Physical Design and Post-Layout Verification of an 8-bit Wallace-Tree Multiplier with TG Adders and 16-bit CLA Adder

Ben Li (Ming Yao Li)
Dept. of Engineering and Management
Purdue University
West Lafayette, Indiana, USA
li5277@purdue.edu

Andrew Lien (Sung En Lien)
Dept. of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana, USA
lien1@purdue.edu

***Abstract*—The multiplier is a major operation block in any processing unit. Among various multiplication architectures, Wallace tree multiplier is the most beneficial in terms of speed of operation. This project presents the design, layout, and verification process of an 8-bit Wallace Tree Multiplier on gpdk45 technology in Cadence Virtuoso. The proposed architecture uses transmission gate (TG) full/half adder and carry-lookahead adder for final stage accumulation. Key layout strategies include cell abutment, well sharing, two-layer metal routing, and contacted poly gates. Through targeted critical-path buffering and voltage scaling, the design achieves a worst-case post layout propagation delay of $2.2\,\mathrm{ns}$ and energy per multiplication of $318\,\mathrm{fJ}$ and area of $1332.67\,\mu\mathrm{m}^2$, successfully meeting all specifications. Post layout simulation confirms DRC/LVS compliance and robust operation under worst case input vector.**

***Index Terms*—Wallace tree, transmission-gate adder, carry-lookahead, physical design, post-layout extraction, energy-latency-area.**

I. Introduction

Wallace-tree multipliers are an attractive option for high throughput arithmetic operations due to their logarithmic reduction depth for partial-product accumulation. This project implements an 8-bit by 8-bit Wallace multiplier in a gpdk 45 nm CMOS technology, using TG-based full/half adders for partial product reduction and a 16-bit CLA for the final adder. The design emphasizes manufacturable layout, parasitic-aware sizing, and post-layout verification with Cadence Quantus. The final post-layout worst-case propagation delay and energy meet the project specifications: $2.2\,\mathrm{ns}$ and $318\,\mathrm{fJ}$.

A. Design Specifications

- Input size: 8 bits
- Input rise/fall time: 50ps
- Output load: 2fF
- DRC/LVS Clean
- Report and justify worse case post layout propagation delay and energy consumption (using same input vectors as delay analysis)
- Pass functionality check
- Worst case propagation delay: ≤ 2.5 ns
- Energy consumption (considering the same input vectors as used in the delay analysis): ≤ 850 fJ
- Minimize Area

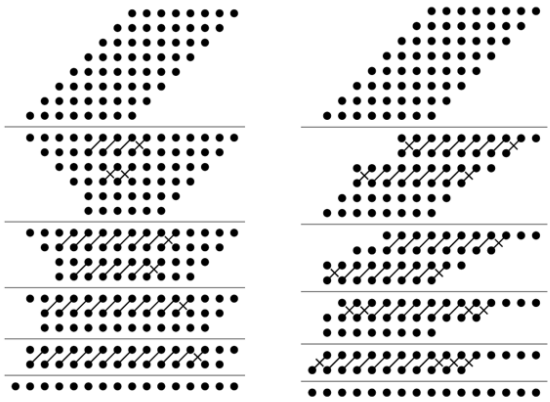


Fig. 1: Dot Diagram for an 8 by 8 Dadda tree (left) and Wallace tree (right) multiplier

B. Architecture Literature Review

Two well known fast multipliers are Wallace and Dadda. Both consist of three stages, partial product generation, reduction, and final addition.

In the Wallace tree method, the partial products are reduced as much as possible for each stage, this behavior can be interpreted as a combinational counting process, where a set of n input bits is reduced to $\lceil \log_2 n \rceil$ output bits representing their weighted sum.

From a PPA perspective, as shown in Fig.1 Dadda tree is generally considered superior. Dadda's method minimizes the total number of half adder and full adder by enforcing strict maximum column heights at each reduction stage, thereby reducing logic area, interconnect complexity, and switching activity [4].

The total delay of a parallel multiplier can be approximated as

$$T_{\text{mult}} \approx T_{\text{PP}} + N_{\text{levels}} \cdot T_{\text{FA}} + T_{\text{final}},$$

where T_{PP} is the partial product generation delay, N_{levels} is the number of reduction stages, T_{FA} is the delay of a full adder, and T_{final} is the delay of the final carry-propagating adder. While the Wallace tree may reduce N_{levels} slightly in some cases, the Wallace and Dadda trees have comparable reduction depths. This implies that performance advantage of Wallace tree is minimal. This tradeoff has been observed in both theoretical analysis and practical implementations [6]

In this project, we selected Wallace tree architecture under the assumption that Wallace and Dadda trees provide comparable delay characteristics, and that aggressive early reduction could simplify floor-planning by limiting long routing paths between partial product generation and reduction stages. From a physical design perspective, aggressive reduction of partial product in early stage enables a regular downward propagating layout that is amenable to pipelining. Although Dadda tree is generally superior in overall PPA performance, we retained the Wallace tree implementation due to time constraints and the advanced stage of the design.

$$T_{\text{tot}} \approx T_{\text{PP}} + N_{\text{levels}} \cdot T_{\text{CSA}} + T_{\text{CLA}} \tag{1}$$

where N_{levels} is smaller for a Wallace tree than for a Dadda tree, resulting in higher maximum operating frequency. However, this aggressive reduction requires more $(3, 2)$ counters, increasing logic area and interconnect complexity, which in turn raises switching activity and dynamic power. The Dadda tree, by contrast, minimizes area and power by postponing reductions until necessary, using the minimum number of adders but at the cost of additional reduction levels and longer delay. Consequently, while the Dadda tree is more area- and power-efficient, the Wallace tree is superior when performance dominates the design constraints, such as in high-speed arithmetic units and critical data paths.

II. Architecture

A. High-level block diagram

The Wallace Tree Multiplier consists of three main blocks:

1) Partial-product generation (AND array)
2) Wallace-tree reduction using TG half/full adders arranged in reduction layers to produce two rows of sum/carry for final addition
3) Final addition with a 10-bit carry-lookahead adder (CLA).

Figure 2 shows the overall architecture

B. Design rationale

TG-PT hybrid adders were selected for compact sum/carry implementation and layout abutment properties such as shared diffusion and contiguous cells. Carry-LookAhead (CLA) was chosen for the final stage to minimize worst case accumulation latency. This is an important metric for meeting the 2.5 ns objective in presence of parasitics.

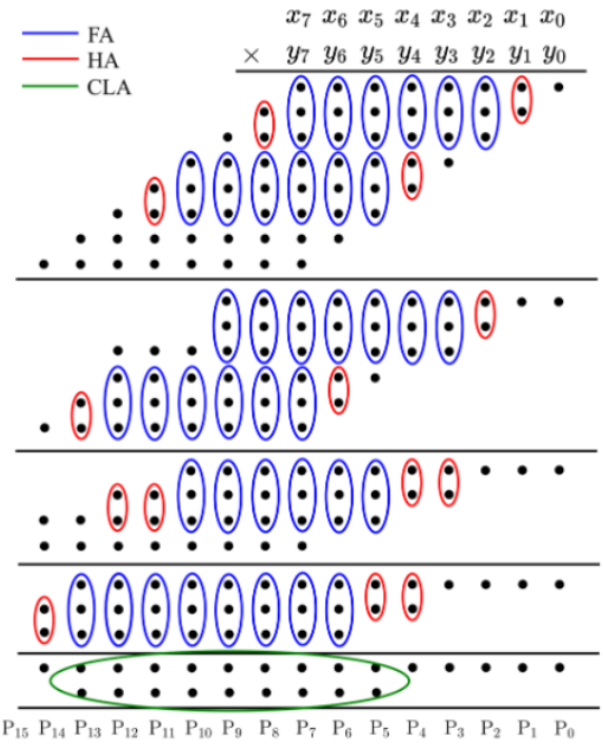


Fig. 2: Top-level architecture: partial product generation, Wallace reduction tree, and 10-bit CLA final adder.

III. Transistor-level Design

A. Partial Product Generator topology

The Partial Product Generator is implemented as an 8 by 8 AND matrix as shown in Fig 3. Each bit of the multiplicand is logically ANDed with each bit of the multiplier to generate the corresponding partial products. The layout of partial product is shown in Fig 4.

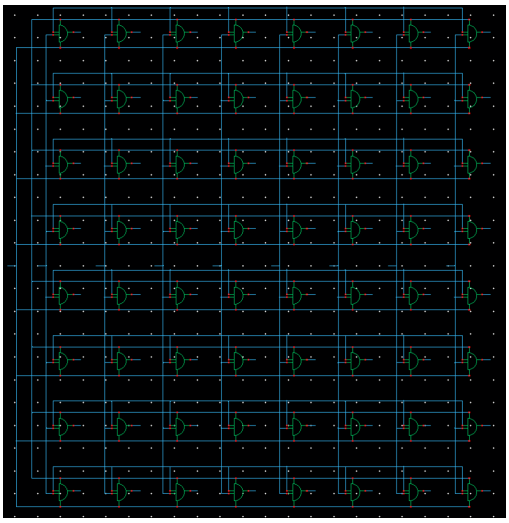


Fig. 3: Partial Product Generator Schematic

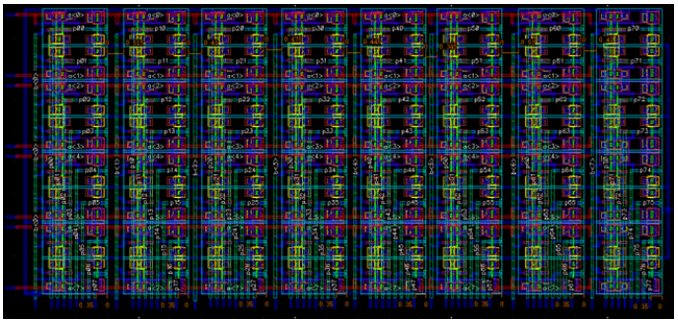


Fig. 4: Partial Product Generator Layout

B. TG full/half adder topology

The employed full adder uses TG and PT-based XOR/XNOR stages and a multiplexing structure for sum and carry. The uploaded schematic (Figure 5) shows transistor labels and TG elements. TGs provide rail-to-rail pass capability when sized appropriately, and their symmetric nature simplifies layout abutment. This provides simultaneous XOR-XNOR generation.

Parallel Generation for XOR-XNOR Conventionally, XOR signal will generate first, then XNOR will be generated after passing an inverter, this incurs sequential delay. Our module generates both signals in parallel [Fig. 5]. When input $A, B = 01$, the TG formed by n_2/p_2 is turned on to pass the strong Logic '1' from B directly to the XOR output. At the same time, n_5 is activated to to pass strong 0 from A to the XNOR output. By eliminating the dependency found in prior XNOR design, the critical path delay to *Sum* is minimized.

Sum and Carry Logic Utilizing previously generated full swing XOR/XNOR signals, the XOR and XNOR signals are transmitted to Sum output without any degradation. For example, when calculating ($C_{in} = 0, XOR = 1$), the circuit activates p_6 . Since PMOS device pass strong '1', the high XOR signal is transmitted directly to sum output. On the other hand, when the logic($XNOR = 1, C_{in} = 0$) is being transmitted, the circuit activates n_7 to pass strong '0'. This ensures that a low impedance path to either V_{DD} or GND is established, providing the drive strength to subsequent stages in the reduction phase.

As shown in TABLE I, we can observe at least one path provides full swing at output node for *sum*. Therefore, the Sum circuit guaranties rail to rail output for all input vectors, eliminating the threshold voltage V_{th} drop degradation which is common in Pass Transistor Logic (PTL).This offers an advantage over other techniques such as Gate Diffusion Input (GDI) which suffer from signal attenuation, modifying the GDI design requires additional logic which would incur additional power and cell area. This compact 4 transistor configuration results in a superior balance of signal integrity and area.

Half Adder is designed in a similar manner where the same TG-PT hybrid XOR/XNOR generation is employed to produce the *Sum* output, while the *Carry* output is generated using a simplified pass-transistor-based logic path.

TABLE I: Truth Table for Proposed Hybrid TG-PT Full Adder

Inputs			***Sum Output***		***Cout Output***	
C_{in}	A	B	**Full Swing Path**	**Logic**	**Full Swing Path**	**Logic**
0	0	0	n_7 (TG)	$C_{in} = 0$	n_8 (PT)	$B = 0$
	0	1	p_6 (TG)	$XOR = 1$	n_9 (PT)	$C_{in} = 0$
	1	0	p_6 (TG)	$XOR = 1$	n_9 (PT)	$C_{in} = 0$
	1	1	n_7 (TG)	$C_{in} = 0$	p_9 (PT)	$A = 1$
1	0	0	p_7 (TG)	$C_{in} = 1$	n_8 (PT)	$B = 0$
	0	1	n_6 (TG)	$XNOR = 0$	p_8 (PT)	$C_{in} = 1$
	1	0	n_6 (TG)	$XNOR = 0$	p_8 (PT)	$C_{in} = 1$
	1	1	p_7 (TG)	$C_{in} = 1$	p_9 (PT)	$A = 1$

Fig. 5: TG full-adder schematic used

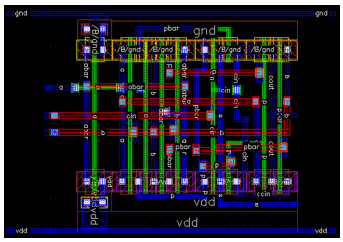


Fig. 6: Full Adder Layout

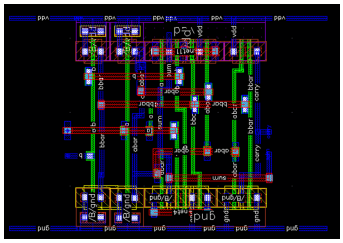


Fig. 7: Half Adder Layout

C. Driver topology

The driver stage is for mitigating long RC time-constant delay paths. As the number of cascaded adders increases, the effective interconnect resistance and capacitive loading also increase, resulting in higher propagation delay. To address this, a driver stage is inserted to break the long RC path and restore signal strength. It consists of two cascaded inverters which form a buffer as shown in Fig. 8. As a result, the overall propagation delay of the circuit is reduced.

D. Final Adder topology

The final adder stage is for adding the two reduced operands produced by the Wallace reduction network. The remaining operands are 10 bits which correspond to bit position [14:5]. This stage typically has a longer delay since it requires signal to propagate from LSB to MSB. To avoid such delay the final adder is implemented using a Carry Look-Ahead (CLA) topology. It consists of one half-adder to produce carry out

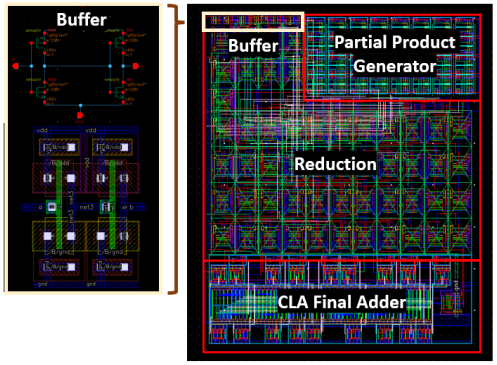


Fig. 8: Buffer Schematic and Layout

and two 4-bit CLA. This structure provides balance between propagation delay, area, and design complexity.

The CLA architecture allows parallel computation for carrying signals. As shown in equation (2)–(6), the carry outputs of each stage can be computed using generate (G) and propagate (P) terms. Figure (9) shows a carry generator that computes C_4 with G and P. Since propagation delay increases with fan-in, the carry generator is limited to 4 bits rather than 5 bits. The limitation to logic depth prevents excessive increases in both propagation delay and area.

$$C_0 = \text{input carry} \tag{2}$$
$$C_1 = G_1 + P_1 C_0 \tag{3}$$
$$C_2 = G_2 + P_2 G_1 + P_2 P_1 C_0 \tag{4}$$
$$C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_0 \tag{5}$$
$$C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_0 \tag{6}$$

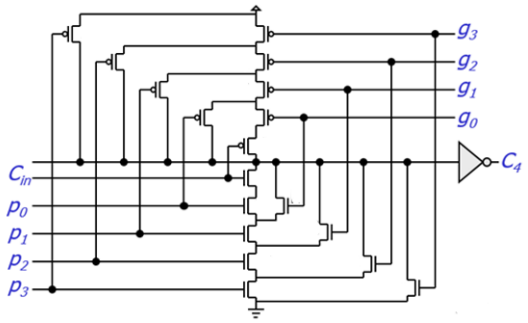


Fig. 9: Carry generation logic of the CLA.

The final sum bits are computed using an XOR operation between the carry output and the corresponding propagate signal. For each bit position n, the sum is determined by the logical expression shown in equation (7).

$$\text{sum}_n = C_{\text{out},n} \oplus P_{n-1} \tag{7}$$

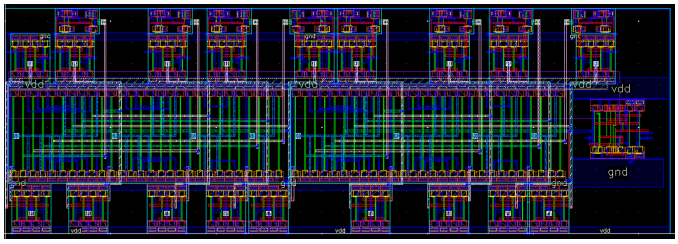


Fig. 10: Final Adder Layout

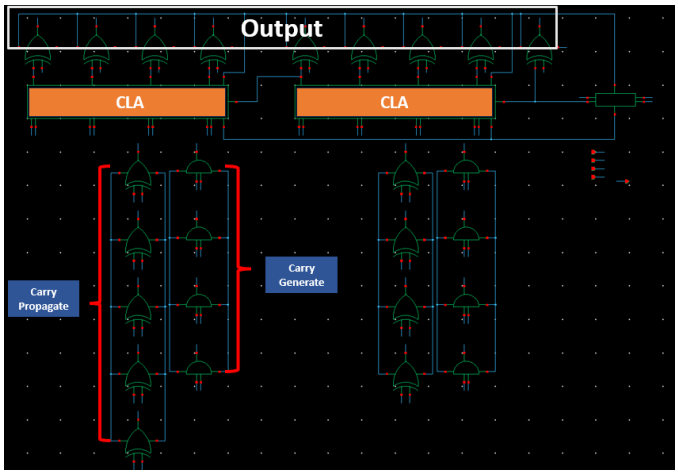


Fig. 11: Final Adder Schematic

E. Sizing strategy

Sizing was chosen to balance delay on critical paths and area/power:

- Non-critical logic transistors: minimum width (W_{min}).
- XOR driving transistors in TG stages: 1.5–2 $\times$ W_{min}.
- Carry-generation and multiplexing devices: up to 3$\times W_{min}$ for those that lies on the critical path.

Generally, we follow the intuition of logical effort. **Increase size for high logical effort stage** We need to create a balanced circuit such that logical effort per stage contributes roughly equally to path delay. See Sec. IV for the delay model used.

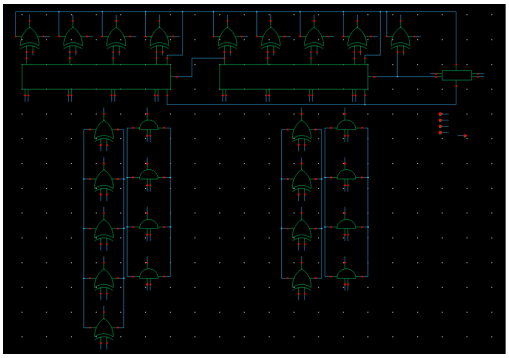


Fig. 12: Final Adder Schematic

IV. Delay and Energy Models

We used standard RC and logical-effort models for pre-layout sizing and margining, then refined with parasitic extraction.

A. Delay model

For a gate stage:

$$\tau = p + gh \tag{8}$$

where p is parasitic delay, g is logical effort, and $h = C_{out}/C_{in}$ is the electrical effort. Total path delay expressed as:

$$t_p = \tau_{inv} \sum_i (p_i + g_i h_i) \tag{9}$$

τ_{inv} is the inverter delay normalization constant for the 45 nm process. For carry dominated stages (CLA generate/propagate networks), we prioritized lowering h by upsizing drivers.

B. Dynamic energy

Dynamic energy per operation can be determined as follows:

$$E_{dyn} = C_i V_{dd}^2 \tag{10}$$

where C_i is load capacitance. For analysis, we assumed that α from worst-case vectors supplied and validated with post layout simulation.

We can also arrange the energy as the integral of the instantaneous power $(P(t))$ over the time period $[t_1, t_2]$:

$$E = \int_{t_1}^{t_2} P(t)\,dt = \int_{t_1}^{t_2} V_{DD}(t) \cdot I_{DD}(t)\,dt \tag{11}$$

Since V_{DD} is usually a constant DC supply voltage (V_{DD}), the equation simplifies to:

$$E = V_{DD} \cdot \int_{t_1}^{t_2} I_{DD}(t)\,dt \tag{12}$$

This is used in Cadence ADE calculator for calculation of energy consumption. The measured post layout energy is 318 fJ.

V. Physical Design

A. Floorplanning and cell placement

Our physical design follows a structured, data driven approach to minimize path length and congestion. The input pin are located on the top right and the output pin is located to the bottom to reduce the likelihood of antenna violations and cross talk. This unidirectional signal flow also significantly reduce the routing complexity.

To minimize routing congestion and parasitic, we adopted the following strategy:

- Vertically and horizontally align FA and HA with cell abutment to share diffusion and reduce parasitic capacitance.
- Well sharing across rows to reduce well contacts and area.
- Vdd and Gnd rail sharing
- Contacted poly gates used where density required.

From Figure 13, we can see that to maximize area efficiency, the design utilized standard cell row topology with alternating row orientation. This allows the column to share vdd and gnd rails by abutment.

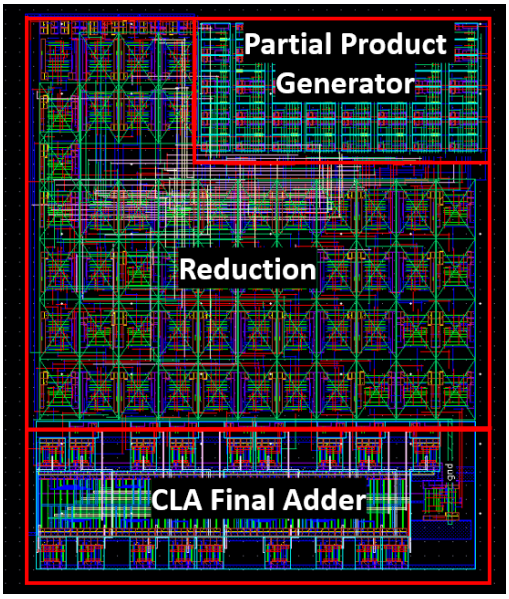


Fig. 13: Wallace Tree Multiplyer Layout

B. Buffering and critical-path optimization

Buffer insertion points were added strategically in the reduction stage. Buffer sizing was tuned to minimize slew and reduce downstream load, improving worst-case delay while controlling energy.

VI. Verification and Results

A. Functional verification and worst-case vectors

Functional correctness was validated for the full input space using mixed-level simulation for partial verification and exhaustive test vectors for corner cases. For worst-case timing and energy, the supplied sequence of vectors was used to capture representative carry propagation and switching patterns in the Wallace reduction and final CLA addition.

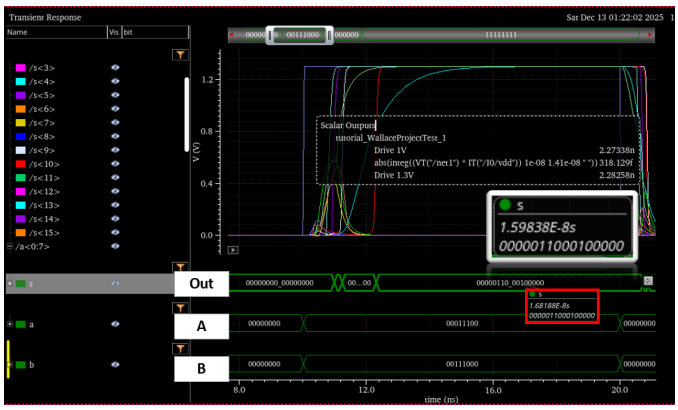


Fig. 14: Worst Case Delay Input (Case 1)

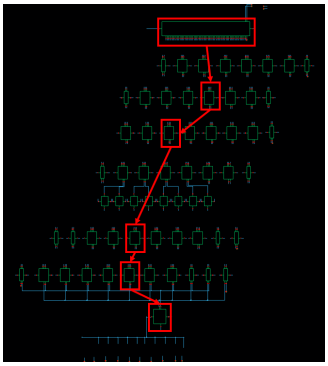
The functionality of the Wallace Tree Multiplier is verified by using various test cases. The results are summarized in the table, where the expected outputs are compared with the simulation outputs to confirm correctness. The first test case

represents the worst case carry propagation scenario. The full adder to generate bit 10 is physically placed at the farthest location in the layout. Therefore, it produces the maximum propagation delay. The second test case applies all 1 input. This pattern forces the multiplier to switch at each position of the bit. It represents the multiplier operating with high-switching activity. The third test case applies all 0 input. This serves as a baseline to check that no bits are generated or propagated. This test verifies that the multiplier function as expected under idle conditions.

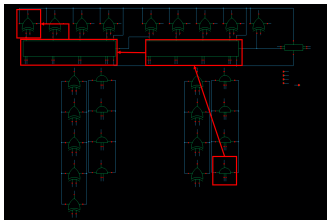
TABLE II: Functional Verification and Simulation Results

Case	Input A	Input B	Expected Output	Actual Output
1	0001 1100	0011 1000	0000 0110 0010 0000	0000 0110 0010 0000
2	1111 1111	1111 1111	1111 1110 0000 0001	1111 1110 0000 0001
3	0000 0000	0000 0000	0000 0000 0000 0000	0000 0000 0000 0000

The worst-case vector can be identified either through schematic or layout. From the schematic, the critical path is shown in Fig. 15: AND → four Full Adders → AND → two 4-bit CLAs → XOR. Post-simulation results indicate that the propagation delay along this schematic-level critical path is 1.18 ns. Fig. 16 shows that the Nanotime analysis identifies the same critical path. From the layout, physical placement and interconnect play an important role. In particular, the full adder responsible for generating bit 10 is located furthest from the rest of the circuit, resulting in the longest interconnect length. Thus, bit 10 exhibits a higher propagation delay of 2.2 ns in post-layout simulation. Comparing the delays obtained from the two analyses, the layout analysis delay produces the maximum delay. It is selected as the worst case scenario as it more accurately reflects the true performance of the design.



(a) Top-level critical path



(b) Final Adder critical path

Fig. 15: Critical path analysis of the Wallace Tree Multiplier.

B. Post-layout timing and energy

All post-layout simulations used PVS-PEX extracted parasitics. Cadence PVS was used for DRC/LVS; results were clean (no netlist mismatches or rule violations). Small increments of

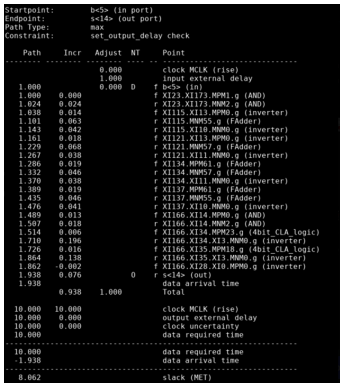


Fig. 16: Nanotime for Wallace Tree Multiplier Path Delay

the supply voltage are applied iteratively to achieve optimal propagation delay. Post-layout worst-case metrics:

- Worst-case propagation delay: 2.2 ns.
- Energy per multiply (post-layout measurement): 318 fJ.

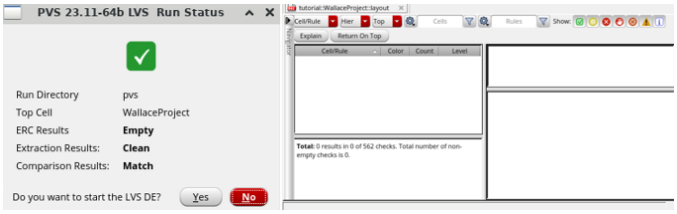


Fig. 17: LVS/DRC Report of Wallace Tree Multiplier

These results meet the stated specifications.

TABLE III: Delay and Energy Summary

Parameter	Schematic	Post-Layout
Energy (fJ)	120.8	318.13
Propagation Delay (ns)	1.18	2.273
Multiplier Area	$33.56\,\mu\text{m} \times 39.71\,\mu\text{m}$ $= 1332.67\,\mu\text{m}^2$	

C. Design-space explorations

We considered three operating points:

- **Area-minimal**: Minimum sizing, no buffering — increases delay beyond spec.
- **Balanced** (selected): Selective upsizing and buffering on reduction path — meets delay and energy targets.
- **Performance-oriented**: Aggressive upsizing and aggressive buffering: lower delay, higher energy and area.

Prior to the optimization phase, a performance analysis was performed using cadence virtuoso on first draft of Wallace tree multiplier layout to evaluate the initial design's PPA metrics, we are able to quantify the impact of preliminary design choices and identify the trade offs within the architecture. As seen in Fig. 18 and Fig. 19, we can observe that the buffered layout extends the pareto frontier of the design. However,

TABLE IV: Component Count and MOS Cell Summary for 8x8 Wallace Tree Multiplier

Component	AND	OR	XOR	XNOR	HA	FA	CLA Blk	Total Gates	PMOS	NMOS	Total MOS
Partial Product Generator	64	0	0	0	0	0	0	**64**	192	192	**384**
Reduction Network	0	0	0	0	15	38	0	**53**	462	462	**924**
Final CLA Adder	8	0	18	0	1	0	2	**29**	160	160	**320**
TOTAL	**72**	**0**	**18**	**0**	**16**	**38**	**2**	**146**	**814**	**814**	**1628**

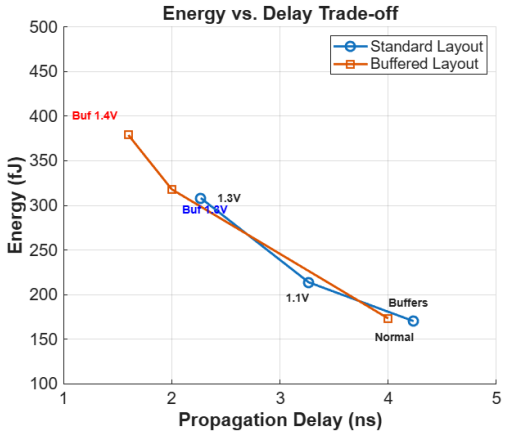


Fig. 18: Energy vs Delay Tradeoff (matlab)

it significantly increase the energy consumption. The steep slope between Buf 1.3V and Buf 1.4V displays a diminishing returns, where marginal gain in speed incurs exponential increases in energy.

We have concluded that buffered layout is necessary choice despite higher dynamic power consumption.

This pre-optimization data served as a critical benchmark, establishing a reference point to measure the effectiveness of subsequent improvements.

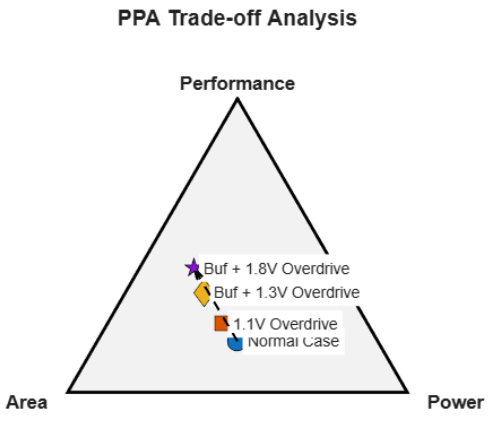


Fig. 19: PPA Analysis

VII. Bonus: Multiply Accumulator Unit (MAC)

MAC primarily compose of two components:

1) Multiplication unit

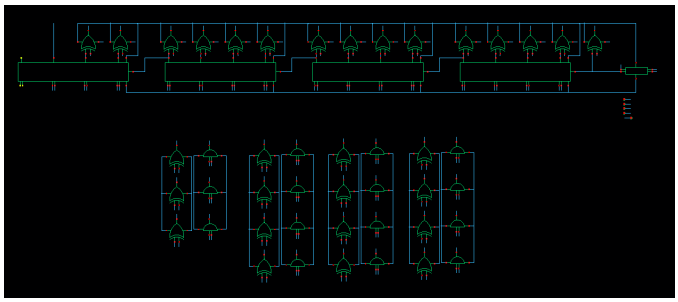


Fig. 20: MAC Schematic

2) Accumulation unit

MAC unit performs both multiply and addition functions. It operates in two stages. Firstly it computes the product of given numbers and forward the result for the second stage operation i.e. addition/accumulate. If both the computing is executed in a single rounding then it is said to be fused multiply-add/accumulate (MAC) unit. [8]

For demonstration purpose, we will build a simple fused multiply add unit. A 16 bit CLA adder will be integrated in addition to final CLA adder. Thus performing the operation

$$s = a \times b + c \tag{13}$$

The 16-bit adder topology composed of one half adder followed by four 4-bit Carry-LookAhead (CLA) adders. The design is evaluated using the same worst-case input patterns as those applied in the configuration without the MAC. Post-layout simulation results indicate a propagation delay of $2.76\,\mathrm{ns}$ and an energy consumption of $780.49\,\mathrm{fJ}$.

TABLE V: MAC Unit Post-Layout Performance Summary

Parameter	Post-Layout Value
Energy	780.49 fJ
Propagation Delay	2.76 ns
Area	$33.5\,\mu\mathrm{m} \times 60.9\,\mu\mathrm{m}$ $= 2040.15\,\mu\mathrm{m}^2$

Work will be done in future to integrate an output feedback path to support accumulation and complete the MAC operation.

$$c_{i+1} = a \times b + c_i \tag{14}$$

VIII. Conclusions

We presented the complete physical design and post-layout verification of an 8-bit Wallace-tree multiplier implemented in

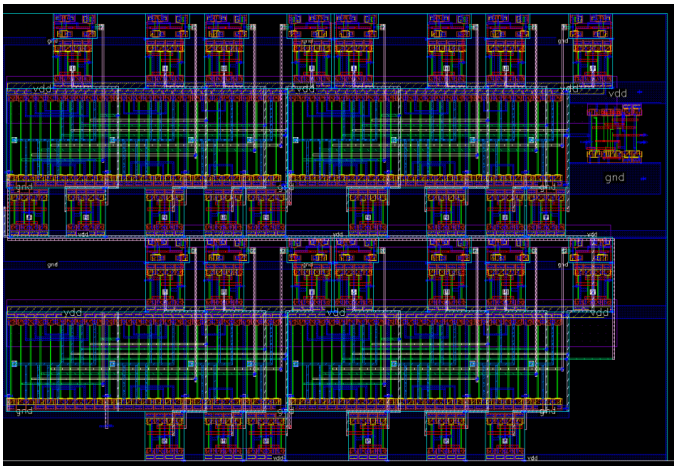


Fig. 21: MAC Layout

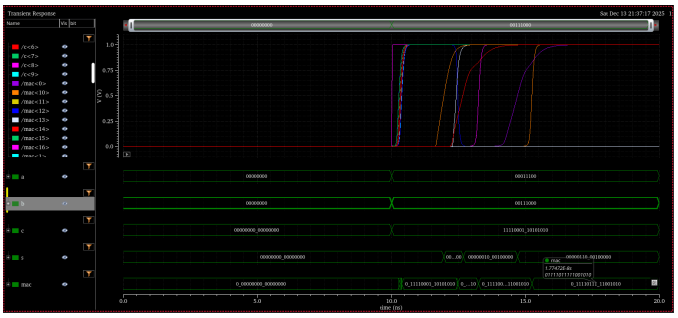


Fig. 22: MAC Simulated Output

TABLE VI: Functional Verification and Simulation Results for MAC Unit

Input A	Input B	Input C (Accumulator)	Output S (Product)	Output MAC (S + C)
0001 1100	0011 1000	1111 0001 1010 1010	0000 0110 0010 0000	0 1111 0111 1100 1010

a 45 nm CMOS process, using TG full and half adders and a 10-bit carry look-ahead final adder. The observed increase in energy from schematic to post-layout simulation is primarily attributed to extracted interconnect capacitance and buffering overhead. Through iterative physical optimization, including cell abutment, well sharing, targeted buffering in the reduction stages, and careful transistor sizing, combined with post-layout tuning, the design achieves a worst-case propagation delay of $2.2\,\mathrm{ns}$ and an energy consumption of $318\,\mathrm{fJ}$ while remaining fully DRC/LVS clean.

IX. Future Work

While the proposed design meets the target performance and energy specifications, several opportunities remain for further PPA optimization. Selective use of multi-threshold voltage devices (VT Swap) could reduce delay on critical carry paths while minimizing leakage power in non-critical logic. Additional interconnect aware placement and clustering of reduction-stage adders may further reduce parasitic RC delay without requiring aggressive transistor upsizing.

For energy optimization, operand isolation and clock gating represent effective extensions, especially if the design is expanded to support multiply accumulate (MAC) operation in the future. Finally, incremental supply voltage scaling based on detailed post layout timing analysis could provide additional energy savings while preserving functional correctness and performance.

Contributions

Ben Li: TG/PL adder schematic and layout, partial product generator schematic and layout, sizing strategy, final layout PnR, post layout extraction and verification, MAC schematic and layout, STA, report drafting.

Andrew Lien: final adder schematic and layout, MAC schematic and layout, final layout PnR, post layout extraction and verification, report editing.

Acknowledgments

We acknowledge the use of the Cadence Virtuoso for layout and verification and the use of Professor Gupta material for the paper.

References

[1] H. Bakoglu, *Circuits, Interconnects and Packaging for VLSI*, 2008.
[2] J. Rabaey, *Digital Integrated Circuits: A Design Perspective*, 2nd Ed.
[3] M. Horowitz, "Low-power digital design," *IEEE Symposium on Low Power Electronics*, 1995.
[4] C. S. Wallace, *A Suggestion for a Fast Multiplier*, IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp. 14–17, 1964.
[5] L. Dadda, *Some Schemes for Parallel Multipliers*, Alta Frequenza, vol. 34, pp. 349–356, 1965.
[6] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford University Press, 2010.
[7] J. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*, 2nd Ed., Prentice Hall, 2003.
[8] P. Nain and G. S. Virdi, *Multiplier-Accumulator (MAC) Unit*, International Journal of Digital Application & Contemporary Research, 2016.