Development Optimization and Future Direction of CRC Checking Technology

Aiwei Lei[1*], Ye Yu[2]

[1]College of Electrical and Information Engineering, Zhengzhou University, Zhengzhou, 450001, China
[2] College of Computer and Information Engineering, Nanjing Tech University, Nanjing, 210000, China
* law@stu.zzu.edu.cn

Abstract. Cyclic Redundancy Check (CRC) is widely used for error detection in communications. As data transfer rates increase, traditional serial CRC algorithms face performance bottlenecks, making it difficult to meet high-speed transmission needs. To address this, researchers have turned to parallel CRC computation, which enhances speed and throughput by processing multiple bits simultaneously, benefiting applications like high-speed communication protocols and storage systems. This paper introduces CRC fundamentals and explains the working principles of serial CRC algorithms, including optimization strategies for variable length and bit width. However, serial methods have limitations in speed and throughput. Thus, the paper explores the evolution of parallel CRC algorithms, their applications, and a detailed comparison of their advantages and disadvantages. Despite improving efficiency, parallel CRC algorithms still face challenges in resource-constrained environments. The paper examines these challenges and discusses the trade-offs between performance and resource consumption in high-speed, large-data scenarios. Finally, it highlights the need for flexible and adaptive hardware designs and suggests future research directions.

Keywords: Cyclic Redundancy Check, Algorithm Optimization, Digital Communication

1 Introduction

In modern digital communication and data storage systems, there are many interfering factors in the reading, writing and transmission of data. Data in the transmission or reading and writing process, is very susceptible to external magnetic fields, voltage fluctuations, and self-instability of the communication equipment, which leads to the occurrence of bit errors.To ensure the reliability of data transmission and the integrity of information, the system usually employs certain check codes to detect and even correct errors. Currently, although system stability can be improved by optimizing the circuit architecture design [1], this tends to greatly increase the design difficulty and significantly increase the hardware cost. Therefore, the most common approach is to

J. Iqbal (ed.), *Proceedings of the 2025 2nd International Conference on Mechanics, Electronics Engineering and Automation (ICMEEA 2025)*, Advances in Engineering Research 272,
https://doi.org/10.2991/978-94-6463-821-9_76

encode the data to detect and localize errors by adding a small amount of redundant information for automatic error correction.

In data communication, there are many common error detection and correction techniques like parity check, sum check, Gray's code, simple XOR cipher, and CRC code. Take the parity check method as an example, this detection method can only determine the parity based on the number of "1" in the binary number [2], which is not able to detect complex situations with a low success rate. CRC has a miss rate of only 0.0047% [3]. It has been widely adopted in practical applications due to its excellent error-detection capability [4].

With the increasing data transfer rate, for example, in application scenarios such as USB 2.0 and high-speed Ethernet, the traditional serial CRC algorithm is no longer fast enough to meet the demand. In order to improve the real-time performance and processing power of the system, accelerating CRC computation through parallel algorithms has become particularly urgent [1]. In addition, in recent years, new error correction codes (e.g., low-density parity-check codes and Polar codes) have been widely applied in areas such as digital television terrestrial transmission by virtue of their superior performance in harsh environments [5]. These techniques mainly focus on error correction of data. However, the hardware implementation of error detection of data, especially based on CRC, still suffers from insufficient parallelization and limited adaptability.

Therefore, it is of great significance to study the serial and parallel algorithms for CRC and analyze their respective advantages, shortcomings, and applications in different scenarios, which will promote the advancement of digital communication and data storage technologies. This paper will review the principles, characteristics and current status of the application of serial and parallel algorithms for CRC and provide an outlook on their future development trends.

2 The Principle of CRC

The basic principle of CRC is to detect possible transmission errors by appending redundant information consisting of check bits during data transmission [6]. Specifically, the sender first treats the sequence of data bits to be transmitted as a polynomial *M(x)*, and then performs a division operation according to a generating polynomial *G(x)* pre-agreed upon by the sender and the receiver to obtain a remainder polynomial *R(x)*. This remainder is appended to the original data to form a new sequence of code words to be sent [7]. The receiver checks the received data by using the same generating polynomial *G(x)*.If the residual polynomial computed is the same as the residual received, the data is transmitted without error. Otherwise, it indicates that an error has occurred and a retransmission is required [6].If the highest power of *G(x)* is x^k, the formula for the residual polynomial *R(x)* can be written as $\frac{M(x)\cdot x^k}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$ (1).

In error control theory, this process is known as the application of "generating polynomials" [6]. It defines the check rules for the data, which determines the precision and length of the CRC check. Generating polynomials has a key role in different types

of CRC checks, affecting the accuracy and applicability of the checks. Table 1 shows the different kinds of CRC check and their corresponding generating polynomials and application are

Table 1. common types of CRC [3][6]

CRC type	Generating polynomials	Application areas
CRC-4	$x^4 + x + 1$	Low-latency and efficient calibration for short-duration data transmission such as Bluetooth and in sensor networks
CRC-8	$x^8 + x^2 + x + 1$	Low power consumption, small devices, embedded systems, etc., applied to device communication or storage inspection
CRC-16	$x^{16} + x^{15} + x^2 + 1$	Areas such as data storage, transport protocols, etc., typically such as disk drivers and communication protocols (e.g., X.25)
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$	Mainly used for data integrity checking in telecommunication and wide area network communication protocols
CRC-32	$x^{32} + x^{26} + x^{23} + \cdots + x^2 + x + 1$	Commonly used in data compression tools (e.g. ZIP, ARJ), file system storage (GIF, TIFF, etc.) and network protocols
CRC-32C	$x^{32} + x^{28} + x^{27} + \cdots + x^8 + x^6 + 1$	Designed for fast network applications, suitable for efficient data compression and transfer, common to storage devices and file transfer protocols

CRC is widely used in the field of data communication and storage, especially in high-speed data transmission systems, such as USB 2.0, high-speed Ethernet, and so on.It can effectively detect bit errors due to external interference (e.g. electromagnetic waves, noise, magnetic field fluctuations, etc.).In addition, CRC technology is widely used in storage devices (e.g., SSD, RAID) and real-time data streaming (e.g., video streaming, audio streaming) to ensure the correctness of data and the stability of transmission.

3 Principles and Optimization of Serial CRC Algorithm

3.1 Traditional Serial CRC Algorithm

The LFSR (Linear Feedback Shift Register) is one of the most fundamental tools in the implementation of the serial method of CRC.The LFSR forms a predictable sequence by performing an XOR operation on specific bits in the register and shifting them each clock cycle.The structure of an LFSR usually consists of a number of D flip-flops and an exclusive OR gate as shown in Fig.1. Its characteristics are determined by a combination of the number of register stages, initial state, feedback logic and clock period [8].In CRC, the feedback logic of the LFSR can be used to generate the check code, which is an efficient and simple hardware implementation.

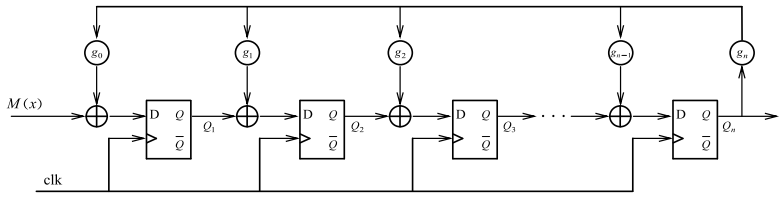

Fig. 1. Structure of n-level LFSR

In Fig.1, $g_0, g_1, g_2, \cdots g_n$ are the feedback coefficient and only be 0 or 1. $Q_1, Q_2, Q_3, \cdots Q_n$ show the output of the LFSR.When the value of g_i is 1, it means that Q_i participates in the feedback.

When the value of g_i is 0, it means that g_i is not involved in the feedback.Where g_0 and g_n must be 1, allowing output Q_n to continuously participate in the feedback.M(x) is the input code letter polynomial.The generating polynomial G(x) can be defined as : $G(x) = g_n x^n + g_{n-1} x^{n-1} + \cdots + g_2 x^2 + g_1 x^1 + g_0$.

From a mathematical perspective, the circuit functions visually as a division circuit.The formula for the remaining data R(x) in the register can be obtained : $R(x) = \frac{M(x)}{G(x)}$ (2).

From the above operating principle of LSFR and the formula (1) and (2), the circuit for CRC check code can be designed.Taking CRC-16 as an example, the corresponding

generating polynomial is : $G(x) = x^{16} + x^{15} + x^2 + 1$.Referring to the meanings of the parts in Fig.1, it is only necessary to let $n = 16, g_{16} = 1, g_{15} = 1, g_2 = 1, g_0 = 1$ the rest are 0.The structure of the serial CRC check circuit can be obtained as shown in Fig 2.

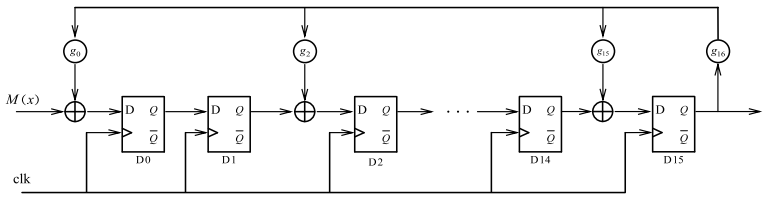

Fig. 2. Structure of serial CRC circuit for CRC-16(Picture credit : Original)

3.2 Optimization of Serial CRC Algorithm (Method1: Variable Length)

Fig.1 and Fig.2 show the structure of the basic serial CRC checking circuit, but the functionality is very limited. The circuit structure needs to be adapted to different generating polynomials and different input polynomials. Therefore, in order to satisfy different CRC standards (e.g., CRC-4, CRC-8, and CRC-32, etc.), literature [9] proposes to optimize LFSR-based serial CRC circuits (as shown in Fig 3). This design enables up to 32-bit generating polynomials and introduces a multiplexer (MUX32). Flexible configuration of the CRC generating polynomial bits can be achieved by using the order control signals order [4:0] to select specific bits in the 32-bit registers to adjust the feedback path.

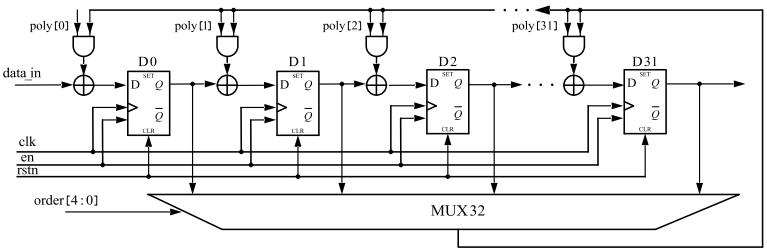

Fig. 3. Optimized diagram of serial CRC circuit (variable length) [9]. (Note: *en* is the enable signal, active high; *rstn* is the asynchronous reset signal, active low; *data_in* is the input code signal; *poly[31:0]* is the binary coefficient corresponding to each term of the generating polynomial)

3.3 Optimization of Serial CRC

As can be seen from the hardware circuits of the conventional serial computing method (Fig.1 and Fig.2), the core components of the circuits are registers and XOR gates. The differences in data for different bit widths are mainly in the number of registers, the number of bits in the XOR gate input and the way to handle the final value to be

checked. If one wants to design a circuit that can handle data of every bit width, although traditional computational methods can achieve functionality, they inevitably lead to a waste of hardware resources. To solve this problem, literature [10] proposes a bit-width adjustable serial computation method that integrates check codes with different bit-widths into a whole. Coded check operations for inputs of different bit widths are realized by controlling the channel with a Moore state machine. This design flow is shown in Fig.4.

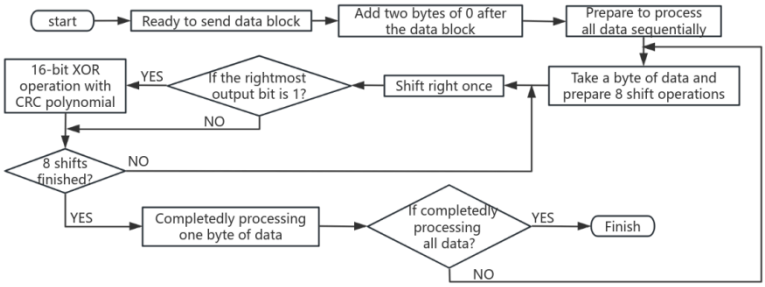

Fig. 4. Optimization algorithm for serial CRC (variable input bit width) [10]

Moore state machines can flexibly control the number of shift and XOR operations according to the bit-width of the input data, thus realizing CRC calculations with different bit-widths. Each state of a Moore state machine corresponds to a specific mode of operation or behavior, and the output is determined only by the current state, independent of the input signal [11]. Therefore, the output behavior can be changed by changing the state. The definition of states is associated with bit widths, and each state can be defined as a mode of operation that handles a specific bit width. For example: state S0: processing 8-bit data; state S1: processing 16-bit data; state S2: processing 32-bit data. Different states are obtained based on different inputs so that you can automatically jump to different bit-width processing states.

4 Basic CRC Parallel Algorithm and its Optimization

4.1 The evolution of different parallel crc algorithms

Due to the fact that serial CRC can only process 1 bit of data at a time, it not only has a slow calculation speed and low throughput, but also struggles to cope with high-speed communication and large data width scenarios. To meet the demands of modern high-speed data transmission and real-time processing, parallel CRC check codes have emerged. By processing multiple bits of data simultaneously, parallel CRC significantly enhances the calculation speed and throughput, and can fully utilize hardware resources. It is suitable for high-speed communication protocols (such as Ethernet, and PCIe), storage systems (such as SSD, RAID), and real-time data processing (such as video streams, audio streams), etc. Currently, the implementation

methods of parallel CRC calculation mainly include a traditional parallel method, split parallel method, lookup table method, matrix method, and parallel logic gate algorithm. Each method has its advantages and disadvantages and is suitable for different scenarios. The evolution of various methods is shown in Fig.5

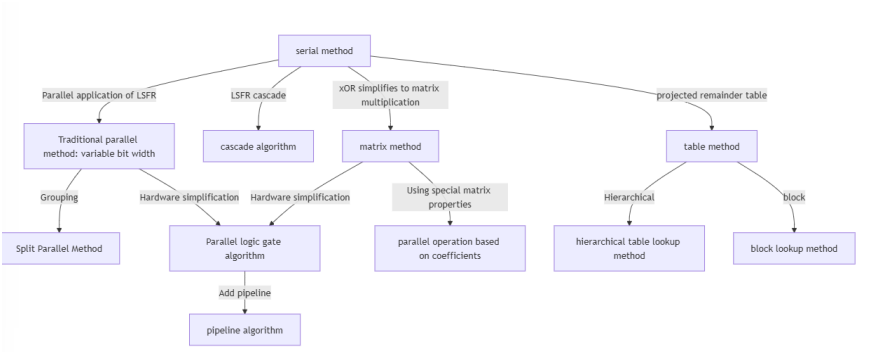

Fig. 5. Evolution of different parallel CRC algorithms(Picture credit : Original)

4.2 Requirements for algorithms in different application scenarios

Cyclic redundancy Check(CRC) algorithm is widely used in various scenarios. Different scenarios have different requirements for CRC algorithm due to its characteristics (such as data volume, real-time, power consumption, reliability, speed, accuracy, resources, etc.). Table 2 shows the requirements for CRC algorithms in typical application scenarios.

Table 2. Requirements for CRC algorithms in different application scenarios

Application scenario	Demand characteristics	Algorithm requirement
Communication system	High speed, low latency, high reliability	Parallelize CRC (e.g. table lookup, matrix method) to reduce computation time. Supports multiple CRC bit widths (such as CRC-16, CRC-32) to adapt to different communication protocols
Internet of Things and low power devices	Small amount of data, frequent transmission, limited resources High reliability, low real-time	Small computational overhead, such as a lightweight implementation of table lookup. Reduce calculation times and hardware consumption, support small bit width CRC

		to reduce the computing burden
Data storage and transmission	Large amount of data, long transmission time. High reliability, low real-time	The parallel CRC algorithm processes large chunks of data and supports high-digit CRC (such as CRC-64) to improve error detection
Industrial control	High real-time, small data volume, high1reliability.With1limited resources, performance and power consumption must be taken into account.	Table lookup method is used to reduce calculation delay.Efficient CRC computing in limited computing and storage resources
Connected cars and autonomous driving	High real-time, low latency required extremely low.Large data volume and high reliability	The parallel CRC algorithm is adopted to meet the real-time requirements. Support for high-digit CRC (such as CRC-32) to ensure data integrity
Cloud data centers and high-performance computing	Large data volume, fast transmission and processing speed. High reliability, low real-time, high efficiency	Parallelization CRC algorithm supports efficient processing of large data blocks. Supports high-digit CRC (such as CRC-64) to ensure data integrity

4.3 The advantages and disadvantages of different parallel CRC algorithms

When the traditional LFSR structure is used for serial CRC calculation, different LFSR units are calculated successively. The traditional parallel method extends it to parallel computing structure through mathematical derivation and optimization. Different LFSR structures work simultaneously to realize parallel processing of multiple data bits directly in hardware. The CRC calculation rate is improved and can be used in high-speed transmission systems. Among them, the timing problem caused by the excessively long combinational logic path of traditional parallel method limits the maximum operating frequency of the system. The complexity of design and implementation of parallel LFSR is increased, and the flexibility is limited, which is difficult to adapt to different data lengths and generate polynomials. In order to obtain the combinative optimal solution in speed, hardware consumption and flexibility, split parallel method comes into being. Split parallel computing The traditional parallel computing is divided into two parts: residual CRC computing and data CRC computing, and the residual CRC computing is divided into cascaded computing modules of

different lengths to adapt to the CRC computing problem of different effective data lengths [12]. With the continuous increase of bit width, the computing efficiency will be significantly reduced, resulting in the overall performance decline. In addition, the split parallel computing method may require more resources in hardware implementation to support complex cascading modules and parallel processing, which increases hardware cost and design complexity. Therefore, although it has certain advantages in flexibility, its applicability is limited in scenarios with high data width processing and resource constraints.

Because the traditional parallel method needs to carry out complex shift and XOR operation for each byte, the efficiency is slow, so a kind of preprocessing idea comes into being. The table lookup method calculates CRC values for all possible input bytes and stores the results in registers. The data can first be different or different from the byte just removed from the register, and the index value of the CRC residual table can be obtained through the repeated shift of high 8 bits and CRC division. The table can be looked up according to the index value, and then the table value XOR register can be used. In this way, the CRC check code [13] can be obtained directly by looking up the table during actual calculation. Therefore, the double calculation is avoided and the calculation efficiency is improved significantly compared with the serial method [14-15], however, such methods usually need to store a large number of CRC residual table entries, and the number of table entries is a quadratic power relationship with the bit width of parallel computing data. Therefore, in the case of high data width, hardware implementation requires a large number of storage units. With the evolution of technology, the table lookup method also uses many derivative methods, such as hierarchical table lookup, block table lookup and so on.

Aiming at the problem of CRC calculation of data with variable length in communication system, the classical CRC calculation implementation [16] needs to realize CRC calculation of all possible lengths. With the gradual increase of data bit width, the number of computing modules in the required CRC computing circuit increases, and the resource consumption is more serious. Literature [17-18] uses a cascaded structure for CRC calculation, which does not need to implement all length computing modules and reduces resource consumption. However, with the increase of the bit width of the input data, the method needs to increase the cascade series of computing modules, which will lead to the decrease of the operating frequency of the system. This method is suitable for the calculation of fixed bit width. However, in the communication system, the data length is usually not fixed, and this method cannot adapt to the calculation of variable data bit width [19]. Parallel CRC calculation needs to be able to flexibly change the calculated bit width

Because the shift and XOR operations of traditional parallel algorithms are troublesome, the matrix method transforms the problem of CRC calculation into matrix operations by constructing the matrix related to the generation of polynomials. The core lies in the use of matrix multiplication, addition and other operations to achieve fast data checksum processing. The matrix method is universal and expandable, and can adapt to different data lengths and generate polynomials. However, matrix method may require high computational resources and mathematical foundation when dealing with complex problems, and its efficiency and applicability may be limited for nonlinear problems or large-scale matrix operations, and it cannot complete the calculation of variable data bit width. Based on the parallel CRC formula, the parallel CRC algorithm

based on coefficient operation is derived by using special matrix properties. Compared with the matrix method, this algorithm reduces the computation amount, and has the same universality as the matrix method, and is suitable for CRC calculation with any degree of parallelism. However, the disadvantage is that it cannot meet the requirements of high speed and multiple degrees of parallelism in the specific application [20]. Literature [21] derived a method of generating logical expressions for parallel CRC calculation by constructing two matrices, which is also a common method for parallel CRC calculation in communication systems at present. Literature [22] uses rollback algorithm to transform CRC operations into matrix linear operations. Large-scale matrix operations need to consume a lot of logical resources and storage units.

In order to combine traditional parallel algorithm and matrix algorithm, hardware logic gate can be used to simplify. The parallel logic gate algorithm decomposes the CRC calculation process into multiple parallel logic gate operations, so as to realize parallel data processing. The core of this method is to use the parallelism of hardware logic gate to divide the input data into several parts, each part performs CRC calculation independently, and finally merge the results. In hardware implementation, by increasing the number of logic gates and optimizing the connection mode of logic gates, the computing speed can be significantly improved. However, this approach also has some challenges, such as as the parallel computation data bit width increases, the number of logic gates will increase, thus limiting the operating frequency of the system. In addition, in order to further optimize the performance, the step5 algorithm and the pipeline backtracking algorithm are proposed to improve the throughput by adding pipeline or segment operation.

Based on different application scenarios and different computing requirements, different computing methods should be selected. The advantages and disadvantages of different algorithms are summarized in Table 3.

Table 3. Advantages and disadvantages of different parallel computing method

Methods	Advantage	Disadvantage
Traditional serial method	Simple implementation, suitable for resource-limited scenarios. Supports arbitrary generation of polynomials and data bit width	Low computational efficiency, slow bit-by-bit processing. Not suitable for high-speed data transmission scenario
Traditional parallel method	Parallel processing data bit width, improve the computing speed Suitable for high-speed transmission systems	The combined logical path is long, causing timing problems and limiting the operating frequency. High design and implementation complexity, limited flexibility.Difficulty adapting to different data lengths and generating polynomials
Split-parallel method	Divided into residual and data CRC calculation, high	Parallelizes the CRC algorithm to process large data blocks. Support for

	flexibility, adapt to different data length calculation	high-digit CRC (for improved error detection)
Look-up table method	Reduce double counting and significantly improve efficiency Suitable for small and medium bit wide data, simple hardware.	Storage requirements increase exponentially with bit width, and hardware resource consumption is large. This parameter is not applicable in scenarios with high or wide parameters. Derived methods increase implementation complexity
Cascade method	Reduces resource consumption, eliminates the need for all length computing modules, and is suitable for the calculation of fixed bit width data	As the cascade series increases with the bit width, the operating frequency decreases. Cannot accommodate the computing requirements of variable data bit width
Matrix method	Adapt to different data lengths and generate polynomials, and use matrix operation to achieve efficient parallel computing	Large consumption of computing resources, especially in large-scale matrix operations. Limited efficiency for nonlinear problems or large-scale matrix operations. Difficult to adapt to the computing requirements of variable data bit width
The parallel operation method based on coefficient	Less computation, more efficient than matrix method Suitable for CRC computation with any degree of parallelism	Cannot meet the requirements of high speed and multiple degrees of parallelism of specific applications
Parallel logic gate algorithm	The hardware logic gate is used to improve the computing speed Suitable for resource-rich hardware implementations	The bit width increases, the logic gate series increases, and the operating frequency is limited. High resource consumption
Step 5 algorithm and pipeline backtracking algorithm	Optimize resource overhead and throughput performance with high frequency	too many flow grades, and the resource consumption is large

5 Challenge and Development

Although the parallel CRC algorithm significantly improves the computing speed, it still faces challenges in some resource-constrained scenarios. The adaptability and

efficiency of the existing methods in high and variable bit width scenarios still need to be optimized, especially in the case of variable data length in the communication system. Although the matrix method and the coefficient-based parallel arithmetic method have good performance in generality, the practical application in the high bandwidth and high speed scenario still needs improvement. Some methods, such as table lookup method and matrix method, require a lot of storage and logical resources in high and wide scenarios, which increases the hardware cost and design complexity. How to balance performance and hardware resource consumption becomes a problem that cannot be ignored. The traditional parallel method and split parallel method limit the operating frequency of the system and affect the overall performance because the combination logic path is too long or the cascade series is too many. How to find the balance between parallelization degree and performance will also be the direction of future research.

The application prospect of CRC algorithm in high-speed data transmission, low-power devices and new communication protocols is worth further discussion. With the rapid development of 5G, fiber optic communications and data center interconnection technologies, data transmission rates and bandwidth requirements are increasing exponentially. The parallel CRC algorithm combined with hardware acceleration technology can significantly improve the efficiency of error detection and meet the requirements of real-time and high reliability. For resource-constrained scenarios such as Internet of Things (IoT) sensors and wearables, the optimized CRC algorithm can effectively reduce computational complexity and power consumption, extend device life, and ensure the reliability of communication. In new communication protocols, such as 6G communication and quantum communication networks, CRC algorithms can be deeply integrated with other advanced error correction coding technologies (such as LDPC, Polar code) to build more robust and intelligent communication systems. In addition, CRC algorithms have shown significant value in other emerging areas such as edge computing, connected vehicles, and industrial iot, providing efficient and low-cost security for data transmission. In the future, with the further optimization of computing resources, algorithm efficiency and hardware architecture, CRC algorithm will continue to play an irreplaceable role in intelligent, efficient and energy-saving communication systems.

6 Conclusion

This paper describes different types of CRC algorithms, optimizes and compares various algorithms, and analyzes the application scenarios in related fields. This paper illustrates the advantages of serial CRC algorithm and its optimization methods, such as variable length and variable input bit width design, by analyzing in detail the performance enhancement.Its limitations in high speed data transmission are also pointed out.To address this problem, this paper further describes the evolution of parallel CRC algorithms and explores and compares multiple parallel CRC algorithms and their advantages and disadvantages.Also, the paper shows their potential for reducing hardware resource consumption and increasing flexibility, as well as their application in a variety of different scenarios.Although parallel CRC algorithms will

significantly improve computational speed, they still face the challenge of how to balance performance and hardware resource consumption in some resource-constrained scenarios.The research in this paper still has many shortcomings, such as failing to analyze and implement the specific code for each parallel CRC algorithm in depth, resulting in a lack of in-depth understanding of the details of the algorithms.Future research will focus on implementing each of the algorithms, especially how they can be implemented and optimized through the use of more efficient hardware.With the development of more sophisticated hardware platforms, such as FPGAs and ASICs, it will also provide more possibilities to further improve the efficiency of CRC computation.

Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

1. Yu, X.: 'Parallel Algorithm and Hardware Implementation of 32-bit CRC Check Codes', Inf. Technol., 2007, (4), pp. 71–74
2. Li, Y., Li, Z.: 'Design and Simulation of Eight-bit Parity Checker Based on Microcontroller', Sci. Technol. Innov. Inf., 2024, (13), pp. 13–16
3. Xu, W., Wang, X.: 'Application of CRC Algorithm in Computer Network Communication', Digit. Technol. Appl., 2014, (2), pp. 119–121
4. Tian, Y., Zhang, H.: 'A New Algorithm for Cyclic Redundancy Checking in UHF RFID Systems', Microcontrollers Embed. Syst., 2013, (9), pp. 1–4
5. Dan, C.: 'Application of Novel Error Correction Coding Techniques in Digital Television Terrestrial Transmission', Video Eng., 2024, (12), pp. 128–130
6. Bi, Y., Wang, Z., Luo, J., et al.: 'Design and Implementation of CRC Generic Check Test Tool', J. Lanzhou Jiaotong Univ., 2007, (3), pp. 122–124+132
7. Li, Y., Wei, W.: 'Implementation of LFSR-based CRC Check Codes on FPGAs', J. Lanzhou Jiaotong Univ., 2015, (6), pp. 91–94
8. Peng, Y., Xiao, S.: 'Implementation of FPGA-based Pseudo-random Number Generator with LFSR Structure', Digit. Technol. Appl., 2018, (3), pp. 91–92
9. Chen, X., Huang, H.: 'A CRC Circuit Design for Multiple Calibration Standards', Microprocessors, 2024, (5), pp. 50–53
10. Zhu, Z., Zhu, X., Li, B., et al.: 'A CRC Algorithm with Variable Bitwidth and Hardware Implementation', Aerosp. Control, 2019, (2), pp. 42–48
11. Ma, G., Hu, S., Yao, Z.: 'A Simulation Study of Low-power Design of Three Coded Moore State Machines', China High Technol. Enterp., 2011, (3), pp. 95–97
12. Zhang, L., Zhang, Y., Zeng, Z., et al.: 'Parallel CRC Computation and Circuit Implementation in High-speed Communication Systems', J. Xi'an Univ. Posts Telecommun., 2024, (1), pp. 71–80
13. Ji, P., Meng, D., Ren, Y.: 'FPGA-based 16bit CRC Checklist Method Design', Chin. J. Electron Devices, 2013, (4), pp. 580–584

14. Huo, Y., Li, X., Wang, W., et al.: 'High Performance Table-based Architecture for Parallel CRC Calculation', Proc. 21st IEEE Int. Workshop Local Metrop. Area Networks, 2015, pp. 1–6
15. Bajrangbali, D., Aparna Anand, P.: 'Design of High Speed CRC Algorithm for Ethernet on FPGA Using Reduced Lookup Table Algorithm', Proc. IEEE Annu. India Conf., 2016, pp. 1–6
16. Chen, X., Yu, Z., Lei, S., et al.: 'An Improved Method for 40 Gb/s Ethernet Cyclic Redundancy Check and Circuit Design', Microelectron., 2015, 45, (2), pp. 245–248
17. Li, B., Huang, Z., Shao, S., et al.: 'Implementation of CRC in 10-gigabit Ethernet Based on FPGA', Appl. Mech. Mater., 2014, 599/601, pp. 1548–1552
18. Yuan, Z., Ye, X., Guo, C.: 'Parallel CRC Design for 10G Ethernet Based on FPGA', Comput. Eng. Des., 2014, 35, (5), pp. 1510–1515
19. Kong, D., Yuan, G., Liu, X.: 'The Design and Implementation of 10 Gigabit Ethernet Link Based on FPGA', Microelectron. Comput., 2019, 36, (12), pp. 21–25
20. Zhang, X., Liang, L., Wang, Z.: 'Parallel CRC Algorithm Based on Coefficient Operation', Electron. Des. Eng., 2018, 26, (7), pp. 165–168+173
21. Stavinoha, E.: 'A Practical Parallel CRC Generation Method', Circuit Cellar, 2010, (234), pp. 38–45
22. Luo, Z., Guo, J.: 'Rollback Cyclic Redundancy Check Algorithm in High Bit-width Situations', J. Electron. Inf. Technol., 2021, 43, (4), pp. 1057–1063