Carry save adder rtl to gds

Project Report

Submitted by

Murahari Venkat-123EC0050

Gurram gowtham-123EC0004

Department of Electronics and Communication Engineering

IIITDM Kurnool

Under the guidance of

Dr.Ranga babu

Contents

1 Introduction

This project describes the semi-custom implementation of a 16-bit Carry Save Adder (CSA) using Cadence tools. The flow covers RTL, verification, synthesis, schematic & layout generation, DRC/LVS, and generation of the GDSII file.

2 Design Flow

The main steps followed:

1. RTL development (Verilog)
2. Functional verification (SimVision / ModelSim)
3. Synthesis (Cadence Genus / Synopsys DC) with SDC constraints
4. Netlist-to-schematic review
5. Place & Route (Cadence Innovus)
6. DRC/LVS, STA, and GDSII generation

3 RTL Design (Source Code)

Below is the Verilog RTL for a 16-bit Carry Save Adder. Replace or parameterize width as needed.

```
module carry_save_adder(
    input [15:0] a, b, c,
    output [15:0] sum,
    output [15:0] carry
);
    genvar i;
    generate
        for (i = 0; i < 16; i = i + 1) begin: CSA_STAGE
            full_adder FA (
                .a(a[i]),
                .b(b[i]),
                .cin(c[i]),
                .sum(sum[i]),
                .cout(carry[i])
```

```
            );
        end
    endgenerate
endmodule

module full_adder(
    input a, b, cin,
    output sum, cout
);
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (b & cin) | (cin & a);
endmodule
```

Listing 1: Verilog RTL — 16-bit Carry Save Adder

Explanation: Each bit is implemented as a full adder using boolean expressions. The CSA produces two vectors: ‘sum[5:0]‘ and ‘carry[5:0]‘ (no inter-bit carry propagation).

4 Testbench

A simple testbench used for functional verification.

```
module carry_save_adder_tb;
    reg [15:0] a, b, c;
    wire [15:0] sum, carry;

    carry_save_adder uut (
        .a(a), .b(b), .c(c),
        .sum(sum),
        .carry(carry)
    );

    initial begin
        $display($time, " << Starting simulation >>");
        a = 16'd0; b = 16'd0; c = 16'd0;
        #10 a = 16'd10; b = 16'd5; c = 16'd7;
        #10 a = 16'd255; b = 16'd128; c = 16'd1;
        #10 a = 16'd32767; b = 16'd1; c = 16'd1;
        #10 a = 16'd65535; b = 16'd0; c = 16'd0;
        #10 a = 16'd12345; b = 16'd54321; c = 16'd11111;
        #10 $finish;
    end
```

```
21
22      initial begin
23          $monitor("a=%d, b=%d, c=%d, sum=%d, carry=%d", a, b, c, sum,
                carry);
24      end
25  endmodule
```

Listing 2: Testbench for the CSA

5 TCL Automation Script

Example TCL script for synthesis automation.

```
1  read_hdl carry_save_adder.v
2  elaborate carry_save_adder
3  set_top carry_save_adder
4  link
5  compile_ultra
6  report_timing > timing_report.txt
7  report_area   > area_report.txt
8  write -format ddc -hierarchy -output csa.ddc
```

Listing 3: Example TCL for Synthesis Automation

6 Timing Constraints (SDC)

Example SDC file for synthesis and STA.

```
1  # Create clock 'clk' with 10 ns period (100 MHz)
2  create_clock -name clk -period 10.0 [get_ports clk]
3
4  # Set input delay of 2 ns relative to clock 'clk'
5  set_input_delay -clock clk 2.0 [get_ports *]
6
7  # Set output delay of 2 ns relative to clock 'clk'
8  set_output_delay -clock clk 2.0 [get_ports *]
9
10 # Set clock uncertainty (jitter) of 0.1 ns for 'clk'
11 set_clock_uncertainty 0.1 [get_clocks clk]
```

Listing 4: input SDC Constraints

```
# ##############################################################################

#  Created by Genus(TM) Synthesis Solution 20.11-s111_1 on Sat Oct 25
    10:47:08 IST 2025

# ##############################################################################

set sdc_version 2.0

set_units -capacitance 1000fF
set_units -time 1000ps

# Set the current design
current_design carry_save_adder

set_clock_gating_check -setup 0.0
set_wire_load_mode "enclosed"
```

Listing 5: output SDC Constraints

7 Netlist

```

// Generated by Cadence Genus(TM) Synthesis Solution 20.11-s111_1
// Generated on: Oct 25 2025 10:47:08 IST (Oct 25 2025 05:17:08 UTC)

// Verification Directory fv/carry_save_adder

module carry_save_adder(a, b, c, sum, carry);
  input [15:0] a, b, c;
  output [15:0] sum, carry;
  wire [15:0] a, b, c;
  wire [15:0] sum, carry;
  ADDFXL g1261__2398(.A (a[15]), .B (b[15]), .CI (c[15]), .CO
       (carry[15]), .S (sum[15]));
  ADDFXL g1265__5107(.A (a[0]), .B (b[0]), .CI (c[0]), .CO (carry[0]),
       .S (sum[0]));
  ADDFXL g1276__6260(.A (a[1]), .B (b[1]), .CI (c[1]), .CO (carry[1]),
       .S (sum[1]));
  ADDFXL g1275__4319(.A (a[2]), .B (b[2]), .CI (c[2]), .CO (carry[2]),
```

```
      .S (sum[2]));
  ADDFXL g1273__8428(.A (a[3]), .B (b[3]), .CI (c[3]), .CO (carry[3]),
      .S (sum[3]));
  ADDFXL g1272__5526(.A (a[4]), .B (b[4]), .CI (c[4]), .CO (carry[4]),
      .S (sum[4]));
  ADDFXL g1269__6783(.A (a[5]), .B (b[5]), .CI (c[5]), .CO (carry[5]),
      .S (sum[5]));
  ADDFXL g1268__3680(.A (a[6]), .B (b[6]), .CI (c[6]), .CO (carry[6]),
      .S (sum[6]));
  ADDFXL g1266__1617(.A (a[7]), .B (b[7]), .CI (c[7]), .CO (carry[7]),
      .S (sum[7]));
  ADDFXL g1264__2802(.A (a[8]), .B (b[8]), .CI (c[8]), .CO (carry[8]),
      .S (sum[8]));
  ADDFXL g1274__1705(.A (a[9]), .B (b[9]), .CI (c[9]), .CO (carry[9]),
      .S (sum[9]));
  ADDFXL g1271__5122(.A (a[10]), .B (b[10]), .CI (c[10]), .CO
      (carry[10]), .S (sum[10]));
  ADDFXL g1267__8246(.A (a[11]), .B (b[11]), .CI (c[11]), .CO
      (carry[11]), .S (sum[11]));
  ADDFXL g1263__7098(.A (a[12]), .B (b[12]), .CI (c[12]), .CO
      (carry[12]), .S (sum[12]));
  ADDFXL g1270__6131(.A (a[13]), .B (b[13]), .CI (c[13]), .CO
      (carry[13]), .S (sum[13]));
  ADDFXL g1262__1881(.A (a[14]), .B (b[14]), .CI (c[14]), .CO
      (carry[14]), .S (sum[14]));
endmodule
```

Listing 6: Netlist generated by genus

8 Schematic (Cell-Level)

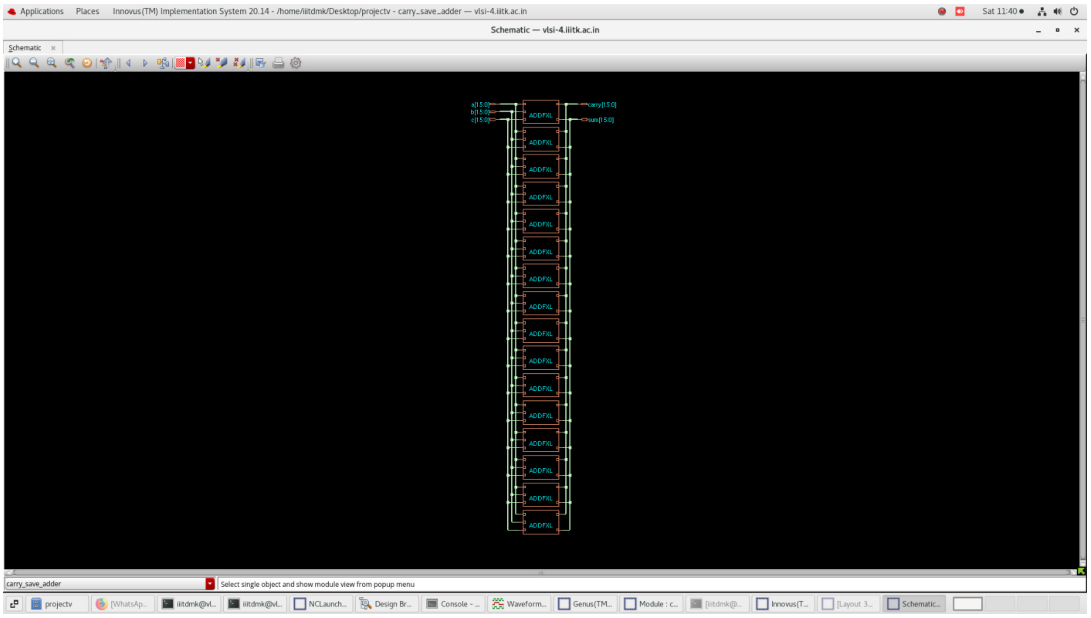

Figure 1: Schematic view (cell-level) of the 16-bit Carry Save Adder after synthesis.

9 Functional Simulation Waveforms

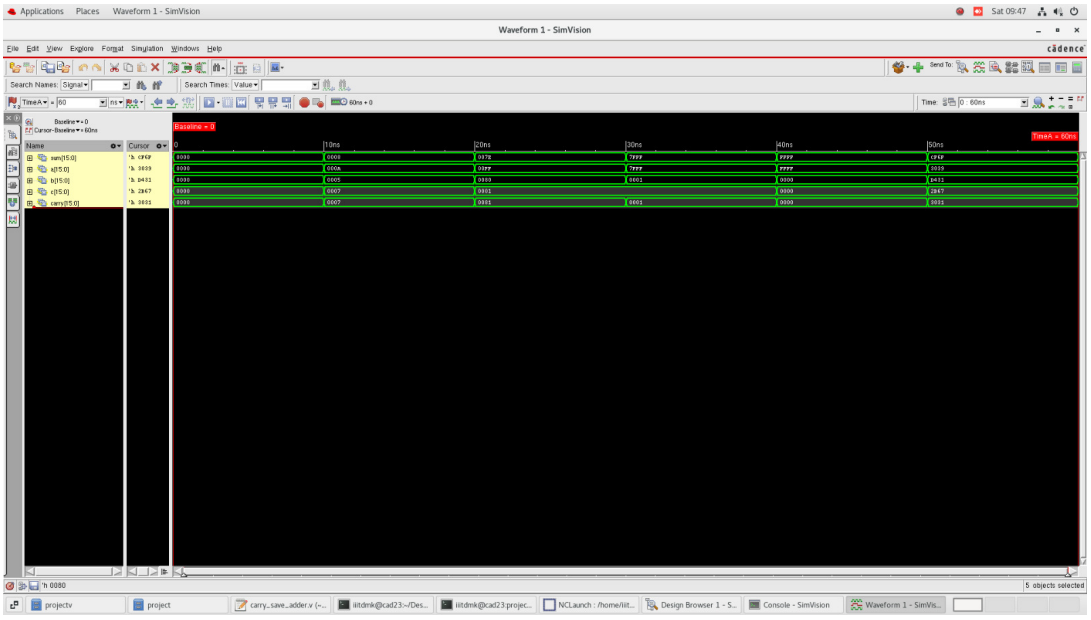

Figure 2: Waveform from SimVision demonstrating correct CSA operation for test vectors.

10 Constraint Setup and Corner Libraries

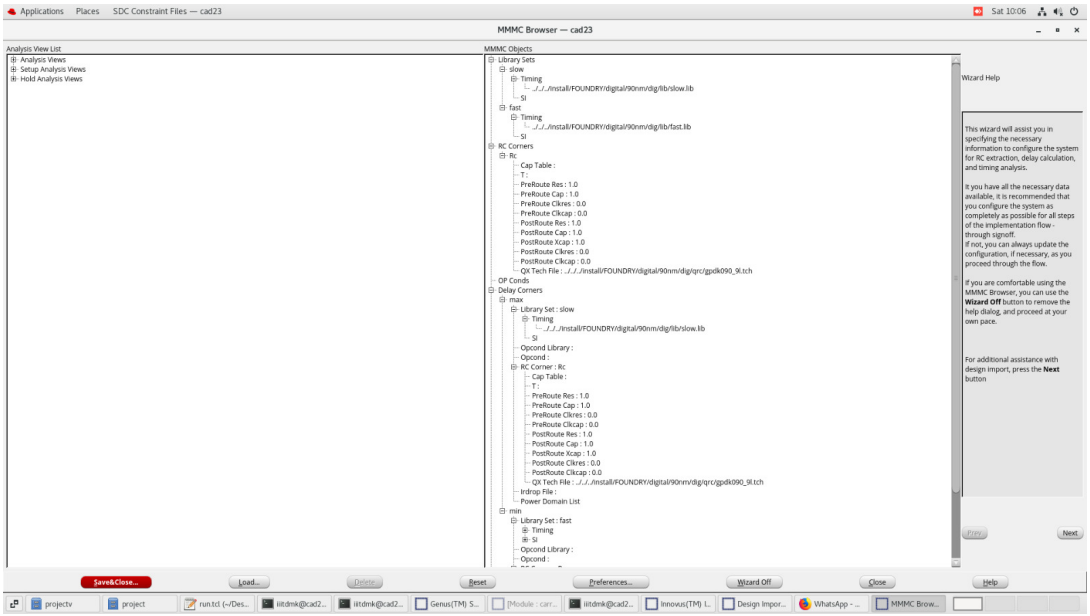

Figure 3: MMMC / library setup used for STA and extraction.

11 Layout (Place & Route)

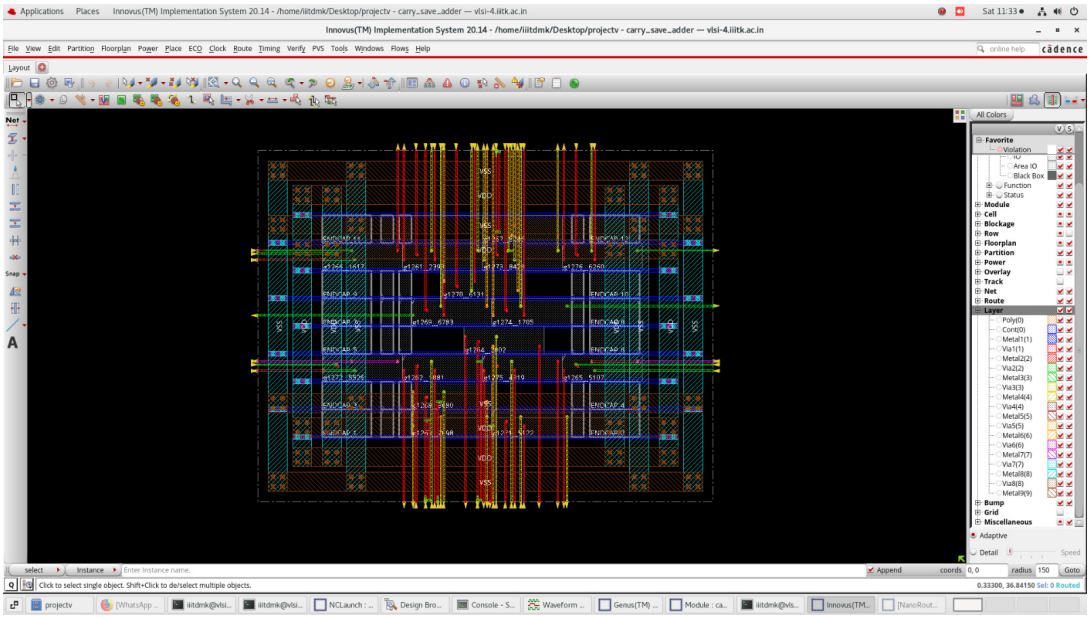

Figure 4: Layout view after placement and routing (Innovus).

12 3D Layout / Metal Stack View

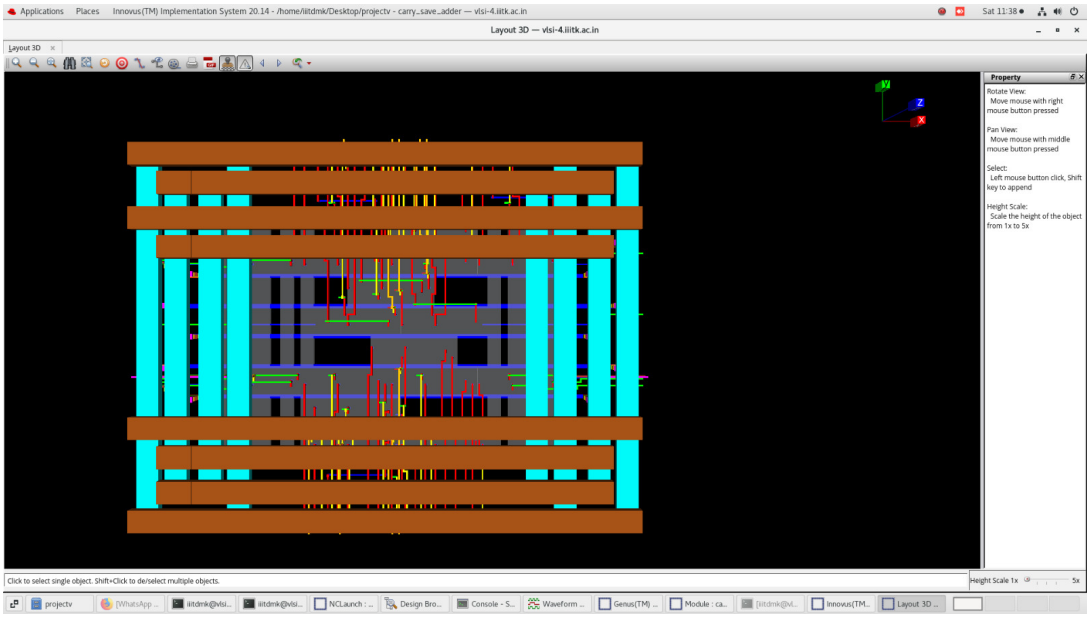

Figure 5: 3D layout view .

13 GDSII File

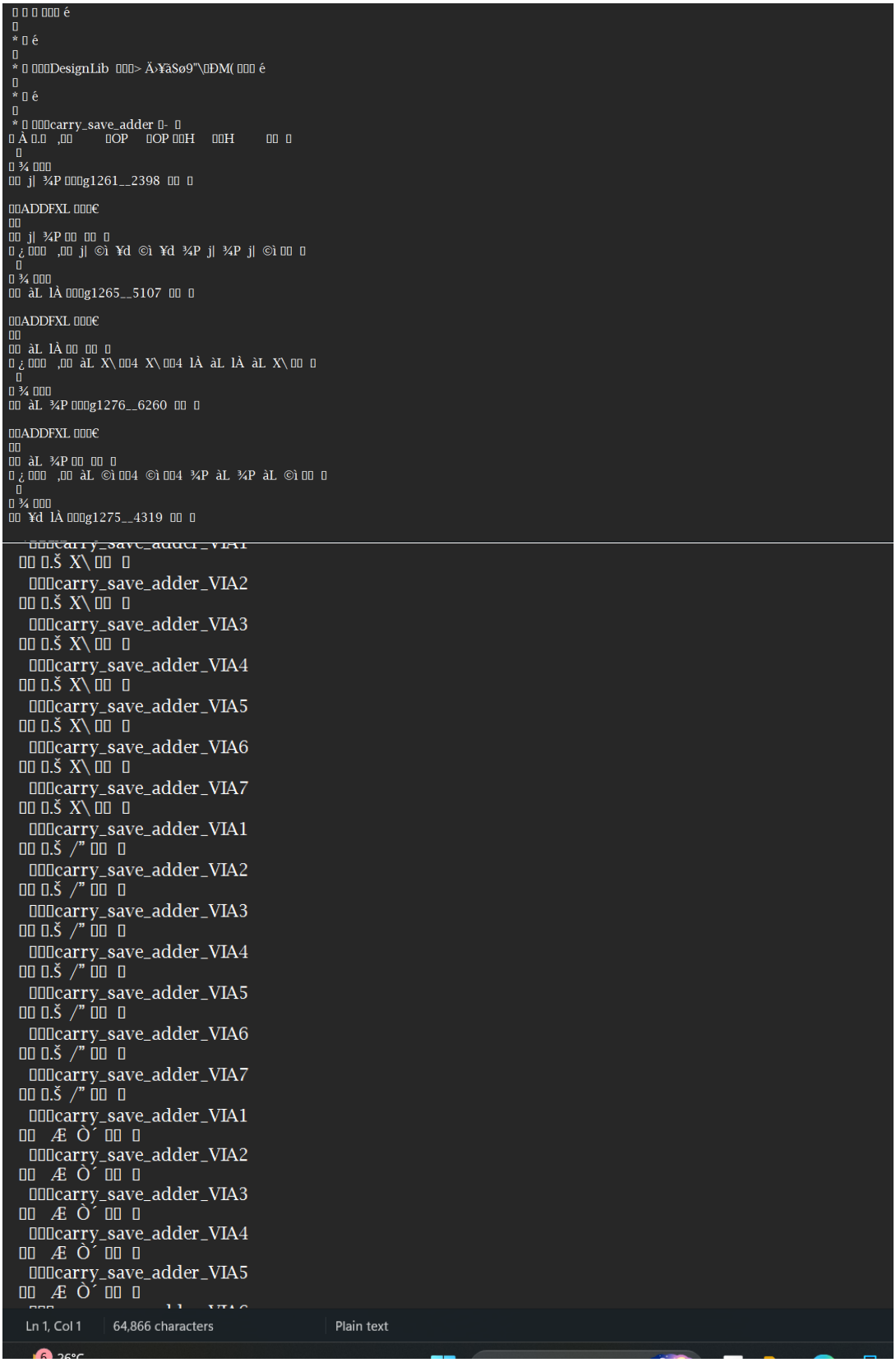

Figure 6: Final GDSII layout view ready for tape-out .

14 Result

The semi-custom CSA was implemented end-to-end (RTL → GDS). Future work includes pipelining, width parameterization, and low-power optimization.