****

1. **J. INSTITUTE OF ENGINEERING & TECHNOLOGY, MANGALURU**

(A Unit of Laxmi Memorial Educational Trust ®)

(Approved by AICTE, Affiliated to Visvesvaraya Technological University, Belagavi)

**Accredited By NBA (BE: CV, CSE, ECE, ISE & ME**)

SEMINAR REPORT

On

AJIET

“**Sequential Digital Circuits & Electronic Circuits and Number System**”

*COURSE TITLE: DIGITAL SYSTEM DESIGN USING VERILOG*

*COURSE CODE: BEC654A*

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

by

|  |
| --- |
| **NAME USN** |
| **G**auresh G Pai 4JK22CS016 |

**Course Instructor: Mrs. PRATHEKSHA RAI N**

Assistant Professor, Dept. of E&CE

CONTENTS

1. INTRODUCTION
2. CHAPTER 1: **NUMBER SYSTEMS**
3. CHAPTER 2: **LOGIC GATES**
4. CHAPTER 3: **SEQUENTIAL DIGITAL CIRCUITS**
5. CHAPTER 4: **ADVANCED SEQUENTIAL CIRCUITS**
6. SIMULATED OUTPUT
7. CONCLUSION

INTRODUCTION

This report offers an exhaustive and detailed overview of the fundamental concepts that guide the study of digital electronics. Digital electronics is the backbone of contemporary computing and embedded systems, and knowledge of its fundamental ideas is necessary for students, engineers, professionals in electronics, computer engineers, and associated fields. The report consists of two broad sections that together present theoretical knowledge as well as practical knowledge.

1. Electronic Circuits and Number Systems

The initial part provides the foundation by presenting electronic circuits and number systems, which are fundamental aspects of any digital system. It starts with the study of varying number representations, such as decimal, binary, octal, and hexadecimal systems. These numerical systems are vital for encoding, processing, and interpreting digital data. The report describes how to perform conversions between these systems and presents ordinary numeric codes employed in digital logic, including Binary-Coded Decimal (BCD) and Gray Code.

Besides, the section explores the basic logic gates—AND, OR, NOT, NAND, NOR, XOR, and XNOR—whose major building blocks are used in creating every digital circuit. Each gate's symbol representation, truth table, and usage in the real world are explained in detail. These basic elements are fundamental to the process of designing and analyzing more intricate digital systems.

2. Sequential Digital Circuits

The second half steps into the realm of sequential digital circuits—circuits whose outputs are a function of not just present inputs, but also past states. This brings in the aspect of memory in digital electronics, which is implemented with different components like latches and flip-flops (SR, D, JK, and T types). The report discusses the ways in which these components store and manipulate information so that digital systems can accomplish tasks like counting, timing, and state management.

Additionally, the unit includes counters (synchronous and asynchronous), shift registers, and other sequential devices that are frequently used within digital design. These elements are critical for the formation of systems such as digital clocks, buffers for data, and finite state machines. The applications of these circuits are presented through examples and diagrams, showing their significance within both simple and complex digital devices.

**CHAPTER 1: NUMBER SYSTEMS**

**1.1 Decimal & Binary Number Systems**

The base-10 or decimal system is our common day-to-day counting system with digits 0 through 9. Digital systems have the binary system (base-2) with two digits only: 0 and 1, which stand for the OFF and ON positions of electric switches.

Properties of the Binary System:

Has only two digits (0,1)

Every position is a power of 2

Rightmost digit is 2⁰, the next is 2¹, then 2², etc.

Conversion from Binary to Decimal

To translate binary into decimal, times each bit by its position value and add the results.

Example: Binary 1011₂

1 × 2³ = 8

0 × 2² = 0

1 × 2¹ = 2

1 × 2⁰ = 1

Decimal value = 11₁₀

Decimal to Binary Conversion:

Convert decimal to binary by repeatedly dividing the decimal number by 2 and writing down remainders in reverse order.

Example: Decimal 25₁₀

25 ÷ 2 = 12 remainder 1

12 ÷ 2 = 6 remainder 0

6 ÷ 2 = 3 remainder 0

3 ÷ 2 = 1 remainder 1

1 ÷ 2 = 0 remainder 1

Binary value = 11001₂

**1.2 Octal & Hexadecimal Number Systems**

Octal System (Base-8):

Works with digits 0-7

Each octal position a power of 8

Compact binary value representation (3 bits per octal digit)

Hexadecimal System (Base-16):

Works with digits 0-9 and letters A-F (values 10-15)

Each hexadecimal position a power of 16

Very compact for representing binary data (4 bits per hex digit)

Commonly used in programming and digital system design

Conversion Between Systems:

Binary to Octal: Arrange binary digits in groups of three from right to left, each group translated into its equivalent octal.

101101₂ = 101 101 = 5 5 = 55₈

Binary to Hexadecimal: Arrange binary digits in groups of four from right to left, each group translated into its equivalent hex.

1010 1111₂ = A F = AF₁₆

Octal/Hex to Binary: Translate each digit into its binary representation (3 bits for octal, 4 bits for hex).

3A₁₆ = 0011 1010₂

**1.3 Numeric Codes**

Digital systems usually need some special coding schemes to encode information. Familiar numeric codes are:

Binary Coded Decimal (BCD):

4 bits to represent each decimal digit (0-9)

Example: Decimal 25 in BCD = 0010 0101

Gray Code:

A circular binary code where values at adjacent locations differ by one bit at most

Used to minimize errors in position encoders and data transmission

Example: First 4 Gray code values: 0000, 0001, 0011, 0010

ASCII (American Standard Code for Information Interchange):

7-bit character, number, and control signal representation

Data communication standard

Example: ASCII for 'A' = 65₁₀ = 1000001₂

**CHAPTER 2: LOGIC GATES**

**2.1 Basic Logic Gates**

Logic gates are the fundamental building blocks of digital electronic circuits. They implement Boolean logic operations and are combined to create complex digital systems.

**NOT Gate (Inverter):**

* Single input, inverts the input signal
* Output is the complement of the input
* Boolean expression: Y = Ā (or Y = NOT A)

**AND Gate:**

* Two or more inputs, outputs 1 only when all inputs are 1
* Boolean expression: Y = A·B (or Y = A AND B)

**OR Gate:**

* Two or more inputs, outputs 1 when any input is 1
* Boolean expression: Y = A+B (or Y = A OR B)

**2.2 Universal Gates**

Universal gates can be used to implement any Boolean function, making them particularly important in digital circuit design.

**NAND Gate:**

* Combination of AND followed by NOT
* Universal gate (can implement any logic function)
* Boolean expression: Y = (A·B)̄

**NOR Gate:**

* Combination of OR followed by NOT
* Universal gate (can implement any logic function)
* Boolean expression: Y = (A+B)̄

**2.3 Applications of Logic Gates**

Logic gates form the basis for more complex digital components:

**Combinational Circuits:**

* Outputs depend solely on current inputs
* Examples: multiplexers, decoders, adders

**Boolean Function Implementation:**

* Any Boolean function can be realized using combinations of gates
* Can be simplified using Boolean algebra and Karnaugh maps
* Example: F (A, B, C) = AB + BC̄ can be implemented using AND, OR, and NOT gates

**CHAPTER 3: SEQUENTIAL DIGITAL CIRCUITS**

Sequential circuits differ from combinational circuits in that their outputs depend not only on the current inputs but also on previous states, effectively giving them "memory."

**3.1 Latches**

Latches are basic memory elements that store a single bit of information.

**SR Latch (Set-Reset):**

* Two inputs: Set (S) and Reset (R)
* When S=1, R=0: Output Q=1 (set state)
* When S=0, R=1: Output Q=0 (reset state)
* When S=0, R=0: Maintains previous state
* When S=1, R=1: Invalid/forbidden state

**D Latch (Data):**

* Improved version of SR latch that eliminates invalid state
* Single data input D and enable input
* When enable is active, output follows D input
* When enable is inactive, output is latched (maintains previous state)

**3.2 Flip-Flops**

Flip-flops are edge-triggered sequential elements that provide more controlled operation than latches.

**Types of Flip-Flops:**

1. **D Flip-Flop:**
   * Single data input
   * Transfers input value to output on clock edge
   * Used for data storage and registers
2. **JK Flip-Flop:**
   * Two inputs: J (set) and K (reset)
   * More versatile than SR flip-flop (no invalid state)
   * When J=K=1, output toggles
   * Used in counters and frequency dividers
3. **T Flip-Flop (Toggle):**
   * Single input T
   * When T=0, maintain state
   * When T=1, toggle output
   * Often used in counters and frequency dividers
4. **Asynchronous Inputs:**
   * Preset and Clear inputs that override normal operation
   * Act independently of the clock
   * Used for initialization and error recovery

**3.3 Counters**

Counters are sequential circuits that progress through a predetermined sequence of states.

**Types of Counters:**

1. **Asynchronous Counters:**
   * Also called ripple counters
   * Clock signal applied only to first flip-flop
   * Each subsequent flip-flop triggered by previous flip-flop's output
   * Simple design but suffers from propagation delay
2. **Synchronous Counters:**
   * All flip-flops triggered by the same clock signal
   * Faster operation than asynchronous counters
   * Requires more complex logic for state transitions

**Counter Specifications:**

* **Modulus:** Maximum count before resetting (e.g., mod-10 counts from 0 to 9)
* **Up/Down:** Direction of counting sequence
* **Load Feature:** Ability to preset to specific count
* **Self-Correcting:** Returns to proper sequence if disturbed

**CHAPTER 4: ADVANCED SEQUENTIAL CIRCUITS**

**4.1 Registers**

Registers are groups of flip-flops used to store multiple bits of information.

**Types of Registers:**

1. **Parallel-In Parallel-Out (PIPO):**
   * Data loaded and retrieved simultaneously
   * Used for temporary storage in processors
2. **Serial-In Serial-Out (SISO):**
   * Data shifts in and out one bit at a time
   * Used in communication systems
3. **Shift Registers:**
   * Can shift data left or right
   * Applications include serial-to-parallel conversion and digital delays

**4.2 State Machines**

State machines are sequential circuits that change states based on inputs and current state.

**State Machine Components:**

* Flip-flops to store current state
* Combinational logic to determine next state
* Output logic to produce required outputs

**Design Process:**

1. Define required states and transitions
2. Create state diagram and state table
3. Assign binary codes to states
4. Derive flip-flop input equations
5. Implement using flip-flops and gates

**Applications:**

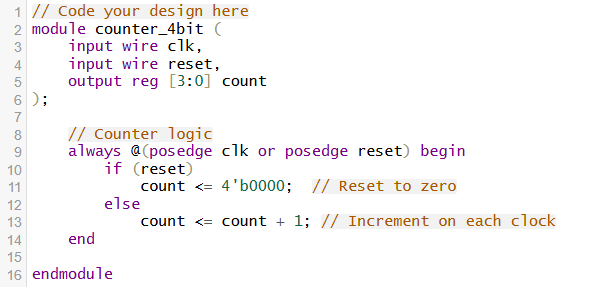
* Traffic light controllers
* Vending machines
* Digital communication protocols
* Computer instruction execution units

**SIMULATED OUTPUT**

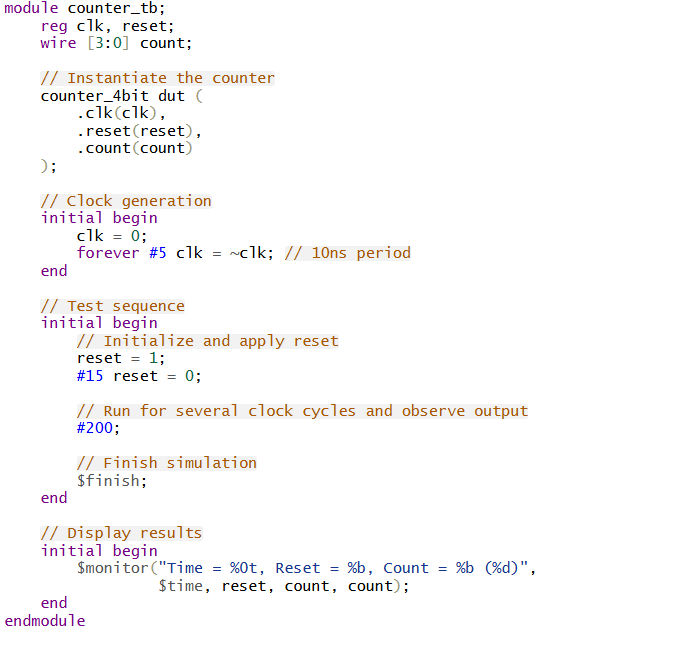
Below is a simulated implementation of a 4-bit binary counter using D flip-flops. This simulation demonstrates the practical application of sequential circuits.

4-bit synchronous counter using D flip-flops:

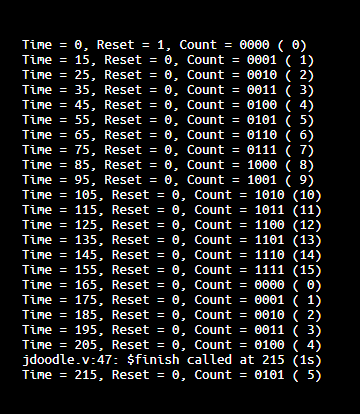
Design Code:



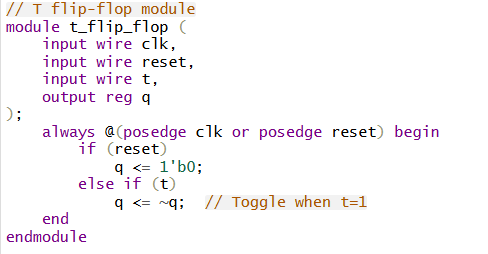
Test bench code:

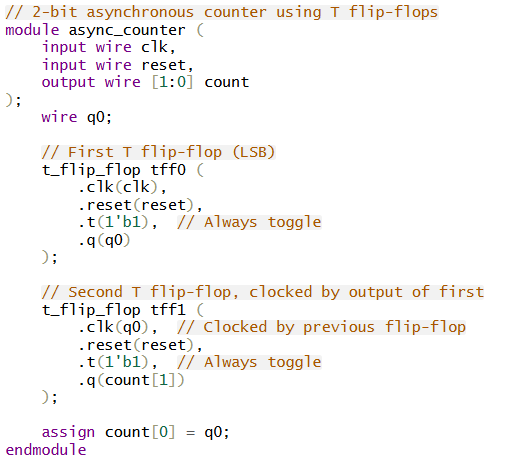


Output:



For a more complex example, here is a T flip-flop implementation and 2-bit asynchronous counter:





Expected Output Pattern (Waveform):

| **clk** | **count[1]** | **count[0]** | **Binary** |
| --- | --- | --- | --- |
| **0** | **0** | **0** | **00** |
| **↑** | **0** | **1** | **01** |
| **↑** | **1** | **0** | **10** |
| **↑** | **1** | **1** | **11** |
| **↑** | **0** | **0** | **00** |
| **↑** | **0** | **1** | **01** |
| **...** | **...** | **...** | **...** |

**CONCLUSION**

In conclusion, this report has explored the essential building blocks of digital electronics, integrating theoretical knowledge with practical applications. Beginning with number systems, we emphasized how binary, octal, and hexadecimal representations serve as the fundamental language for digital computation. The ability to convert between these systems is not only a critical academic skill but also an indispensable tool in real-world digital system design, programming, and debugging.

Moving beyond representation, we examined electronic circuits, starting with logic gates—AND, OR, NOT, NAND, NOR, XOR—which are the cornerstone of all digital devices. These gates form the basis for performing logical operations and are combined to create more complex combinational circuits. We also explored how universal gates like NAND and NOR can implement any Boolean function, making them powerful and flexible components in circuit design.

The report then transitioned into sequential circuits, which add the vital dimension of memory to digital systems. Flip-flops, latches, and counters enable the storage and timing of digital information, forming the foundation for time-based operations. These components are essential in constructing devices such as digital clocks, communication systems, data storage units, and control systems. The asynchronous counter example, built using T flip-flops, demonstrated how simple logic elements can be combined to produce meaningful and functional systems capable of counting in binary—an operation central to nearly all digital devices.

As digital systems continue to grow in complexity—driven by fields such as embedded systems, telecommunications, artificial intelligence, and IoT—the importance of mastering these foundational principles becomes increasingly apparent. Whether designing with FPGAs or developing custom ASICs, understanding number systems, logic gates, and sequential circuits remains crucial. This knowledge empowers engineers and technologists to design efficient, reliable, and scalable digital systems that drive innovation across industries.

Ultimately, these concepts serve not only as academic theory but as the gateway to building the intelligent, responsive technologies that shape our modern world.