Proceedings of the 5th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/121/2025.19846

High-performance Divider Performance Improvement

Zhaomeng Yan[1,a,*]

[1]*College of Electronic and Electrical Engineering, Henan Normal University, Henan, 453007, China*
a. yanzhaomeng11@gmail.com
**corresponding author*

Abstract: Based on the applications of high-performance dividers in research and high-tech fields as well as their important roles in research, this paper summarizes the methods for improving the performance of high-performance dividers. Through the analysis of high-performance divider schemes, this study investigates the shortcomings of ordinary dividers, with an emphasis on analyzing the operating methods of two ordinary dividers, namely the restoring remainder divider and the non-restoring remainder divider. This paper mainly analyzes how three types of high-performance dividers improve their performance, which are Radix-16 SRT divider, Goldschmidt algorithm divider, and Goldschmid double-precision floating-point divider. It illustrates from aspects such as area, parallel computing, etc. why these algorithms can enhance the performance and energy efficiency of dividers. This article aims to provide some directional assistance to those researching the performance of dividers. I hope that during their research, they can utilize this article to obtain useful information that will help them more smoothly conduct research on high-performance dividers and advance digital circuit design technology, especially in terms of improving computational speed, reducing latency, and optimizing power consumption.

Keywords: Divider, software, hardware.

1. Introduction

A high-performance divider is a specialized hardware component used in digital circuits to perform division operations. It is widely applied in fields such as signal processing, image processing, communication systems, and computer graphics. The design goal is to complete division calculations at a higher speed and efficiency to meet the computational performance demands of modern applications. Traditional division algorithms, such as restoring and non-restoring remainder methods, may be limited in terms of speed and resource occupation. Therefore, high-performance dividers often employ more advanced methods. For instance, iterative algorithms (like the Newton-Raphson method) can achieve rapid convergence by gradually approximating the quotient value to accelerate calculations. Additionally, parallel processing and pipelining structures can effectively reduce operational latency and enhance system throughput. In terms of hardware implementation, high-performance dividers are typically based on FPGA or ASIC designs to optimize power consumption and performance for specific applications. With proper architectural design, high-precision calculations can be achieved while controlling power consumption, making them suitable for embedded and mobile devices. Moreover, error analysis and robustness testing are indispensable parts

© 2025 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/121/2025.19846

of the design process, ensuring stable operation under various conditions. In summary, high-performance dividers are an indispensable part of modern digital systems. Their optimized design not only improves computational speed but also provides strong support for a variety of applications. Therefore, this paper aims to summarize and outline schemes for improving the performance of dividers, hoping to guide their application in the fields of computer and chip design.

2. Technical overview

Improvement methods for high-performance dividers mainly involve hardware design, algorithm optimization, and parallel computing. For instance, hardware design improvements include using high-performance computing units and pipelining to reduce latency. Efficient division algorithms optimize remainder processing and parallel computing.

Utilizing multi-core processors and GPUs can accelerate software optimization. Compilation optimizations involve the use of optimized libraries, etc. However, the most critical aspects are still based on algorithm optimization and parallel computing. For example, algorithm optimization can reduce computational steps, thereby decreasing the number of iterations and the amount of computation. It can also improve computational accuracy and reduce errors that occur during the process. Additionally, it can decrease resource consumption. Optimized algorithms can reduce the demand for processors and improve the utilization of hardware resources, thus enhancing efficiency.

Parallel computing can perform multiple calculations within the same clock cycle, which can obviously speed up data processing, algorithm development, and numerical simulation. Parallel computing can also divide problems into different parts, which can be executed in parallel on multiple processors. In the case of dividers, this means dividing the dividend and divisor into different parts, with each processor calculating the division result of a part, and finally merging these results to obtain the answer.

3. Schemes of ordinary dividers and design background of high-performance dividers.

As Wang Weisheng mentioned in his article, there are two basic types of dividers: one is the divider using the restoring remainder algorithm, and the other is the divider using the non-restoring remainder algorithm. The following analysis will focus on these two basic dividers, examining the reasons for their low efficiency and the shortcomings in their design [1].

3.1. Restoring division algorithm

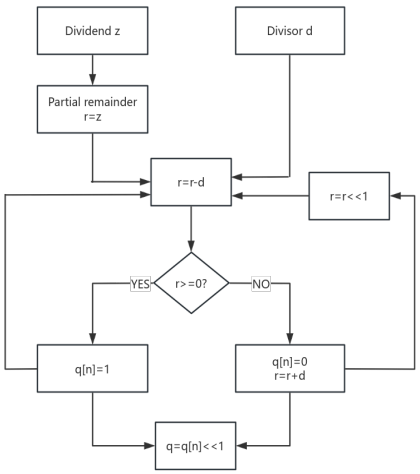


Figure 1: Restoring division algorithm

DOI: 10.54254/2755-2721/121/2025.19846

The restoring remainder algorithm is essentially equivalent to the handwritten vertical calculation can be seen in figure 1. Its principle is to subtract the divisor from the partial remainder. If the result is negative, then the process will restore the original partial remainder.

When using this method to calculate binary division, the dividend is taken as the remainder for calculation. When the result is 1, it means that the division operation can be carried out, that is, "divisible", and then the highest bit of the quotient is 1 [2]. If the result is negative, it means that the operation at this step cannot be carried out, which is usually called "not divisible". At this time, the highest bit of the quotient is 0, and the original partial remainder value is restored, and the partial remainder value is shifted one bit to the left and the above process is continued until the result is calculated.

This scheme cannot perform parallel computing, and the number of cycles will be very large, so it will reduce the efficiency of the divider. Moreover, shift and addition/subtraction operations are required in each iteration process, and it is usually difficult to complet5e within one clock cycle, so it will increase the area of the circuit and reduce the performance of the divider [3].

3.2. Non-restoring division algorithm

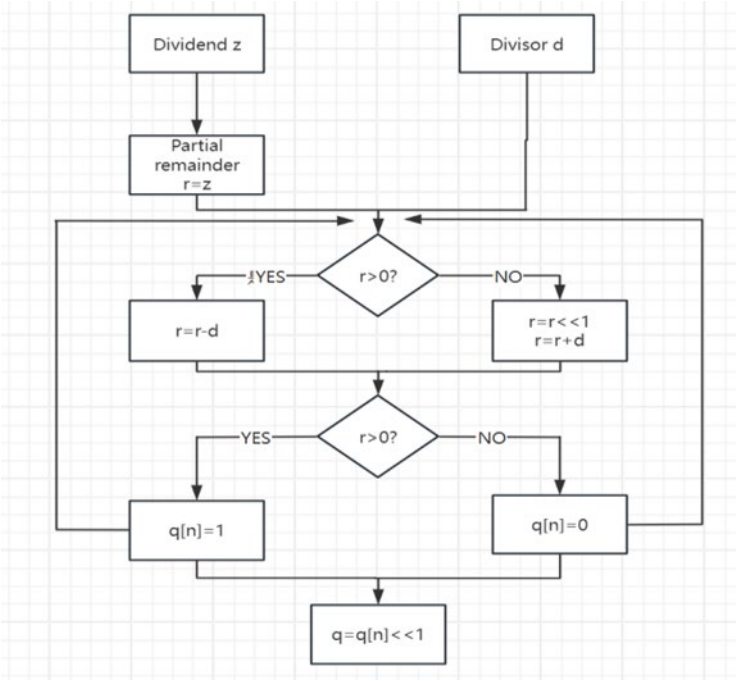


Figure 2: The process of an algorithm without restoring the remainder.

The process of algorithms with and without remainder restoration differs only slightly. In the remainder restoration algorithm, when the partial remainder is reduced by the divisor and becomes negative, the divisor is added to restore the original partial remainder value can be seen in figure 2. In contrast, the non-remainder restoration algorithm simply proceeds to the next loop when the partial remainder is less than the divisor after subtraction. The final results of both methods are the same.

These algorithms may yield negative values for the quotient and remainder during computation. If negatives occur, more computational power is required to convert these numbers into standard binary numbers, significantly increasing the computational load. Additionally, similar to the remainder restoration algorithm, it requires a greater number of iterations, and the efficiency decrease due to multiple iterations should not be overlooked. From the two common dividers mentioned above, it is evident that with the rapid advancement of technology, they can no longer meet the demands of

DOI: 10.54254/2755-2721/121/2025.19846

technological development. In modern high-precision and high-speed computing technologies, high-performance dividers are indispensable tools. Only by replacing common dividers with those of higher performance and efficiency can we optimize algorithm efficiency, reduce power consumption, and address complex applications. High-performance dividers play a broad and critical role in the modern computing field. In scientific computing areas such as astrophysical simulations, climate model building, and complex mathematical analysis, a large number of high-precision division operations are needed. High-performance dividers can quickly and accurately handle these calculations, providing strong support for scientific research.

4. Strategies for enhancing the performance of high-performance dividers

4.1. Radix-16 SRT divider

Based on Zhao Juan's research and design, to enhance the division speed, a series of bit-level short adders were adopted to implement complex multiplication values for odd and even multiples of divisors. This scheme, while slightly increasing the area, improved the rate of computing results, reduced computational redundancy, and decreased the delay in calculation time [4]. Below is the diagram of the divider.

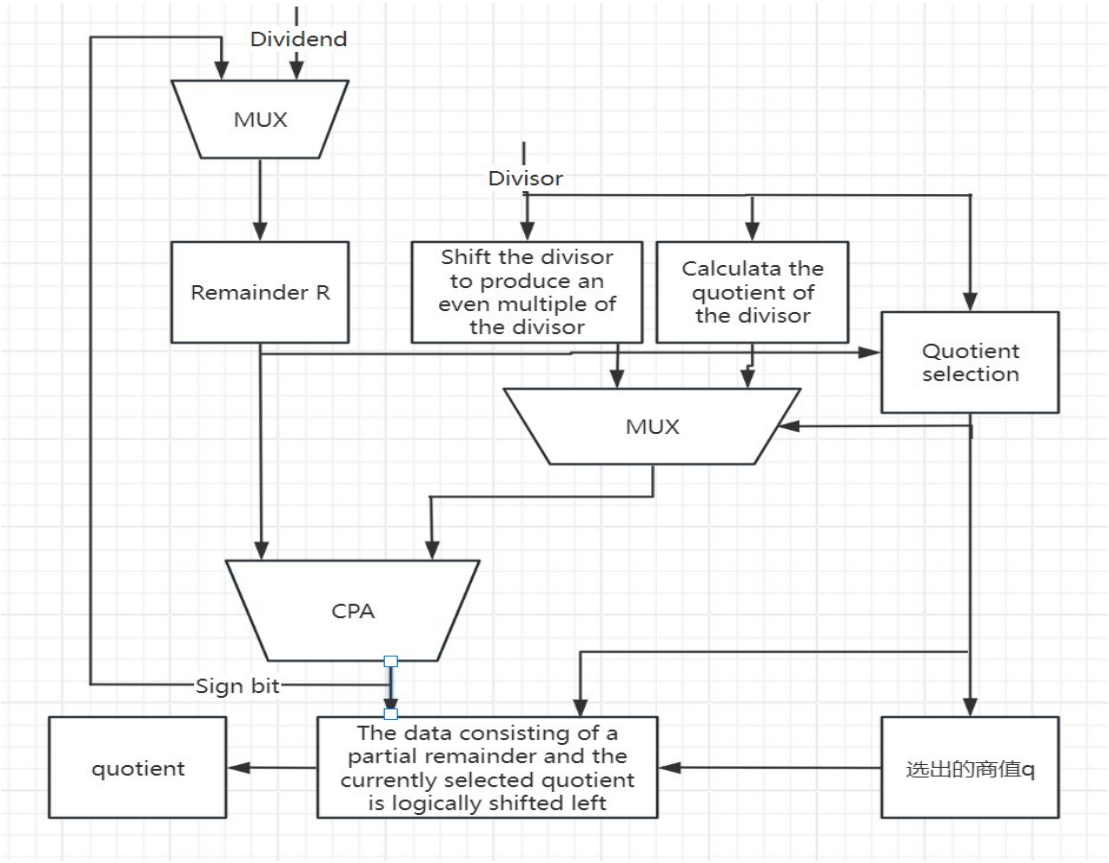


Figure 3: The process of the SRT divider.

First, after the divisor and dividend are inputted, parallel processing is carried out, allowing multiple bits of computation to occur simultaneously, unlike the sequential calculations required by both the restoring and non-restoring remainder algorithms can be seen in figure 3. This enables more division work to be completed within the same timeframe. Second, the method of using short adders to handle complex multiplication values for odd and even multiples of the divisor effectively reduces

DOI: 10.54254/2755-2721/121/2025.19846

the number of iterations. While traditional dividers require multiple iterations to approximate results, using short adders for segmented computation allows each iteration to be completed more quickly, reducing the time needed for iterations and thus decreasing the time it takes for the divider to compute results, enhancing the divider's efficiency. Third, short adders have a smaller bit width, which means they have less delay than standard adders. With short adders, input data can pass through the adder more quickly to obtain results, effectively speeding up the calculation process and improving the divider's performance. Lastly, the separation of odd and even numbers can also accelerate the divider's operation speed. As shown in the diagram, shifting the divisor to produce even multiples and calculating the odd multiples separately significantly reduces the complexity of the computation [5][6].

4.2. The goldschmidt algorithm divider.

According to the summary by He Tingting and others, the Goldschmidt algorithm can be described as an iterative-based division method. Its core concept is to transform division problems into multiplication problems, requiring two parallel multiplication operations in each iteration. The quotient and remainder of the division are calculated through an iterative approximation approach [7].

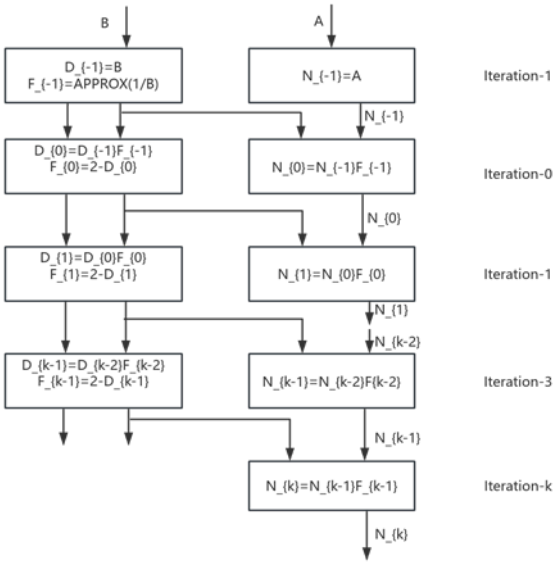


Figure 4: The calculation process of A divided by B.

The diagram illustrates the process of calculating A/B. Let the initial dividend be z and the divisor be d; during the iteration process, the values of the dividend and divisor are continuously updated. In the divider of this algorithm can be seen in figure 4, iterations can be expanded, allowing two iterations to be completed at once. Changes can be made to the right side of the equation in the diagram, potentially resulting in simpler forms, which reduces computational difficulty and workload. Additionally, A and B are processed to ensure the divisor is close to an integer, minimizing errors. It can also be inferred from the text that the divider can perform calculations with lower precision at the beginning of the iteration, and then increase the precision as required [8]. Similar to the SRT divider, the Goldschmidt divider can perform parallel computations, significantly enhancing its efficiency and calculation speed.

DOI: 10.54254/2755-2721/121/2025.19846

4.3. Goldschmidt double-precision floating-point divider

The Goldschmidt double-precision floating-point divider begins with an initialization phase where a minimax polynomial is used to approximate and obtain an estimated value of the divisor's reciprocal. During the iteration phase, the method of hardware reuse is employed to perform two Goldschmidt iterations [9]. By using this method, only one multiplier and one complement unit are used for the two iterations. Furthermore, by utilizing multiplication with a lower bit width, hardware resource consumption is reduced, and computation speed is increased. This approach maintains precision while reducing the complexity and resource requirements of multiplication operations, achieving performance without compromising and simultaneously reducing the use of hardware resources [10].

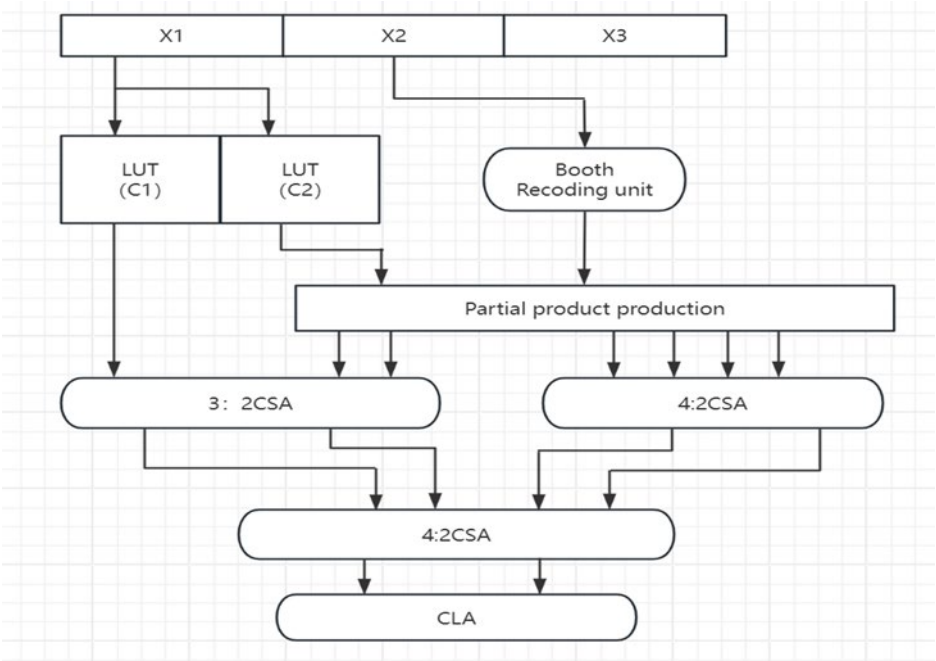


Figure 5: Minimax polynomial approximation

The figure shows the minimax polynomial approximation of the reciprocal function[11]. This process is used to optimize the initial estimated value can be seen in figure 5, and the optimization of the initial estimate can effectively reduce the required number of iterations. By designing a lookup table to provide a better initial iteration value x^{(0+)} , the number of iterations needed for the algorithm to reach the target accuracy can be reduced. In this process, the optimized polynomial coefficients can be implemented as a lookup table. During the actual division operation, the estimated value of the reciprocal of the divisor can be directly obtained through the lookup table, thus avoiding complex calculations and improving the efficiency of the division operation.

5. Conclusion

This paper introduces the shortcomings of ordinary dividers in algorithms and their inability to perform parallel computations. It also describes three schemes for high-performance dividers. High-performance dividers can promote the application of new technologies. For example, in new computing systems such as Ternary Optical Computers (TOC), designing efficient parallel MSD (Modified Signed Digit) dividers is of great significance for improving the operational efficiency of big data division and promoting the application of TOC in the field of numerical computation. This

DOI: 10.54254/2755-2721/121/2025.19846

shows that the research on high-performance dividers helps to promote the development and application of new technologies.

In the future, the development trend of high-performance dividers will be more diversified. For example, with the growing demand for approximate computing technologies in the field of digital signal processing, the research on high-performance dividers will continue to deepen. Future divider designs will focus more on reducing energy consumption and improving computational efficiency while maintaining or improving computational accuracy. Secondly, high-performance dividers will be applied in more high-tech fields, such as artificial intelligence, big data analysis, and the Internet of Things. These applications will bring new growth points to the divider industry.

References

[1] *Wang, W. (2023). Design of 32-bit SoC based on ARMv7 instruction set and optimization of divider. Electronic Information.*

[2] *Yuan, J. (2018). Design and performance improvement of high-performance divider based on SRT algorithm. Software Engineering Professional Degree.*

[3] *Fan, H., & He, J. (2017). Design and evaluation of integer divider based on upper and lower approximation algorithm. Computer Hardware Technology.*

[4] *Cui, L., & Li, G. (2015). Improved Goldschmidt double-precision floating-point divider. Computer Hardware Technology.*

[5] *Zhao, J. (2008). Research and design of high-performance specialized FPGA arithmetic components. Computer Application Technology.*

[6] *Yao, T., & Xu, Z. (2024). A non-exact trust region algorithm for non-convex-strongly concave min-max problems. Mathematics.*

[7] *Zhu, J., & Shen, H. (2007). Implementation of high-performance single-precision divider. Microelectronics and Computer.*

[8] *Li, L. (2013). Research on divider algorithms based on FPGA. Science and Technology Information.*

[9] *He, T., Peng, Y., & Lei, Y. (2015). Design of high-performance double-precision floating-point divider based on Goldschmidt algorithm. Computer Hardware Technology.*

[10] *Wang, C., Zhang, K., Liu, K., & Yu, M. (2011). Research on high-performance double-precision floating-point divider. Computer Hardware Technology.*

[11] *Ma, H. (2016). Design and FPGA implementation of lightweight arithmetic operation circuit units. Electronics and Communications Engineering.*