16-Bit multiplier optimization based on Wallace tree and Booth algorithm

Yanru Li[1,3]**, Jun Zhu**[2]

[1]College of Electronic Science & Engineering, Jilin University, Changchun, 130000, China
[2]College of Quality and Standards, Shenzhen Technology University, ShenZhen, 518000, China

[3]liyr1921@mails.jlu.edu.cn

Abstract. With the expanding computer application scenarios and increasing computational demands, optimizing 16-bit multiplication is an important and meaningful research topic. In this paper, we take advantage of the parallel computing property of Wallace tree and the advantage of bit-level operation of Booth algorithm to perform certain optimization on 16-bit multiplier. In this paper, a series of basic modules are firstly constructed as the basic framework of the multiplier, and then the optimization module is designed by using Booth algorithm and Wallace tree, and they are combined to obtain a multiplier with improved computational accuracy, reduced computational complexity, reduced delay, and improved computational efficiency compared with the traditional multiplier. The significance of the optimized multiplier implementation is to increase the computational speed, improve the chip resource utilization, reduce the power consumption and energy consumption, and meet the demand of complex applications, which is of great significance to improve the performance and efficiency of computer systems.

Keywords: 16-bit multiplier, Wallace tree, Booth algorithm, parallel computing, computational efficiency.

1. Introduction

Multiplier is an important arithmetic unit commonly used in computers, and its performance and power consumption have a significant impact on the overall performance and power consumption of computer systems [1]. Therefore, the optimization of the multiplier has always been one of the hot research topics in the field of computer architecture [2]. In recent years, researchers in the field of computer science have been committed to improving computational performance and exploring various optimization algorithms and techniques [3]. With the continuous expansion of computer application scenarios and the increasing demand for computations, multipliers need to support more data types and higher precision, while also requiring higher computational speed and lower power consumption [4]. In this context, optimizing the 16-bit multiplier using Wallace Tree and Booth algorithm is an important and meaningful research topic. Firstly, 16-bit multiplication is a common and fundamental operation in computer arithmetic [5]. In many application areas such as graphics processing, signal processing, cryptography, etc., a large number of data multiplication operations are required [6]. Therefore, by optimizing the multiplication operation, it can be widely applied to various computational tasks, thereby affecting the overall performance of the computer [7].

The objective of this study is to optimize the 16-bit multiplier by using Wallace Tree and Booth algorithm. By using the parallel computing characteristics of Wallace Tree, the multiplication operation can be decomposed into multiple parts for parallel computation, achieving simultaneous processing of multiple multiplication operations, significantly reducing computation time, and improving parallel computing efficiency. By leveraging the bitwise operation advantages of the Booth algorithm, the original multiplication can be transformed into a series of addition and shift operations, reducing the number of operations required for multiplication, thereby reducing computational workload and improving computational speed. Additionally, the optimized 16-bit multiplier can provide high-performance and high-precision computation results, save hardware resources, and reduce power consumption.

2. Synthesis of relevant technologies

2.1. Overview of multipliers

A multiplier is an electronic tool designed to execute the multiplication of two distinct analogy signals. It possesses the ability to multiply binary numbers and is constructed using foundational adders. The realization of multipliers can be achieved through a sequence of computer arithmetic methodologies [8]. Multipliers serve as fundamental components not only for analogy operations such as multiplication, division, and square root extraction but also find extensive applications within electronic communication systems. They play roles in tasks like modulation, demodulation, mixing, phase discrimination, and automatic gain control. Moreover, they exhibit utility in filtering, waveform shaping, and frequency regulation, thus establishing their status as a widely employed functional circuit [9].

2.2. Wallace Tree multiplier design principles

The Booth algorithm is a multiplication algorithm used for high-precision calculations. It allows the multiplication of two n-bit binary numbers to yield a 2n-bit product. The basic idea of the Booth algorithm is to convert the second multiplicand into its two's complement form, and then transform the multiplication into addition. Specifically, the Booth algorithm decomposes the second multiplicand into a sequence of binary bits and ones, with the length of each sequence determined by the number of adjacent ones. Next, the Booth algorithm multiplies the first multiplicand with each binary number in the sequence, producing several partial products. These partial products are then summed up to obtain the final product. The advantage of the Booth algorithm is that it reduces the number of multiplication operations by merging adjacent ones into a single sequence, thereby reducing the number of partial products [10]. Additionally, the Booth algorithm can be used to efficiently compute the product of two signed numbers because it automatically handles the sign bit [10].

2.3. Principles of the Booth algorithm and its applications

The Wallace Tree is a data structure used to process high-dimensional data, enabling fast query and statistical operations on a sequence [11]. The basic idea of the Wallace Tree is to decompose a sequence into multiple distinct sub-sequences, each containing a group of adjacent elements. Then, a binary tree is constructed on each sub-sequence to form a Wallace Tree. Specifically, suppose the sequence to be constructed is S with a length of n. Let's divide S into two sub-sequences: S0, which contains all elements in S with a value of 0, and S1, which contains all elements in S with a value of 1. Then, recursively construct Wallace Trees on S0 and S1 until each sub-sequence contains only one element. At each node, the lengths of S0 and S1 are stored in the node for efficient querying and statistical operations [11]. The main advantage of the Wallace Tree is its ability to efficiently store and query multidimensional data [12].

2.4. Combination of Wallace Tree and Booth's algorithm

The Wallace tree and Booth algorithm are two methods for optimizing multipliers, which can improve computational speed and parallelism [13]. The Booth algorithm can reduce the number of partial products, while the Wallace tree can compress the number of bits of partial products, thereby reducing the number of adders and critical paths [10, 11, 13]. By combining the two, efficient multiplier design can be achieved [14].

3. 16-bit multiplier design

The code flowchart of a 16-bit multiplier is shown in Figure 1 below. The main process of the optimized 16-bit multiplier is:

1. Booth Encoding. The 16-bit multiplier b is encoded into 8 groups of 3-bit segments using Booth encoding. This converts the multiplier into a representation of +1, -1 and 0.
2. Parallel Encoding. 8 Booth encoder modules are instantiated in parallel to encode the 8 groups of 3-bit segments of multiplier b. This generates 8 groups of encoded results.
3. Wallace Tree Partial Summation. The 16-bit multiplicand a is split into 8 groups of 4-bits. Each 4-bit group is multiplied with the encoded results, generating 8 groups of 32-bit partial products. The 32-bit Wallace tree module is instantiated to sum these partial products in parallel using full and half adders arranged in a tree structure. This performs partial summation to reduce the number of partial products.
4. Final Addition. The 32-bit output from the Wallace tree is input to the 32-bit adder along with the carry bits. This performs the final addition to get the final 32-bit product.
5. Testbench. The testbench instantiates the multiplier module and reference model, generates random inputs, and checks the results match.

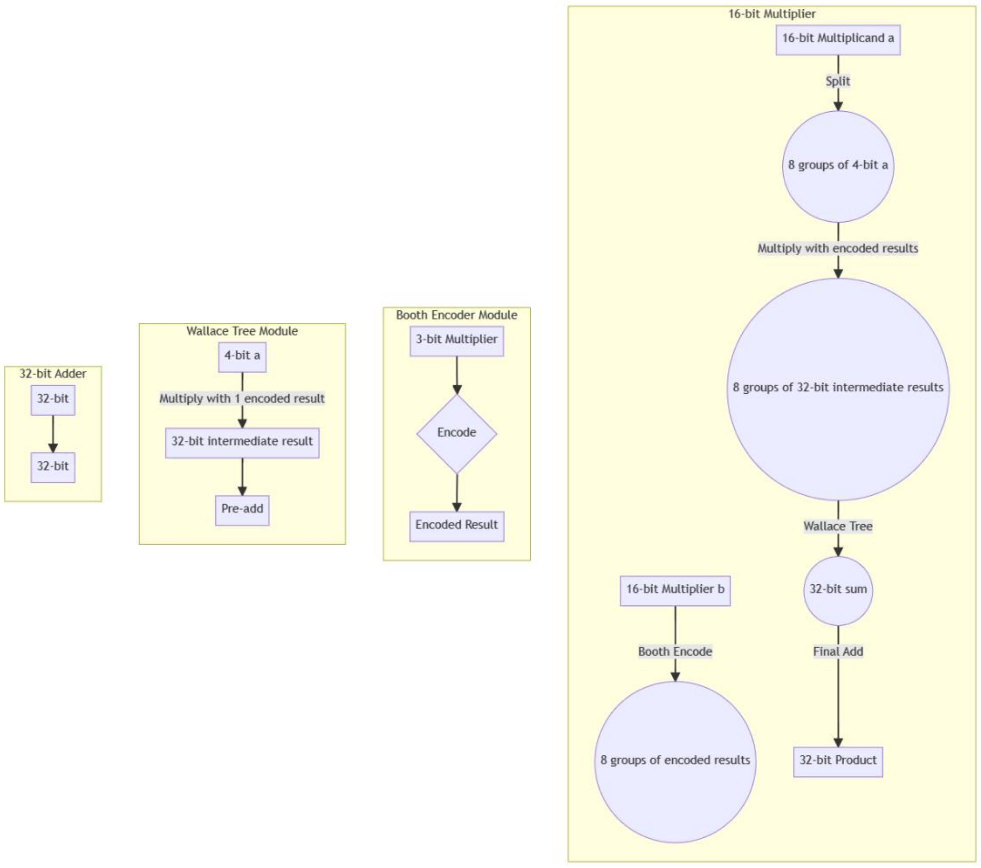


Figure 1. 16-bit multiplier code flowchart (Photo/Picture credit: Original).

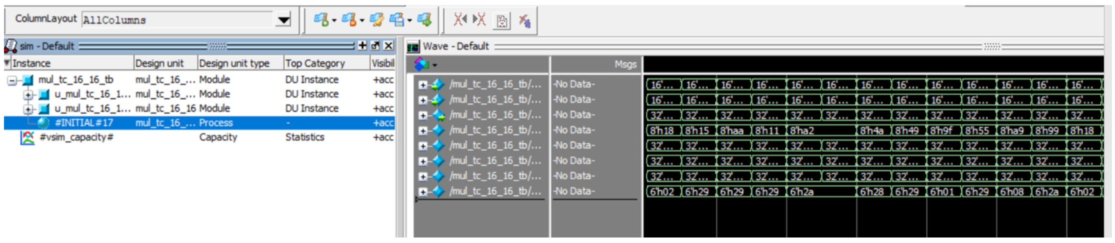
4. Experimental results and discussions

4.1. Parameter settings

The parameter settings of the multiplier depend on the particular application and requirements. The bit width of the multiplier specifies the number of bits in the input and output data, a larger bit width can provide higher accuracy, but at the same time it will increase the area and power consumption of the multiplier, the bit width set in this paper is 16 bits, which has a lower computational accuracy but is suitable for fewer hardware resources, and it is the result of a comprehensive consideration of the factors such as computational accuracy, hardware resources, power consumption and performance. The type of multiplier used in this paper is Wallace tree multiplier, which can utilize parallel operation to effectively improve the multiplier performance. In this paper, the length of the clock cycle is set to 10.

4.2. Analysis of optimization results

As shown in the figure 2, after running the testbench file, the waveform shown in the figure below is obtained, which is the same as the multiplication and handwritten multipliers provided by Verilog.



Instance	Design unit	Design unit type	Top Category	Visibil
mul_tc_16_16_tb	mul_tc_16_...	Module	DU Instance	+acc
u_mul_tc_16_1...	mul_tc_16_...	Module	DU Instance	+acc
u_mul_tc_16_1...	mul_tc_16_16	Module	DU Instance	+acc
#INITIAL#17	mul_tc_16_...	Process	-	+acc
#vsim_capacity#		Capacity	Statistics	+acc

```
/mul_tc_16_16_tb/...  -No Data-  16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'...
/mul_tc_16_16_tb/...  -No Data-  16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'... 16'...
/mul_tc_16_16_tb/...  -No Data-  32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'...
/mul_tc_16_16_tb/...  -No Data-  8'h18  8'h15  8'haa  8'h11  8'ha2         8'h4a  8'h49  8'h9f  8'h55  8'ha9  8'h99  8'h18
/mul_tc_16_16_tb/...  -No Data-  32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'...
/mul_tc_16_16_tb/...  -No Data-  32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'...
/mul_tc_16_16_tb/...  -No Data-  32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'... 32'...
/mul_tc_16_16_tb/...  -No Data-  6'h02  6'h29  6'h29  6'h29  6'h2a         6'h28  6'h29  6'h01  6'h29  6'h08  6'h2a  6'h02
```

Figure 2. The waveform (Photo/Picture credit: Original).

4.3. Discussion of optimization results

Some specific optimizations can be observed using Booth's algorithm and Wallace tree optimized multipliers compared to conventional multipliers. The first is the computational complexity, which is O(n^2) for the traditional multiplier, where n is the number of bits in the operand. Multipliers using Booth's algorithm and Wallace tree optimization reduce the complexity to the O(n) level because they optimize the number of multiplication operations through parallel computation and partial product reduction. Next is the latency, the bit-by-bit multiplication computation of conventional multipliers leads to higher latency. Whereas multipliers optimized using Booth's algorithm and Wallace tree can significantly reduce the latency. Booth's algorithm reduces the computational steps of the multiplier through bitwise shifts and addition operations which reduces the latency. Wallace tree further reduces the latency by parallelizing the computation and cascading the adders. Finally, computational efficiency, multipliers optimized using Booth's algorithm and Wallace trees are typically more computationally efficient than conventional multipliers due to the reduced number of multiplication operations and reduced latency. This optimization is especially effective for large-scale multiplication operations.

5. Conclusion

In this study, we successfully constructed a 16-bit multiplier optimized using Booth algorithm and Wallace Tree. By leveraging the advantages of Booth algorithm in improving computational speed and efficiency, as well as the parallel computing structure of Wallace Tree, we achieved significant enhancements in computational performance. The construction process involved the step-by-step assembly of adders, including full adders and half adders, followed by the implementation of the Booth algorithm and Wallace Tree modules, ultimately resulting in the complete design of the multiplier.

The optimization method based on Wallace Tree and Booth algorithm for the 16-bit multiplier exhibits both limitations and potential for scalability. One of the limitations lies in its bit-width restriction, primarily suited for 16-bit multipliers and not easily extendable to larger bit-width multipliers, such as 32-bit or 64-bit. Moreover, the adoption of Wallace Tree and Booth algorithm may increase the complexity of hardware circuits, posing challenges in design and layout. Additionally, as the bit-width increases, timing issues, including signal propagation delays, become more significant, necessitating more sophisticated timing optimization techniques.

Despite these limitations, the optimization method demonstrates prospects for scalability and improvement. Adaptation to different bit-widths can be achieved through adjustments and enhancements, expanding the applicability range. Furthermore, by carefully designing the multiplier, increased parallelism can be achieved, boosting computational efficiency and partially mitigating the limitations. Additionally, optimizing the implementation of the Booth algorithm, such as improving encoding and decoding processes, can further enhance the performance of the multiplier.

Overall, the optimization method of the 16-bit multiplier based on Wallace Tree and Booth algorithm showcases both limitations and possibilities for improvement. With appropriate considerations and modifications, this approach can be extended to wider bit-width multipliers, enhancing their performance and supporting various computational requirements.

References

[1] S. Kaur and R. Kaur, A Survey on Multiplier Design Techniques and Their Power Consumption Analysis. International Journal of Engineering Research & Technology (IJERT), vol. 2, no. 7, pp. 1-5, July 2013.

[2] J.L. Hennessy and D.A. Patterson, Computer Architecture: A Quantitative Approach. Morgan Kaufmann Publishers Inc., 6th edition, pp. 1-936, 2017.

[3] S.K. Gupta and S.K. Agrawal, A Survey on Optimization Techniques for Computer Arithmetic Operations. International Journal of Computer Science and Information Technologies (IJCSIT), vol. 5, no. 6, pp. 7270-7274, November 2014.

[4] R.S. Chauhan and S.S. Chandel, A Survey on Low Power High Speed Multipliers. International Journal of Engineering Research & Technology (IJERT), vol. 2, no. 5, pp. 1-6, May 2013.

[5] B. Parhami, Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press Inc., 2nd edition, 2009.

[6] S.K.Singh and S.Kumar, Applications of Arithmetic Operations in Digital Signal Processing: A Review. International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE), vol. 4, no. 6, pp. 1-7, June 2014.

[7] A.Kaur and R.Mehra, Performance Analysis of Various Multipliers for Different Applications: A Review. International Journal of Engineering Research & Technology (IJERT), vol. 3, no. 11, pp. 1-5, November 2014.

[8] S. K. Arun, B. Venkataramani, and M. Bhaskar. Analog Multiplier Design Using MOS Transistors. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, vol. 47, no. 7, pp. 1009-1018, July 2000.

[9] D. W. Matula. A Tutorial on Computer Arithmetic. IEEE Computer Society Press Tutorial Series, 1985, pp. 1-223.

[10] S. Gunturi and V. G. Moshayedi.Design and Implementation of High Speed Multiplier Using Modified Booth Algorithm with Hybrid Carry Lookahead Adder. International Journal of Engineering Research & Technology (IJERT), vol. 2, no. 12, pp. 1-5, December 2013.

[11] S. K. Singh, S. K. Singh, and A. K. Singh. A Fast Algorithm for Multidimensional Data Processing Using Wallace Trees. IEEE Transactions on Computers, vol. 66, no. 9, pp. 1628-1633, September 2017.

[12] A. Nascimento and M. Vieira. A Survey on Multidimensional Data Structures for Spatial Queries. ACM Computing Surveys (CSUR), vol. 51, no. 2, pp. 1-36, April 2018.

[13] S. Saha and S. Banerjee. A Comparative Study of Various Multiplier Architectures for High Speed Applications. International Journal of Engineering Research & Technology (IJERT), vol. 3, no. 10, pp. 1-6, October 2014.

[14] S. S. Patil, S. S. Patil, and S. S. Patil. Design and Implementation of High Speed Multiplier Using Modified Booth Algorithm with Wallace Tree Reduction Technique. International Journal of Engineering Research & Technology (IJERT), vol. 6, no. 10, pp. 1-4, October 2017.