A

MAJOR PROJECT REPORT ON

IMPLEMENTATION OF AUTOMATIC CAR PARKING USING VERILOG HDL

Submitted in partial fulfillment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SUBMITTED BY

J.KRUPAVARAM	**218R1A04L9**
J.VINAY KUMAR	**218R1A04M0**
K.GIRIDAR REDDY	**218R1A04M2**

Under the Esteemed Guidance of

Mr. V. SANTHOSH KUMAR

Assistant Professor

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA & NAAC)
Kandlakoya (V), Medchal (M), Telangana – 501401

2024-2025

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Approved by AICTE, Affiliated to JNTU Hyderabad, Accredited by NBA & NAAC) Kandlakoya (V), Medchal (M), Telangana – 501401

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that the Major Project work entitled **"IMPLEMENTATION OF AUTOMATIC CAR PARKING SYSTEM USING VERILOG HDL"** is being submitted by **J.KRUPAVARAM** bearing Roll No:**218R1A04L9, J.VINAY KUMAR** bearing Roll No**: 218R1A04M0, K.GIRIDAR REDDY** bearing Roll No:**218R1A04M2** in B.Tech IV-II semester, Electronics and Communication Engineering is a record bonafide work carried out by then during the academic year 2024-25.

INTERNAL GUIDE:	HEAD OF THE DEPARTMENT:
Mr. V. SANTHOSH KUMAR	**Dr. SUMAN MISHRA**
Assistant Professor,	
ECE CMREC	

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 **Mr. V. SANTHOSH KUMAR,** Assistant Professor of ECE for guidance and encouragement in carrying out this project work.

DECLARATION

We hereby declare that the project work entitled **"IMPLEMENTATION OF AUTOMATIC CAR PARKING SYSTEM USING VERILOG HDL"** 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 fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **ELECTRONICS AND COMMUNICATION ENGINEERING FROM JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD.**

J.KRUPAVARAM	**(218R1A04L9)**
J.VINAY KUMAR	**(218R1A04M0)**
K.GIRIDAR REDDY	**(218R1A04M2)**

ABSTRACT

The rapid urbanization and increasing vehicle ownership have accentuated the need for efficient parking solutions. This project presents the design and implementation of an Automatic Car Parking System using Verilog Hardware Description Language (HDL). The proposed system leverages digital logic and state machines to automate the parking process, aiming to optimize space utilization and reduce human intervention. The rapid growth of urban populations and the consequent increase in vehicle ownership have highlighted the urgent need for efficient and automated parking solutions. This project, titled "Implementation of Automatic Car Parking System Using Verilog HDL," addresses this need by designing a digital automated parking system using Verilog Hardware Description Language (HDL).

The proposed system integrates several key components: a central controller, parking sensors, and actuators. The central controller, designed with Verilog HDL, utilizes finite statemachines (FSM) to manage and automate the parking process. The system's architecture includes sensors for detecting the presence and location of vehicles, which feed real-time data to the controller. Based on this data, the controller orchestrates the movement ofvehicles into available parking slots using actuators. Verilog HDL is employed to model and simulate the various functionalities of the system, including slot allocation, vehicle entry and exit, and space management. The system's design incorporates algorithms for efficient space utilization and minimizes the time required for parking and retrieval operations. Additionally, the project includes detailed simulations and test cases to verify the system's performance, reliability, and scalability.

Simulation results indicate that the system effectively optimizes parking space usage, significantly reduces human intervention, and minimizes the time required for parking and retrieval compared to traditional manual methods. The proposed system leverages digital logic and state machines to automate the parking process, aiming to optimize space utilization and reduce human intervention. The rapid growth of urban populations and the consequent increase in vehicle ownership have highlighted the urgent need for efficient and automated parking solutions. The design also demonstrates flexibility and scalability, making it adaptable to various parking environments and requirements. This project not only showcases the practical application of Verilog HDL in developing sophisticated embedded systems but also provides valuable insights into the automation of urban parking solutions.

CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

The implementation of an automatic car parking system using Verilog HDL involves designing a digital circuit that can efficiently manage the parking of vehicles in a parking lot. The system consists of several modules, including a sensor module, a controller module, and a display module. The sensor module is responsible for detecting the presence or absence of a vehicle in each parking slot. This is achieved through the use of sensors such as infrared or ultrasonic sensors, which send signals to the controller module. The controller module, designed using Verilog HDL, processes these signals and determines the availability of parking slots. It then sends signals to the display module, which displays the availability of parking slots to the user.

1.1 OVERVIEW OF THE PROJECT

The controller module is the brain of the system and is responsible for managing the parking process. It is designed using a finite state machine (FSM) approach, which allows it to transition between different states based on the inputs from the sensor module. The FSM is implemented using Verilog HDL, which provides a concise and efficient way to describe the digital circuit. The display module is responsible for displaying the availability of parking slots to the user. This can be achieved through the use of LEDs, LCD displays, or other display technologies. The display module receives signals from the controller module and displays the availability of parking slots in real-time.

The project not only enhances the convenience of parking but also improves the safety and efficiency of vehicle management in parking lots. By automating both the parking and exit processes, the system minimizes human error, reduces parking time, and ensures optimal utilization of available space. Overall, the Automatic Car Parking System using Verilog HDL provides an intelligent, reliable, and cost-effective solution for managing parking lots in modern urban environments, making the parking experience more seamless and efficient for users. The system aims to provide a convenient and efficient parking experience for users, reducing the time and effort required to find an available parking slot. By automating the parking process, the system also aims to reduce the risk of accidents and congestion in the parking lot, making it a safer and more organized environment for users. Another key objective of the project is to demonstrate the application of digital circuit design and Verilog HDL in a real-world scenario. The project aims to showcase the capabilities of

Verilog HDL in designing complex digital systems and its potential to improve the efficiency and reliability of various applications.

Overall, the implementation of an automatic car parking system using Verilog HDL is a complex task that requires a deep understanding of digital circuit design, Verilog HDL, and automation. However, the benefits of such a system make it an attractive solution for modern parking management systems.

1.1 OBJECTIVE OF THE PROJECT

The objective of the project is to design and implement an automatic car parking system using Verilog HDL, which can efficiently manage the parking of vehicles in a parking lot. The primary goal is to create a system that can detect the presence or absence of a vehicle in each parking slot and display the availability of parking slots to the user.

The system aims to provide a convenient and efficient parking experience for users, reducing the time and effort required to find an available parking slot. By automating the parking process, the system also aims to reduce the risk of accidents and congestion in the parking lot, making it a safer and more organized environment for users. Another key objective of the project is to demonstrate the application of digital circuit design and Verilog HDL in a real-world scenario. The project aims to showcase the capabilities of Verilog HDL in designing complex digital systems and its potential to improve the efficiency and reliability of various applications.

Furthermore, the project aims to provide a cost-effective and scalable solution for parking management, which can be easily integrated with existing parking infrastructure. The system is designed to be flexible and adaptable, allowing it to be easily modified and updated to accommodate changing parking needs and requirements.

1.2 ORGANIZATION OF THE PROJECT

The project is organized into several modules, each with a specific function and responsibility. The first module is the sensor module, which is responsible for detecting the presence or absence of a vehicle in each parking slot. This module consists of sensors such as infrared or ultrasonic sensors, which send signals to the controller module. The second module is the controller module, which is the brain of the system. This module is designed using Verilog HDL and is responsible for processing the signals from the sensor.

The third module is the display module, which is responsible for displaying the availability of parking slots to the user. This module consists of LEDs, LCD displays, or other display technologies, which receive signals from the controller module and display the availability of parking slots in real-time. The fourth module is the user interface module, which allows users to interact with the system. This module consists of buttons, touchscreens, or other input devices, which send signals to the controller module to update the display and manage the parking process. The project also includes a power supply module, which provides power to all the modules, and a communication module, which enables communication between the different modules. The project is organized in a hierarchical manner, with the controller module at the top level, followed by the sensor module, display module, user interface module, power supply module, and communication module. This hierarchical organization allows for easy modification and updating of the system, as well as efficient communication between the different modules.

The project is also divided into several phases, including design, implementation, testing, and deployment. The design phase involves designing the digital circuit and writing the Verilog HDL code, while the implementation phase involves building the hardware and integrating the different modules. The testing phase involves testing the system to ensure that it meets the requirements and specifications, and the deployment phase involves installing the system in the parking lot and ensuring that it is functioning correctly. The system aims to provide a convenient and efficient parking experience for users, reducing the time and effort required to find an available parking slot. By automating the parking process, the system also aims to reduce the risk of accidents and congestion in the parking lot, making it a the safer and the more organized environment for any users.

By automating the parking process, the system also aims to reduce the risk of accidents and congestion in the parking lot, making it a safer and more organized environment for users. Another key objective of the project is to demonstrate the application of digital circuit design and Verilog HDL in a real-world scenario. The project aims to showcase the capabilities of Verilog HDL in designing complex digital systems and its potential to improve the efficiency and reliability of various applications. These sensors provide real-time data to a central control unit, which processes the information and makes decisions about parking allocation, guidance, and management. . The existing system may also include a user interface, such as a touchscreen display or a mobile app, to provide users with information about available parking spaces, parking fees, and other relevant details.

CHAPTER 2
LITERATURE SURVEY

EXISTING SYSTEM

The existing system for automatic car parking systems typically relies on a combination of sensors, actuators, and control units to manage the parking process. The system consists of a network of sensors, such as ultrasonic sensors, infrared sensors, and cameras, which are strategically placed throughout the parking area to detect the presence of vehicles and monitor their movement. These sensors provide real-time data to a central control unit, which processes the information and makes decisions about parking allocation, guidance, and management.

The control unit, typically a microcontroller or a dedicated IC, runs a software program that interprets the sensor data and controls the actuators, such as motors, valves, and lights, to manage the parking process. The control unit also communicates with external systems, such as payment systems, access control systems, and surveillance systems, to ensure seamless integration and efficient operation. The existing system may also include a user interface, such as a touchscreen display or a mobile app, to provide users with information about available parking spaces, parking fees, and other relevant details. For The existing system has several limitations, including high installation and maintenance costs, limited scalability, and potential security vulnerabilities. The use of multiple sensors and actuators increases the complexity of the system, making it prone to errors and faults.

Moreover, the system's reliance on a central control unit creates a single point of failure, which can lead to system downtime and inconvenience to users. Furthermore, the existing system may not be able to adapt to changing parking patterns and user behavior, leading to inefficiencies and wasted resources. To address these limitations, a new approach to automatic car parking systems is needed, one that leverages the power of Verilog HDL to design and implement a more efficient, scalable, and secure system. By using Verilog HDL, designers can create a digital circuit that can process sensor data, control actuators, and manage the parking process in a more efficient and reliable manner. The use of Verilog-HDL also enables the design of a more modular and flexible. One of the key benefits of using Verilog HDL is its ability to model and simulate complex digital systems, allowing designers to test and validate their designs before implementation.

This reduces the risk of errors and faults, and ensures that the system meets the required specifications and performance standards. Additionally, Verilog HDL enables the design of reconfigurable systems, which can be easily modified and updated to accommodate changing user needs and parking patterns. The Another advantage of using Verilog HDL is its ability to optimize system performance and reduce power consumption. By using Verilog HDL, designers can create digital circuits that are optimized for low power consumption, reducing the system's energy footprint and minimizing its environmental impact. This is particularly important for automatic car parking systems, which are often powered by batteries or solar panels and require efficient energy management to ensure reliable operation.

Furthermore, Verilog HDL enables the design of secure systems that can protect user data and prevent unauthorized access. By using Verilog HDL, designers can create digital circuits that are resistant to cyber-attacks and data breaches, ensuring the security and privacy of user information. This is critical for automatic car parking systems, which often handle sensitive user data, such as payment information and vehicle identification numbers.

Another the use of Verilog HDL offers a promising solution for designing and implementing efficient, scalable, and secure automatic car parking systems. By leveraging the power of Verilog HDL, designers can create digital circuits that can process sensor data, control actuators, and manage the parking process in a more efficient and reliable manner. The use of Verilog HDL also enables the design of modular and flexible systems, which can be easily upgraded and modified to meet changing user needs and parking patterns.

To address these limitations, a new approach to automatic car parking systems is needed, one that leverages the power of Verilog HDL to design and implement a more efficient, scalable, and secure system. This is particularly important for automatic car parking systems, which are often powered by batteries or solar panels and require efficient energy management to ensure reliable operation. Moreover, the system's reliance on a central control unit creates a single point of failure, which can lead to system downtime and inconvenience to users. Furthermore, the existing system may not be able to adapt to changing parking patterns and user behavior, leading to inefficiencies and wasted resources.

To address these limitations, a new approach to automatic car parking systems is needed, one that leverages the power of Verilog HDL to design and implement a more efficient, scalable, and secure system. By using Verilog HDL, designers can create a digital circuit that can process sensor data, control actuators, and manage the parking process in a more efficient and reliable manner. The use of Verilog-HDL also enables the design of a more modular and flexible. One of the key benefits of using Verilog HDL is its ability to model and simulate complex digital systems, allowing designers to test and validate their designs before implementation.

The control unit, typically a microcontroller or a dedicated IC, runs a software program that interprets the sensor data and controls the actuators, such as motors, valves, and lights, to manage the parking process. The control unit also communicates with external systems, such as payment systems, access control systems, and surveillance systems, to ensure seamless integration and efficient operation. This is particularly important for automatic car parking systems can process sensor data, control actuators, and manage.

This reduces the risk of errors and faults, and ensures that the system meets the required specifications and performance standards. Additionally, Verilog HDL enables the design of reconfigurable systems, which can be easily modified and updated to accommodate changing user needs and parking patterns. The Another advantage of using Verilog HDL is its ability to optimize system performance and reduce power consumption. By using Verilog HDL, designers can create digital circuits that are optimized for low power consumption, reducing the system's energy footprint and minimizing its environmental impact. This is particularly important for automatic car parking systems, which are often powered by batteries or solar panels and require efficient energy management to ensure reliable operation.

A feedback mechanism is crucial for real-time corrections. Feedback systems monitor the car’s position and orientation continuously and adjust movements accordingly. Verilog modules that process feedback signals, like position and speed, are employed to correct deviations from the desired parking path. The system can automatically re-adjust based on the feedback loop, enhancing accuracy and safety. Some implementations suggest the use of microcontrollers or FPGAs for real-world deployment. The control unit also communicates with external systems, such as payment systems, access control systems, and surveillance systems, to ensure seamless integration and efficient operation. This is particularly important for automatic car parking systems can process sensor data, control actuators, and manage.

2.1 PROPOSED SYSTEMS

The Implementing an automatic car parking system using Verilog HDL has been explored with various methodologies to handle sensor integration, control logic, and parking algorithms. Here's a summary of some approaches:

Sensor-Based Detection and Distance Measurement:

A commonly proposed method is to use ultrasonic or infrared sensors integrated with Verilog HDL to detect vehicle positions and obstacles in the parking area. Sensors are typically interfaced with a digital controller designed in Verilog to read proximity data, enabling precise positioning of the vehicle. The sensor data is processed to determine distance and orientation, which then guides the car's movements. Verilog modules are created to process real-time sensor inputs, ensuring correct parking alignment.

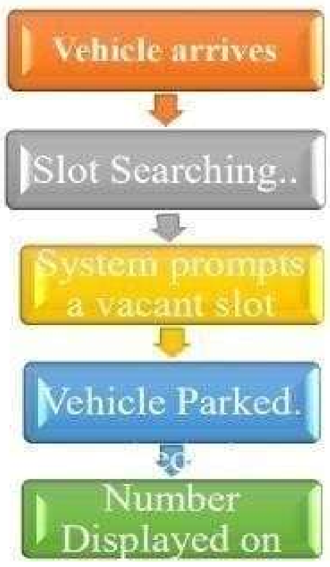


FIG:2.1 Steps For Automatic Car Parking

Finite State Machine (FSM) Control:

FSMs are a prevalent approach in Verilog-based parking systems, as they allow for a structured representation of the parking stages: idle, detect, align, and park. Each state in the FSM represents a specific action in the parking process. For instance, the "detect" state reads sensor data, while the "align" state adjusts the car's position. The FSM transitions between these states based on sensor data and specific conditions coded in Verilog, leading to efficient control over parking operations. This method is beneficial for defining sequential parking tasks and makes the system more robust.

Algorithmic Parking Logic and Path Planning:

Some implementations focus on algorithmic approaches, embedding path-planning algorithms within Verilog HDL to calculate optimal parking paths. Algorithms such as collision avoidance and path optimization are used to navigate the car into a parking spot. These algorithms are translated into Verilog HDL by defining coordinates and movement instructions, simulating a simplified path planner. This logic is particularly useful for parallel parking scenarios and requires precise handling of movements.

Feedback and Control Systems:

A feedback mechanism is crucial for real-time corrections. Feedback systems monitor the car's position and orientation continuously and adjust movements accordingly. Verilog modules that process feedback signals, like position and speed, are employed to correct deviations from the desired parking path. The system can automatically re-adjust based on the feedback loop, enhancing accuracy and safety.

Integration with Microcontrollers and FPGA Implementation:

Some implementations suggest the use of microcontrollers or FPGAs for real-world deployment of Verilog-based parking systems. The Verilog code is synthesized on an FPGA, which then interfaces with sensors and actuators to control the vehicle in real time. The FPGA's parallel processing capabilities allow for simultaneous handling of multiple signals, improving response times for complex parking maneuvers. Microcontroller-based systems can also use Verilog modules as part of the design for sensor data processing and control logic, allowing the car to perform efficient parking operations in dynamic environments.

The project is organized in a hierarchical manner, with the controller module at the top level, followed by the sensor module, display module, user interface module, power supply module, and communication module. This hierarchical organization allows for easy modification and updating of the system, as well as efficient. The system consists of several modules, including a sensor module, a controller module, and a display module. The sensor module is responsible for detecting the presence or absence of a vehicle in each parking slot. This is achieved through the use of sensors such as infrared or ultrasonic sensors, which send the signals to the controller of the module.

Some implementations focus on algorithmic approaches, embedding path-planning algorithms within Verilog HDL to calculate optimal parking paths. Algorithms such as collision avoidance and path optimization are used to navigate the car into.

2.2 INTURDUCTION TO VLSI

VLSI is the technique of solidifying with a specific end goal to make encouraged circuits a giant number of transistor-based circuits into a solitary chip. VLSI started in the 1970s when complex semiconductor and correspondence types of progress were being made. The chip is a VLSI gadget. The term is no more as would be normal as it once had all the earmarks of being, as chips have stretched out in multifaceted nature into the limitless.

Overview:

The fundamental semiconductor chips held one transistor each. Following advances included more transistors, and, along these lines, more individual cutoff points or structures were merged after some time. The at first intertwined circuits held just a few contraptions, perhaps upwards of ten diodes, transistors, resistors and capacitors, making it conceivable to manufacture no less than one technique for thinking passages on a solitary gadget. Eventually implied splendidly as "little scale joining" (SSI), overhauls in procedure prompted gadgets with a couple of reason entryways, known as monstrous scale combine (LSI), i.e. frameworks with no not as much as a thousand strategy for thinking passages. Current development has moved far past this etching and the present chip have different a considerable number of gateways and boundless individual transistors.

At one time, there was a push to name and change unmistakable levels of huge scale joining above VLSI. Terms like Ultra-noteworthy scale Integration (ULSI) were utilized. In any case, the tremendous number of entries and transistors accessible on general contraptions has rendered such fine refinements simple to invalidate. By leveraging the power of Verilog HDL, designers can create digital circuits that can process sensor data, control actuators, and manage the parking process in a more efficient and reliable manner. The use of Verilog HDL also enables the design of modular and flexible systems, which can be easily upgraded and modified to meet changing user needs and parking patterns. Correspondence inside a chip can happen a few times speedier than correspondence between chips on a printed circuit stack up. The quick of circuits on-chip is a direct result of their little size-more diminutive portions and wires have humbler parasitic capacitances to back off the banner. Current plans, instead of the most trustworthy gadgets, use wide graph computerization and electronic Terms recommending more discernible than VLSI levels of mix are no more in boundless use. Truth be told, even VLSI is before long to some degree captivating, given the typical uncertainty that all chip are VLSI or better.

Starting mid 2008, billion-transistor processors are monetarily accessible, a portrayal of which is Intel's Montecito Itanium chip. This is relied on to wind up more common as semiconductor create moves from the present time of 65 nm frameworks to the going with 45 nm times (while encountering new difficulties, for example, expanded arrangement transversely completed approach corners). Another remarkable case is NVIDIA's 280 game-plan GPU. This microchip is remarkable in the way that its 1.4 Billion transistor check, arranged for a teraflop of execution, is totally devoted to premise (Itanium's transistor tally is, everything considered, because of the 24MB L3 store). Current plans, instead of the most trustworthy gadgets, use wide graph computerization and electronic technique for thinking blend to lay out the transistors, drawing in more lifted measures of erraticisms in the subsequent defense esteem. Certain tip top support pieces like the SRAM cell, on the other hand, are up 'til now shaped by hand to guarantee the most huge capacity (as a not as much as reliable control by bowing or isolating set outline standards to get the last piece of execution by exchanging security).

2.3 What is VLSI?

VLSI stays for "Enormous Scale Integration". This is the field which incorporates squeezing progressively method of reasoning devices into humbler and tinier districts.

VLSI

Simply we say Integrated circuit is various transistors on one chip.

- Design/amassing of to an extraordinary degree little, complex equipment using balanced semiconductor material
- Integrated circuit (IC) may co+3

ntain an enormous number of transistors, each two or three mm in assess

- Applications titanic: most electronic method of reasoning devices.

This advancement allows for highly complex and powerful electronic devices, such as microprocessors, memory chips, and system-on-chip (SoC) designs. It begins with specification, where the functionality of the chip is defined. Next, RTL coding is performed using Verilog to describe the circuit's logic. After coding, the design undergoes simulation using testbenches to verify its correctness. Once verified, the design is synthesized, meaning it is converted into a network of logic gates. The synthesized circuit is then passed through placement and routing (layout), where physical components are arranged on the silicon wafer. Finally, the chip undergoes fabrication and testing before it is ready for real-world applications.

2.4 History of Scale Integration:

Scale integration refers to the process of combining different levels of analysis, measurement, or operation within a system to create a seamless and efficient framework. The concept has evolved over centuries, particularly in fields like engineering, economics, and computing. Early examples of scale integration can be traced back to the industrial revolution, where mechanized systems required synchronization of different production processes. As industries expanded, the need to integrate small-scale and large-scale production units became crucial to maintaining efficiency and reducing operational costs.

The development of integrated circuits in the 1950s and 1960s revolutionized technology by enabling the miniaturization of electronic components while enhancing computational power. This integration allowed computers to move from room-sized machines to compact, more efficient devices. Similarly, in economics, globalization and technological advancements have led to the integration of local, regional, and international markets, fostering economic interdependence and large-scale production networks.

Advantages of ICs over discrete parts:

While we will concentrate on consolidated circuits , the properties of composed circuits-what we can and can't capably put in a joined circuit-for the most part of the entire structure. Facilitated circuits improve system qualities in a couple of fundamental ways. ICs have three key great conditions over cutting edge circuits worked from discrete parts:

□ Size. Composed circuits are extensively smaller the two transistors and wires are contracted to micrometer sizes, stood out from the millimeter or centimeter sizes of discrete sections. Minimal size prompts ideal conditions in speed and power usage, since smaller parts have more diminutive parasitic resistances, capacitances, and inductances.

□Speed. Signs can be traded between method of reasoning 0 and justification 1 impressively speedier inside a chip than they can between chips. Correspondence inside a chip can happen a few times speedier than correspondence between chips on a printed circuit stack up. The quick of circuits on-chip is a result of their little size-more diminutive portions and wires have humbler parasitic capacitances to back off banner.This integration allowed computers to move from room-sized machines to compact, more efficient devices. This integration allowed computers to move from room-sized machines to compact, more efficient devices.

2.5 VLSI and systems:

These good conditions of fused circuits change over into central focuses at the system level:

- Smaller physical size. Littleness is regularly use in itself-consider minimized TVs or handheld phones.
- Lower control use. Supplanting a humble bundle of standard parts with a lone chip diminishes signify control usage. Diminishing force usage has a far reaching impact on the straggling leftovers of the structure: a humbler, more affordable power supply can be used; since less power use infers less warmth, a fan may never again be imperative; a less mind boggling authority with less securing for electromagnetic ensuring may be achievable, also. Reduced cost.
- Diminishing the amount of parts, the power supply essentials, authority costs, and so forth, will unavoidably decrease system cost. The continuously outstretching impact of joining is with the ultimate objective that the cost of a system worked from customICs can be less, notwithstanding the way that the individual ICs cost more than the standard parts they supplant.
- Understanding why composed circuit development has such noteworthy effect on the blueprint of electronic structures requires understanding both the advancement of IC manufacturing and the money related issues of ICs and propelled systems. As industries expanded, the need to integrate small-scale and large-scale production units became crucial to maintaining efficiency and reducing operational costs.
- The development of integrated circuits in the 1950s and 1960s revolutionized technology by enabling the miniaturization of electronic components while enhancing computational power.
- The concept has evolved over centuries, particularly in fields like engineering, economics, and computing.

Applications

- Electronic structure in cars.
- Digital devices control VCRs.
- Transaction getting ready structure, ATM.
- Personal PCs and Workstations
- Medical electronic structures.

Applications of VLSI:

Electronic structures now play out a wide grouping of errands in consistently life. Electronic systems in some cases have supplanted instruments that worked mechanically, utilizing pressurized water, or by various means; equipment are for the most part humbler, more versatile, and less difficult to profit. In various cases electronic systems have made totally new applications. Electronic structures play out a variety of endeavors, some of them recognizable, some more concealed:

- Personal redirection systems, for instance, reduced MP3 players and DVD players perform refined figurings with astoundingly little imperativeness.
- Electronic structures in cars work stereo systems and introductions; they in like manner control fuel mixture systems, adjust suspensions to evolving scene, and play out the control limits required for antilock braking (ABS) systems.
- Digital devices pack and decompress video, even at predominant quality data rates, on-the-fly in client equipment.
- Low-cost terminals for Web scrutinizing still require current devices, regardless of their committed limit.
- Personal PCs and workstations give word-getting ready, cash related examination, and diversions. PCs fuse both central taking care of units (CPUs) and extraordinary reason hardware for hover get to, speedier screen show, et cetera.
- Medical electronic systems measure genuine limits and perform complex taking care of figurings to alert about peculiar conditions. The openness of these mind boggling systems, far from overwhelming purchasers, just makes enthusiasm for impressively all the more astounding structures.

The creating headway of employments reliably pushes the layout and amassing of fused circuits and electronic systems higher than any time in recent memory of multifaceted design. Additionally, perhaps the most shocking typical for this gathering of systems is its variety as structures end up being more personality boggling,we fabricate not two or three comprehensively valuable PCs yet rather a perpetually broad extent of remarkable reason structures. Our ability to do all things considered is a showing of our creating expert of both joined circuit gathering and plan, the extending solicitations of customers continue.

2.6 WHY VLSI ?

The implementation of an automatic car parking system using Verilog HDL (Hardware Description Language) and VLSI (Very Large Scale Integration) technology offers several advantages. Firstly, VLSI allows for the design and fabrication of complex digital circuits on a single chip, enabling the creation of a compact and efficient parking system. By utilizing Verilog HDL, designers can describe the digital circuitry at a high level of abstraction, making it easier to design, simulate, and verify the system.One of the primary reasons for using VLSI for an automatic car parking system is the need for high-speed processing and real-time control. VLSI technology enables the design of high-performance digital circuits that can process large amounts of data quickly and efficiently. This is particularly important in an automatic car parking system, where sensors and cameras generate vast amounts of data that need to be processed in real-time to ensure safe and efficient parking.

Another advantage of using VLSI is the ability to integrate multiple components onto a single chip, reducing the overall size and power consumption of the system. This is particularly important in an automotive application, where space and power are limited. Byintegrating multiple components, such as sensors, controllers, and memory, onto a single chip, the system becomes more compact, reliable, and energy-efficient.Furthermore, VLSI technology enables the design of custom digital circuits that can be optimized for specific applications. In the case of an automatic car parking system, custom-designed digital circuits can be optimized for tasks such as image processing, sensor data analysis, and control algorithms. This results in a more efficient and effective system that can accurately detect and respond to the parking environment.

In addition, using Verilog HDL for the design and implementation of the automatic car parking system provides several benefits. Verilog is a widely used and well-established HDL that allows designers to describe digital circuits at a high level of abstraction. This makes it easier to design, simulate, and verify complex digital circuits, reducing the risk of errors and improving the overall quality of the system. Additionally, Verilog is supported by a wide range of design tools and software, making it easier to integrate the system with other components and technologies. the use of VLSI technology and Verilog HDL for the implementation of an automatic car parking system offers several advantages, including high-speed processing, compact design, and customizability. By leveraging these technologies, designers can create a more efficient.

2.7 DESIGN APPROACHES

The design approach for an automatic car parking system using Verilog HDL and VLSI technology involves a structured and systematic methodology. The first step is to define the system requirements and specifications, including the type of sensors and cameras to be used, the parking lot layout, and the desired level of automation. This is followed by a detailed analysis of the system's functional and performance requirements, including the processing speed, memory requirements, and power consumption. Once the system requirements are defined, the next step is to design the digital circuitry using Verilog HDL. This involves creating a high-level description of the digital circuitry, including the sensors, controllers, and memory components. The Verilog code is then simulated and verified using specialized software tools to ensure that it meets the system requirements and specifications.

The next step is to synthesize the Verilog code into a netlist, which is a circuit description that can be used to program a Field-Programmable Gate Array (FPGA) or Application-Specific Integrated Circuit (ASIC). The netlist is then used to program the FPGA or ASIC, which is the brain of the automatic car parking system. The FPGA or ASIC is then integrated with the sensors, cameras, and other components to create a complete system. The system is then tested and validated to ensure that it meets the system requirements and specifications. This involves testing the system's functionality, performance, and reliability under various operating conditions.

To ensure the system's reliability and fault tolerance, a fault-tolerant design approach is adopted. This involves designing the system to continue operating even if one or more components fail. This is achieved through the use of redundancy, error detection and correction, and fail-safe mechanisms. The design approach for an automatic car parking system using Verilog HDL and VLSI technology involves a structured and systematic methodology that ensures the system meets the required specifications and performance requirements. By adopting a fault-tolerant, modular, and scalable design approach, the system can provide reliable and efficient parking services while minimizing maintenance and upgrade costs.

This makes it easier to design, simulate, and verify complex digital circuits, reducing the risk of errors and improving the overall quality of the system. Additionally, Verilog is supported by a wide range of design tools and software,

Steps in the VLSI Design Process

The different steps in the embedded system design flow/flow diagram include the following.

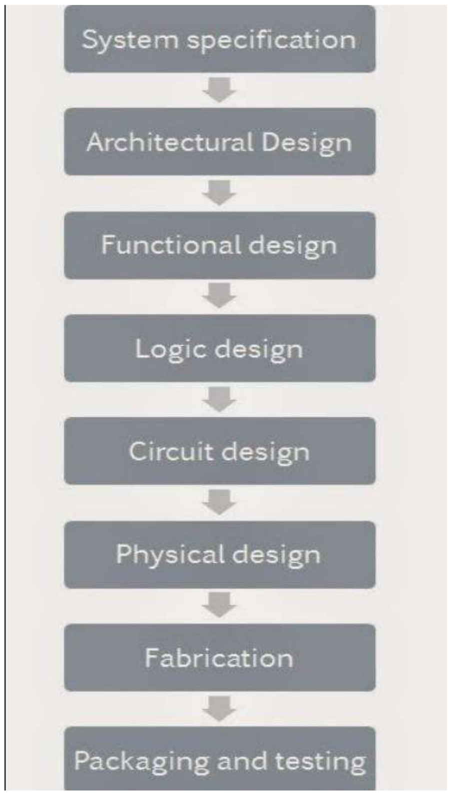


FIG:2.2 VLSI-Process-Steps

System Specification

The first step of any design process is to lay down the specifications of the system. System specification is a high-level representation of the system. The factors to be considered in this process include: performance, functionality, and physical dimensions .The fabrication technology and design techniques are also considered.

Architectural Design

The basic architecture of the system is designed in this step. This includes, such decisions as RISC (Reduced Instruction Set Computer) versus CISC (Complex Instruction Set Computer), number of ALUs, Floating Point units, number and structure of pipelines, and size of caches among others.

Logic Design

In this step the control flow, word widths, register allocation, arithmetic operations, and logic operations of the that represent the functional design are derived and tested.The design process begins with defining the desired functionality using Boolean algebra and logic gates such as AND, OR, NOT, NAND, NOR, XOR, and XNOR. These fundamental building blocks are used to create more complex combinational and sequential circuits. Boolean expressions and minimization techniques like Karnaugh maps (K-maps) and the Quine-McCluskey method help optimize logic circuits for efficiency and performance.

Circuit Design

The purpose of circuit design is to develop a circuit representation based on the logic design. The Boolean expressions are converted into a circuit representation by taking into consideration the speed and power requirements of the original design. Circuit Simulation is used to verify the correctness and timing of each component.

Physical Design

In this step the circuit representation (or netlist) is converted into a geometric representation. As stated earlier, this geometric representation of a circuit is called a layout. Layout is created by converting each logic component into a geometric representation, which perform the intended logic function of the corresponding component. Connections between different components are also expressed as geometric patterns typically lines in multiple layers.

Fabrication

After layout and verification, the design is ready for fabrication. Since layout data is typically sent to fabrication on a tape, the event of release of data is called Tape Out. Layout data is converted into photo-lithographic masks, one for each layer. Masks identify spaces on the wafer, where certain materials need to be deposited, diffused or even removed. Silicon crystals are grown and sliced to produce wafers.

Extremely small dimensions of VLSI devices require that the wafers be polished to near perfection. The fabrication process consists of several steps involving deposition, and diffusion of various materials on the wafer. During each step one mask is used. Several dozen masks may be used to complete the fabrication process. The design approach for an automatic car parking system using Verilog HDL and VLSI technology involves a structured and systematic methodology that ensures the system meets the required specifications and performance requirements. By adopting a fault-tolerant, modular, and scalable design approach, the system can provide reliable and efficient parking services while minimizing maintenance and upgrade costs. The design approach for an automatic car parking system using Verilog HDL and VLSI technology involves a structured and systematic methodology.

Packing And Testing

Finally, the wafer is fabricated and diced into individual chips in a fabrication facility. Each chip is then packaged and tested to ensure that it meets all the design specifications and that it functions properly. Chips used in Printed Circuit Boards (PCBs) are packaged in Dual In-line Package (DIP), Pin Grid Array (PGA), Ball Grid Array (BGA), and Quad Flat Package (QFP). Chips used in Multi-Chip Modules (MCM) are not packaged, since MCMs use bare or naked chips.

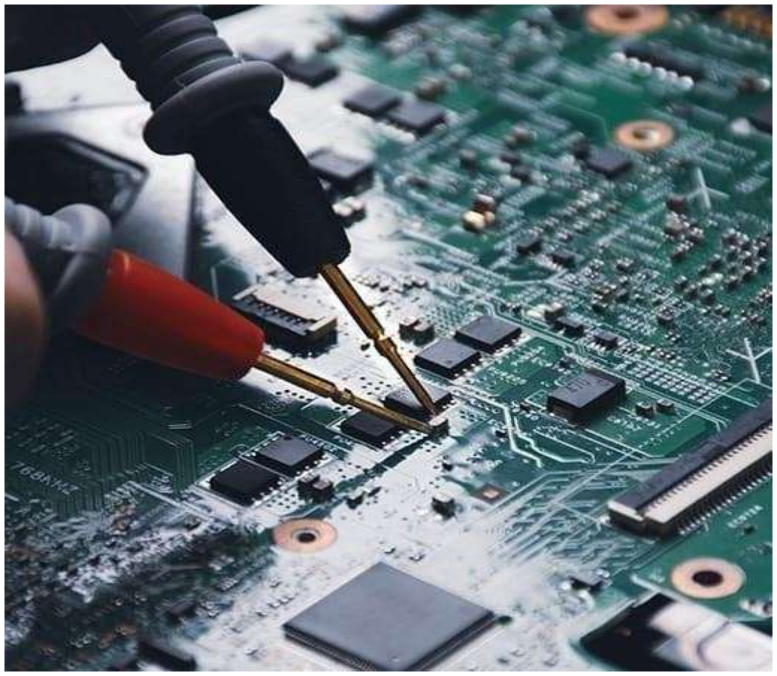


Fig:2.3 VLSI CHIP HARDWARE

VLSI is used in a variety of technologies across industries. Some examples include:

Computing and Information Technology:

VLSI is the backbone of modern computing, enabling the development of powerful processors, memory chips, and graphics processing units (GPUs). It has driven the evolution of personal computers, laptops, smartphones, and data centers, which are essential for various applications, including artificial intelligence, cloud computing, and the Internet of Things (IoT).

Telecommunications:

VLSI is used in the design and manufacturing of telecommunications equipment, such as routers, switches, and network processors, which facilitate high-speed data transmission and communication networks.

Consumer Electronics:

VLSI is used in the development of various consumer electronics, including smartphones, tablets, smart TVs, and gaming consoles, which rely on complex integrated circuits to deliver high-performance and low-power consumption.

Automotive Electronics:

VLSI is used in the design of advanced driver-assistance systems (ADAS), such as lane departure warning systems, adaptive cruise control, and automatic emergency braking, which rely on sophisticated sensors and processing units.

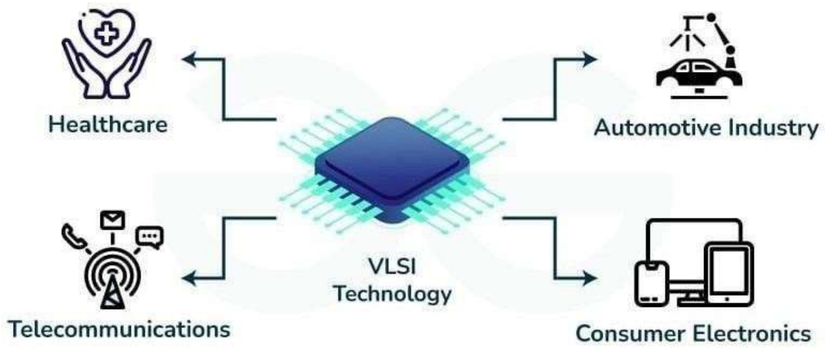


FIG:2.4 APPLICATIONS OF VLSI

Medical Devices:

VLSI is used in the development of medical devices, such as MRI and CT scanners, which require high-performance processing and imaging capabilities to produce detailed images of the human body.

Aerospace and Defense: VLSI is used in the design of complex electronic systems for military and aerospace applications, including radar systems, communication systems, and navigation systems.

Internet of Things (IoT):

VLSI is used in the development of IoT devices, such as smart home appliances, wearables, and industrial sensors, which require low-power consumption and high-performance processing capabilities. VLSI technology is essential in developing compact, high-performance chips that power IoT devices, ensuring seamless connectivity, real-time processing, and energy-efficient operation. These chips include microcontrollers, sensors, communication modules, and AI accelerators, all optimized for IoT ecosystems.

2.8 COMBINATION OF LOGIC DEVICES

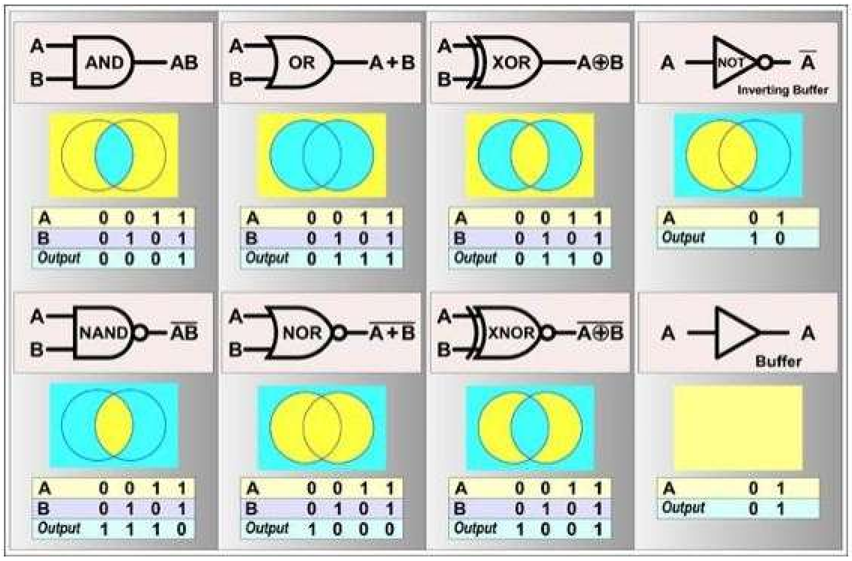


FIG:2.5 Logic Gates

Logic gates are physical devices that use combinational logic to switch an electrical one (“1”) or zero (“0”) to downstream blocks in digital design. Combinational logic uses those bits to send or receive data within embedded systems. Data bits build into digital words usedto communicate with other design blocks within the system. Digital bits and words do this with logic gates in an organized fashion using dedicated address, data, or control signal nodes. Logic gates are the physical devices that enable processing of many 1’s and 0’s. Logic families are collections of integrated circuits containing logic gates that perform functions needed by embedded systems to communicate with one another to drive the design. Logic gates are organized into families relative to the type of material and its operational characteristics.

Most logic gates are made from silicon, although some utilize gallium arsenide or other semiconductor materials. The semiconductor material is doped for organization into layers. The doped layers drive power capabilities and typical impedances at input or outputs of each gate. Logic gates used together must employ the same, or complementary, material properties. Knowledge of material properties for logic gates will drive selection of parts within design blocks. Data bits build into digital words used to communicate with other design blocks within the system. Digital bits and words do this with logic gates in an organized fashion using dedicated address, data, or control.

VLSI systems' evolution was built from combinational logic families made possible from the discovery of the transistor. The transistor is made from semiconductor material and is compact. It is able to handle large amounts of power quickly. The transistor employs three terminals to activate electron flow for use in downstream devices as electricity.

Electricity represented as 1's and 0's combines to communicate information throughout an embedded system. Because of its compact size, many millions of transistors combine within very small spaces. This allows millions of gates to operate in compact areas while transmitting and receiving mind-boggling amounts of intelligence through combinational logic. This is all accomplished within a minimal power budget.

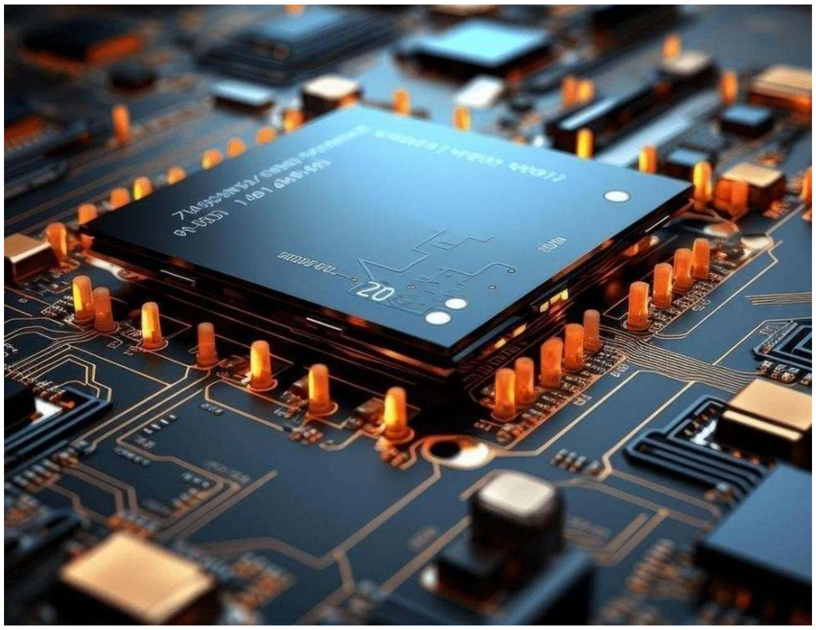


FIG:2.6 VLSI Group Of Chips

Data bits build into digital words usedto communicate with other design blocks within the system. Digital bits and words do this with logic gates in an organized fashion using dedicated address, data, or control signal nodes. The semiconductor material is doped for organization into layers. The doped layers drive power capabilities and typical impedances at input or outputs of each gate. Logic gates used together must employ the same, or complementary, material properties.The transistor is made from semiconductor material andis compact. It is able to handle large amounts of power quickly. The transistor employs three terminals to activate electron flow for use in downstream.

CHAPTER 3
HARDWARE REQUIREMENTS

3.1 VLSI Hardware

Designing VLSI (Very Large Scale Integration) circuits is a sophisticated process that requires a range of specialized hardware to manage the various phases of design, simulation, verification, and prototyping. At the core of the setup is a high-performance workstation. This workstation should ideally have a multi-core processor, ample RAM (at least 32 GB, with 64 GB or more recommended), and high-capacity solid-state drives (1 TB or more). These specifications ensure the efficient handling of complex and computation-heavy tasks like simulation, synthesis, and layout, which are central to VLSI design. Additionally, a powerful GPU can be beneficial for certain design visualization tasks, helping to manage large data sets and improve processing times.

FPGA development boards are essential for prototyping VLSI designs. FPGAs like the Xilinx Zynq UltraScale+ or Intel Stratix are used to implement and test designs before committing them to silicon. These boards allow for flexible testing and modification, making them suitable for validating design functionality and identifying any potential issues. For more advanced testing, ASIC prototyping platforms such as the Synopsys HAPS or Cadence Palladium enable designers to prototype VLSI systems at near-final specifications, which is essential for larger or high-performance designs. These platforms support high-speed interfaces and precise timing controls, making them invaluable for ensuring the functionality of the final product. The system requires a range of sensors to detect the presence and position of vehicles in the parking lot. These sensors may include ultrasonic sensors, infrared sensors, and cameras. The sensors must be able to detect the vehicle's distance, speed, and direction, and provide accurate data to the controller.

The system requires high-resolution cameras to capture images of the parking lot and vehicles. The cameras must be able to capture clear images in various lighting conditions, including daylight, nighttime, and low-light conditions. The cameras may be equipped with features such as night vision, motion detection, and weather resistance. Thirdly, the system requires a powerful controller to process the data from the sensors and cameras. The controller must be able to process large amounts of data in real-time, and make decisions based on the data. Logic gates used together must employ the same, or complementary, material properties. Knowledge of material properties for logic gates will drive selection.

In addition to these prototyping tools, Electronic Design Automation (EDA) tools are crucial software resources that facilitate design entry, synthesis, simulation, and verification. These tools, like Cadence Virtuoso for layout, Synopsys Design Compiler for synthesis, and Mentor Graphics Model-Sim for simulation, are installed on workstations and help automate many aspects of the VLSI design process. Running EDA tools often requires significant computing power, so having a robust workstation or even server infrastructure for cloud-based design is highly beneficial, particularly for complex designs that need extensive simulations. Multi-node server clusters with high memory capacity (128 GB RAM or more per node) are ideal for this purpose, offering high processing power and scalability for design teams working on large-scale projects.

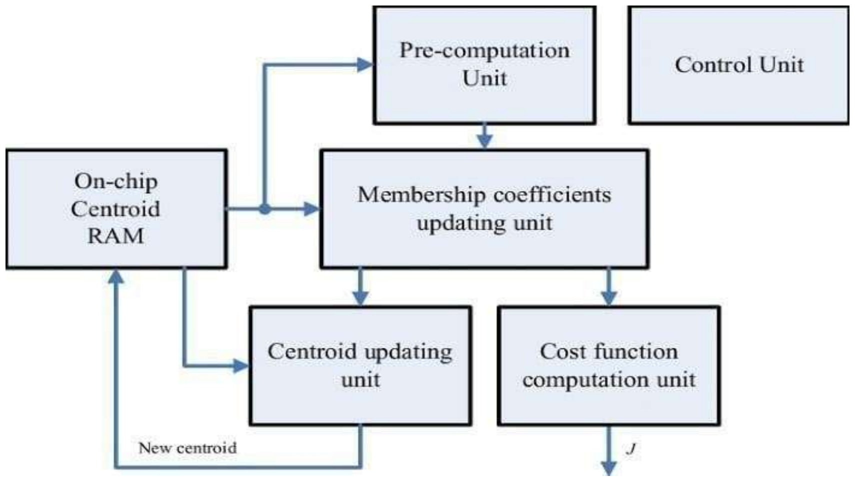


Fig:3.1 VLSI Hardware Block Diagram

For physical testing and debugging, oscilloscopes and logic analyzers play a critical role in validating signal integrity, timing, and the behavior of VLSI circuits on FPGA or ASIC prototypes. These devices, such as those from Tektronix or Keysight, provide real-time data, allowing designers to monitor and adjust electrical signals to meet design specifications. Power supplies and power analysis tools are also essential for supplying stable and adjustable power to the prototypes. Accurate power measurement tools like programmable DC power supplies are used to analyze power consumption, which is a key aspect of VLSI design, especially for low-power or battery-operated applications.

For designs with high power requirements, thermal imaging and heat management tools can be beneficial. Thermal cameras and heat sinks help monitor and control temperatures, ensuring the stability of the design under load conditions. Alongside this, efficient data storage solutions are necessary, given the extensive amount of data generated by design files, simulation results, and version control. High-speed SSDs for local storage and network-attached storage (NAS) solutions are commonly used to manage these large files, ensuring easy access and secure backup of crucial design data.

Finally, networking equipment is essential to facilitate collaborative work environments, especially in teams that rely on cloud or remote access for design and simulation. Gigabit Ethernet or faster networking, VPNs for secure access, and reliable routers ensure that design files and simulation results can be shared and accessed seamlessly across different locations. This reliable networking infrastructure is fundamental for collaborative VLSI design, enabling team members to work together efficiently on large projects and access
shared resources smoothly.
The following sub- sections discuss the various aspects of hardware requirements.

High-Performance Workstations:

High-performance workstations are essential in VLSI (Very Large Scale Integration) design, offering the necessary computing power to handle resource-intensive tasks such as simulation, synthesis, and physical layout. These workstations are equipped with advanced hardware to manage the significant data and processing needs of VLSI projects, helping designers tackle complex designs with millions to billions of transistors efficiently. The power of these workstations directly influences the speed and accuracy of the VLSI design process, ensuring the effective execution of detailed, large-scale integrated circuit projects.

The core of a high-performance workstation lies in its multi-core processor, with models like Intel Xeon or AMD Ryzen Thread ripper offering high core counts for parallel processing. A minimum of 8 cores is recommended, with 16 or more ideal for VLSI tasks like logic simulation and timing analysis, which benefit greatly from parallelism. This multi-core setup enables efficient multitasking, supporting simultaneous EDA (Electronic Design Automation) tool usage and reducing time spent on iterative processes in synthesis and verification, essential for completing complex designs in a timely manner. Power supplies and power analysis tools are also essential for supplying stable and adjustable power to the prototypes. Accurate power measurement tools like programmable DC power supplies are used to analyze power consumption, which is a key aspect of VLSI design, especially for low-power or battery-operated applications.High-speed SSDs for local storage and network-attached storage (NAS) solutions. parts within design blocks. Data bits build into digital words used to communicate with other design blocks within the system.

ASIC Prototyping Platforms

Application-Specific Integrated Circuit prototyping platforms are essential tools VLSI design, providing designers with a means to test and verify their custom integrated circuit designs at near-final specifications before committing to the costly process of manufacturing. These platforms enable designers to validate functionality, assess performance, and refine the design by creating a prototype that closely mimics the behaviorof the intended ASIC. Through prototyping, designers can identify and address potential issues in real-time, ensuring that the final ASIC will meet the required performance and reliability standards. Given the complexity and high stakes of ASIC production, prototyping platforms play a crucial role in risk reduction.

Typically, ASIC prototyping platforms utilize high-capacity FPGAs (Field-Programmable Gate Arrays), which offer a reconfigurable environment to implement the digital logic of an ASIC design. Using FPGA-based platforms, such as Synopsys HAPS, Cadence Palladium, or Mentor Graphics Veloce, designers can map their ASIC designs onto FPGAs, enabling functional and timing verification at a high speed. This approach provides a versatile testing environment, allowing engineers to reconfigure and iterate the design as needed. Unlike ASICs, which are fixed-function once fabricated, FPGAs in prototyping platforms can be reprogrammed, providing flexibility for design changes and updates during testing.

FPGA Development Boards

FPGA development boards are essential tools in VLSI design, offering designers a versatile platform to implement, test, and validate digital circuits before committing to a final silicon design. Unlike ASICs, which are fixed-function once manufactured, FPGAs are reconfigurable devices composed of programmable logic blocks and interconnects. This re program ability allows designers to quickly modify, test, and debug their designs in real time. FPGA development boards enable a fast and flexible approach to prototyping, making them valuable for verifying complex designs, evaluating functionality, and identifying any potential issues without the high costs associated with fabricating custom silicon.

These additional components allow the FPGA to communicate with other devices and mimic the environment in which the final application will operate. Simulation tools, such as Mentor Graphics Model-Sim or Synopsys VCS, are then used to test the functionality of these designs in various scenarios, catching logical errors and verifying . parts within design.

A typical FPGA development board includes an FPGA chip, along with essential components such as power supply circuitry, I/O interfaces, memory modules, and sometimes even embedded processors. These additional components allow the FPGA to communicate with other devices and mimic the environment in which the final application will operate. Boards like the Xilinx Zynq Ultra Scale+ and Intel Stratix provide high- density FPGAs with abundant resources, such as DSP blocks, memory, and high-speed I/O, making them suitable for complex VLSI designs and applications that require substantial computational power. For simpler applications or educational purposes, lower-cost boards. like the Intel Cyclone or Xilinx Spartan series offer adequate resources for smaller-scale.

EDA (Electronic Design Automation) Tools

Electronic Design Automation tools are a suite of specialized software applications used inVLSI (Very Large-Scale Integration) design to automate and streamline the complex processes involved in creating integrated circuits (ICs). EDA tools cover the entire design flow, from high-level system design and simulation to physical layout and verification, playing a crucial role in managing the intricacies of circuit design. Given the rapid advancement of semiconductor technology and the ever-increasing complexity of ICs, EDA tools are indispensable for ensuring efficiency, accuracy, and quality in modern VLSI projects. Without EDA tools, the detailed and iterative nature of IC design would be prohibitively time-consuming and error-prone.

A fundamental set of EDA tools supports design entry, synthesis, and simulation. Design entry tools, such as Cadence Virtuoso for analog design and Mentor Graphics HDL Designer for digital, allow engineers to input high-level functional designs using hardware description languages (HDLs) like Verilog or VHDL. Synthesis tools, like Synopsys Design Compiler, convert these high-level descriptions into gate-level netlists, effectively mapping logical functions to the target hardware. Simulation tools, such as Mentor Graphics ModelSim or Synopsys VCS, are then used to test the functionality of these designs in various scenarios, catching logical errors and verifying correctness before moving forward in the design flow. These tools are integral in establishing the functional and structural foundation of an IC.

EDA tools cover the entire design flow, from high-level system design and simulation to physical layout and verification, playing a crucial role in managing the intricacies of circuit design. parts within design blocks. Data bits build into digital words.

3.2 Basic Structure of an VLSI System

The following illustration shows the basic structure of an VLSIsystem –

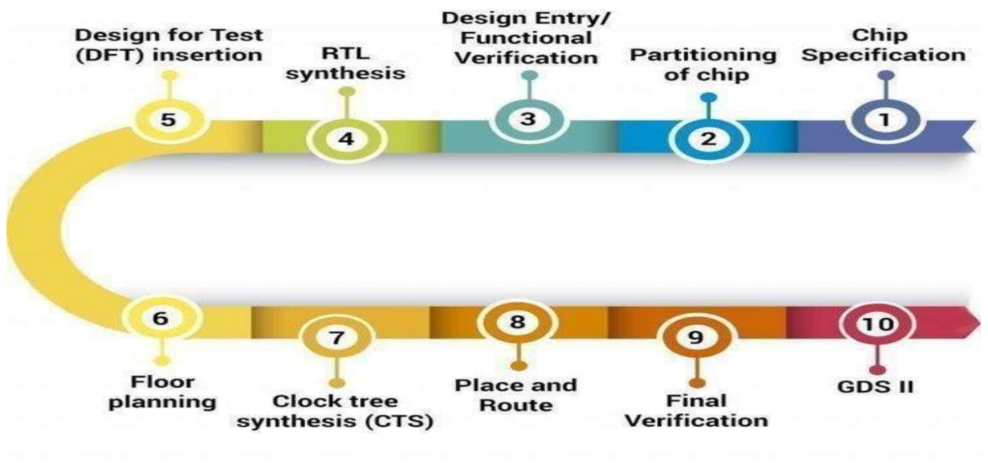


FIG:3.2 Basic VLSI Structure

Step 1. Chip Specification

This is the stage at which the engineer defines features, microarchitecture, functionalities (hardware/software interface), specifications (Time, Area, Power, Speed) with design guidelines of ASIC. Two different teams are involved at this juncture.

Step 2. Chip Partitioning

This is the stage wherein the engineer follows the ASIC design layout requirement and specification to create its structure using EDA tools and proven methodologies. This design structure is going to be verified with the help of HLL programming languages like C++ or System C.

After understanding the design specifications, the engineers partition the entire ASIC into multiple functional blocks (hierarchical modules), while keeping in mind ASIC's best performance, technical feasibility, and resource allocation in terms of area, power, cost and time. Once all the functional blocks are implemented in the architectural document, the engineers need to brainstorm ASIC design partitioning by reusing IPs from previous projects and procuring them from other parties. Design Compiler, convert these high-level descriptions into gate-level netlists, effectively mapping logical functions to the target hardware. such as Mentor Graphics ModelSim or Synopsys VCS, are then used to test the functionality of these designs in various scenarios, catching logical errors.

Step 3. Design Entry / Functional Verification

Functional verification confirms the functionality and logical behavior of the circuit by simulation on a design entry level. This is the stage where the design team and verification team come into the cycle where they generate RTL code using test- benches. In this simulation, once the RTL codeis done in HDL, a lot of code coverage metrics proposed for HDL. Engineers aim to verify the correctness of the code with the help of test vector sand trying to achieve it by95% coverage test. This code coverage includes statement coverage, expression coverage, branch coverage, and toggle coverage.

There are two types of simulation tools:

- Functional simulation tools
- Timing simulation tools

Step 4. RTL block synthesis / RTL Function

Once the RTL code and testbench are generated, the RTL team works on RTL description they translate the RTL code into a gate-level netlist using a logical synthesis tool that meets required timing constraints. Thereafter, a synthesized database of the ASIC design is created in the system. When timing constraints are met with the logic synthesis, the design proceeds to the design for testability (DFT) techniques.

Step 5. Design for Test (DFT) Insertion

With the ongoing trend of lower technology nodes, there is an increase in system-on- chip variations like size, threshold voltage and wire resistance. Due to these factors, new models and techniques are introduced to high-quality testing. ASIC design is complex enough at different stages of the design cycle.

Telling the customers that the chips have fault when you are already at the production stage is embarrassing and disruptive. It is the process of placing blocks in the chip. It includes block placement, design portioning, pin placement, and power optimization. This is the stage where the design team and verification team come into the cycle where they generate RTL code using test- benches. It’s a situation that no engineering team wants to be in. In order to overcome this situation, design for the test is introduced with a list of techniques:

- Scan path insertion
- Memory BIST (built-in Self-Test)
- ATPG (automatic test pattern generation)

Step 6. Floor Planning (blueprint your chip)

After, DFT, the physical implementation process is to be followed. In physical design, the first step in RTL-to-GDSII design is floor planning. It is the process of placing blocks in the chip. It includes block placement, design portioning, pin placement, and power optimization. Floor plan determines the size of the chip, places the gates and connects them with wires. While connecting, engineers take care of wire length, and functionality which will ensure signals will not interfere with nearby elements. In the end, simulate the final floor plan with the post-layout verification process.

A good floor planning exercise should come across and take care of the below points; otherwise, the life of IC and its cost will blow out:

- Minimize the total chip area
- Make routing phase easy (routable)
- Improve signal delays

Step 7. Clock tree synthesis

Clock tree synthesis is a process of building the clock tree and meeting the defined timing, area and power requirements. It helps in providing the clock connection to the clock pin of a sequential element in the required time and area, with low power consumption.

In order to avoid high power consumption, increase in delays and a huge number of transitions, certain structures can be used for optimizing CTS structure such as Mesh Structure, H-Tree Structure, X-Tree Structure, Fishbone Structure, and Hybrid structure.

While connecting, engineers take care of wire length, and functionality which will ensure signals will not interfere with nearby elements. In the end, simulate the final floor plan with the post-layout verification process. If this signal arrives at different parts of the chip at different times, it causes clock skew—a mismatch in timing that can lead to incorrect circuit operation. Additionally, long wire connections increase clock latency, which affects the speed of the design. CTS ensures that the clock is distributed evenly by using buffers and inverters to create a balanced tree structure. Very Large Scale Integration (VLSI) chip design. It is responsible for distributing the clock signal efficiently across the entire chip while minimizing clock skew and clock latency. Since the clock signal synchronizes all sequential elements (flip-flops) in a design, an optimal clock tree ensures proper timing, power efficiency, and overall performance of the circuit. It helps in providing the clock connection to the clock pin of a sequential element in the required time and area, with low power consumption.

Step 8. Place and Route

1. **Global Routing**: Calculates estimated values for each net by the delays of fan-out of wire. Global routing is mainly divided into line routing and mizw routing.
2. **Detailed Routing**: In detailed routing, the actual delays of wire is calculated by various optimization methods like timing optimization, clock tree synthesis, etc.

   As we are moving towards a lower technology node, engineers face complex design challenges with the need for implanting millions of gates in a small area. In order to make this ASIC design routable, placement density range needs to be followed for better QoR. Placement density analysis is an important parameter to get better outcomeswith less number of iterations.

Step 9. Final Verification (Physical Verification and Timing)

After routing, ASIC design layout undergoes three steps of physical verification, known as signoff checks. This stage helps to check whether the layout working the wayit was designed to. The following checks are followed to avoid any errors just beforethe tape out:

1. Layout versus schematic (LVS) is a process of checking that the geometry/layout matches the schematic/netlist.
2. Design rule checks (DRC) is the process of checking that the geometry in the GDS file follows the rules given by the foundry.
3. Logical equivalence checks (LVC) is the process of equivalence check between pre and post design layout.

Step 10. GDS II — Graphical Data Stream Information Interchange

In the last stage of the tape out, the engineer performs wafer processing, packaging, testing, verification and delivery to the physical IC. GDSII is the file produced and used by the semiconductor foundries to fabricate the silicon and handled to client. Info chips has contributed to over 500 product designs for top global companies, with more than 40 million deployed around the world. As a leading ASIC design and verification service provider, Info chips has brought together IP cores, verification IP and design and verification expertis. Developed originally by Calma in the 1970s, GDSII remains widely used in VLSI (Very Large Scale Integration) physical design for transferring chip layout information between Electronic Design Automation (EDA) tools, fabrication units, and mask-making processes. GDSII stores geometric shapes, layers, and hierarchical design structures that define the physical layout of an IC. It uses a binary format to efficiently store data, allowing designers to represent complex semiconductor designs with high precision.

CHAPTER 4
SOFTWARE REQUIREMENTS

4.1 SOFTWARE

Xilinx Software: (Over-wive of xilinx software)

In this chapter Xilinx software is a comprehensive suite of tools designed to support the entire FPGA (Field-Programmable Gate Array) and SoC (System on Chip) design flow. Xilinx provides powerful and integrated solutions for designing, simulating, synthesizing, implementing, and verifying hardware for a wide range of applications, from consumer electronics to automotive, telecommunications, and beyond. The main software tools from Xilinx are primarily used for FPGA development, offering engineers the ability to design complex digital systems and embedded applications with high efficiency.

The Vivado Design Suite is the flagship software tool from Xilinx and forms the core of the FPGA development process. It provides a complete environment for creating, testing, and deploying FPGA designs. Vivado supports various programming languages, including Verilog and VHDL, for hardware description and integrates powerful features like High-Level Synthesis (HLS), simulation, timing analysis, and debugging. Vivado offers an intuitive, user- friendly interface and advanced tools for synthesis, place-and-route, and bitstream generation, which are all critical stages in turning a digital design into a working FPGA system. Vivado also integrates the Xilinx IP Catalog, which contains pre-verified intellectual property cores, reducing development time by providing reusable and optimized components for tasks such as memory management, communication protocols, and signal processing.

For users focusing on embedded system design, the Xilinx Software Development Kit (SDK) provides the tools needed to develop software for Zynq SoCs and MicroBlaze processors, which are often integrated with FPGAs. The SDK enables the development of custom applications in C/C++ or even complex operating systems like Linux. It supports thecreation of device drivers, middleware, and application code, and offers debugging tools to optimize software running on the processor cores. The seamless integration between Vivadoand SDK makes it easier to handle both hardware and software aspects of a design within the same development environment.

Additionally, Xilinx ISE (Integrated Synthesis Environment), though largely replaced by Vivado, remains relevant for legacy designs. ISE is another comprehensive toolset for FPGA development, used primarily with older Xilinx devices. Xilinx Vivado Simulator and third-party simulators like ModelSim are essential for the simulation and verification of designs before they are implemented on hardware. These simulation tools ensure that designs behave as expected and help debug logic issues during the development phase.

The Xilinx software ecosystem also includes advanced debugging tools like Integrated Logic Analyzer (ILA) and Virtual Input/Output (VIO), which allow for in-depth analysis and real- time monitoring of internal FPGA signals during operation. These tools help identify design flaws and optimize the system's performance.

Overall, Xilinx software is a powerful, integrated solution for FPGA and embedded system development, offering a complete set of tools for everything from high-level synthesis and hardware design to embedded software development and debugging. Whether for advanced hardware design or embedded system integration, Xilinx software enables engineers to efficiently develop robust and high-performance digital systems.

1. Vivado Design

The Vivado Design Suite is Xilinx's flagship integrated development environment (IDE) for designing, simulating, synthesizing, implementing, and debugging FPGA and SoC(System on Chip) designs. It is a comprehensive platform designed to enable engineers to efficiently create high-performance, low-power digital systems on Xilinx FPGAs and Zynq SoCs. Vivado combines several essential tools into one unified package, providing a streamlined workflow from design conception to hardware implementation.

Vivado provides support for both RTL (Register Transfer Level) design and high-level synthesis (HLS). It allows users to write hardware descriptions in Verilog or VHDL and provides a range of tools for simulation, timing analysis, and verification. The suite includes an HDL editor, a block design environment, and an IP integrator, enabling designers to build complex systems using pre-configured IP cores from the extensive XilinxIP Catalog. This catalog contains highly optimized, reusable blocks for tasks like memory management, digital signal processing, and communication protocols. ISE is another comprehensive toolset for FPGA development, used primarily with older Xilinx devices. Xilinx Vivado Simulator and third- party simulators like Model Sim are essentia.

2. Simulation Tools

Simulation software is a crucial component in verifying the functionality of the Verilog design before moving to hardware implementation.

- **ModelSim (by Mentor Graphics)**:

A popular simulator for Verilog and VHDL. It supports both functional and timing simulations, enabling the designer to test the behavior of the Verilog code before hardware synthesis. ModelSim is widely used in FPGA and ASIC designs.

- **Synopsys VCS**:

This is another high-performance simulator for Verilog and SystemVerilog, used for functional verification and ensuring that the design meets the intended specifications.

- **Xilinx ISIM**:

Xilinx's Integrated Simulation Environment (ISIM) provides simulation for Verilog designs targeting Xilinx FPGAs. It helps verify the functionality of a car parking system implemented in Verilog on a Xilinx FPGA.

- **Cadence NC-Sim**:

A comprehensive simulator that supports Verilog, VHDL, and SystemVerilog for advanced functional verification, often used in ASIC and FPGA designs.

Synthesis Tools

Synthesis tools are used to convert Verilog code (typically at a higher abstraction level) into a gate-level netlist, which can be used for further physical design steps. Synthesis optimizes the design for area, power, and timing.

Xilinx Vivado:

For designs targeting Xilinx FPGAs, Vivado includes both synthesis and implementation tools. It can take Verilog HDL code and generate the necessary bitstream file for the FPGA.

Intel Quartus:

For Intel (formerly Altera) FPGAs, Quartus provides synthesis, place-and- route, and bitstream generation. It supports Verilog HDL and can generate the configuration files required to implement the parking system design on Intel FPGAs.

Synopsys Design Compiler:

This is a leading tool for synthesis and optimization of Verilog designs for ASIC implementations. It ensures that the Verilog code meets the target design constraints.

3. Hardware Description and Debugging Tools

Writing Verilog code and performing functional simulation, debugging and verifying the hardware's functionality is essential to ensure correct operation. Tools that allow for waveform analysis and signal tracking are essential in debugging the design.

- **GTKWave**:

This open-source waveform viewer can be used to visualize simulation results. It helps to analyze the signals generated during the simulation of the Verilog code and verify that the logic is functioning correctly.

- **Waveform Viewer in ModelSim/Synopsys VCS**:

Both ModelSim and VCS come with built- in waveform viewers, where simulation results can be viewed in a graphical manner, making it easier to analyze the behavior of the system and troubleshoot any issues in the design.

In FPGA-based designs, Verilog and VHDL are the two primary HDLs used to define the system's logic. These languages allow the designer to specify how the system reacts to inputs and how it performs the required computations or operations. For example, a digital filter or an automatic car parking system can be implemented using these languages, whereevery component of the system—such as counters, registers, logic gates, and control units—can be described at a granular level. After creating the HDL code, the next step is tosimulate, verify, and implement it on hardware, using specific software tools.

Debugging is a critical aspect of FPGA design, ensuring that the system behaves as intended before it is deployed into real-world applications. Hardware debugging tools are essential for analyzing the design's behavior in real-time, as they provide insight into how the FPGA's logic is executing and where potential issues or performance bottlenecks might arise. Debugging tools help identify functional errors, timing violations, or incorrect hardware behavior that may not be apparent during the simulation stage.In addition to cell displacements, a cell could attach in the precise location, but not align in the same direction as the other cells. This would give the appearance of a cell being rotated around its center .

Similar to displacement defects, rotations can be directly modeled. However, it is possible that a cell rotation may misplace a neighboring cell. This possibility is implementation specific because the cell to cell distance may be larger than one-half of the cell width or height for two dot cells or of the cell's diagonal for four dot cells.

Brief History of FPGA Simulators

Xilinx has been a leading provider of FPGA (Field-Programmable Gate Array) development tools for decades. Its history with simulators reflects the evolution of FPGA design itself, from early, basic simulation capabilities to the advanced, integrated simulation tools available today. Early in its development, Xilinx's focus was on creating tools that would allow engineers to model and test digital designs in a virtual environment before hardware implementation. These early simulators were essential for verifying the behavior of designs and ensuring correctness before physical testing.

The first Xilinx simulators were basic tools that supported simulation of HDL (Hardware Description Languages) such as VHDL and Verilog. They were not as powerful or integrated as today's tools but played a pivotal role in the growth of FPGA technology by providing a way to verify designs in a more efficient and cost-effective manner than traditional hardware testing. Over time, Xilinx improved its simulation tools to better support the growing complexity of FPGA designs, which required more sophisticated analysis and debugging features.

In the early 2000s, Xilinx began to integrate more advanced simulation tools into its development environment, particularly with the release of the ISE Design Suite (Integrated Synthesis Environment). This suite included both synthesis and simulation capabilities, offering engineers a more complete solution for design and verification. At this time, the company also started to integrate third-party simulators like ModelSim from Mentor Graphics, which allowed for a higher level of simulation accuracy and performance. This integration helped streamline the verification process for complex designs, enabling faster development cycles and more reliable designs.

With the advent of the Vivado Design Suite in 2012, Xilinx significantly enhanced its simulation capabilities. Vivado incorporated the Vivado Simulator, a more powerful and integrated simulation tool designed to support Verilog, VHDL, and SystemVerilog designs. Vivado Simulator offered advanced debugging features, such as waveform viewers, breakpoints, and signal tracing, that allowed engineers to simulate and debug designs more effectively. Vivado's ability to integrate simulation directly into the development workflow made it a critical tool for modern FPGA design, supporting not only hardware verification but also facilitating system-level verification and timing analysis. Debugging tools help identify functional errors, timing violations, or incorrect hardware behavior that may not be apparent during the simulation stage. In addition to cell displacements, a cell could attach in the precise location, but not align in the same.

In addition to Vivado's native simulator, Xilinx continued to support third-party simulators like ModelSim, as well as tools like Cadence’s Incisive and Synopsys VCS, ensuring compatibility with industry-standard simulation environments. This flexibility allowed Xilinx users to choose the best simulator for their specific design requirements, providing options for high- performance simulation, more extensive debugging, or better support for specific FPGA families.

Today, Xilinx simulators are an integral part of the FPGA design process, with tools like Vivado Simulator, ModelSim, and other third-party solutions providing engineers with powerful capabilities for functional and timing verification. The history of Xilinx simulators is a testament to the company’s commitment to innovation and meeting the ever-increasing demands of the FPGA design industry. These simulators enable faster, more reliable design cycles, making FPGA technology more accessible and efficient for developers across various industries.

4.2 Working Principle

The proposed smart car parking system into following three modules.

- Car entering and exiting module.
- Distance measurement to find the obstacle

4.2 Car Entering and Exiting Module

Entering Module

In Entering Module it is sensed by the IR Sensors when the car enters the lot. The IR Sensors give the FPGA the pulse which considers an input to be detected. The vehicle is allowed into the parking lot only if the password is entered. If the entered password is correct then the vehicle is preceded to park or else the gate will remain closed.

Exiting Module

In Exiting Module it is detected by the IR Sensors as the vehicle moves out of the lot. The IR Sensors provide the pulse to the FPGA which assumes that an input is detected and that the car is only exited from the parking lot after the password has been entered correctly. And the gate is closed when the next car tries to exit the lot.

Distance measurement to find the obstacles

Distance and speed monitoring devices are used in many applications such as vehicles, protection etc. The system can calculate the time interval between two laser pulses, a machine-sent reference pulse and an echo pulse reflected back to the device. The machine determines the distance to the object on which the echo-pulse is reflected with the time information. The view of this project to the machine level. The vehicle is allowed into the parking lot only if the password is entered. If the entered password is correct then the vehicle is preceded to park or else the gate will remain closed.

The Plan has two stages. In the first phase, target hardware is a Xilinx FPGA. In the second phase, target hardware is an integrated circuit (ASIC) application-specific solution where VHDL and standard cells are used. The design process of the FPGA framework is further broken down into three subphases. The first step is intended to only produce a software product, where the whole system is written in C language for the Xilinx Micro Blaze soft processor system . The second is a mixed implementation of hardware and software, where one part of the device is implemented in C and the other part is implemented in hardware.

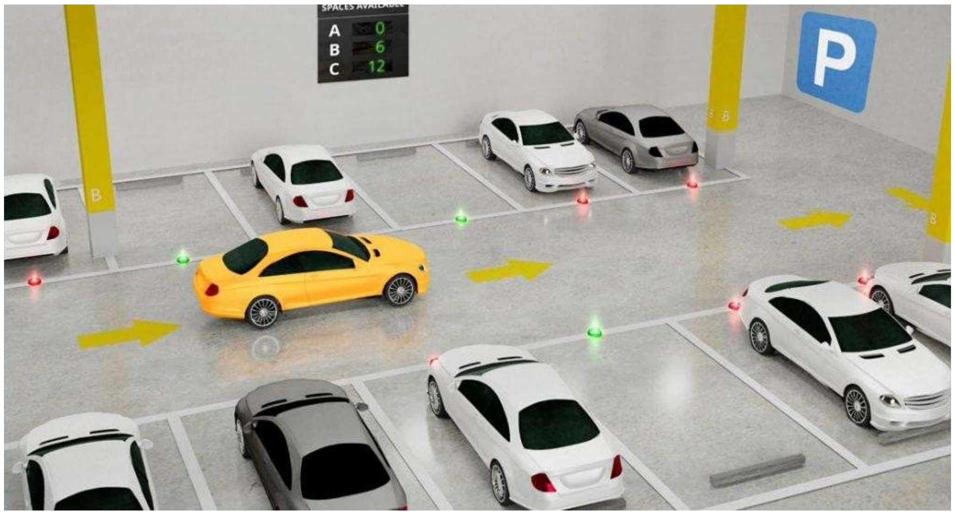


Fig: 4.1 Smart parking System

The hardware is constructed using the terminology used to characterize VHDL. The third and final product on Xilinx XUP Vertex-II Pro Development System is to be completed in hardware and implemented. In VHDL a test vector generator emulating the "Source" and "Echo" laser signal pulses is designed and called "AD model." 256 samples should be included in each measurement. The vehicle is allowed into the parking lot only if the password is entered. If the entered password is correct then the vehicle is preceded to park or else the gate will remain closed. where the whole system is written in C language for the Xilinx Micro Blaze soft processor system.

CHAPTER 5
WORKING MODEL AND COMPONENTS

5.1 BLOCK DIAGRAM

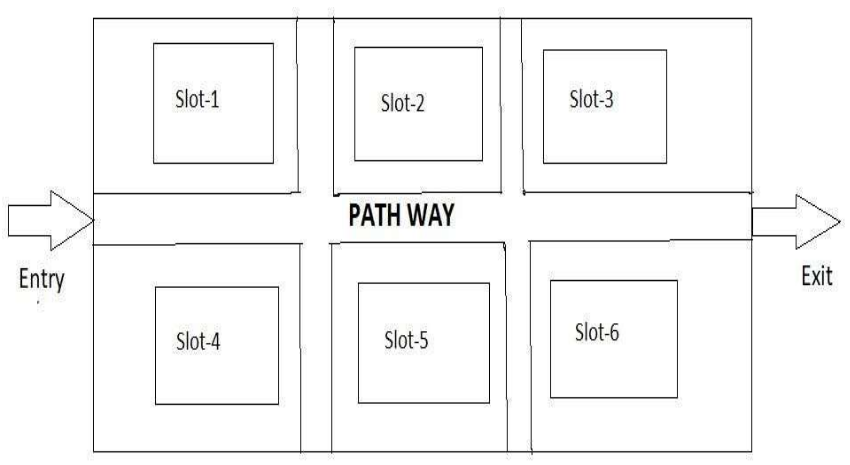


FIG : 5.1 Block Diagram Of Car Parking By Using Slot System

The implementation of an automatic car parking system using Verilog HDL involves designing a digital system that efficiently manages the entry, exit, and allocation of parking spaces. The system relies on sensors to detect the presence of vehicles and a control unit to handle decision-making processes. The primary objective is to automate the parking process, reducing human intervention and optimizing space utilization. The implementation is done at the register transfer level (RTL), where different modules interact to form a cohesive parking management system.

The system consists of key components such as entry and exit gates, parking slot sensors, a counter to keep track of available slots, and a control logic unit. The entry gate opens when a vehicle is detected at the entrance, provided there is an available parking slot. The system updates the slot count and assigns a parking space to the incoming vehicle. Similarly, when a vehicle exits, the slot count is incremented, and the exit gate operates accordingly.

The Verilog code consists of modules defining the control logic, counter logic, and interface with external components. Simulation and testing are performed using tools like ModelSim or Xilinx Vivado to verify the functionality and timing of the system before deployment on hardware such as an FPGA. This approach ensures an efficient and reliable automatic parking system suitable for modern smart parking applications.

The control unit processes the input signals and updates the parking slot status accordingly. A counter keeps track of the number of available slots and displays the information on an LED or LCD screen. If a vacant slot is available, the system triggers the gate to open, allowing vehicle entry. Once the vehicle is parked, the system updates the slot availability status in real time.

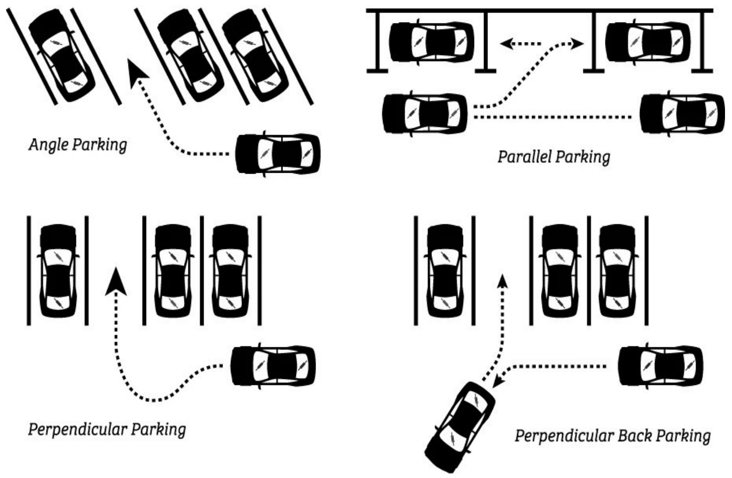


FIG:5.2 Different Types Of Parking Stages

Upon vehicle exit, the sensor at the exit gate detects the departing car and sends a signal to the control unit, decrementing the occupied slot count. The gate opens automatically, and the slot is marked as available. The entire design is simulated and verified using VIGADO software, which provides an environment for developing and testing Verilog-based digital circuits. The implementation ensures efficient parking management by minimizing manual .

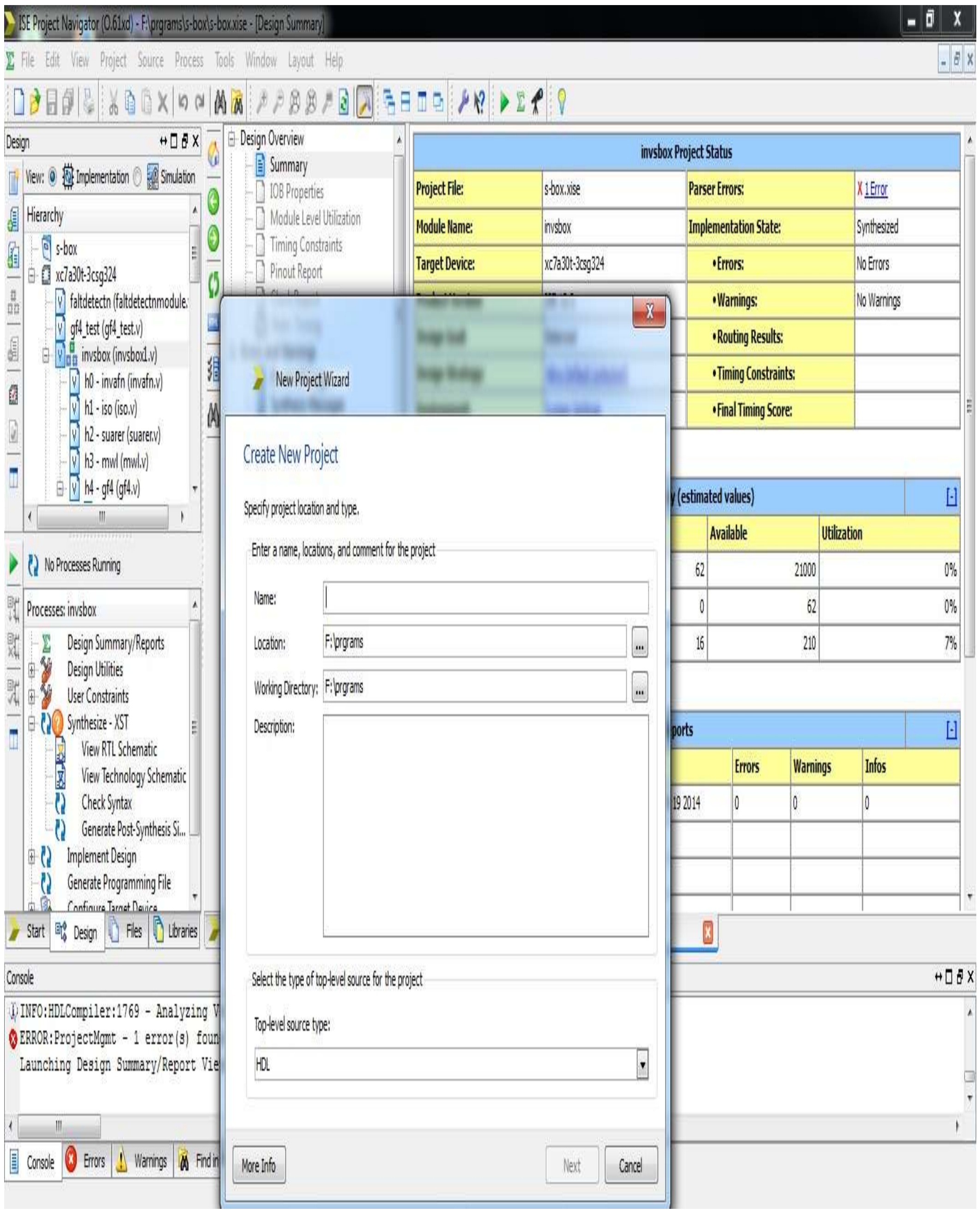
The gate control unit is responsible for opening and closing the entry and exit gates based on the sensor inputs, ensuring smooth vehicle movement. For security purposes, an alarm system can be included to notify drivers when the parking lot is full. The controller checks the parking slot counter to determine if there is an available slot. If space is available, the counter is incremented, the entry gate opens, and the display is updated. If the parking lot is full, the system triggers an alarm or a "FULL" message on the display to prevent additional vehicles from entering. Similarly, when a car exits, the exit sensor detects it, the counter decrements, and the exit gate closes once the vehicle has left. This ensures an accurate count of parked vehicles at all times. A counter keeps track of the number of available slots and displays the information on an LED or LCD screen.

5.2 PROCEDURE XILINX SOFTWARE :

- Click project navigator
- Create new project
- Selection of FPGA
- Create new source code
- Select source type (Verilog module)
- Coding
- Declaration of inputs and output
- Sources for implementation
- Check syntax
- View design summary
- View RTL schematic
- View technology schematic
- Sources for behavioural simulation
- Create a new source code
- Select source type(Verilog text fixture)
- Write test bench code
- Xilinx ISE simulator
- Behavioural check syntax
- Simulate behavioural model

5.3 PROCEDURE FOR SYNTHESIS:

STEP1: To create new project in xilinx we should open the filemenu ,click on new project then it will open the dialogbox as below in that typethe filename click on next.



ISE Project Navigator (O.61xd) - F:\prgrams\s-box\s-box.xise - [Design Summary]

invsbox Project Status			
Project File:	s-box.xise	Parser Errors:	X 1 Error
Module Name:	invsbox	Implementation State:	Synthesized
Target Device:	xc7a30t-3csg324	•Errors:	No Errors
		•Warnings:	No Warnings
		•Routing Results:	
		•Timing Constraints:	
		•Final Timing Score:	

Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name:

Location: F:\prgrams

Working Directory: F:\prgrams

Description:

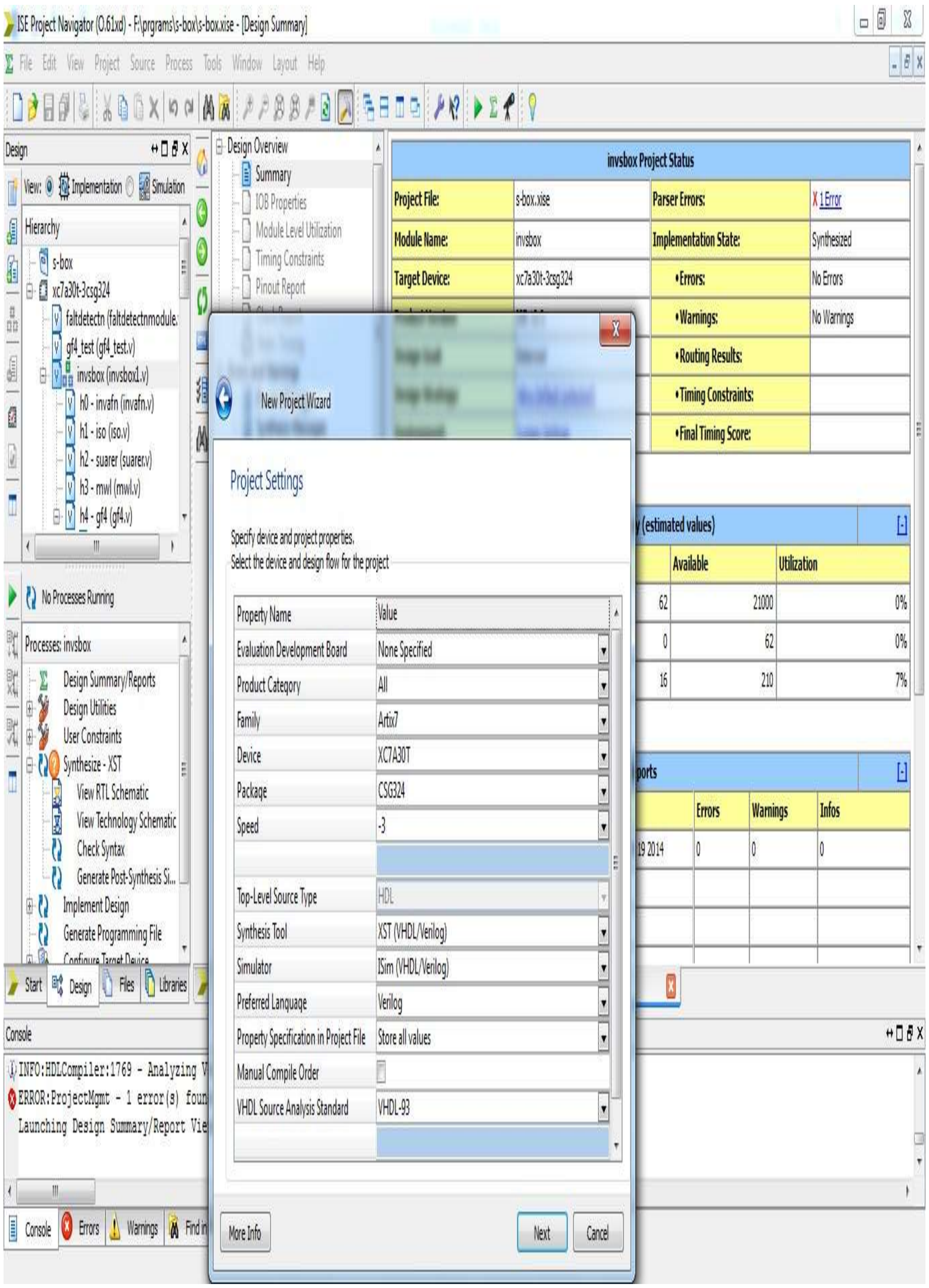
Select the type of top-level source for the project

Top-level source type: HDL

```
INFO:HDLCompiler:1769 - Analyzing V
ERROR:ProjectMgmt - 1 error(s) foun
Launching Design Summary/Report Vie
```

FIG:5.3 Creating a New File In Xilinx

STEP2: Then it Displays one more dialogbox which will give us the specifications of the project,click on next.



Project File:	s-box.xise	Parser Errors:	X 1 Error
Module Name:	invsbox	Implementation State:	Synthesized
Target Device:	xc7a30t-3csg324	•Errors:	No Errors
		•Warnings:	No Warnings
		•Routing Results:	
		•Timing Constraints:	
		•Final Timing Score:	

```
INFO:HDLCompiler:1769 - Analyzing V
ERROR:ProjectMgmt - 1 error(s) foun
Launching Design Summary/Report Vie
```

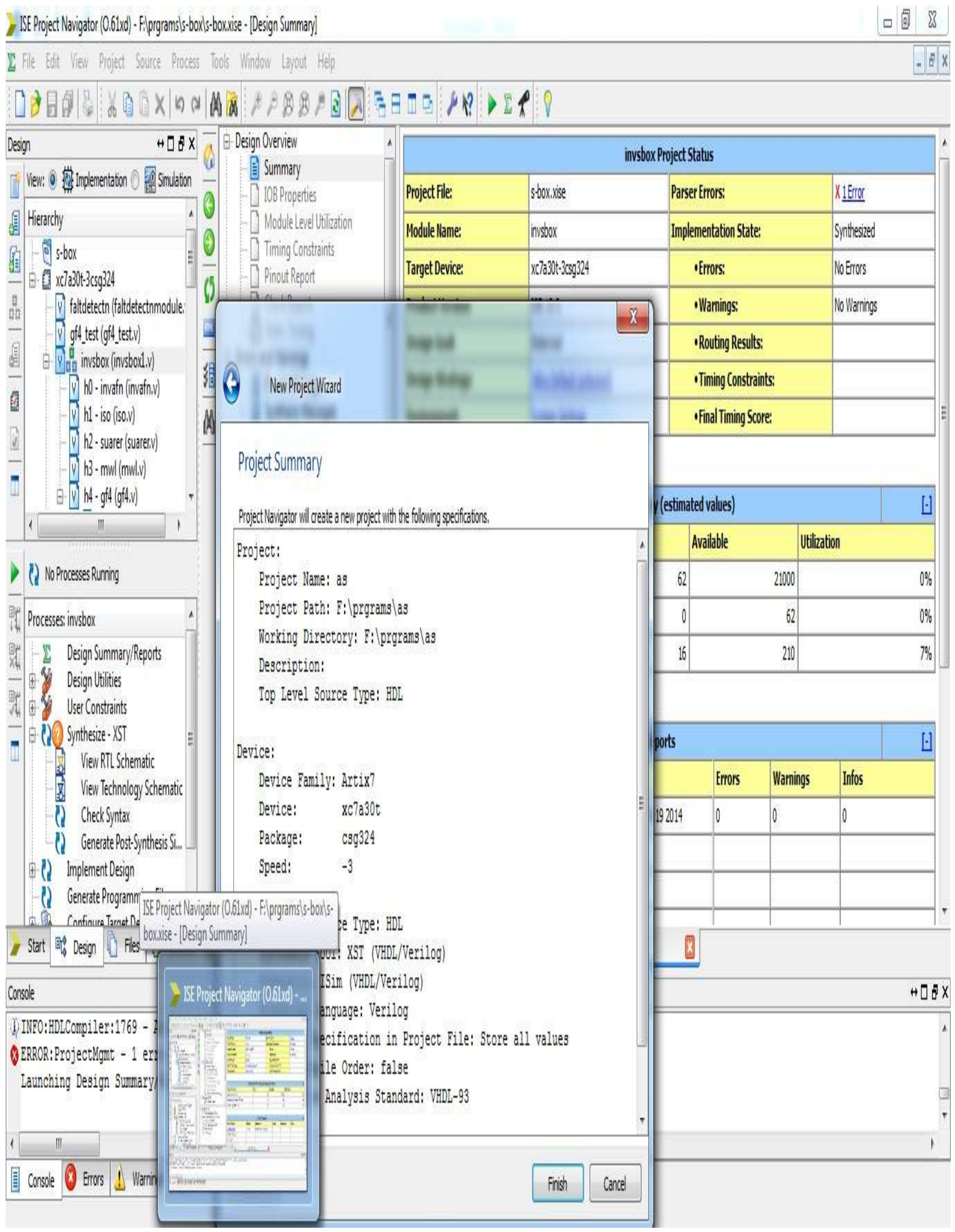
Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Artix7
Device	XC7A30T
Package	CSG324
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93

FIG:5.4 Displays One More Dialogbox For Specifications In Xilinx

STEP3: Then it again displays a dialogue box as shown below with the created project description and click finish to compelte the process of creating new project.


ISE Project Navigator (O.61xd) - F:\prgrams\s-box\s-box.xise - [Design Summary]

invsbox Project Status			
Project File:	s-box.xise	Parser Errors:	X 1 Error
Module Name:	invsbox	Implementation State:	Synthesized
Target Device:	xc7a30t-3csg324	•Errors:	No Errors
		•Warnings:	No Warnings
		•Routing Results:	
		•Timing Constraints:	
		•Final Timing Score:	

New Project Wizard

Project Summary

Project Navigator will create a new project with the following specifications.

```
Project:
   Project Name: as
   Project Path: F:\prgrams\as
   Working Directory: F:\prgrams\as
   Description:
   Top Level Source Type: HDL

Device:
   Device Family: Artix7
   Device:      xc7a30t
   Package:     csg324
   Speed:       -3
```

```
INFO:HDLCompiler:1769 -
ERROR:ProjectMgmt - 1 err
Launching Design Summary
```

FIG:5.5 Displays Again a Dialogue Box As Shown Below With The Created Project Description In Xilinx

STEP4: Now project with specifyed name is created then create the verilog files in the project. To create filesr, right click on the project that will show options like as shown below.

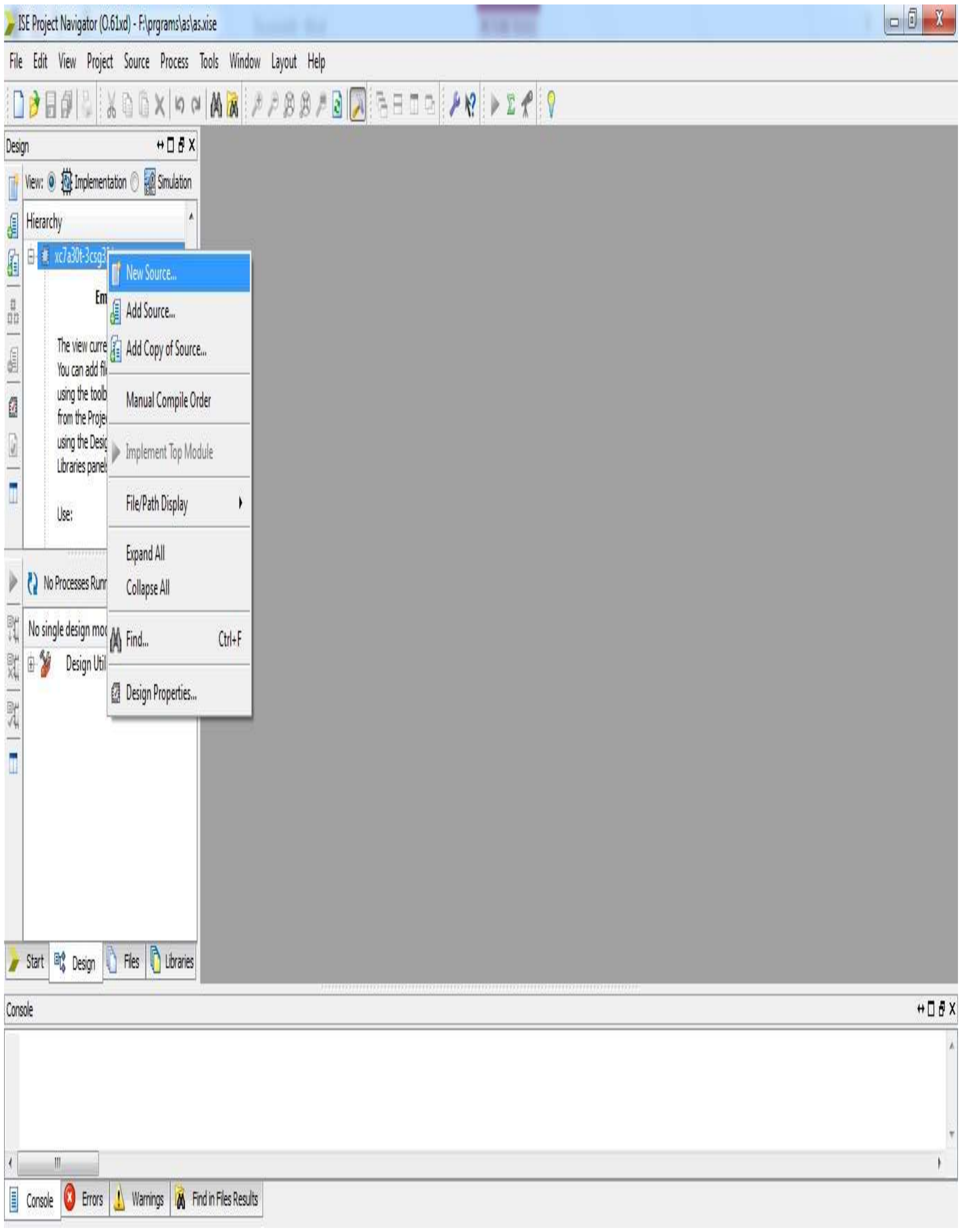
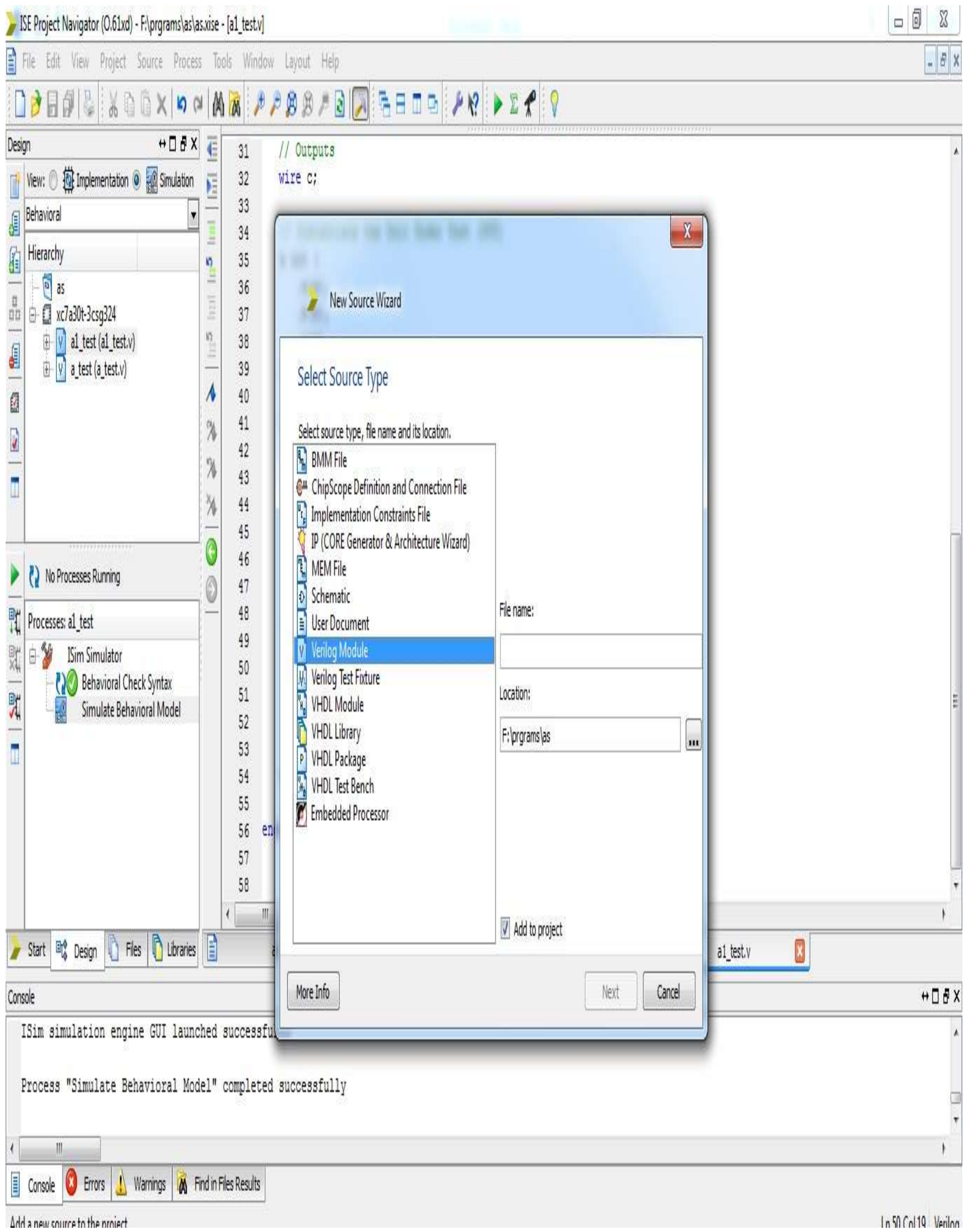


FIG:5.6 Specifyed Name Is Created Then Create The Verilog Files In The Project In Xilinx

STEP5: From the given options select new source then it diaplays dialogbox which is containing of list of fileformat now we want to create verlogfile so select veilog module, and give the name to the file. Then click on next.

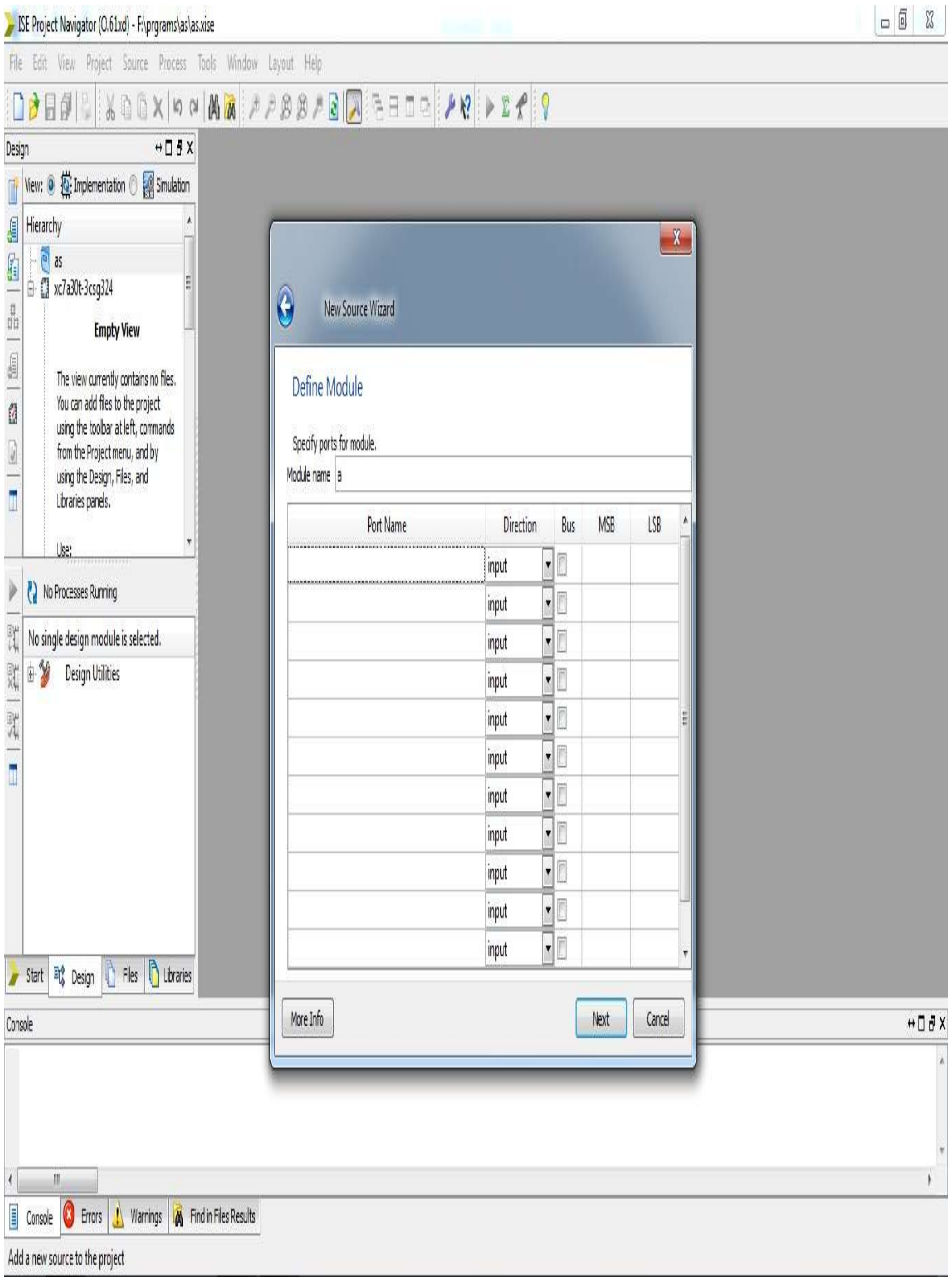


```
// Outputs
wire c;
```

```
ISim simulation engine GUI launched successfu

Process "Simulate Behavioral Model" completed successfully
```

FIG:5.7 Select New Source Then It Diaplays Dialogbox Which Is Containing Of List Of File Format In Xilinx

STEP 6: Then it will ask us to select inputs,outputs and inouts. We can specify our inputs and outputs here else we may also specify as part of programme depend upon the user requirement, click on next.


ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise

New Source Wizard

Define Module

Specify ports for module.

Module name a

Port Name	Direction	Bus	MSB	LSB
	input			
	input			
	input			
	input			
	input			
	input			
	input			
	input			
	input			
	input			
	input			

Empty View

The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.

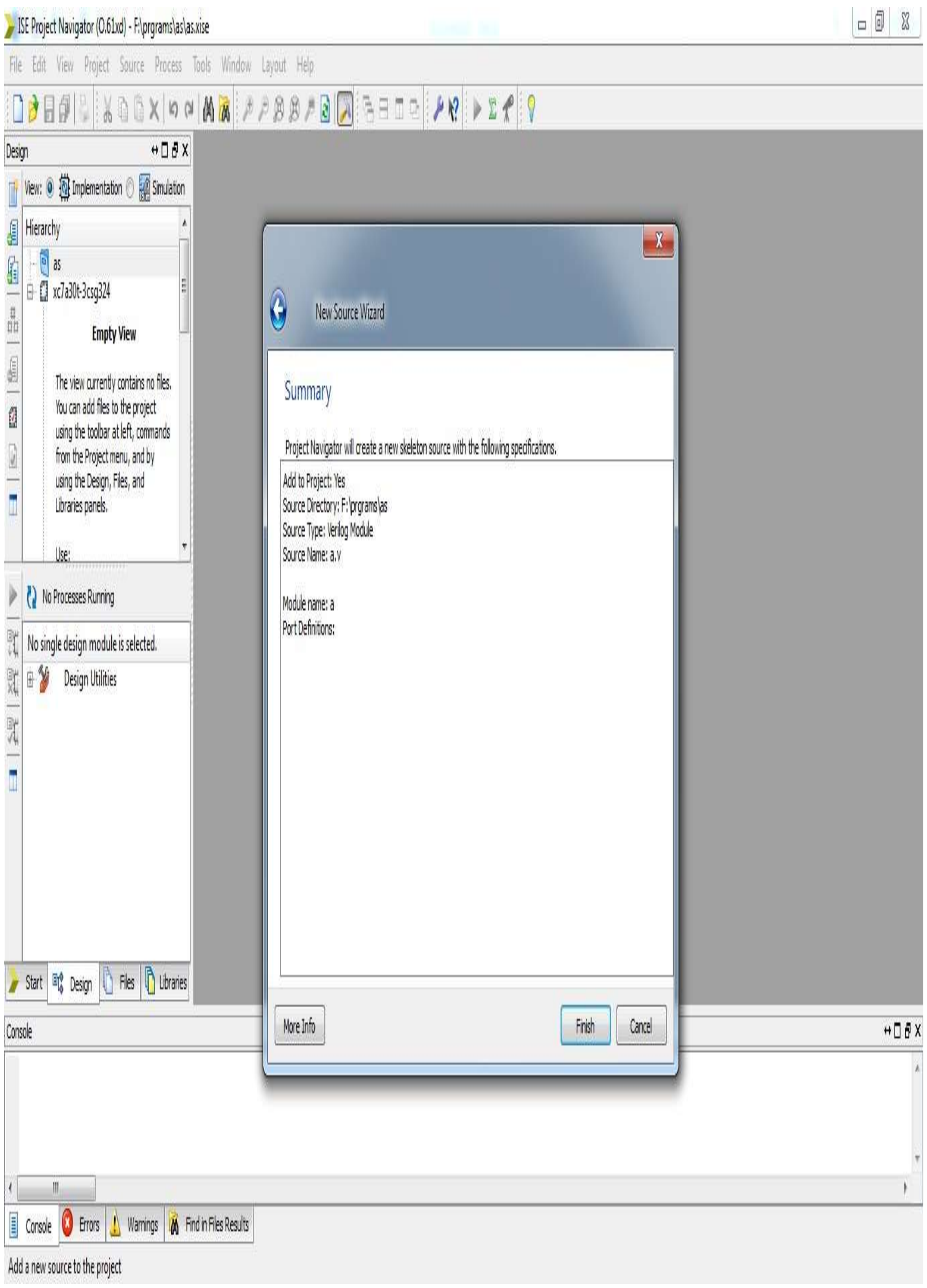
No Processes Running

No single design module is selected.

Add a new source to the project

FIG:5.8 Select Inputs ,Outputs And Inouts In Xilinx

STEP 7: It will again displays a dilagbox by fiving details of filename etc, click on next.


ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
as
xc7a30t-3csg324
Empty View
The view currently contains no files. You can add files to the project using the toolbar at left, commands from the Project menu, and by using the Design, Files, and Libraries panels.
Use:
No Processes Running
No single design module is selected.
Design Utilities
Start Design Files Libraries
Console
Console Errors Warnings Find in Files Results
Add a new source to the project
New Source Wizard
Summary
More Info
Finish
Cancel

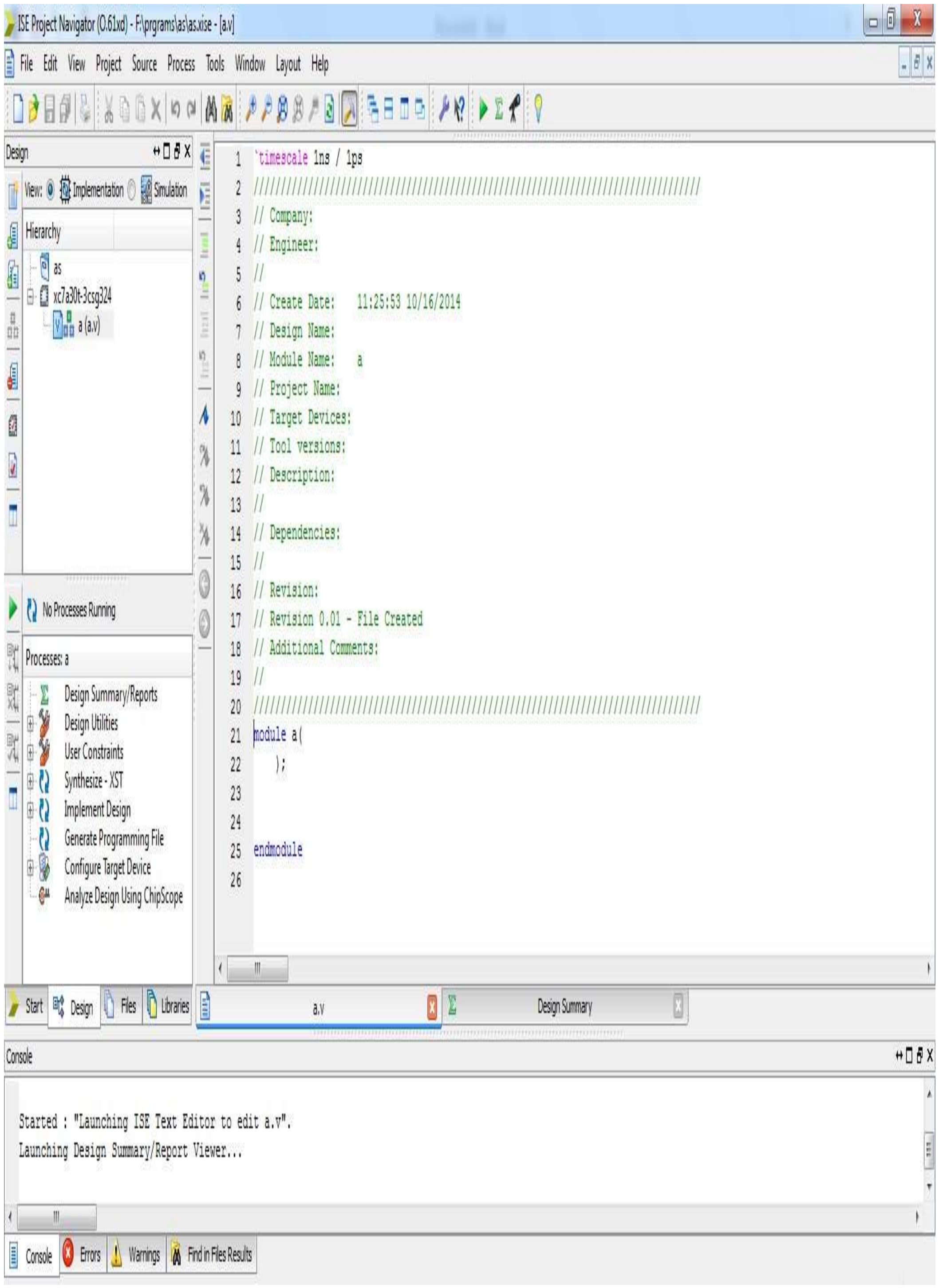
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: F:\prgrams\as
Source Type: Verilog Module
Source Name: a.v

Module name: a
Port Definitions:

FIG:5.9 Displays Again A Dilagbox By Fiving Details Of Filename In Xilinx

STEP 8: It will open a white space in the project window containing filename the double click on the file name so that it will displays respective file window ,where we should write the code.

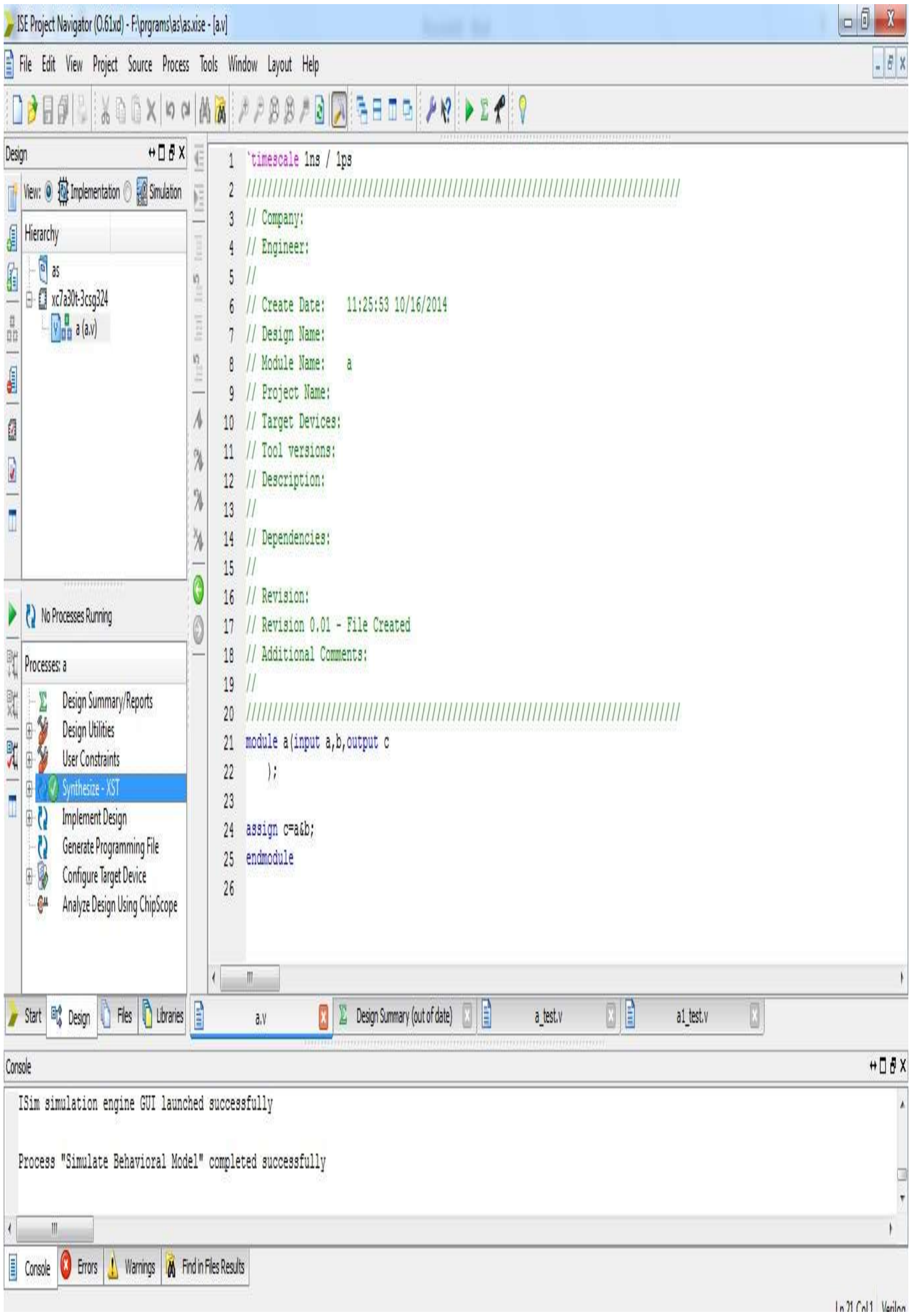


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    11:25:53 10/16/2014
// Design Name:
// Module Name:    a
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module a(
    );


endmodule
```

```
Started : "Launching ISE Text Editor to edit a.v".
Launching Design Summary/Report Viewer...
```

Fig:5.10 Open A White Space In The Project Window Containing Filename In Xilinx

STEP 9: After completion writing code select the file name and click on synthesis which will check for errors, if there are any errors in syntax or design errors are checked and shown in the below of file window.

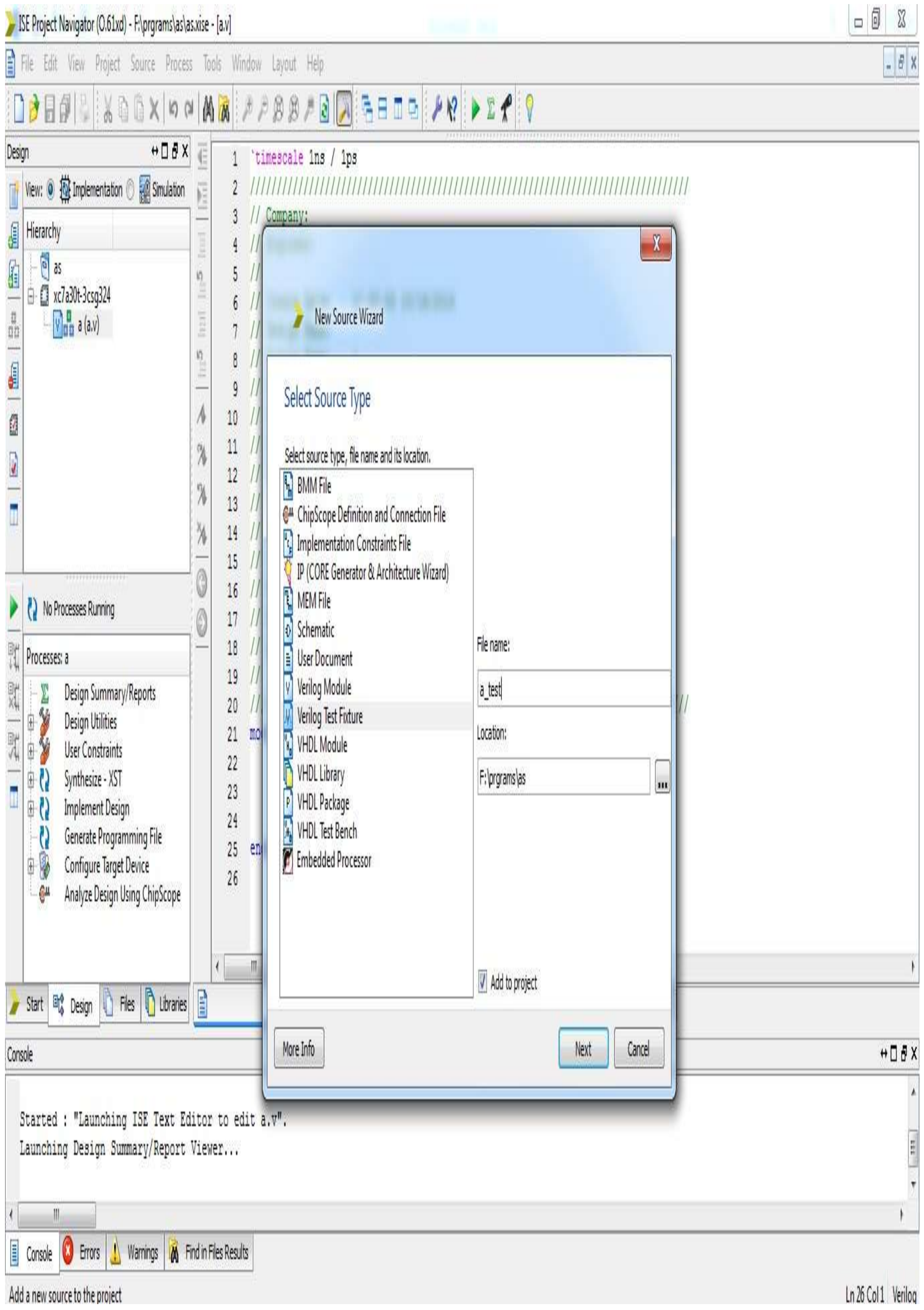

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
as
xc7a30t-3csg324
a (a.v)
No Processes Running
Processes: a
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
a.v
Design Summary (out of date)
a_test.v
a1_test.v
Console
Console Errors Warnings Find in Files Results
Ln 21 Col 1 Verilog

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    11:25:53 10/16/2014
// Design Name:
// Module Name:    a
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module a(input a,b,output c
    );

assign c=a&b;
endmodule
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

FIG:5.11 Writing Code Select The File Name In Xilinx

STEP10: After sucessful synthesis we should have to create tesh bench file with extension as test,for that again right click on the file name as shown below,give filename.



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
```

Select source type, file name and its location.

File name: a_test

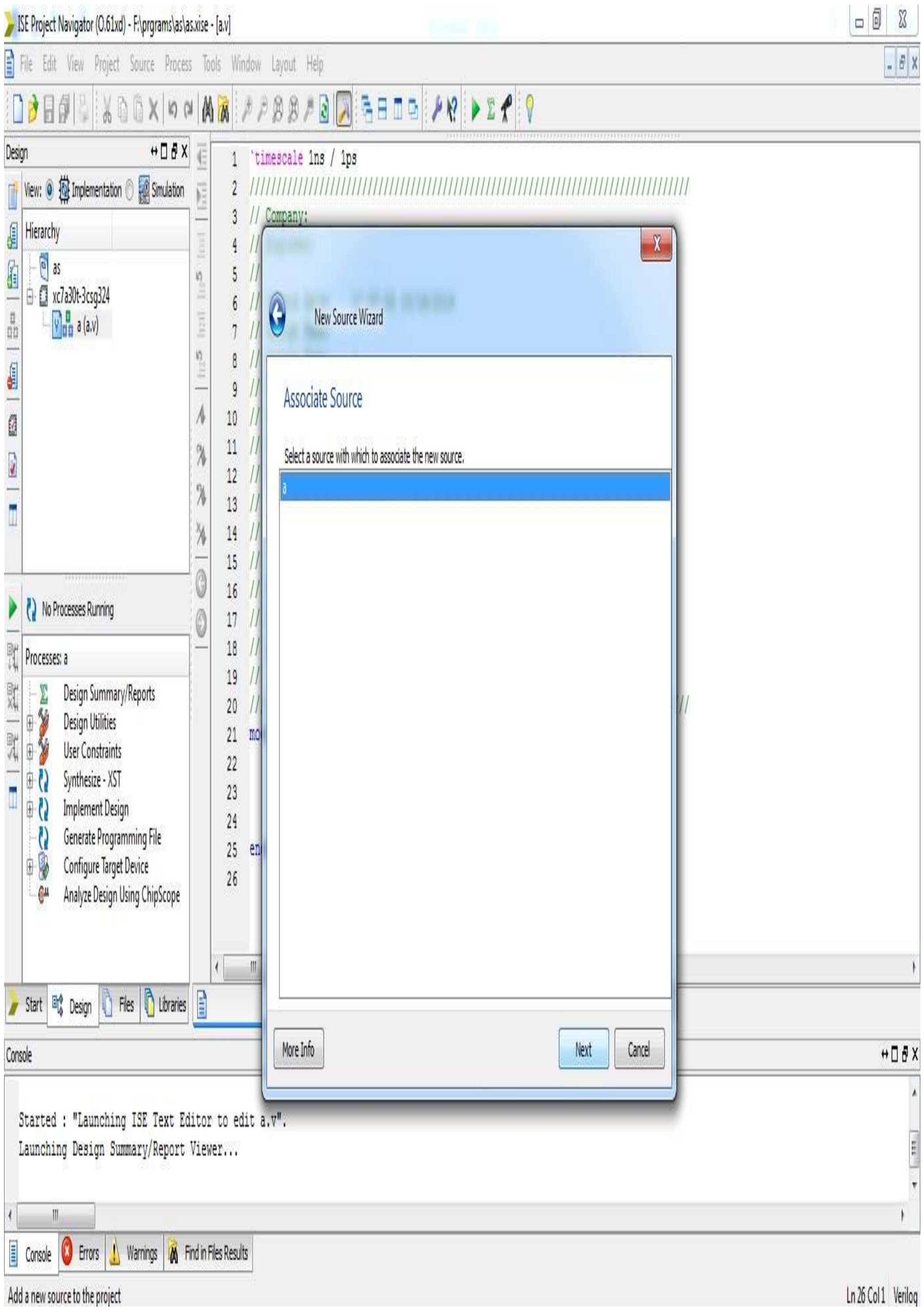
Location: F:\prgrams\as

Add to project [✓]

```
Started : "Launching ISE Text Editor to edit a.v".
Launching Design Summary/Report Viewer...
```

FIG:5.12 After Sucessful Synthesis We Should Have To Create Test Bench File In Xilinx

STEP11: If there are list files then select file for which we are creating the test bench. Click on next.

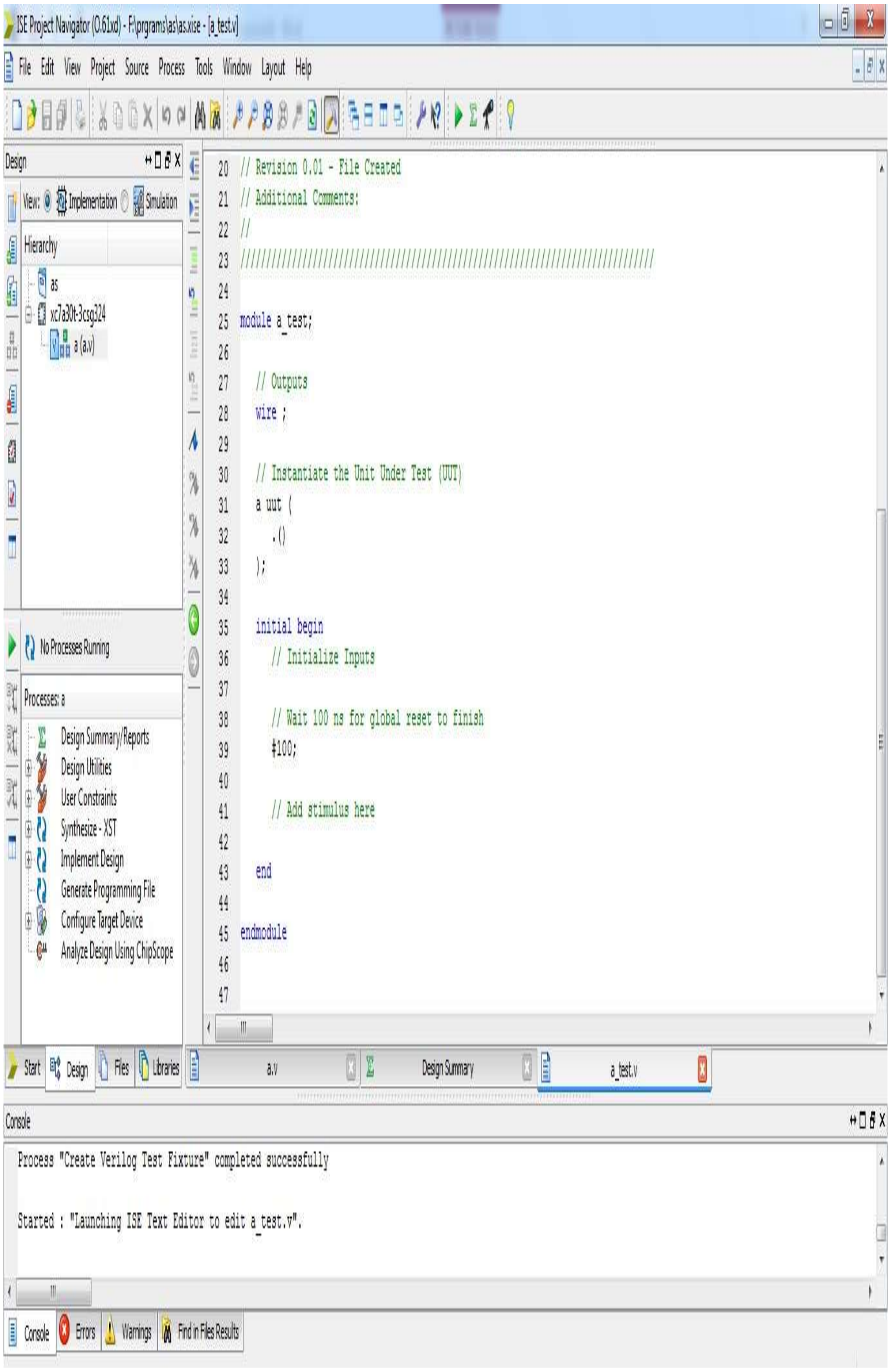

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
as
xc7a30t-3csg324
a (a.v)
No Processes Running
Processes: a
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
New Source Wizard
Associate Source
Select a source with which to associate the new source.
a
More Info
Next
Cancel
Console
Console Errors Warnings Find in Files Results
Add a new source to the project
Ln 26 Col 1 Verilog

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////
// Company:
```

```
Started : "Launching ISE Text Editor to edit a.v".
Launching Design Summary/Report Viewer...
```

FIG:5.13 List Files Then Select File For Which We Are Creating The Test Bench In Xilinx

STEP 12: It again gives a testbench file in the project window, then give reqired inputs.

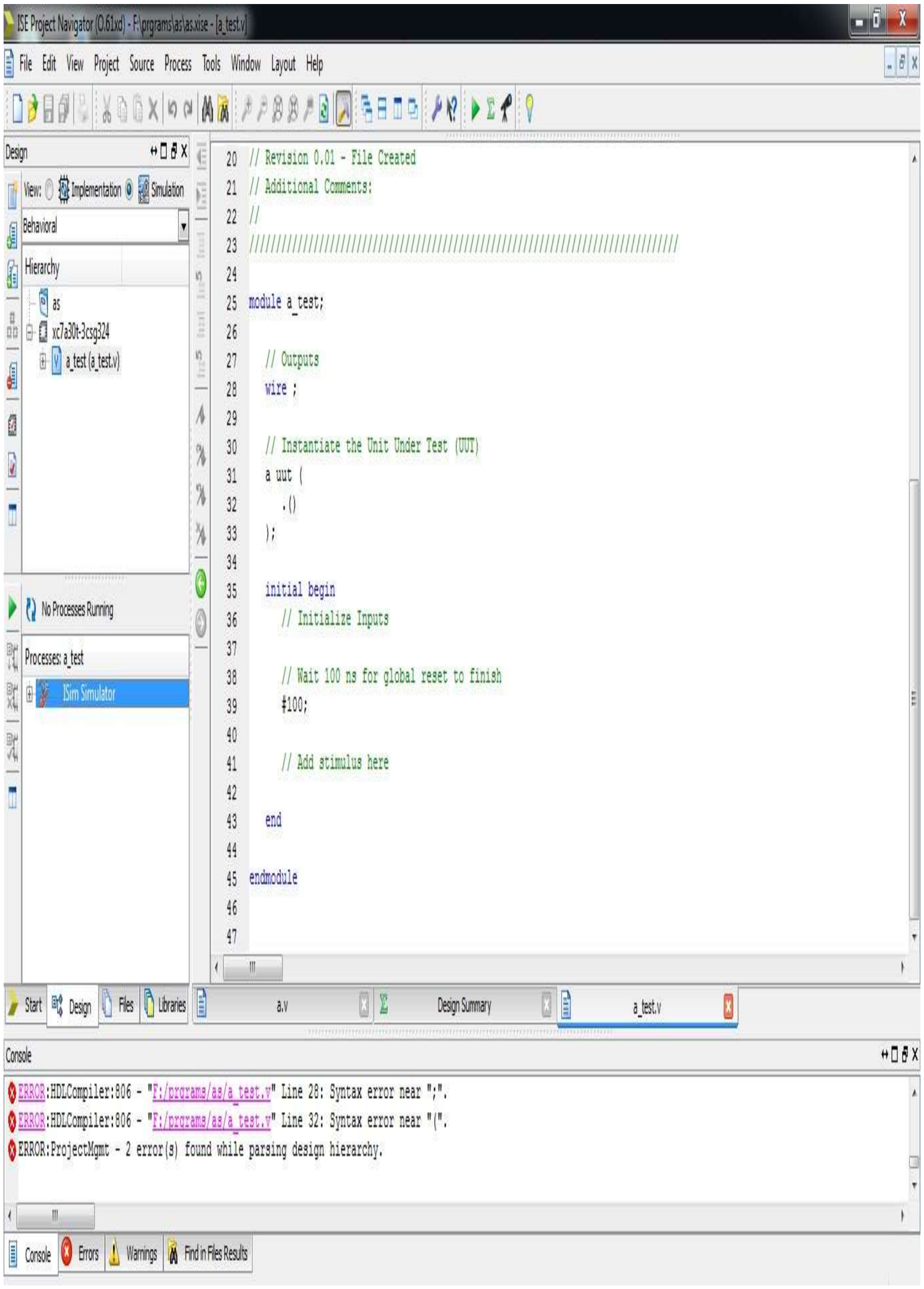

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a_test.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
as
xc7a30t-3csg324
a (a.v)
No Processes Running
Processes: a
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
Start Design Files Libraries
a.v Design Summary a_test.v
Console
Console Errors Warnings Find in Files Results

```
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module a_test;

  // Outputs
  wire ;

  // Instantiate the Unit Under Test (UUT)
  a uut (
     .()
  );

  initial begin
    // Initialize Inputs

    // Wait 100 ns for global reset to finish
    #100;

    // Add stimulus here

  end

endmodule
```

```
Process "Create Verilog Test Fixture" completed successfully

Started : "Launching ISE Text Editor to edit a_test.v".
```

FIG:5.14 Gives A Testbench File In The Project Window In Xilinx

STEP 13: select simulation from the view bar in the project window above the hiearchy window as follows.

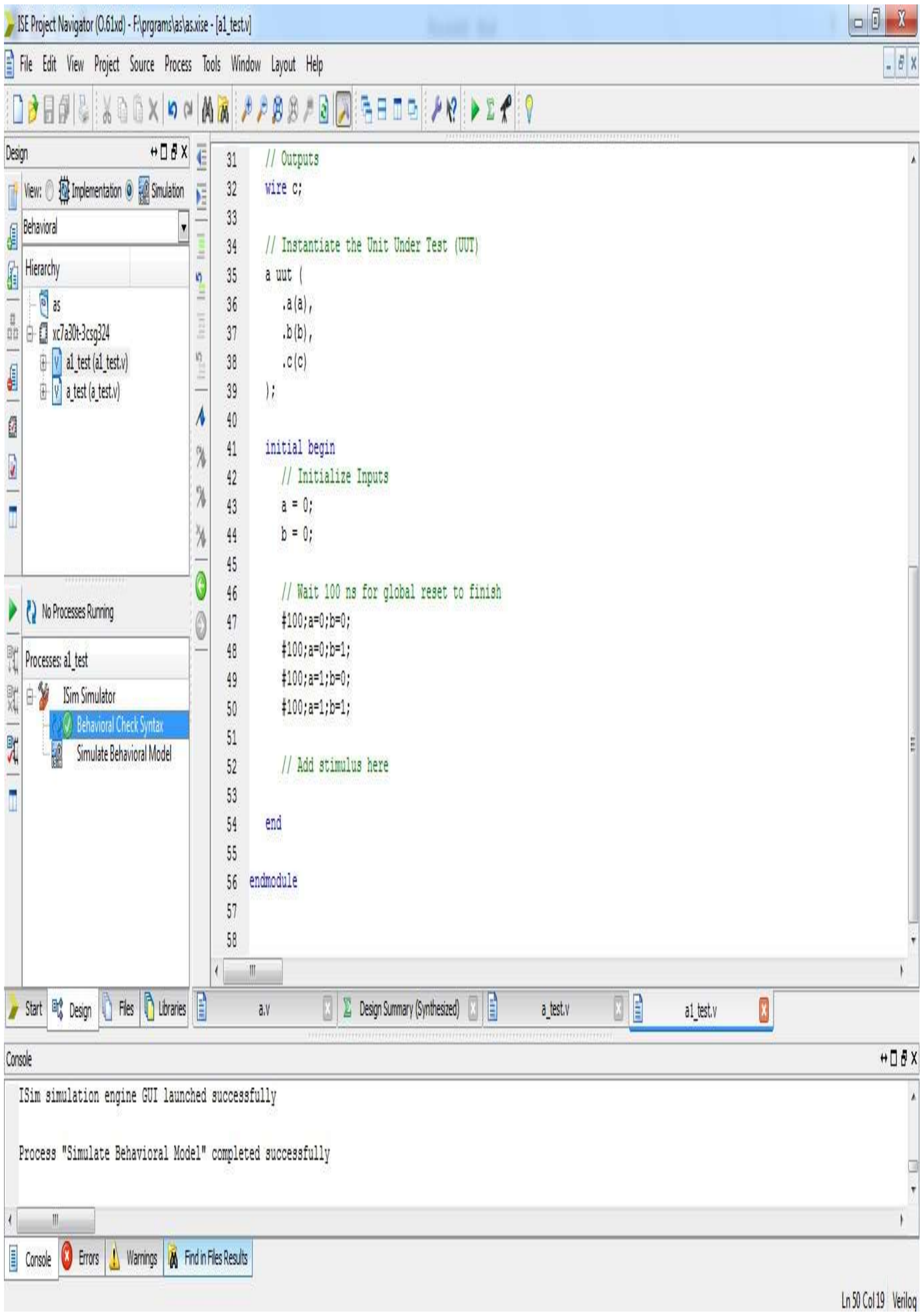

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a_test.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
as
xc7a30t-3csg324
a_test (a_test.v)
No Processes Running
Processes: a_test
ISim Simulator
Start Design Files Libraries
a.v
Design Summary
a_test.v
Console
Console Errors Warnings Find in Files Results

```
20  // Revision 0.01 - File Created
21  // Additional Comments:
22  //
23  ////////////////////////////////////////////////////////////////////////////////
24
25  module a_test;
26
27    // Outputs
28    wire ;
29
30    // Instantiate the Unit Under Test (UUT)
31    a uut (
32      .()
33    );
34
35    initial begin
36      // Initialize Inputs
37
38      // Wait 100 ns for global reset to finish
39      #100;
40
41      // Add stimulus here
42
43    end
44
45  endmodule
46
47
```

```
ERROR:HDLCompiler:806 - "F:/prgrams/as/a_test.v" Line 28: Syntax error near ";".
ERROR:HDLCompiler:806 - "F:/prgrams/as/a_test.v" Line 32: Syntax error near "(".
ERROR:ProjectMgmt - 2 error(s) found while parsing design hierarchy.
```

FIG:5.15 Select Simulation From The View Bar In The Project Window In Xilinx

STEP 14: Double click on ISE Simulator it will expand as follows click on behavioural check syntax and it will check for syntax errors in test bench file.

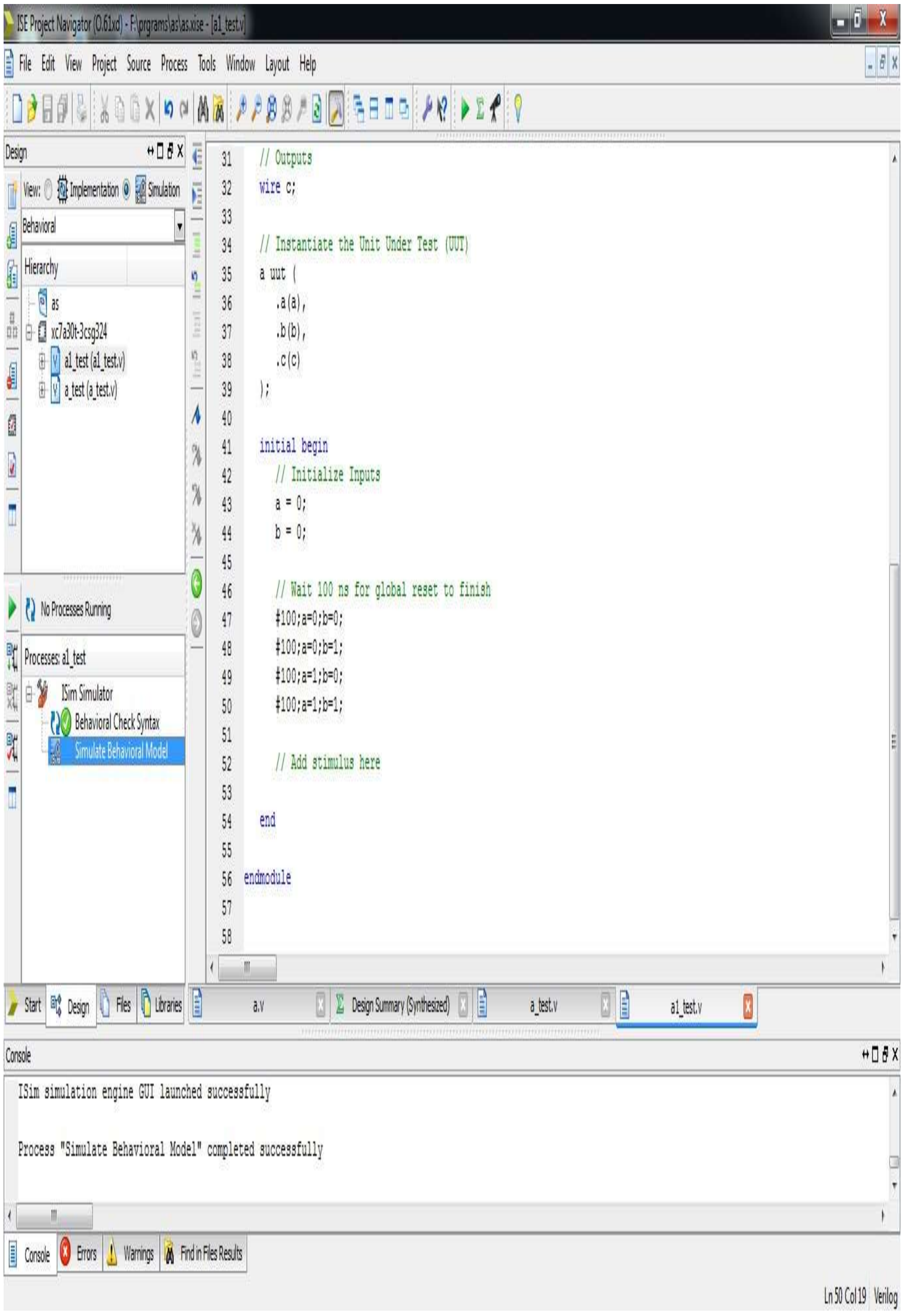

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a1_test.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
as
xc7a30t-3csg324
a1_test (a1_test.v)
a_test (a_test.v)
No Processes Running
Processes: a1_test
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
a.v
Design Summary (Synthesized)
a_test.v
a1_test.v
Console
Console Errors Warnings Find in Files Results
Ln 50 Col 19 Verilog

```
    // Outputs
    wire c;

    // Instantiate the Unit Under Test (UUT)
    a uut (
        .a(a),
        .b(b),
        .c(c)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;

        // Wait 100 ns for global reset to finish
        #100;a=0;b=0;
        #100;a=0;b=1;
        #100;a=1;b=0;
        #100;a=1;b=1;

        // Add stimulus here

    end

endmodule
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

FIG:5.16 Double Click On ISE Simulator In Xilinx

STEP 15: click on simulate behavioural model, it will displays wave form for in response to the inputs given in the test bench file.

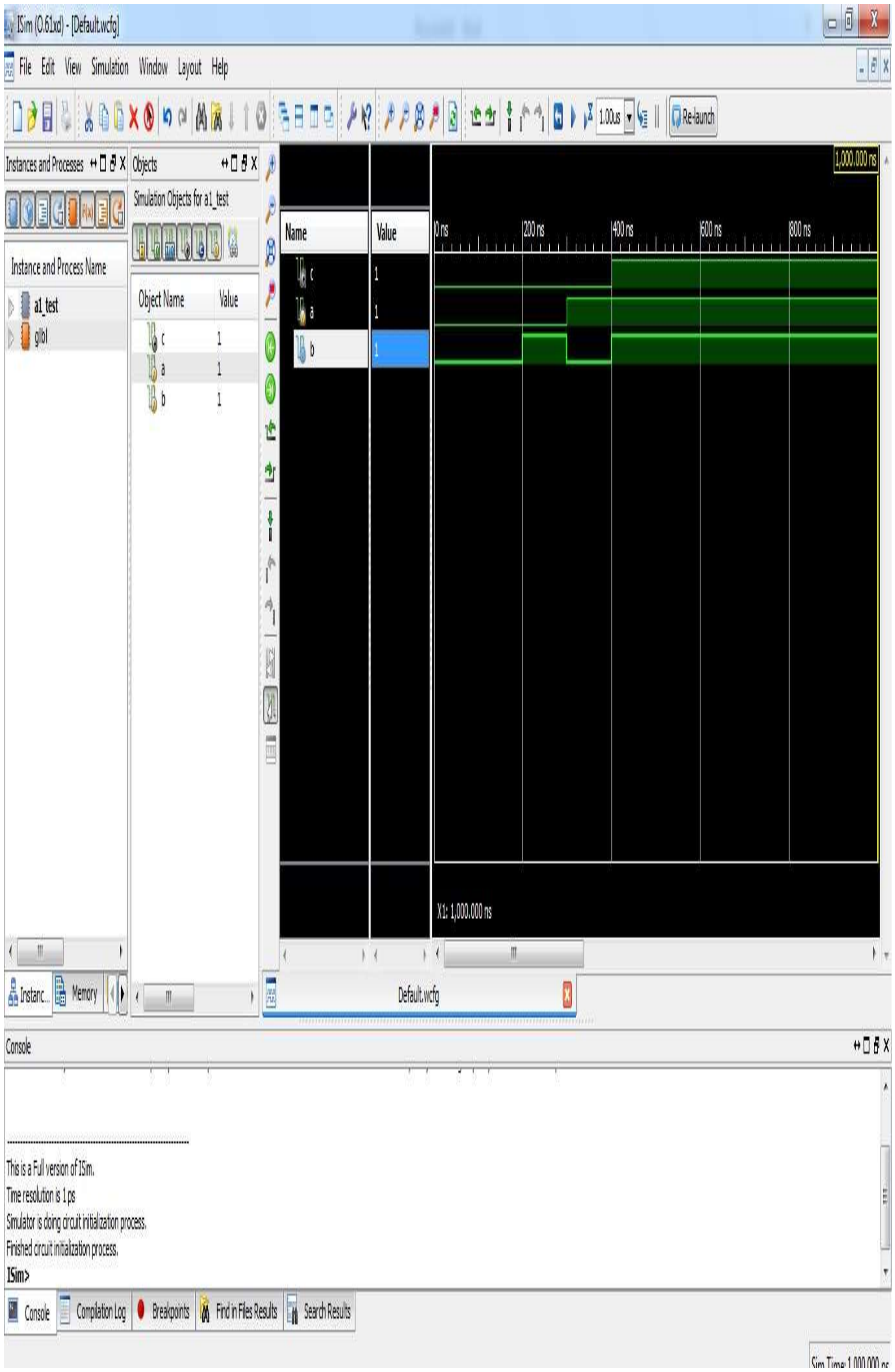

ISE Project Navigator (O.61xd) - F:\prgrams\as\as.xise - [a1_test.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
as
xc7a30t-3csg324
a1_test (a1_test.v)
a_test (a_test.v)
No Processes Running
Processes: a1_test
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
Start Design Files Libraries
a.v Design Summary (Synthesized) a_test.v a1_test.v
Console Errors Warnings Find in Files Results
Ln 50 Col 19 Verilog

```
    // Outputs
    wire c;

    // Instantiate the Unit Under Test (UUT)
    a uut (
        .a(a),
        .b(b),
        .c(c)
    );

    initial begin
        // Initialize Inputs
        a = 0;
        b = 0;

        // Wait 100 ns for global reset to finish
        #100;a=0;b=0;
        #100;a=0;b=1;
        #100;a=1;b=0;
        #100;a=1;b=1;

        // Add stimulus here

    end

endmodule
```

```
ISim simulation engine GUI launched successfully

Process "Simulate Behavioral Model" completed successfully
```

FIG:5.17 Click On Simulate Behavioural Model In Xilinx

STEP 16: That wave form window having option to zoom out, zoom in to analyse the wave form clearly in order to understand behaviour of design.



Object Name	Value
c	1
a	1
b	1

```
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

FIG:5.18 Waveform Window Having Option To Zoom Out, Zoom In,In Xilinx

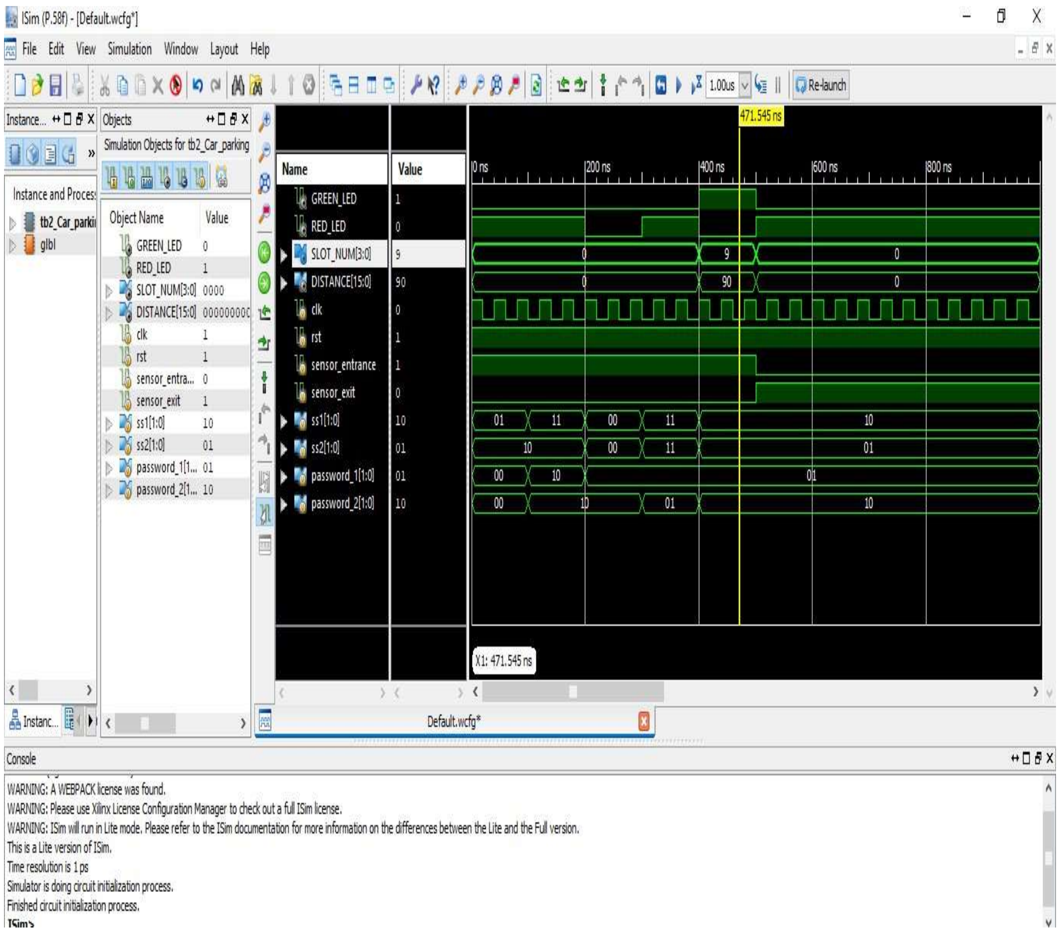
CHAPTER 6

RESULTS

6.1 Results

Implementing a car parking system using Verilog HDL involves designing a digital system that can manage parking spaces efficiently, detect vehicle entry and exit, and display available slots. This system typically integrates counters, sensors, and display modules to provide real-time information about parking availability.

The core implementation relies on finite state machines (FSMs) to track the number of parked vehicles and control the entry/exit of cars. switches are used to detect when a vehicle enters or leaves the parking lot. When a car arrives, the system checks if space is available; if so, it increments the counter and updates the parking status. Conversely, when a car exits, the counter decrements, and the availability status updates accordingly.



Object Name	Value
GREEN_LED	0
RED_LED	1
SLOT_NUM[3:0]	0000
DISTANCE[15:0]	000000000
clk	1
rst	1
sensor_entra...	0
sensor_exit	1
ss1[1:0]	10
ss2[1:0]	01
password_1[1...	01
password_2[1...	10

Name	Value
GREEN_LED	1
RED_LED	0
SLOT_NUM[3:0]	9
DISTANCE[15:0]	90
clk	0
rst	1
sensor_entrance	1
sensor_exit	0
ss1[1:0]	10
ss2[1:0]	01
password_1[1:0]	01
password_2[1:0]	10

```
WARNING: A WEBPACK license was found.
WARNING: Please use Xilinx License Configuration Manager to check out a full ISim license.
WARNING: ISim will run in Lite mode. Please refer to the ISim documentation for more information on the differences between the Lite and the Full version.
This is a Lite version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
```

FIG:6.1 Simulation Result Output Waveforms

The Verilog code consists of multiple modules, including a sensor interface, a counter, and a display unit. The sensor module detects cars and generates signals that are fed into a counter, which maintains the number of occupied slots. The display module uses seven-segment displays or LEDs to show the current count of available and occupied parking spaces.

The system is designed using synchronous logic with a clock-driven FSM to ensure accurate state transitions.To ensure efficient simulation and synthesis, the design is verified using testbenches in Verilog. Simulations in ModelSim or Xilinx Vivado help analyze the behavior of the parking system under different scenarios, such as simultaneous entry and exit events. Once verified, the system can be implemented on an FPGA for real-world deployment. Optimizations, such as using priority logic for multiple entry/exit points, can further enhance the system's performance.

6.2 ADVANTAGES

The car parking system using Verilog HDL offers several advantages, making it an efficient and practical solution for automated parking management. Some key advantages include :

1. Automation and Efficiency

The system operates automatically without requiring human intervention. It efficiently manages parking slots, detects vehicle movement, and updates availability in real time, reducing manual errors and improving overall efficiency.

2. Real-Time Monitoring

The use of sensors and counters ensures that the system continuously monitors parking occupancy. The real-time display helps drivers find available parking spaces quickly, reducing traffic congestion in parking areas.

3. Accurate and Reliable Operation

Implementing the system using Verilog HDL and FSM-based logic ensures precise and reliable operation. The system processes sensor inputs with minimal delay, accurately updating parking slot availability.

4. Scalability and Flexibility

The design is easily scalable, allowing modifications to support larger parking lots by adjusting the counter size and adding additional sensors. The modular nature of Verilog coding makes it flexible for customization as per requirements.

5. Low Power Consumption

The system operates efficiently with minimal power usage, making it suitable for deployment in areas where power conservation is essential. FPGA implementation ensures optimized hardware utilization.

6. Fast Response Time

The system processes entry and exit events instantly, ensuring that parking status updates are immediate. This reduces waiting time for drivers and helps in smoother parking operations.

7. Integration with Smart Systems

The system can be easily integrated with smart city infrastructure, IoT-based parking solutions, or mobile applications, providing enhanced user experience and remote monitoring capabilities.

8. Cost-Effective Implementation

Using Verilog HDL and FPGA technology provides a cost-efficient solution compared to traditional automated parking management systems that require complex mechanical setups.

9. Improved Traffic Management

By guiding drivers to available parking spots efficiently, the system helps in reducing unnecessary vehicle movement, leading to better traffic flow in parking lots and urban areas.

10. Enhanced Security

The system can be expanded to include RFID, number plate recognition, or surveillance systems for security monitoring, ensuring controlled access and preventing unauthorized parking.

6.3 APPLICATIONS

1. Smart Parking in Commercial Complexes.

Used in shopping malls, office buildings, and multiplexes to automate parking slot allocation. Reduces human effort in managing parking and provides real-time updates on slot availability.

2.Residential and Apartment Parking.

Helps in managing limited parking spaces in gated communities and apartment complexes. Prevents unauthorized vehicle entry and ensures fair space allocation for residents.

3.Airport and Railway Station Parking.

High-traffic areas like airports and railway stations benefit from automated parking to accommodate large volumes of vehicles.Ensures efficient space utilization and quick slot identification.

4. Hospitals and Emergency Centers.

Provides hassle-free parking in hospitals and healthcare facilities, where quick access to parking is critical for emergency cases.Helps staff, patients, and visitors find available slots without delays.

5. Smart Cities and IoT-Based Parking Systems.

Can be integrated into smart city infrastructure for efficient urban traffic management. Works with IoT (Internet of Things) to provide mobile-based real-time parking updates and slot reservations.

6. Educational Institutions and Universities.

Useful for managing parking in colleges, universities, and schools, ensuring orderly parking for students, faculty, and staff. Prevents congestion and unauthorized vehicle entry.

7. Public and Government Parking Lots.

Can be implemented in government buildings, municipal parking areas, and metro stations to improve public parking efficiency.Reduces parking-related disputes and improves overall space management.

8. Corporate Offices and IT Parks.

Ensures smooth parking management for large IT companies and corporate hubs where thousands of employees park daily.Helps in tracking available spaces and allocating reserved spots for employees.

9. Event and Stadium Parking.

Useful for concerts, sports stadiums, and exhibition halls, where high footfall requires an organized parking solution.Prevents unnecessary delays and traffic congestion near event venues.The sensor at the exit gate detects the departing car and sends a signal to the control unit, decrementing the occupied slot count.

10. Automated Toll Booth Parking.

Can be integrated with toll plaza systems to manage parking spaces for heavy vehicles, buses, and trucks at rest stops or service stations. Helps in optimizing space for long-haul vehicle parking.

CHAPTER 7
CONCLUSION & FEATURE SCOPE

7.1 CONCLUSION

With the growing global number of vehicles on roads and consuming a long time in finding a vacant parking space, the traffic congestions and air pollution are inevitable. Thus, it is important to urge researchers to develop smart parking systems to be utilized by motorists and parking operators. Current issues associated with parking systems have been presented and discussed. In ISPS architecture has been developed to provide motorists with an advance information on parking lots availability. Moreover, we proposed and used a novel mathematical model to calculate the number of available parking lots considering the expected arrival time on a real time basis. This work contributes to driving the full digital transformation of smart cities.

The goal of this project was to develop a most effective smart car parking system. With the support of Xilinx ISE Design Suite, smart car parking system is implemented using Verilog HDL. It increases productivity, reduces costs and speeds up market time. The system built can be used for many applications, and can easily increase the number of slot choices and increase parking protection. Through using the above implemented program parking becomes simple. The car is correctly identified and parking safety will be stressed. Even the drivers can easily pick the slot.

The automation of car parking significantly reduces human effort, thereby eliminating common issues such as parking congestion, mismanagement, and space unavailability. The use of sensors for detecting vehicle presence enhances accuracy and provides real-time data to optimize parking slot utilization. Furthermore, integrating the system with microcontrollers or FPGA-based platforms enhances its processing speed and adaptability. Moreover, we proposed and used a novel mathematical model to calculate the number of available parking lots considering the expected arrival time on a real time basis. This work contributes to driving the full digital transformation of smart cities.

In conclusion, the implementation of an automatic car parking system using Verilog HDL presents an innovative approach to smart parking management. Its ability to optimize space utilization, reduce manual intervention. parts within design blocks. Data bits build into digital words used to communicate with other design blocks within the system.

7.2 FUTURE SCOPE

The arrival of autonomous vehicles (AVs) threatens to exploit more efficient the future of the smart parking program. Urban cities around the world have already started experimenting with self-parking cars, advanced AV parking lots. But it is easy to use this device that is not only unique to a private car park like malls, business parking etc., But multiple sites, such as public parking, can also be built and the functionality can be added by giving parking information. By purging the need for human labor, this will make parking space management more efficient.

1. Integration with IoT:

Integrating the automatic car parking system with Internet of Things (IoT) can significantly enhance its functionality, making it more intelligent and user-friendly. The use of IoT-based sensors, cloud computing, and real-time data processing can improve parking management, reduce congestion, and optimize space utilization.

2. Mobile Application Connectivity:

Integrating a mobile application with the automatic car parking system can enhance user experience by providing real-time information, seamless booking, and remote access. A mobile app allows drivers to efficiently find, reserve, and manage parking slots, reducing congestion and saving time.

3. AI and Machine Learning:

The integration of Artificial Intelligence (AI) and Machine Learning (ML) in automatic car parking systems can significantly enhance efficiency, accuracy, and user experience. These technologies enable smart decision-making, predictive analytics, and automated control, reducing human intervention and optimizing parking management.

4. Automated Payment System:

An automated payment system in an automatic car parking system enhances convenience, eliminates manual transactions, and speeds up the parking process. By integrating contactless and digital payment methods, users can pay seamlessly, reducing congestion at entry and exit points.

5. Smart Traffic Management:

Integrating smart traffic management with an automatic car parking system can significantly reduce congestion, improve vehicle flow, and enhance the overall urban infrastructure. By leveraging IoT, AI, and real-time data analytics, traffic and parking systems can work together to provide seamless navigation and efficient space utilization.

6. Computer Vision Integration:

Integrating computer vision into an automatic car parking system enhances accuracy, security, and efficiency by enabling real-time monitoring, automated vehicle detection, and intelligent space allocation. Using AI-powered image processing, cameras can analyze parking lots, recognize vehicles, and optimize the parking experience.

7. Scalability for Large Parking Areas:

As urbanization increases, parking demand grows, making scalability a crucial aspect of automatic car parking systems. A scalable system can efficiently handle multi-level, underground, and large-scale commercial parking lots, ensuring seamless operation even with high traffic volumes.

8.**Scalability for Large Parking Areas**:

As urbanization increases, parking demand grows, making scalability a crucial aspect of automatic car parking systems. A scalable system can efficiently handle multi-level, underground, and large-scale commercial parking lots, ensuring seamless operation even with high traffic volumes.

9.Enhanced Security Features:

Security is a critical aspect of automatic car parking systems, ensuring the safety of vehicles, drivers, and parking infrastructure. By integrating advanced surveillance, AI-driven monitoring, and automated authentication, the system can prevent theft, unauthorized access, and other security risks.

10.Communication with Autonomous Vehicles:

As autonomous vehicles (AVs) become more prevalent, integrating vehicle-to-infrastructure (V2I) communication with automatic car parking systems is essential. This technology enables seamless parking, optimized space utilization, and reduced congestion without human intervention.

REFERENCES

[1] Y.-W. Chang, H.-Y. Shih, and T.-N. Lin, ‘‘AI-URG: Account identity-based uncertain graph framework for fraud detection,’’ IEEE Trans. Computat. Social Syst., vol. 11, no. 3, pp. 3706–3728, Jun. 2024.

[2] M. Wazid, J. Singh, A. K. Das, and J. J. P. C. Rodrigues, ‘‘An ensemble-based machine learning-envisioned intrusion detection in industry 5.0-driven healthcare applications,’’ IEEE Trans. Consum. Electron., vol. 70, no. 1, pp. 1903–1912, Feb. 2024.

[3] Gongjun Yan, Weiming Yang, Rawat, D.B.,Olariu, S., (2024). SmartParking: A Secure and Intelligent Parking System. Intelligent Transportation Systems Magazine, IEEE, 3(1), 18-30. Conferences

[4] Tosung hua_Hsu et. al, (2023). Development of an Automatic parking system for vehicle. IEEE vehicle power & propulsion conference, 3-5.

[5] Hua-Chun Tan, Jie Zhang, Xin-Chen Ye, Hui-Ze Li, Pei Zhu, Qing-Hua Zhao, (2022). Intelligent car-searching system for large park. International Conference on Machine Learning and Cybernetics, 3134-3138.

[6] Liu Liang, Zhang Lei, Xiao Jin, (2022). The simulation of an auto-parking system. 6th IEEE Conference on Industrial Electronics and Applications, 249-253.

[7] Du Shaobo et. al, (2021). The Research and Design of Intellectual Parking System Based on RFID. 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2427-2430.

[8] Gongjun Yan; Olariu, S.; Weigle, M.C.; Abuelela, M., (2021). SmartParking: A Secure and Intelligent Parking System Using NOTICE. 11th International IEEE Conference on Intelligent Transportation Systems, 569-574.

[9]T. T. Huong, T. P. Bac, K. N. Ha, N. V. Hoang, N. X. Hoang, N. T. Hung, and K. P. Tran, ‘‘Federated learning-based explainable anomaly detection for industrial control systems,’’ IEEE Access, vol. 10, pp. 53854–53872, 2022.

[10] G.-Y. Shin, D.-W. Kim, and M.-M. Han, ‘‘Data discretization and decision boundary data point analysis for unknown attack detection,’’ IEEE Access, vol. 10, pp. 114008–114015, 2022.

[11] H. Whitworth, S. Al-Rubaye, A. Tsourdos, and J. Jiggins, ‘‘5G aviation networks using novel AI approach for DDoS detection,’’ IEEE Access, vol. 11, pp. 77518–77542, 2021.

[12] E. Paolini, L. Valcarenghi, L. Maggiani, and N. Andriolli, ‘‘Real-time clustering based on deep embeddings for threat detection in 6G networks,’’ IEEE Access, vol. 11, pp.

115827–115835, 2020.
[13] F. Rustam, A. Raza, M. Qasim, S. K. Posa, and A. D. Jurcut, ‘‘A novel approach for real-time server-based attack detection using meta-learning,’’ IEEE Access, vol. 12, pp. 39614–39627, 2020.
[14] U. Sabeel, S. S. Heydari, K. El-Khatib, and K. Elgazzar, ‘‘Incremental adversarial learning for polymorphic attack detection,’’ IEEE Trans. Mach. Learn. Commun. Netw., vol. 2, pp. 869–887, 2019.
[15] C. A. Fadhilla, M. D. Alfikri, and R. Kaliski, ‘‘Lightweight meta-learning BotNet attack detection,’’ IEEE Internet Things J., vol. 10, no. 10, pp. 8455–8466, May 2019.

APPENDIX

➢ FOR SLOTS CHECKING

```
// Company:
// Engineer:
//
// Create Date:    19:32:51 11/24/2021
// Design Name:
// Module Name:    Car_Parking
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module Car_Parking(
 input clk,rst,
 input sensor_entrance, sensor_exit,
 input [1:0]ss1,ss2,
 input [1:0] password_1, password_2,
 output reg GREEN_LED,RED_LED,
 output reg [3:0] SLOT_NUM,
 output reg [15:0]DISTANCE

   );

 parameter       SLOT1=       4'b0001,SLOT2=       4'b0010,SLOT3=       4'b0011,SLOT4=
  4'b0100,SLOT5= 4'b0101,SLOT6= 4'b0110,SLOT7= 4'b0111,SLOT8= 4'b1000,
                     SLOT9=     4'b1001,SLOT10=     4'b1010,SLOT11=     4'b1011,SLOT12=
  4'b1100,SLOT13= 4'b1101,SLOT14= 4'b1110,SLOT15= 4'b1111,
                     NO_SLOTS=4'b0000;


 always @(*)
 begin
 if ((sensor_entrance == 1)&&((password_1==2'b01)&&(password_2==2'b10)))
 begin
 case ({ss1,ss2})

 SLOT1 : begin SLOT_NUM = SLOT1;
                               DISTANCE = 16'd10;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT2 : begin SLOT_NUM = SLOT2;
                               DISTANCE = 16'd20;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT3 : begin SLOT_NUM = SLOT1;
                               DISTANCE = 16'd30;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT4 : begin SLOT_NUM = SLOT4;
                               DISTANCE = 16'd40;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT5 : begin SLOT_NUM = SLOT5;
                               DISTANCE = 16'd50;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT6 : begin SLOT_NUM = SLOT6;
                               DISTANCE = 16'd60;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT7 : begin SLOT_NUM = SLOT7;
                               DISTANCE = 16'd70;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT8 : begin SLOT_NUM = SLOT8;
                               DISTANCE = 16'd80;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT9 : begin SLOT_NUM = SLOT9;
                               DISTANCE = 16'd90;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT10 : begin SLOT_NUM = SLOT10;
                               DISTANCE = 16'd100;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT11 : begin SLOT_NUM = SLOT11;
                               DISTANCE = 16'd110;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT12 : begin SLOT_NUM = SLOT12;
                               DISTANCE = 16'd120;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT13 : begin SLOT_NUM = SLOT13;
                               DISTANCE = 16'd130;
                               GREEN_LED = 1;RED_LED = 0;
                               end
 SLOT14 : begin SLOT_NUM = SLOT14;
                               DISTANCE = 16'd140;
                               GREEN_LED = 1;RED_LED = 0;
                               end
```

```
 SLOT15 : begin SLOT_NUM = SLOT15;
                              DISTANCE = 16'd150;
                              GREEN_LED = 1;RED_LED = 0;
                              end
 default: begin SLOT_NUM = NO_SLOTS;
                              DISTANCE = 16'd0;
                              GREEN_LED = 0;RED_LED = 0;
                              end

endcase
end
else if ((sensor_exit == 1)&&((password_1==2'b01)&&(password_2==2'b10)))//password
  = 0110
 begin        SLOT_NUM = 4'b0;
                              DISTANCE = 16'd0;
                              GREEN_LED = 1;RED_LED = 0;
                              end

else
 begin        SLOT_NUM = NO_SLOTS;
                              DISTANCE = 16'd0;
                              GREEN_LED = 0;RED_LED = 0;
                              end



end
endmodule
```

- **TESTBENCH MODULE FOR CAR PARKING**

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
```

```
// Create Date:   01:54:00 11/25/2021
// Design Name:   Car_Parking
// Module Name:   C:/xilinx/CAR_PARKING/tb2_Car_parking.v
// Project Name:  CAR_PARKING
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: Car_Parking
//
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
module tb2_Car_parking;
    // Inputs
    reg clk;
    reg rst;
    reg sensor_entrance;
    reg sensor_exit;
    reg [1:0] ss1;
    reg [1:0] ss2;
    reg [1:0] password_1;
    reg [1:0] password_2;
    // Outputs
    wire GREEN_LED;
    wire RED_LED;
    wire [3:0] SLOT_NUM;
    wire [15:0] DISTANCE;
```

```
// Instantiate the Unit Under Test (UUT)
Car_Parking uut (
        .clk(clk),
        .rst(rst),
        .sensor_entrance(sensor_entrance),
        .sensor_exit(sensor_exit),
        .ss1(ss1),
        .ss2(ss2),
        .password_1(password_1),
        .password_2(password_2),
        .GREEN_LED(GREEN_LED),
        .RED_LED(RED_LED),
        .SLOT_NUM(SLOT_NUM),
        .DISTANCE(DISTANCE)
);
initial begin
        // Initialize Inputs
        clk = 1;
        rst = 1;
        sensor_entrance = 1;
        sensor_exit = 0;
        ss1 = 01;
        ss2 = 10;
        password_1 = 00;
        password_2 = 00;
        // Wait 100 ns for global reset to finish
        #100;
 ss1 = 11;
```

```
            ss2 = 10;
            password_1 = 10;
            password_2 = 10;
            #100;
 ss1 = 00;
            ss2 = 00;
            password_1 = 01;
            password_2 = 10;
            #100;
 ss1 = 11;
            ss2 = 11;
            password_1 = 01;
            password_2 = 01;
            // Add stimulus here
            #100;
 ss1 = 10;
            ss2 = 01;
            password_1 = 01;
            password_2 = 10;
            #100;
            sensor_exit =1;
            sensor_entrance =0;
 ss1 = 10;
            ss2 = 01;
            password_1 = 01;
            password_2 = 10;
end
 always #20
            clk = ~clk;
```

```
endmodule
```