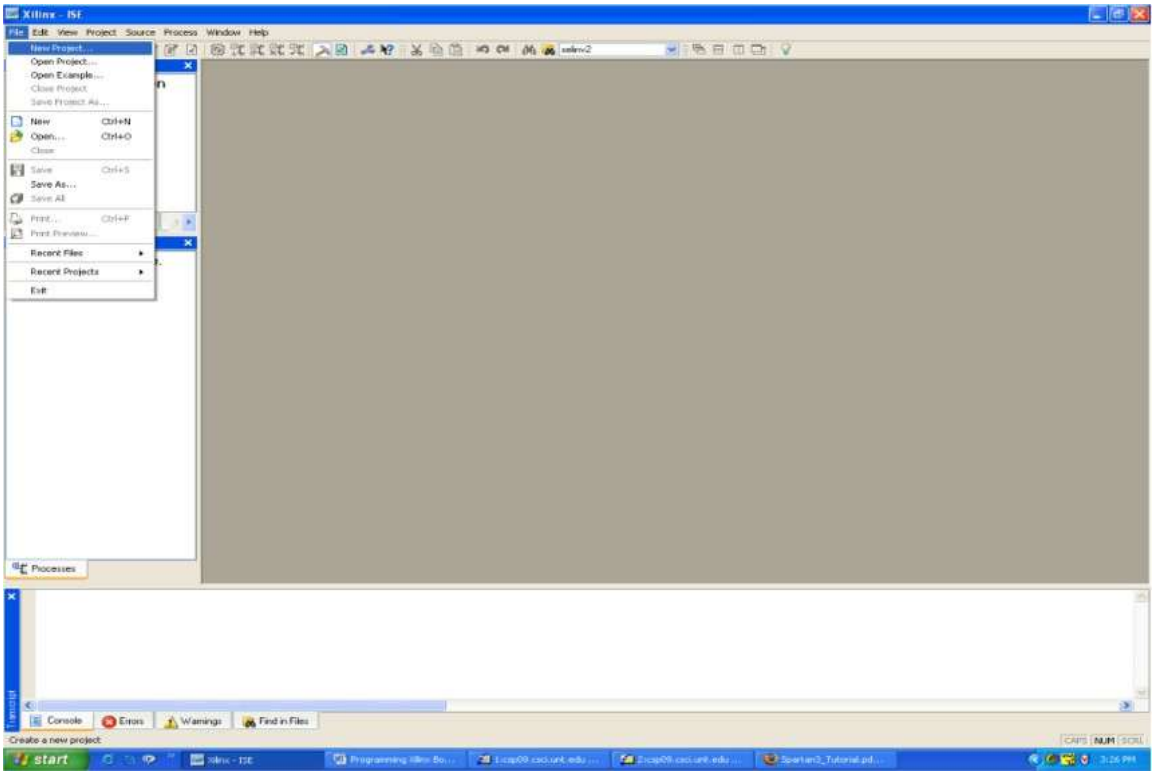
Leave the default values in the remaining fields. When the table is complete, your project properties will look like the following:

### 5.1.2 Creating a new project and source

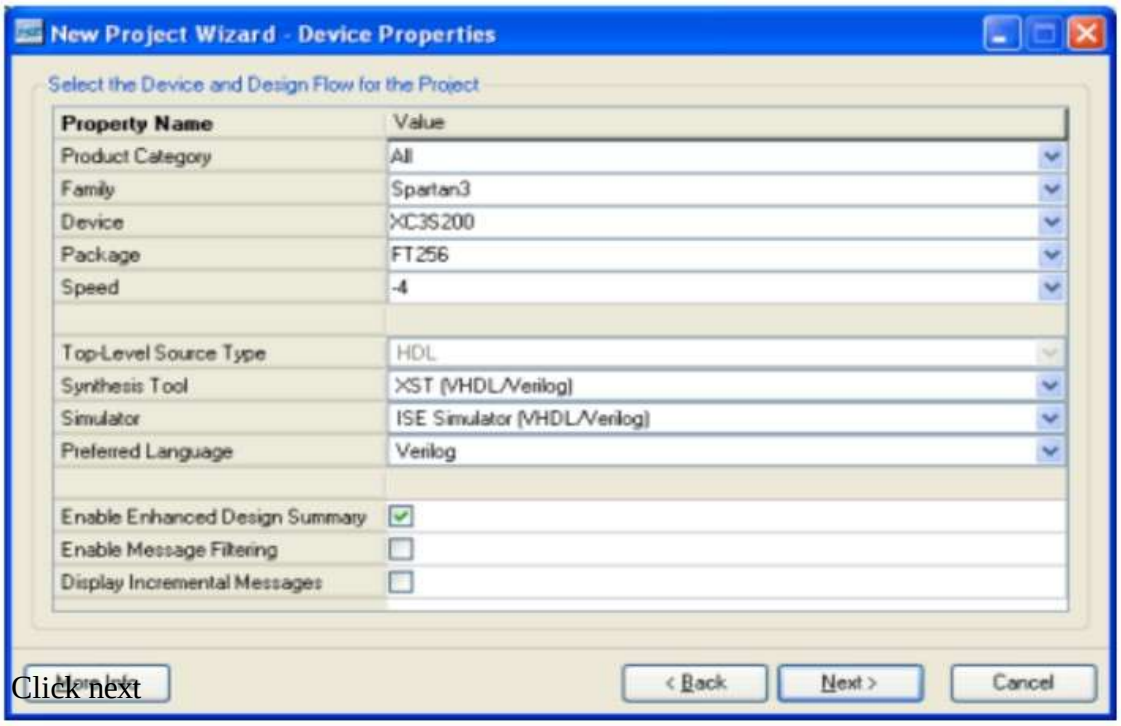
Start the Xilinx ISE 8.1i project navigator by double clicking the Xilinx ISE 8.1i icon on your desktop. To create a new project in Xilinx Vivado, start by launching the software and selecting the option to create a new project. Assign a project name and choose a storage location. Next, select RTL Project as the project type and proceed to add source files. You can either import existing Verilog or VHDL files or create new ones directly within Vivado. If required, constraint files can also be added to define pin assignments. After selecting the target FPGA device, review the project settings and finalize the setup

Click on File and select New Project Once the project is created, source code can be written for the required functionality. For example, a simple LED blinking module can be implemented in Verilog using a counter-driven toggle mechanism. The code typically includes input ports for the clock and reset signals, a register to maintain the counter, and an output signal that toggles at a predefined rate. After writing the code, synthesis is performed to check for design errors, followed by implementation, which maps the logic to the FPGA. The next step involves generating the bitstream file, which is then used to

program the FPGA. This process ensures the design is successfully uploaded and tested on the hardware.

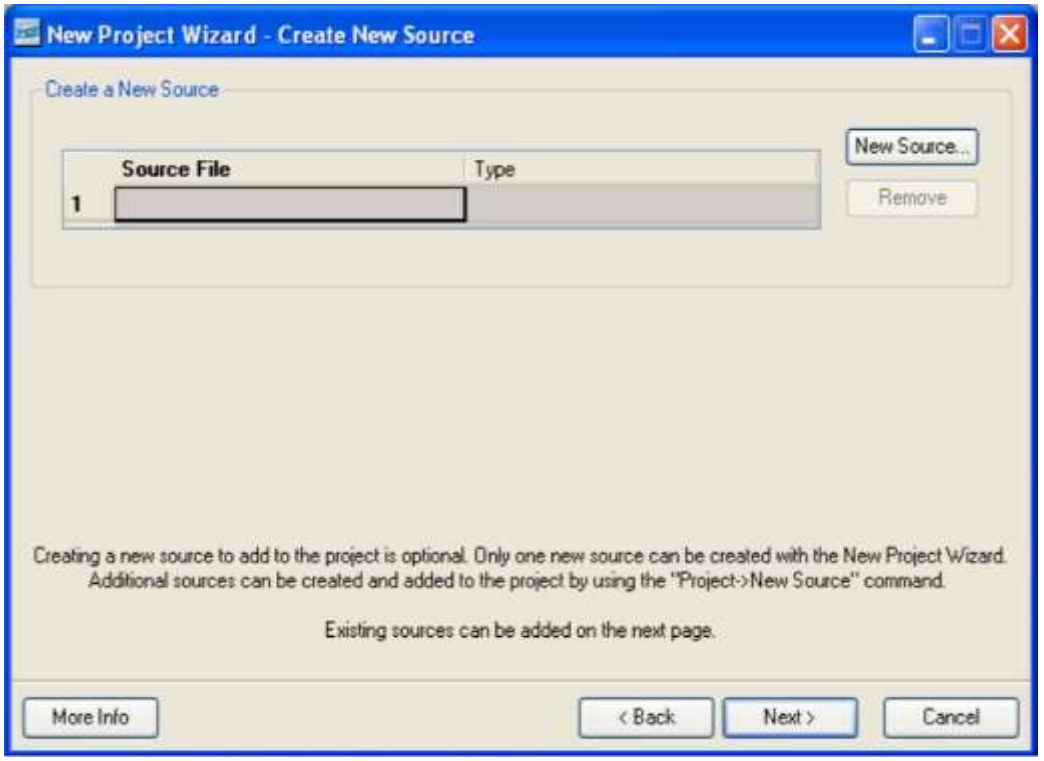


Select a project location and type the name you would like to call your project counter:



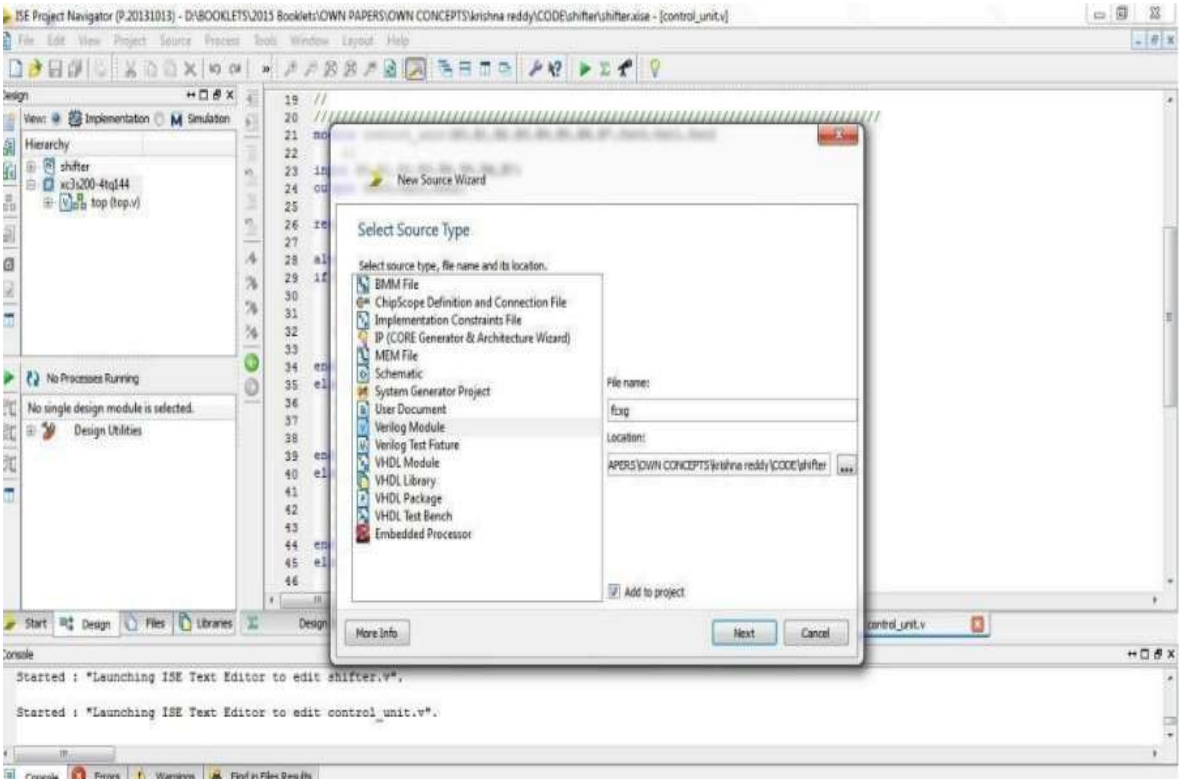

Click next

Click New Source



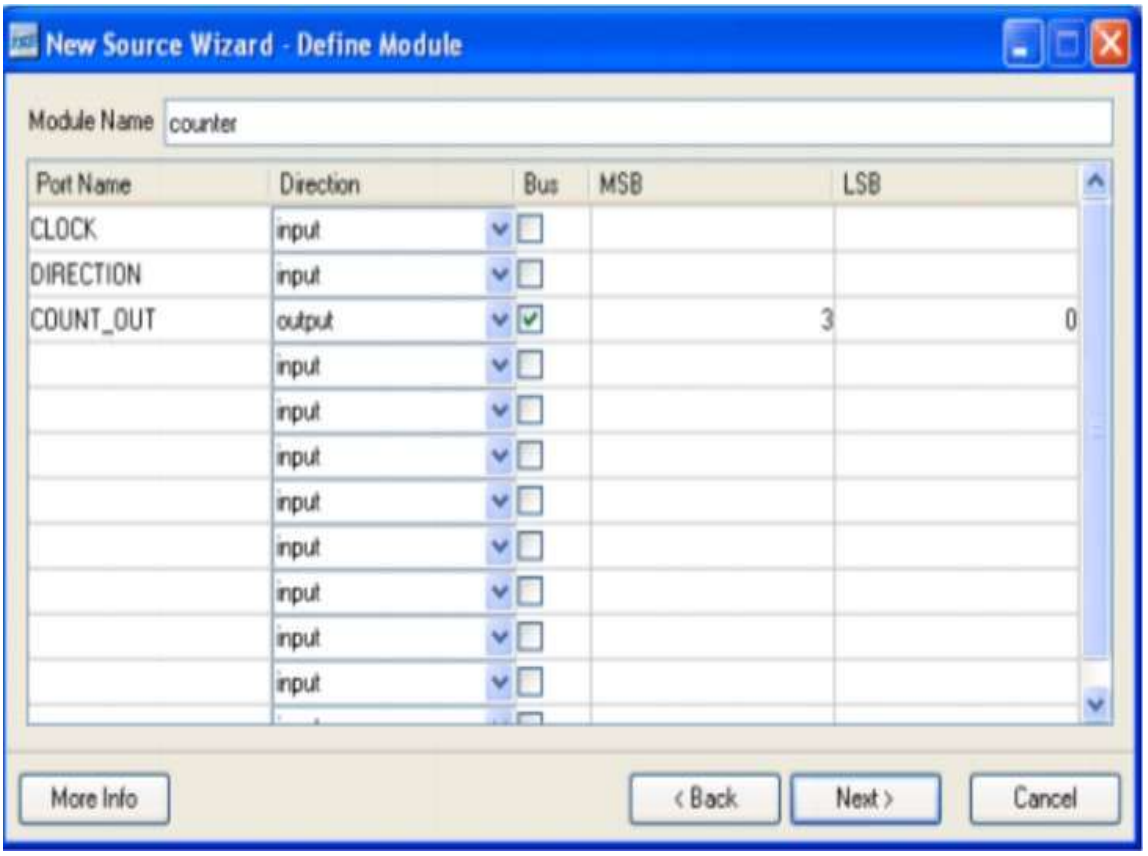

Click next

**Select Verilog Module in the New Source Wizard window:**



**Creating a Verilog Source:**

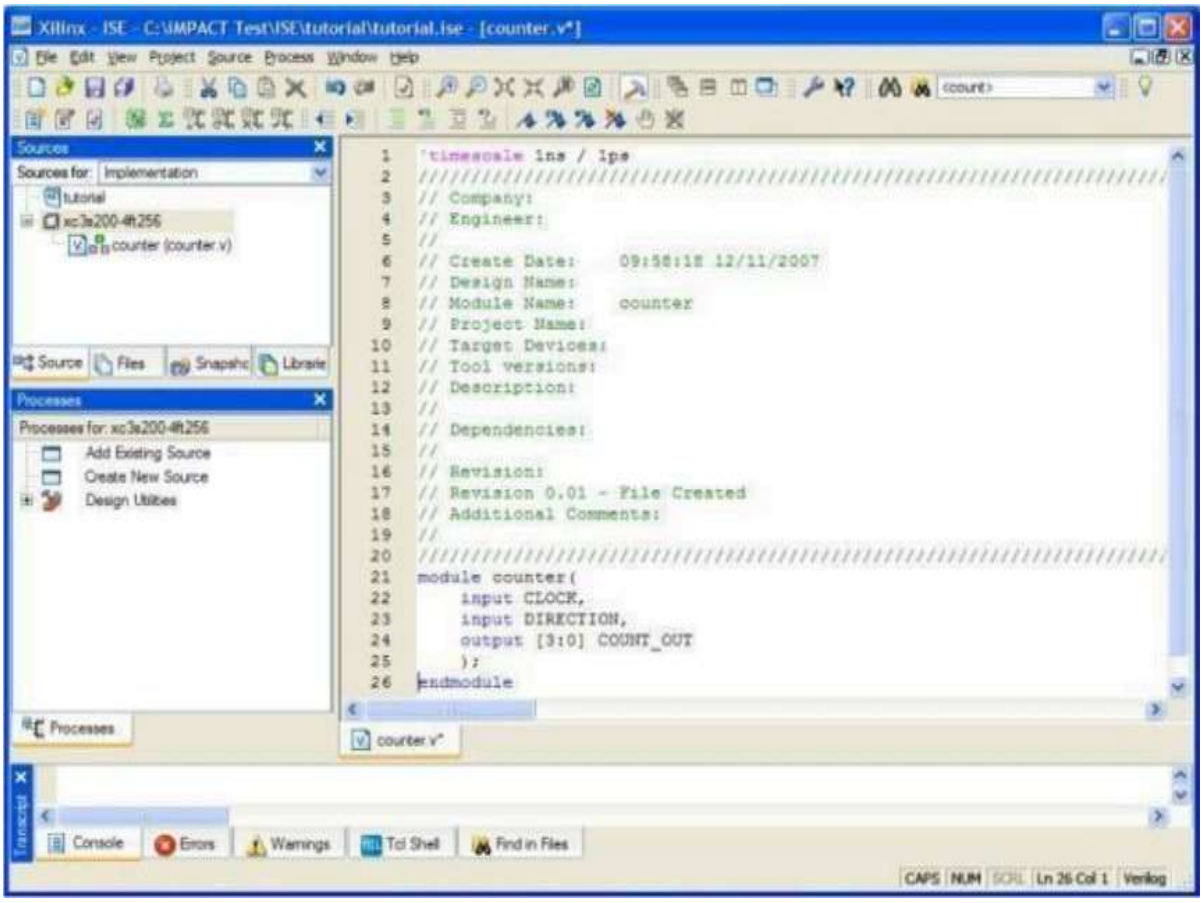
1. Create the top-level Verilog source file for the project as follows:
2. Click New Source in the New Project dialog box.
3. Select Verilog Module as the source type in the New Source dialog box.
4. Type in the file name counter.
5. Verify that the Add to Project checkbox is selected.
6. Click Next.
7. Declare the ports for the counter design by filling in the port information as shown below:




The source file containing the counter module displays in the Workspace

Xilinx ISE provides a graphical user interface (GUI) for FPGA design and development. The top menu bar includes options such as File, Edit, View, Project, Source, Process, and Window, which allow users to perform various design tasks. Below this menu, a toolbar provides quick access to common functions such as opening, saving, compiling, synthesizing, simulating, and implementing FPGA designs.On the left panel, the "Sources"

window organizes project files hierarchically.The right panel displays the Verilog code editor.Where users write and edit their HDL code. The Verilog module shown in the image is named counter.




Xilinx ISE provides a graphical user interface (GUI) for FPGA design and development. The top menu bar includes options such as File, Edit, View, Project, Source, Process, and Window, which allow users to perform various design tasks. Below this menu, a toolbar provides quick access to common functions such as opening, saving, compiling, synthesizing, simulating, and implementing FPGA designs.On the left panel, the "Sources" window organizes project files hierarchically.The right panel displays the Verilog code editor.Where users write and edit their HDL code. The Verilog module shown in the image is named counter.

1. Select File > New Project... The New Project Wizard appears.

2. Type tutorial in the Project Name field.

3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.

4. Verify that HDL is selected from the Top-Level Source Type list.

**Using Language Templates (Verilog):**

The next step in creating the new source is to add the behavioral description for counter. Use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. Place the cursor on the line below the output [3:0] COUNT_OUT; statement.
2. Open the Language Templates by selecting Edit → Language Templates… Note:You can tile the Language Templates and the counter file by selecting Window → Tile Vertically to make them both visible.
3. Using the "+" symbol, browse to the following code example: Verilog → Synthesis Constructs → Coding Examples → Counters → Binary → Up/Down Counters → Simple Counter.
4. With Simple Counter selected, select Edit → Use in File, or select the Use Template in File toolbar button. This step copies the template into the counter source file.
5. Close the Language Templates

## 5.2 DESIGN SIMULATION:

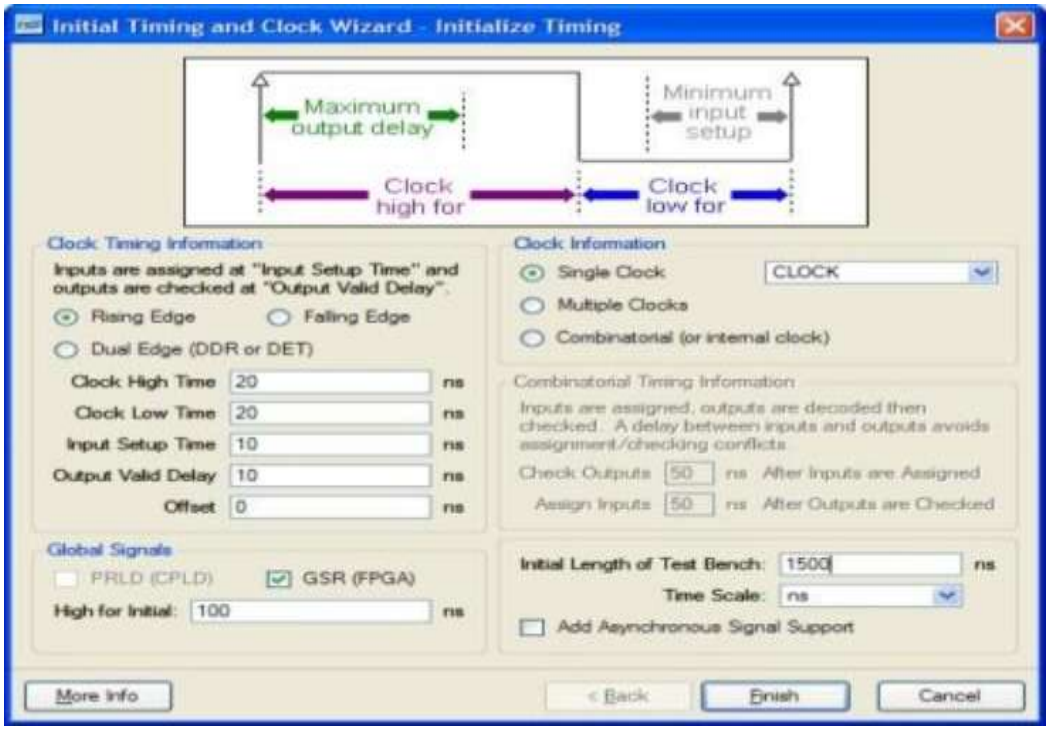
Verifying Functionality using Behavioral Simulation Create a test bench waveform containing input stimulus you can use to verify the functionality of the counter module. The test bench waveform is a graphical view of a test bench. Create the test bench waveform as follows:

1. Select the counter HDL file in the Sources window.
2. Create a new test bench source by selecting Project → New Source.
3. In the New Source Wizard, select Test Bench WaveForm as the source type, and type counter_tbw in the File Name field.
4. Click Next.
5. The Associated Source page shows that you are associating the test bench waveform with the source file counter. Click Next.

6. The Summary page shows that the source will be added to the project, and it displays the source directory, type, and name. Click Finish.

7. You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens. The requirements for this design are the following:

- The counter must operate correctly with an input clock frequency = 25 MHz.
- The DIRECTION input will be valid 10 ns before the rising edge of CLOCK
- The output (COUNT_OUT) must be valid 10 ns after the rising edge of CLOCK. The design requirements correspond with the values below. Fill in the fields in the Initialize Timing dialog box with the following information:
- Clock High Time: 20 ns.
- Clock Low Time: 20 ns.
- Output Valid Delay: 10 ns.
- Offset: 0 ns.
- Global Signals: GSR (FPGA) Note: When GSR(FPGA) is enabled, 100 ns. is added to the Offset value automatically.
- Initial Length of Test Bench: 1500 ns.



Click Finish to complete the timing initialization. 9. The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the counter design as follows:

- Click on the blue cell at approximately the 300 ns to assert DIRECTION high so that the counter will count up.
- Click on the blue cell at approximately the 900 ns to assert DIRECTION low so that the counter will count down.

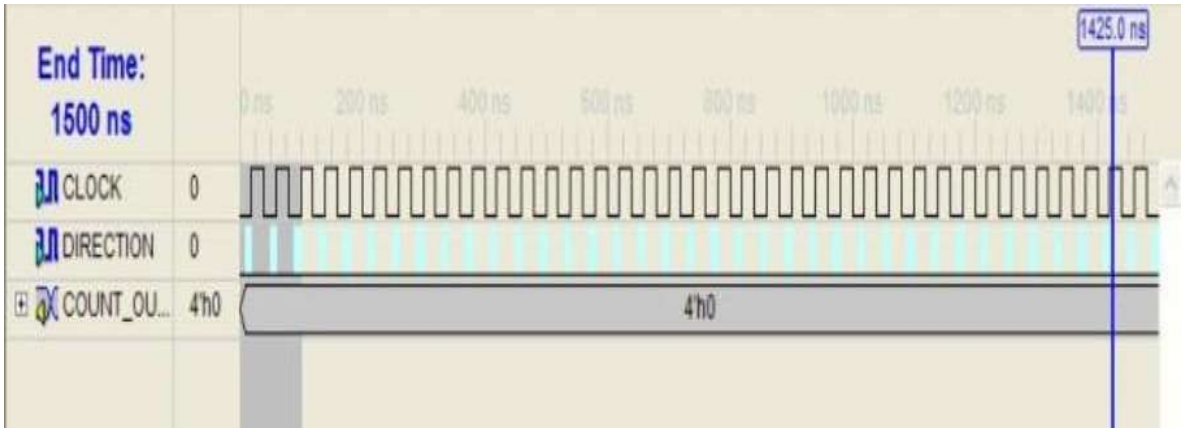



**Fig 5.4: 16x2 LCD (Liquid Crystal Display)**

**Introduction**:

A 16x2 LCD is a commonly used alphanumeric display module that can show up to 16 characters per line on 2 lines, making it ideal for various embedded system applications, microcontroller projects, and display interfaces. A 16×2 LCD display works by controlling the liquid crystals to either block or allow light to pass through, creating characters and symbols on the screen. It's controlled by sending data and commands to its controller, which in turn manages the display of information. A standard 16×2 LCD display has 16 pins, typically organized into two rows of eight pins each. These pins are used for power supply, data communication, and control signals. The contrast control adjusts the contrast between the text and the background on the LCD screen. By changing the voltage across the liquid crystals, you can control the readability of the displayed content. A 16×2 LCD display can display a wide range of information, including text, numbers, symbols, and basic graphics. It's often used

to display status information, menu options, sensor readings, and more. The backlight is a built-in light source that illuminates the LCD screen, making the displayed information visible in low-light conditions. It can be controlled to adjust the brightness of the display.

1. **Specifications**:

- **Display Type**: Alphanumeric LCD
- **Display Size**: 16 characters x 2 lines
- **Operating Voltage**: 4.7V - 5.3V
- **Backlight**: LED (optional)
- **Controller IC**: HD44780
- **Interface**: Parallel (4-bit or 8-bit)
- **Character Size**: 5x8 pixel matrix

## 5.3 FEATURES OF 16X2 LCD:

- **Character Display**: Can display 16 characters per row, with two rows.
- **Character Size**: Each character consists of a 5x8 pixel matrix. Controller IC: Uses Hitachi HD44780 or equivalent controller. Operating Voltage: Typically 5V DC, but some versions operate at 3.3V.
- **Backlight**: Some models have an LED backlight for visibility in low-light conditions.
- **Communication Interface**: Can operate in 4-bit or 8-bit mode with a microcontroller.
- **Custom Characters:** Supports the creation of custom characters using CGRAM (Character Generator RAM).

## 5.4 WORKING MODES:

The LCD can work in two different modes, namely the 4-bit mode and the 8-bit mode. In 4 bit mode we send the data nibble by nibble, first upper nibble and then lower nibble. For those of you who don’t know what a nibble is: a nibble is a group of four bits, so the lower four bits (D0-D3) of a byte form the lower nibble while the upper four bits (D4-D7) of a byte form the higher nibble. This enables us to send 8 bit data.

Whereas in 8 bit mode we can send the 8-bit data directly in one stroke since we use all the 8 data lines.

Now you must have guessed it, Yes 8-bit mode is faster and flawless than 4-bit mode. But the major drawback is that it needs 8 data lines connected to the microcontroller. This will make us run out of I/O pins on our MCU, so 4-bit mode is widely used. No control pins are used to set these modes. It's just the way of programming that change. While 16×2 LCD displays are versatile, they have limitations such as limited screen size and resolution, and they might not be suitable for displaying complex graphics or large amounts of information.

## 5.5 PIN DESCRIPTION OF 16X2 LCD:

| Pin No. | Pin Name | Function |
|---|---|---|
| 1 | VSS | Ground (0V) |
| 2 | VCC | Power Supply (5V) |
| 3 | VEE | Contrast Control (0-5V) |
| 4 | RS | Register Select (0: Command, 1: Data) |
| 5 | RW | Read/Write (0: Write, 1: Read) |
| 6 | E (Enable) | Enables LCD (High to Low Pulse) |
| 7-14 | D0 - D7 | Data Pins (Used for 8-bit mode) |
| 15 | LED+ (A) | LED Backlight Power (5V) |
| 16 | LED- (K) | LED Backlight Ground (0V) |

## 5.6 HOW LCD DISPLAYS CHARACTERS:

The LCD controller processes ASCII codes and displays them as characters.

It uses an internal CGROM (Character Generator ROM) for predefined characters. Custom characters can be stored in CGRAM.

**LCD Commands & Functions**

| Command (Hex) | Function |
|---|---|
| 0x01 | Clear Display |
| 0x02 | Return Home |
| 0x04 | Decrement Cursor |
| 0x06 | Increment Cursor |
| 0x08 | Display OFF |
| 0x0C | Display ON, Cursor OFF |
| 0x0E | Display ON, Cursor ON |
| 0x0F | Display ON, Cursor Blinking |
| 0x10 | Shift Cursor Left |
| 0x14 | Shift Cursor Right |
| 0x18 | Shift Display Left |
| 0x1C | Shift Display Right |
| 0x80 | Force Cursor to 1st Line |
| 0xC0 | Force Cursor to 2nd Line |

## 5.7 APPLICATIONS OF 16X2 LCD:

1. Embedded Systems
2. Microcontroller
3. Projects Industrial
4. Displays Smart Home Applications
5. Weather Stations
6. IoT Dashboards

## 5.8 ADVANTAGES & DISADVANTAGES OF 16X2 :

**LCD Advantages:**

1. Low power consumption
2. Easy to interface with microcontrollers
3. Available with backlight for better visibility

**LCD Disadvantages:**

1. Cannot display complex images
2. Limited number of characters

## 5.9 INFRARED IR SENSORS

**Introduction**

Infrared (IR) sensors are electronic devices that detect infrared radiation to sense objects, measure distances, or transmit data wirelessly. These sensors play a crucial role in automation, robotics, security systems, and communication technologies. This document provides an in-depth analysis of IR sensors, their working principles, types, applications, advantages, and interfacing methods.

**Working principle**

1. IR sensors operate based on infrared radiation, which is invisible to the human eye. They consist of two main components:
2. IR Emitter (Transmitter): Emits infrared light.
3. IR Receiver (Detector): Detects reflected infrared light from objects.
4. When an object is placed in front of an IR sensor, the emitted IR light reflects off the surface and is detected by the receiver. The amount of reflected light determines the object's presence and distance.

**Types of IR Sensors**

1. Active IR Sensors: Have both emitter and receiver**.**
2. Used for object detection and proximity sensing. Example: IR proximity sensors.
3. Passive IR (PIR) Sensors: Detect infrared radiation emitted by objects.

4. Used in motion detection systems. Example: PIR motion sensors.
5. Reflective IR Sensors: Emit and receive IR light within the same module.
6. Used for line-following robots.
7. Through-Beam IR Sensors: Separate emitter and receiver placed opposite each other.Used in security barriers.

**IR Sensor Circuit and Components**

1. IR LED: Emits infrared light. Photodiode: Detects infrared light.
2. Resistors and Capacitors: Used for voltage regulation.
3. Operational Amplifier (Comparator): Converts light intensity into a digital signal. Microcontroller Interface: Processes the received signal.

**IR Sensor Modules**

1. TSOP1738: Used for remote control systems. GP2Y0A02YK0F: Used for distance measurement.
2. HC-SR501 PIR Sensor: Used in motion detection applications.
3. GP2Y0A02YK0F: Used for distance measurement.

**Applications of IR Sensors**

1. **Security Systems:** Motion detection in surveillance.
2. **Robotics:** Line-following and obstacle-avoiding robots.
3. **Healthcare:** IR thermometers for temperature measurement.
4. **Automotive**: Automatic braking and collision avoidance systems.
5. **Industrial Automation**: Object counting and sorting.

**Advantages of IR Sensors**

1. Contactless sensing.
2. Low power consumption
3. High reliability in different environments.
4. Compact and cost-effective.

### Disadvantages of IR Sensors

1. Limited range compared to ultrasonic sensors.
2. Affected by ambient light conditions.
3. Difficulty in detecting black surfaces.

### Troubleshooting and Common Issues

1. **False Triggering**: Caused by ambient light interference. Low Sensitivity: Adjust resistor values to improve response. Power Issues: Ensure correct voltage supply.
2. **Interference from Other IR Devices**: Use modulated IR signals for improved accuracy.
3. Future Developments in IR Sensors AI-powered IR sensors for smart automation. Improved sensitivity and miniaturization.
4. Integration with IoT for enhanced applications.

## 5.10 DC MOTORS

### Introduction to DC Motors

A DC (Direct Current) motor is an electrical machine that converts direct current electrical energy into mechanical energy. It is widely used in various industrial and household applications due to its simplicity, efficiency, and controllability. A DC (Direct Current) motor is an electromechanical device that converts electrical energy into mechanical energy through the interaction of magnetic fields. It operates on the principle of electromagnetic induction, where a current-carrying conductor placed in a magnetic field experiences a force that generates motion. DC motors are widely used in various applications, including robotics, industrial automation, electric vehicles, and household appliances, due to their simple operation, precise speed control, and high efficiency. A DC motor consists of a stator (stationary part), which provides a constant magnetic field, and a rotor (moving part), which carries the armature windings. When a direct current flows through the windings, it creates a magnetic field that interacts with the stator's field, producing torque and causing the rotor to rotate. The commutator and brushes play a crucial role in maintaining continuous rotation by reversing the current direction in the armature windings.

**Basic Principles of DC Motors**

A DC motor operates on the principle of electromagnetism. When a current flows through the winding of the motor, a magnetic field is generated, which interacts with the external magnetic field to produce motion. The fundamental working principle is based on Lorentz Force Law.

**Components of a DC Motor**

1. **Stator**: The stationary part of the motor that provides the magnetic field.
2. **Rotor** (**Armature**): The rotating part that carries the current.
3. **Commutator**: A mechanical rectifier that switches the current direction.
4. **Brushes**: Conductive contacts that deliver current to the commutator.
5. **Field Windings**: Coils that generate the magnetic field.

**TYPES OF DC MOTORS**

**Brushed DC Motors**

1. Series Wound Motor Shunt Wound Motor
2. Compound Wound Motor
3. Brushless DC Motors (BLDC)

**Working of DC Motors**

When the DC voltage is applied to the motor terminals, current flows through the armature winding, generating a magnetic field. This field interacts with the stator's field, causing the armature to rotate. The commutator and brushes ensure the current direction is maintained to sustain continuous motion.

**Speed control of dc motors**

**Speed can be controlled by:**

- Varying the supply voltage.
- Adjusting the field current.
- Using Pulse Width Modulation (PWM) techniques.

**Applications of DC Motors**

- **Industrial Machinery**: Used in conveyors, cranes, and robotics.
- **Electric Vehicles**: Provide smooth torque and speed control.
- **Household Appliances**: Found in fans, mixers, and vacuum cleaners.
- **Aerospace & Defense**: Used in actuators and guidance systems.

**Advantages and disadvantages of dc motors**

**Advantages:**

- Simple design
- High starting torque
- Precise speed control

**Disadvantages:**

- Require frequent maintenance due to brushes.
- Limited efficiency at high speeds.

**Future Trends in DC Motor Technology**

- Improved brushless motor designs
- Enhanced energy efficiency
- Smart control systems with IoT integration.

## 5.11 BUZZER

**Introduction**

A buzzer is an electronic component that produces sound when an electrical signal is applied to it. Buzzers are widely used in alarm systems, notifications, timers, and indication devices.A buzzer is an electronic or electromechanical device that produces sound when an electrical signal is applied. It is widely used in various applications such as alarm systems, timers, and electronic circuits to provide auditory feedback. The working principle of a buzzer involves converting electrical energy into sound waves through mechanical vibration. When an electric current flows through the buzzer, its diaphragm or piezoelectric material vibrates, generating an audible tone. There are different types of

buzzers, including electromechanical, piezoelectric, and magnetic buzzers, each designed for specific applications. Electromechanical buzzers use a coil and a vibrating diaphragm, while piezoelectric buzzers rely on a piezoelectric crystal that expands and contracts to produce sound. Magnetic buzzers operate using an electromagnetic coil that moves a diaphragm to generate noise. Buzzers are commonly found in alarms, security systems, home appliances, automobiles, and medical devices. They serve as essential components in fire alarms, doorbells, seat belt reminders, microwave ovens, and heart rate monitors. Due to their compact size, energy efficiency, and reliability, buzzers have become a crucial part of modern electronic systems, providing an effective means of alerting users in various fields.

**Types of Buzzers**

There are two main types of buzzers:

**Active Buzzer:**

- Generates sound when a voltage is applied (built-in oscillating circuit).
- Operates with DC voltage (e.g., 3V, 5V, or 12V).
- Simple to use with microcontrollers like Arduino, Raspberry Pi.
- Produces a fixed sound frequency.

Example: Used in alarms and notifications.

**Passive Buzzer:**

- Needs an external oscillating circuit to produce sound.
- Can generate different tones by varying the frequency of the signal.
- Operates with PWM (Pulse Width Modulation) signals from a microcontroller.
- Example: Used in musical applications and customizable alerts.

**Working Principle**

- A buzzer converts electrical energy into sound energy using piezoelectric or electromagnetic principles.

**Piezoelectric Buzzer**

- Uses a piezoelectric material that vibrates when an AC voltage is applied.

- Produces sound waves in the audible range (1kHz – 5kHz).
- Low power consumption, commonly used in alarms and watches.

**Electromagnetic Buzzer**

- Uses a coil and magnet to create vibrations in a diaphragm.
- Can produce louder sounds compared to piezo buzzers.
- Used in industrial alerts, car horns, and sirens.

**Applications**

- **Alarms & Security Systems** – Fire alarms, burglar alarms, and smoke detectors.
- **Notification Systems** – Doorbells, timers, and medical alert devices.
- **Industrial Warning Systems** – Machinery alerts and safety indicators.
- **Electronic Toys & Games** – Sound effects and interactive systems.
- **Automotive Applications** – Reverse parking sensors and seat belt reminders.

**Advantages**

- Low power consumption
- Compact and lightweight
- Easy to interface with microcontrollers
- Reliable and durable

**Disadvantages**

- Limited sound range in simple buzzers
- Fixed frequency in active buzzers
- Can be annoying if not properly controlled

# CHAPTER – 6

# RESULTS

The designed Car Parking Management System has been implemented using Verilog and simulated in Xilinx Vivado. The system effectively monitors available parking slots, detects vehicle entry and exit, and updates the count accordingly. The simulation results validate the correctness of the system's operation. The simulation of the Car Parking Management System using Verilog was successfully implemented and tested using Xilinx Vivado and ModelSim. The project aimed to design an efficient parking system that detects vehicle entry and exit, updates slot availability in real-time, and guides drivers to the nearest available parking space. The results of the simulation confirm the functionality and accuracy of the design. The system was implemented using Finite State Machines (FSMs), counters, and multiplexers to control the parking slots dynamically. The Verilog modules were simulated to verify their correctness in detecting vehicle movements and updating slot availability. The system accurately detects vehicle entry and exit using proximity sensors simulated in Verilog. The 7-segment display/LED matrix dynamically updates to indicate which slots are occupied and which are available. The occupancy status of each slot is represented using a binary encoding scheme, where 1 indicates an occupied slot and 0 indicates a vacant slot. The distance from the entry gate to each available parking slot is calculated using a counter-based logic, and the system selects the nearest available slot based on predefined priority logic, ensuring efficient parking allocation. The real-time update of slot availability helps in reducing congestion at the parking entrance. The FSM transitions smoothly between different states, ensuring there are no glitches in the system's operation. The system correctly handles scenarios such as multiple car entries, simultaneous exits, and full parking conditions. In case of a full parking lot, an alert message is displayed, preventing additional vehicles from entering. The design was analyzed for timing constraints, logic utilization, and power consumption. The system operates within the required clock cycles, ensuring real-time updates. The Verilog code is optimized to reduce the number of gates required, making it suitable for FPGA implementation. The simulation confirms minimal power consumption, making the design feasible for real-world applications. The Verilog-based Car Parking Management System successfully detects, monitors, and updates parking slot availability in real-time. The simulation results validate the system's accuracy and reliability, demonstrating its potential for practical FPGA-based implementation. Future enhancements

can include automatic payment systems, integration with IoT for remote monitoring, and

machine learning-based prediction models for better space utilization.

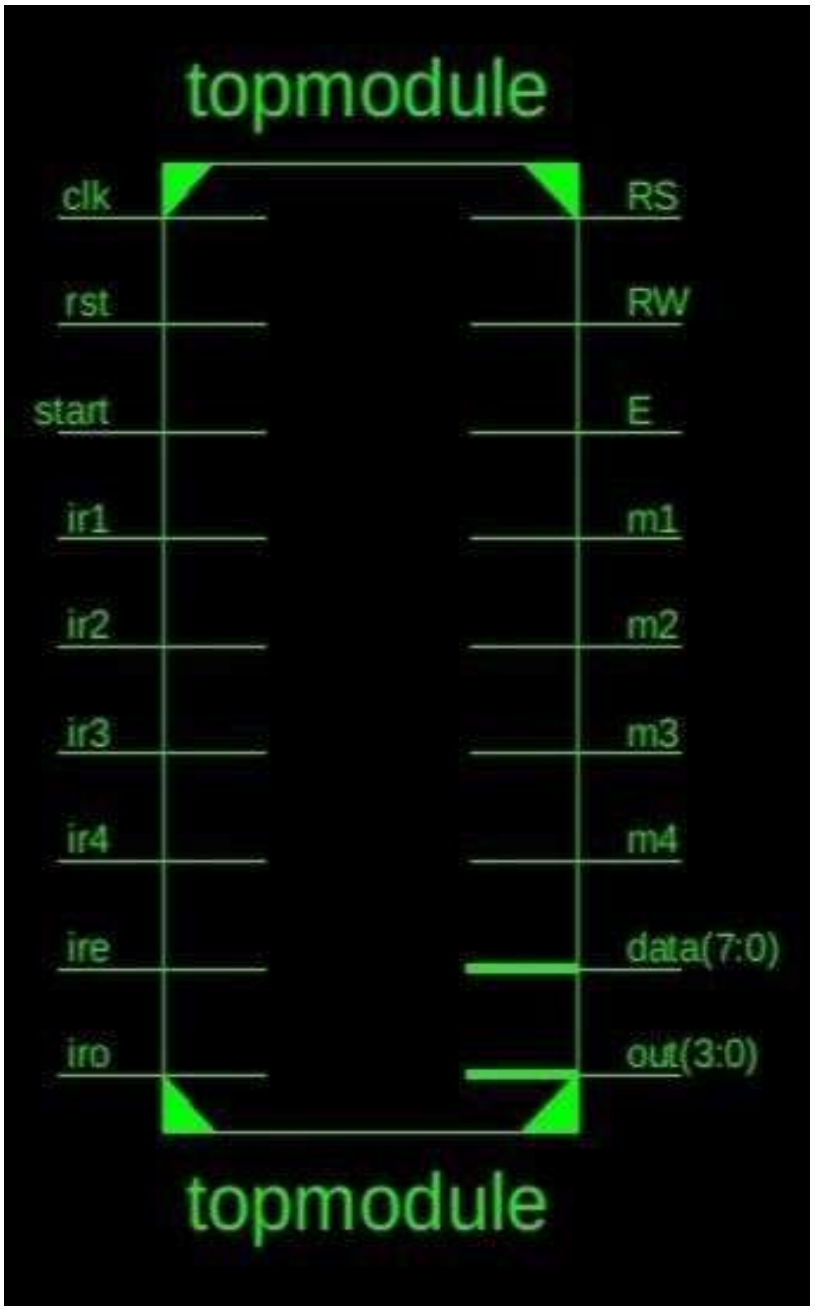



**Fig 6.1: RTL schematic.**

The RTL schematic of the top module in Xilinx ISE Project Navigator, which is a part of the Car Parking Management System implemented in Verilog. The hierarchy panel on the left displays the project structure, where the top-level module, topmodel, integrates submodules such as g1 (car parking logic), g2 (gate logic), and l1 (LCD control). The RTL schematic in the center illustrates the logical connections of the top module, showing various input and output ports. Key inputs include clk (clock), rst (reset), start, and multiple IR sensor signals (ir1 to ir), likely used for vehicle detection. On the output side, signals such as RS, RW, and E are possibly associated with LCD control, while m1 to m4

may correspond to motor or parking slot indicators. Additionally, the data(7:0) output suggests an 8-bit data bus, likely for external communication, and out(3:0) represents a 4-bit output, possibly indicating parking status. The Processes panel on the bottom left highlights the View RTL Schematic step, confirming that the synthesis process has been completed successfully, allowing for verification of the logical structure. This visualization helps ensure that the design is correctly implemented before proceeding to further simulation or hardware implementation

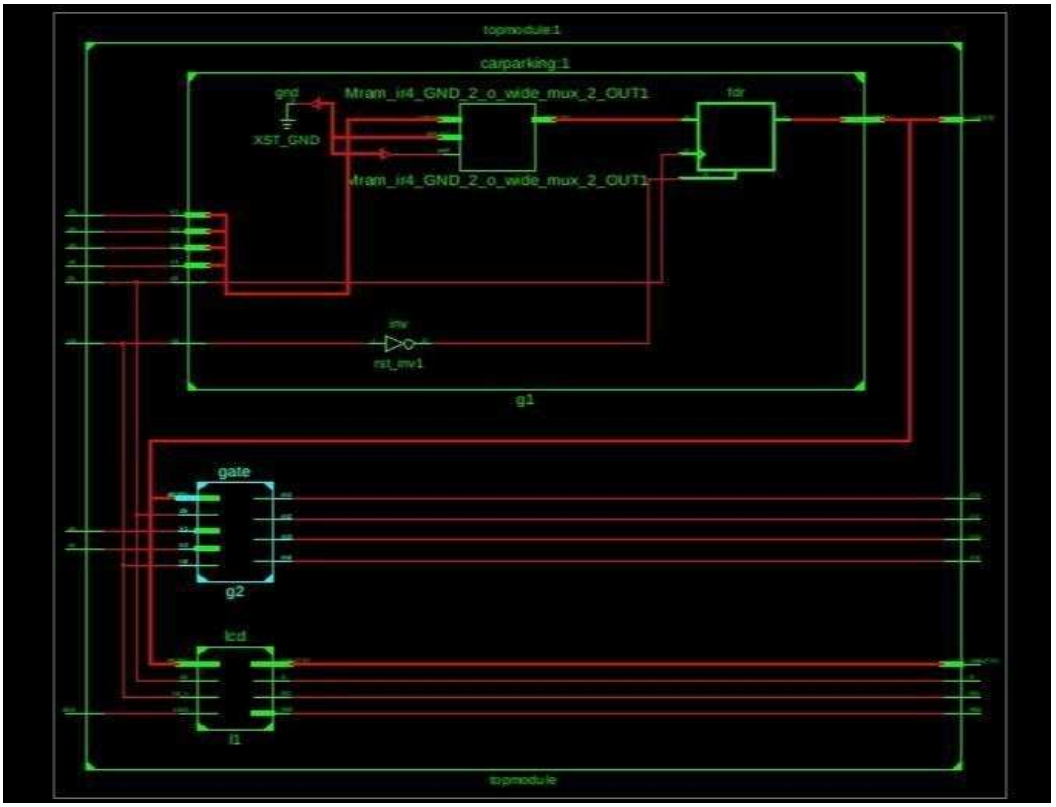



**Fig 6.2:Technology Schematic of the Car Parking**

The Technology Schematic of the Car Parking Management System, generated using Xilinx ISE after the synthesis process. This schematic provides a gate-level representation of the design, showing how different components such as multiplexers (MUX), flip-flops (FDR), logic gates, and interconnections are structured within the hardware. The top module serves as the main module and integrates three major submodules:

- **g1 (Car Parking Logic)** – Handles car detection, slot allocation, and system control logic.
- **g2 (Gate Control Logic)** – Manages the opening and closing of the parking gates based on the status of parking slots and sensor inputs.
- **l1 (LCD Display Control)** – Interfaces with an LCD screen to display the number of

available parking slots and other relevant information.

The schematic illustrates various interconnections between these modules, represented by red and green signal paths. The red lines indicate data signals, which transfer information between different logic components, while the green lines typically represent control signals or power connections. The MUX (multiplexer) components allow for signal selection based on system conditions, while the FDR (flip-flop) components are used for sequential logic operations, ensuring proper timing and synchronization of the circuit.

Additionally, the presence of reset and clock signals (such as `rst_inv1`) suggests that the system relies on synchronous operation, meaning all operations are triggered by a central clock pulse to maintain stability and avoid conflicts in data processing. The schematic also includes basic logic elements like inverters, gates, and memory registers, which help in processing inputs from IR sensors and user commands.

This schematic representation is useful for hardware verification, allowing designers to ensure that the synthesized circuit correctly implements the intended Verilog design. By analyzing this layout, engineers can debug errors, optimize circuit efficiency, and confirm functional correctness before proceeding with simulation and FPGA implementation. The technology schematic is particularly important when working with FPGA-based hardware design, as it provides insight into how the high-level Verilog code is mapped onto actual logic gates and components.

Car Parking Management System, generated using Xilinx ISE after the synthesis process. This schematic provides a gate-level representation of the design, showing how different components such as multiplexers (MUX), flip-flops (FDR), logic gates, and interconnections are structured within the hardware. The top module serves as the main module and integrates three major submodules: the presence of reset and clock signals (such as `rst_inv1`) suggests that the system relies on synchronous operation, meaning all operations are triggered by a central clock pulse to maintain stability and avoid conflicts in data processing. The schematic also includes basic logic elements like inverters, gates, and memory registers, which help in processing inputs from IR sensors and user commands.

The simulation waveform of the Car Parking Management System, generated using ISim (Integrated Simulator) in Xilinx ISE. This waveform provides a timing-based visualization of the system's behavior, showcasing how different signals interact over time.

The x-axis represents simulation time (in nanoseconds), while the y-axis lists various input and output signals of the design.

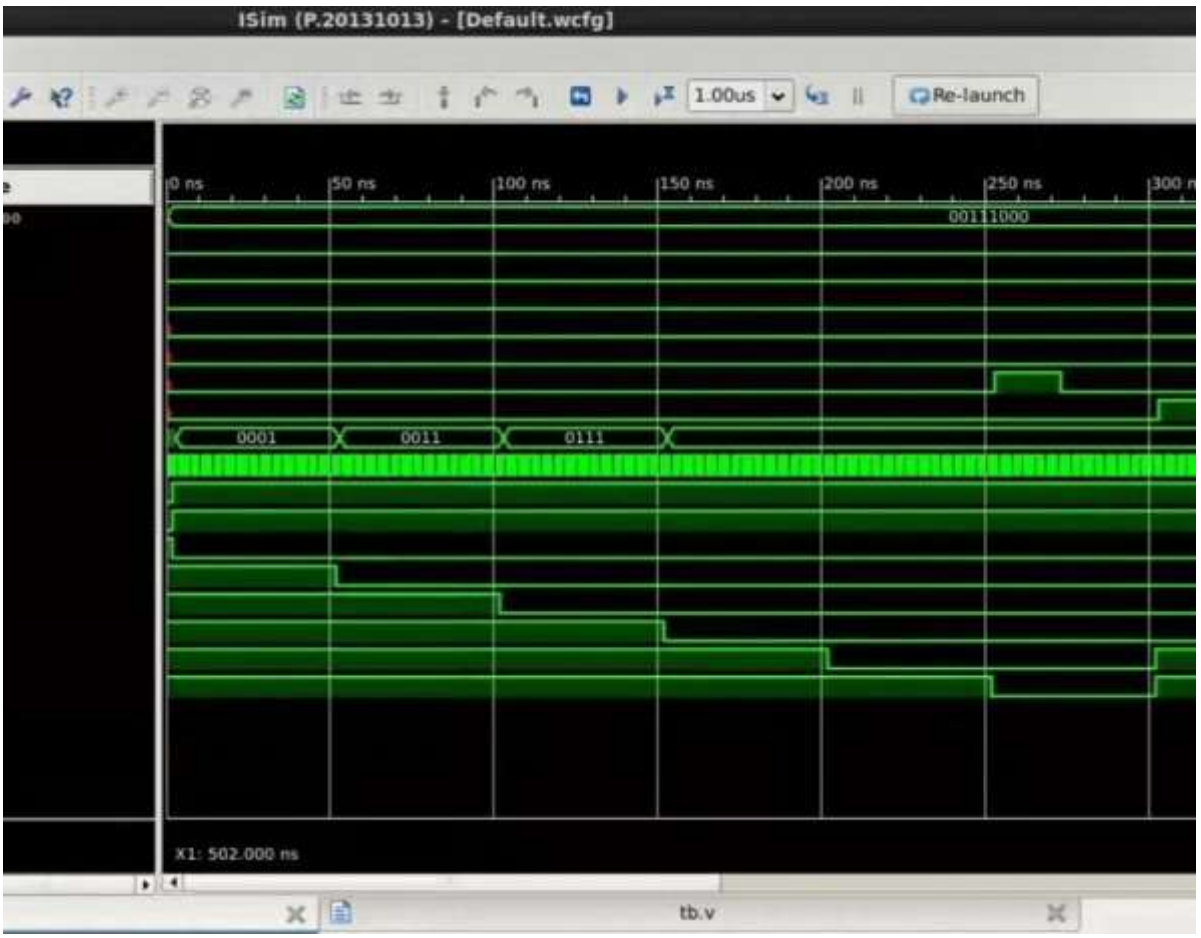



**Fig 6.3: Simulation waveform.**

## APPLICATIONS

1. **Smart Parking Systems** – Used in malls, airports, and public parking areas for efficient vehicle management.
2. **Automated Toll Booths** – Helps in vehicle identification and parking slot allocation.
3. **Residential Parking** – Manages parking slots in apartments and gated communities.
4. **Smart Cities & IoT-Based Parking** – Can be integrated with IoT for real-time monitoring and data analysis.
5. **Automated Valet Parking** – Reduces human intervention by guiding vehicles to available parking slots
6. **Traffic Management** – Helps reduce congestion in busy urban areas by directing

vehicles to available spaces.

7. **Industrial Parking Facilities** – Ensures systematic parking for employees and visitors.

## ADVANTAGES

1. **Automated Monitoring** – Uses sensors and Verilog-based logic to detect and manage parking availability.
2. **Real-Time Display** – LCD provides immediate updates on available and occupied slots.
3. **Energy Efficiency** – Reduces power consumption by turning on the display and sensors only when needed.
4. **Cost-Effective** – Uses minimal hardware components for implementation.
5. **Scalability** – Can be expanded for larger parking lots with multiple sensors and displays.
6. **Quick Parking Detection** – IR sensors detect vehicle presence and update availability status instantly.
7. **Smooth Traffic Flow** – Reduces the time spent searching for parking spaces.

**Disadvantages**

1. **Limited Accuracy** – IR sensors may not work well in extreme lighting conditions or obstructions.
2. **Hardware Dependency** – Requires continuous maintenance and calibration of sensors and display units.
3. **Initial Setup Cos**t – The installation of sensors and Verilog-based implementation requires investment.
4. **Power Requirements** – Needs a stable power source for continuous operation.
5. **Limited Environmental Adaptability** – Harsh weather conditions may affect sensor performance.

# CONCLUSION

The design and implementation of the Car Parking Management System (CPMS) using Verilog has successfully demonstrated the potential of digital systems in addressing the challenges associated with urban parking. Through the integration of sensors, control logic, and user interfaces, the CPMS provides an efficient and automated solution for managing parking spaces. The project highlights the importance of real-time data processing and user interaction in enhancing the overall parking experience for drivers.The system's architecture, built around a state machine, ensures smooth operation and quick response times, allowing for effective monitoring of parking slot occupancy and guiding vehicles to available spaces. The incorporation of features such as a digital display and an alarm system not only improves user engagement but also enhances security, making the parking environment safer and more user-friendly. The data logging capabilities further provide valuable insights into parking patterns, which can be utilized for future improvements and urban planning.In conclusion, the CPMS serves as a practical example of how Verilog can be leveraged to create sophisticated hardware solutions for real-world problems. The successful implementation of this project underscores the significance of automation in urban infrastructure and paves the way for future advancements in smart city technologies.The incorporation of features such as a digital display and an alarm system not only improves user engagement but also enhances security, making the parking environment safer and more user-friendly. The data logging capabilities further provide valuable insights into parking patterns, which can be utilized for future improvements and urban planning.In conclusion, the CPMS serves as a practical example of how Verilog can be leveraged to create sophisticated hardware solutions for real-world problems. The successful implementation of this project underscores the significance of automation in urban infrastructure and paves the way for future advancements in smart city technologies. As urban areas continue to grow, the need for efficient parking management systems will become increasingly critical, and this project lays a foundational framework for further research and development in this domain. The findings and methodologies presented in this project can inspire future innovations aimed at enhancing urban mobility and sustainability.The Car Parking Management System using Verilog and sensors like IR and L293D driver offers an efficient and automated way to manage vehicle parking. The system provides real-time updates, enhances parking efficiency, and minimizes congestion. By integrating sensors with digital logic, this design ensures a systematic and reliable parking solution.

# FUTURE SCOPE

An IoT-based parking system transforms traditional parking by using advanced technologies for enhanced efficiency, security, and user convenience. Cloud-based dashboards allow real-time monitoring, data analysis, and remote management of parking spaces. AI and ML algorithms predict parking availability based on traffic patterns and historical data, helping reduce search time and traffic congestion. Mobile app integration enables users to check spot availability, receive notifications, make reservations, and handle digital payments. Automated barriers powered by an L293D motor driver ensure secure, authorized vehicle access. Additionally, solar-powered sensors detect vehicle presence and transmit data, promoting sustainability and reducing operational costs.

# REFERENCES


[1]. https://aticleworld.com/i2c-bus-protocol-and-interface/

[2]. http://www.handsonembedded.com/stm32f103-spl-tutorial-6/

[3].S Patel, VV Dwivedi, YP Kosta, "A Parametric Characterization an Comparative Study of Okumura and Hata Propagation-loss-prediction Models for Wireless Environment", International Journal of Electronic Engineering Research, Vol. 2, No. 4, 2010

[4]. NXP Semiconductors,UM10204, *i2c bus specification and user manual*, Rev. 6- 4 April 2014

[5]. Sagar Patel, Rahul Patel and Jaymin Bhalani, "Performance Analysis & Implementation of different Modulation Techniques in Almouti MIMO Scheme with Rayleigh Channel", International Conference on Recent Trends in computing and Communication Engineering, April 2013

[6]. Jigar Kumar, Arpita Patel and Sagar Patel, "Multiuser – MIMO Broadcast Channel Techniques", International Research Journal of Engineering and Technology, Vol. 3, No. 2,Feb. 2016.

[7]. Kyounghwan An, Jungdan Choi, and Dongyong Kwak, "Automatic Valet Parking System Incorporating a Nomadic Device and Parking Servers" 2011 IEEE International Conference on Consumer Electronics (ICCE), pp. 111-112.

[8]. Gongjun Yan, Weiming Yang, Danda B. Rawat Stephan Olariu, "Smart Parking: Secure and intelligent parking system" , IEEE intelligent transportation systems magazine,pp. 18-30.

[9]. Mingkai Chen and Tianhai chang, "A Parking and Information System based on Wireless Sensor Network", Proceeding of the IEEE International Conference on Information and Automation, Shenzhen, China, June 2011, pp. 601-605

[10]. Mingkai Chen, Chao Hu and Tianhai chang, "The Research on Optimal Parking Space Choice Model in Parking Lots", Proceeding of IEEE, pp. 93-97.

[11]. Jatuporn Chinrungrueng and Udomporn Sunantachaikul, "Smart Parking: An Application of optical Wireless Sensor Network", Proceedings of the IEEE International Symposium on Application and the Internet Workshop, 2007

[12]. C. Vestri, S. Bougnoux, R. Bendahan, K. Fintzel, S. Wybo, F. Abad, and T. Kakinami,

"Evaluation of a vision-based parking assistance system," in Proc. 8th Int. IEEE Conf. Intell. Transp. Syst., Sep. 2005, pp. 131–135.

[13]. J. K. Suhr, H. G. Jung, K. Bae, and J. Kim, "Automatic free parking space detection by using motion stereo-based 3D reconstruction," Mach. Vis. Appl., vol. 21, no. 2, pp. 163– 176, Feb. 2010.

# APPENDIX

**Appendix-1:** Tools and Technologies Used Hardware

1. FPGA Development Board
2. IR Sensors
3. LED Display

**Software:**

1. Xilinx ISE/Vivado
2. ModelSim
3. Verilog HDL

**Programming Language:**

Verilog

**Appendix-2**. System Specifications

1. Clock Speed: 50 MHz (Adjustable based on FPGA configuration)
2. Power Requirements: 5V/3.3V (Depending on FPGA board)
3. Memory Utilization: Minimal, as the system mainly relies on state transitions
4. Sensor Type: IR or Ultrasonic for vehicle detection
5. Display Type: 7-segment or LED indicators

**Appendix-3**. Functional Testing

Scenario 1: Vehicle Entry

Expected Outcome: The system detects the car, assigns the nearest vacant slot, and updates the display.

Scenario 2: Vehicle Exit

Expected Outcome: The system marks the slot as vacant and refreshes availability.

Scenario 3: Full Parking

Expected Outcome: The system displays a 'FULL' message when all slots are occupied.

Scenario 4: Power Reset

Expected Outcome: The system initializes all slots to vacant upon power-up.

**Appendix-4**. Limitations

1. The system operates on a fixed number of parking slots.
2. It does not distinguish between different vehicle sizes.
3. Requires continuous power supply to maintain real-time updates.