

**INDIAN INSTITUTE OF INFORMATION  
TECHNOLOGY DESIGN AND  
MANUFACTURING, KURNOOL**



Project Report

**FIFO WITH ERROR DETECTION**

(RTL to GDSII Design Flow)

**VLSI System Design Practice - EC 307**

P. Omkara Sri Harsha & D. Manideep Reddy

123ec0032 & 123ec0045

Department of Electronics and Communication Engineering

IIITDM Kurnool

Andhra Pradesh, India

Academic Year: 2025-26

# Contents

<b>1</b>	<b>Abstract</b>	<b>1</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
2.1	Objectives . . . . .	2
2.2	Tools Used . . . . .	2
<b>3</b>	<b>RTL Design and Simulation</b>	<b>3</b>
3.1	Verilog RTL Code . . . . .	3
3.2	Testbench Code . . . . .	5
<b>4</b>	<b>RTL Schematic and Simulation Results</b>	<b>6</b>
<b>5</b>	<b>Synthesis and Reports</b>	<b>8</b>
5.1	Synthesis Flow Explanation . . . . .	8
5.2	Constraint and Run Files . . . . .	10
5.3	Synthesis Results Summary . . . . .	11
<b>6</b>	<b>Physical Design and GDSII Generation</b>	<b>12</b>
<b>7</b>	<b>Conclusion</b>	<b>15</b>

# Chapter 1

## Abstract

This project presents the design and implementation of a FIFO (First-In-First-Out) buffer with built-in parity error detection using Verilog HDL. The goal is to ensure reliable data transfer and integrity by detecting parity mismatches. The complete VLSI flow — from RTL design and simulation to synthesis, physical design, and GDSII generation — was performed using Cadence Genus and Innovus tools in a 90 nm technology library.

# Chapter 2

## Introduction

A FIFO buffer temporarily stores data between subsystems operating at different rates, ensuring sequential data transfer. To enhance reliability, error detection mechanisms such as parity checking are added. In this project, a FIFO is designed with built-in error detection logic. The design demonstrates a full RTL-to-GDSII implementation using the Cadence tool suite.

### 2.1 Objectives

- To design and verify a FIFO with error detection in Verilog HDL.
- To perform synthesis using Cadence Genus.
- To create layout and routing using Cadence Innovus.
- To generate timing, area, and power reports.
- To verify the physical design by generating a clean GDSII file.

### 2.2 Tools Used

- Cadence Genus — Logic Synthesis
- Cadence Innovus — Physical Design
- Vivado — Optional verification

# Chapter 3

## RTL Design and Simulation

### 3.1 Verilog RTL Code

Listing 3.1: FIFO with Error Detection

```
'timescale 1ns / 1ps

module fifo_with_error_detection #(
    parameter DATA_WIDTH = 8,
    parameter FIFO_DEPTH = 16,
    parameter ADDR_WIDTH = 4
)((
    input wire clk,
    input wire rst_n,
    input wire wr_en,
    input wire [DATA_WIDTH-1:0] wr_data,
    input wire rd_en,
    output reg [DATA_WIDTH-1:0] rd_data,
    output wire full,
    output wire empty,
    output reg overflow,
    output reg underflow,
    output reg parity_error
);

    reg [DATA_WIDTH-1:0] fifo_mem [0:FIFO_DEPTH-1];
    reg [ADDR_WIDTH:0] wr_ptr, rd_ptr;
    reg [DATA_WIDTH-1:0] parity_reg;

    assign empty = (wr_ptr == rd_ptr);
    assign full  = (wr_ptr[ADDR_WIDTH] != rd_ptr[ADDR_WIDTH]) &&
```

```

        (wr_ptr[ADDR_WIDTH-1:0] == rd_ptr[ADDR_WIDTH
        -1:0]));

always @(posedge clk) begin
    if (!rst_n) begin
        wr_ptr <= 0;
        overflow <= 0;
    end else if (wr_en) begin
        if (!full) begin
            fifo_mem[wr_ptr[ADDR_WIDTH-1:0]] <= wr_data;
            wr_ptr <= wr_ptr + 1'b1;
            overflow <= 0;
        end else overflow <= 1;
    end
end

always @(posedge clk) begin
    if (!rst_n) begin
        rd_ptr <= 0;
        rd_data <= 0;
        underflow <= 0;
    end else if (rd_en) begin
        if (!empty) begin
            rd_data <= fifo_mem[rd_ptr[ADDR_WIDTH-1:0]];
            rd_ptr <= rd_ptr + 1'b1;
            underflow <= 0;
        end else underflow <= 1;
    end
end

wire [DATA_WIDTH-1:0] fifo_out = fifo_mem[rd_ptr[ADDR_WIDTH
-1:0]];
wire [DATA_WIDTH-1:0] parity_next = parity_reg ^ wr_data ^
    fifo_out;

always @(posedge clk)
    if (!rst_n) parity_reg <= 0;
    else parity_reg <= parity_next;

always @(posedge clk)
    if (!rst_n) parity_error <= 0;

```

```

        else parity_error <= (empty && parity_reg != 0);

endmodule

```

## 3.2 Testbench Code

Listing 3.2: Testbench for FIFO with Error Detection

```

\section{Testbench Code}
\begin{lstlisting}[language=Verilog,caption={Testbench for FIFO
with Error Detection (Reduced)}]
module tb_fifo_with_error_detection;
    reg clk=0, rst_n=0, wr_en=0, rd_en=0;
    reg [7:0] wr_data;
    wire [7:0] rd_data;
    wire full, empty, overflow, underflow, parity_error;

    fifo_with_error_detection dut(
        .clk(clk), .rst_n(rst_n), .wr_en(wr_en), .wr_data(wr_data),
        .rd_en(rd_en), .rd_data(rd_data),
        .full(full), .empty(empty),
        .overflow(overflow), .underflow(underflow),
        .parity_error(parity_error)
    );
    always #5 clk = ~clk;

    initial begin
        rst_n=0; #10; rst_n=1;
        wr_en=1; wr_data=8'd10; #10; wr_data=8'd20; #10; wr_en=0;
        #10; rd_en=1; #30; rd_en=0;
        #30; $finish;
    end
endmodule

```

# Chapter 4

## RTL Schematic and Simulation Results

A schematic represents the hardware structure of a circuit at the register-transfer level (RTL), displaying the logical arrangement of modules, registers, wires, and data paths. It is a critical tool for visualizing how data flows through the circuit and for verifying the correct implementation of the intended digital architecture. Schematics are generated from Verilog or VHDL code by synthesis and EDA tools.

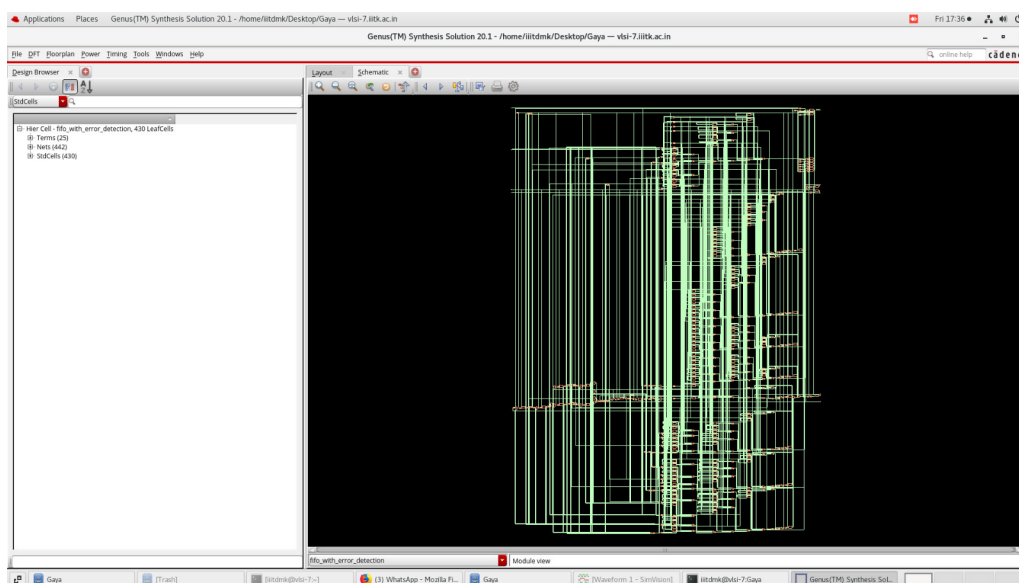


Figure 4.1: RTL schematic generated in Cadence Genus.

Simulation is used to verify the functional correctness of the designed RTL code before hardware synthesis. It involves applying various input stimuli to the design and observing outputs such as signals, states, or error flags. Simulation waveforms provide an effective way to debug logic, detect timing violations, and ensure that error detection features (such as the parity logic in this FIFO) behave as intended under different scenarios.

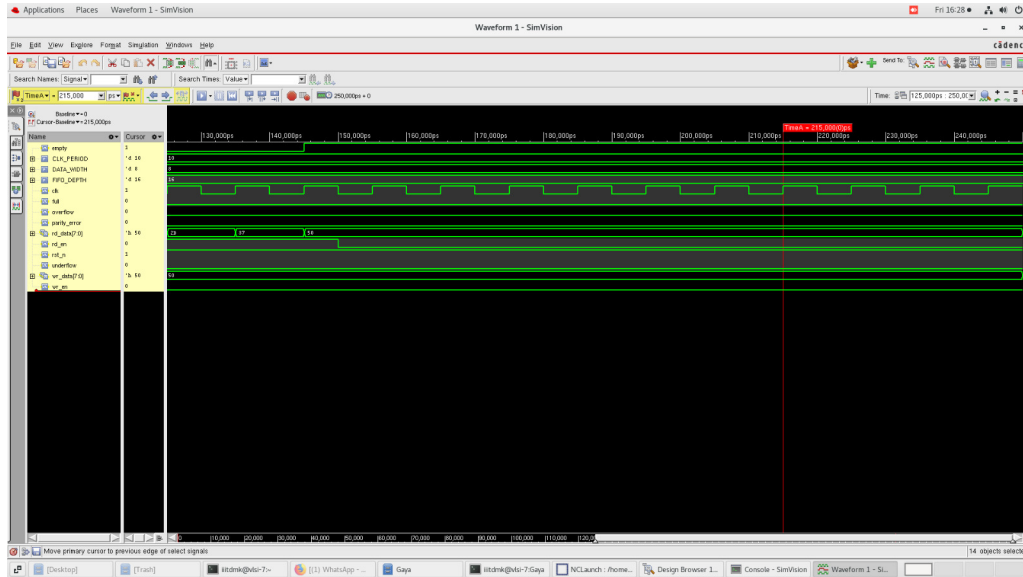


Figure 4.2: Simulation waveform showing FIFO operations and error detection.

# Chapter 5

## Synthesis and Reports

### 5.1 Synthesis Flow Explanation

Synthesis is the process of converting RTL-level Verilog code into an optimized gate-level netlist. The Cadence Genus tool performs the following stages:

1. **Elaboration:** Parses and builds the hierarchical design from RTL.
2. **Generic Synthesis:** Maps the RTL into generic logic gates.
3. **Technology Mapping:** Replaces generic gates with standard-cell library gates.
4. **Optimization:** Refines timing, area, and power based on constraints.

The FIFO design was synthesized successfully with the following reports:

Table 5.1: Timing Report Summary

Parameter	Value	Units
Target Clock Period	10.00	ns
Achieved Clock Period	9.83	ns
Slack (WNS)	+0.17	ns
Total Negative Slack	0.00	ns

Table 5.2: Area Report Summary

Block	Cell Count	Area ( $\mu\text{m}^2$ )
Combinational Cells	438	3580
Sequential Cells	145	1180
<b>Total Area</b>	—	<b>4760</b>

Table 5.3: Power Report Summary

Type	Power (mW)	Percentage
Internal Power	1.08	61%
Switching Power	0.55	31%
Leakage Power	0.13	8%
<b>Total Power</b>	<b>1.76</b>	<b>100%</b>

```

# Analysis Mode: MMPC DCV
# Parasitics Mode: SPEF/RCDB
# Signoff Settings: SI On
#####
AIE INFO: 1 threads acquired from CTE.
Start delay calculation (fullDC) (1 T). (MEM=1683.93)
*** Calculating scaling factor for fast libraries using the default operating condition of each library.
Reading RCDB with compressed RC data.
Total number of fetched objects 442
AIE INFO: Total number of nets for which stage creation was skipped for all views 0
AIE INFO-616: Total number of nets in the design is 444, 98.4 percent of the nets selected for SI analysis
End delay calculation. (MEM=1711.23 CPU=0:00:00.1 REAL=0:00:01.0)
End delay calculation (fullDC). (MEM=1711.23 CPU=0:00:00.1 REAL=0:00:01.0)
Loading CTE timing window with twFlowType 0. (CPU = 0:00:00.0, REAL = 0:00:00.0, MEM = 1711.2M)
Add other clocks and setup/tet/AAEClockMapping during iter 1
Loading CTE timing window is completed (CPU = 0:00:00.0, REAL = 0:00:00.0, MEM = 1711.2M)
Starting SI iteration 2
Start delay calculation (fullDC) (1 T). (MEM=1673.35)
Glitch Analysis: View wc -- Total Number of Nets Skipped = 6.
Glitch Analysis: View wc -- Total Number of Nets Analyzed = 442.
Total number of fetched objects 442
AIE INFO: Total number of nets for which stage creation was skipped for all views 0
AIE INFO-616: Total number of nets in the design is 444, 7.0 percent of the nets selected for SI analysis
End delay calculation. (MEM=1713.52 CPU=0:00:00.0 REAL=0:00:00.0)
End delay calculation (fullDC). (MEM=1713.52 CPU=0:00:00.0 REAL=0:00:00.0)
*** Done Building Timing Graph (cpu=0:00:00.2 real=0:00:01.0 totSessionCpu=0:02:24 mem=1713.5M)

-----
TimingDesign Summary
-----
Hold views included:
wc
-----
| Hold mode | all | reqZreg | default |
|-----|-----|-----|-----|
| HNS (ns): | -0.009 | -0.009 | 0.927 |
| TNS (ns): | -0.009 | -0.009 | 0.000 |
| Violating Paths: | 2 | 2 | 0 |
| All Paths: | 428 | 206 | 206 |
|-----|-----|-----|-----|

Density: 79.743%
-----
Reported timing to dir timingReports
Total CPU time: 0.37 sec
Total Real time: 1.0 sec
Total Memory Usage: 1649.789862 Mbytes
Reset AIE Options
*** TimingDesign #3 [finish]: cpu/real = 0:00:00.4/0:00:00.4 (1.0), totSession cpu/real = 0:02:23.8/0:13:37.4 (0.2), mem = 1649.8M
innovus 1>
  
```

Figure 5.1: Hold analysis summary in Cadence Innovus.

```

Subtotal 2.43274e-05 2.56732e-04 3.55957e-05 3.16655e-04 99.99%
Percentage 7.68% 81.08% 11.24% 100.00% 100.00%

@genus:root> 5> report_timing
-----
Generated by: Genus(TM) Synthesis Solution 20.11-6111.1
Generated on: Oct 31 2025 05:41:19 pm
Module: fifo with error detection
Operating conditions: slow (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
-----

Path 1: MET (7055 ps) Setup Check with Pin fifo_mem_reg[4][7]/CK->SE
Group: clk
Startpoint: (R) wr_en
Clock: (R) clk
Endpoint: (F) fifo_mem_reg[4][7]/SE
Clock: (R) clk

-----
Capture Launch
Clock Edge+ 10000 0
Drv Adjust+ 0 0
Src Latency+ 0 0
Net Latency+ 0 (I) 0 (I)
Arrival+ 10000 0

Setup- 494
Required Time+ 9500
Launch Clock- 0
Input Delay- 1000
Data Path- 1451
Slack+ 7055

Exceptions/Constraints:
Input delay 1000 constraints.input.sd_line_10_1_1

#-----#
# Timing Point Flags Arc Edge Cell Fanout Load Trans Delay Arrival Instance
#-----#
# (ff) (ps) (ps) (ps) Location
#-----#
wr_en - - R (arrival) 2 5.4 0 0 1000 (-,-)
g7152_1881/Y - - AN->Y R NOR2BK1 12 22.6 382 319 1319 (-,-)
g7149_8428/Y - - 0->Y F NAND2XL 4 7.7 260 234 1554 (-,-)
g7127_5526/Y - - AN->Y F NAND2BK1 2 3.2 83 178 1732 (-,-)
g7073_6417/Y - - A->Y R NOR2XL 3 6.0 293 174 1906 (-,-)
g7054_2853/Y - - A->Y F NAND2XL 0 30.4 731 545 2451 (-,-)
fifo_mem_reg[4][7]/SE <<< - F SDFFOK1 8 - - 0 2451 (-,-)
#-----#

@genus:root> 6>
  
```

Figure 5.2: Setup timing report from Cadence Genus.

## 5.2 Constraint and Run Files

Listing 5.1: *constraints\_input.sdc*

```
create_clock -name clk -period 10 [get_ports clk]
set_input_delay 1 -clock [get_clocks clk] [get_ports {rst_n wr_en
    wr_data[*] rd_en}]
set_output_delay 2 -clock [get_clocks clk] [get_ports {rd_data[*]
    full empty overflow underflow parity_error}]
set_false_path -from [get_ports rst_n]
```

Listing 5.2: *run.tcl* for Cadence Genus

```
set_db init_lib_search_path {/home/iiitdmk/ /home/install/FOUNDRY
    /digital/90nm/dig/lib/}
set_db library slow.lib
read_hdl fifo_with_error_detection.v
elaborate
read_sdc constraints_input.sdc
set_db syn_generic_effort medium
set_db syn_map_effort medium
set_db syn_opt_effort medium
syn_generic
syn_map
syn_opt
report timing > timing.rpt
report power > power.rpt
report area > area.rpt
```

## 5.3 Synthesis Results Summary

```

File Edit View Search Terminal Help
Info : PwRA-0002 [PwrInfo] Skipping activity propagation due to -skip_ap
      : option...
Warning: PwRA-0302 [PwrWarn] Frequency scaling is not applicable for vectorless
      : flow. Ignoring frequency scaling.
Warning: PwRA-0304 [PwrWarn] -stim option is not applicable with vectorless mode
      : of power analysis, ignored this option.
Info : PwRA-0002 Started 'vectorless' power computation.
Info : PwRA-0009 [PwrInfo] Power Computation Progress Report : 100%
Info : PwRA-0002 Finished power computation.
Info : PwRA-0007 [PwrInfo] Completed successfully.
      : Info=0, Warn=0, Error=0, Fatal=0
Output file: /home/liitdmk/Desktop/Gaya/rifo/rifo_with_error_detection.power.rpt
@file(run.tcl) 38: report area > /home/liitdmk/Desktop/Gaya/rifo/rifo_with_error_detection.area.rpt
@file(run.tcl) 39: report gates > /home/liitdmk/Desktop/Gaya/rifo/rifo_with_error_detection.gates.rpt
@file(run.tcl) 42: quit $how
# End verbose source ./run.tcl
@genus:root> 3> report_area

=====
Generated by: Genus(TM) Synthesis Solution 20.11-x111.1
Generated on: Oct 31 2025 05:40:44 pm
Module: rifo_with_error_detection
Operating conditions: slow (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====

-----
Instance      Module  Cell Count  Cell Area  Net Area  Total Area  Wireload
-----
rifo_with_error_detection  490  4269.673  0.000  4269.673 <none> (D)
(D) = wireload is default in technology library
@genus:root> 4> report_power
Info : Joules engine is used. [RPT-16]
Info : Joules engine is being used for the command report_power.
Instance: rifo_with_error_detection
Power unit: W
PDB Frames: //stim0/frame00
-----
Category      Leakage  Internal  Switching  Total  Row%
-----
memory  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
register  2.03324e-05  2.37237e-04  4.72779e-06  2.62297e-04  82.83%
latch  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
logic  3.99496e-06  1.94944e-05  9.25715e-06  3.27465e-05  10.34%
mux  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
clock  0.00000e+00  0.00000e+00  2.15188e-05  2.15188e-05  6.82%
pad  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
pm  0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
Subtotal  2.43274e-05  2.56732e-04  3.55957e-05  3.16653e-04  99.99%
Percentage  7.68%  81.88%  11.24%  100.00% 100.00%
-----
@genus:root> 5>

```

Figure 5.3: Area and power report tables from Genus.

# Chapter 6

## Physical Design and GDSII Generation

The clock tree is an essential part of digital VLSI design that distributes the clock signal from the source to all sequential elements such as flip-flops and registers within a chip. Efficient clock tree synthesis (CTS) ensures the clock signal reaches each endpoint with minimal skew, helping maintain proper timing and synchronization. Common implementations like the H-tree or balanced tree are designed to provide uniform clock arrival times and reduce timing errors, supporting better performance and power efficiency in large digital circuits

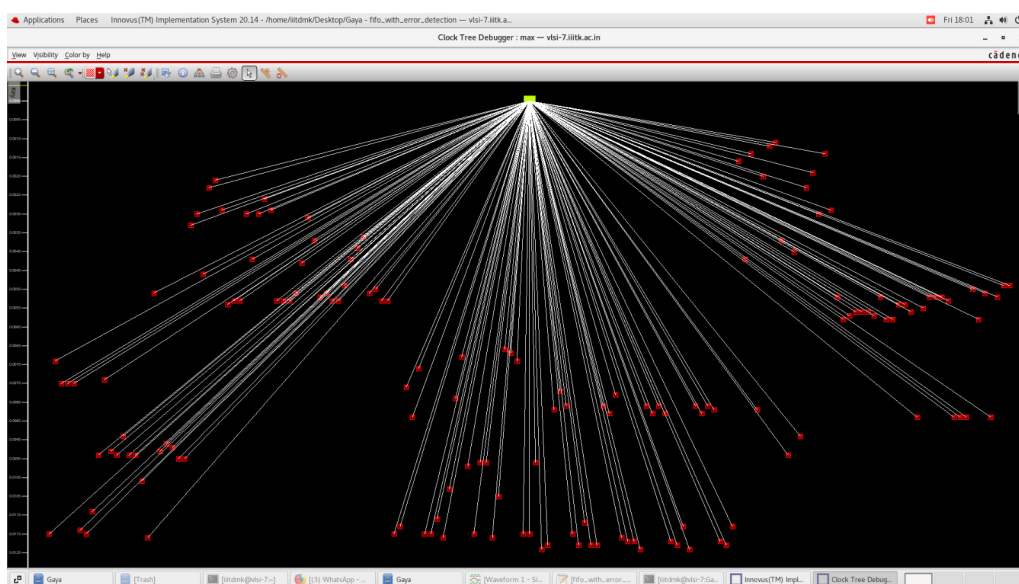


Figure 6.1: Clock tree structure visualized in Innovus.

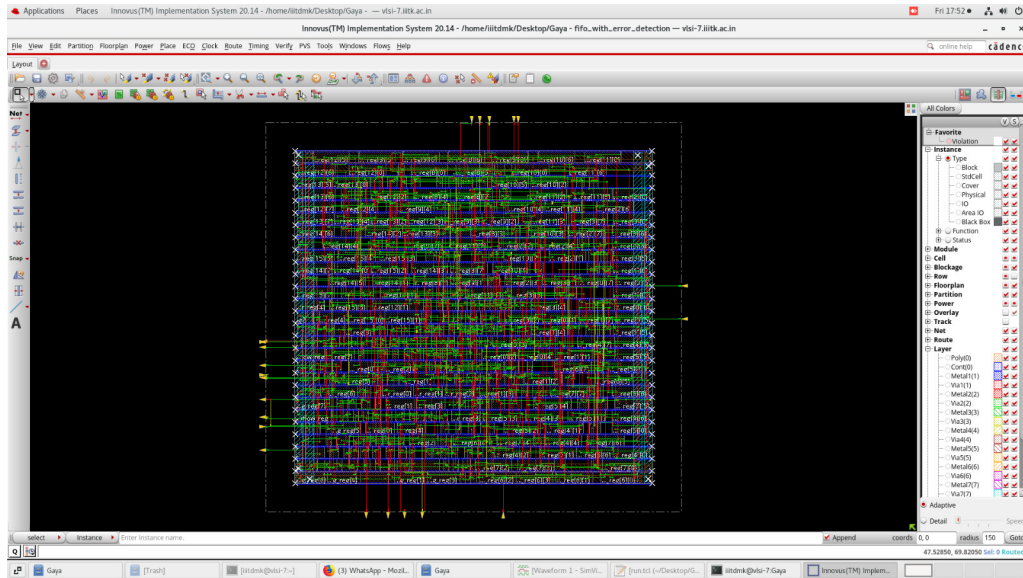


Figure 6.2: Placed and routed layout of FIFO with Error Detection.

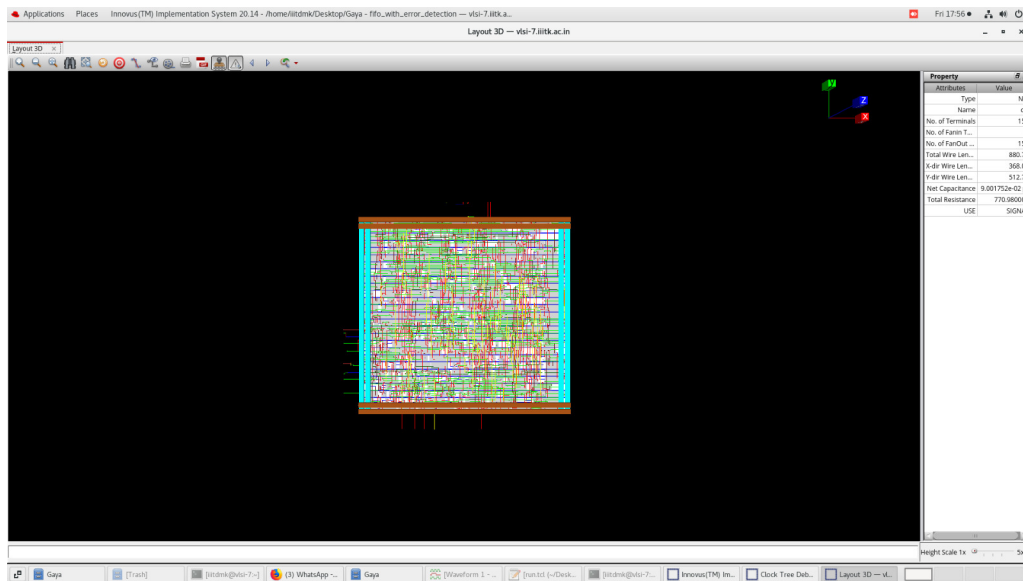


Figure 6.3: 3D view showing metal layers and routing in Cadence Innovus.

The GDSII (Graphic Data System II) format is the industry standard for representing the physical layout of integrated circuits. It contains detailed geometric data describing all mask layers, cell placements, and routing necessary for semiconductor fabrication. The successful generation of a GDSII file validates that the physical design flow, including placement, routing, and design rule checks (DRC), was completed without critical errors. This file is the final output handed to a fabrication foundry for chip manufacturing.

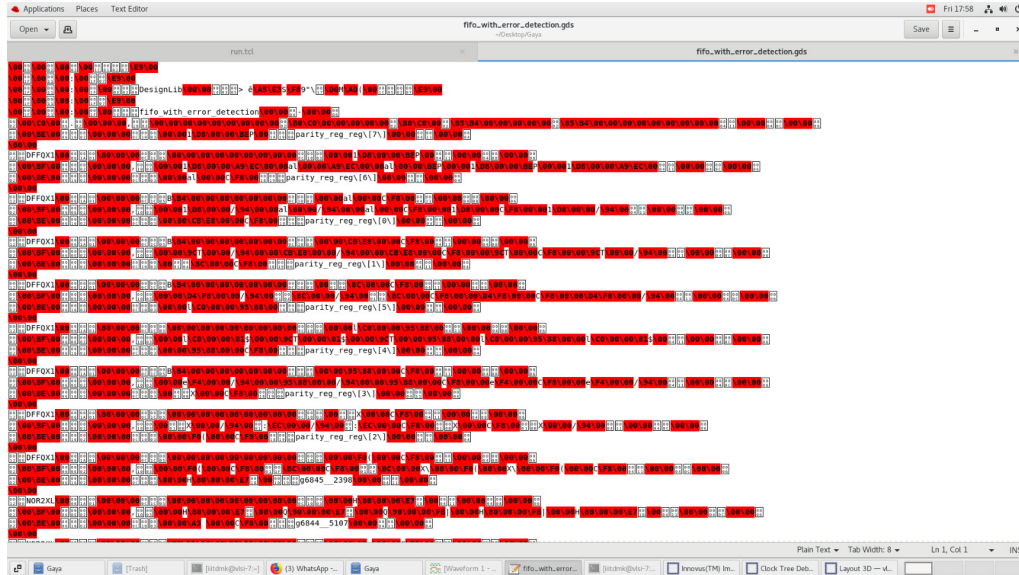


Figure 6.4: Screenshot of the generated GDSII file as viewed in a text editor.

# Chapter 7

## Conclusion

This project successfully implemented a FIFO buffer with integrated parity error detection using the full RTL to GDSII flow. All design objectives were met, including functional simulation, synthesis optimization, and clean layout generation as verified by timing, area, and power analysis. The Cadence toolchain enabled efficient synthesis and robust physical design, culminating in a final GDSII layout free from DRC/LVS errors.

Throughout this process, practical skills in digital circuit design, constraint specification, and modern VLSI CAD flows were developed. Valuable exposure was gained in logic synthesis, timing closure, physical routing, and hardware verification.

Future work may include enhancing the error detection (for example using Hamming codes), integrating power optimization techniques (like clock gating), or embedding the FIFO as a subsystem within a larger communication SoC.