Proceedings of the 5th International Conference on Materials Chemistry and Environmental Engineering
DOI: 10.54254/2755-2721/127/2025.20275

Systematic Analysis on Performance Optimization of Tree Multipliers

Haoran Yu[1,a,*]

[1]*School of Electronic Information Engineering, Anhui University, Hefei, China*
a. P12214186@stu.adu.edu.cn
**corresponding author*

Abstract: The microprocessor chip is the core technology of the computer industry, and the multiplier is an important computing unit of the microprocessor. Its computing speed and area performances determine the performance of the microprocessor, and its structure determines whether it is easy to implement. Tree multipliers are widely used due to their superior performance. This article analyzes and compares the performance of several typical tree structures, including Wallace tree, Dadda tree, ZM tree, and OS tree, and compares various performance optimization methods, such as optimizing compressor structure, optimizing tree structure, and using PPA. This article found that the Wallace tree and Dadda tree have the fastest speed, but their irregular structures are not conducive to implementation. ZM tree and OS tree are two types of delay balancing trees with regular structures, but sacrifice speed and area performances. The optimization of compressor structure and tree structure, as well as the use of PPA, have reduced latency, but may lead to an increase in area. This article summarizes the current development of tree multipliers and has high reference value in future research.

Keywords: Microprocessor, Tree multiplier, Performance optimization.

1. Introduction

The design of microprocessor chips is one of the core technologies in the entire computer industry, and the performance of microprocessors is constantly improving at an astonishing speed, which is driving the rapid development of the entire information industry. The multiplier is an important computing unit in microprocessors, serving as the core of real-time image processing and digital signal processing. Its computing speed determines the speed of the microprocessor. Its area affects the size of the microprocessor. The characteristics of its structure determine whether it is easy to implement in VLSI. That's why optimizing the speed, area, and structure of the multiplier is crucial for the overall performance of the microprocessor. Wallace first proposed and designed the tree multiplier structure in 1964 [1]. Several scholars proposed other typical tree multiplier structures, which have different advantages in speed and area. Many scholars have optimized these structures from multiple perspectives and achieved excellent performance.

This paper begins with an introduction to the fundamental principles and classifications of tree multipliers, including common structures such as Wallace tree and Dadda tree. Subsequently, this review delves into the diverse performance optimization strategies posited within extant literature, including the design of novel compressors, the refinement of tree structures, and the implementation

DOI: 10.54254/2755-2721/127/2025.20275

of more advanced adders for the final summation. The paper concludes by summarizing the current development in the field, providing a valuable reference for subsequent studies.

2. Analysis of typical tree structures

2.1. Wallace tree

Wallace noted that while it is possible to add two or more numbers within a single adder, the growth in logical complexity is not proportional to the increase in speed, making it an impractical solution. A novel approach involves utilizing a pseudo adder, which can add three numbers together but yields two outputs instead of one. This reduces the number of remaining digits to be added, hence the term '3-2 Compressor' or 'Carry Save Adder (CSA)'. Since it can operate along its digital stages without carry propagation, it operates faster than traditional adders.

Furthermore, Wallace highlighted that employing multiple pseudo adders for parallel operation can decrease the number of digits by 1.5 times. By achieving a significant enhancement in computational speed while avoiding an increase in hardware costs, it also simplifies the control circuitry. Wallace employed this methodology in the design of tree-structured multipliers.

The Wallace tree multiplier comprises three phases. The initial phase involves the formation of a partial product matrix. The second phase entails the reduction of the height of this partial product matrix to two through parallel operations. The final phase employs a carry-propagate adder to combine the two rows. Figure 1 below shows the addition tree as designed in the Wallace tree architecture. Figure 2 shows dot diagram for an 8 by 8 Dadda multiplier.

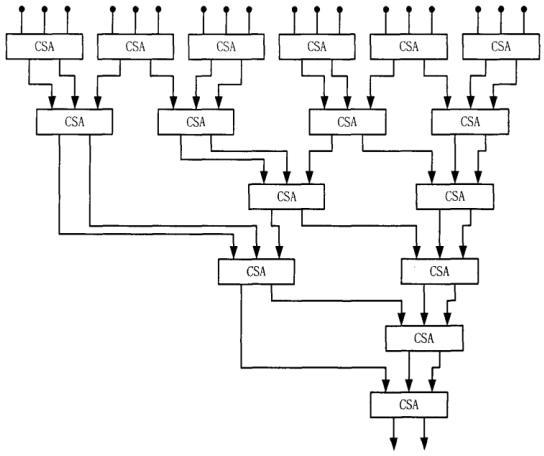

Figure 1: Wallace tree structure using CSA capable of summing up 18 partial products.

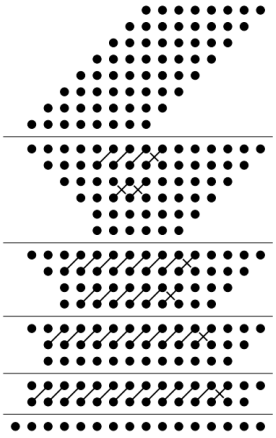

Figure 2: Dot diagram for an 8 by 8 Dadda multiplier.

2.2. Dadda tree

Dadda introduced an alternative classical structure for parallel multipliers [2]. In the design, Dadda replaces Wallace's pseudo-adder with 'parallel (m, n) counters'. These counters are characterized by having 'm' inputs and 'n' outputs, where 'm' is less than or equal to 'n'. The binary value represented by the 'm' outputs corresponds to the count of '1's present among the 'n' inputs. The Dadda tree multiplier, akin to the Wallace tree multiplier, operates through three distinct stages; however, Dadda's approach employs the minimal number of (3, 2) and (2, 2) counters at each level during the partial product reduction phase. Dadda invented the dot diagram, and the figure 2 above shows the 8 by 8 multiplier designed by Dadda. This multiplier requires four reduction levels, with matrix heights of 6, 4, 3, and 2, respectively The points connected diagonally in the figure represent

DOI: 10.54254/2755-2721/127/2025.20275

the output of the (3,2) counter. The two points connected by cross diagonals represent the output of the (2,2) counter. A total of 64 AND gates, 35 (3,2) counters, 7 (2,2) counters, and a 14 bit carry propagation adder are required to form a 16-bit product.

Dadda further hypothesized that counters of varying dimensions could be feasibly implemented. The 4-2 compressor has excellent characteristics, including outstanding balance and symmetry. Related studies have shown that the multiplier composed of 4-2 compressors has good performance [3]. It has been widely used as a fundamental element in tree multiplier architectures.

Habibi and Wintz believe that Dadda's method of placing counters is optimal [4]. Townsend's research shown in table 1 also indicates that under the exclusive use of (3,2) and (2,2) counters, Dadda tree multipliers exhibit reduced delay, suggesting a superior speed, along with a lower complexity compared to Wallace tree multipliers [5].

Table 1: The comparison of delay and complexity between Dadda tree and Wallace tree.

Multiplier Size	Dadda Delay	Wallace Delay	Dadda Complexity	Wallace Complexity
4 by 4	19	21	104	104
8 by 8	37	42	528	552
16 by 16	69	77	2336	2476
32 by 32	133	145	9792	10283

2.3. ZM tree

Zuras and McAllister proposed a method for constructing balanced delay trees, which has a better area-time product than binary tree [6]. Figure 3 shows the structure of ripple chain.

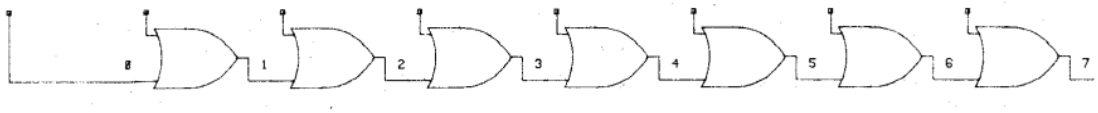

Figure 3: The structure of ripple chain.

The number represent the accumulated gate delay. Zuras and McAllister decomposed the ripple chain into sub-chains and then serially combined the results of these sub-chains. This structure shown in figure 4 introduces limited additional layout complexity but offers substantial acceleration.

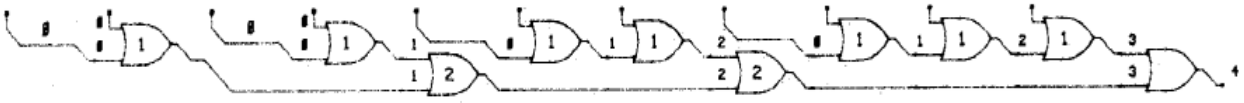

Figure 4: Second-order balanced delay tree.

When the total delay of the up chain equals that of the down chain, a delay balance is achieved. If 64 inputs are to be combined to produce a single result, this structure, utilizing 2 wires, would incur 11 delays. The ripple circuit, despite having only one wire, would incur 63 delays. A binary tree, with only 6 delays, necessitates 6 wires. Zuras and McAllister demonstrated that a k-order ZM tree can accomplish the tasks of a 2k-order binary tree, albeit with higher latency, yet it saves nearly half of the area.

Its structure is very regular and requires very few macro modules, making it highly valuable for use. The figure 5 shows the ZM tree multiplier structure. In the research shown in figure 6 conducted by Zuras and McAllister, it was observed that the delay performance of the fourth-order

DOI: 10.54254/2755-2721/127/2025.20275

balanced delay tree gradually deteriorates in comparison to the Wallace tree. However, it demonstrates significant advantages in optimizing the performance of small-scale multipliers.

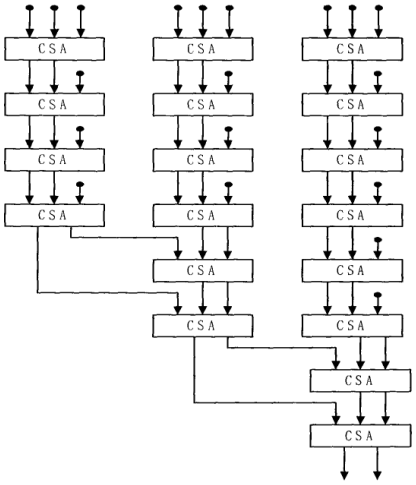

Figure 5: The structure of first-order ZM tree.

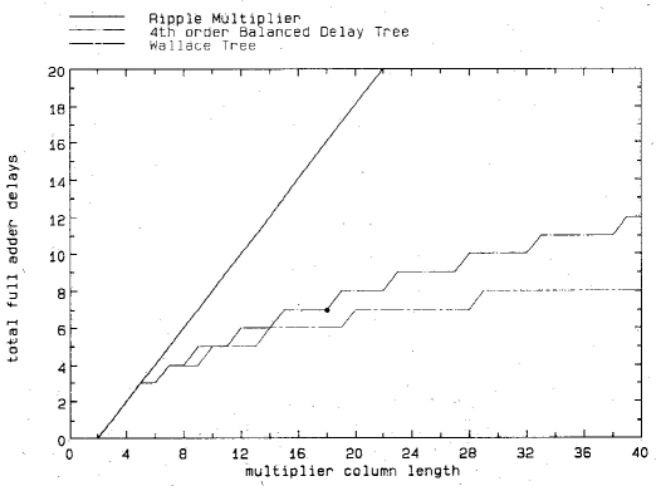

Figure 6: Delay for fourth-order ZM tree, Wallace tree, and ripple chain.

2.4. OS tree

Mou and Jutand invented an 'Overturned Stairs' tree, which has a regular and compact layout resulting from their recursive structure [7]. Taking a first-order OS tree shown in figure 7 as an example, it is divided into a subject and a root, and the subject can be constructed recursively. A first-order OS tree can add N operands in $O((2N)^{\frac{1}{2}})$ time, requiring 3 TLWs. By connecting first-order OS Trees through a linear structure, one can derive higher-order OS Tree. The figure 8 shows the structures of second-order OS tree multiplier.

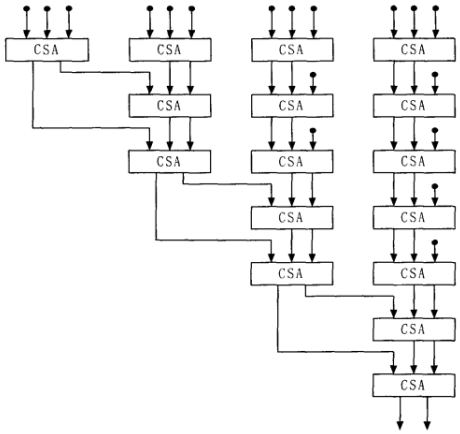

Figure 7: The structure of first-order OS tree.

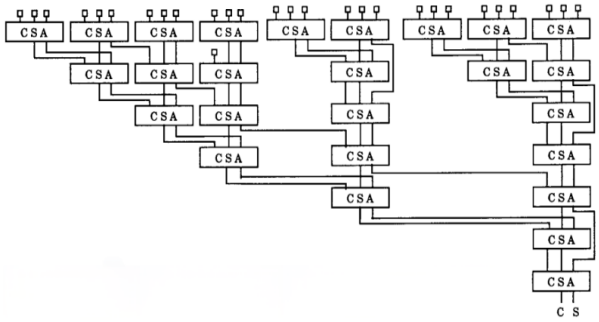

Figure 8: The structure of second-order OS tree.

In the study shown in table 2 conducted by Mou and Jutand, it was observed that, compared with 4-2 tree, second-order and third-order OS tree have excellent performance. In addition, combining OS tree of different orders with ZM tree can achieve better performance. Higher order ZM tree perform poorly.

DOI: 10.54254/2755-2721/127/2025.20275

Table 2: The maximum number of operands that can be added by trees of height n.

n	ZM	OS1	ZM2	OS1-ZM	OS2	OS2-ZM	OS3	4-2	Wallace
1	3	3							3
2	4	4						4	4
3	6	6							6
4	8	9						8	9
5	11	13	12						13
6	14	18	16	19	19			16	19
7	18	24	23	27	28				28
8	22	31	30	38	41			32	42
9	27	39	41	52	59	60	60		63
10	32	48	52	70	83	87	88	64	94
11	38	58	68	92	114	125	129		141
12	44	69	84	119	153	177	188	128	211
13	51	81	106	151	201	247	271		316
14	58	94	128	189	259	339	385	256	474

2.5. Comparison of different tree structures

The iterative CSA array and the Dadda tree represent two extremes in the realm of arithmetic circuits. The former is characterized by its conventionality, albeit at the expense of speed; conversely, the latter offers superior speed but poses significant implementation challenges. Various other tree structures necessitate a delicate balance between operational velocity and chip area. ZM tree and OS tree sacrifice speed performance, require more latency and area. But their circuit structure is more regular, which is more conducive to the implementation of VLSI.

The author compared various structures composed of CSA that compressed 18 partial products and focused on studying some key factors. Among them, the additional unbalanced path refers to the path that requires additional waiting time, which is related to the operation speed and hardware utilization efficiency of the multiplier. Cross unit connections are closely related to layout design, and the fewer the number, the more regular the layout. Simultaneously, the hardware resource consumption of these structures was calculated. The study shown in Table 3 found that in structures using CSA as the basic addition unit, the Wallace tree has the fastest speed but but its structure is the most irregular. The running speed of OS tree is faster than ZM tree of the same order, but it has more cross cell connections and is not as regular as ZM tree. If a 4-2 compressor is used as the basic addition unit, the delay of these structures will be reduced. However, considering the structure of the 4-2 compressor itself, the delay of the tree structure has not been significantly improved. It is worth noting that the impact of cross cell wiring on the area is limited, as the wiring can often be placed on top of the device. The number of basic addition units and their own structural size have the greatest impact on the area. Therefore, based on the data in Table 3, the ZM tree uses more CSAs, which means a larger area. The OS tree improves the regularity of the Wallace tree structure without affecting the area.

Table 3: The comparison between typical structure.

Tree Structure	Delay	Additional unbalanced paths	Cross unit connections	Hardware resource consumption
Linear arrays	16*(CSA)	0	0	16*(CSA)
Wallace tree	6*(CSA)	0	5	16*(CSA)

DOI: 10.54254/2755-2721/127/2025.20275

Table 3: (continued).

First-order ZM tree	8*(CSA)	2	2	18*(CSA)
First-order OS tree	6*(CSA)	1	3	16*(CSA)

3. Optimization and Analysis of Tree Structure

3.1. Challenge

The Wallace tree multiplier is renowned for its exceptional speed. However, an examination of the circuit architecture of carry-save adder reveals that the delay introduced by XOR gates is significantly greater, resulting in a longer delay for the SUM output compared to that of the CARRY output. Furthermore, the process by which the carry-save adder compresses three input bits into two is inherently unbalanced. This imbalance contributes to the irregular structure of Wallace tree multipliers, which are predicated on CSA, thereby rendering them susceptible to issues such as clock skew and constraining their applicability in VLSI environments. Designing a basic unit with good performance is challenging.

Researchers have sought to enhance performance of multipliers through several avenues: refining the Booth algorithm, exploring alternative algorithms, designing circuits with superior performance characteristics, or optimizing the adder configurations utilized in the final summation of multipliers. The challenge lies in designing a regular circuit structure that makes the multiplier easy to implement, while also having good speed performance and a smaller area.

3.2. Researches on Optimization

The conventional 3-2 compressor is characterized by a delay of two XOR gates along its critical path. Veeramachaneni has refined the architecture of the 3-2 compressor, thereby reducing the time required for transistor switching along the critical path [8]. Specifically, Veeramachaneni substitutes the first XOR gate with an XOR-XNOR configuration and replaces the second XOR gate with a MUX. In contrast, the traditional 4-2 compressor comprises two CSAs and exhibits a delay corresponding to four XOR gates along its critical path. The research conducted by Hsiao et al. has optimized the structure of the 4-2 compressor, resulting in a reduction of its delay to that equivalent to three XOR gates [9]. Similar to the 3-2 compressor, Veeramachaneni employs XOR-XNOR and XOR techniques to enhance the structural efficiency of the 4-2 compressor, resulting in a reduction of the critical path delay to two MUX and one XOR gate. Likewise, the architecture of the 5-2 compressor can be optimized using the same methodology, yielding a critical path delay of three multiplexers and one XOR gate. Some scholars have designed and optimized compressors with high compression ratios, which exhibit superior performance characteristics. Rouholamini refined the architecture of the conventional 7-2 compressor, resulting in a reduction of the critical path delay from seven XOR gates to six, thereby achieving enhanced operational speed [10].

Table 4: The comparison between different compressor.

Partial products	Compressors	Hardware resource consumption	Delay
17	3*(7-2)	30*(XOR)+15*(MUX)	12
	5*(5-2)	30*(XOR)+10*(MUX)+10*(additional logic circuits)	12
	7*(4-2)+1*(3-2)	30*(XOR)+15*(MUX)	11
	13*(3-2)+1*(4-2)	30*(XOR)+15*(MUX)	11

DOI: 10.54254/2755-2721/127/2025.20275

Table 4: (continued).

22	4*(7-2)	40*(XOR)+20*(MUX)	12
	6*(5-2)+1*(4-2)	40*(XOR)+14*(MUX)+12*(additional logic circuits)	11
	10*(4-2)	40*(XOR)+20*(MUX)	12
	17*(3-2)+1*(5-2)	40*(XOR)+19*(MUX)+2*(additional logic circuits)	12
	18*(3-2)+1*(4-2)	40*(XOR)+20*(MUX)	13

The author compared the hardware resource consumption and delay of structures that compress 17 partial products and 22 partial products using different compressors. For ease of comparison, it is assumed that a 3-2 compressor consists of 2 XORs and 1 MUX, with a delay of 2 XORs. A 4-2 compressor consists of 4 XORs and 2 MUXes, with a delay of 3 XORs. A 5-2 compressor consists of 6 XORs, 2 MUXes, and 2 additional logic circuits with a delay of 4 XORs. A 7-2 compressor consists of 10 XORs and 5 MUXes, with a delay of 6 XORs. The research shown in table 4 above shows that various compressors seem unable to reduce hardware resource consumption. Note that the delay here is in XOR units. When the number of partial products is appropriate, mixing compressors with different compression ratios can bring advantages.

Numerous scholars choose to use a mixture of compressors with different compression ratios to optimize the Wallace tree structure for instance, Shangshang Yao designed a Wallace tree structure that utilizes a combination of 3-2, 4-2, and 5-2 compressors to compress 13 partial products [11]. In this structure, the first stage uses four 4-2 compressors, the second stage uses two 3-2 compressors, and the third stage uses one 5-2 compressor. The number of XOR gates is 23, the number of binary data selectors is 11, the compression level is 3, and the delay is 9 XORs. Compared with traditional Wallace tree, its critical path delay is significantly reduced. Zhongmin Zhao integrated CSAs and 4-2 compressors to compress 18 partial products when designing the Wallace tree structure [12]. The first stage uses CSA, the second stage uses 4-2 compressors, the third stage uses CSA, and the fourth stage uses 4-2 compressors. This structure is relatively complex, but it can effectively utilize hardware resources and improve summation efficiency. Its relatively regular structure avoids the imbalance of delay.

Dunshan Yu proposed a new structure for delay balanced tree [13]. Dunshan Yu first points out that the TLW between subtrees cannot accurately indicate the complexity of connecting lines, as the TLW of the first-order ZM tree and OS tree composed of 4-2 adders are both 2. Dunshan Yu proposed using the TLWT (Trans Layer Wires of Tree) of the entire tree to measure the complexity of connections. The calculation method is as follows.

$$\mathrm{TLWT} = \sum_{i=1}^{n-1} TLW_i \times BC_i \quad (1)$$

In the formula, n is the number of subtree, TLW is the number of connecting lines between adjacent subtrees, and BC is the minimum number of 4-2 compressors in adjacent subtrees. The TLWT of the OS tree is 12, and the TLWT of the ZM tree is 10. And the TLWT of the Wallace tree that completes 32-bit addition is 34. Therefore, TLWT can effectively demonstrate the complexity of connections in various trees. Dunshan Yu proposed a tree with a width of 2 and a structure as shown in the figure 9. Its TLWT is 8, which is lower than ZM tree and OS tree, and its regularity is also better.

DOI: 10.54254/2755-2721/127/2025.20275

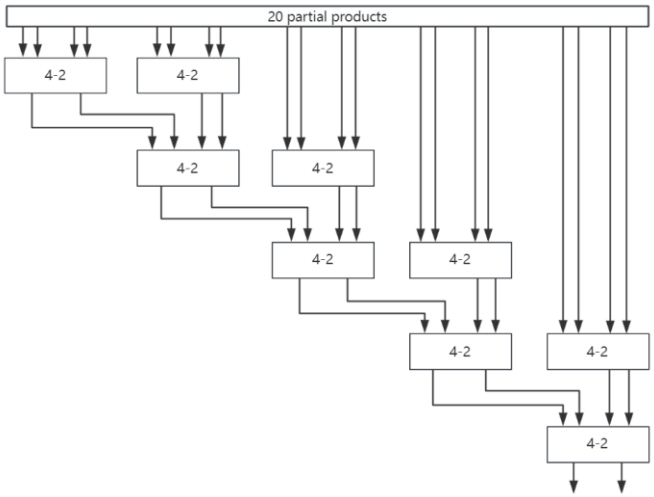

Figure 9: Proposed structure of tree multiplier.

Ykuntam et al. have chosen to utilize parallel prefix adders (PPAs) to combine the partial products of the last row with the sum from the preceding stage [14]. Table 5 below shows their analysis of fast Wallace tree multipliers designed with various PPAs, including Kogge-Stone adder, Sklansky adder, Brent-kung adder, Ladner-Fischer adder, and Han-Carlson adder. The analysis reveals that these configurations attain a reduction in latency without substantially augmenting the area. Among them, Kogge-Stone adder used more resources and occupied more area, but achieved faster speed. In contrast, Brent-kung used less area but also achieved satisfactory speed. In conclusion, the utilization of PPAs in Wallace tree multipliers presents a viable strategy for optimizing both speed and area, and the choice of PPA should be guided by a thorough understanding of the trade-offs involved.

Table 5: The comparison between different compressor.

Input size	Multiplier structure	Area(No. of LUTs)	Delay(ns)
16-bit	Traditional Wallace tree multiplier	590	36.7
	Wallace tree multiplier using Kogge-Stone adder	634	29.44
	Wallace tree multiplier using Sklansky adder	599	30
	Wallace tree multiplier using Brent-kung adder	598	32.37
	Wallace tree multiplier using Ladner-Fischer adder	596	31.57
	Wallace tree multiplier using Han-Carlson adder	601	30.19

4. Conclusion

This article introduces various typical tree multiplier structures and their performance, and compares them. It was found that the Dadda tree has the best speed performance but is difficult to implement due to its irregular circuit structure and require a large area, while the delay balanced tree circuit is regular and easy to implement but sacrifices speed performance. Among them, the ZM tree has a higher delay but has advantages in small-scale multipliers. The performance of

DOI: 10.54254/2755-2721/127/2025.20275

higher-order OS tree is closer to that of Wallace tree. Over the years, many scholars have optimized the structure of compressors, designed novel multiplier addition array structures, and used better performing adders to achieve final summation, overall optimizing the speed and area performance of tree multipliers. The optimization strategies discussed in this article are not exhaustive; in fact, numerous other methods exist to enhance the performance of multipliers, such as refining the Booth algorithm, which is instrumental in generating partial products. The focus of multiplier optimization varies significantly depending on the specific applications of the microprocessor. By integrating the demands for microprocessors with varying performance characteristics within the IC industry or other industry, and subsequently analyzing and contrasting the optimization strategies for multipliers, the discourse will acquire enhanced persuasiveness and practical applicability.

References

[1] *Wallace C S. A suggestion for a fast multiplier. IEEE Transactions on electronic Computers, 1964 (1): 14-17.*

[2] *Dadda L. Some schemes for parallel multipliers. IEEE Computer Society Press, 1990.*

[3] *Jessani R M, Putrino M. Comparison of single-and dual-pass multiply-add fused floating-point units. IEEE Transactions on Computers, 1998, 47(9): 927-937.*

[4] *Habibi A, Wintz P A. Fast multipliers. IEEE Transactions on Computers, 1970, 100(2): 153-157.*

[5] *Townsend W J, Swartzlander Jr E E, Abraham J A. A comparison of Dadda and Wallace multiplier delays. Advanced signal processing algorithms, architectures, and implementations XIII. SPIE, 2003, 5205: 552-560.*

[6] *Zuras D, McAllister W H. Balanced delay trees and combinatorial division in VLSI. IEEE journal of solid-state circuits, 1986, 21(5): 814-819.*

[7] *Mou Z J, Jutand F. A class of close-to-optimum adder trees allowing regular and compact layout. Proceedings., 1990 IEEE International Conference on Computer Design: VLSI in Computers and Processors. IEEE, 1990: 251-254.*

[8] *Veeramachaneni S, Krishna K M, Avinash L, et al. Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors. 20th International Conference on VLSI Design held jointly with 6th International Conference on Embedded Systems (VLSID'07). IEEE, 2007: 324-329.*

[9] *Hsiao S F, Jiang M R, Yeh J S. Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers. Electronics Letters, 1998, 34(4): 341-342.*

[10] *Rouholamini M, Kavehie O, Mirbaha A P, et al. A new design for 7: 2 compressors. 2007 IEEE/ACS International Conference on Computer Systems and Applications. IEEE, 2007: 474-478.*

[11] *Shangshang Y, Li S. Design of a New Floating Point Multiplier Based on Hybrid Compression Structure. Microelectronics and Computer, 2021, 38 (09): 74-78.*

[12] *Zhongmin Z. Design and Implementation of Multiplier Unit in 64 bit High Performance Embedded CPU. Tongji University, 2007*

[13] *Dunsha Y, Xubang S. Design of 32-bit fixed/floating point multiplier. Journal of Semiconductors, 2001, (01): 91-95*

[14] *Ykuntam Y, Pavani K, Saladi K. Design and analysis of High speed wallace tree multiplier using parallel prefix adders for VLSI circuit designs. 2020 11th international conference on computing, communication and networking technologies (ICCCNT). IEEE, 2020: 1-6.*