Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

Division of Electronics and Communication Engineering
2024-2025 (ODD SEM)

REPORT

for

Digital IC Design

Title of the Report:

ASIC Implementation and Power Analysis of a Verilog-Based Carry Select Adder

A report submitted by

Name of the Student	**ALDRIN G, BASIL RAYMUND A, LIGIN JITTO E A**
Register Number	***URK22EC1019, URK22EC1033, URK22EC1035***
Subject Name	***Digital IC Design***
Subject Code	***18EC2019***
Date of Report submission	***21/04/2025***

Project Rubrics for Evaluation

First Review: Project title selection - PPT should have four slides (Title page, Introduction, Circuit/Block Diagram, and Description of Project).

Second Review: PPT should have three slides (Description of Concept, implementation, outputs, results and discussion)

Rubrics for project (III IA - 40 Marks):
Content - 4 marks (based on Project)
Clarity - 3 marks (based on viva during presentation) Feasibility - 3 marks (based on project)
Presentation - 10 marks Project
Report - 10 marks
On-time submission - 5 marks (before the due date) Online submission-GCR - 5 marks

Total marks:_____/ 40 Marks

Signature of Faculty with date:

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
MoE, UGC & AICTE Approved
NAAC A++ Accredited

TABLE OF CONTENTS

Abstract

This project report outlines the complete ASIC flow for implementing a Carry Select Adder (CSA) designed in Verilog. The process begins with simulation using a dedicated testbench and waveform analysis with GTKWave, followed by hardware validation on a Basys3 FPGA board. A detailed power analysis with Xilinx Vivado compares the CSA with a traditional ripple adder, highlighting the speed-power trade-off. The design is then synthesized using Yosys and advanced through an RTL-to-GDSII flow via the OpenLane framework and SkyWater 130nm PDK, encompassing critical steps such as floor planning, placement, and routing. This report offers a comprehensive view of both the functional performance and physical design challenges, demonstrating the complete journey from simulation to chip-ready GDSII for a high-speed arithmetic unit.

CHAPTER 1

INTRODUCTION

The evolution of digital systems has necessitated the continuous improvement of arithmetic circuits, where the trade-offs between speed, power consumption, and area become critical design parameters. This project focuses on the design and implementation of a Carry Select Adder (CSA) using Verilog, which has been widely recognized for its superior speed performance compared to traditional ripple adders.

The project journey begins with the development of a Verilog model of the CSA, accompanied by a comprehensive testbench to verify its functionality. Simulation results were captured in VCD format and analyzed using GTKWave, ensuring that the adder operated correctly under a variety of test conditions. The design was further validated through hardware implementation on a Basys3 FPGA board, bridging the gap between simulation and practical application.

A significant portion of this work involved a detailed power consumption analysis using Xilinx Vivado, comparing the performance of the CSA with that of a conventional ripple adder. This comparative study provided insights into the balance between enhanced speed and increased power usage, which is crucial for optimizing digital circuits for specific applications.

To complete the design flow, the project also explored modern synthesis and fabrication methodologies. The CSA design was synthesized using Yosys, and an RTL-to-GDSII implementation was carried out using the OpenLane framework in conjunction with the SkyWater 130nm PDK. This process included critical steps such as floor planning, placement, and routing, culminating in the generation of GDSII files ready for chip fabrication.

CHAPTER 2

ASIC DESIGN FLOW BLOCK DIAGRAM

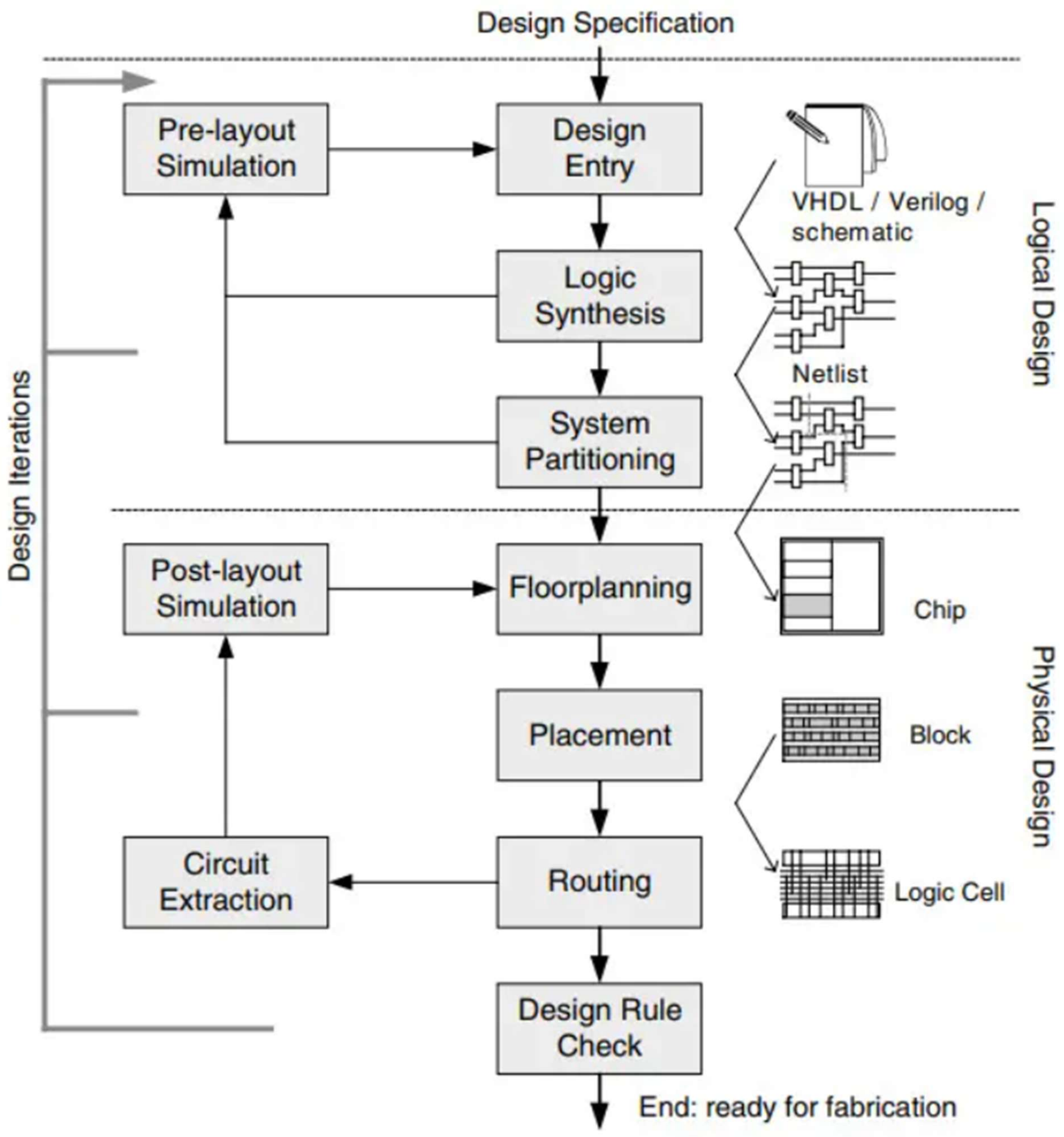


Fig.1 Block Diagram of the ASIC Design Flow

CHAPTER 3
DESCRIPTION OF THE PROCESS FLOW

Design Description:

The project centers on implementing a Carry Select Adder (CSA) using Verilog. The design is broken down into several key stages:

1. **Verilog Implementation:**
   The CSA is implemented in Verilog, structured to take advantage of its parallel carry computation. The design splits the addition process into smaller blocks that compute results in parallel for both possible carry-in values, and then selects the result once the actual carry is known.

2. **Testbench and Simulation:**
   A dedicated testbench is created to verify the functionality of the CSA. Simulation is conducted to generate a VCD (Value Change Dump) file using iVerilog. The waveform is then analyzed using GTKWave.

3. **FPGA Implementation:**
   After verifying the design through simulation, the CSA is synthesized and implemented on a Basys3 FPGA board. This step validates the design in a real hardware environment and demonstrates its practical applicability.

4. **Power Analysis:**
   The project includes a power consumption comparison between the CSA and a traditional ripple adder using Xilinx Vivado. This analysis highlights that while the CSA offers improved speed due to parallel computation, it also incurs a higher power cost.

5. **Physical Design Flow:**

   For a complete design cycle, the project extends into physical design. The CSA design is synthesized using Yosys, and then the RTL-to-GDSII flow is executed via the OpenLane framework with the SkyWater 130nm PDK. This stage encompasses:

   - **Floor Planning:** Organizing the layout for efficient space and performance.
   - **Placement and Routing:** Assigning cells to physical locations and connecting them appropriately.
   - **GDSII Generation:** Finalizing the design in GDS format, making it ready for fabrication.

Working Principle Of CSA:

The Carry Select Adder (CSA) design operates by breaking down the addition of binary numbers into smaller blocks that work concurrently, enabling faster computation compared to traditional ripple adders. Here's a breakdown of its functionality:

1. **Parallel Processing:**

   The CSA divides the addition process into segments, where each segment calculates two potential sums simultaneously—one assuming a carry-in of 0 and the other assuming a carry-in of 1.

2. **Multiplexer-Based Selection:**

   Once the actual carry-in is known from the preceding block, a multiplexer selects the correct result from the two precomputed values. This significantly reduces the overall delay in obtaining the final sum.

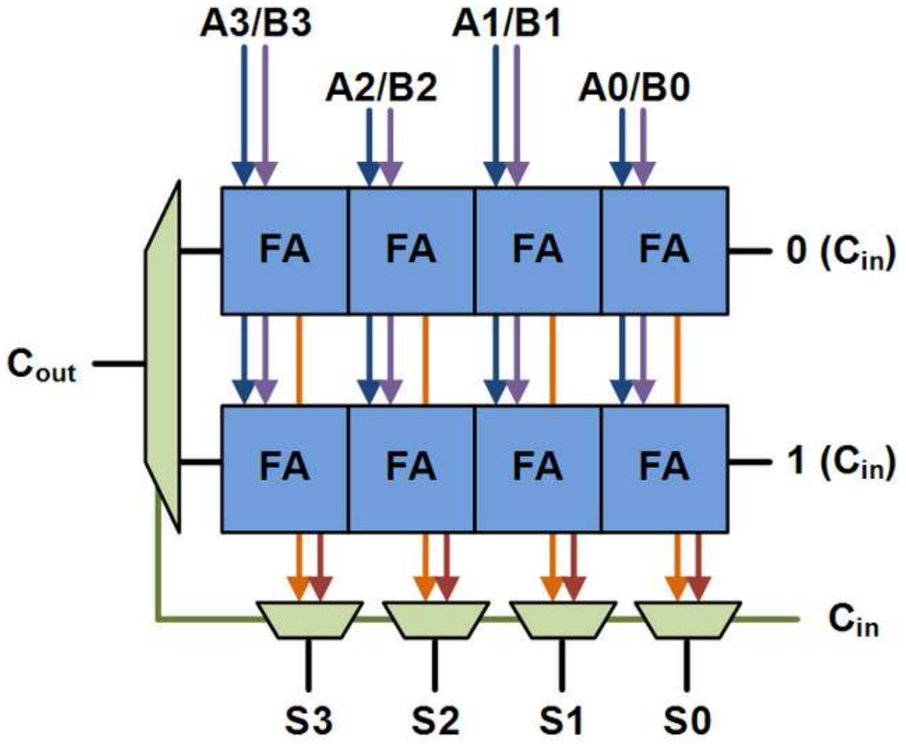
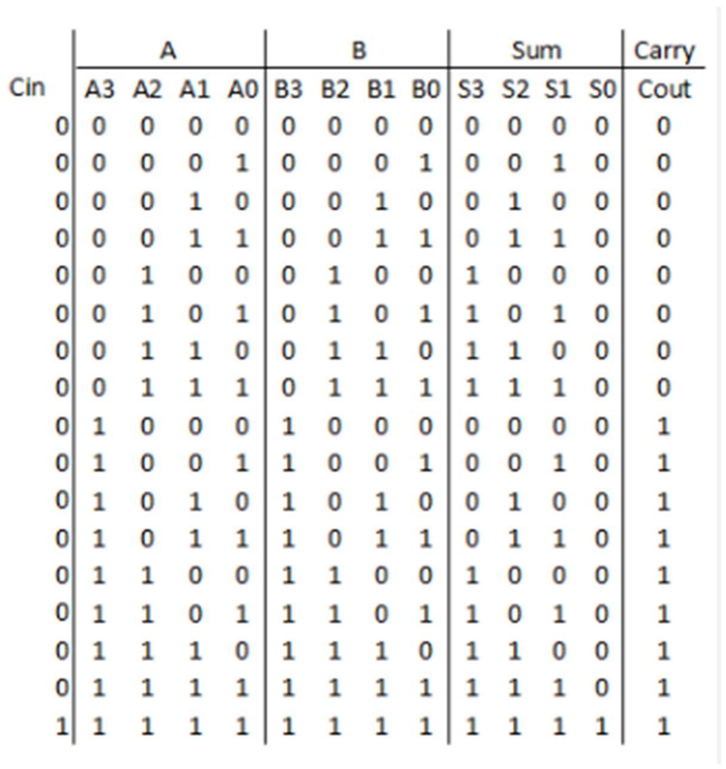


Fig.2 Block Diagram of a Carry Select Adder



	A				B				Sum				Carry
Cin	A3	A2	A1	A0	B3	B2	B1	B0	S3	S2	S1	S0	Cout
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	0	0	1	0	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	1	1	0	1	1	0	0
0	0	1	0	0	0	1	0	0	1	0	0	0	0
0	0	1	0	1	0	1	0	1	1	0	1	0	0
0	0	1	1	0	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	1	1	1	1	1	0	0
0	1	0	0	0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	1
0	1	0	1	1	1	0	1	1	0	1	1	0	1
0	1	1	0	0	1	1	0	0	1	0	0	0	1
0	1	1	0	1	1	1	0	1	1	0	1	0	1
0	1	1	1	0	1	1	1	0	1	1	0	0	1
0	1	1	1	1	1	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig.3 Adder Truth Table

CHAPTER 4

TOOLS AND SOFTWARES USED

1. iVerilog

A popular open-source Verilog simulation and synthesis tool used to write and simulate digital designs. It helps verify HDL code before hardware implementation.

2. GTKWave

A waveform viewer used alongside iVerilog for debugging digital circuits by visualizing signal transitions and timing diagrams.

3. Xilinx Vivado (for Power Analysis)

An advanced FPGA design suite from Xilinx, which includes power analysis tools to estimate static and dynamic power consumption of FPGA designs during implementation.

4. Skywater 130nm PDK (130PDK) with Yosys

A 130nm open-source Process Design Kit by SkyWater, used for ASIC design. Yosys is an open-source synthesis tool compatible with this PDK for RTL to gate-level synthesis.

5. OpenLane

A fully open-source digital ASIC design flow that integrates tools like Yosys, OpenLane, and others for physical design using the Skywater 130nm PDK.

6. KLayout

A powerful open-source layout viewer and editor used in ASIC and MEMS design, ideal for inspecting GDSII layouts and verifying DRC/LVS rules.

CHAPTER 5

DESIGN METHODOLOGY

I. MAKING OF VERILOG DESIGN AND TESTBENCH:

a. Carry Select Adder Verilog Code:

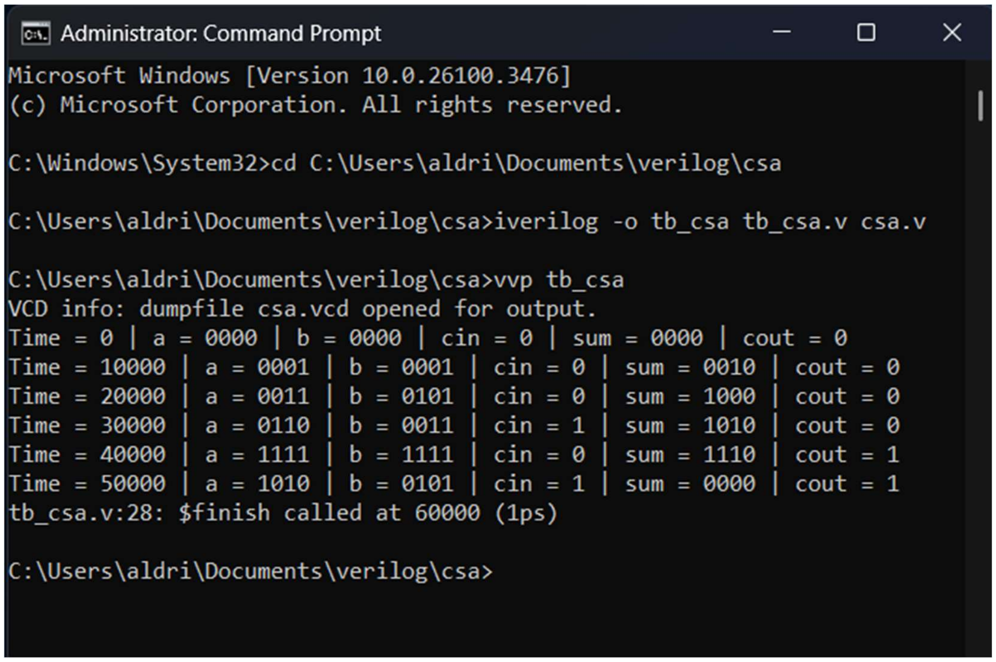
```
module csa(
    input [3:0] a, b,
    input cin,
    output [3:0] sum,
    output cout
);

    wire [3:0] sum0, sum1;
    wire cout0, cout1, sel;

    // Ripple Carry Adder with carry-in = 0
    ripple_carry_adder_4bit rca0 (
        .a(a), .b(b), .cin(1'b0), .sum(sum0), .cout(cout0)
    );

    // Ripple Carry Adder with carry-in = 1
    ripple_carry_adder_4bit rca1 (
        .a(a), .b(b), .cin(1'b1), .sum(sum1), .cout(cout1)
    );

    // Carry Select Logic
    assign sel = cin;
    assign sum = sel ? sum1 : sum0;
    assign cout = sel ? cout1 : cout0;

endmodule

// 4-bit Ripple Carry Adder Module
module ripple_carry_adder_4bit(
    input [3:0] a, b,
    input cin,
    output [3:0] sum,
    output cout
);
    wire c1, c2, c3;

    full_adder fa0 (.a(a[0]), .b(b[0]), .cin(cin),  .sum(sum[0]), .cout(c1));
    full_adder fa1 (.a(a[1]), .b(b[1]), .cin(c1),   .sum(sum[1]), .cout(c2));
    full_adder fa2 (.a(a[2]), .b(b[2]), .cin(c2),   .sum(sum[2]), .cout(c3));
    full_adder fa3 (.a(a[3]), .b(b[3]), .cin(c3),   .sum(sum[3]), .cout(cout));

endmodule
```

```
// Full Adder Module
module full_adder(
    input a, b, cin,
    output sum, cout
);
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (b & cin) | (a & cin);

endmodule
```

b. Carry Select Adder Testbench Code:

```
`timescale 1ns / 1ps

module tb_csa;
    reg [3:0] a, b;
    reg cin;
    wire [3:0] sum;
    wire cout;

    // Instantiate the Carry Select Adder
    csa uut (
        .a(a), .b(b), .cin(cin), .sum(sum), .cout(cout)
    );

    initial begin
        // Open VCD file for waveform dumping
        $dumpfile("csa.vcd");
        $dumpvars(0, tb_csa);

        // Test cases
        a = 4'b0000; b = 4'b0000; cin = 0; #10;
        a = 4'b0001; b = 4'b0001; cin = 0; #10;
        a = 4'b0011; b = 4'b0101; cin = 0; #10;
        a = 4'b0110; b = 4'b0011; cin = 1; #10;
        a = 4'b1111; b = 4'b1111; cin = 0; #10;
        a = 4'b1010; b = 4'b0101; cin = 1; #10;

        // End simulation
        $finish;
    end

    initial begin
        $monitor("Time = %0t | a = %b | b = %b | cin = %b | sum = %b | cout = %b",
                 $time, a, b, cin, sum, cout);
    end

endmodule
```

II. COMPILATION OF THE DESIGN USING ‘ICARUS VERILOG’:



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>cd C:\Users\aldri\Documents\verilog\csa

C:\Users\aldri\Documents\verilog\csa>iverilog -o tb_csa tb_csa.v csa.v

C:\Users\aldri\Documents\verilog\csa>vvp tb_csa
VCD info: dumpfile csa.vcd opened for output.
Time = 0 | a = 0000 | b = 0000 | cin = 0 | sum = 0000 | cout = 0
Time = 10000 | a = 0001 | b = 0001 | cin = 0 | sum = 0010 | cout = 0
Time = 20000 | a = 0011 | b = 0101 | cin = 0 | sum = 1000 | cout = 0
Time = 30000 | a = 0110 | b = 0011 | cin = 1 | sum = 1010 | cout = 0
Time = 40000 | a = 1111 | b = 1111 | cin = 0 | sum = 1110 | cout = 1
Time = 50000 | a = 1010 | b = 0101 | cin = 1 | sum = 0000 | cout = 1
tb_csa.v:28: $finish called at 60000 (1ps)

C:\Users\aldri\Documents\verilog\csa>
```

Fig.4 Shows the compilation and execution of the Verilog Design and the Testbench

III. WAVEFORM ANALYSIS USING ‘GTKWAVE”:

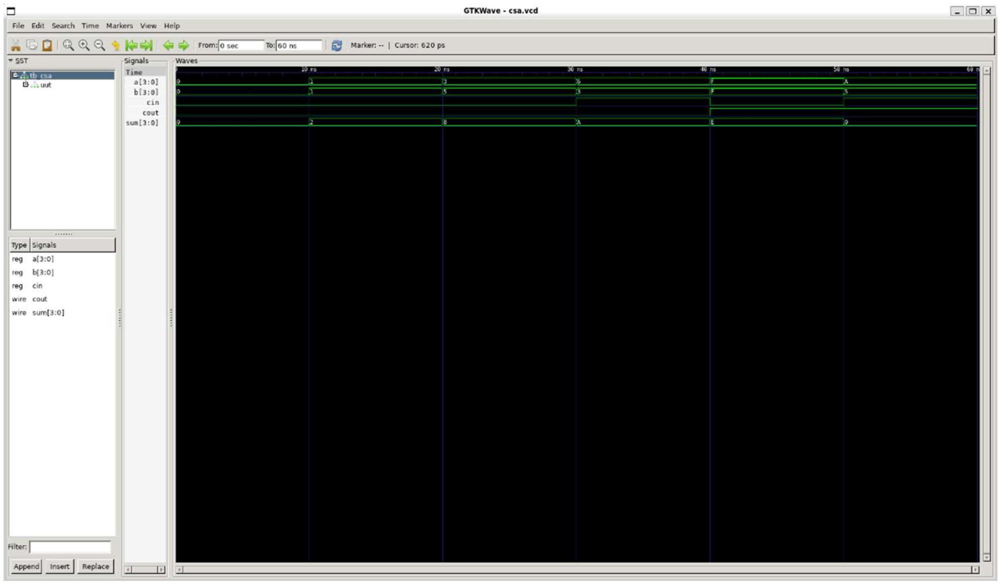
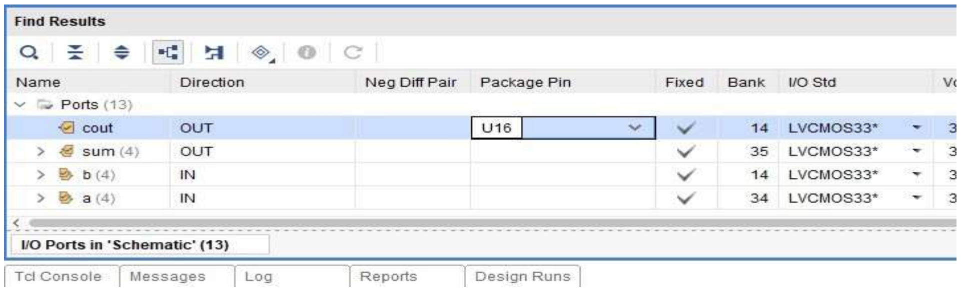


Fig.5 Analysis of the Waveform file generated using GTKwave

IV. PRE-LAYOUT SIMULATION AND VERIFICATION IN BASYS3 FPGA

Procedure For Implementation of the Verilog design in Basys3 FPGA Board:

1. Create a New RTL Project Named “BCD_Adder” and create a verilog file named “BCD_Adder in add sources
2. Select the part name “XC7A35TCPG236-1” to select the BASYS 3 FPGA board
3. Write your verilog code and save it
4. Click Run SimulationRun Behavioural Simulation, Check the correctness of the code by forcing constant as input and verifying the output waveform
5. Click “Open Elabourated Design” under RTL Analysis
6. Click the I/O ports option in the schematic generated, a “Find Results” windows open:



Find Results

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vc
Ports (13)							
cout	OUT		U16	✓	14	LVCMOS33*	3
sum (4)	OUT			✓	35	LVCMOS33*	3
b (4)	IN			✓	14	LVCMOS33*	3
a (4)	IN			✓	34	LVCMOS33*	3

I/O Ports in 'Schematic' (13)

Fig.6 Assigning Port definitions and creation of the constraints file

- Select I/O std as “LVCMOS33” in all ports
- Select the package pin for each port as per the pin out/pin name given in the board as required for the design
- Click “Ctrl+S” and give a file name in the prompt opened to save the constraints

7. Click “Run Synthesis” under Synthesis
   - Change the ‘Number of jobs’ as ‘4’ in the prompt which opens
   - Click ‘OK’

8. After Synthesis is completed, a prompt opens:
    - Select ‘Run Implementation’ option
    - Click ‘OK’

9. After Implementation is completed, a prompt opens:
    - Select ‘Open Implemented Design’
    - Click ‘OK’
    - A prompt opens to close the elaborated design, Click ‘Yes’

10. Click “Generate Bitstream” under Program and Debug
    - Click Ok in the prompt which opens

11. After Bitstream is generated, a prompt opens:
    - Click ‘OK’

12. Now connect the BASYS 3 FPGA board to PC

13. Click “Open Hardware Manager Open Target” under Program and Debug
    - Click Auto-connect
    - After successful connection, click “Program device”

14. Now vary the input for different cases, and note the output

15. Verify with the expected output

Block Diagram:

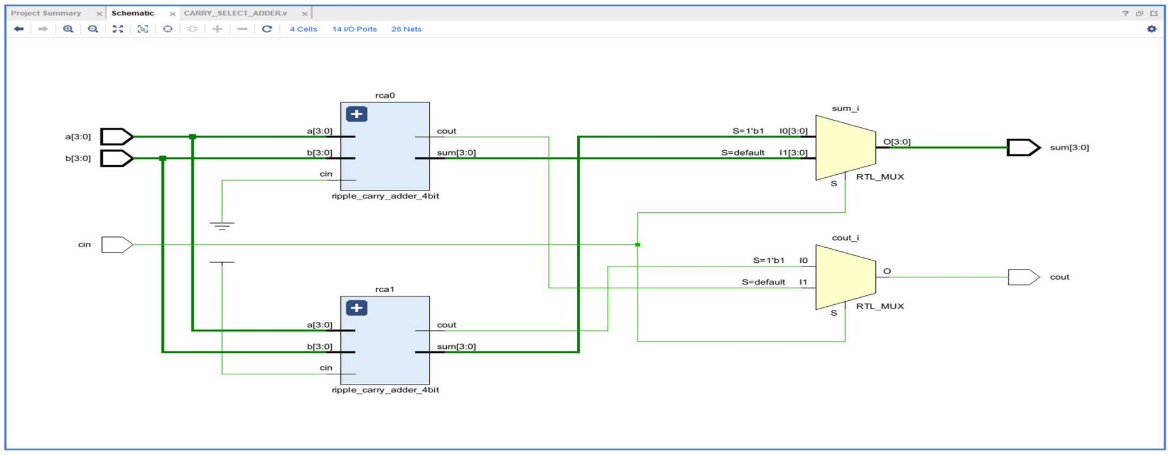


Fig.7 Shows the Schematic of the Carry select Adder

Port Mapping:

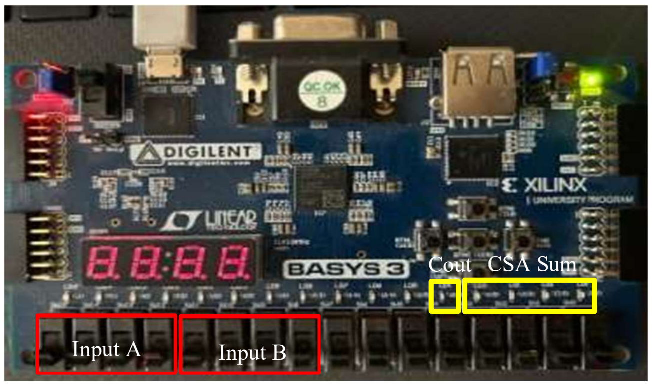
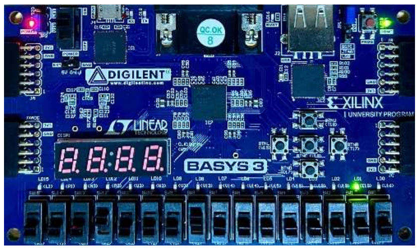
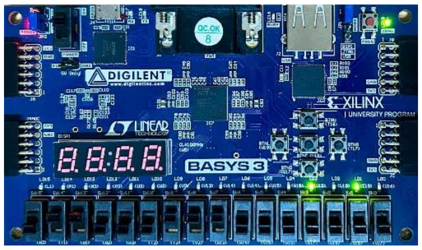
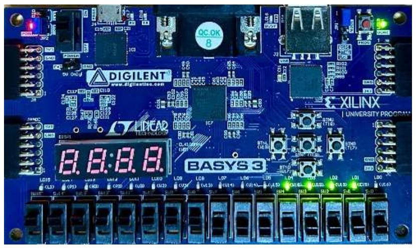
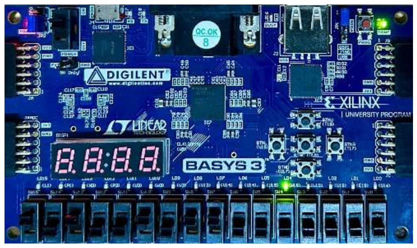


Fig.8 Shows Port Mapping in the BASYS3 FPGA

Results:

Case 1: **1 + 1:** • A = 0000 • B = 0000 Cin = 0  Sum: 0010 Carry: 0	**Case 2:** **6 + 3:** • A = 0110 • B = 0011 Cin = 1  Sum: 1010 (A) Carry: 0
Case 3: **15 + 15:** • A = 1111 (F) • B = 1111 (F)  Sum: 1110 Carry: 1	**Case 4:** **10 + 5:** • A = 1010 (A) • B = 0101  Sum: 0000 Carry: 1

V. PHYSICAL LAYOUT USING SKYWATER 130PDK:

SkyWater 130nm PDK –

The SkyWater 130nm Process Design Kit (Sky130 PDK) is an open-source CMOS technology platform developed by SkyWater Technology. It provides all the necessary files, models, and libraries required for ASIC design, including standard cells, DRC/LVS rules, and technology specifications.

In this project, the Sky130 PDK was used in combination with the OpenLane toolchain to implement the RTL-to-GDSII flow for the Carry Select Adder. The use of Sky130 enabled a complete, fabrication-level physical design using an industry-grade, open-source technology.

1. Synthesis Using 'YOSYS':

Yosys was used as the first step in the ASIC flow to synthesize the Carry Select Adder (CSA) design. It translated the high-level Verilog description into a netlist using standard cells from the SkyWater 130nm PDK.

A **flow diagram** was created to illustrate the complete synthesis and implementation process.

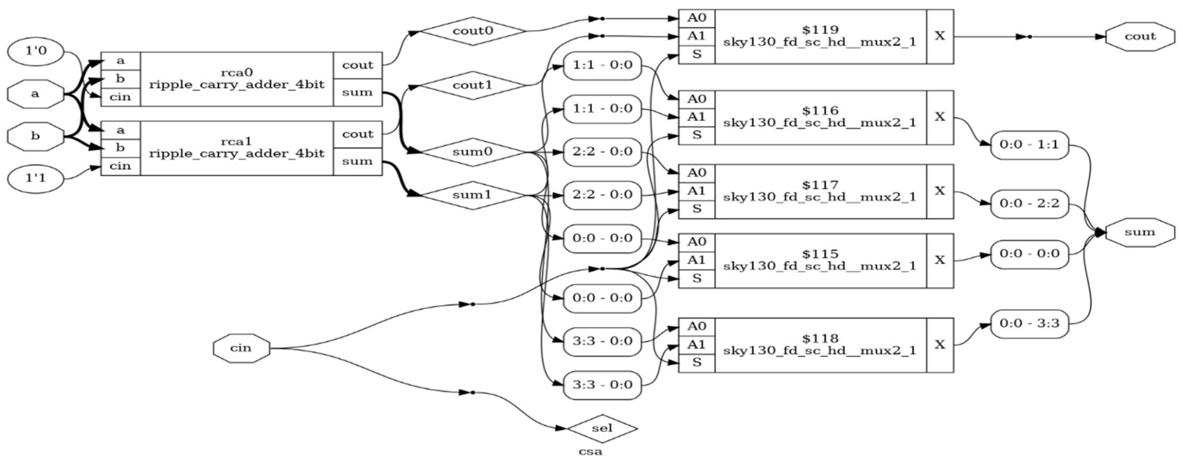


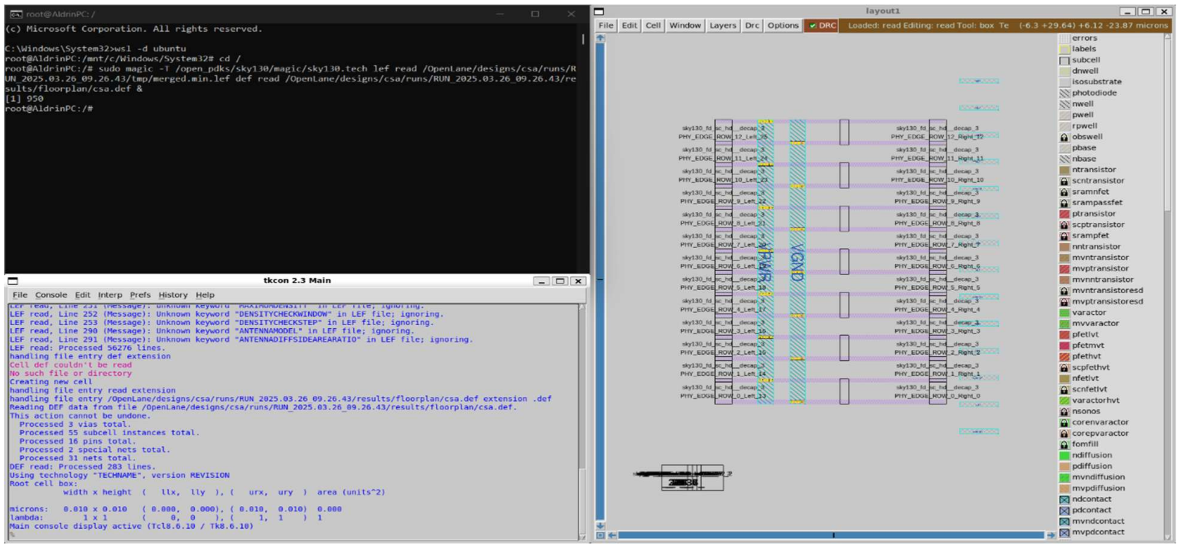
Fig.9 Flow diagram generated by synthesis using YOSYS tool

2. Physical Layout Using ‘OPENLANE’:

OpenLane is an open-source digital ASIC flow tool used to convert synthesized netlists into a physical layout. In this project, OpenLane was used to generate the physical layout of the Carry Select Adder (CSA) using the SkyWater 130nm PDK. The flow included key steps like floor planning, placement, clock tree synthesis, routing, and final GDSII generation. This process produced a layout ready for fabrication, completing the RTL-to-GDSII implementation.

A. FLOOR PLANNING:

Floor planning is the initial step in the physical design process where the basic layout of the chip is defined. It involves allocating space for the core area, placing input/output ports, and defining power and ground regions.


root@AldrinPC: /
tkcon 2.3 Main
File Console Edit Interp Prefs History Help
layout1
File Edit Cell Window Layers Drc Options
DRC
Loaded: read Editing: read Tool: box

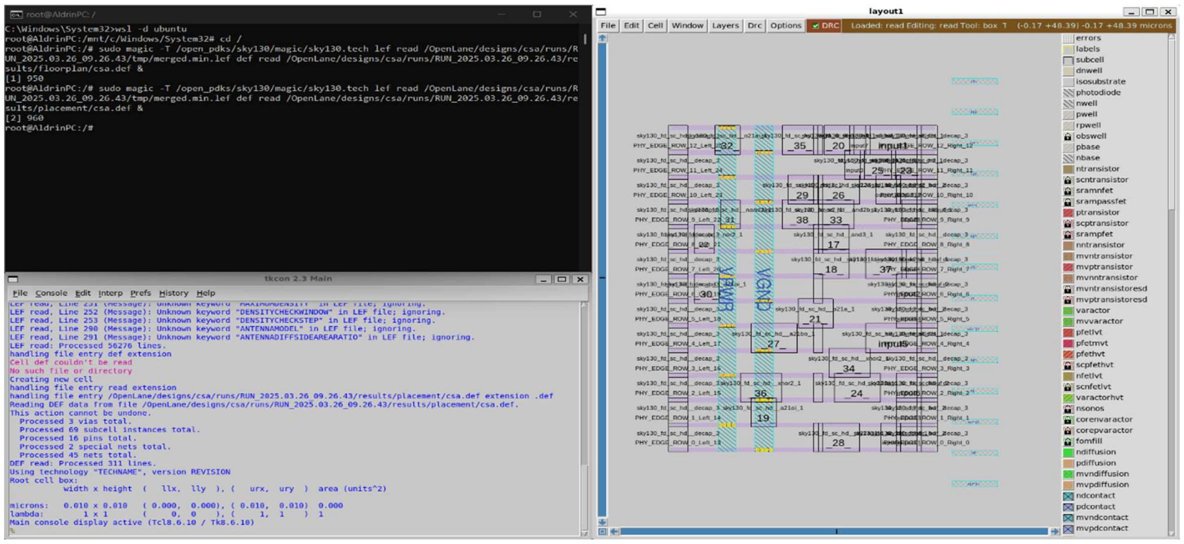
```
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>wsl -d ubuntu
root@AldrinPC:/mnt/c/Windows/System32# cd /
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/floorplan/csa.def &
[1] 950
root@AldrinPC:/#
```

```
LEF read, Line 252 (Message): Unknown keyword "DENSITYCHECKWINDOW" in LEF file; ignoring.
LEF read, Line 253 (Message): Unknown keyword "DENSITYCHECKSTEP" in LEF file; ignoring.
LEF read, Line 290 (Message): Unknown keyword "ANTENNAMODEL" in LEF file; ignoring.
LEF read, Line 291 (Message): Unknown keyword "ANTENNADIFFSIDEAREARATIO" in LEF file; ignoring.
LEF read: Processed 56276 lines.
handling file entry def extension
Cell def couldn't be read
No such file or directory
Creating new cell
handling file entry read extension
handling file entry /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/floorplan/csa.def extension .def
Reading DEF data from file /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/floorplan/csa.def.
This action cannot be undone.
  Processed 3 vias total.
  Processed 55 subcell instances total.
  Processed 16 pins total.
  Processed 2 special nets total.
  Processed 31 nets total.
DEF read: Processed 283 lines.
Using technology "TECHNAME", version REVISION
Root cell box:
           width x height  (   llx,  lly  ), (   urx,  ury  )  area (units^2)

microns:   0.010 x 0.010   ( 0.000,  0.000), ( 0.010,  0.010)  0.000
lambda:        1 x 1       (     0,  0    ), (     1,  1    )  1
Main console display active (Tcl8.6.10 / Tk8.6.10)
```

Fig.10 Floor Planning process

B. PLACEMENT:

Placement is the stage in physical design where the standard cells from the synthesized netlist are assigned fixed locations within the chip layout. The tool positions the cells to minimize wire length and timing delays, ensuring optimal connectivity and efficient use of chip area, while preparing the design for routing.



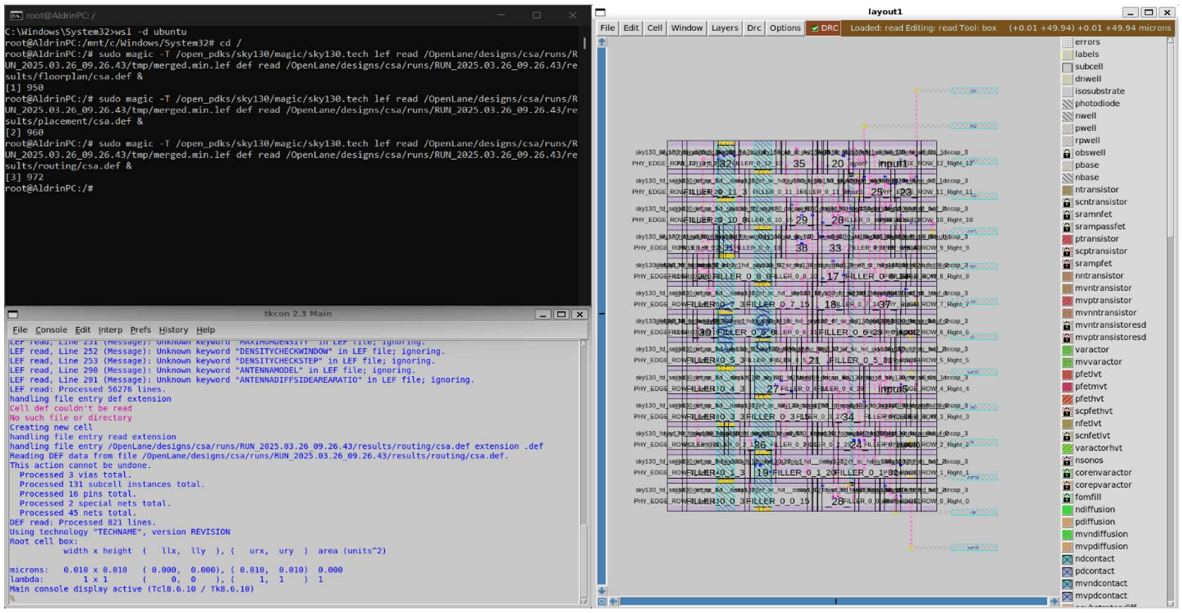
```
C:\Windows\System32>wsl -d ubuntu
root@AldrinPC:/mnt/c/Windows/System32# cd /
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/floorplan/csa.def &
[1] 950
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/placement/csa.def &
[2] 960
root@AldrinPC:/#
```

```
LEF read, Line 251 (Message): Unknown keyword "MAXIMUMDENSITY" in LEF file; ignoring.
LEF read, Line 252 (Message): Unknown keyword "DENSITYCHECKWINDOW" in LEF file; ignoring.
LEF read, Line 253 (Message): Unknown keyword "DENSITYCHECKSTEP" in LEF file; ignoring.
LEF read, Line 290 (Message): Unknown keyword "ANTENNAMODEL" in LEF file; ignoring.
LEF read, Line 291 (Message): Unknown keyword "ANTENNADIFFSIDEAREARATIO" in LEF file; ignoring.
LEF read: Processed 56276 lines.
handling file entry def extension
Cell def couldn't be read
No such file or directory
Creating new cell
handling file entry read extension
handling file entry /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/placement/csa.def extension .def
Reading DEF data from file /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/placement/csa.def.
This action cannot be undone.
  Processed 3 vias total.
  Processed 69 subcell instances total.
  Processed 16 pins total.
  Processed 2 special nets total.
  Processed 45 nets total.
DEF read: Processed 311 lines.
Using technology "TECHNAME", version REVISION
Root cell box:
           width x height  (   llx,  lly  ), (   urx,  ury  )  area (units^2)

microns:   0.010 x 0.010   ( 0.000,  0.000), ( 0.010,  0.010)  0.000
lambda:        1 x 1       (     0,  0    ), (     1,  1    )  1
Main console display active (Tcl8.6.10 / Tk8.6.10)
```

Fig.11 Placement Process

C. ROUTING:

Routing is the process of connecting the placed standard cells using metal layers to form the actual signal paths. The tool automatically created the necessary interconnections for power, ground, and signals, ensuring that all design rules are met.



```
C:\Windows\System32>wsl -d ubuntu
root@AldrinPC:/mnt/c/Windows/System32# cd /
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/floorplan/csa.def &
[1] 950
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/placement/csa.def &
[2] 960
root@AldrinPC:/# sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read /OpenLane/designs/csa/runs/R
UN_2025.03.26_09.26.43/tmp/merged.min.lef def read /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/re
sults/routing/csa.def &
[3] 972
root@AldrinPC:/#
```

```
LEF read, Line 251 (Message): Unknown keyword "MAXIMUMDENSITY" in LEF file; ignoring.
LEF read, Line 252 (Message): Unknown keyword "DENSITYCHECKWINDOW" in LEF file; ignoring.
LEF read, Line 253 (Message): Unknown keyword "DENSITYCHECKSTEP" in LEF file; ignoring.
LEF read, Line 290 (Message): Unknown keyword "ANTENNAMODEL" in LEF file; ignoring.
LEF read, Line 291 (Message): Unknown keyword "ANTENNADIFFSIDEAREARATIO" in LEF file; ignoring.
LEF read: Processed 56276 lines.
handling file entry def extension
Cell def couldn't be read
No such file or directory
Creating new cell
handling file entry read extension
handling file entry /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/routing/csa.def extension .def
Reading DEF data from file /OpenLane/designs/csa/runs/RUN_2025.03.26_09.26.43/results/routing/csa.def.
This action cannot be undone.
  Processed 3 vias total.
  Processed 131 subcell instances total.
  Processed 16 pins total.
  Processed 2 special nets total.
  Processed 45 nets total.
DEF read: Processed 821 lines.
Using technology "TECHNAME", version REVISION
Root cell box:
           width x height  (   llx,  lly  ), (   urx,  ury  )  area (units^2)

microns:   0.010 x 0.010   ( 0.000,  0.000), ( 0.010,  0.010)  0.000
lambda:        1 x 1       (     0,  0    ), (     1,  1    )  1
Main console display active (Tcl8.6.10 / Tk8.6.10)
```

Fig.12 Routing Process

3. Final Physical Layout:

After completing placement and routing, the **final physical layout** of the Carry Select Adder was generated, showing all cells, interconnections, and metal layers.
The layout was then converted into a **.mag file** using tools like Magic, which is used to view, edit, and verify the physical layout.

4. GDSII File Format & Klayout:

The **GDS (Graphic Data System) file** is the final output of the ASIC design flow and the close to fabrication design, containing all geometric information of the chip layout, including layers, cells, and routing. It is the standard format used for chip fabrication.

KLayout is a powerful open-source layout viewer used to visualize and verify the GDSfile.

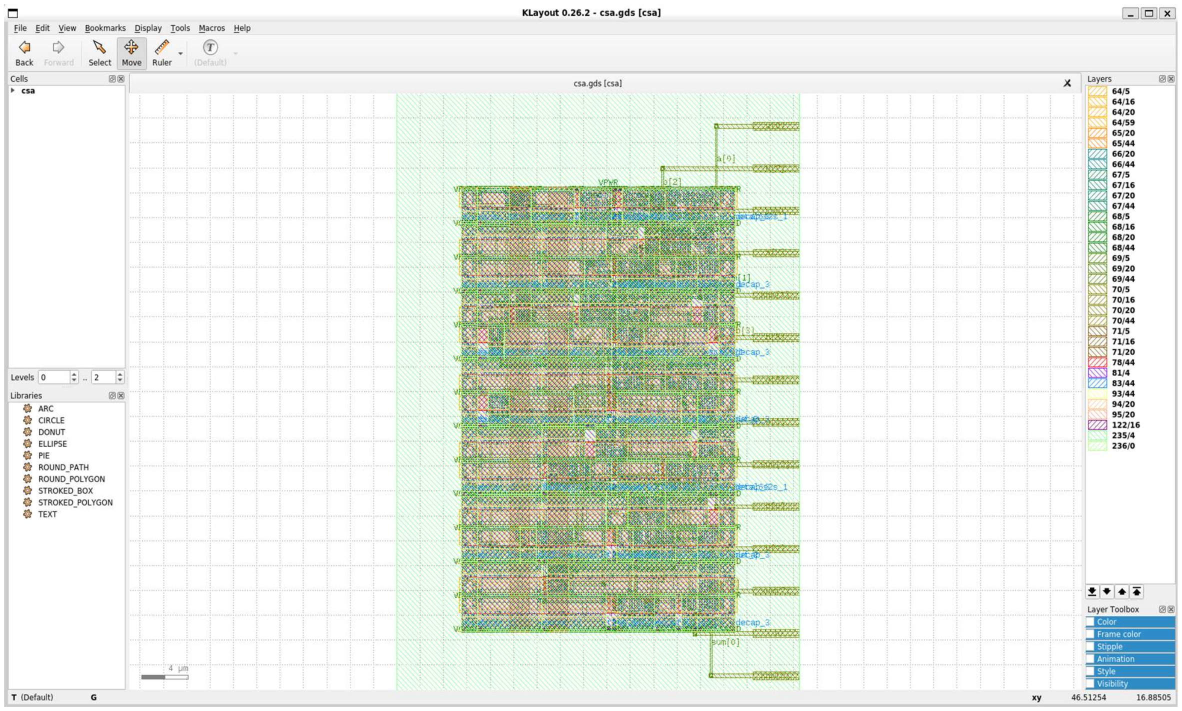


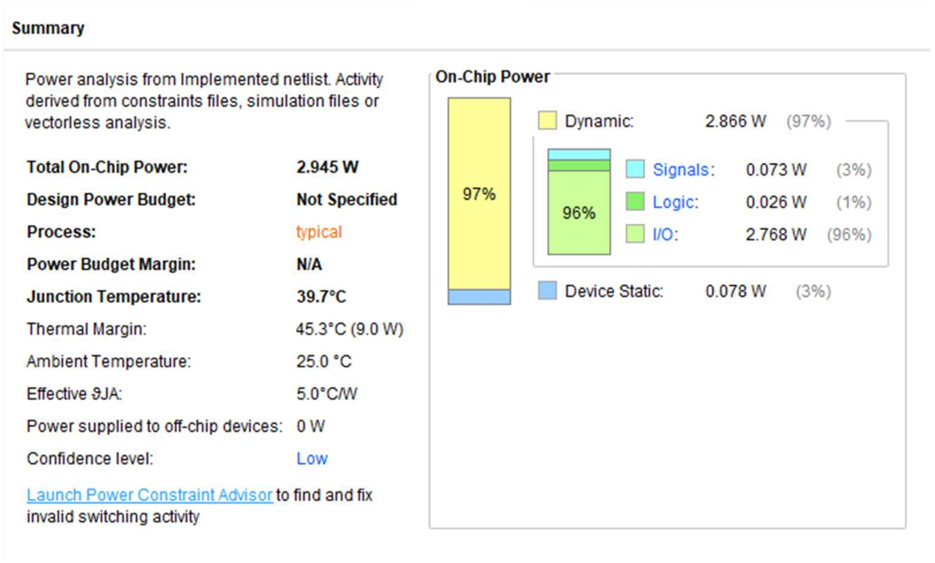
Fig.13 GDS File diagram of the Carry Select Adder

CHATPER 6

RESULTS AND DISCUSSION

I. POWER ANALYSIS:

a. Power Report of Ripple Carry Adder (Traditional Adder)



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**2.945 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**39.7°C**
Thermal Margin:	45.3°C (9.0 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

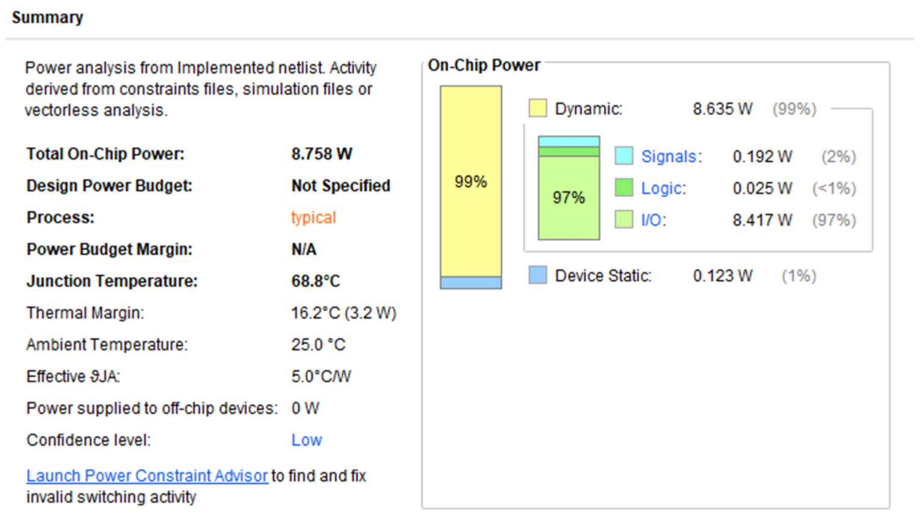
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	2.866 W	(97%)
Signals:	0.073 W	(3%)
Logic:	0.026 W	(1%)
I/O:	2.768 W	(96%)
Device Static:	0.078 W	(3%)

Fig.14 Shows the Power Report of the Ripple Carry Adder

b. Power Report of Carry Select Adder



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**8.758 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**68.8°C**
Thermal Margin:	16.2°C (3.2 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	5.0°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

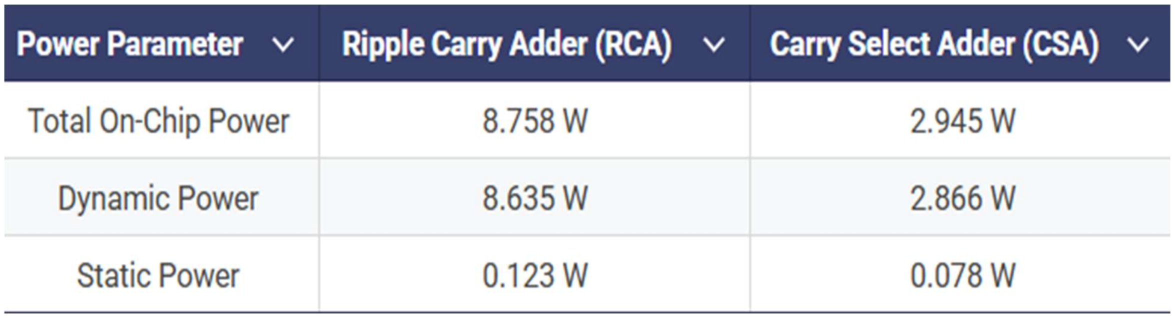
Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:	8.635 W	(99%)
Signals:	0.192 W	(2%)
Logic:	0.025 W	(<1%)
I/O:	8.417 W	(97%)
Device Static:	0.123 W	(1%)

Fig.15 Shows the Power Report of the Carry Select Adder

Comparison Table of Power Consumed by Ripple Carry Adder Vs Carry Select Adder



Power Parameter	Ripple Carry Adder (RCA)	Carry Select Adder (CSA)
Total On-Chip Power	8.758 W	2.945 W
Dynamic Power	8.635 W	2.866 W
Static Power	0.123 W	0.078 W

Fig.16 Shows the comparison in Power for Ripple Carry Adder and the Carry Select Adder

Inference:

The **Carry Select Adder** offers a significant **speed advantage** over the **Ripple Carry Adder** but at the cost of increased **hardware complexity and area utilization**. However, in optimized designs, CSA also provides notable **power savings**, making it a better choice for high-speed digital systems where performance is a priority.

II. PHYSICAL LAYOUT COMPARISION:

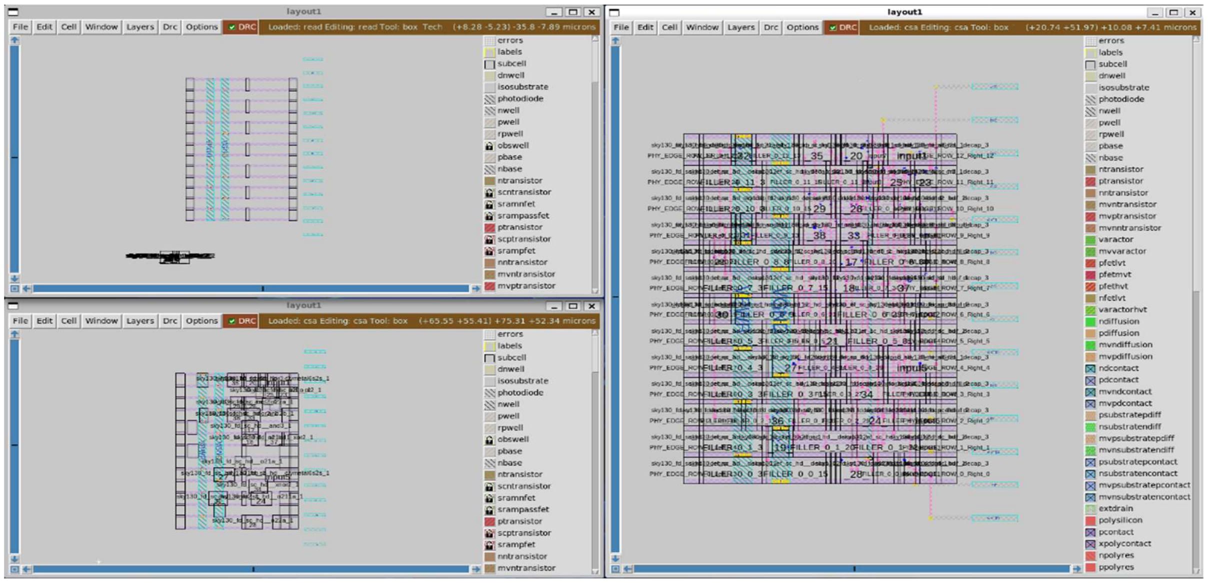

layout1
File Edit Cell Window Layers Drc Options DRC
Loaded: read Editing: read Tool: box
Loaded: csa Editing: csa Tool: box

Fig.17 Shows the Physical Layout generation and comparison

III. RTL TO GDS:

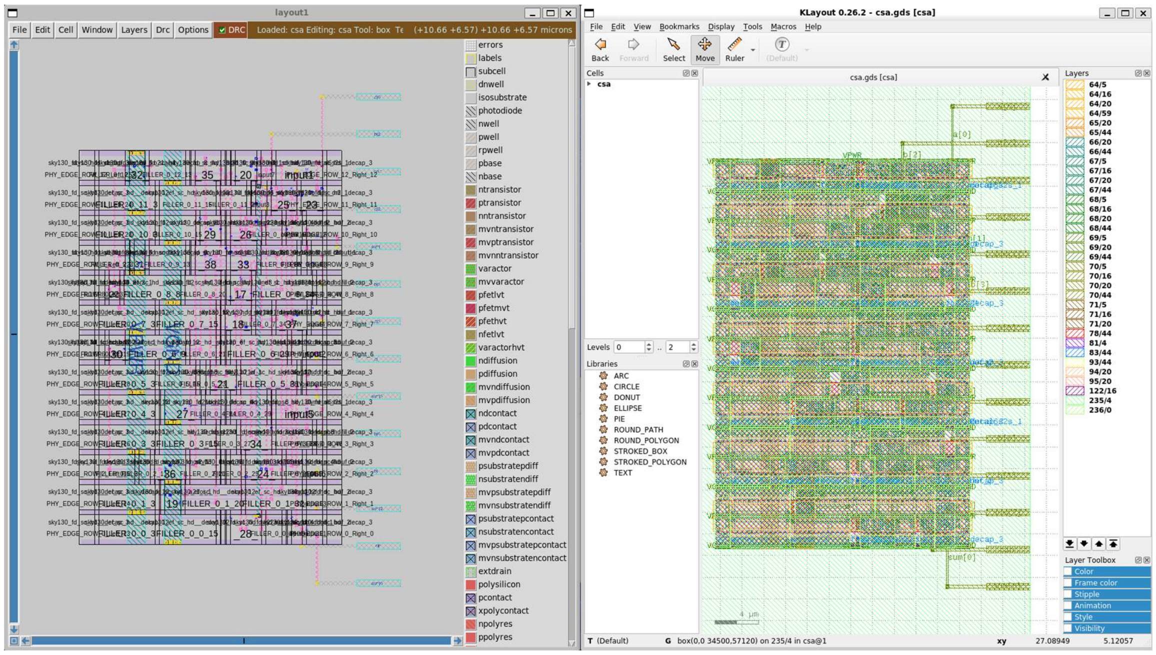


Fig.18 RTL Vs GDS diagram of the Carry Select Adder

Linux Commands Used

GTKWave:(In Windows)

```
cd C:\Users\aldri\Documents\verilog

iverilog -o tb_csa tb_csa.v csa.v

vvp tb_csa

gtkwave tb_csa.vcd
```

Yosys:

```
wsl -d Ubuntu

cd /

cd sky130RTLDesignAndSynthesisWorkshop/DC_WORKSHOP/verilog_files

Yosys

read_liberty -lib
/sky130RTLDesignAndSynthesisWorkshop/lib/sky130_fd_sc_hd__tt_025C_1
v80.lib

read_verilog
/sky130RTLDesignAndSynthesisWorkshop/DC_WORKSHOP/verilog_files/cs
a.v

synth -top csa

abc -liberty
/sky130RTLDesignAndSynthesisWorkshop/lib/sky130_fd_sc_hd__tt_025C_1
v80.lib

show -format ps
```

```
sudo -i

cd /root

ls -la

# cp /root/.yosys_show.dot ~/yosys_graph.dot


dot -Tpng /root/.yosys_show.dot -o csa.png

xdg-open csa.png
```

OpenLane:

```
cd OpenLane

sudo make mount

flow.tcl -design csa

sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/tmp/merged.min.lef def
read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/results/floorplan/csa.def
&

sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/tmp/merged.min.lef def
read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/results/placement/csa.def
&

sudo magic -T /open_pdks/sky130/magic/sky130.tech lef read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/tmp/merged.min.lef def
read
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/results/routing/csa.def &

sudo magic -T /open_pdks/sky130/magic/sky130.tech
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/results/final/mag/csa.ma
g

sudo klayout
/OpenLane/designs/csa/runs/RUN_XXXXXXXXXX/results/final/gds/csa.gds
```

CHATPER 7
CONCLUSION

This project successfully implemented the ASIC flow for a Carry Select Adder (CSA), from Verilog design to physical realization. The CSA was verified through simulation, tested on a Basys3 FPGA, and analyzed for power-speed trade-offs using Xilinx Vivado. The RTL-to-GDSII flow was completed using Yosys and OpenLane with the SkyWater 130nm PDK, covering synthesis, floor planning, placement, and routing. The results highlight the CSA’s speed advantage over ripple adders at the cost of higher power consumption. This project bridges FPGA validation with ASIC design, providing hands-on experience in digital circuit optimization and fabrication readiness.

References:

[1] **Harris, D.M., & Harris, S.L.**
Digital Design and Computer Architecture, 2nd Edition, Morgan Kaufmann, 2012.
(For understanding adder architectures and hardware design basics)

[2] **Weste, N.H.E., & Harris, D.**
CMOS VLSI Design: A Circuits and Systems Perspective, 4th Edition, Pearson, 2011.
(For ASIC design methodology and layout concepts)

[3] **SkyWater Technology Foundry.**
Sky130 PDK Documentation.
https://github.com/google/skywater-pdk
(Official open-source 130nm Process Design Kit used in RTL-to-GDS flow)

[4] **OpenLane Project – Efabless Corporation.**
OpenLane: Open-source ASIC implementation flow.
https://github.com/The-OpenROAD-Project/OpenLane
(Tool used for floorplanning, placement, and routing)

[5] **Clarke, L., & Shukla, S.**
"Comparison of Adder Architectures for Low Power and High Speed Applications", *IEEE Conference on VLSI Design*, 2020.
(For comparative analysis of adder types)

[6] **Brent, R.P., & Kung, H.T.**
"A Regular Layout for Parallel Adders", *IEEE Transactions on Computers*, vol. C-31, no. 3, 1982.
(Original conceptual foundation for Carry Select Adder and similar parallel adder structures)

[7] **Yosys Open SYnthesis Suite**
https://yosyshq.net/yosys/
(Tool used for RTL synthesis of Verilog designs)

[8] **GTKWave**
http://gtkwave.sourceforge.net/
(Tool used for waveform visualization from VCD files)

[9] **KLayout - Layout Viewer for IC Design**
https://www.klayout.de/
(Used to view and verify GDSII layouts)