8-bit Wallace Tree Multiplier

Shubham Saluja Kumar Agarwal
Elmore School of Electrical and Computer Engineering, *Purdue University*, West Lafayette, IN
skumarag@purdue.edu

***Abstract*—A high-speed 8×8 Wallace Tree multiplier was designed using 45nm CMOS technology. The architecture reduces partial product rows using half and full adders across four summation stages, ending with a ripple-carry adder. The complete design uses 2474 transistors and achieves a maximum delay of around 680ps and an average power dissipation of 186.2μW, as verified through Cadence simulations. These results demonstrate that the proposed design offers superior speed and competitive power consumption compared to other multiplier architectures, making it suitable for high-performance digital systems.**

***Index Terms*—Wallace multiplier, 45nm CMOS, adders**

I. Introduction

Various multiplier architectures have been developed to optimize multiplication parameters, including Array, Booth, Dadda, and Wallace Tree multipliers.

In this project, we focus on the design and implementation of an 8×8 Wallace Tree multiplier using 45 nm CMOS technology. To contextualize our design choice, we compare the performance metrics of several 8×8 multiplier architectures:

Multiplier	Delay	Power
Wallace Tree[1]	0.80ns	7.84mW
Booth[2]	1.068ns	
Array[2]	1.553ns	
Dadda[3]	1.3916ns	13.33μW

A. Proposed Technique

The Wallace Tree Adder was selected for two main reasons. First, for its ability to reduce the number of sequential addition stages, thereby decreasing propagation delay. The second reason is that the layer based breakdown of the calculation makes it easy to compartmentalize and make the design.

II. Methodology

All the possible partial products are calculated using AND gates, and arranged as shown in the second layer the figure below.

The Wallace Multiplier has two main building blocks:

- 3:2 compressor $\rightarrow$ Full Adder
- 2:2 "compressor" $\rightarrow$ Half Adder
- 1:1 "compressor" $\rightarrow$ Buffer

Depending on how many elements there are in the column we use the compressors or send the bit directly to the next layer using a buffer. This can be seen in Fig. 1., where all elements in a column of the same color are sent into the same building block. Out of the two outputs of the compressors, the sum goes to the same column while the carry goes to the next one. Once each column has at most two bits, they are all sent into a ripple-carry adder to produce the procuct.

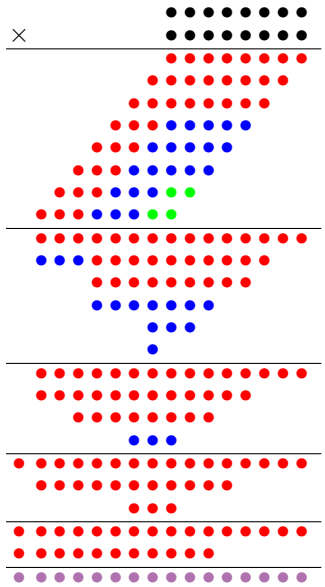


Fig. 1. Multiplication Process

III. Circuit Design

We created a hierarchical design for this circuit.

A. Gate Level

This layer contains all the gates used to make the rest of the circuit:

- Inverter $\rightarrow$ 2 transistors
- AND $\rightarrow$ 6 transistors
- OR $\rightarrow$ 6 transistors
- XOR $\rightarrow$ 12 transistors

All of these use minimum sized transistors.

B. Building Block Level

This layer contains the blocks mentioned in Methodology

Component	Equations	Transistors
Buffer	$OUT = !(!IN)$	4
Half Adder	$S = A \oplus B$, $C = AB$	18
Full Adder	$S = A \oplus B \oplus C_{in}$ $C_{out} = C_{in}(A \oplus B) + AB$	42

These layers were all designed using the Gate Level designs.

C. Module Level

This layer contains the modules where the bulk of the calculation, shown in Fig. 1. is done;

- Partial Product: AND gates to calculate partial products.
- Sum Layers: half and full adders to reduce the bits.
- Ripple Carry Adder: all remaining pairs of bits are added using full adders, or a half adder for the first pair, in series with each other.

Their schematics can be found in the schematic section down below.

D. Top Level

This is the complete multiplier, that is, all the previous modules combined to create the fully functioning multiplier. It consists of the Partial Product Module followed by each of the four SUM layers, and ends with the Ripple Carry Adder. It has a total of **2474 transistors**.

$$\text{AB}\rightarrow \boxed{PP}\rightarrow\boxed{S1}\rightarrow\boxed{S2}\rightarrow\boxed{S3}\rightarrow\boxed{S4}\rightarrow\boxed{RCA} \rightarrow \text{M}$$

IV. Theoretical Results

This analysis is based on the following assumptions:

- **NMOS resistance**: $R_{\text{NMOS}} = 3\,\text{k}\Omega$
- **PMOS resistance**: $R_{\text{PMOS}} = 6\,\text{k}\Omega$
- **Unit capacitance**: $C_{\text{unit}} \approx 2\,\text{fF}$
- **Transistor sizing**: All transistors are minimum size
- **Activity factor**: $\alpha = 0.25$
- **Maximum frequency before error**: $f = 1\,\text{GHz}$
- **Supply voltage**: $V_{\text{DD}} = 1.2\,\text{V}$

A. Delay

Estimates for the delay are made using $t = 0.69RC$.
The total delay will be the sum of the delay of each of the layers.

- Partial Products $\rightarrow t_{AND}$
- Sum Layers $\rightarrow 4 \times t_{FA,sum}$
- Ripple Carry Adder $\rightarrow 12 \times t_{FA,carry} + t_{FA,sum}$

$$t_{AND} = t_{NAND} + t_{INV} \approx 8.28ps + 4.14ps \approx 12.5ps$$
$$t_{FA,sum} = 2 \times t_{XOR} \approx 2 \times (t_{INV} + 16.66ps) \approx 40ps$$
$$t_{FA,carry} = t_{AND} + t_{OR} \approx 12.5ps + 18.75ps \approx 31.25ps$$

Which makes the total delay approximately:

$$t_M \approx 5 \times 40ps + 12 \times 31.25ps + 12.5ps = 587.5ps$$

B. Power

There were 40 full adders in the Sum Layers, 17 Half Adders, 26 buffers, and finally, 64 AND gates from the Partial Product Generator.

By considering the fan-ins and fan-outs of the components, we can approximate $C_{FA} = 3C_{unit}$, $C_{HA} = 2C_{unit}$ and $C_{AND} = C_{buffer} = C_{unit}$. The total capacitance, C_{load}, comes to:

$$C_{load} = (40 * 3C_{unit}) + (17 * 2C_{unit}) + (26 * C_{unit}) + (64 * C_{unit}) = 244C_{unit} = 488fF$$

Since $P = \alpha \times C_{load} \times V_{dd}^2 \times f$,

$$P_{avg} \approx 0.25 \times 488fF \times 1.2V^2 \times 1GHz = 175.68\mu W$$

V. Simulation Results

A. Functionality

The following is the list of tested values, alongside their expected results:

A	B	M	A	B	M
CB	11	0D7B	EE	33	2F6A
11	00	0000	1A	A3	108E
C4	6B	51EC	1F	F1	1D2F
55	AC	391C	C3	E2	AC26
32	28	07D0	FF	FF	FE01

The resulting plot can be seen in the Plots section below, with all the resulting multiplications being exactly what we expected them to be.

B. Delay

To estimate the critical delay, we visually selected the calculation which took the longest time to stabilize.

Once this had been located, the timestamp of the midpoint of the input wave was logged, $15.0565ns$, and the timestamp of the midpoint of the last changing output wave was logged, $15.7366ns$. The maximum delay was thus defined as:

$$t_M = 15.7366ns - 15.0565ns = 0.6801ns \approx 680ps$$

This value is reasonably close to the estimated value of $587ps$.

C. Power

To calculate the power, we used:

$$P = avg(IV) = \text{average(VT("/VDD")*IT("/V0/PLUS"))} = -186.2\mu W$$

The real value of the consumed power is reasonably close to the predicted theoretical value of the power.
The power plot can also be seen in the Plots section below.

VI. Conclusion

An 8×8 Wallace Tree multiplier was designed and verified in 45nm CMOS. Simulation and theoretical results showed close agreement, with a delay of 680ps and average power of $186.2\mu W$. The architecture outperformed alternatives in speed while maintaining low power, validating its suitability for high-performance applications.

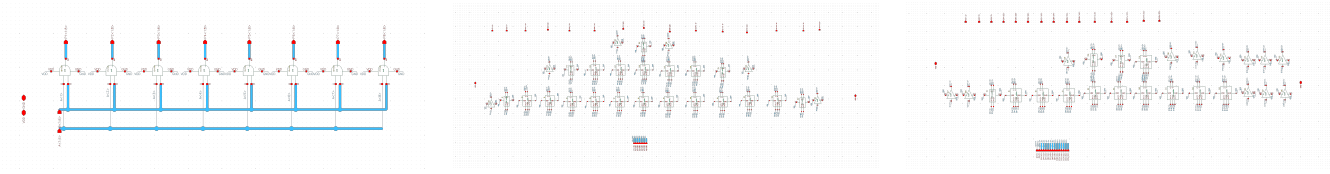
VII. References

[1] Pradeep Kumar Kumawat and Gajendra Sujediya, "Design and Comparison of 8x8 Wallace Tree Multiplier using CMOS and GDI Technology," IOSR Journal of VLSI and Signal Processing, vol. 7, no. 4, pp. 57-62, 2017.

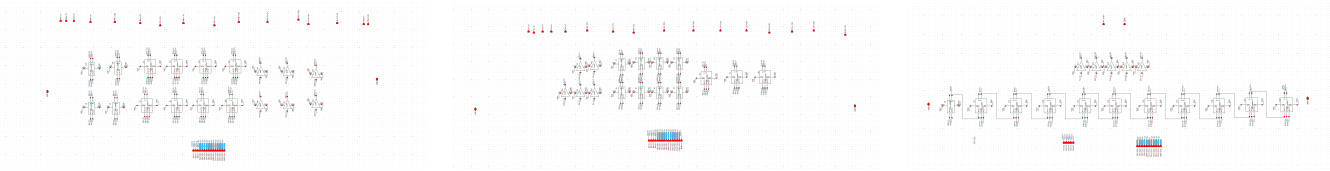
[2] Kalaiyarasi D. and M. Saraswathi, "Design of an Efficient High Speed Radix-4 Booth Multiplier for both Signed and Unsigned Numbers," 2018.

[3] P. Gupta, A. Gupta, and A. R. Asati, "Ultra Low Power MUX Based Compressors for Wallace and Dadda Multipliers in Sub-threshold Regime," American Journal of Engineering and Applied Sciences, vol. 8, no. 4, pp. 702-716, 2015.

VIII. Schematics



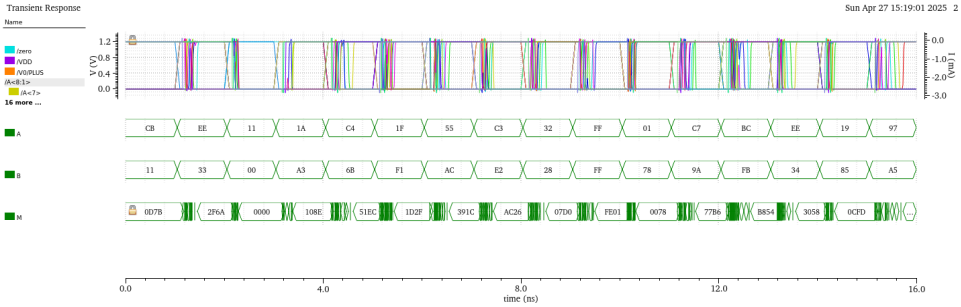
(a) Partial Products (b) Sum Layer 1 (c) Sum layer 2



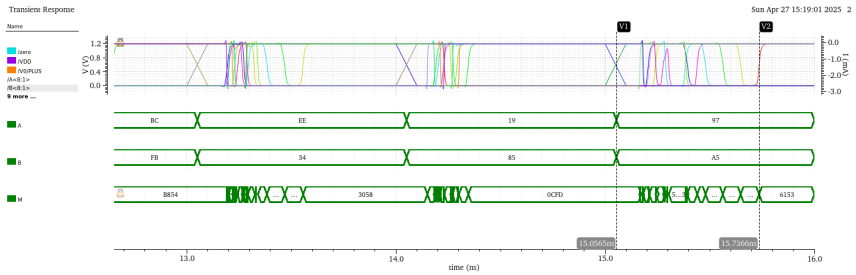
(d) Sum Layer 3 (e) Sum Layer 4 (f) Ripple Carry Adder

Fig. 2. Module Level Schematics

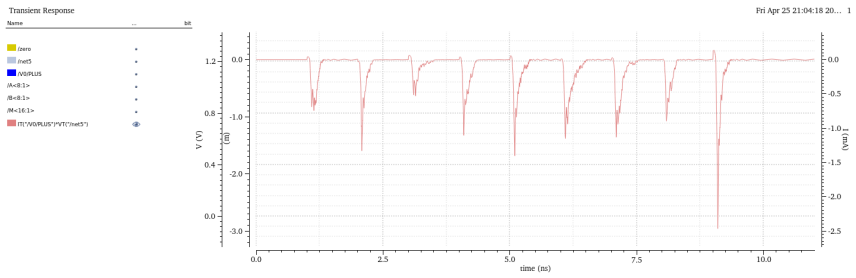
IX. Plots



(a) Functionality Verification



(b) Delay Calculation



(c) Power vs. Time

Fig. 3. Relevant Simulation Plots for functionality, propagation delay and power.