

DAA Multiplier Semi-Custom (RTL to GDS) Design

A Mini Project Report

Submitted by

Spoorthi

Indian Institute of Information Technology, Kurnool



Under the guidance of

Associate professor Ranga Babu, Department of ECE

November 2025

Abstract

The DAA (Distributed arithmetic algorithm) Multiplier is implemented using a semi-custom VLSI design flow, beginning from Register Transfer Level (RTL) description and continuing through synthesis, placement, routing, and GDSII generation. The main objective is to design a low-power and low-complexity multiplier using the Cadence suite of tools. The design is optimized for performance, area, and power efficiency. The project demonstrates the complete RTL-to-GDSII flow, including constraint application, synthesis reports, timing analysis, and layout generation.

Contents

1	Introduction	3
1.1	Objectives	3
1.2	Tools Used	3
2	RTL Design and Simulation	4
2.1	RTL Code	4
2.2	Testbench	5
2.3	Schematic	6
3	Synthesis and Constraint Setup	7
3.1	Constraint and Tcl Files	7
3.2	Synthesis Reports	8
4	Physical Design and GDS Generation	10
4.1	3D View	11
5	Conclusion	12

Chapter 1

Introduction

A multiplier is a key arithmetic circuit used in many signal processing and embedded systems. The DAA (Distributed Arithmetic Algorithm) multiplier uses a simplified logic approach to reduce complexity compared to conventional array multipliers. This project demonstrates a complete semi-custom implementation using industry-standard EDA tools, beginning from Verilog design up to the generation of a verified GDSII file.

1.1 Objectives

- To design and simulate the DAA multiplier in Verilog HDL.
- To perform synthesis and analyze timing and area reports.
- To implement physical design (place and route) and generate GDSII.
- To verify power and timing post-layout.

1.2 Tools Used

- Cadence Genus – Synthesis
- Cadence Innovus – Physical Design

Chapter 2

RTL Design and Simulation

2.1 RTL Code

The RTL design of the DAA multiplier is implemented. The design was simulated using in nclaunch simulation.

```
1 module daa_multiplier (
2     input clk,
3     input reset,
4     input [7:0] A,
5     input [7:0] B,
6     output reg [15:0] result
7 );
8     integer i;
9
10    always @(posedge clk or posedge reset) begin
11        if (reset) begin
12            result <= 16'b0;
13        end else begin
14            result <= 16'b0; // reset result at every clock
15            for (i = 0; i < 8; i = i + 1) begin
16                if (B[i])
17                    result <= result + (A << i);
18            end
19        end
20    end
21 endmodule
```

2.2 Testbench

The testbench was written to verify functionality for multiple input cases. Correct output waveforms confirmed logical accuracy.

```
1
2 module testbench;
3
4     reg clk;
5     reg reset;
6     reg [7:0] A;
7     reg [7:0] B;
8     wire [15:0] result;
9
10    daa_multiplier uut (
11        .clk(clk),
12        .reset(reset),
13        .A(A),
14        .B(B),
15        .result(result)
16    );
17
18    initial begin
19        clk = 0;
20        reset = 1;
21        A = 0;
22        B = 0;
23        #15 reset = 0;
24
25        // Test case 1
26        A = 8'h0F; // 15 decimal
27        B = 8'h03; // 3 decimal
28        #20;
29        $display("Test 1 result = %d", result); // Expected 45
30
31        // Test case 2
32        A = 8'hFF; // 255 decimal
33        B = 8'h02; // 2 decimal
34        #20;
35        $display("Test 2 result = %d", result); // Expected 510
```

```

36
37 // Add more test cases...
38
39 #10 $finish;
40 end
41
42 always #5 clk = ~clk; // 10 time unit clock period
43
44 endmodule

```

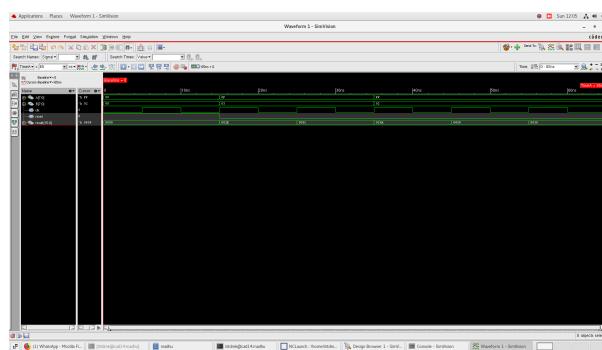


Figure 2.1: Simulation waveform of DAA Multiplier in nclaunch

2.3 Schematic

The generated RTL schematic from Genus is shown in Figure 2.2.

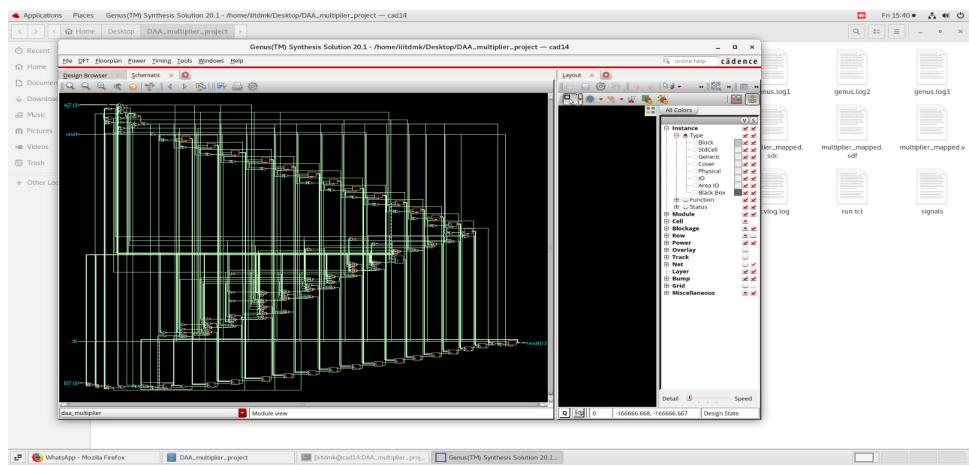


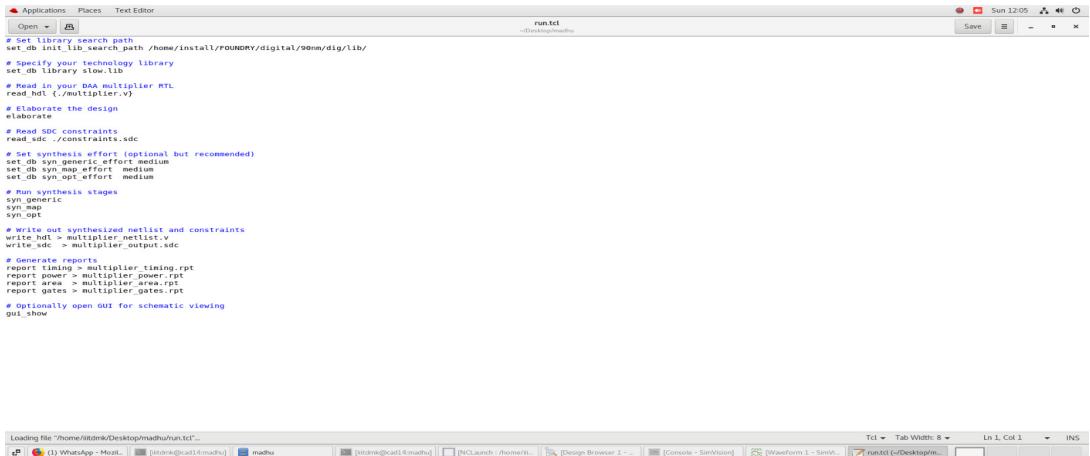
Figure 2.2: Schematic view of synthesized DAA Multiplier

Chapter 3

Synthesis and Constraint Setup

3.1 Constraint and Tcl Files

Design constraints were written in TCL format to define input/output delays, clock period, and design environment.



The screenshot shows a terminal window titled "run.tcl" with the following content:

```
#!/usr/bin/tclsh
set db [db::init lib_search_path /home/install/rounDMY/digital/90nm/dig/lib]
# Specify your technology library
set db_library slow.lib
# Read in your DAA multiplier RTL
read_digital ./multiplier.v
# elaborate the design
elaborate
# Read SDC constraints
read_sdc ./constraints.sdc
# Set synthesis effort (optional but recommended)
set db syn generic_effort medium
set db syn map_effort medium
set db impl_effort medium
# Run synthesis stages
syn generic
syn map
syn opt
# Write out synthesized netlist and constraints
write_sdc > multiplier.netlist.v
write_sdc > multiplier.output.sdc
# Generate reports
report timing > multiplier_timing.rpt
report area > multiplier_power.rpt
report area > multiplier_area.rpt
report gates > multiplier_gates.rpt
# Optionally open GUI for schematic viewing
gui show
```

Figure 3.1: Screenshot of run.tcl file used for synthesis

```

set_db init_lib_search_path /home/install/FOUNDRY/digital/90nm/dig/lib/
set_db library slow.lib

read_hdl {./counter.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

write_hdl > counter_netlist.v
write_sdc > counter_output.sdc

report timing > counter_timing.rpt
report power > counter_power.rpt
report area > counter_cell.rpt
report gates > counter_gates.rpt

gui_show

```

Figure 3.2: Screenshot of constraint file used for synthesis

3.2 Synthesis Reports

- **Area Report:** Displays cell usage and total area after mapping.
- **Power Report:** Estimates dynamic and leakage power.
- **Timing Report:** Ensures the design meets setup and hold time requirements.

```

me@enu:~/Desktop/source$ ./run.tcl
@genus:root: 2> GTD-INFO: Parsing file top.mtarpt...
report_area
=====
Generated by:          Genius(TM) Synthesis Solution 20.11-s111_1
Generated on:          Oct 24 2025  03:42:21 pm
Module:                daa_multiplier
Operating conditions: slow (balanced_tree)
Wireload mode:         enclosed
Area mode:             timing library
=====

      Instance   Module   Cell Count   Cell Area   Net Area   Total Area   Wireload
-----+-----+-----+-----+-----+-----+-----+-----+
daa_multiplier           114    1008.948     0.000    1008.948 <none> (D)
(D) = wireload is default in technology library
@genus:root: 3> []

```

Figure 3.3: Area report generated by Cadence Genus

```

iiitdmk@cad14:DAA_multiplier_project
File Edit View Search Terminal Help
dab_multiplier          114  1008.948    0.000  1008.948 <none> (D)
  (D) = wireload is default in technology library
@genus:root: 3> report_power
Info   : Joules engine is used. [RPT-16]
      : Joules engine is being used for the command report_power.
Instance: /dab_multiplier
Power Unit: W
PDB Frames: /stim#0/frame#0
-----
Category        Leakage      Internal      Switching      Total      Row%
-----
memory         0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
register       1.60723e-06  4.10439e-05  1.93097e-06  4.45821e-05  58.47%
latch          0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
logic           2.61294e-06  1.99924e-05  6.91867e-06  2.95240e-05  38.72%
bbox            0.00000e+00  0.00000e+00  0.00000e+00  0.00000e+00  0.00%
clock           0.00000e+00  0.00000e+00  2.13840e-06  2.13840e-06  2.80%
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       4.22017e-06  6.10363e-05  1.09880e-05  7.62445e-05  99.99%
Percentage    5.54%       80.05%       14.41%       100.00%  100.00%
-----
@genus:root: 4> report []

```

Figure 3.4: power report generated by Cadence Genus

```

Applications Places Terminal
iiitdmk@cad14:DAA_multiplier_project
File Edit View Search Terminal Help
End delay calculation (fullDC) (MEM=1655.74 CPU=0:00:00.1 REAL=0:00:00.0)
Loading CT timing window with TwFlowType 0..(CPU = 0:00:00.0, REAL = 0:00:00.0, MEM = 1655.7M)
Add other clocks and setup/clockToAAEclockTiming during iter 1
Load initial CT timing window with TwFlowType 0..(CPU = 0:00:00.0, REAL = 0:00:00.0, MEM = 1655.7M)
Starting SI iteration 2 (fullDC) (MEM=1593.88)
SI completed successfully (fullDC) (MEM=1593.88)
Glitch Analysis: View bc -- Total Number of Nets Skipped = 0
Glitch Analysis: Total Number of Objects Analyzed = 148
Total number of fetched objects 148
All nets were worked on for all views 0
AAE INFO-e18: Total number of nets in the design is 148, 0.7 percent of the nets selected for SI analysis
End delay calculation (fullDC) (MEM=1638.6 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=0:19:59 mem=1638.6M)
-----
timeDesign Summary
-----
Setup views included:
bc
+-----+-----+-----+
| Setup mode | all | reg2reg | default |
+-----+-----+-----+
| WNS (ns): | 2.642 | 5.446 | 2.642 | |
| TNS (ns): | 0.000 | 0.000 | 0.000 |
| Violations Paths: | All Paths: | 48 | 16 | 48 |
+-----+-----+-----+
+-----+-----+-----+
| DRVs | Real | Total |
| Nr nets(terms) | Worst Vio | Nr nets(terms) |
+-----+-----+-----+
| max_cap | 1 (1) | -0.002 | 1 (1) |
| max_fanin | 0 (0) | 0.000 | 0 (0) |
| max_fanout | 0 (0) | 0.000 | 0 (0) |
| max_length | 0 (0) | 0 | 0 (0) |
+-----+-----+-----+
Density: 79.48%
Total number of glitch violations: 0
Reported timing to dir timingReports
Total Real time: 2.0 sec
Total Memory usage: 1628.566406 Mbytes
Reset AAE Options
TimeDesign #0 [finish] : cpu/real = 0:00:01.6/0:00:02.5 (0.6), totSession cpu/real = 0:19:58.9/1:04:04.8 (0.3), mem = 1628.6M
innovus[1]

```

Figure 3.5: timing report generated by Cadence Genus

Chapter 4

Physical Design and GDS Generation

The post-synthesis netlist was imported into Cadence Innovus for floorplanning, placement, clock tree synthesis (CTS), and routing. The final layout was verified for DRC and LVS clean.

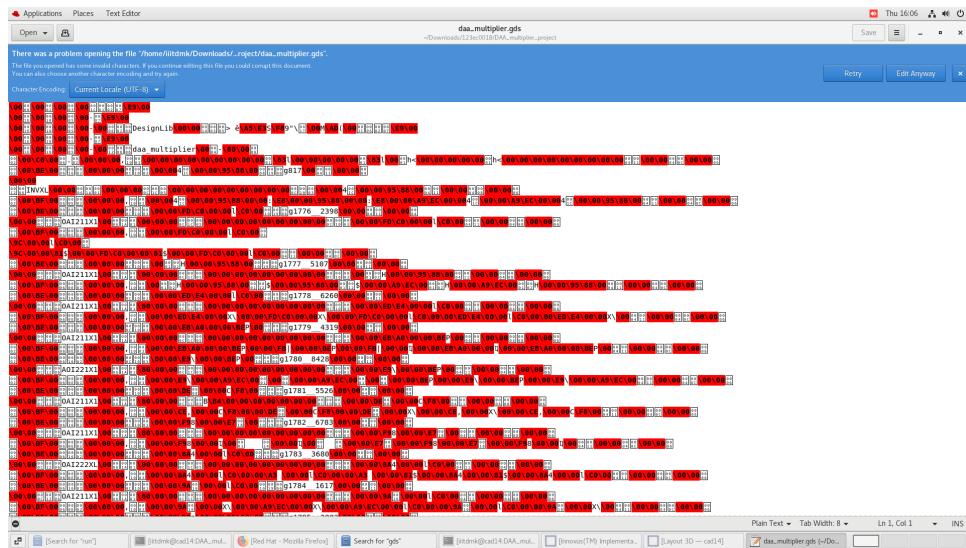


Figure 4.1: GDSII layout view of the DAA Multiplier

4.1 3D View

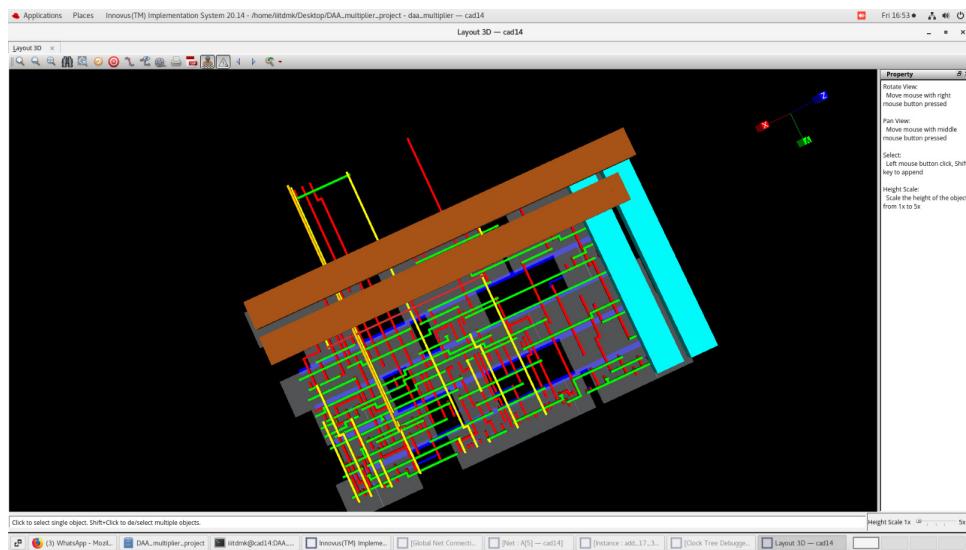


Figure 4.2: 3D view of the routed layout

Chapter 5

Conclusion

The DAA Multiplier was successfully implemented using the semi-custom VLSI design flow. The RTL to GDSII process was completed with verified functionality, optimized power, and acceptable timing performance. This project provided hands-on exposure to professional EDA tools and a clear understanding of ASIC flow stages.

Future Work

Future improvements may include:

- Optimization for smaller technology nodes (45nm, 28nm).
- Integration with power gating and clock gating techniques.

References

J.Bamela Mary , K.Ramamoorthy *Implementation of Low-Complexity Multiplier using distributed arithmetic algorithm.*