**ABSTRACT**

In the entrance of the parking system, there's a detector that is activated to find a vehicle returning. Once the detector is triggered, a secret is requested to open the gate. If the entered secret is correct, the gate would hospitable let the vehicle get in. Otherwise, the gate remains fast. If this automotive is getting into the park being detected by the exit detector and another the automotive comes, the door are going to be fast and needs the approaching cars to enter passwords.

**CHAPTER 1**

**INTRODUCTION**

In this pragmatic world, several tasks were performed by every individual without being evasive. So, in order to sort out all the tasks for the efficient usage of time, wise steps should be taken to curb the wastage of time at unproductive areas such as at the most frequently performed action, which is the parking of vehicles. So, our paper provides a better alternative to have an efficient usage of time at parking correlated with the security issue which serves at its best. The major discussion involves the following solutions given below for the efficient usage of time which doesn’t spare much time for parking purpose and also in order to have a safe park without involving any sort of crashes. Systematic parking with security is the main motto. Security includes the usage of password at the time of park; Indication of number of available adjacent vacancies as well as their positions where only the adjacent vacancies are needed in particular; Indication of the total number of vacancies available in a particular slot; Indication of the nearest vacancy of a particular row or slot among all the rows of a parking area. Parking systems can also take advantage of innovative technologies in order to improve the ease and convenience of paying for parking.

Now a days, smart cards minimize transaction time by allowing user to simply wave their card in front of a reader. Mobile devices can also be used in payment transactions Public utilities need a parking system that can function efficiently and be integrated with the other urban city utilities. For the allotment of parking slots there is no proper way, thus parking management system fails in coordination and centralizing the information for an effective system. To avoid these problems, design of secured car parking management system is proposed, which will be implemented on FPGA to check vacancies and give security to car . Recently, a reconfigurable FPGA design is efficient method to implement a digital logic, because FPGA provides a compromise between general-purpose processors and ASIC. The FPGA based design is also more flexible, programmable and can be re-programmed. FPGA based design can be easily modified by modifying design’s software part. Our proposed system is designed in FPGA design style and gate level modeling.

In this realistic world every person dispensed several tasks while not being evasive. Thus, so as to hold out all the activities for the effective use of resources, wise steps ought to be taken to curb waste of your time in ineffective areas like the foremost ofttimes performed follow, that is un successful vehicle parking. Therefore, this paper offers associate choice for an efficient use of your time in parking relevant to the protection drawback that serves at its best. The key topic includes the subsequent solutions given below for the economical use of your time that doesn't take abundant time for parking functions and conjointly for providing a secure park while not involving risks of any kind. the most goal is systematic parking with protection. Protection needs the employment of secret once parking, indication of range of accessible vacancies still as their locations wherever solely the adjacent vacancies square measure required specially, total range of vacancies out there during a specific slot and even distance calculation to obstacles. Public services want a parking network which may effectively operate and be combined with alternative public utilities. there's no effective thanks to assign parking slots associated parking management system fails to assist and organize info for an economical system. so as to avoid these issues, style is planned for secured automotive parking management system, which can be enforced on FPGA to envision vacancies and FPGA design is an efficient methodology for implementing a digital logic, as FPGA offers a rendezvous between processors for general use and ASIC. The FPGA design is strong, programmable and might be re-functioned. It will simply amendment the FPGA primarily based style by dynamical the software package element. Our planned system {is styleed|is meant|is intended} for FPGA design and also the modeling of gate levels. HE main explanation for parking issues worldwide is ever increasing population and with it the quantity of vehicles.

The place out there for parking is restricted however the quantity of vehicles increasing on road daily has no check. in keeping with the recent trends, there has been associate exponential rise within the sale of cars. the quantity of cars soldout worldwide hyperbolic hugely from thirtynine.2 million in 1999 to eighty one.57 million in 2018. In twenty years the sales were raised by a worth as high as around equal forty million.

The statistics within the variety of a graph square measure diagrammatical in Figure one. Due to this huge increase in traffic on road, the immediate next consequence that was ascertained was a eruption in congestion issues on road. Figure two depicts the rise in congestion issues from 987 to 2003. This gave rise to associate increasing range of accidents on roads and conjointly the traveling time needed to succeed in destination hyperbolic. Even to hide atiny low distance consumed abundant. To solve and kind out these parking issues in large malls, instructional inst itutes and to avoid wastage of your time at parking, of individuals visiting such organizations, here is a shot created to create the parking method a great deal easier. Also, with the assistance of evolving technologies.

The system is updated more to give birth to a totally machine-controlled parking system. At places with parking issues, folks got to wait during a long queue and seek for free slots out there as there's no methodology to trace the vacant slots mechanically. notwithstanding there squ are measure slots out there, guests don’t get to grasp thanks to large infrastructure and that they find yourself dawdling in sorting out slots, at that point this planned methodology would are helpful.

**CHAPTER 2**

**LITERATURE SURVEY**

Soh Chun Khang, et.al (2010) conferred work on parking system {in that|during which|within which} the quantity of slot which is accessible for parking is send as a message to driver. Driver will resend sms hard to please for a replacement position once the sooner assigned slot gets crammed. Huachun tan, et.al (2009) planned a system that is useful to search out the park at places wherever there an outsized parking zone. this is often done by capturing pictures through camera mounted at every parking slot. info like range plate of automotive and color square measure recognized and hold on in knowledge. This knowledge so includes info concerning a ll automotives lay within the heap and thus it's attainable to search out any car simply.

1. V.Srikant, et.al(2009) camp up with system to find the free parking slots. Author has used wireless communication technology to create the parking system a lot of economical. Gongjun Yan, et.al (2011), planned associate intelligent parking system that was supported secured wireless system and detector communication. economical auto mobile parking space utilization and fast search of free slot was the work concerned.

Insop Song et.al, (2006) worked upon system exploitation Field Programmable Gate Array (FPGA) exploitation symbolic logic Controller. Advantage of this technique is reduction in computation time. Several literatures are done associated with the planned work. In proves the practicability of the approach. This strategy is efficient and includes several of the aspects of sensible automotive parking management. The project's central plan is to avoid troubles we have a tendency to face within the daily routine of parking our cars. Day by day the matter of parking cars proliferates. to the present finish a literature survey was performed so as to make sure that this is able to not be replicated as before. this is often created exploitation MATLAB, and it uses cameras to find the free parking slots. exploitation this program photos collected by a police work camera were processed in real time to check the parking zone occupancies. the knowledge is processed through a central management unit and is directed to the show panels set at strategic parking lot locations. Through the small print shown on the panels the drivers can understand the empty parking zone.

In today’s world parking lots have become redundant and needs lot of manpower to handle and maintain it. These parking lots are not user friendly and do not provide data regarding availability of free spaces. Many researchers have contributed to this issue and formalized with various methods to better optimize the parking lot to serve the needs. The author proposed smart parking reservation system using short message services (SMS), for that he uses Global System for Mobile(GSM) with microcontroller to enhances security[9]. The ZigBee technique is used along with the GSM module for parking management and reservation [11]. The author uses Global Positioning System(GPS) and Android platform to show available parking spaces. However, reservation for the same is not available [12].

The suggested work has been the subject of a number of studies. [2] demonstrates that the method is feasible. This technique is cost-effective and incorporates a number of smart parking management features. The main goal of the project is to eliminate problems that we encounter in our daily lives when parking our automobiles.[1] Parking automobiles is becoming more and more of a challenge. In order to assure that this would not be repeated, a literature survey was conducted. This was made with MATLAB and relies on cameras to find available parking spaces. This application was used to assess parking lot occupancies by processing images taken by a surveillance camera in real time. A central control unit is used to process the data. The data is analysed by a central control unit before being sent to display panels strategically placed across the parking lot. The drivers will be able to see the vacant parking lot thanks to the details on the panels [3].

**CHAPTER 3**

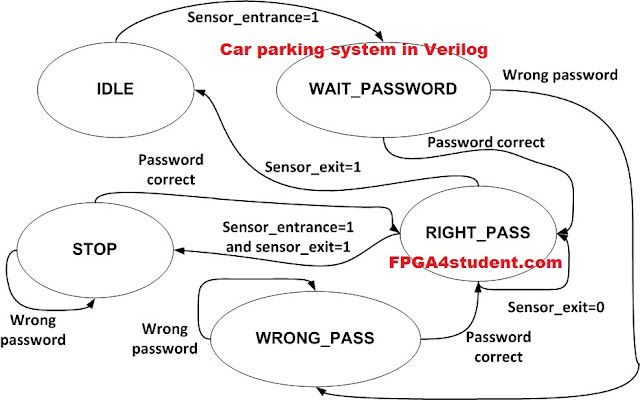
**EXISTING SYSTEM**

**CHAPTER 4**

**PROPOSED SYSTEM**

In the entrance of the parking system, there is a sensor which is activated to detect a vehicle coming. Once the sensor is triggered, a password is requested to open the gate. If the entered password is correct, the gate would open to let the vehicle get in. Otherwise, the gate is still locked. If the current car is getting in the car park being detected by the exit sensor and another the car comes, the door will be locked and requires the coming car to enter passwords. What is an FPGA, as some of you might already be aware of an FPGA is a type of integrated circuit (IC) that can be programmed for different algorithms after fabrication. Modern FPGA devices consist of up to two million logic cells that can be configured to implement a variety of software algorithms.

Although the traditional FPGA design flow is more similar to a regular IC than a processor, an FPGA provides significant cost advantages in comparison to an IC development effort and offers the same level of performance in most cases. Another advantage of the FPGA when compared to the IC is its ability to be dynamically reconfigured. This process, which is the same as loading a program in a processor, can affect part or all of the resources



available in the FPGA fabric. For this documentation the FPGA being used is Zynq™-7000 All programmable SoC devicein which we specifically use the Zynq 7020 (ZC702) FPGA board To solve and sort out the problems in parking system, here is a solution. A sensor is at the entrance of the parking system which is activated to detect a vehicle coming. When a car enters in, a password is needed. If the password entered is correct the gate will open or else it will be locked. This is also the same for the exit process. And with the help of ultrasonic sensor the distance is measured in which the next car is available, the number of vacant slots and the number of cars parked already will be given in the form of a message.

In which the output depends only on the state of the system. Hence, in state transition diagrams for Moore machines, the outputs are labeled in the circles. Recall that mealy machines are much like Moore machines, but the outputs can depend on inputs as well as the current state. Hence, in state transition diagrams for Mealy machines, the outputs are labeled on the arcs instead of in the circles. The block of combinational logic that computes the outputs uses the current state and inputs, as was shown in Fig. 1

A finite state machine is a mathematical model of computation usually represented as a graph, with a finite number of nodes describing the possible states of the system, and a finite number of arcs representing the transitions that do or do not change the state, respectively. Such a machine is mostly used to model computer programs and sequential logic. There are two types of FSMs: mealy machine, where the output values are determined based on the current state together with the current input, and Moore machines, where the output is determined solely based on the current state. Extended finite state machine (EFSMs) [ allow for internal variables than can store more detailed internal state information. Thus, EFSMs allow for a larger number of internal states. Mapping the large number of internal states to a smaller number of visible states requires an abstraction of the system, which can influence the testing process

The above Fig. 2 shows the demo of the proposed parking system

• It is a simple project is to implement a car parking system in verilog

• In the entrance of the parking system, there is a sensor which is activated to detect a vehicle coming

• Once the sensor is activated, a password is request to open the gate

• Till that time the car will be in the idle state. Before entering the sensor entrance the current state will be in idle

• In the sensor entrance it ask the password if it is wait password then it will be idle state

• If it is correct password then it goes into parking

• If it is wrong password then it give another chance and ask password again

• If it is correct password it goes to parking if it is wrong password then again it goes to idle state

• It give 2 chances for password

• If the entered password is correct,the gate would open to let the vechicle get in

• Otherwise the gate is still locked.if the current car is getting in the car park being detected by the exit sensor and the another the car comes ,the door will be locked and requires the coming car to enter the pass.

• The simulation is designed with the help of Xilinx software.

**Chapter 5**

**INTRODUCTION TO VLSI**

Very-large-scale integration (VLSI) is the process of creating integrated circuits by combining thousands of transistor-based circuits into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device. The term is no longer as common as it once was, as chips have increased in complexity into the hundreds of millions of transistors.

**5.1 Overview:**

The first semiconductor chips held one transistor each. Subsequent advances added more and more transistors, and, as a consequence, more individual functions or systems were integrated over time. The first integrated circuits held only a few devices, perhaps as many as ten diodes, transistors, resistors and capacitors, making it possible to fabricate one or more logic gates on a single device. Now known retrospectively as "small-scale integration" (SSI), improvements in technique led to devices with hundreds of logic gates, known as large-scale integration (LSI), i.e. systems with at least a thousand logic gates. Current technology has moved far past this mark and today's microprocessors have many millions of gates and hundreds of millions of individual transistors.

At one time, there was an effort to name and calibrate various levels of large-scale integration above VLSI. Terms like Ultra-large-scale Integration (ULSI) were used. But the huge number of gates and transistors available on common devices has rendered such fine distinctions moot.

Terms suggesting greater than VLSI levels of integration are no longer in widespread use. Even VLSI is now somewhat quaint, given the common assumption that all microprocessors are VLSI or better.

As of early 2008, billion-transistor processors are commercially available, an example of which is Intel's Montecito Itanium chip. This is expected to become more commonplace as semiconductor fabrication moves from the current generation of 65 nm processes to the next 45 nm generations (while experiencing new challenges such as increased variation across process corners). Another notable example is NVIDIA’s 280 series GPU.

This microprocessor is unique in the fact that its 1.4 Billion transistor count, capable of a teraflop of performance, is almost entirely dedicated to logic (Itanium's transistor count is largely due to the 24MB L3 cache). Current designs, as opposed to the earliest devices, use extensive design automation and automated logic synthesis to lay out the transistors, enabling higher levels of complexity in the resulting logic functionality. Certain high-performance logic blocks like the SRAM cell, however, are still designed by hand to ensure the highest efficiency (sometimes by bending or breaking established design rules to obtain the last bit of performance by trading stability).

**5.2 What is VLSI?**

## VLSI stands for "Very Large Scale Integration". This is the field which involves packing more and more logic devices into smaller and smaller areas.

## VLSI

## Simply we say Integrated circuit is many transistors on one chip.

## Design/manufacturing of extremely small, complex circuitry using modified semiconductor material

## Integrated circuit (IC) may contain millions of transistors, each a few mm in size

## Applications wide ranging: most electronic logic devices

# 5.3 History of Scale Integration:

* late 40s Transistor invented at Bell Labs
* late 50s First IC (JK-FF by Jack Kilby at TI)
* early 60s Small Scale Integration (SSI)
* 10s of transistors on a chip
* late 60s Medium Scale Integration (MSI)
* 100s of transistors on a chip
* early 70s Large Scale Integration (LSI)
* 1000s of transistor on a chip
* early 80s VLSI 10,000s of transistors on a
* chip (later 100,000s & now 1,000,000s)
* Ultra LSI is sometimes used for 1,000,000s
  + - SSI - Small-Scale Integration (0-102)
    - MSI - Medium-Scale Integration (102-103)
* LSI - Large-Scale Integration (103-105)
* VLSI - Very Large-Scale Integration (105-107)
* ULSI - Ultra Large-Scale Integration (>=107)

### 5.4 Advantages of ICs over discrete components:

While we will concentrate on integrated circuits , the properties of integrated circuits-what we can and cannot efficiently put in an integrated circuit-largely determine the architecture of the entire system. Integrated circuits improve system characteristics in several critical ways. ICs have three key advantages over digital circuits built from discrete components:

* Size. Integrated circuits are much smaller-both transistors and wires are shrunk to micrometer sizes, compared to the millimeter or centimeter scales of discrete components. Small size leads to advantages in speed and power consumption, since smaller components have smaller parasitic resistances, capacitances, and inductances.
* Speed. Signals can be switched between logic 0 and logic 1 much quicker within a chip than they can between chips. Communication within a chip can occur hundreds of times faster than communication between chips on a printed circuit board. The high speed of circuits on-chip is due to their small size-smaller components and wires have smaller parasitic capacitances to slow down the signal.
* Power consumption.Logic operations within a chip also take much less power. Once again, lower power consumption is largely due to the small size of circuits on the chip-smaller parasitic capacitances and resistances require less power to drive them.

#### 5.5 VLSI and systems:

These advantages of integrated circuits translate into advantages at the system level:

* Smaller physical size. Smallness is often an advantage in itself-consider portable televisions or handheld cellular telephones.
* Lower power consumption. Replacing a handful of standard parts with a single chip reduces total power consumption. Reducing power consumption has a ripple effect on the rest of the system: a smaller, cheaper power supply can be used; since less power consumption means less heat, a fan may no longer be necessary; a simpler cabinet with less shielding for electromagnetic shielding may be feasible, too.
* Reduced cost. Reducing the number of components, the power supply requirements, cabinet costs, and so on, will inevitably reduce system cost. The ripple effect of integration is such that the cost of a system built from custom ICs can be less, even though the individual ICs cost more than the standard parts they replace.

Understanding why integrated circuit technology has such profound influence on the design of digital systems requires understanding both the technology of IC manufacturing and the economics of ICs and digital systems.

##### **Applications**

* Electronic system in cars.
* Digital electronics control VCRs
* Transaction processing system, ATM
* Personal computers and Workstations
* Medical electronic systems.
* Etc….

### 5.6 Applications of VLSI:

Electronic systems now perform a wide variety of tasks in daily life. Electronic systems in some cases have replaced mechanisms that operated mechanically, hydraulically, or by other means; electronics are usually smaller, more flexible, and easier to service. In other cases electronic systems have created totally new applications. Electronic systems perform a variety of tasks, some of them visible, some more hidden:

* Personal entertainment systems such as portable MP3 players and DVD players perform sophisticated algorithms with remarkably little energy.
* Electronic systems in cars operate stereo systems and displays; they also control fuel injection systems, adjust suspensions to varying terrain, and perform the control functions required for anti-lock braking (ABS) systems.
* Digital electronics compress and decompress video, even at high-definition data rates, on-the-fly in consumer electronics.
* Low-cost terminals for Web browsing still require sophisticated electronics, despite their dedicated function.
* Personal computers and workstations provide word-processing, financial analysis, and games. Computers include both central processing units (CPUs) and special-purpose hardware for disk access, faster screen display, *etc*.
* Medical electronic systems measure bodily functions and perform complex processing algorithms to warn about unusual conditions. The availability of these complex systems, far from overwhelming consumers, only creates demand for even more complex systems.

The growing sophistication of applications continually pushes the design and manufacturing of integrated circuits and electronic systems to new levels of complexity. And perhaps the most amazing characteristic of this collection of systems is its variety-as systems become more complex, we build not a few general-purpose computers but an ever wider range of special-purpose systems. Our ability to do so is a testament to our growing mastery of both integrated circuit manufacturing and design, but the increasing demands of customers continue to test the limits of design and manufacturing

### 5.7 ASIC:

An Application-Specific Integrated Circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. For example, a chip designed solely to run a cell phone is an ASIC. Intermediate between ASICs and industry standard integrated circuits, like the 7400 or the 4000 series, are application specific standard products (ASSPs).

As feature sizes have shrunk and design tools improved over the years, the maximum complexity (and hence functionality) possible in an ASIC has grown from 5,000 gates to over 100 million. Modern ASICs often include entire 32-bit processors, memory blocks including ROM, RAM, EEPROM, Flash and other large building blocks. Such an ASIC is often termed a SoC (system-on-a-chip). Designers of digital ASICs use a hardware description language (HDL), such as Verilog or VHDL, to describe the functionality of ASICs.

Field-programmable gate arrays (FPGA) are the modern-day technology for building a breadboard or prototype from standard parts; programmable logic blocks and programmable interconnects allow the same FPGA to be used in many different applications. For smaller designs and/or lower production volumes, FPGAs may be more cost effective than an ASIC design even in production.

* An application-specific integrated circuit (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use.

## A Structured ASIC falls between an FPGA and a Standard Cell-based ASIC

## Structured ASIC’s are used mainly for mid-volume level design. The design task for structured ASIC’s is to map the circuit into a fixed arrangement of known cells.

## CHAPTER 6

## INTRODUCTION TO XILINX

**6.1 Migrating Projects from Previous ISE Software Releases:**

When you open a project file from a previous release, the ISE® software prompts you to migrate your project. If you click Backup and Migrate or Migrate Only, the software automatically converts your project file to the current release. If you click Cancel, the software does *not* convert your project and, instead, opens Project Navigator with no project loaded.

**Note:**After you convert your project, you *cannot* open it in previous versions of the ISE software, such as the ISE 11 software. However, you can optionally create a backup of the original project as part of project migration, as described below.

**To Migrate a Project**

1. In the ISE 12 Project Navigator, select **File > Open Project**.
2. In the Open Project dialog box, select the .xise file to migrate.

**Note**You may need to change the extension in the Files of type field to display .npl (ISE 5 and ISE 6 software) or .ise (ISE 7 through ISE 10 software) project files.

1. In the dialog box that appears, select **Backup and Migrate** or **Migrate Only**.
2. The ISE software automatically converts your project to an ISE 12 project.

**Note**If you chose to Backup and Migrate, a backup of the original project is created at *project\_name\_*ise12migration.zip.

1. Implement the design using the new version of the software.

**Note**Implementation status is *not* maintained after migration.

**6.2 Properties:**

For information on properties that have changed in the ISE 12 software, see [ISE 11 to ISE 12 Properties Conversion](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\ise_r_properties_conversion.htm).

**6.3 IP Modules:**

If your design includes IP modules that were created using CORE Generator™ software or Xilinx® Platform Studio (XPS) and you need to modify these modules, you may be required to update the core. However, if the core netlist is present and you do not need to modify the core, updates are not required and the existing netlist is used during implementation.

**6.4 Obsolete Source File Types:**

The ISE 12 software supports all of the source types that were supported in the ISE 11 software.

If you are working with projects from previous releases, state diagram source files (.dia), ABEL source files (.abl), and test bench waveform source files (.tbw) are no longer supported. For state diagram and ABEL source files, the software finds an associated HDL file and adds it to the project, if possible. For test bench waveform files, the software automatically converts the TBW file to an HDL test bench and adds it to the project. To convert a TBW file *after* project migration, see [Converting a TBW File to an HDL Test Bench](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_p_converting_tbw.htm)

**6.5 Using ISE Example Projects:**

To help familiarize you with the ISE® software and with FPGA and CPLD designs, a set of example designs is provided with Project Navigator. The examples show different design techniques and source types, such as VHDL, Verilog, schematic, or EDIF, and include different constraints and IP.

**To Open an Example**

1. Select **File > Open Example**.
2. In the [Open Example dialog box](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_open_example_project.htm), select the Sample Project Name.

**Note**To help you choose an example project, the Project Description field describes each project. In addition, you can scroll to the right to see additional fields, which provide details about the project.

1. In the Destination Directory field, enter a directory name or browse to the directory.
2. Click **OK**.

The example project is extracted to the directory you specified in the Destination Directory field and is automatically opened in Project Navigator. You can then run processes on the example project and save any changes.

**Note**If you modified an example project and want to overwrite it with the original example project, select **File > Open Example**, select the Sample Project Name, and specify the same Destination Directory you originally used. In the dialog box that appears, select **Overwrite the existing project** and click **OK**.

**6.6 Creating a Project:**

Project Navigator allows you to manage your FPGA and CPLD designs using an ISE® project, which contains all the source files and settings specific to your design. First, you must create a project and then, add source files, and set process properties. After you create a project, you can run processes to implement, constrain, and analyze your design. Project Navigator provides a wizard to help you create a project as follows.

**Note**If you prefer, you can create a project using the **[New Project dialog box](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_new_project.htm)** instead of the New Project Wizard. To use the New Project dialog box, deselect the **Use New Project wizard** option in the **[ISE General page](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_ise_general_options.htm)** of the Preferences dialog box.

**To Create a Project**

1. Select **File > New Project** to launch the New Project Wizard.
2. In the **[Create New Project page](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_npw_create_new_project.htm),** set the name, location, and project type, and click **Next**.
3. *For EDIF or NGC/NGO projects only*: In the **[Import EDIF/NGC Project page](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_npw_import_edif_ngc_project.htm)**, select the input and constraint file for the project, and click **Next**.
4. In the **[Project Settings page](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_npw_device_properties.htm)**, set the device and project properties, and click **Next**.
5. In the **[Project Summary page](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_db_npw_project_summary.htm)**, review the information, and click **Finish** to create the project

Project Navigator creates the project file (*project\_name*.xise) in the directory you specified. After you add source files to the project, the files appear in the Hierarchy pane of the

**6.7 [Design panel](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_r_design_panel.htm):**

Project Navigator manages your project based on the design properties (top-level module type, device type, synthesis tool, and language) you selected when you created the project. It organizes all the parts of your design and keeps track of the processes necessary to move the design from design entry through implementation to programming the targeted Xilinx® device.

**Note**For information on changing design properties, see **[Changing Design Properties](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pn_p_changing_design_properties.htm).**

You can now perform any of the following:

* Create new source files for your project.
* Add existing source files to your project.
* Run processes on your source files.

Modify process properties.

**6.8 Creating a Copy of a Project:**

You can create a copy of a project to experiment with different source options and implementations. Depending on your needs, the design source files for the copied project and their location can vary as follows:

* Design source files are left in their existing location, and the copied project points to these files.
* Design source files, including generated files, are copied and placed in a specified directory.
* Design source files, excluding generated files, are copied and placed in a specified directory.

Copied projects are the same as other projects in both form and function. For example, you can do the following with copied projects:

* Open the copied project using the File > Open Project menu command.
* View, modify, and implement the copied project.
* Use the Project Browser to view key summary data for the copied project and then, open the copied project for further analysis and implementation, as described in
  1. **[Using the Project Browser](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\ise_c_project_browser.htm)**:

Alternatively, you can create an archive of your project, which puts all of the project contents into a ZIP file. Archived projects must be unzipped before being opened in Project Navigator. For information on archiving, see **[Creating a Project Archive](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\ise_c_project_archive.htm).**

**To Create a Copy of a Project**

1. Select **File > Copy Project**.
2. In the Copy Project dialog box, enter the **Name** for the copy.

**Note**The name for the copy can be the same as the name for the project, as long as you specify a different location.

1. Enter a directory **Location** to store the copied project.
2. Optionally, enter a **Working directory**.

By default, this is blank, and the working directory is the same as the project directory. However, you can specify a working directory if you want to keep your ISE® project file (.xise extension) separate from your working area.

1. Optionally, enter a **Description** for the copy.

The description can be useful in identifying key traits of the project for reference later.

1. In the Source options area, do the following:

Select one of the following options:

* **Keep sources in their current locations -** to leave the design source files in their existing location.

If you select this option, the copied project points to the files in their existing location. If you edit the files in the copied project, the changes also appear in the original project, because the source files are shared between the two projects.

* **Copy sources to the new location -** to make a copy of all the design source files and place them in the specified Location directory.

If you select this option, the copied project points to the files in the specified directory. If you edit the files in the copied project, the changes do *not* appear in the original project, because the source files are not shared between the two projects.

Optionally, select **Copy files from Macro Search Path directories** to copy files from the directories you specify in the Macro Search Path property in the**[Translate Properties](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\pp_db_translate_properties.htm)** dialog box. All files from the specified directories are copied, not just the files used by the design.

**Note:** If you added a netlist source file directly to the project as described in **[Working with Netlist-Based IP](file:///C:\\Users\\BHANUPRAKASH\\Downloads\\ise_c_using_fixed_netlist_ip.htm)**, the file is automatically copied as part of Copy Project because it is a project source file. Adding netlist source files to the project is the preferred method for incorporating netlist modules into your design, because the files are managed automatically by Project Navigator.

Optionally, click **Copy Additional Files** to copy files that were not included in the original project. In the Copy Additional Files dialog box, use the **Add Files** and **Remove Files** buttons to update the list of additional files to copy. Additional files are copied to the copied project location after all other files are copied.To exclude generated files from the copy, such as implementation results and reports, select

**6.10 Exclude generated files from the copy**:

When you select this option, the copied project opens in a state in which processes have not yet been run.

1. To automatically open the copy after creating it, select **Open the copied project**.

**Note**By default, this option is disabled. If you leave this option disabled, the original project remains open after the copy is made.

Click **OK**.

**6.11 Creating a Project Archive:**

A project archive is a single, compressed ZIP file with a .zip extension. By default, it contains all project files, source files, and generated files, including the following:

* User-added sources and associated files
* Remote sources
* Verilog `include files
* Files in the macro search path
* Generated files
* Non-project files

**6.12 To Archive a Project:**

1. Select **Project > Archive**.
2. In the Project Archive dialog box, specify a file name and directory for the ZIP file.
3. Optionally, select **Exclude generated files from the archive** to exclude generated files and non-project files from the archive.
4. Click **OK**.

A ZIP file is created in the specified directory. To open the archived project, you must first unzip the ZIP file, and then, you can open the project.

**Note**Sources that reside outside of the project directory are copied into a remote\_sources subdirectory in the project archive. When the archive is unzipped and opened, you must either specify the location of these files in the remote\_sources subdirectory for the unzipped project, or manually copy the sources into their original location

**CHAPTER 7**

**RESULTS**

**CHAPTER 8**

**CONCLUSION**

The goal of this project was to develop a most effective smart car parking system. This was the key impetus in deciding to incorporate the FPGA method. With the support of Xilinx ISE Design Suite, smart car parking system is implemented using Verilog HDL. The design is tested on FPGA kit Spartan6. The FPGA 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.

**REFRENCES**

1. Ramneet Kaur and Balwinder Singh (2013) “Design And Implementation Of Car Parking System On FPGA”. Du Shaobo; Sun Shibao;,(2012) "The research and design of intellectual parking system based on
2. RFID," Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on,pp.2427-2430.Gongjun Yan; Weiming Yang; Rawat, D.B.; Olariu, S.,(2011) "SmartParking: A Secure and Intelligent Parking System," Intelligent Transportation Systems Magazine, IEEE , vol.3, no.1, pp.18-30.
3. Liu Liang; Zhang Lei; Xiao Jin;,(2011) "The simulation of an auto-parking system," Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference on , pp.249-253.
4. Soh Chun Khang; Teoh Jie Hong; Tan Saw Chin; Shengqiong Wang;(2010), "Wireless Mobile-Based Shopping Mall Car Parking System (WMCPS)," Services Computing Conference (APSCC), 2010 IEEE Asia-Pacific , pp.573-577.
5. Liu Liang, Zhang Lei, Xiao Jin,(2011). The simulation of an auto-parking system. 6th IEEE Conference on Industrial Electronics and Applications, 249-253.
6. Du Shaobo et. al, (2012). The Research and Design of Intellectual Parking System Based on RFID. 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2427-2430.

**SOURCE CODE:**

module PasswordBasedParkingSystem(

input [3:0] password, // 4-bit password input

input reset, // reset signal

input enter, // signal to check password

output reg gate\_open // output to control the gate

);

reg [3:0] stored\_password; // stored password

always @(posedge enter or posedge reset) begin

if (reset) begin

stored\_password <= 4'b1010; // preset password

gate\_open <= 0; // initially gate is closed

end

else begin

if (password == stored\_password) begin

gate\_open <= 1; // open the gate

end

else begin

gate\_open <= 0; // keep the gate closed

end

end

end

Endmodule

module Testbench;

reg [3:0] password; // 4-bit password input

reg reset; // reset signal

reg enter; // signal to check password

wire gate\_open; // output to control the gate

// Instantiate the parking system module

PasswordBasedParkingSystem uut (

.password(password),

.reset(reset),

.enter(enter),

.gate\_open(gate\_open)

);

initial begin

// Initialize inputs

reset = 1;

password = 4'b0000;

enter = 0;

// Wait for the reset to complete

#10 reset = 0;

#10 enter = 1; password = 4'b1010; // Correct password

#10 enter = 0;

#10 enter = 1; password = 4'b1111; // Incorrect password

#10 enter = 0;

#10 enter = 1; password = 4'b1010; // Correct password

#10 enter = 0;

// Finish simulation

#10 $finish;

end

initial begin

$monitor("Time: %0d | Password: %b | Gate Open: %b", $time, password, gate\_open);

end

endmodule