VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI - 590 018, KARNATAKA

Internship Report on

"VLSI Design Engineer: Design 3-8 Decoder using the Verilog HDL"

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

In

ELECTRONICS AND COMMUNICATION

For the academic year 2024-2025

Internship work carried out at

Rooman Technologies Pvt Ltd
30, 12th Main Rd, 1st Stage, Rajajinagar,
Bengaluru, Karnataka- 560010

Submitted by
PRAJWAL N G USN: 1CR21EC151

Under the guidance of

Dr. M Venkatesh
Asst. Professor
Dept. of ECE, CMRIT
Bengaluru- 560 037

ELECTRONICS AND COMMUNICATION ENGINEERING
CMR Institute of Technology, Bengaluru - 560037

NCVET
कौशल गुणवत्ता प्रगति

इलेक्ट्रॉनिक्स सेक्टर स्किल्स काउंसिल ऑफ इंडिया
Electronics Sector Skills Council of India

राष्ट्रीय व्यावसायिक शिक्षा एवं प्रशिक्षण परिषद द्वारा मान्यता प्राप्त
Recognised by NCVET

कौशल योग्यता प्रमाणपत्र
Certificate for Skill Competency

प्रमाणपत्रसंख्या
Certificate No:
AEKAA0010QG-05-EH-00428-2023-V1.1-ESSC-227347

प्रमाणित किया जाता है कि श्री/सुश्री/एमएक्स
This is to certify that Mr./Ms./Mx **Mr. Prajwal N G**

सुपुत्र
Son of Guru

जन्म तिथि
Date of Birth 25/05/2003

नामांकन संख्या
Enrolment No CAN_33901016

ने जॉब रोल/अहर्ता का आंकलन सफलतापूर्वक
has successfully cleared the assessment in the job role/qualification Vlsi Design Engineer

अवधि
of Duration 780 Hrs

अर्जित किया
having earned 26

क्रेडिट एनसीआरएफ/एनएसक्यूएफ स्तर
Credits at NCrF/NSQF Level 5

प्रशिक्षण केन्द्र
Training Centre CMR INSTITUTE OF TECHNOLOGY, BENGALURU

जिला
District KOPPAL

राज्य
State KARNATAKA

प्रतिशत/श्रेणी के साथ उत्तीर्ण किया।
with B %/Grade

Skill India
कौशल भारत-कुशल भारत

जारी करने का स्थान
Place of Issue: Delhi

जारी करने की तिथि
Date of Issue: 12/05/2025

PMKVY
प्रधानमंत्री कौशल विकास योजना

ESSCI

नाम Name:
Amrit Manwani
पद Designation:
Chairperson
हस्ताक्षर Signature:

ई-सत्यापन लिंक
e-Verification link:
https://admin.skillindiadigital.gov.in/documentverification.nsdcindia
Digitally Generated Certificate

NCrF - National Credit Framework
NSQF - National Skills Qualification Framework

WADHWANI FOUNDATION

CERTIFICATE OF COMPLETION

This is to certify that

PRAJWAL N G

from **Rooman Technologies Pvt Ltd**

has successfully completed

Life Skills (Jeevan Kaushal) 2.0

on **February 28, 2025**

Ajay Kela
CEO
Wadhwani Foundation

Scan & verify

www.opportunity.wfglobal.org

This certificate confirms the completion of 93 hours of course.

CMR INSTITUTE OF TECHNOLOGY

Bengaluru – 560037

DEPARTMENT OF ELECTRONICS & COMMUNICATION

CERTIFICATE

This is to certify that the internship work entitled **"VLSI Design Engineer: Design 3-8 Decoder using the Verilog HDL"** is carried out by **PRAJWAL N G,** bearing USN: **1CR21EC151** a bonafide student of **CMR INSTITUTE OF TECHNOLOGY** in partial fulfillment for the award of **Bachelor of Engineering** in **Electronics and Communication** from **Visvesvaraya Technological University,** Belagavi during the academic year **2024 - 2025**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The Internship Report is approved and satisfies the academic requirements with respect to internship work prescribed for the said degree.

Signature of Guide	Signature of HOD	Signature of Principal
……	……	……
Dr. M Venkatesh	**Dr. Pappa M.**	**Dr. Sanjay Jain**
Asst. Professor,	HoD, Dept. of ECE,	Principal
Dept. of ECE,	CMRIT, Bengaluru.	CMRIT, Bengaluru
CMRIT, Bengaluru.		

Viva:

Internal Evaluator	External Evaluator
1 …….	…….
2 …….	……

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose consistent guidance and encouragement crowned our efforts with success.

I consider it as my privilege to express the gratitude to all those who guided in the completion of the project.

I express my gratitude to Principal, **Dr. Sanjay Jain,** for having provided me the golden opportunity to undertake this project work in their esteemed organization.

I sincerely thank **Dr. Pappa M.,** HoD**,** Department of Electronics and Communication Engineering, CMR Institute of Technology for the immense support given to me.

I express my gratitude to our project guide **Dr. M Venkatesh**, Asst. Professor, for their support, guidance, and suggestions throughout the project work.

Last but not the least, heartfelt thanks to our parents and friends for their support.

Above all, I thank the Lord Almighty for His grace on us to succeed in this endeavor.

ABSTRACT

This report presents the design and implementation of a 3-to-8 decoder using Verilog HDL. A decoder is a combinational circuit that converts N input lines into a maximum of 2^N output lines. The study explores the fundamental concepts of Verilog, including module-based design, behavioral and structural modeling, testbench creation, and verification strategies. The design of the 3-to-8 decoder is simulated and functionally verified using Universal Verification Methodology (UVM). Performance metrics such as power consumption, area utilization, and timing characteristics are analyzed using the OpenROAD tool on the gf180 platform. The results confirm that the design is functionally correct, achieving efficient implementation with minimal power consumption and optimal area utilization. Additionally, the decoder's applications in memory systems, code conversion, data routing, and digital control circuits are discussed, highlighting its significance in digital hardware design.

ABOUT THE COMPANY

ROOMAN TECHNOLOGIES- Overview

Rooman Technologies is a **leading IT training and solutions provider** in India, specializing in **hardware, networking, cybersecurity, and software development**. Established in **1999**, the company has built a strong reputation for **high-quality technical education** and **industry-relevant certification programs**.

With a network of **100+ training centers** across India, Rooman Technologies collaborates with **government initiatives and enterprises** to provide **vocational training, skill development, and corporate upskilling programs**. Their curriculum includes certifications from **Cisco, Microsoft, Red Hat**, and other global tech leaders, ensuring learners receive **cutting-edge industry expertise**.

Beyond training, Rooman Technologies develops **IoT solutions, ERP systems, cybersecurity services, and data center management**. Their mission is to **empower individuals with jobready skills**, bridging the gap between **education and employability** to make India a hub for **technological excellence**.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Chapter1

INTRODUCTION

Verilog HDL (Hardware Description Language) plays a crucial role in digital system design by providing a structured language for describing, simulating, and synthesizing hardware components. Developed in the mid-1980s, Verilog became a standard in the electronic design automation (EDA) industry. A Verilog program is built around modules that encapsulate design functionality with defined input, output, and inout ports. Behavioral and structural modeling were introduced. In behavioral modeling, circuits are described using high-level constructs like always and initial blocks. In structural modeling, modules interconnect lower-level building blocks such as gates. This session emphasized continuous assignment (assign) for combinational logic, and procedural blocks (always, initial) for both combinational and sequential logic.

1.1 3-8 Decoder

A 3-to-8 decoder is a combinational logic device that takes three input lines and produces eight output lines. For each possible combination of the three input binary lines, one and only one output signal will be logic 1. The decoder acts as a min-term generator, where each output corresponds to one min-term. Decoders find various applications in the design of digital systems. In digital electronics, a combinational logic circuit that converts an N-bit binary input code into M output channels in such a way that only one output channel is activated for each one of the possible combinations of inputs is known as a decoder.

In other words, a combinational logic circuit which converts N input lines into a maximum of 2N output lines is called a decoder. Therefore, a decoder is a combination logic circuit that is capable of identifying or detecting a particular code. The operation that a decoder performs is referred to as decoding.

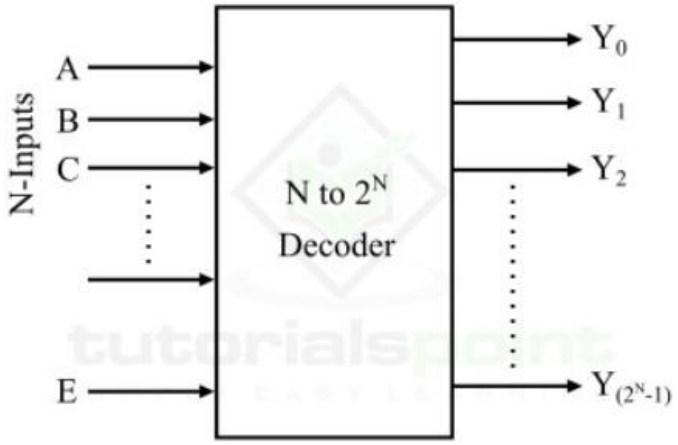

Figure 1.1 Decoder

The 3 to 8 decoder is one that has 3 input lines and 8 (23) output lines.

1.1.1 Introduction to Verilog and Basic Constructs

Focusing on the concept of modules, ports, and fundamental data types such as wire and reg. A Verilog program is built around modules that encapsulate design functionality with defined input, output, and inout ports. Behavioral and structural modeling were introduced. In behavioral modeling, circuits are described using high-level constructs like always and initial blocks. In structural modeling, modules interconnect lower-level building blocks such as gates. This session emphasized continuous assignment (assign) for combinational logic, and procedural blocks (always, initial) for both combinational and sequential logic. Understanding the difference between wire (used in continuous assignments and connecting modules) and reg (used for storage within procedural blocks) was critical.

1.1.2 Gate Level Modeling and Primitives

Focused on gate-level modeling, a form of structural modeling that describes circuits using Verilog's predefined primitive gates. Students explored logic primitives such as AND, OR, NOT, and XOR gates and learned how to instantiate them directly in Verilog. Multiple input versions of these gates were discussed to accommodate more complex logic combinations. Tristate devices (bufif1, notif0, etc.) were introduced, critical for designing shared data buses and ensuring only one driver at a time is active. Pullup and pulldown primitives modeled nets with

default logic levels when undriven. Delays were discussed, allowing the modeling of propagation times realistically using rise and fall delay parameters.

1.1.3 Testbench Creation and Verification

A testbench in Verilog is a wrapper module that applies stimuli to the Design Under Test (DUT) and observes outputs without synthesizing into hardware. Students learned to instantiate DUTs within the testbench, generate stimuli using initial blocks, and monitor responses using $display, $monitor, and $dumpvars commands. Good testbench practices were emphasized, including modular test structures using tasks and functions to ensure scalability and reusability. The introduction to writing self-checking testbenches, where the outputs are automatically compared against expected results, provided practical insight into robust verification techniques.

1.1.4 Advanced Gate Modeling and Wire Concepts

Tristate logic was particularly emphasized, allowing the construction of shared data buses where multiple devices can output to a single line under control conditions. Pullup and pulldown primitives were revisited to model default signal levels, ensuring stable behavior in the absence of active drivers. Gate delays were explored in greater depth using the min:typ:max specification, modeling best-case, typical-case, and worst-case propagation times, critical for performance and reliability analysis.

1.1.5 VCD Files, Specify Blocks, and Timing Analysis

To use $dumpfile and $dumpvars system tasks to record all signal transitions during simulation, enabling waveform analysis through visualization tools. VCD files simplify debugging complex simulations. Specify blocks were detailed, enabling explicit timing relationships between input and output ports. Using constructs like specparam, designers defined custom delay values. Timing checks, such as setup, hold, recovery, and removal times, were addressed, ensuring designs meet temporal constraints.

1.2 3 to 8 Decoder Specification

When this decoder is enabled with the help of enable input E, then it's one of the eight outputs will be active for each combination of inputs.

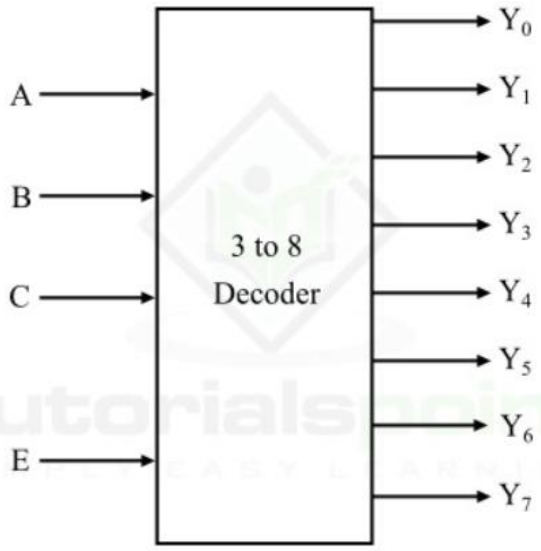

Figure 1.2 3 to 8 Decoder Specification

The operation of this 3-line to 8-line decoder can be analyzed with the help of its function table which is given below.

Inputs				Outputs							
E	A	B	C	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Table 1.2 3-line to 8-line decoder

1.2.1 Inputs

- A: This is one of the three input lines.
- B: This is another of the three input lines.
- C: This is the third of the three input lines.

1.2.2 Outputs

- Y0: This is the first of the eight output lines.
- Y1: This is the second of the eight output lines.
- Y2: This is the third of the eight output lines.
- Y3: This is the fourth of the eight output lines.
- Y4: This is the fifth of the eight output lines.
- Y5: This is the sixth of the eight output lines.
- Y6: This is the seventh of the eight output lines.
- Y7: This is the eighth of the eight output lines.

In overall, the 3-to-8 decoder takes three input bits (A, B, C). It produces eight output bits (Y0 through Y7).

1.2.3 Enable Signal

- "E" could be an enable input for the decoder.
- If "E" is active (usually high or 1), the decoder functions normally, and the outputs respond to the A, B, and C inputs.
- If "E" is inactive (low or 0), the decoder is disabled, and all outputs might be forced to a specific state (e.g., all low).

1.2.4 Error Signal

- Less likely, but "E" could represent an error output.
- The decoder might have some internal error detection mechanism, and "e" would signal if an error condition is detected.

Chapter 2

SYSTEMVERILOG DESIGN AND VERIFICATION

System Verilog is a comprehensive hardware description and verification language (HDVL) that extends Verilog. It bridges design and verification by adding high-level modeling, randomization, functional coverage, assertions, and OOP features. System Verilog enhances simulation performance and code maintainability, empowering engineers to model complex digital systems effectively.

2.1 Introduction to System Verilog and Module Basics

This session introduces the significance of SystemVerilog in modern digital design and verification. It explains how to structure hardware designs using modules, covering port types (input, output, inout) and syntax.

Key Concepts:

- Importance of SystemVerilog for design and verification.
- Defining simple modules for design modularity.
- Understanding ports and module connections

2.2 Net Data Types and Module Instantiation

This session focuses on net data types (wire, tri, wand, wor) and their role in representing hardware connections. It introduces module instantiation for reusable hierarchical designs.

Key Concepts:

- Using net types to model real-world connections.
- Instantiating modules to build hierarchical structures.
- Promoting design reusability and scalability.

2.3 Gate Level Modeling

This session covers gate-level modeling using predefined primitives (and, or, not, etc.).

Topics include multi-input/output gates, tristate gates with control signals, and pull gates.

Key Concepts:

- Using logic primitives for circuit modeling.
- Handling multiple input and output gates.
- Modeling tristate and pull behaviors.

2.4 Gate Delays and Wire Concepts

This session introduces gate delays (rise, fall, turn-off) to model real-world timing behavior. It explains wires and continuous assignments using the assign keyword.

Key Concepts:

- Simulating timing behavior via gate delays.
- Continuous assignment modeling with assign.
- Realistic simulation of combinational circuits.

2.5 Pre-defined Primitives and Testbench Creation

The final session explains the use of predefined primitives and basic testbench creation.

Testbenches simulate input conditions to verify the correctness of a design.

Key Concepts:

- Instantiating predefined primitives like gates.
- Writing simple testbenches for design validation.
- Effective verification using simulation.

Keyword	Explanation	Example
module	Encapsulates design functionality.	module adder; endmodule
wire	Represents a physical connection.	wire sum;
assign	Continuous assignment in dataflow modeling	assign sum = a ^ b;
and, or, not	Predefined gate-level primitives.	and G1 (out, in1, in2);
initial	Executes once at simulation start.	initial begin a=0; end
input, output	Define the direction of ports	input logic a;
#delay	Specifies a time delay.	#5 a = 1;
testbench	Module for testing other modules.	module tb; endmodule
primitive	Basic logic building block.	and (out, a, b);
tristate	Three-state output (Z, 0, 1).	bufif1(out, in, enable);
pull0, pull1	Pull wire value to 0 or 1.	pullup(p1); pulldown(p0);

Table 2 Keywords and Concepts

Chapter 3

VERILOG DESIGN

SystemVerilog, an extension of Verilog, is a powerful hardware description and verification language widely used in semiconductor and digital systems industries. It enhances traditional HDL (Hardware Description Language) features with advanced constructs for verification, design modeling, and testbench creation. The need for SystemVerilog arises from the ever-growing complexity of hardware designs where traditional techniques fall short in expressing high-level modeling, verification efficiency, and simulation management. Systematically explores SystemVerilog from basics to advanced Object-Oriented Programming (OOPs) concepts. The journey starts with simple gate-level modeling and gradually covers behavioral modeling, data types, procedural blocks, timing control, conditional statements, loops, tasks, functions, and introduces OOPs concepts essential for building reusable and scalable testbenches.

SystemVerilog's power lies in its abstraction mechanisms, which enable engineers to model not just signal transitions but complete transactions. With features like classes, inheritance, polymorphism, randomization, and coverage, it supports creating intelligent and adaptive testbenches that can handle industry-level SoC (System on Chip) verification.

3.1 Fundamentals of Verilog and Hardware Description Languages

The need for Hardware Description Languages (HDLs) to model digital systems at various levels of abstraction. We discussed Verilog's basic structure, the importance of modules, ports, and the hierarchy in hardware design.

- HDL (Hardware Description Language): Used to model digital systems.
- Module: Basic building block containing ports and internal logic.
- Ports: Communication interface (input, output, inout).

Laid the foundation of hardware design thinking using HDLs. We explored how modules serve as self-contained units, and how ports allow communication between these units.

3.2 Gate-Level Modeling and Structural Design

Gate-level modeling focuses on the physical realization of circuits using primitive logic gates such as AND, OR, NOT, etc.

- Gate-Level Modeling: Describing circuits by instantiating basic gates.
- Wire: Represents connections between components.
- Primitive Gates: Built-in gates like and, or, not, nand, nor, etc.

Gate-level modeling simulates real-world wiring and switching. Primitive gates and wires were introduced to create physical connections and logic functionalities.

3.3 Behavioral Modeling and Procedural Blocks

We moved from structure to behavior how a circuit behaves based on input stimuli. Behavioral modeling uses always and initial blocks to describe logic, timing, and conditions.

- Behavioral Modeling: Describes circuit behavior using procedural blocks.
- Initial Block: Executes once at simulation start.
- Always Block: Runs whenever events in sensitivity list occur.
- Reg: A variable type that holds values across simulation steps

Behavioral modeling introduced procedural thinking inside hardware descriptions.

3.4 Gate Delays, Tri-State Buffers, and Testbench Basics

To simulate realistic circuit behavior, this session added timing models using delays and introduced tri-state buffers for bus systems.

- Gate Delay: Simulates real-world propagation delay.
- Tri-State Buffer: Allows multiple drivers on a shared line.
- Testbench: Separate module to test design behavior.

This session bridged real-world circuit behavior with simulation. Concepts like gate delay and tri-state logic helped model shared buses, and writing a testbench started our journey into simulation and verification.

3.5 Data Types, Control Flow, and Loops

More control over execution was needed, so loops, conditionals, and more data types.

- Control Statements: Branching logic.
- Loops: Repeating code blocks.
- Integer and Real types: For computations.

This session made designs dynamic and conditionally responsive

3.6 Reusable Coding with Tasks and Functions

To avoid repeating code and improve readability, tasks and functions were introduced.

- Task: Multi-line block.
- Function: Single output computation.

Using tasks and functions made hardware descriptions modular and reusable.

3.7 SystemVerilog Enhancements and Randomization

SystemVerilog with new data types, randomization methods for test automation.

- Class: User-defined data types.
- Randomization: Random value assignment.
- Constructor (new): For initialization.

It introduced abstraction through classes and randomization.

3.8 Object-Oriented Programming (OOP) in SystemVerilog

This session formalized OOP principles like inheritance, polymorphism, and object copying.

- Object: Instance of a class.
- Handle: Pointer to object.
- Static Variable: Shared variable.
- Inheritance and Polymorphism.

It formalized verification abstraction through OOP concepts.

Keyword	Meaning	Example
Module	Hardware building block	module counter(...); endmodule
Ports	Interfaces (input/output)	input a, output b
Gate-Level Modeling	Structure of gates	and g1 (out, a, b);
Wire	Connection medium	wire a;
Behavioral Modeling	Logic behavior description	always @(posedge clk)
Gate Delay	Timing control	#(3) (out, a, b);
Tri-State Buffer	Bus sharing	bufif1 (bus, data, enable)
Loop	Repeated execution	for (i=0; i<10; i++)
Control Statement	Decision making	if (a>b)
Task	Modular code block	task reset; begin...end
Function	Single output computation	function add(a,b);
Class	Custom data type	class Packet; endclass
Randomization	Random value generation	pkt.randomize();
Object	Instance of class	Packet p = new();
Inheritance	Parent-child classes	class EthernetPacket extends Packet;

Table 3 Summary of the Keywords

Chapter 4

SIMULATION AND USE OF UVM

The Simulation and Use of UVM (Universal Verification Methodology) ensures that the 3-to-8 decoder functions correctly under various conditions.

4.1 Simulation & verification

Testbench Setup: The decoder is tested using a Verilog testbench, where different input values are applied.

Waveform Analysis: The simulated output waveforms confirm the expected behavior.

Verification Metrics: The design is checked for setup and hold violations, ensuring timing correctness**.**

- Input values are provided for 3-bit input and Enable signal.
- Output values 8-bit output.
- Multiple test cases are used to verify functionality

4.2 Simulation results

Expected Output:

en = 0; in = 3'b000;
en = 0; in = 3'b101;
en = 1; in = i;
en = 0; in = 3'b011;

Simulated Input-Output Waveforms

The design was simulated and the waveform confirmed correct for different test cases.

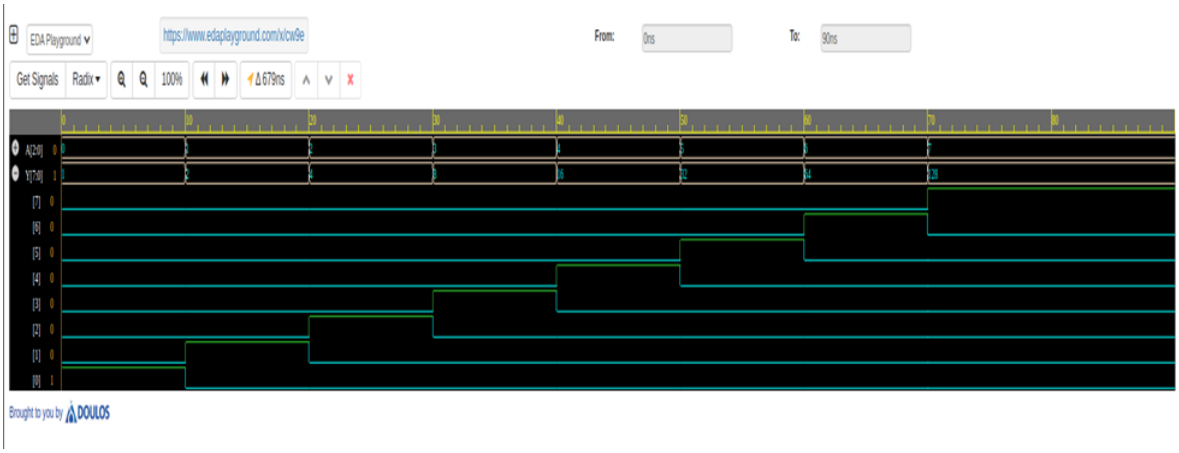

Figure 4.2 Simulation results Waveform 1

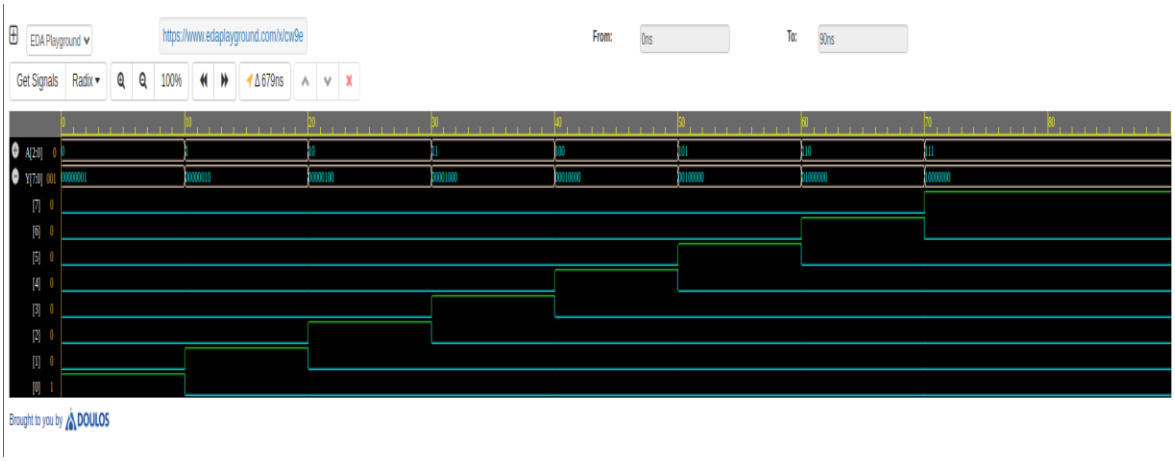

Figure 4.2 Simulation results Waveform 2

4.3 Do Functional Verification of the design using UVM

The Functional Verification of the 3-to-8 decoder using UVM (Universal Verification Methodology) ensures that the design operates correctly under various conditions.

4.3.1 Testbench Architecture (50%)

- Proper use of UVM components. Includes Driver, Monitor, Agent, Environment, and Test components
- Adherence to the UVM factory and configuration mechanism.
- Proper use of virtual sequences and sequence layering if applicable.

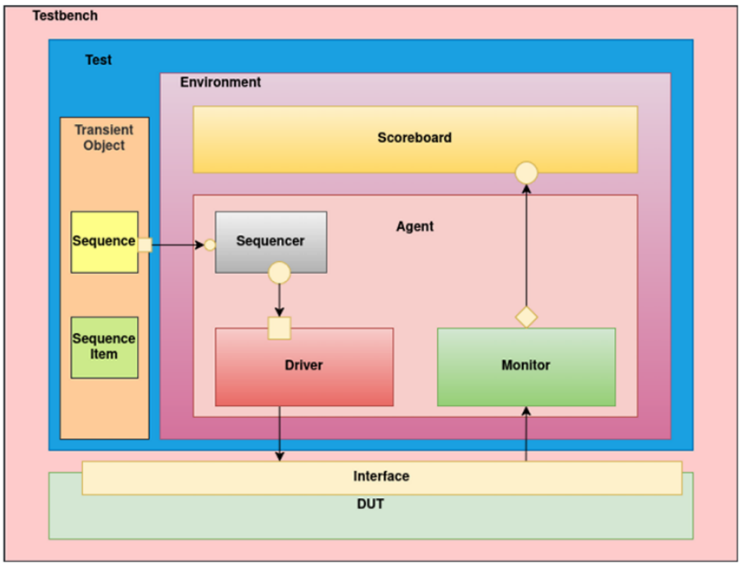

Figure 4.3.1 Testbench Architecture (50%)

4.3.2 Stimulus Generation (15%)

- Development of constrained-random and directed test sequences.
- Use of UVM sequences and transaction-based stimulus generation.
- Ability to generate different corner cases and invalid scenarios.
- Parameterization and reuse of sequences.

4.3.3 Score boarding and Checking (25%)

- Implementation of functional and self-checking scoreboard.
- Use of predictive models and golden reference comparison.
- Effective use of UVM phases for checking.

4.3.4 Debugging and Logs (5%)

- Effective use of UVM messaging and verbosity levels.
- Debugging skills and ability to interpret waveforms and logs.
- Error detection.
- Documentation of issues and resolutions.

4.3.5 Code Quality and Best Practices (5%)

- Consistency in naming conventions and coding style.
- Use of parameterized and reusable components.
- Proper comments and documentation within the code.
- Efficient and optimized coding practices.

The UVM-based verification confirms that the decoder design meets all functional requirements and operates correctly under all tested conditions

EDA Link:

https://www.edaplayground.com/x/CxT5

TAR file Link:

https://drive.google.com/drive/folders/1cj19AJeWJ3jUJ4r68ULSv954mquOVe4

Chapter 5

RESULTS

The 3 to 8 decoder was successfully implemented and verified. The simulation results matched the expected behavior, confirming the correctness of the design.

5.1 Layout of the Design

The Layout of the Design refers to the physical arrangement of the 3-to-8 decoder circuit after synthesis and placement using the OpenROAD tool. The layout was generated using the gf180 platform, ensuring optimized power and area utilization.

The OpenROAD tool was used to generate the GDS layout, which is the final representation of the decoder in a fabrication-ready format. This ensures that the design meets performance, power, and area constraints before manufacturing.

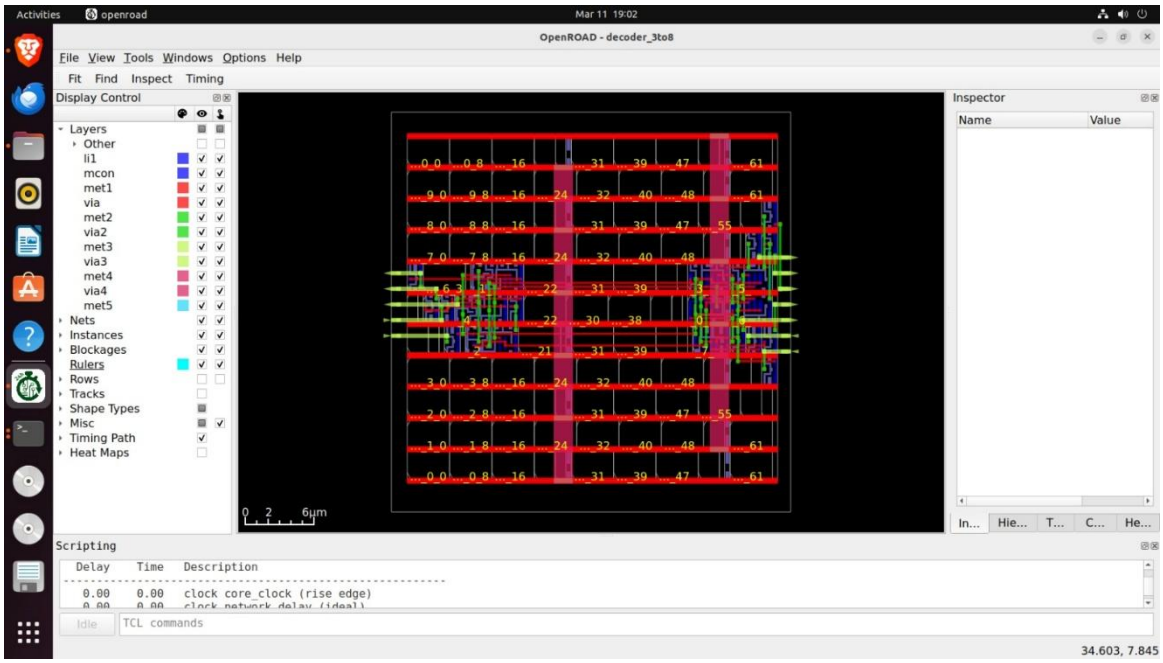

Figure 5.1 Layout of the Design

5.2 Performance Analysis

The Performance Analysis of the 3-to-8 decoder designed using Verilog HDL evaluates key metrics such as power consumption, area utilization, and timing performance.

5.2.1 Power Measurement

The Power Measurement evaluates the power consumption of the 3-to-8 decoder designed using Verilog HDL.

The decoder consumes 233nW, ensuring low-power operation. The power measurement confirms that the design is efficient and meets low-power requirements for digital applications.

```
>>> report_power
Group                  Internal  Switching    Leakage      Total
                          Power      Power      Power      Power (Watts)
----------------------------------------------------------------
Sequential             0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Combinational          1.41e-05   9.17e-06   5.64e-11   2.33e-05 100.0%
Clock                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Macro                  0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
Pad                    0.00e+00   0.00e+00   0.00e+00   0.00e+00   0.0%
----------------------------------------------------------------
Total                  1.41e-05   9.17e-06   5.64e-11   2.33e-05 100.0%
                          60.6%      39.4%       0.0%
```

Figure 5.2.1 Power Measurement

5.2.2 Area Measurement

The Area Measurement evaluates the physical space occupied by the 3-to-8 decoder after synthesis and layout generation. The decoder occupies 130 square micrometers. The design achieves 14% utilization, meaning it efficiently uses available chip space (Design area 130 u^2 14% utilization). The layout was generated using the gf180 platform, ensuring optimized area usage.

5.2.3 Timing Information

The Timing Information evaluates the setup and hold timing constraints of the 3-to-8 decoder designed using Verilog HDL.

- The analysis confirms no setup and hold violations, ensuring reliable operation.
- The design uses a 1.1 ns clock period, ensuring high-speed performance.
- The constraints set input delay at 0.22 ns and output delay at 0.22 ns, optimizing signal integrity.

```
  Delay    Time   Description
---------------------------------------------------------
   0.00    0.00   clock core_clock (rise edge)
   0.00    0.00   clock network delay (ideal)
   0.22    0.22 v input external delay
   0.00    0.22 v in[0] (in)
   0.11    0.33 v input2/X (sky130_fd_sc_hd__buf_1)
   0.28    0.61 ^ _0_/Y (sky130_fd_sc_hd__nor4b_1)
   0.09    0.70 ^ output5/X (sky130_fd_sc_hd__buf_1)
   0.00    0.70 ^ out[0] (out)
           0.70   data arrival time

   1.10    1.10   clock core_clock (rise edge)
   0.00    1.10   clock network delay (ideal)
   0.00    1.10   clock reconvergence pessimism
  -0.22    0.88   output external delay
           0.88   data required time
---------------------------------------------------------
           0.88   data required time
          -0.70   data arrival time
---------------------------------------------------------
           0.18   slack (MET)
```

Figure 5.2.3 Timing Information

5.3 Generated GDS

The layout was generated using the gf180 platform, ensuring optimized power and area utilization. The GDS layout was successfully created, making the design ready for fabrication.

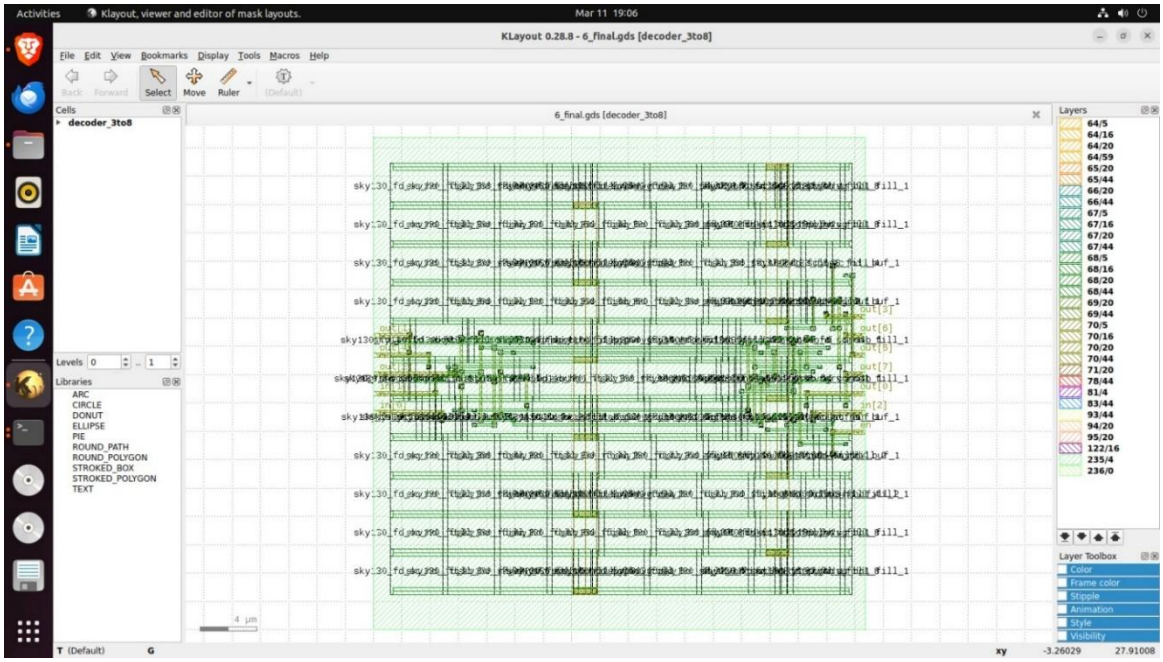

Figure 5.3 Generated GDS

Chapter 6

APPLICATIONS AND ADVANTAGES

Decoders are used in the cases where an output or a collection of outputs is to be activated only on the occurrence of a particular combination of input codes.

6.1 Applications

Some important applications of decoders are listed below –

- Decoders are used for code conversions.
- Decoders are extensively used in memory systems of computers.
- Decoders are also used for de-multiplexing or data distribution.
- Decoders are also used in data routing applications where very short propagation delay is required.
- Decoder may also be used for timing or sequencing purposes.
- Decoders are also utilized to turn on and off digital devices at a specific time.

6.2 Advantages

Designing a 3-to-8 decoder using Verilog HDL offers several advantages:

- **Efficient Hardware Implementation**: Verilog allows for direct synthesis into hardware, making the design efficient for FPGA and ASIC implementations.
- **Modular and Scalable**: The design can be easily modified or expanded to create larger decoders.
- **Ease of Simulation and Debugging**: Verilog provides built-in simulation capabilities, enabling thorough testing before hardware deployment.
- **Optimized Resource Utilization**: The design can be synthesized to minimize power consumption and area usage.
- **High-Speed Operation**: Verilog-based decoders can be optimized for speed, ensuring minimal propagation delay.
- **Integration with Other Digital Circuits**: The decoder can be seamlessly integrated into larger digital systems, such as memory addressing and data routing applications.

Chapter 7

CONCLUSIONS

In this report, the RTL code of 3 to 8 decoder has been designed in Verilog. The code is successfully verified with the UVM with 100% test case pass. The design code is further processed in the openROAD tool to generate its GDS using the gf180 platform. It has shown that the generated layout consumes 233nW power which occupies 130 sq. um area. There is no setup and hold violations.

REFERENCES

[1] Advanced Digital Design With the Verilog HDL, Michael D. Ciletti, 2nd Edition, PHI, ISBN: 978–0– 07–338054–4 2015.

[2] Digital Systems Design Using Verilog, Charles Roth, Lizy K. John, Byeong KilLee, Cengage Learning, ISBN-10: 1285051076, 2015.

[3] Fundamentals of Digital Logic with Verilog Design, Stephen Brown and Zvonko Vranesic, 6th Edition, McGraw Hill publication, ISBN: 978–0–07–338054–4, 2014.

[4] System Verilog for Design - A Guide to Using System Verilog for Hardware Design and Modeling, Stuart Sutherland, Simon David mann and Peter Flake, 2E, Springer Science, ISBN-13: 978-0387-3339-91, 2006.

[5] System Verilog for Verification-A Guide to Learning the Testbench Language Features, C Spear, Springer Science, IEEE press, ISBN-13: 978-0387-2703-64,2006.

[6] System Verilog golden reference guide-A concise guide to System Verilog Doulos, IEEE Standard-1800- 2009, Version 5.0,ISBN: 0-9547345-9-9, 2012.

[7] Step-by-Step Functional Verification with System Verilog and OVM, SasanIman, Hansen Brown Publishing Company,ISBN-13: 978-0-9816-5621-2, 2008.

[8] J. Y. Hwang, J. K. Kim, S. J. Moon, M. T. Hong, E. J. Yun and B. S. Bae, "BCD to 7-segment decoder with oxide thin film transistors", *Electron. Lett.*, vol. 52, no. 23, pp. 1952-1954, Nov. 2016.

[9] A. F. Thuslim and V. Kannan, *The Novel design method of 3 to 8 decoder*, vol. 10, pp. 92-95.

[10] R. Chakrabarty, S. Roy, T. Pathak, D. Ghosh and N. K. Mandal, "DESIGN OF 2:4 AND 3:8 DECODER CIRCUIT USING QCA TECHNOLOGY", *Наносистемы: физика химиямателматика*, vol. 12, no. 4, pp. 442-452, 2021.