4-bit Carry Save Multiplier RTL to GDS

Project Report

Submitted by
Your Name:R sevya naik
Roll No: 123EC0056
Department of Electronics and Communication Engineering
IIITDM Kurnool

Under the Guidance of
Dr. Ranga Babu

November 2025

Contents

1. Introduction

This project presents the RTL-to-GDSII design flow of a **4-bit Carry Save Multiplier (CSM)**. The objective is to design, verify, synthesize, and implement the multiplier on FPGA and ASIC platforms. The design was verified in Vivado, simulated using behavioral testbenches, and synthesized for hardware implementation.

2. Design Flow

The steps followed for the 4-bit Carry Save Multiplier are:

1. RTL design using Verilog HDL.
2. Functional verification using testbench and simulation.
3. Synthesis and implementation on FPGA (ZedBoard).
4. Post-synthesis simulation and timing analysis.
5. Physical layout (place & route) and generation of GDSII.

3. RTL Design (Source Code)

The Verilog source code for the 4-bit Carry Save Multiplier is given below:

Listing 1: Verilog RTL of 4-bit Carry Save Multiplier

```verilog
module carry_save_multiplier_4bit (
    input  [3:0] A,
    input  [3:0] B,
    output [7:0] P
);
    wire [3:0] pp0, pp1, pp2, pp3;
    wire [7:0] s1, s2, s3;

    // Partial Products
    assign pp0 = A & {4{B[0]}};
    assign pp1 = A & {4{B[1]}};
    assign pp2 = A & {4{B[2]}};
    assign pp3 = A & {4{B[3]}};

    // Align partial products
    assign s1 = {4'b0, pp0};
    assign s2 = {3'b0, pp1, 1'b0};
    assign s3 = {2'b0, pp2, 2'b0};
    wire [7:0] pp4 = {1'b0, pp3, 3'b0};

    // Stage-wise addition
    wire [7:0] sum1, sum2, sum3, carry1, carry2, carry3;
    assign {carry1, sum1} = s1 + s2;
    assign {carry2, sum2} = sum1 + s3;
    assign {carry3, sum3} = sum2 + pp4;

    // Final output
    assign P = sum3 + carry3;
endmodule
```

4. Testbench

The testbench used for functional verification is shown below:

Listing 2: Testbench for 4-bit CSM

```
`timescale 1ns / 1ps
module tb_carry_save_multiplier_4bit;
    reg [3:0] A, B;
    wire [7:0] P;

    carry_save_multiplier_4bit uut (.A(A), .B(B), .P(P));

    initial begin
        $monitor("Time=%0t A=%b B=%b -> P=%b (%d)", $time, A, B, P, P);
        A=4'b0011; B=4'b0101; #10;
        A=4'b1111; B=4'b1111; #10;
        A=4'b1010; B=4'b0110; #10;
        A=4'b0001; B=4'b0010; #10;
        A=4'b1000; B=4'b0011; #10;
        $finish;
    end
endmodule
```

5. FPGA Pin Mapping (XDC Constraints)

The XDC file connects switches as inputs and LEDs as outputs on the ZedBoard.

Listing 3: XDC Pin Mapping

```
## SWITCHES for Inputs A[3:0] and B[3:0]
```

```
set_property PACKAGE_PIN F22 [get_ports {A[0]}]
set_property PACKAGE_PIN G22 [get_ports {A[1]}]
set_property PACKAGE_PIN H22 [get_ports {A[2]}]
set_property PACKAGE_PIN F21 [get_ports {A[3]}]
set_property PACKAGE_PIN H19 [get_ports {B[0]}]
set_property PACKAGE_PIN H18 [get_ports {B[1]}]
set_property PACKAGE_PIN H17 [get_ports {B[2]}]
set_property PACKAGE_PIN M15 [get_ports {B[3]}]

## LEDs for Output P[7:0]
set_property PACKAGE_PIN T22 [get_ports {P[0]}]
set_property PACKAGE_PIN T21 [get_ports {P[1]}]
set_property PACKAGE_PIN U22 [get_ports {P[2]}]
set_property PACKAGE_PIN U21 [get_ports {P[3]}]
set_property PACKAGE_PIN V22 [get_ports {P[4]}]
set_property PACKAGE_PIN W22 [get_ports {P[5]}]
set_property PACKAGE_PIN U19 [get_ports {P[6]}]
set_property PACKAGE_PIN U14 [get_ports {P[7]}]
```

6. Simulation Results

The simulation was performed in Vivado. The waveform below shows correct multiplication outputs.

7. Schematic and Layout

The synthesized RTL schematic and physical layout views are shown below.

8. FPGA Implementation

The design was implemented on the Digilent ZedBoard FPGA. The product output is displayed on LEDs.

9. Results

The 4-bit Carry Save Multiplier successfully computes partial products and sums them in a carry-save manner. Simulation and hardware outputs confirm correct operation.

- RTL and behavioral simulation verified in Vivado.

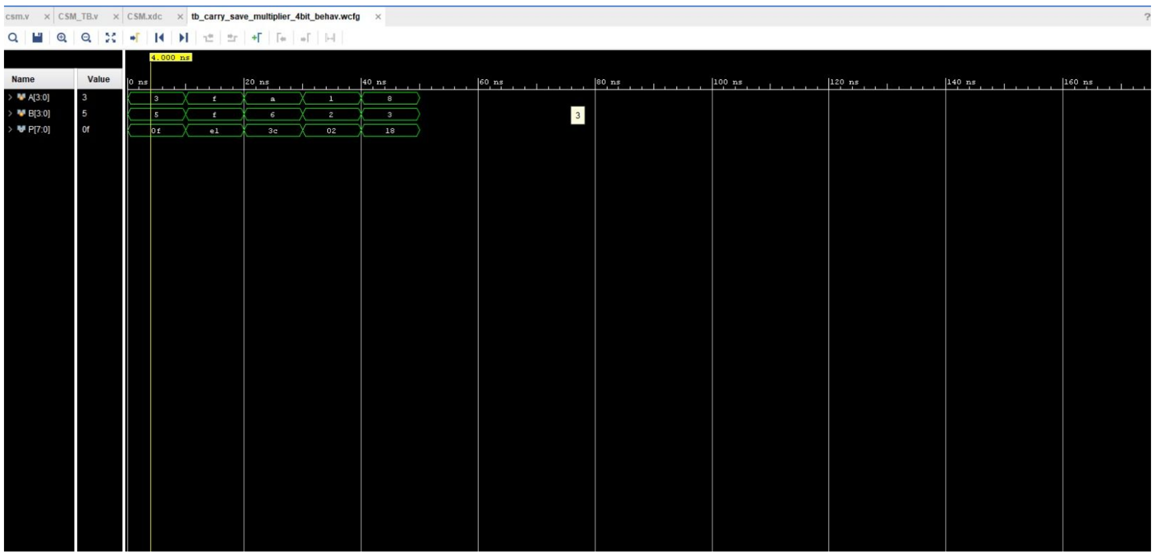

Figure 1: Functional Simulation Waveform in Vivado

- FPGA implementation successfully performed on ZedBoard.
- Layout and GDSII generated after synthesis and P&R.

10. Conclusion

The 4-bit Carry Save Multiplier was designed and implemented using Verilog HDL, verified through simulation, and realized on FPGA hardware. The design can be extended to 8-bit or 16-bit versions for higher performance.

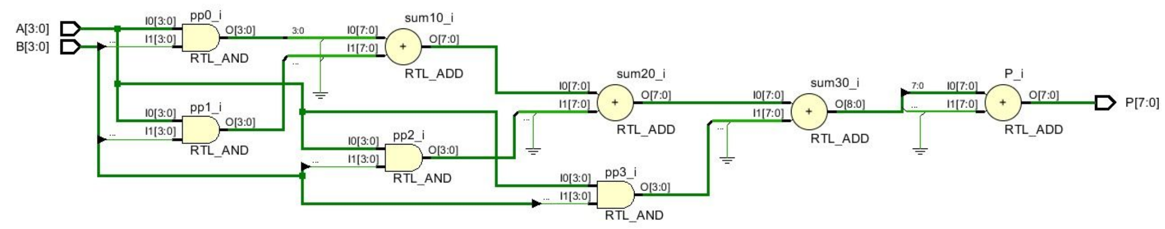

Figure 2: RTL Schematic of 4-bit Carry Save Multiplier

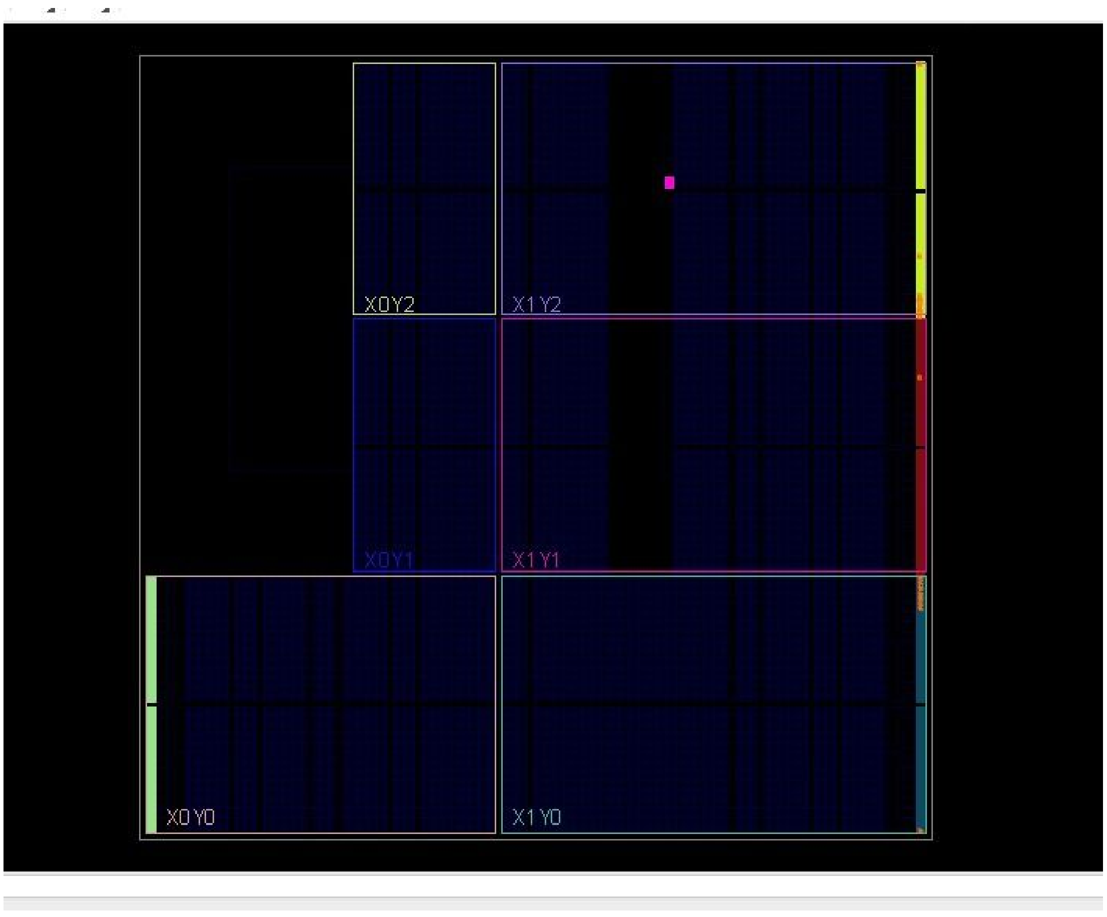

Figure 3: Layout View after Place and Route

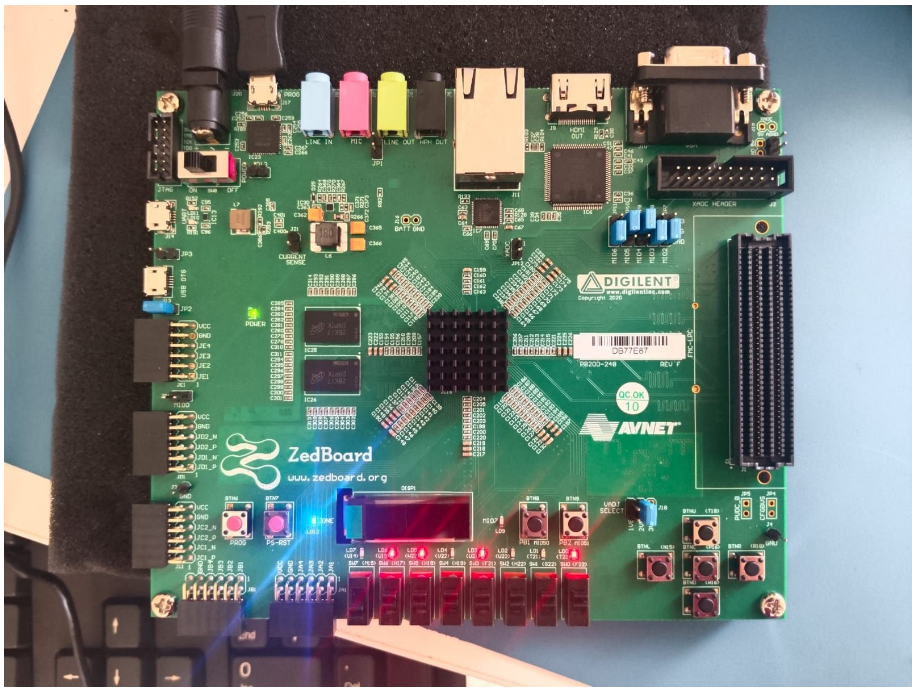

Figure 4: Hardware Implementation on ZedBoard