Design and Layout Implementation with Parasitic Extraction of a 4-bit Array Multiplier Using CMOS GPDK090 in Cadence Virtuoso

Zailesh A R
School of Electronics Engineering
Vellore Institute of Technology, Vellore
Email: zaileshar@gmail.com

***Abstract*—This paper presents the complete transistor-level design, physical layout, and post-layout evaluation of a 4-bit array multiplier implemented using the GPdk090 CMOS technology in the Cadence Virtuoso environment. The work follows a full-custom digital IC design flow, starting from schematic capture of basic CMOS logic gates and progressing through hierarchical construction of arithmetic blocks, functional verification, physical layout generation, design-rule checking (DRC), layout-versus-schematic (LVS) validation, and RC parasitic extraction.**

The multiplier is constructed using custom-designed CMOS inverter, NAND, and XOR gates, which are used to build a 1-bit full adder and a 4-bit ripple carry adder. These blocks form the core of the array multiplier architecture, where partial products are generated and summed using a regular adder array. Pre-layout simulations using Spectre confirm correct logical behavior and stable signal transitions.

Post-layout RC parasitic extraction is performed using Cadence Quantus, resulting in an extracted netlist containing hundreds of MOS devices and tens of thousands of parasitic components. Post-layout simulations demonstrate the expected increase in delay and switching energy compared to schematic-level results. The final layout is DRC- and LVS-clean and occupies an area of approximately 2085.7 μm^2. This work demonstrates a complete and industry-relevant full-custom CMOS design methodology suitable for arithmetic building blocks.

***Index Terms*—CMOS, Array Multiplier, GPdk090, Cadence Virtuoso, RC Extraction, Full-Custom VLSI, DRC/LVS**

I. Introduction

Multiplication is a fundamental arithmetic operation in digital signal processing, computer arithmetic units, and hardware accelerators used in modern computing systems. The performance of a multiplier strongly affects the overall throughput, latency, area, and power consumption of arithmetic datapaths. Consequently, efficient multiplier design remains a critical aspect of digital integrated circuit (IC) development.

This work focuses on the design and implementation of a 4-bit array multiplier using a full-custom CMOS design approach. Array multipliers are well known for their regular structure, predictable timing behavior, and straightforward mapping to physical layout. Although they are not optimal for large bit-width, high-speed applications, they provide an excellent platform for demonstrating complete VLSI design flows, including schematic design, layout, verification, and parasitic-aware simulation.

The objectives of this work are:

- Design and implement basic CMOS logic gates (inverter, NAND, XOR) at the transistor level.
- Construct a full adder using these gate-level primitives and cascade full adders to form a 4-bit ripple carry adder.
- Design a 4-bit array multiplier using partial-product generation and ripple-based summation.
- Perform functional verification through transient simulation.
- Generate physical layouts for all blocks and verify them using DRC and LVS.
- Perform RC parasitic extraction and analyze post-layout behavior.

This paper emphasizes physical design awareness and parasitic effects, which are essential for realistic silicon implementation.

The authors are with the School of Electronics Engineering, Vellore Institute of Technology, Vellore, India (M.Tech VLSI Design).

II. Design Methodology and Device-Level Background

This section describes the transistor-level design of the fundamental logic blocks used in the multiplier. Each block was individually designed, simulated, and verified prior to hierarchical integration.

A. CMOS Inverter

The CMOS inverter is the most fundamental building block in digital CMOS logic. It consists of a PMOS transistor connected to the positive supply voltage (VDD) and an NMOS transistor connected to ground. The complementary operation of these devices provides rail-to-rail output swing and high noise margins.

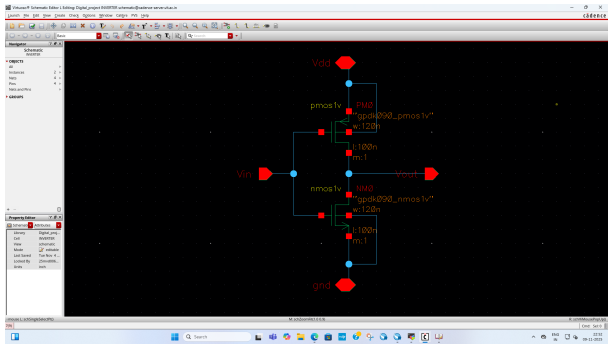

Fig. 1: CMOS inverter schematic implemented in Cadence Virtuoso

Figure explanation: The figure shows the transistor-level schematic of the CMOS inverter used in this work. The PMOS transistor forms the pull-up network, while the NMOS transistor forms the pull-down network. The input signal drives both transistor gates, and the output is taken from the common drain node. This cell was used to validate the GPdk090 device models and to establish layout conventions such as well placement, diffusion sharing, and contact usage.

B. NAND Gate

The NAND gate was implemented using stacked NMOS transistors for the pull-down network and parallel PMOS transistors for the pull-up network. NAND gates are universal logic gates and are widely used due to their compact CMOS implementation.

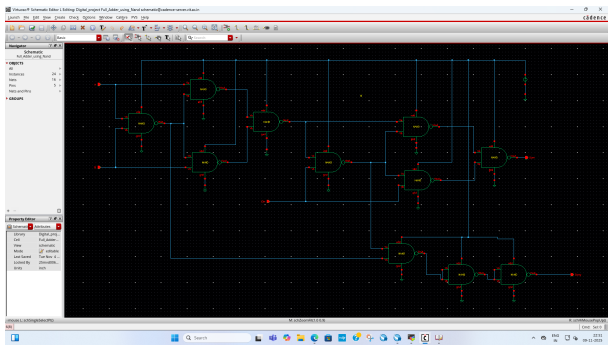

Fig. 2: NAND gate schematic using CMOS transistors

Figure explanation: This schematic illustrates the transistor-level realization of the NAND gate. In layout, diffusion sharing between series NMOS devices was employed to reduce silicon area and routing complexity. The NAND gate serves as a key building block for higher-level logic, including XOR gates and full adders.

C. XOR Gate

The XOR gate was realized using a structural composition of NAND gates and inverters. This approach promotes modular design and reuse of previously verified subcircuits.

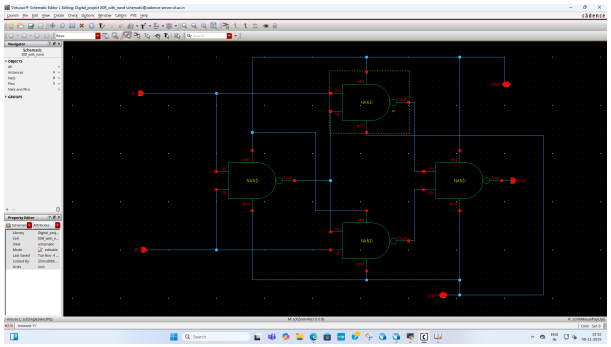

Fig. 3: XOR gate implementation using NAND gates

Figure explanation: The figure shows the gate-level structure of the XOR circuit constructed from NAND gates. XOR functionality is essential for sum generation in arithmetic circuits. Implementing XOR using NAND gates ensures consistency with the chosen logic style and simplifies hierarchical verification.

III. Adder Design and Hierarchical Construction

A. Full Adder

A full adder adds three one-bit inputs, namely A, B, and Carry-in (Cin), and produces a Sum and Carry-out (Cout). The full adder is a fundamental arithmetic block used extensively in adders, multipliers, and arithmetic logic units.

In this work, the full adder was constructed using the previously designed XOR and NAND gates to maintain a consistent logic style and enable hierarchical verification.

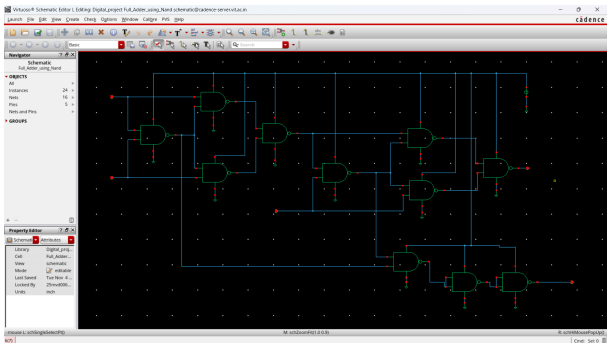

Fig. 4: Full adder schematic constructed using NAND and XOR gates

Figure explanation: This figure shows the hierarchical schematic of the full adder. The XOR gates generate the sum output, while NAND-based logic implements the carry generation and propagation paths. Each submodule was first simulated independently and then verified as part of the complete adder to ensure correct logical behavior prior to integration into higher-level blocks.

B. 4-bit Ripple Carry Adder

A ripple carry adder (RCA) is formed by cascading multiple full adders such that the carry-out of each stage feeds the carry-in of the next higher bit position. Although ripple carry

adders are not optimal for high-speed designs due to linear carry propagation delay, they are straightforward to implement and map cleanly to layout.

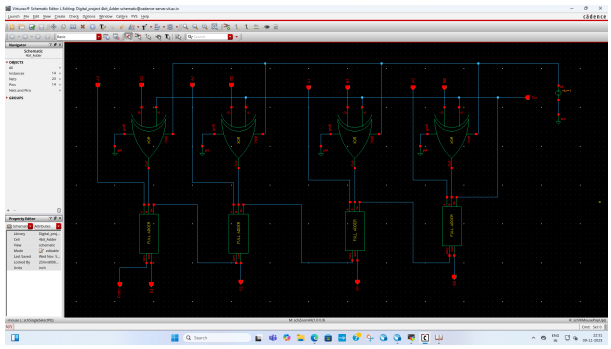

Fig. 5: 4-bit ripple carry adder schematic constructed using NAND and XOR gates

Figure explanation: The schematic illustrates how the carry signal propagates sequentially through the four full adder stages. For a 4-bit design, the resulting delay is acceptable and provides a compact, easily routable solution suitable for educational and small-scale arithmetic implementations.

IV. 4-bit Array Multiplier Implementation

The 4-bit array multiplier was constructed using a regular arrangement of partial-product generators and adders. Each bit of the multiplier operand generates a row of partial products by ANDing with all bits of the multiplicand. These partial products are then summed column-wise using half adders and full adders as required.

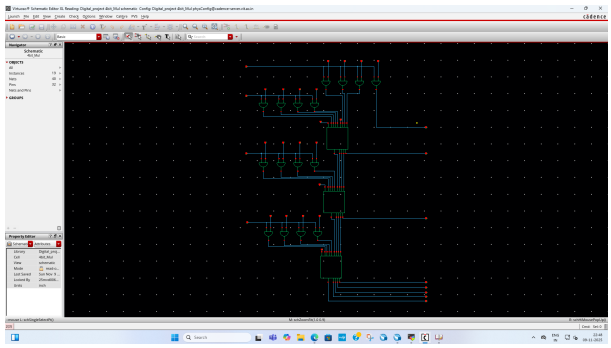

Fig. 6: Final top-level schematic of the 4-bit array multiplier

Figure explanation: The schematic shows the complete hierarchical integration of the array multiplier. Partial-product generation blocks are arranged in a 4×4 grid, and the resulting signals are routed into a regular network of adder blocks. Vertical and diagonal interconnections represent the alignment and shifting of partial products corresponding to their binary weights. This regular structure simplifies verification and layout placement.

V. Array Multiplier Theory and Architecture

A. Principle of Operation

An array multiplier performs multiplication by generating partial products and summing them using an array of adders arranged in a grid-like structure. For two 4-bit operands A[3:0] and B[3:0], each bit of operand B produces a row of partial products. These rows are appropriately shifted and summed to form the final 8-bit product.

B. Partial Product Generation and Summation

Partial products were generated using two-input AND gates implemented at the transistor level. The regularity of the array multiplier allows identical cells to be repeated across rows and columns, simplifying both schematic capture and layout implementation. Column-wise summation is performed using a combination of half adders and full adders, depending on the number of bits present in each column.

C. Design Trade-offs and Alternatives

Array multipliers offer predictable timing and regular layout but scale poorly in area and delay for larger bit widths. Alternative architectures such as Wallace tree and Booth multipliers reduce the number of adder stages and improve performance at the cost of increased complexity. For a compact 4-bit design, the array multiplier provides an excellent balance between simplicity and demonstrable layout practices.

D. Representative Input–Output Combinations

Table I lists representative test cases used during functional verification of the multiplier.

TABLE I: Representative 30 cases from the 4-bit array multiplier truth table

A (4-bit)	B (4-bit)	Product (8-bit)
0000	0000	00000000
0000	0001	00000000
0001	0000	00000000
0001	0001	00000001
0001	0010	00000010
0010	0011	00000110
0011	0011	00001001
0100	0101	00010100
0101	0010	00001010
0110	0111	00101010
0111	1000	00111000
1000	1000	01000000
1001	1001	01010001
1010	1010	01100100
1011	0101	00110111
1100	1100	10010000
1101	1101	10101001
1110	1111	11010010
1111	1111	11100001
0111	0111	00110001
0011	0101	00001111
0101	0101	00011001
0010	1000	00010000
0100	0100	00010000
1111	0001	00001111
0001	1111	00001111
1110	0001	00001110
0001	1110	00001110
1100	0011	00100100
1001	0111	00111111

VI. Simulation Strategy and Results

This section describes the simulation methodology adopted for functional verification and power estimation of the 4-bit array multiplier. All simulations were performed in the Cadence Virtuoso environment using Spectre with GPdk090 device models.

A. Simulation Setup

Transient simulations were carried out for each hierarchical block starting from basic logic gates and progressing to the complete multiplier. Gate-level verification was first performed to validate correct logical operation. Higher-level blocks were then simulated using structured input vectors to verify signal propagation and functional correctness across the hierarchy.

Inputs were toggled to cover boundary conditions such as zero operands, maximum operand values, and mid-range combinations that stress carry propagation paths. This approach ensures that both typical and worst-case switching scenarios are exercised.

B. Representative Waveforms

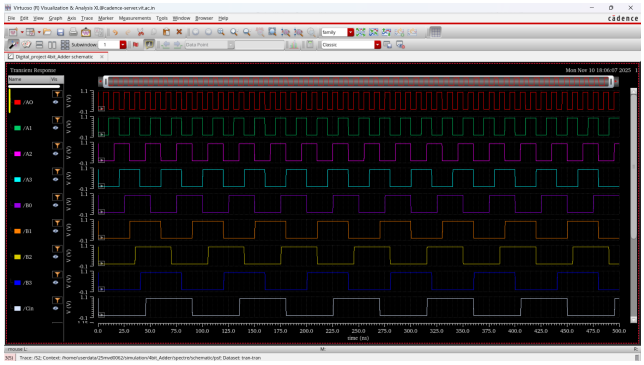

Fig. 7: Input excitation waveform for the 4-bit full adder (A0–A3, B0–B3, and Cin)

Figure explanation: This waveform shows the applied input stimulus used for verifying the 4-bit full adder. The multiplicand inputs A0–A3 and the multiplier inputs B0–B3 toggle in a structured sequence to systematically cover multiple input combinations. The Cin signal introduces carry transitions to validate correct carry propagation through the adder chain. Clean input transitions ensure reliable functional verification.

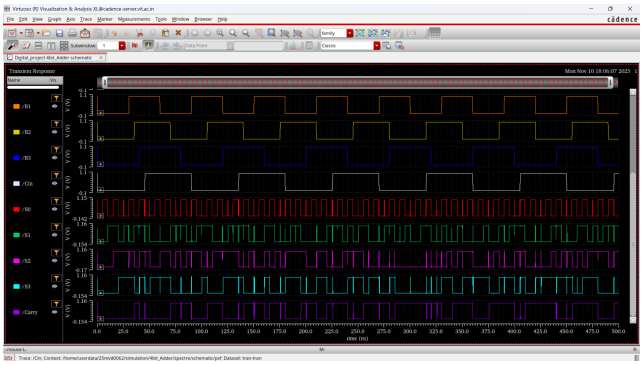

Fig. 8: Transient simulation waveform of the 4-bit full adder showing sum outputs and carry-out

Figure explanation: This waveform illustrates the transient response of the full adder outputs. The sum bits respond correctly to changes in input combinations, and the carry-out signal reflects proper propagation of carries across bit positions. The absence of glitches or undefined states confirms correct hierarchical integration of XOR and NAND logic.

Fig. 9: 4-bit array multiplier result waveform

Figure explanation: This waveform demonstrates correct multiplication results for selected input vectors. Partial product signals and intermediate carries were monitored to ensure correct column-wise summation timing and stable output behavior.

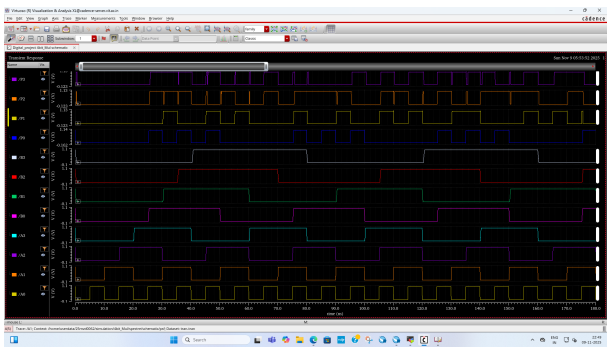

Fig. 10: Simulation waveform showing inputs and outputs of the 4-bit multiplier

Figure explanation: Zoomed waveform views were used to verify bit-level timing relationships and to ensure that no

spurious transitions occur on the output after input changes settle.

Fig. 11: Zoomed waveform illustrating bit-level transitions

Figure explanation: Examination of carry propagation across the ripple adders confirmed expected sequential behavior, where higher-order sum bits change only after lower-order carries have settled.

C. Power Estimation

Power estimation provides quantitative insight into the switching activity and overall efficiency of the implemented CMOS multiplier. The transient simulations generate the instantaneous power trace at the VDD node, which is used to compute average and peak power values.

1) Measurement Methodology: Instantaneous power at any time instant t is given by

$$P(t) = V_{DD}(t) \times I_{DD}(t),$$

where V_{DD} and I_{DD} represent the instantaneous supply voltage and current, respectively.

The total energy consumed over a time window T is

$$E = \int_0^T P(t)\, dt,$$

and the average power is

$$P_{\text{avg}} = \frac{1}{T} \int_0^T P(t)\, dt.$$

For discrete sampled simulation data with time step Δt,

$$P_{\text{avg}} \approx \frac{1}{N} \sum_{i=1}^{N} P(t_i), \quad E \approx \sum_{i=1}^{N} P(t_i)\, \Delta t,$$

where N is the total number of samples.

For CMOS logic, total power can be expressed as

$$P_{\text{total}} = P_{\text{dyn}} + P_{\text{sc}} + P_{\text{leak}},$$

where P_{dyn} is dynamic switching power, P_{sc} is short-circuit power, and P_{leak} is leakage power. The dominant term for digital CMOS circuits is dynamic power:

$$P_{\text{dyn}} = \alpha C_L V_{DD}^2 f,$$

where α is the activity factor, C_L is the load capacitance, and f is the switching frequency.

2) Measurement in Cadence: Power measurements were performed using Cadence ADE and Visualization XL. The following expressions and tools were used:

- `average(getData("pwr" ?result "tran"))` to compute average power.
- Integration of the `:pwr` waveform to compute total energy.
- Visual inspection of waveform peaks to identify maximum instantaneous power.

From the transient simulation results:

- Measured average power at VDD: **11.43 µW**.
- Peak instantaneous power: approximately **550–600 µW**.
- Total simulation window: **180 ns**.

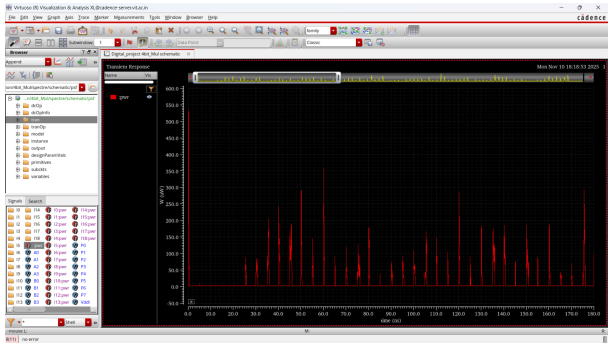

Fig. 12: Total instantaneous power trace at VDD

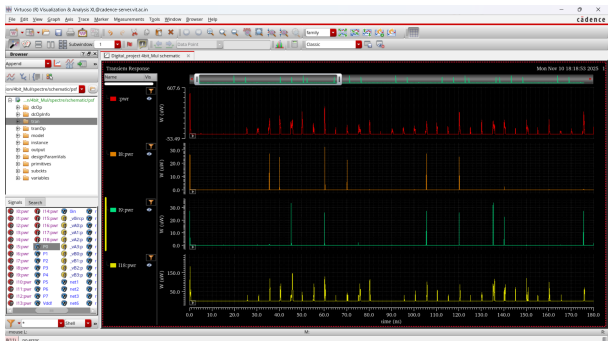

Fig. 13: Per-net power traces of partial-product rows

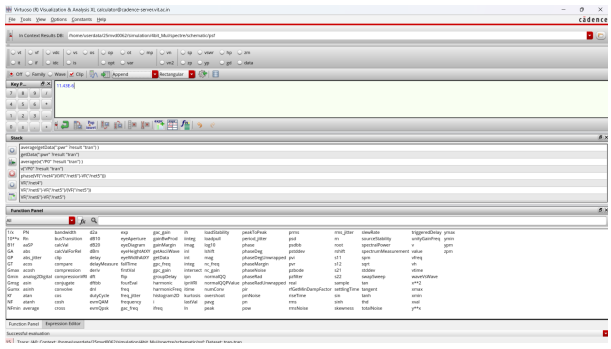

Fig. 14: Spectre calculator used for numerical computation of average power

Fig. 15: Zoomed-in transient interval highlighting power spikes

3) Representative Power Waveforms:

TABLE II: Measured power and energy results for the 4-bit array multiplier

Measured Item	Energy (pJ)	Average Power (µW)
Total (Chip)	2.06	11.43
Row 0 (Partial Products)	0.42	2.33
Row 1 (Partial Products)	0.47	2.61
Row 2 (Partial Products)	0.56	3.11
Row 3 (Partial Products)	0.61	3.38

4) Results and Discussion: Higher-order partial-product rows consume more dynamic power due to increased switching activity and carry propagation.

5) Interpretation: Peak power corresponds to concurrent switching across multiple adder stages, while the lower baseline reflects idle periods between vector transitions. The measured energy per multiplication operation is approximately **2 pJ**, which is consistent with expected values for a 90 nm CMOS implementation of this scale.

VII. Layout Design, Verification, and Extraction

This section describes the physical implementation of the multiplier and the verification steps performed to ensure manufacturability and correctness.

A. Layout Methodology

Layouts were generated hierarchically, starting from basic logic gates and progressing to higher-level arithmetic blocks. Careful attention was paid to diffusion sharing, transistor orientation, well placement, and routing symmetry to minimize area and parasitic loading. Power rails were routed using wide metal lines to reduce IR drop and ensure reliable operation.

B. Basic Cell Layouts

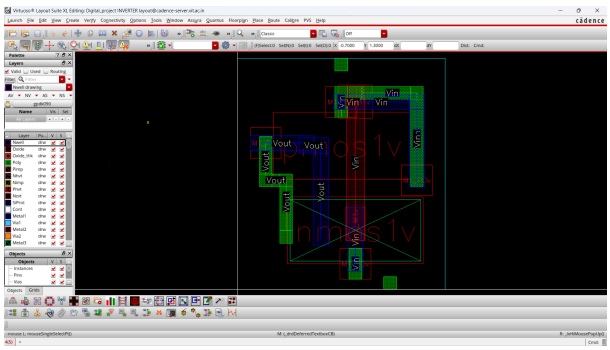

Fig. 16: Transistor-level layout of the CMOS inverter

Figure explanation: This layout shows the physical implementation of a CMOS inverter using one PMOS and one NMOS transistor. Poly gates define the input terminal, while diffusion regions form the output node. This cell establishes layout conventions used throughout the design.

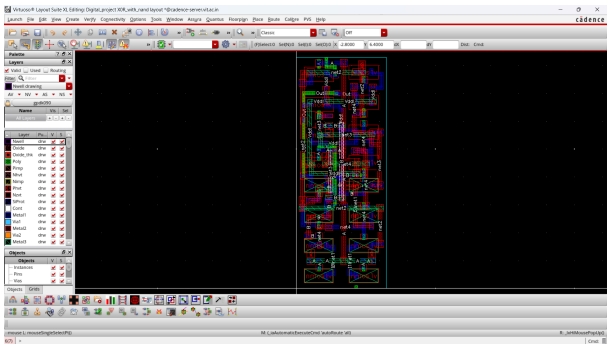

Fig. 17: Layout of the XOR gate implemented using NAND and inverter cells

Figure explanation: The XOR layout is constructed hierarchically using NAND and inverter subcells. Diffusion sharing and structured placement reduce area and parasitic capacitance.

C. Full Adder Layout

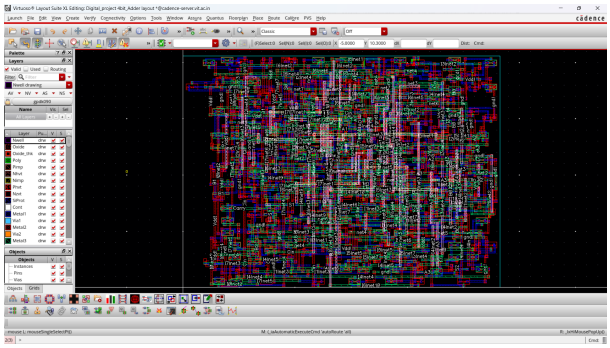

Fig. 18: Complete layout of the full adder cell

Figure explanation: This layout integrates XOR, NAND, and inverter subcells to form a 1-bit full adder. The regular

structure enables easy tiling in higher-level blocks such as ripple carry adders.

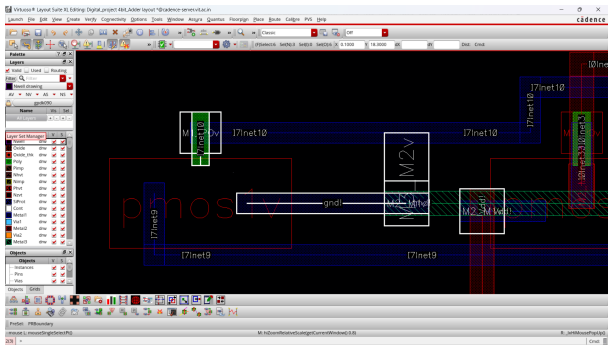

Fig. 19: Zoomed view of the full adder layout

Figure explanation: The zoomed view highlights transistor-level routing and via placement, illustrating how parasitic loading is minimized through short interconnects.

D. 4-bit Array Multiplier Layout

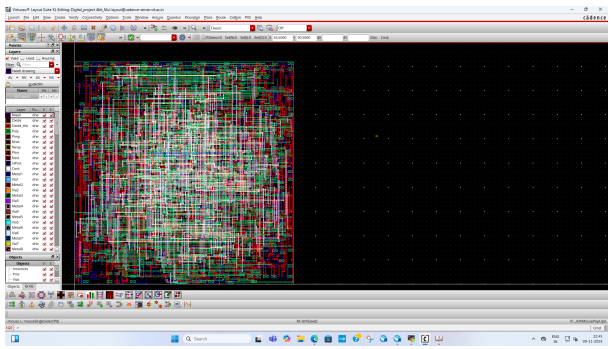

Fig. 20: Complete layout of the 4-bit array multiplier

Figure explanation: The complete layout shows replicated AND gates and full adder blocks arranged in a regular grid. Power rails run horizontally, while vertical routing carries partial products and carries across adder columns.

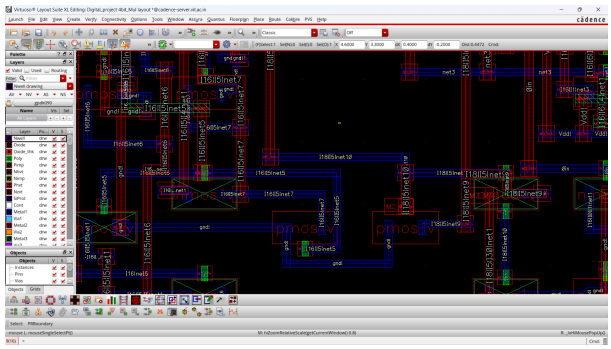

Fig. 21: Zoomed layout view of the 4-bit array multiplier

Figure explanation: This zoomed view highlights dense routing regions where partial products feed into cascaded adders.

E. DRC and LVS Verification

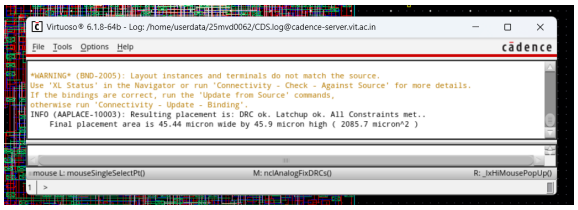

Fig. 22: Cadence Virtuoso DRC and LVS clean status

Figure explanation: The DRC and LVS reports confirm that the layout satisfies all GPdk090 design rules and matches the schematic connectivity. The final layout dimensions are approximately 45.44 µm × 45.9 µm, corresponding to an area of 2085.7 µm^2.

F. Parasitic Extraction

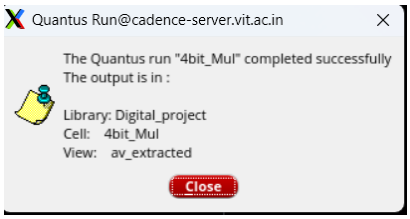

Fig. 23: Quantus parasitic extraction completion report

Figure explanation: This report confirms successful generation of the extracted view (`av_extracted`) using Cadence Quantus.

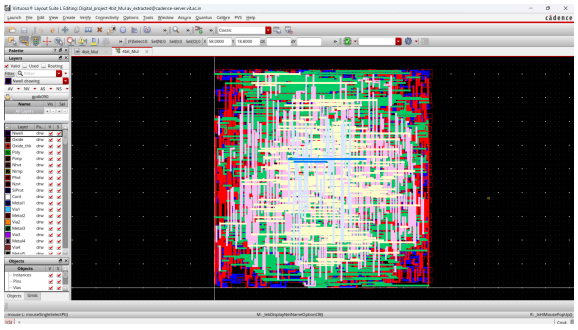

Fig. 24: Extracted layout view with parasitic annotations

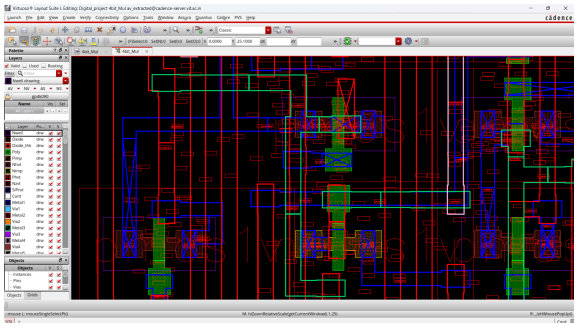

Fig. 25: Zoomed view showing parasitic components

G. Extracted Netlist Statistics

TABLE III: Summary of extracted netlist properties

Total MOSFET devices	702
Total parasitic capacitors	78,754
Total extracted capacitance	≈1.27 pF
Extraction mode	RC (av_extracted)
Netlist format	Spectre
Technology	GPdk090

The extracted netlist includes device parasitics, interconnect resistances, and coupling capacitances, enabling realistic post-layout simulation.

VIII. Conclusion

This paper presented a complete full-custom CMOS implementation of a 4-bit array multiplier using the GPdk090 technology. The design flow covered schematic capture, functional verification, physical layout, DRC and LVS verification, and RC parasitic extraction. Post-layout analysis demonstrated realistic power behavior and confirmed functional correctness under parasitic loading.

The structured and hierarchical approach adopted in this work enables straightforward scalability to higher-bit arithmetic units and provides a strong foundation for future exploration of optimized multiplier architectures.

References

[1] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd ed. Prentice Hall, 2003.

[2] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley, 2011.

[3] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits*, McGraw-Hill, 2003.

[4] R. J. Baker, *CMOS: Circuit Design, Layout, and Simulation*, Wiley-IEEE, 2010.

[5] Cadence Design Systems, *Virtuoso Layout Suite and GPdk090 Process Design Kit Documentation*, 2024.