Design and Layout of an 8x4 NOR-Type ROM using Magic VLSI

Madoori Sowmith 231EC228
Meghana R 231EC230
Dhanya D 231EC214
Department of Electronics and Communication Engineering
National Institute of Technology Karnataka (NITK)

Abstract

This report presents the design, implementation, and verification of an 8-word × 4-bit NOR-type Read-Only Memory (ROM) using the Magic VLSI layout tool. Each memory bit is implemented using a NOR-based pull-down transistor configuration with a PMOS pull-up network. IRSIM was used for functional verification using multiple word-line activation patterns. The report includes the architecture, working principle, cell design, layout details, simulation results, test vectors, and reference material required to understand the design flow.

1 Introduction

Read-Only Memory (ROM) is a non-volatile storage element used to store fixed logic values. In this project, an **8×4 NOR-type ROM** is designed at the transistor level and implemented completely in the Magic VLSI layout tool. The ROM uses:

- 8 word lines (WL0–WL7)
- 4 bit lines (BL0–BL3)
- A diffusion-based pull-down structure implementing logic '0'
- PMOS pull-up network always ON for restoring bit lines

This work demonstrates a memory design flow: schematic, layout, extraction, simulation.

2 Working Principle of NOR-Type ROM

The NOR-type ROM works on the following principle:

- Each cell contains an NMOS transistor connecting the bitline to ground.
- If the transistor is **present**, the stored bit is logic '0'.
- If the transistor is **absent**, the stored bit is logic '1'.
- When a word line is activated (WL = high), the NMOS transistors in that row can pull the bitline low.
- Bitlines are normally kept high by PMOS pull-up transistors.

3 1-Bit Cell Design

Two ROM bit-cell variants are created.

3.1 Cell Storing Logic '0'

A transistor connecting WL and BL diffusion is included. When WL activates, the BL is pulled to ground.

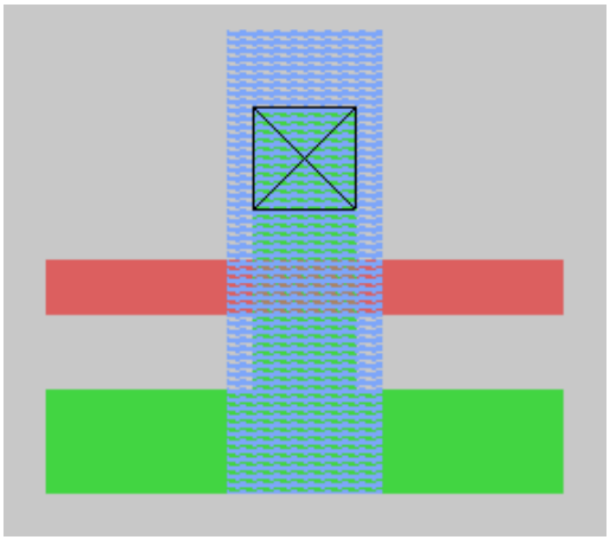

Figure 1: 1-bit ROM cell storing logic 0

3.2 Cell Storing Logic '1'

The transistor is omitted; BL remains pulled up to logic HIGH.

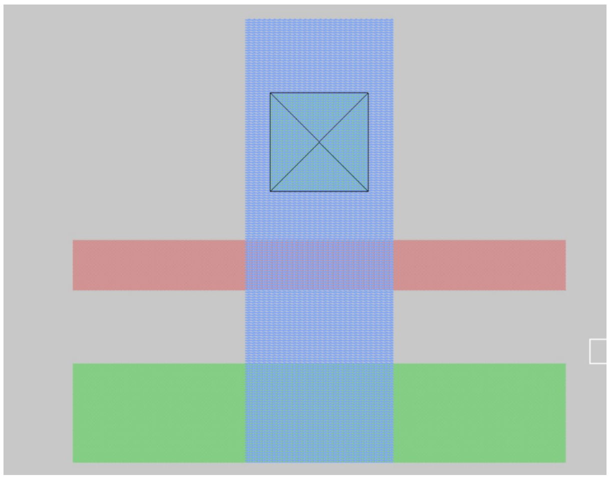

Figure 2: 1-bit ROM cell storing logic 1

4 ROM Array Architecture

The full ROM consists of:

- A row decoder (abstracted as activating WL0–WL7 manually)
- 8 parallel word lines

- 4 vertically routed bit lines with PMOS pull-ups
- 32 total ROM bit cells arranged as an 8×4 matrix

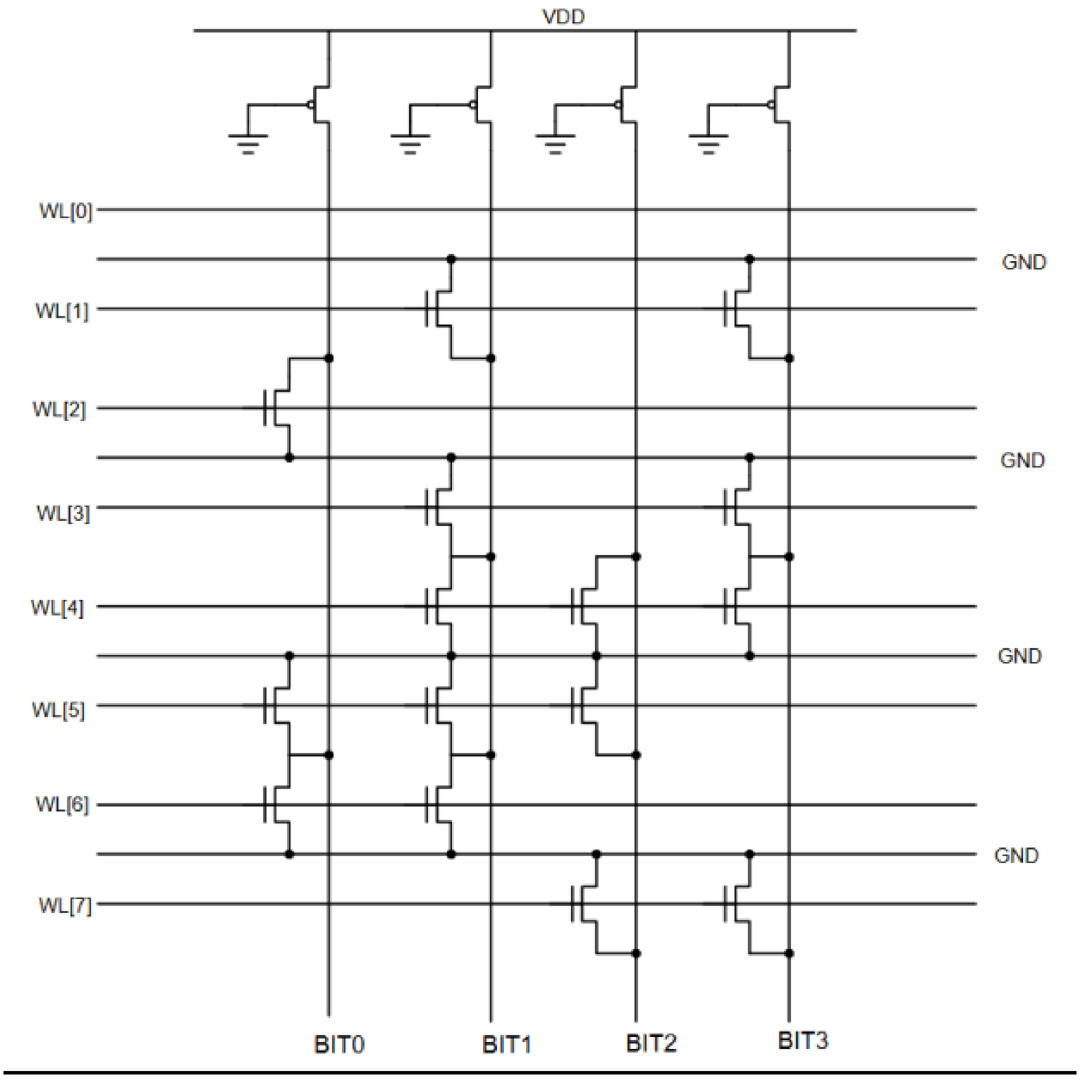

Figure 3: 8×4 ROM architecture block diagram

5 Magic Layout Design

The ROM layout was constructed using:

- Polysilicon for word lines
- Diffusion for NMOS pull-downs
- Metal1 for bit line routing
- PMOS pull-up at the top connected to VDD

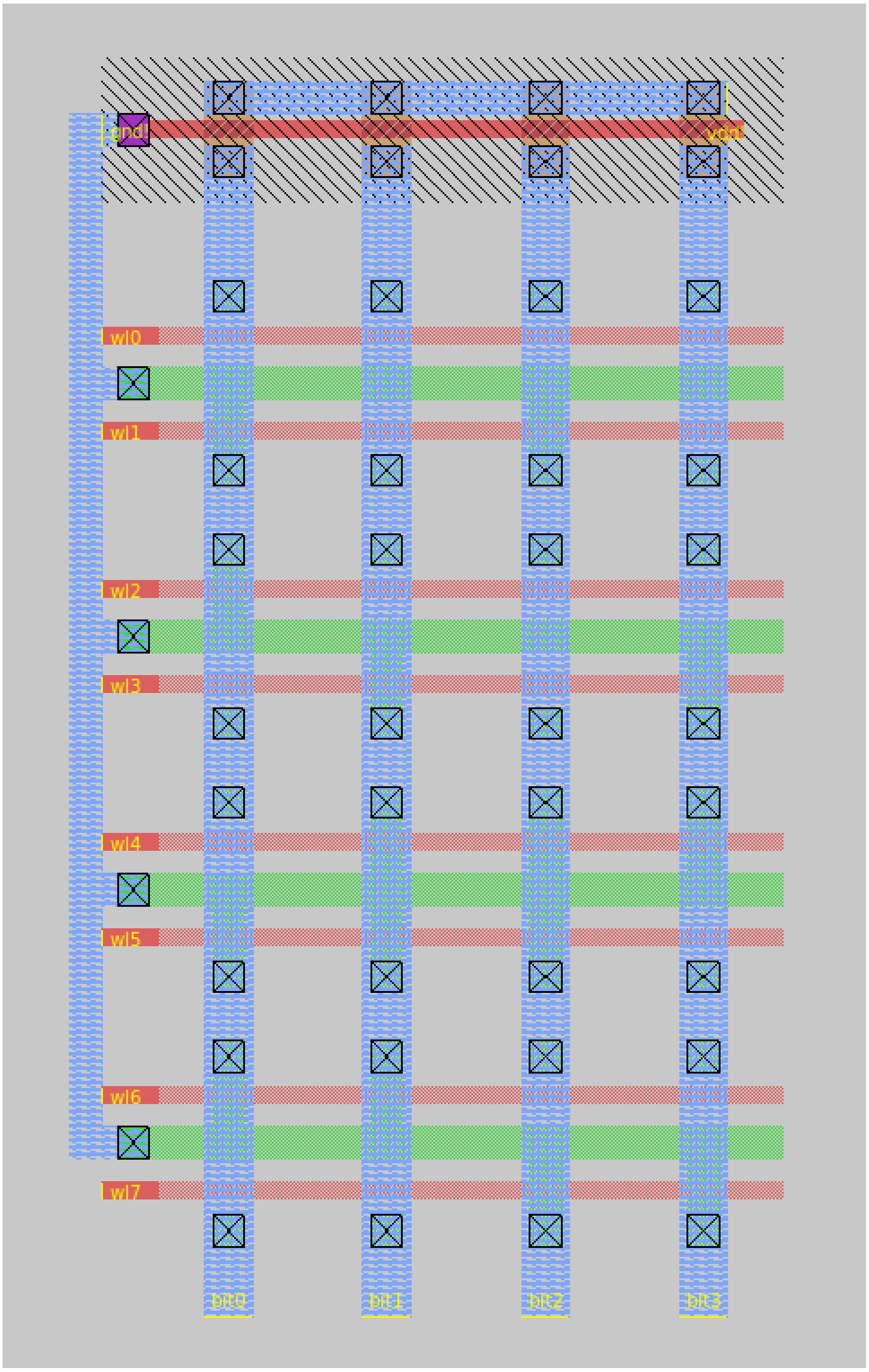

Figure 4: 8x4 ROM architecture layout

6 IRSIM Simulation and Verification

The simulation uses extracted `.sim` file from Magic.
Test cases were based on activating WL0, WL1, WL7, etc.

6.1 Example IRSIM Waveforms

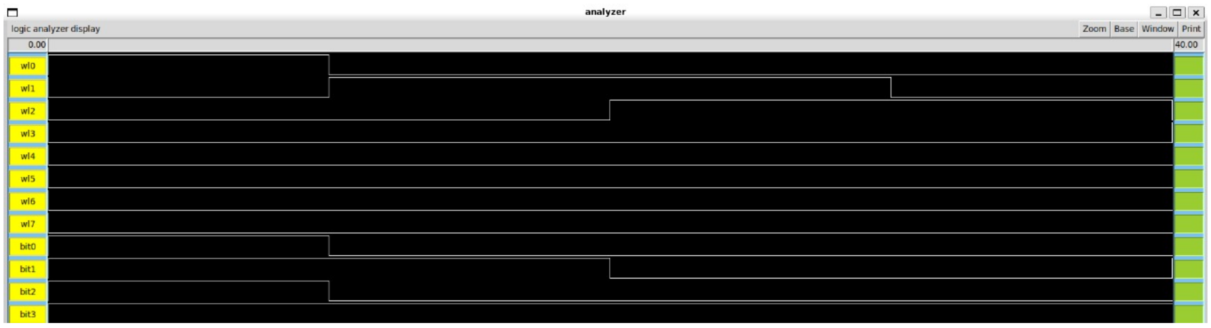

Figure 5: wl0, 1, 2 are active at t=0, 1, 2

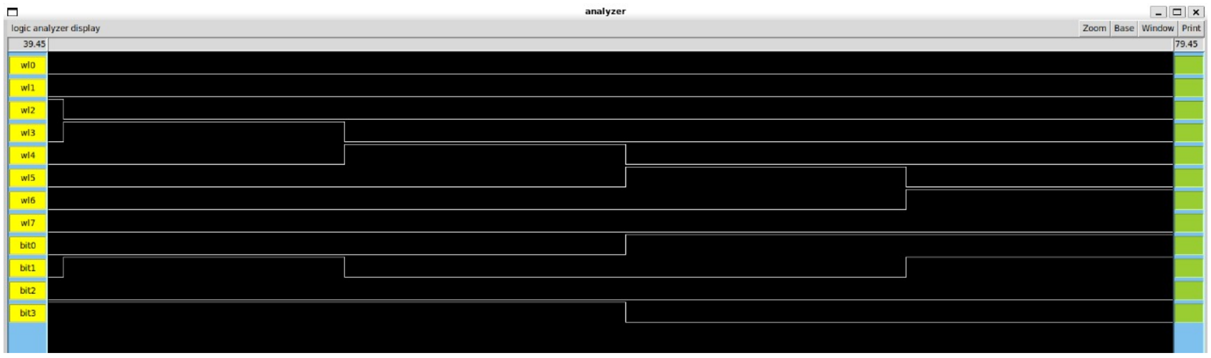

Figure 6: wl 3, 4, 5 are active

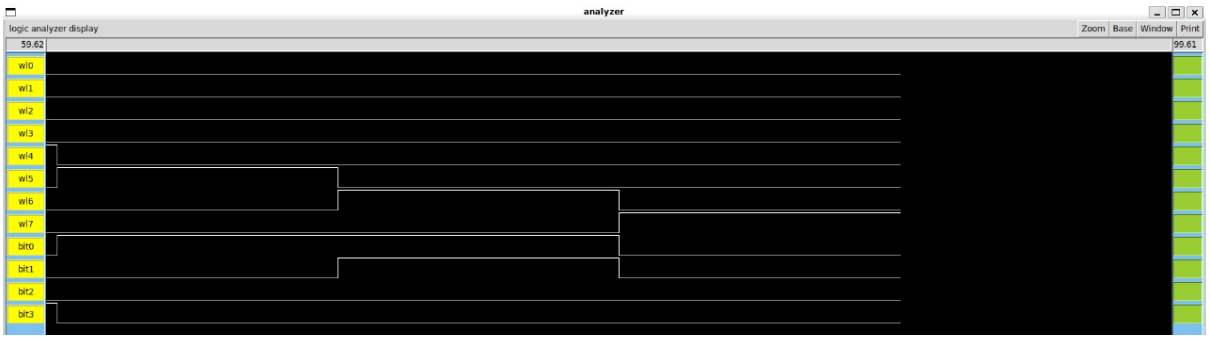

Figure 7: wl 6, 7 are active

6.2 Script to run the simulation

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
int main() {
    FILE *fp = popen("irsim final.sim > results.txt", "w");
    FILE *log = fopen("results.txt", "w");

    if (!fp || !log) {
        printf("Error: could not open IRsim or output file.\n");
        return 1;
    }

    fprintf(fp, "h vdd\n");
    fprintf(fp, "l gnd\n");

    for (int i = 0; i < 8; i++) {
        fprintf(log, "=== WL%d HIGH ===\n", i);

        for (int wl = 0; wl < 8; wl++) {
            if (wl == i)
                fprintf(fp, "h wl%d\n", wl);
            else
                fprintf(fp, "l wl%d\n", wl);
        }

        fprintf(fp, "s 50\n");

        fprintf(fp, "p bl0 bl1 bl2 bl3\n");
        fflush(fp);
    }

    fclose(log);
    pclose(fp);

    return 0;
}
```

7 Result Tabulation

Address (WL)	**Simulated Output**
000	1111
001	0101
010	1000
011	1010
100	1000
101	0001
110	0011
111	1100

Table 1: ROM simulation results comparison.

8 Conclusion

An 8$\times$4 NOR-type ROM was successfully designed, laid out, extracted, and simulated using the Magic VLSI toolchain. The two types of ROM bit cells (logic 0 and logic 1) were implemented correctly, and the final simulation results matched the expected data for all tested word lines. The project demonstrates foundational concepts in VLSI memory design and transistor-level layout methodology.

9 References

1. Magic VLSI Layout Tool, Open Circuit Design.

2. Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective*, Pearson.