Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080072

ARTICLE

BFROU: A Reconfigurable Operation Unit Design Approach Using NPN Equivalence and Reed-Muller Logic Unit for Boolean Functions in Stream Ciphers

Zhaoxu Zhou, Yanjiang Liu, Zibin Dai* and Junwei Li

Information Engineering University, Zhengzhou, China

*Corresponding Author: Zibin Dai. Email: daizb2004@126.com

Received: 02 February 2026; Accepted: 25 March 2026; Published: 15 June 2026

ABSTRACT: Stream ciphers are simple to implement and fast at encrypting and decrypting data, making them very important in information security. Boolean functions are a core part of stream ciphers. However, their mainstream hardware implementations face two main problems, including wasted area resources and excessive critical path delay. These issues limit the energy efficiency and integration level of stream cipher chips. To address these problems, this paper proposes an energy-efficient design method for a 64-bit Boolean function reconfigurable operation unit (BFROU), aiming to improve the computational efficiency of Boolean functions in stream ciphers. To optimize the design of BFROU, this paper takes the NPN equivalence theory as a guide. First, customized designs at the transistor level were performed for both 2- and 3-variable RM logic units (denoted as TRM). On this basis, this paper uses the port sharing strategy to further optimize the design of 4-to-6-variable TRM logic units and construct a multi-variable TRM process library. Then, by combining multi-variable TRM logic units with the mathematical definition of Boolean functions, this paper proposes a theoretical model of BFROU. Based on this model and combined with the statistical analysis results of Boolean functions, the optimal TRM unit configuration is determined, and the overall optimization of the 64-bit BFROU is finally completed. Experimental results show that when TRM-3 and TRM-4 units are mixed as the first-level operation module of BFROU, its area-delay product (ADP) reaches the minimum. The 64-bit BFROU unit implemented according to this scheme has an actual measured area of 137.28 μm^2 and a critical path delay of 0.278 ns under the SMIC 40 nm typical process corner. This unit supports Boolean function operations with up to 64 variables, and 94.4% of the functions can complete mapping within 2 iterations. Compared with existing schemes such as look-up table (LUT) architecture and And-Inverter Cone (AIC) array, the BFROU proposed in this paper has obvious advantages in area, delay, ADP and number of iterations, providing effective hardware support for the design of high-energy-efficiency stream cipher chips.

KEYWORDS: Boolean function reconfigurable operation unit; stream cipher; TRM units; NPN equivalence theory

1 Introduction

Stream ciphers have the characteristics of high efficiency, low delay and limited error propagation, and play an irreplaceable role in information security. Boolean functions are the basic operation components of stream ciphers, and their implementation structure largely determines the processing capacity and operation performance of stream ciphers. At present, due to the lack of dedicated functional units, RISC processors usually need to combine two-variable logic function instructions and shift instructions to complete Boolean operations, which makes such operations often require multiple clock cycles to finish. Therefore,

this paper strives to design and implement an extended unit supporting Boolean function operations in RISC processors, and realizes it by means of reconfigurable hardware. Currently, common implementation methods of reconfigurable arithmetic circuits for Boolean functions include those based on logic arrays, those based on LUT, and new implementation technologies that have emerged in recent years, such as AIC structures and RM logic units.

Programmable logic arrays implement different Boolean functions through full traversal and support more variables by increasing bit width, but this causes circuit area redundancy and increased delay. To solve this problem, researchers have proposed some optimized structures, such as RFSR [1], Alone [2], etc. However, in stream ciphers, Boolean functions usually have characteristics such as a large number of variables and significant differences in the degrees of product terms. This characteristic poses significant challenges for Boolean function unit design When implementing complex Boolean functions, more logic resources are needed, which leads to a larger unit area. When implementing simple functions, there is obvious redundancy because resources are not fully utilized. This problem of insufficient flexibility and low resource utilization greatly increases the design difficulty of the BFROU.

Hardware design based on LUT is simple in principle and has no restrictions on the form of functions, so it has become another mainstream method for implementing Boolean functions [3]. Researchers have successively proposed several LUT-based optimization schemes, such as ACLM [4], and SLM [5]. However, the high flexibility of LUT comes at the cost of increased area, delay, and power consumption. The core component of the above schemes is still the LUT unit, which does not solve the fundamental problems such as poor size scalability and the exponential growth of configuration information and area.

In summary, existing mainstream implementation technologies find it difficult to compute Boolean functions efficiently under the constraints of small area and low delay. To address this challenge, there is an urgent need to explore new implementation paths. To break the limitations of traditional structures, researchers have begun to investigate new logic architectures, with AIC structures and RM logic units becoming the focus of research. However, AIC has several issues, including high resource consumption in its interconnect structure, large area overhead for individual logic units, and immature mapping algorithms. In contrast, RM logic has gained attention because it does not face adaptation difficulties with Boolean functions in stream ciphers. Zhang et al. [6] and Zhou et al. [7], among others, have designed reconfigurable Boolean function arithmetic units based on a customization approach, achieving significant results in Boolean function implementation and providing new insights for the engineering application of RM logic. Building on previous research, this paper further deepens the adaptability design between reconfigurable hardware and Boolean function operations, and proposes an optimized architecture for reconfigurable arithmetic units targeting Boolean functions. Specifically, the core contributions of this paper include the following aspects.

- Based on transmission gate technology, the RM logic unit is optimized and designed at the transistor level. A complete TRM unit library is built based on these TRM units. The TRM unit library developed in this paper includes a complete set of files, specifically including logic library, physical library, netlist view and layout view. With these library files, TRM units can be directly integrated into backend design processes such as chip synthesis and placement and routing, providing support for the complete chip implementation work.
- This study designed a theoretical model, which is a BFROU based on TRM units. The researchers used this model to conduct optimization calculations and finally determined the optimal size of the TRM-v units inside the BFROU. This study constructed a BFROU theoretical model based on TRM units. The study used area, delay, and ADP as core indicators, and conducted a quantitative evaluation of the hardware performance of TRM units with different sizes.

- This study takes the optimal TRM-v units and the previously constructed theoretical model as the basis, and proposes a 64-bit optimized architecture of BFROU. This paper combines the optimal TRM-v units and the distribution characteristic that low-order and medium-scale product terms are dominant in Boolean functions, and finally determines the combination of TRM-3 and TRM-4 as the optimal configuration. Afterwards, based on the BFROU theoretical model, this paper proposes a BFROU architecture. This architecture not only achieves the minimum area and minimum ADP but also reaches the optimal delay performance, effectively balancing resources and performance.

The structure of this paper is organized as follows. Section 2 summarizes and analyzes the related research on BFROU. Section 3 focuses on analyzing the structural characteristics of Boolean functions, including the number of input variables and the distribution of product terms. Section 4 proposes a theoretical model of BFROU based on TRM units. Section 5 details the overall architecture of the 64-bit BFROU and elaborates on the specific design methods for TRM units. Section 6 conducts experimental verification and result analysis from multiple dimensions, including parameter comparison between TRM and Universal Reed-Muller (URM), and the mapping structure of Boolean functions. Section 7 summarizes the work of the entire paper and points out future research directions.

2 Preliminaries

2.1 Related Work

In the field of hardware implementation of Boolean functions for stream ciphers, several research teams have proposed different forms of reconfigurable computing architectures. These schemes can be broadly classified into those based on AND-XOR logic arrays, those based on LUTs, and the newly emerging implementation technologies in recent years, including the AIC structure and RM logic units.

In the hardware implementation schemes of Boolean functions based on programmable logic arrays, multiple structures have been proposed. These structures still face their respective performance bottlenecks. Ref. [1] proposes an implementation method. It uses XOR trees to process the linear part and AND-OR networks to process the nonlinear part. This method offers strong versatility but suffers from high delay, large hardware overhead, and complex configuration. To optimize performance, Ref. [8] improved this structure. It combines AND tree networks, AND arrays and XOR arrays to implement two input ports for the linear part, high-order terms and low-order terms of the nonlinear part respectively. This shortens the critical path to a certain extent. However, in this scheme, high-order terms still occupy more hardware resources. Low-order terms are restricted by the port sharing mechanism, complicating mapping operations and limiting overall applicability. Anole proposed a coarse-grained cryptographic logic array. The logic units of this array can be used to implement Boolean functions, but the calculation process requires multiple clock cycles, resulting in low efficiency. Ref. [9] designed a reconfigurable structure for symmetric cryptographic algorithms. This structure reduces wiring complexity and area through a tree-like interconnection method. It has the characteristics of low power consumption and high area utilization. It can also support most block ciphers and some stream ciphers, but the adaptability of its processing units to Boolean functions is still insufficient. In addition, the RHMCA unit proposed in Ref. [10] has mixed functions and can be directly used for Boolean function calculation, but it brings significant area overhead.

Due to their high flexibility, LUT structures are widely adopted for hardware reconfiguration of Boolean functions. However, their resource consumption scales exponentially with the number of input variables, and their capacity to support high-order term functions remains particularly constrained. To alleviate this problem, Ref. [11] proposed an AND-LUT hybrid architecture that employs AND arrays for high-fan-in logic and LUTs for low-fan-in parts, achieving a 46% area reduction compared to the LUT cascade structure

in traditional FPGAs. However, this method does not conduct special optimization for the characteristics of Boolean functions in cryptographic algorithms, leading to still unsatisfactory actual resource utilization. On the other hand, methods such as Shannon decomposition [12] are used to reduce function complexity and thus reduce the implementation area, but these methods have low efficiency when processing a single expression. The SLM structure proposed later combines input/output controllers with small-sized LUT cores to achieve partial function sharing and dynamic switching. It shows certain advantages in reducing configuration overhead and resource consumption. However, since it is still based on LUT units, this structure cannot overcome some inherent bottlenecks, such as limited number of inputs and the exponential relationship between configuration information and area. At the same time, it also has problems such as large critical path delay and limited number of supportable functions. Furthermore, ACLM attempt to combine function characteristics with hardware optimization more closely. Although they significantly improve unit utilization, their type-dependent variable retrieval mechanism also introduces additional complexity to the algorithm mapping process, which limits versatility and practical application effects.

To overcome performance bottlenecks in area and latency, academic research has increasingly turned to novel hardware technologies such as AIC-based architectures and Reed–Muller (RM) logic. Although the arithmetic unit based on the AIC structure proposed in Ref. [13] enhances architectural flexibility through multi-stage selection and output multiplexing, it remains constrained by high interconnect resource consumption and significant per-unit area overhead. In contrast, RM logic exhibits a natural algebraic alignment with cryptographic Boolean functions, demonstrating superior potential for adaptability.

Refs. [7,10] further explored the integration of RM logic with transistor-level custom design, substantially improving the implementation efficiency of Boolean functions; nonetheless, there remains scope for enhancement in structural optimization depth and resource utilization. Beyond hardware optimization targeting specific operators, the co-design of cryptographic algorithms and hardware architectures has emerged as a key research direction. For instance, the PrivLSTM framework [14] achieves privacy-preserving inference in multi-party data scenarios by integrating cryptographic techniques with neural network architectures. Such deep co-design between algorithms and hardware provides important methodological inspiration for this work in exploring the system-level integration of Boolean function units with processor architectures.

In summary, this paper proposes an optimized architecture for a low-latency, low-overhead Boolean function reconfigurable operation unit, aiming to achieve a comprehensive optimum in security, efficiency, and flexibility.

2.2 Motivation

Nowadays, the information security situation is becoming increasingly severe, and efficient and reliable cryptographic algorithms have become the cornerstone of safeguarding various security services. The execution efficiency of stream cipher algorithms directly affects the real-time protection capability and service quality of information security systems. Boolean functions are the core operation units of stream ciphers, and their hardware implementation methods directly affect the throughput and delay performance of cryptographic processing. Currently, general-purpose processors based on the RISC architecture lack dedicated Boolean function operation units, so they have to implement a single Boolean operation by combining multiple instructions. This implementation method not only significantly increases the clock cycle overhead but also struggles to meet the strict requirement of millisecond-level response in high-real-time application scenarios.

For existing hardware implementation technologies of Boolean functions, severe challenges arise at three levels. At the adaptation level, Boolean functions in stream ciphers have several typical characteristics, including a large number of input variables, significant differences in the order of product terms, and

uneven distribution of the number of terms. Traditional programmable logic arrays have the problem of insufficient structural flexibility, while LUT structures face a dilemma where their hardware resources grow exponentially with the number of input variables. At the structural efficiency level, emerging AIC structures are limited by issues such as high interconnection complexity and immature mapping algorithms. Although RM logic has formal adaptation advantages, most existing designs are implemented based on standard cell libraries and do not undergo in-depth transistor-level optimization, which prevents the unit performance from reaching the optimal state. At the engineering implementation level, existing solutions generally lack a full-chain design from circuit optimization to chip integration, and also lack key supports such as complete physical libraries and timing models. This makes it difficult for them to be actually integrated into the processor backend design process.

In response to these challenges, this study is committed to breaking through traditional design paradigms and conducting innovative work from three dimensions. Specifically, transmission gate technology is adopted to perform transistor-level optimization on TRM units, a strategy that significantly improves unit performance. This study also constructs a complete cell library, which includes logic function libraries, physical libraries, netlist views, and layout views, and can provide comprehensive support for chip backend integration. Meanwhile, this study establishes a theoretical model based on multi-objective optimization such as area and delay, and this model can provide a quantitative basis for parameter selection of reconfigurable architectures. Through these innovations, this study aims to realize a Boolean function operation architecture, which has high efficiency, flexibility, and engineering feasibility. It can provide a new hardware solution for improving stream cipher processing performance and also promote the development of high-real-time security applications.

3 Structural Characteristics Analysis of Boolean Function

In order to construct a reconfigurable Boolean operation unit with low delay and low overhead, it is necessary to systematically study the structural characteristics of Boolean functions in stream ciphers. This paper is based on 54 Boolean functions included in various stream cipher algorithms and conducts statistical analysis from multiple aspects such as the number of input variables of these functions and the distribution of product term orders. The specific statistical results are shown in Fig. 1.

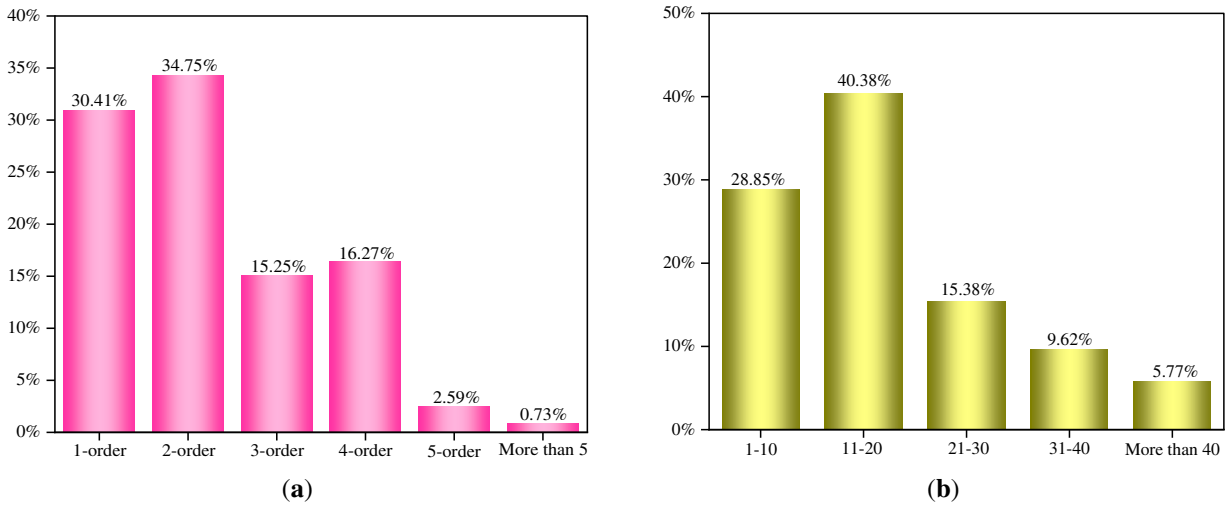


Figure 1: The statistical analysis presents the Boolean functions; (**a**) Distribution of the number of input variables; (**b**) Distribution of the number of product terms.

From the perspective of the distribution of product term orders, the average proportion of first-order product terms is approximately 30.41%, and such product terms play an important role in maintaining the balance of the function. The proportion of 1-order product terms is about 34.75%, and the variables between most 2-order product terms do not overlap with each other. This characteristic helps to achieve high nonlinearity and also conforms to the typical structural features of Bent functions. The proportion of 3-order product terms is 15.25%, and the proportion of 4-order product terms is 16.27%. 3-order and 4-order product terms can maintain high nonlinearity while improving the algebraic degree and resiliency order, thereby coordinating multiple cryptographic indicators. The proportion of 5-order product terms is approximately 2.59%, and high-order product terms with orders higher than five account for only 0.73%; moreover, such high-order terms only appear in individual algorithms. Turning to the distribution of the number of product terms, Boolean functions with 1 to 10 product terms account for 28.85%, those with 11 to 20 product terms account for 40.38%, those with 21 to 30 product terms account for 15.38%, those with 31 to 40 product terms account for 9.62%, and those with more than 40 product terms account for 5.77%. This distribution allows designers to clearly understand the application frequency of Boolean functions with different numbers of product terms in stream cipher algorithms, thereby making them more targeted in the resource allocation and architecture design of reconfigurable operation units.

In summary, the 1 to 4-order product terms in Boolean functions occupy an absolute dominant position, with their total proportion being approximately 96.68%. At the same time, the scale of the number of product terms is generally concentrated within 20 terms, and the proportion of this part is 69.23%. This structural characteristic indicates that when designing reconfigurable operation units, emphasis should be placed on optimizing the implementation efficiency of product terms of the 4-order and below. Meanwhile, the number of product terms of medium scale should be taken as the main basis for hardware resource allocation. Doing so can effectively control the delay and area of the circuit while ensuring multiple cryptographic indicators.

4 Theoretical Model of BFROU

Boolean functions are important mathematical tools in digital logic circuits. The implementation of stream cipher and block cipher systems relies on digital logic circuits, so Boolean functions occupy an important position in these cipher systems. According to the definition of Boolean functions, when a Boolean function is expressed as a polynomial, it is represented by Formula (1).

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = a_0 \oplus a_1 x_n \oplus a_2 x_{n-1} \oplus a_3 x_{n-1} x_n \oplus \cdots \oplus a_{i-1} x_{n-i+1} \cdots x_n \oplus a_{2^n-1} x_1 \cdots x_n \quad (1)$$

In Formula (1), the symbol $\oplus$ denotes the XOR operation. The coefficients $a_0, a_1, \ldots, a_{2^n-1}$ take values from the set $\{0,1\}$. If a coefficient a_i is 1, the corresponding product term is included in the XOR sum. if a_i is 0, the product term is excluded. Formula a_1 covers all possible variable combinations, ranging from the 0-order term (i.e., the constant term a_0) to the highest-order term (i.e., the product term involving all input variables, $x_1, x_2, \ldots, x_n$) . The algebraic normal form of each Boolean function is unique. Moreover, Formula (1) can be further expressed as a combination of several Boolean functions, as shown in Formula (2).

$$f(x_1, x_2, \ldots, x_i, \ldots, x_n) = f_1 \nabla f_2 \nabla \cdots \nabla f \quad (2)$$

The ∇ in Formula (2) represents the AND operation or the XOR operation. Then, in the design of the BFROU, each TRM unit in the first stage is made to process a subfunction $f_1, f_2, \ldots, f_k$ that contains the same number of input variables. At the same time, TRM units of uniform size are used to achieve this, and this design has significant advantages in multiple aspects.

First, from the perspective of hardware implementation, a unified TRM unit structure helps achieve regularized allocation and layout of resources. TRM units of the same size are convenient for uniform arrangement and wiring on the chip, which not only significantly reduces wiring complexity and interconnection delay but also helps optimize the area and timing of physical design. If TRM units have different sizes, it is easy to cause structural fragmentation and unbalanced resource utilization, which in turn increases the complexity of control logic and the overall circuit area.

Second, a unified unit structure can enhance the reconfigurability and flexibility of operation units. Boolean functions in cryptographic algorithms have diverse structures, but their algebraic normal forms can be systematically decomposed into XOR combinations of multiple subfunctions. If each TRM unit has consistent input capacity and processing capability, it can flexibly implement Boolean functions with different numbers of terms and different orders through parallel combination and cascading, thereby improving the adaptability of hardware to different cryptographic algorithms.

Moreover, this strategy is also consistent with the previous statistical analysis conclusion, that is, the number of product terms of Boolean functions is generally concentrated between 11 and 20, and product terms of the 4-order and below occupy a dominant position. Therefore, according to this distribution characteristic, the number of inputs of TRM units can be set to a moderate common scale. This not only balances implementation efficiency but also avoids over-designing the unit structure for a very small number of high-complexity functions.

In summary, using TRM units of a unified size offers two advantages. On the one hand, it helps reduce the complexity and overhead of hardware design. On the other hand, it can efficiently support most Boolean functions while maintaining reconfigurability. This design aligns with the design goal of low-delay and small-area arithmetic units for stream ciphers. Therefore, when TRM-v is the optimal partitioning unit, the overall architectural model of BFROU can be represented as shown in Fig. 2.

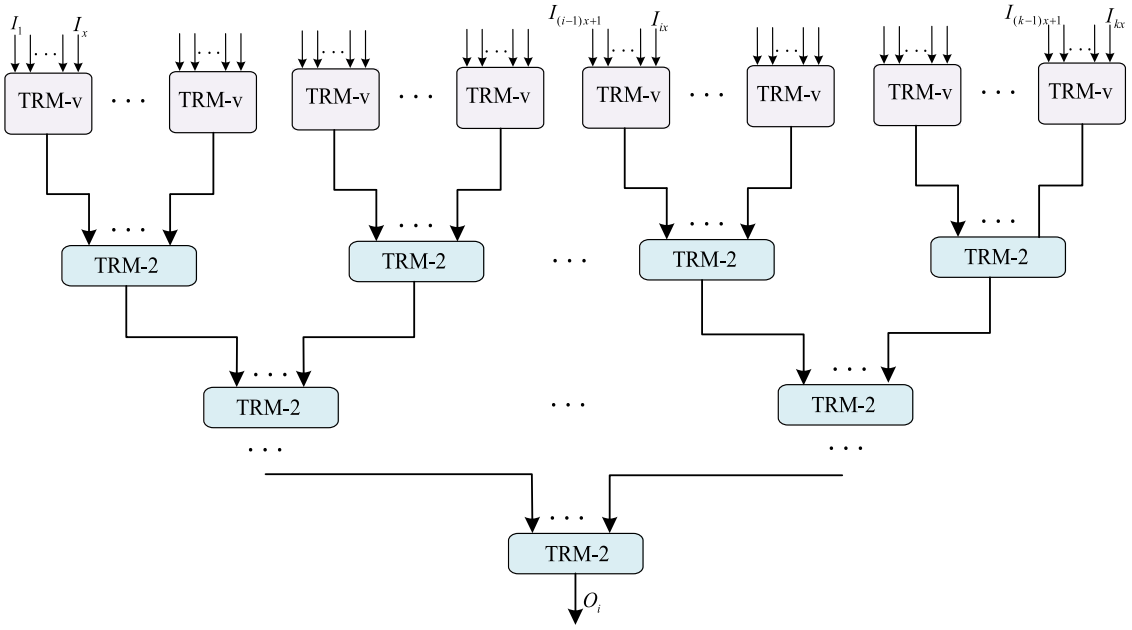


Figure 2: The theoretical model of BFROU; This model is composed of TRM-v units and TRM-2 units.

The number of input ports of a single TRM-v unit in Fig. 2 is x. When k TRM-v units form input data, the total number of input data is kx without counting the configuration information. The finally computed data is represented by O_i. To determine the optimal size v of the TRM-v unit, we need to construct a BFROU model. This model uses area, delay, and ADP as the core evaluation indicators. The model will quantify the hardware efficiency of TRM units with different sizes v when implementing the target Boolean function, and then minimize ADP through optimal selection. The definitions and descriptions of the variables involved in the model are shown in Table 1.

Table 1: Definition of variables.

Abbreviation	Meaning
ζ_i	Sets of Boolean functions for different stream ciphers
k	The number of TRM-v units
η_v	The number of unit stages consumed for constructing BFROU with different numbers of TRM-v units
v	The optimal size range of TRM units is $2 \le v \le 6$.
α_v	The area of a single TRM-v unit
β_v	The delay of a single TRM-v unit
γ_v	The total area consumed by the TRM unit to implement the Boolean function.
ξ_v	The unit delay consumed by the TRM-v unit to implement the Boolean function
χ_v	The routing delay consumed by the TRM-v unit to implement the Boolean function
φ_v	The total delay of building BFROU with the optimal size of TRM-v
l_{avg}	The average routing length between units at each level
ρ	Resistivity of conductor materials
A	Wire cross-sectional area
ε	Dielectric constant of the wire insulation layer
t	Thickness of the insulation layer
R_{drv}	The output resistance of the driving unit

To begin with, we extract the Boolean functions contained in the stream cipher that needs to be implemented. These functions include key stream generation functions and feedback functions. After extraction, a set ζ_i is formed, and this set is represented by Formula (3).

$$\zeta_i = \{F_1, F_2, \ldots, F_n\} \tag{3}$$

Then, we sequentially divide the Boolean functions in set ζ_i into functions f_i according to Formula (2). These functions f_i can all be implemented using TRM-v units, and the number of TRM-v units can be represented by k. According to Formulas (1) and (2) in this paper, when the number of input variables is n, the Boolean function needs to be divided and implemented in hardware using k TRM-v units, and the specific value of k is represented by Formula (4).

$$\begin{cases} k \le \left\lfloor \dfrac{C_n}{v} \right\rfloor \\ k \ge \left\lceil \dfrac{n}{v} \right\rceil \end{cases} \tag{4}$$

Finally, we combine k TRM-v units with customized TRM-2 units. The data output after combination is the final calculation result of the Boolean function f_i. Different TRM-v units spend a certain number of unit stages when implementing different Boolean functions, and this number of unit stages is represented by η_v in Formula (5).

$$\eta_v = \lceil \log_2 k \rceil + 1 \tag{5}$$

When different TRM-v units implement Boolean functions, the number of TRM-2 units required is also different. When the number of TRM-v units is k, the consumed quantity of TRM-2 units is $k-1$. For the BFROU built with k TRM-v units, the area resources it consumes can be represented by λ_v, and specifically by Formula (6).

$$\lambda_v = (\alpha_v \times k) + \alpha_2 \times (k-1) \tag{6}$$

For the BFROU built with k TRM-v units, the unit delay it incurs can be represented by ξ_v, and specifically by Formula (7).

$$\xi_v = \beta_v + \beta_2 \times \lceil \log_2 k \rceil \tag{7}$$

During the placement and routing process, calculating the routing delay χ_v requires combining the unit stage count η, the physical characteristics of wires, and process parameters. The core source of this routing delay is the resistance-capacitance delay characteristic of wires. The following is a routing delay calculation method based on a physical model, and this calculation method is specifically represented by Formula (8).

$$\chi_v = \eta \times \left(\rho \times \frac{l_{avg}}{A}\right) \times \left(\varepsilon \times \frac{w \times l_{avg}}{t}\right) + R_{drv} \times \left(\varepsilon \times \frac{w}{t} \times l_{avg} \times \eta\right) \tag{8}$$

From Formulas (7) and (8), it can be concluded that when TRM-v serves as the optimal division unit, the actual delay φ_v is represented by Formula (9).

$$\varphi_v = \xi_v + \chi_v \tag{9}$$

According to the aforementioned Formulas (6) to (9), we can accurately calculate the ADP value when TRM-v maps to a certain Boolean function. Therefore, through the above analysis, it can be concluded that Formula (10) is the mathematical model for calculating ADP. We can calculate the optimal division unit size based on this model.

$$\begin{cases} ADP_{\min} = \lambda_v \times \varphi_v \\ \left\lfloor \frac{n}{v} \right\rfloor \le k \le \left\lfloor \frac{C_n^v}{v} \right\rfloor \\ 2 \le v \le 6 \end{cases} \tag{10}$$

Through this model, we calculate and conclude that when v = 3, the area consumption is the smallest, and the value of ADP is also the smallest. When v = 4, the delay is minimized, and the ADP is slightly larger than that of the former. Therefore, by comprehensively considering the order distribution and quantity distribution characteristics of the product terms of Boolean functions, as well as the analysis results of the performance model centered on ADP, it can be known that the proportion of product terms of the first to fourth orders in Boolean functions exceeds 96%, and the number of product terms of nearly 70% of the functions is concentrated within 20 terms. This situation determines that the reconfigurable operation unit needs to focus on covering the product term logic of low order and medium scale. Considering the quantitative optimization results of ADP, when a unit combination that includes both TRM-3 and TRM-4 is adopted, it can not only realize the minimization of area consumption and ADP with the help of TRM-3 units, but also achieve the optimal performance of delay through TRM-4 units. In this way, better effects can be achieved in terms of cryptographic index guarantee and the balance between circuit delay and area, providing efficient reconfigurable hardware support for the Boolean function operation of stream ciphers.

Based on the above model analysis and configuration optimization results, subsequent work will focus on translating these theoretically optimal unit combinations into concrete hardware architectures. Section 5 elaborates on the overall design of the BFROU based on this strategy and its integration scheme within the processor, aiming to realize the mapping from theoretical performance to practical circuitry.

5 Design of BFROU

As stated in Section 4, the hybrid configuration of TRM-3 and TRM-4 is determined. The BFROU is subsequently integrated into the RISC processor as an independent computational module and operates in parallel with the ALU. The unit requires no internal configuration registers, as control signals are directly generated from instruction decoding results and operands, thereby enabling low-latency flexible configuration. The RISC processor adopts a dual-source operand architecture, in which operands are fetched from the general-purpose register file and the computation result is written back to the destination register [15]. Given that the combined width of the two source operands is 64 bits, this paper designs a nonlinear Boolean function computation component featuring 64 input terminals. This design ensures data path alignment between the computation unit and the register file, avoiding the overhead of data concatenation and splitting that would otherwise arise from bit-width mismatch [16].

5.1 The Overall Framework of 64-BIT BFROU

As mentioned above, 3rd-order and lower product terms account for 80.41%, and 4th-order and lower ones 96.68%. This statistical result indicates that it is appropriate to adopt a mixed method of TRM-3 units and TRM-4 units during the Boolean function mapping process. This mixed method can significantly improve the adaptability and coverage of hardware resources, meeting the implementation needs of most product terms. Based on the constraint of 64 input ports, this paper adopts three types of units, which are TRM-4 units sharing two ports, TRM-4 units sharing one port, and TRM-3 units, and configures them in a ratio of 2:2:4. These three types of units occupy a total of 60 input ports. These TRM-4 units have dual implementation capabilities; they can be used as complete 4-variable units to implement 4-input Boolean functions, and can also be split into independent TRM-3 units to implement 3-variable Boolean functions respectively. This dual capability greatly enhances the flexibility and reusability of hardware resources. This mixed architecture makes full use of the high efficiency of TRM-3 units in processing low-order product terms, while combining the good support of TRM-4 units for high-order product terms, and finally achieves a balance between resource efficiency and flexibility. The remaining 4 input ports are reserved for expanding specific high-efficiency product terms. The overall hardware architecture of this design is shown in Fig. 3.

The BFROU in Fig. 3 adopts a hierarchical structure design. In each level of the structure, a redundancy strategy and an error checking mechanism are introduced. This is done to comprehensively improve the reliability and fault-tolerance performance of the operation unit. The first-level structure of the BFROU is composed of two types of TRM-4 units and TRM-3 units, specifically including two dual-port-sharing TRM-4 units, two single-port-sharing TRM-4 units, and four TRM-3 units. These units all have the ability to implement any 3-variable Boolean function, and can also handle specific Boolean functions containing up to 5 variables. In the first-level structure, through the redundantly deployed TRM-3 units, the detection and isolation of calculation errors are realized by comparing the output results of multiple units. In this way, when a fault occurs, the fault can be identified and localized in a timely manner.

The second layer of the BFROU theoretically contains six TRM-2 units. This type of TRM-2 unit can support the operation of Boolean functions with no more than 11 variables. The output part of the third layer further expands the processing capability to support Boolean functions with up to 23 variables. By arranging

redundant units at multiple levels, the system not only enhances the adaptability to multi-variable functions, but also significantly improves the efficiency of error detection and fault isolation.

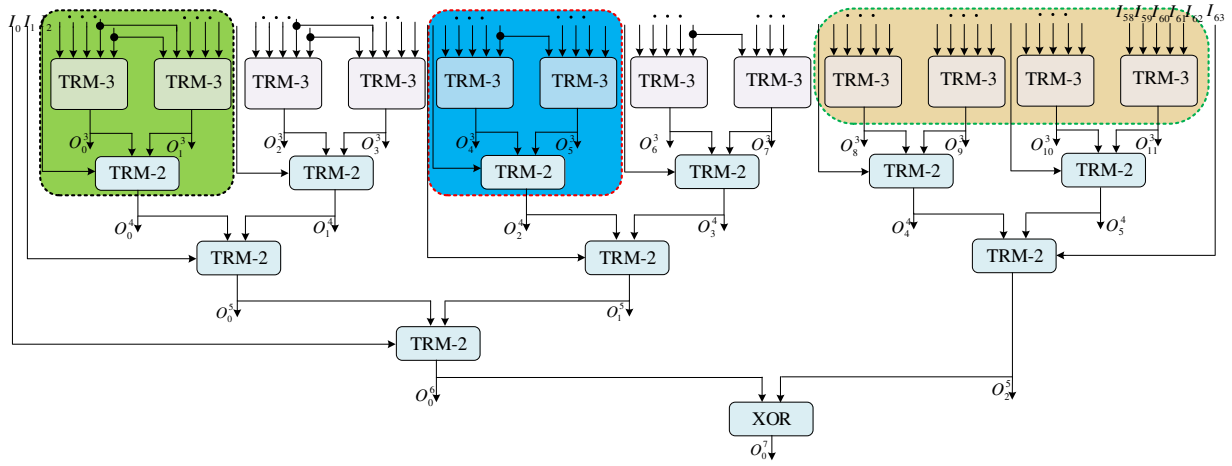


Figure 3: 64-bit BFROU; The green part represents the TRM units sharing two ports; The blue part represents the TRM units sharing one port; The yellow part represents four TRM-3 units.

Finally, the XOR unit located at the output stage has processing capability, and it can handle Boolean functions with up to 64 variables. While providing strong computational flexibility, this XOR unit can also effectively suppress the risk of fault propagation and bring more reliable fault-tolerance guarantee to the system. In general, relying on its own multi-level redundancy design and distributed error handling mechanism, BFROU exhibits excellent stability and anti-interference capability in practical applications. This structure of BFROU can ensure that when hardware faults or external interferences occur, the system can still maintain correct operation. In this way, the reliability of the overall operation architecture is significantly enhanced.

In the BFROU circuit shown in Fig. 4, the variables used for operation and the corresponding configuration information occupy a total of 64-bit. The output result of this circuit is 23-bit, and these output results occupy the upper 23-bit of the target register. The circuit has multi-level and multi-granularity Boolean function processing capabilities, and also has the following computing capabilities.

(1) Highest complexity coverage: The O_0^7 port can implement the vast majority of 7-variable Boolean functions. Meanwhile, it breaks through the conventional limitations and supports special Boolean functions with up to 64 variables and a maximum degree of 6, providing hardware support for the operation of ultra-large-scale and high-degree Boolean functions.

(2) Full-featured 6-variable coverage: The O_0^6 port is capable of realizing any 6-variable Boolean function, and at the same time are compatible with a special Boolean function that involves up to 41 variables, achieving dead-end coverage of functionality in 6-variable scenarios.

(3) Flexible operation of 5 variables: Ports such as O_i^5 can implement any 5-variable Boolean function and also support special Boolean functions containing up to 23 variables, taking into account both general and special scenario requirements.

(4) Efficient implementation of 4-variable functions: The O_i^4 port focuses on implementing arbitrary 4-variable Boolean functions and can simultaneously handle special Boolean functions with up to 11 variables, ensuring computational efficiency in the 4-variable domain.

(5) Three-variable basic support: The O_i^3 port is used to implement any three-variable Boolean function and also supports 12 special Boolean functions of up to five variables, which serve as the basic operation units for scenarios with a small number of variables.

In summary, the proposed design combining layered coverage with port sharing achieves full-scenario adaptability for Boolean functions ranging from three to seven variables, while effectively optimizing area and latency performance through hardware resource reuse, thereby providing an efficient and flexible reconfigurable hardware solution for diverse Boolean function operations in domains such as stream ciphers. Specifically, compared to the CRM-unit-based design in Ref. [7], whose 64-bit architecture contains only 11 CRM-3 units, the proposed 64-bit BFROU incorporates 12 TRM-3 units through port sharing and unit reuse optimization. This increase in unit count not only verifies the area efficiency of the proposed design but also enables the hybrid architecture to achieve more comprehensive coverage of low-degree product term logic, thereby further improving latency performance.

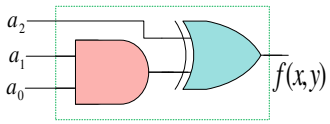


Figure 4: URM-2 logic unit; This unit can implement any two-variable Boolean function in combination with the NPN equivalence class theory.

5.2 Design and Implementation of the TRM Unit Based on Transmission Gates

In the field of reconfigurable hardware, the position change of input ports can be conveniently realized by adjusting the external routing configuration of logic units. The inversion operation can also be completed by adding a small amount of circuits. Therefore, the application of NPN equivalence classes can significantly improve the resource utilization rate of reconfigurable logic units [17,18]. With the help of Polya theory, we can derive the number of NPN equivalence classes, and the lower bound of this number is given by Formula (11).

$$C(K) \geq \frac{2^{2^K}}{K! \times 2^K \times 2} \tag{11}$$

Formula (11) shows that for functions in NPN equivalence classes, there are at most $K! \cdot 2^{2^K} \cdot 2$ different input permutations, output permutations, and inversion combinations. The number of all possible functions is 2^{2^K}. Therefore, the number of equivalence classes is not lower than the ratio of the upper limit to the total number of functions.

By referring to the performance characteristics of k-LUTs in FPGAs, we can know that their optimal energy efficiency and area-delay balance point usually lies in the range from 4-LUT to 6-LUT [17]. Based on this conclusion, this paper focuses on the TRM-v series units. Among them, v represents the core logic dimension of the unit, and we are committed to carrying out customized optimization design at the transistor level for these units. It can be seen from Formulas (4)–(9) that the number of NPN equivalent functions that can be realized by TRM-2 and TRM-3 units is relatively small, which facilitates functional completeness verification. The number of NPN equivalence classes corresponding to the TRM-4 unit increases slightly, but its functional completeness can still be effectively verified through algorithms. In contrast, the number of NPN equivalence classes covered by TRM-5 and TRM-6 units is hundreds or even thousands of times that of the previous several units, which brings great challenges to the verification work.

Therefore, to construct a transistor-level customized TRM cell library, this paper conducts transistor-level customized design for TRM-2 and TRM-3 cells. For the TRM-4 cell, this paper optimizes it at the circuit level based on the TRM-2 and TRM-3 cells. For the TRM-5 and TRM-6 cells, this paper further constructs them based on the TRM-4 cell. This is done to control the design complexity while ensuring functional coverage

5.2.1 Design of TRM-2 Unit

Under the principle of NPN equivalence classification, two-variable Boolean functions can be divided into four equivalence types: constant, single-variable, x_1x_2, and $x_1 \oplus x_2$. Building on RM logic, this paper proposes a general unit structure capable of implementing any two-variable Boolean function. The unit is composed of AND gates and XOR gates. The input signals a_0 and a_1 can be configured as 0, 1, x_1 and $\overline{x_1}$, and a_2 can be configured as x_2. This AND-XOR circuit constructed based on standard cells is named URM-2.

The following is a further analysis of the algebraic relationship of the three-input AND-XOR operation.

$$Z = AB \oplus C \tag{12}$$

The truth table of Formula (12) is listed in Table 2.

Table 2: Truth table of Formula (12).

A	B	C	Z
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0

By conducting an in-depth analysis of the truth table shown in Table 2, we can find that when the input signal C is logic 0, the output Z equals AB, which is the logical AND of A and B. When C is 1, the value of Z is equal to $\overline{AB}$. Based on this Boolean relationship, we can summarize the operation logic described in Formula (12), which can further guide the transistor-level circuit to implement the corresponding functional module.

Fig. 5 shows the transistor-level circuit structure of the TRM-2 unit proposed in this paper. The circuit is mainly composed of two parts. First, CMOS transistors m1 to m4 form a NAND gate structure of $\overline{AB}$. Then, CMOS transistors m9 to m12 form two transmission gates, namely TG1 and TG2, which are used for gating output between $\overline{AB}$ and AB, thereby jointly realizing the composite logic function of the AND-XOR type. Specifically, this can be expressed by Formulas (11) and (12).

When C is 0, the transmission gate TG is turned on, and the output Z is $\overline{AB}$, which corresponds to the expression of Formula (13).

$$AB \oplus C = \overline{AB} \oplus 0 = \overline{AB} \tag{13}$$

When C is 1, the transmission gate TG is turned on, and the output Z is AB, which corresponds to the expression of Formula (14).

$$AB \oplus C = \overline{AB} \oplus 1 = AB \tag{14}$$

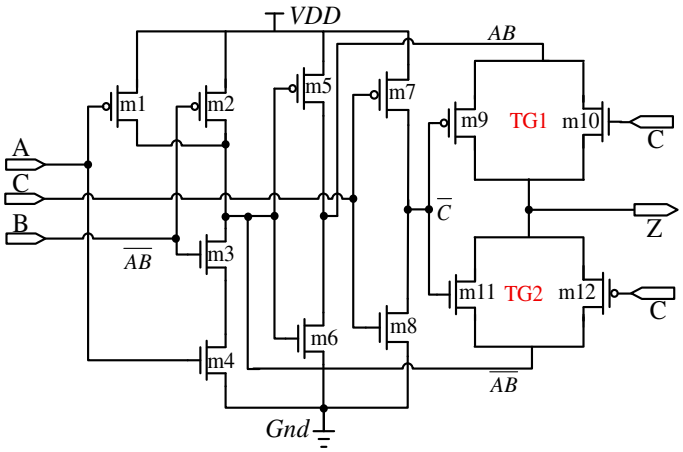
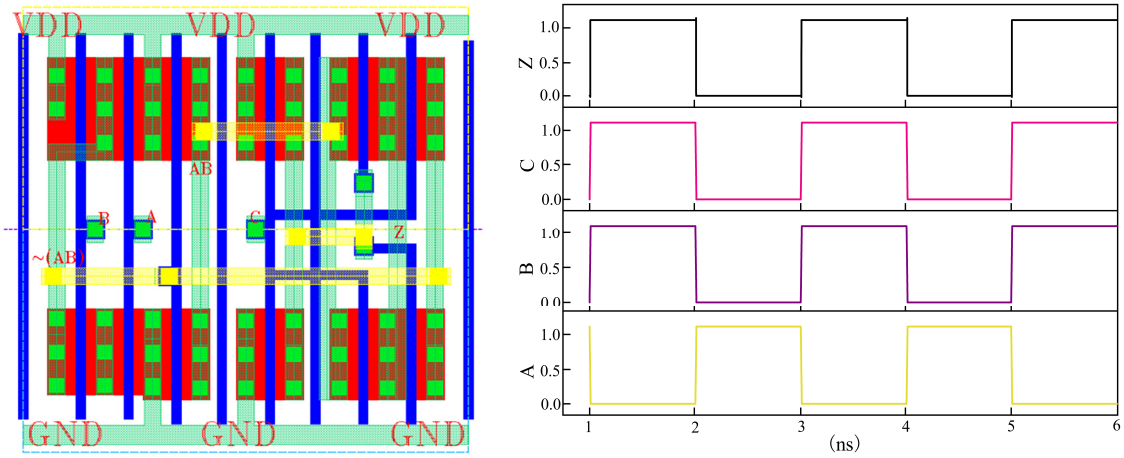


Figure 5: Transistor-level circuit structure diagram of the TRM-2 unit.

In summary, with the help of the algebraic characteristics described in Formulas (12)–(14), this paper improves the traditional circuit structure. During the improvement process, a transmission gate architecture is adopted to realize signal selection and logic generation, and finally a transistor-level customized AND-XOR circuit is proposed, which is TRM-2. While improving logical compatibility, this design also optimizes the utilization efficiency of transistors and has good performance and scalability.

After completing the theoretical design of the transistor-level circuit for the TRM-2 unit, it is necessary to use EDA tools such as Virtuoso to perform library characterization (K library) on it to generate standard cell library files for logic synthesis and place-and-route. This process mainly includes several steps. We first perform functional simulation on the transistor-level circuit shown in Fig. 5 to verify the correctness of its logical function. After obtaining the waveform results that meet expectations, we systematically adjust the width-to-length ratio of the transistors to optimize the timing performance of the circuit and finally determine the transistor size configuration that minimizes the unit delay. Next, based on the determined optimal transistor size, we draw the physical layout of the circuit. Specifically, Fig. 6 shows the layout of TRM-2. During the layout design of Fig. 6, layout methods such as matching alignment, device sharing, and topology optimization need to be adopted to compress the unit area as much as possible, thereby improving the overall integration and manufacturing economy.

After completing the layout design, it is necessary to perform parasitic parameter extraction to obtain the resistance and capacitance information of the wires. After that, using K library tools (such as SiliconSmart), a large number of SPICE simulation netlists are automatically generated under conditions such as different process corners, input transition time, and output load capacitance. Then, large-scale simulations are carried out to collect data such as timing and power consumption. After the simulation is completed, the tool will process and model the massive raw data. Finally, a timing library file described in the form of a lookup table is generated, providing a key reusable cell model for the subsequent digital integrated circuit design process.



(**a**) (**b**)

Figure 6: TRM-2 logic unit: (**a**) basic layout; (**b**) simulation waveform.

Fig. 6a clearly marks the three input ports (A, B, C) and one output port (Z) of the TRM-2 unit. At the same time, Fig. 6 also specifies the corresponding nodes of these ports in the transistor-level circuit shown in Fig. 5. To ensure that the TRM-2 unit can meet the requirements of large-scale chip integration, before designing this unit, the parameter specifications of the SMIC 40 nm standard cell library were first analyzed. Taking the typical two-input AND gate unit AND2X2_12TL as an example, its 12T indicates that the unit height is equal to the width of 12 M1 layer metal tracks. According to the definition of this process library, the pitch of the M1 layer metal track is 0.14 μm, so the uniform height of the standard cell is determined to be 1.68 μm. When designing the layout of the TRM-2 unit shown in Fig. 6a, to meet the requirements of large-scale integration for layout regularity and coordinated adaptation with surrounding standard cells, the size of this unit strictly follows the specifications of the standard cell library. The height of the TRM-2 is consistent with that of the standard cell, set to 1.68 μm, to ensure accurate alignment with other units during layout. Its width is set to an integer multiple of the minimum layout unit in the process library, and the site is the basic unit to ensure the uniformity of the layout. In the SMIC40 nm process, the width of a single site is 0.09 μm, so the layout width of the TRM-2 is determined to be 1.89 μm.

Following the completion of the layout design, functional simulation verification was performed on the TRM-2 unit. Fig. 6b presents the transient simulation waveforms under various combinations of input signals A, B, and C. The simulation results clearly demonstrate that output Z strictly adheres to the logic expression $Z = AB \oplus C$, with the output waveforms matching the expected truth table across all input combinations, thereby validating the functional correctness of the designed transistor-level circuit.

In summary, the final layout area of the TRM-2 unit is the product of its height and width, and this area is equal to 3.1752 μm^2. This size not only conforms to the integration specifications of the standard cell library but also provides good compatibility for the overall place-and-route of subsequent chips.

5.2.2 Design of TRM-3 Unit

In the design process of the TRM-3 logic gate, using NPN equivalence classes for classification is an effective method. According to the content in Table 3, when the number of variables is three, there are 14 types of NPN equivalence classes in total. If we further exclude constant functions and classes that only depend on 1 or 2 variables, we can obtain NPN equivalence classes that truly involve three variables, and there are 10 such

equivalence classes in total. These classes represent the typical structures of all three-variable functions, and their specific forms can be uniformly expressed through Formula (15). Therefore, this classification method not only helps to systematically understand the functional characteristics of the TRM-3 logic gate but also provides a clear theoretical basis for subsequent logic optimization and circuit design [19–21].

$$\begin{aligned} f &= a_3 \oplus a_4 a_1 \oplus a_4 a_2 \oplus a_5 a_1 a_2 \\ &= (a_3 \oplus a_4 a_1) \oplus a_2 (a_4 \oplus a_5 a_1) \end{aligned} \tag{15}$$

Table 3: Ten typical three-variable NPN equivalent class functions implemented based on Formula (15).

	Typical Functions of Three-Variable NPN Equivalence Classes	Typical Function Adaptation Scheme	a_1	a_2	a_3	a_4	a_5
1	$f_{mux} = z \oplus xy \oplus xz$	$f = (x \oplus yz) \oplus x\,(y \oplus 0{*}z)$	y	z	z	x	0
2	$f_{\text{xorand}} = xy \oplus xz$	$f = (0 \oplus xy) \oplus z\,(x \oplus 0{*}z)$	y	z	0	x	0
3	$f_{\text{onehot}} = x \oplus y \oplus z \oplus xyz$	$f = (x \oplus 1{*}y) \oplus z\,(1 \oplus xy)$	y	z	x	1	x
4	$f_{maj} = xy \oplus xz \oplus yz$	$f = (0 \oplus xy) \oplus z\,(x \oplus 1{*}y)$	y	z	0	x	1
5	$f_{\text{orand}} = x \oplus xyz$	$f = (x \oplus 0{*}y) \oplus z\,(0 \oplus xy)$	y	z	x	0	x
6	$f_{xor3} = x \oplus y \oplus z$	$f = (x \oplus 1{*}y) \oplus z\,(1 \oplus 0{*}y)$	y	z	x	1	0
7	$f_{dot} = x \oplus y \oplus xyz$	$f = (x \oplus yy) \oplus z\,(y \oplus \bar{x}y)$	y	z	x	y	$\bar{x}$
8	$f_{\text{andxor}} = x \oplus yz$	$f = (x \oplus 0{*}y) \oplus z\,(0 \oplus 1{*}y)$	y	z	x	0	1
9	$f_{and3} = xyz$	$f = (0 \oplus 0{*}y) \oplus z\,(0 \oplus xy)$	y	z	0	0	x
10	$f_{\text{camble}} = x \oplus xy \oplus xz \oplus yz$	$f = (x \oplus xy) \oplus z\,(x \oplus 1{*}y)$	y	z	x	x	1

If the two ports of the TRM-3 unit are fixed, that is, $a_1 = y$ and $a_2 = z$. The remaining ports serve as the input ports for the residual variables. Then, there are input schemes as shown in Table 3, where all variables can take the original phase or the inverse phase.

The TRM-3 unit is mainly composed of AND gates and XOR gates, and its structure is shown in Fig. 7. This circuit has five input ports from a_1 to a_5. Each input can be configured as a fixed logic value of 0 or 1, and can also be connected to the original-phase signals or inverted-phase signals of independent variables x, y, and z. By flexibly setting the connection mode of the inputs, this gate circuit can realize all three-variable logic functions.

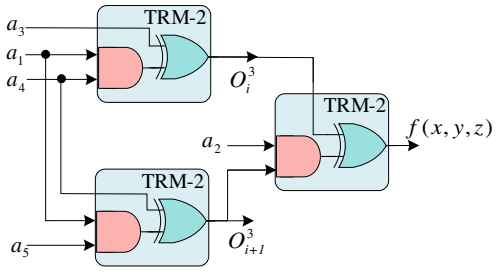


Figure 7: TRM-3 unit; this unit is composed of three TRM-2 units.

Then, the TRM-3 unit is built based on the transistor-level circuit of the TRM-2 unit. After that, EDA tools such as Virtuoso are used to perform library characterization on the TRM-3 unit to generate standard cell library files for logic synthesis and place-and-route. The specific process of this procedure is consistent

with the process of performing K library for the TRM-2 unit. The specific layout of the TRM-3 unit is shown in Fig. 8.

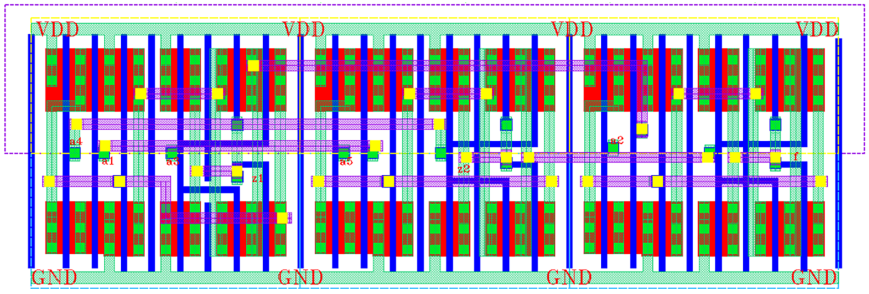


Figure 8: Layout of the TRM-3 unit.

Fig. 7 shows the circuit structure diagram of the TRM-3 unit, and Fig. 8 shows its corresponding layout. Figs. 7 and 8 match each other in circuit logic and physical layout, achieving precise correspondence between logical functions and physical implementation. From the port connections, the layout in Fig. 8 includes five input ports from the circuit in Fig. 7, which are a1, a_2, a_3, a_4, and a_5. The final output is f, and this output corresponds to the logical function represented by Formula (13). Additionally, the layout in Fig. 8 has two intermediate outputs: $O_i^3 = (a_1 \& a_4) \oplus a_3$, and $O_{i+1}^3 = (a_1 \& a_5) \oplus a_4$. These intermediate outputs correspond to the intermediate signal nodes in the circuit of Fig. 7. In terms of layout customization specifications, the TRM-3 unit is consistent with the TRM-2 unit. The height is uniformly set to 1.68 μm, and the width corresponds to 63 times the site size, calculated as 5.67 μm^2, which ensures compatibility with the layout requirements of the standard cell library.

Fig. 9 presents the layout-level simulation waveforms of the TRM-3 unit, with the function $f_{and3} = xyz$ employed as a case study for functional verification. The simulation results demonstrate that, under all combinations of the input variables x, y, and z, the output waveforms consistently match the expected logic values, thereby validating the functional correctness of the designed TRM-3 unit after physical implementation.

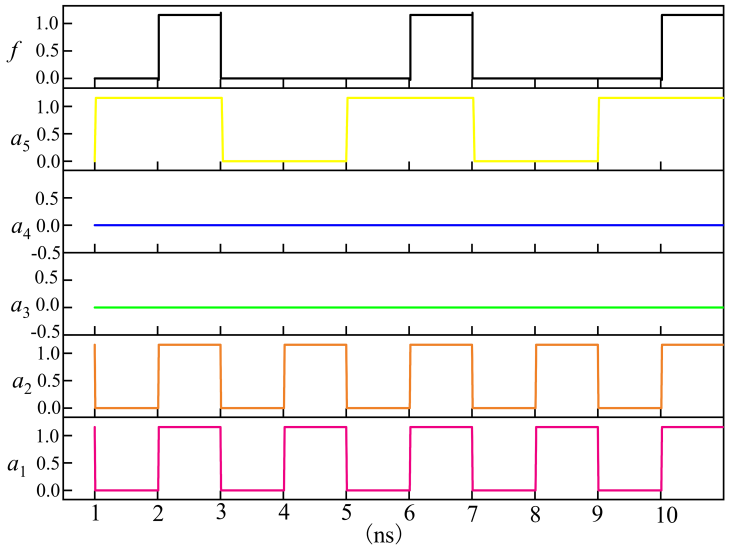


Figure 9: Layout-level simulation waveforms of the TRM-3 unit.

5.3 The Expansion Method for Multi-Variables/High-Terms of TRM

In the design of multi-variable TRM units, the transformation operation of Boolean functions can be carried out with the help of the Shannon decomposition theorem. According to the Shannon decomposition theorem, for a Boolean function $F(x_1, x_2, ..., x_n)$ with n variables, there is always a relationship as shown in Formula (16) for any variable x_i $(i = 1, 2, 3, ..., n)$ among them.

$$F(x_1, x_2, \ldots, x_n) = x_i f(x_1, x_{i-1}, x_{i+1}, \ldots, x_n) \oplus g(x_1, x_{i-1}, x_{i+1}, \ldots, x_n) \tag{16}$$

If a multi-variable TRM unit is to be implemented, it can be constructed in a combined manner according to Formula (16). This stepwise optimization strategy not only ensures the design consistency of units with different dimensions but also can fully adapt to their respective logical complexities, and finally provides a hierarchical and efficient solution for the reconfigurable hardware of stream cipher Boolean functions.

5.3.1 Design of TRM-4 Unit without Port Sharing

The design of the TRM-4 unit allows the application of Formula (16) to the split set with a split granularity of 4, which is specifically expressed by Formula (17).

$$F(x_1, x_2, x_3, x_4) = x_i f(x_1, x_{i-1}, x_{i+1}, \ldots, x_4) \oplus g(x_1, x_{i-1}, x_{i+1}, \ldots, x_4) \tag{17}$$

In the split set, the number of variables in both sub-functions f and g does not exceed 3. Therefore, the function $F(x_1, x_2, x_3, x_4)$ can be implemented based on the TRM-3 unit. The specific implementation structure is shown in Fig. 10.

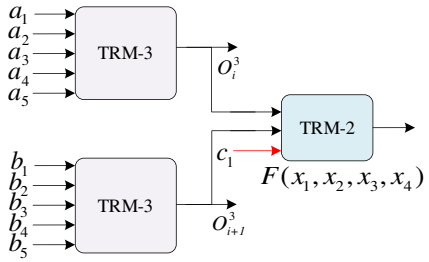


Figure 10: TRM-4 logic unit. This unit is implemented by two independent TRM-3 units without sharing their input ports.

Both TRM-3 units in Fig. 10 can implement any three-variable Boolean function. At this time, it is necessary to calculate 10 inputs a_1, a_2, a_3, a_4, a_5 and b_1, b_2, b_3, b_4, b_5 according to the Boolean function expression. Each input can be configured as a fixed logic value of 0 or 1, or connected to the original-phase or inverted-phase signals of four variables. Here, c_1 belongs to $\{0, 1, x_4, \overline{x_4}\}$. Based on the analysis of the circuit in Fig. 10, the logic circuit uses a total of 11-bit configuration information and operation variables. It has three types of programmable computing functions.

In terms of four-variable Boolean function generation, when c_1 is set to x_4, and a_i takes values from the set $\{0, 1, x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}$, and b_j takes values from the set $\{0, 1, x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}$, this circuit can implement any four-variable Boolean function from x_1 to x_4. At this time, the output function can be expressed by Formula (18).

$$F(x_1, x_2, x_3, x_4) = x_4 f(x_1, x_2, x_3) \oplus g(x_1, x_2, x_3) \tag{18}$$

In terms of XOR combination of dual three-variable functions, if c_1 is set to 1, and a_i takes values from the set $\{0, 1, x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}$, and b_j takes values from the set $\{0, 1, x_4, \overline{x}_4, x_5, \overline{x}_5, x_6, \overline{x}_6\}$, then this circuit can implement the XOR operation of two independent three-variable Boolean functions $f(x_1, x_2, x_3)$ and $g(x_1, x_2, x_3)$. The corresponding output is expressed by Formula (19).

$$F = 1 \cdot f(x_1, x_2, x_3) \oplus g(x_4, x_5, x_6) \tag{19}$$

In terms of independent three-variable function output, when c_1 is set to 0, and a_i is specified to take values from the set $\{0, 1, x_1, \overline{x}_1, x_2, \overline{x}_2, x_3, \overline{x}_3\}$, and b_j takes values from the set $\{0, 1, x_4, \overline{x}_4, x_5, \overline{x}_5, x_6, \overline{x}_6\}$, the circuit can output a three-variable function $g(x_4, x_5, x_6)$ separately. At the same time, another output port O_i^3 can provide the function $f(x_1, x_2, x_3)$, which is expressed by Formula (21), and O_{i+1}^3 can be used for other logic requirements. The overall output relationship of F is expressed by Formula (20).

$$F = 0 \cdot f(x_1, x_2, x_3) \oplus g(x_4, x_5, x_6) = g(x_4, x_5, x_6) \tag{20}$$

$$z_2 = f(x_1, x_2, x_3) \tag{21}$$

Through flexible input configuration, this circuit realizes efficient support for multiple combinational logic functions, demonstrating strong universality and reconfigurability.

5.3.2 Design of TRM-4 Unit Based on Port Sharing Strategy

From Table 3, it can be seen that when two ports are fixed, the TRM-3 can still express any three-input Boolean function. However, these two ports must have two common variables with the same polarity, and these common variables are input from ports a1 and a2. Therefore, by sharing two ports of two TRM-3 units, the circuit structure of the TRM-4 unit shown in Fig. 8 is designed. This unit has 9 input ports and 3 output port, named TRM-4-9. Moreover, the TRM-4-9 unit can implement any four-variable Boolean function.

From the structural diagram of the TRM-4-9 logic unit in Fig. 11, it can be deduced that the expression of any four-variable Boolean function can be represented by Formula (22).

$$F_4 = c_6 \left[(a_3 \oplus x_4 a_1) \oplus a_2 (a_4 \oplus a_5 a_1) \right] \oplus \left[(b_1 \oplus b_2 a_1) \oplus a_2 (b_2 \oplus b_3 a_1) \right] \tag{22}$$

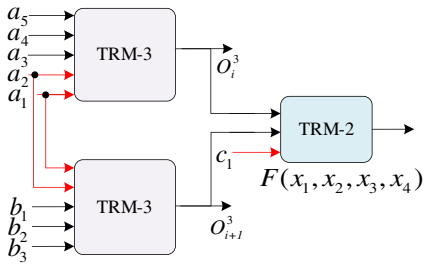


Figure 11: Logic unit of TRM-4-9; this unit is implemented by sharing the input ports of two TRM-3 units.

To further investigate the minimum port requirement for a TRM-4 unit in implementing arbitrary 4-variable Boolean functions, this paper examines whether its configurability can be further reduced to eight ports, given that nine ports are already known to be sufficient for implementing all functions. Under the premise that hardware circuit latency remains unchanged, reducing the number of input ports can effectively decrease configuration information bit-width, simplify interconnect complexity between units, and improve overall chip integration density as well as placement and routing efficiency, thereby achieving resource

overhead optimization without sacrificing timing performance. Such an optimization holds significant importance for resource-constrained hardware designs. To address this problem, we propose a spatial search algorithm for Boolean function adaptability verification, which is used to systematically evaluate whether the simplest expressions of the 222 NPN equivalence classes of 4-variable Boolean functions can be implemented using an eight-port TRM-4 unit.

This spatial search algorithm is used to verify whether the 8-port TRM-4 unit can implement all four-variable Boolean functions. The algorithm selects 222 representative functions of NPN equivalence classes as the target set F_k. Based on the Boolean expression F_{4-9} represented by Formula (20), F_{4-9} is simplified into an 8-input function F_{4-8} by constraining any two input variables to be equal. Subsequently, all value combinations of each variable in F_{4-8} within the set $\{0, 1, a, b, c, d, \overline{a}, \overline{b}, \overline{c}, \overline{d}\}$ are traversed to generate all possible 8-port configurations, which are then matched with each target function. If there is a function in F_k that cannot be implemented by any configuration, the function is added to the failure set, and the program is terminated. If the failure set is non-empty in the end, it is determined that TRM-4-8 cannot implement all four-variable Boolean functions. The pseudocode of the spatial search algorithm is as Algorithm 1 follows.

Algorithm 1: Determine the feasibility of the 8-port TRM-4 unit

```
Input:
F4-9, Fk
Output:
    ζ              //Set of non-adaptable functions Fk
Procedure Space-search(F4-9, Fk, max_cutting_size)
1.   for (k = 1, k < 222, k++) do
2.     for i = 1:9 do
3.       for (j = i + 1, j =< 9, j++) do
4.         if (i ≠ j && j ≤ 9) then
5.         F4-8 ← F4-9 (xi = xj)
6.           if [config(F4-8, Fk) == 1] then
7.           print "Successfully configured"
8.           else
9.           ζ ← Fk
10.            if (|ζ| ≠ ø) then
11.            print "Configuration failed"
12.            break
13.            end if
14.          end if
15.        end if
16.      end for
17.    end for
18. end for
```

The pseudocode systematically verifies the implementation ability of the 8-port TRM-4 unit for all 222 typical functions under all NPN equivalences of 4 variables through three layers of nested loops, variable constraint operations, and configuration verification. In the first line of the code, the outermost loop iterates through the 222 target functions F_k to ensure that each Boolean function to be verified is covered. The middle and inner loops in lines 2–4 iterate through any two different variables in the 9-input function F_{4-9}. By applying the constraint $x_i = x_j$, the 9 ports are reduced to 8 ports, simulating all possible port reduction

scenarios. In line 5, the 9-input F_{4-9} is transformed into an 8-input function F_{4-8} through variable constraint, completing the conversion of the number of ports. Lines 6–14 call the config function to verify whether F_{4-8} can be configured as the current target function F_k. If the adaptation is successful, a success message is printed. Otherwise, F_k is added to the non-adaptable set. Then, it is determined whether set ζ is empty. If set ζ is non-empty, the code terminates, and the conclusion is drawn that the 8-port TRM-4 unit does not have the ability to implement all 4-variable Boolean functions. If set ζ is an empty set, the execution continues. The entire algorithm traverses the entire space of target functions and variable constraint combinations and combines the configuration verification logic to ensure the comprehensiveness and rigor of the conclusion, providing a systematic verification method for the functional feasibility of the TRM-4 unit.

The final results indicate that, for the logical function relationship described by Formula (23), the TRM-4-8 logic unit fails to achieve correct adaptation. This function cannot be fully expressed or effectively implemented under the current 8-port configuration, suggesting that the TRM-4-8 structure has limitations in logical expression ability.

$$f = abcd \oplus abc \oplus ab \oplus ac \oplus bd \oplus cd \oplus a \oplus d \tag{23}$$

Therefore, it can be concluded that to implement any four-variable Boolean function, the TRM-4 unit requires at least 9 input ports. This conclusion further limits the functional coverage of such programmable logic units under resource constraints from the structural perspective, and provides an important theoretical basis for subsequent unit design and adaptation strategies. Specifically, when a structure has 9 ports and can implement all target functions, the redundant configuration capability brought by adding 1 more port will not weaken its original functions. This redundant configuration capability can also provide greater flexibility and fault tolerance for circuit implementation. We named this structure with 10 input ports and 3 output port as TRM-4-10. The specific form of this structure is shown in Fig. 12.

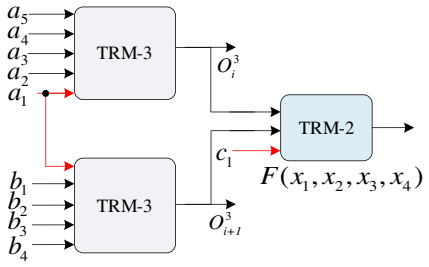


Figure 12: TRM-4-10 logic unit; this unit is implemented through the design of sharing one input port.

5.3.3 Design of TRM-5 and TRM-6 Units

The design of the TRM-5 unit enables the application of Formula (16) to the split set with a split granularity of 5, which is specifically expressed by Formula (24).

$$F(x_1, x_2, x_3, x_4, x_5) = x_i \cdot f(x_1, x_{i-1}, x_{i+1}, \ldots, x_5) \oplus g(x_1, x_{i-1}, x_{i+1}, \ldots, x_5) \tag{24}$$

In the split set, the number of variables of subfunction f and subfunction g both does not exceed 4. Therefore, function $F(x_1, x_2, x_3, x_4, x_5)$ can be implemented with the help of the TRM-4-9 unit, and the specific implementation structure is shown in Fig. 11. To reduce the steps of unit function verification, this structure uses two independent TRM-3 units to implement subfunctions in parallel, and these two units do not share input ports, so as to maintain the independence of logical paths.

Correspondingly, the design of the TRM-6 unit follows the same idea, and it is composed of two TRM-5 units and one TRM-2 unit. However, as the unit scale expands, the number of ports increases significantly. Among them, the TRM-5 unit requires 19 ports, and the number of ports of the TRM-6 unit reaches 39. Such a large number of ports will cause issues such as a sharp increase in interconnection complexity, difficulty in place-and-route, and excessive area overhead in the layout, and these issues are not conducive to the application of such units in large-scale integrated circuits. Therefore, this study does not conduct layout-level custom design for the TRM-5 unit and the TRM-6 unit, but focuses on the TRM-2 to TRM-4 units with fewer ports and more compact structures, so as to achieve a reasonable balance between implementation complexity and hardware efficiency

6 Experiment

6.1 Comparative Analysis of Unit Performance Parameters

To validate the performance of the proposed TRM unit, transistor-level circuit design and simulation verification were completed based on the SMIC 40 nm CMOS process under a standard temperature of 25°C and a supply voltage of 1.1 V. In terms of transistor count, the proposed TRM-2 unit employs a transmission gate structure, utilizing only 12 transistors to realize the logic function Z = AB⊕C. In contrast, implementing the same function using conventional AND and XOR gates in a standard cell library typically requires approximately 16 transistors. Thus, the TRM-2 unit achieves a reduction in transistor count of about 25%. The performance evaluation took area, delay, ADP, and power consumption as key indicators, and conducted a comparative analysis with existing implementation schemes such as CRM, URM, K-LUT, and AND-XOR logic arrays. Simulation results show that the TRM unit exhibits significant advantages in all indicators. Specifically, the area of the TRM-2 unit is only 3.1752 μm^2, and its area is optimized by 8.6%, 26.5%, 66.9%, and 69.9% respectively compared with the CRM, URM, K-LUT, and AND-XOR structures. In terms of timing performance, the propagation delay of the TRM-2 unit is 37 ps, which is optimized by 17.78%, 36.21%, 69.17%, and 62.24% respectively compared with the above-mentioned comparative structures. This result fully proves that the TRM unit has higher hardware efficiency and operation speed when realizing the same logic function.

To verify the scalability of the TRM unit, this study further designed multi-variable TRM units such as TRM-4, TRM-5, and TRM-6. These units are all optimized and expanded based on the core structure of TRM-2 or TRM-3, so they can still maintain excellent area and delay characteristics in more complex logic implementations. As shown in Fig. 13, as the number of variables increases from 2 to 6, the TRM unit still maintains obvious advantages in terms of delay, area, and ADP.

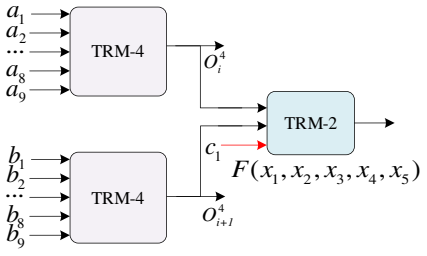


Figure 13: TRM-5 logic unit; This unit is implemented by two independent TRM-4 units and one TRM-2 unit, and its input ports are not shared.

As shown in Fig. 14a, when implementing 2-to-6-variable Boolean functions, the two implementation methods of AND-XOR logic array and K-LUT have the largest hardware resource overhead in terms of

area and delay, and this result is consistent with the theoretical analysis in the previous section. In addition, as the number of variables increases, the delay and area of all logic units show an upward trend. Among them, the TRM unit proposed in this paper maintains the lowest resource consumption within the entire variable range, demonstrating excellent scalability. Further analysis of Fig. 14b shows that the ADP value increases monotonically with the increase in the number of variables, which is consistent with the trend reflected in Fig. 14a. This increase is due to two reasons: on the one hand, the expansion of the unit area directly increases the value of ADP. On the other hand, the increase in the number of variables increases the complexity of the logical traversal phase, which in turn increases the delay and further increases the value of ADP. Based on the comprehensive evaluation results of ADP, it can be concluded that the TRM unit designed in this paper is superior to implementation methods such as CRM, URM, K-LUT, and AND-XOR within the entire variable range, thus verifying the advantage of the TRM unit in comprehensive performance at the system level.

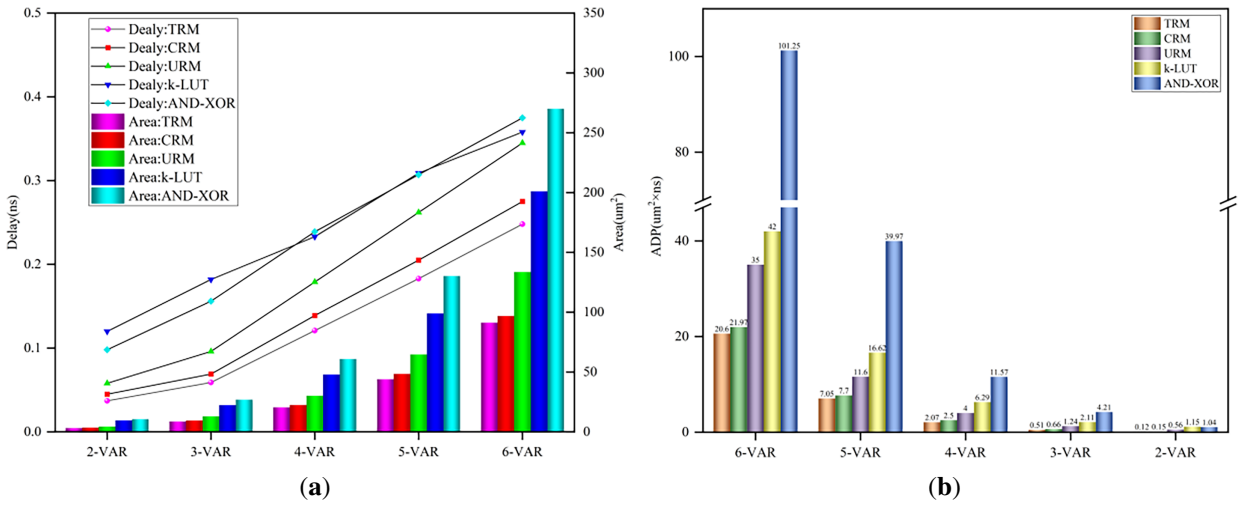


Figure 14: Comparative analysis of the performance of multivariable logic units; (**a**) Parameter comparison of area and delay; (**b**) Parameter comparison of ADP.

In terms of power consumption characteristics, the TRM units also show significant energy efficiency advantages. Experimental data shows that the total power consumption of the TRM-2 unit is 0.56 μW, which is optimized by 27.3% and 22.2% respectively compared with the CRM unit and URM unit. The total power consumption of the TRM-3 unit is 1.57 μW, which optimizes the energy efficiency by 26.6% and 33.2% respectively compared with these two comparative units. This result further proves the effectiveness of the TRM architecture in circuit structure optimization, indicating that the TRM units not only have advantages in speed and area but also perform well in power consumption control.

In summary, the TRM unit proposed in this paper is superior to existing mainstream implementation schemes in multiple key indicators such as area, delay, ADP, and power consumption, showing comprehensive performance advantages. This result verifies the effectiveness and advancement of the TRM unit in the design of low-overhead logic units from the experimental level, and provides valuable reference for the design of low-power integrated circuits in the future.

6.2 Experimental Result Analysis of BFROU

Under the SMIC 40 nm process at 25°C and 1.1 V supply voltage, this paper presents a comprehensive evaluation of the proposed 64-bit BFROU. The evaluation is based on timing library files generated after

library characterization and K library generation, with synthesis performed using Design Compiler. The results reflect the evaluation status at the synthesis stage based on an actual cell library. The results show that the unit achieves an area of 137.28 μm^2, a delay of 0.278 ns, and a total power consumption of 28.75 µW. To validate the performance of the BFROU in implementing various Boolean functions, this study investigates seven different operation units: ACLM, AIC, Yang, RHMCA, TNFSR, and CRM_BF. To ensure fairness and comparability of the experimental results, this paper adopts the standardized evaluation methodology proposed in Ref. [22] to uniformly normalize the eight selected Boolean function operation units. Specifically, all units are mapped to the same process technology node, and their processing bit-widths are uniformly scaled to 64-bit to mitigate the impact of bit-width differences on area, timing, and energy efficiency metrics.

The comparison results are shown in Fig. 15. Fig. 15a shows the performance of each unit in terms of area and delay, and Fig. 15b shows the comparison of ADP. It can be clearly seen from the figures that the 64-bit BFROU proposed in this paper is significantly superior to other comparison schemes in the ADP indicator. This indicates that the unit has obvious advantages in the comprehensive balance of area and delay, and further verifies the efficiency and practicality of this architecture.

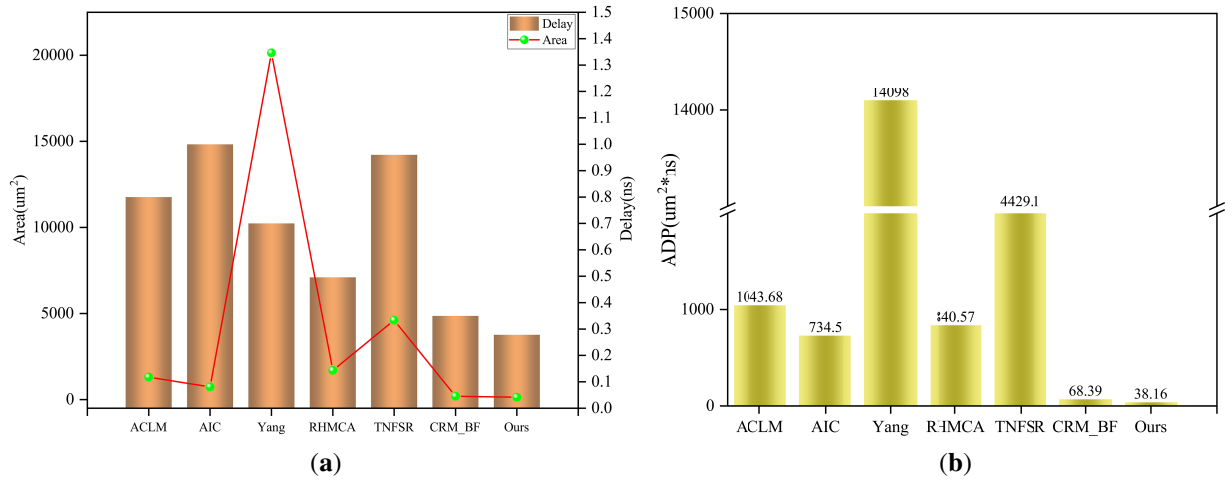
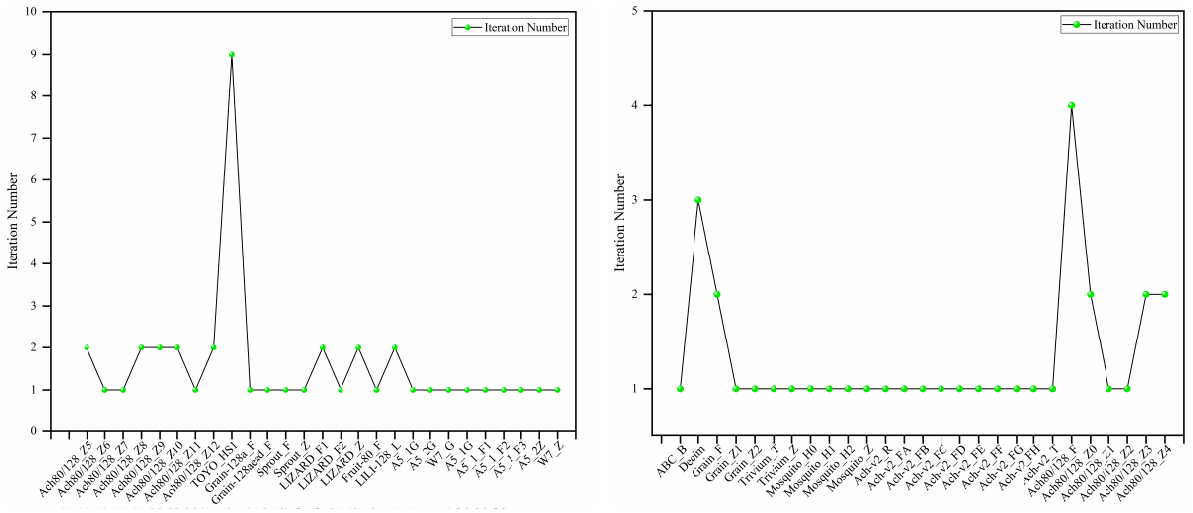


Figure 15: Comparative study on key performance indicators of different reconfigurable computing units. (**a**) Performance of area and delay; (**b**) ADP indicator reflecting the comprehensive efficiency of area-delay.

6.3 Performance Evaluation of BFROU Mapping

The mapping experiments in this paper are based on the implementation capability of the three-variable NPN equivalence classes described in Table 3 and Formula (15). The ten types of typical three-variable Boolean functions listed in Table 3 can all be uniformly implemented using the structure shown in Formula (15), which constitutes the basic mapping rule for the TRM unit. On this basis, for complex Boolean functions with more than three variables, we first recursively decompose them into combinations of sub-functions, each with no more than three input variables, using Shannon expansion based on the algebraic normal form of the Boolean function. Subsequently, these sub-functions are matched against the basic function types in Table 3 to determine the corresponding TRM unit configuration scheme for each sub-function. If a Boolean function cannot be fully computed within a single stage of the BFROU, requiring intermediate results to be cascaded and temporarily stored, multiple iterations are introduced. Following the above procedure, we manually mapped the Boolean functions actually used in 54 stream cipher algorithms,

one by one, and recorded the number of iterations required for each function to complete its full computation. The resulting distribution is presented in Fig. 16.



(**a**) (**b**)

Figure 16: The number of BFROU unit calls required for mapping 54 Boolean functions, where (**a**) shows the mapping results of 27 Boolean functions and (**b**) shows the mapping results of the remaining Boolean functions. The algorithm suffixes corresponding to each Boolean function in the figure represent different function types, among which G represents the clock-controlled function, F represents the feedback function, Z represents the keystream generation function, B represents the state transition function, T represents the algorithm initialization function, H represents the process update function, R represents the combination function, and L represents the filter function. These function types cover the mainstream application scenarios of Boolean functions in stream ciphers.

Experimental results demonstrate that the BFROU hardware architecture offers an efficient solution to the challenges of implementing diverse Boolean functions in stream ciphers. By optimizing the configuration of iteration counts, the architecture not only significantly reduces computational complexity and time overhead, but also successfully covers a broad spectrum of requirements, encompassing simple logic as well as complex algorithms such as TOYOCRYPT-HS1, Decim, and Ach80/128-F, thereby presenting a novel approach for the hardware implementation of Boolean functions. Its high efficiency and flexibility endow the BFROU with the potential to serve as a reconfigurable IP core, providing critical technical support for the design of future stream cipher chips.

7 Conclusions

This paper proposes a design method for reconfigurable operation units targeting Boolean functions in stream ciphers, grounded in customized RM logic units and NPN equivalence theory, thereby establishing a reconfigurable operation model that provides a theoretical basis for the hardware implementation of the BFROU. On this basis, this paper further selects the TRM operation unit with optimal configuration and finally constructs a 64-bit BFROU hardware architecture. Under the premise of maintaining complete functionality, this architecture effectively reduces area overhead and transmission delay. Compared with various existing implementation schemes for Boolean functions, the BFROU proposed in this paper exhibits better comprehensive performance in key indicators such as area efficiency and timing performance.

Acknowledgement: This work was supported by National Natural Science Foundation of China under Grant 62302519.

Funding Statement: This research was funded by the National Natural Science Foundation of China, grant number 62302519. The APC was funded by the same funder.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design: Zhaoxu Zhou, Zibin Dai; Data collection: Yanjiang Liu, Junwei Li; Analysis and interpretation of results: Zhaoxu Zhou, Yanjiang Liu; Draft manuscript preparation: Zhaoxu Zhou; Critical revision of the manuscript for important intellectual content: Zibin Dai, Junwei Li. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request. However, due to institutional network security policies, all data are stored within the university's internal network and cannot be exported externally. Access may therefore be limited to on-site collaboration.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dai Z, Li W, Chen T, Ren Q. Design and implementation of a high-speed reconfigurable feedback shift register. In: Proceedings of the 2008 4th IEEE International Conference on Circuits and Systems for Communications; 2008 May 26–28; Shanghai, China. doi:10.1109/iccsc.2008.79.
2. Liu L, Wang B, Deng C, Zhu M, Yin S, Wei S. Anole: a highly efficient dynamically reconfigurable crypto-processor for symmetric-key algorithms. IEEE Trans Comput Aided Des Integr Circuits Syst. 2018;37(12):3081–94. doi:10.1109/TCAD.2018.2801229.
3. Anderson JH, Wang Q. Area-efficient FPGA logic elements: architecture and synthesis. In: Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011); 2011 Jan 25–28; Yokohama, Japan. doi:10.1109/aspdac.2011.5722215.
4. Wang Z, Dai Z, Li W, Chuan X. Reconfigurable design for NBF based on optimal area utilization model. In: Proceedings of the 2016 10th IEEE International Conference on Anti-Counterfeiting, Security, and Identification (ASID); 2016 Sep 23–25; Xiamen, China.
5. Amagasaki M, Araki R, Iida M, Sueyoshi T. SLM: a scalable logic module architecture with less configuration memory. IEICE Trans Fundam. 2016;E99.A(12):2500–6. doi:10.1587/transfun.e99.a.2500.
6. Zhang Z, Dai Z, Liu Y, Zhou Z, Jiang D. Design of hybrid-granularity multifunctional computing unit based on And-Xor-Inv graph. J Electron Inf Technol. 2023;45(9):3370. (In Chinese). doi:10.11999/JEIT230021.
7. Zhou Z, Huang Z, Li J, Liu Y, Dai Z. CRM_BF: a low-overhead, high-efficient and reconfigurable operation unit design approach using the customized Reed-Muller unit for Boolean functions of sequence cipher algorithms. ACM Trans Des Autom Electron Syst. 2025;48:3725869. doi:10.1145/3725869.
8. Su Y, Shen J, Wang W. Reconfigurable design and implementation of nonlinear Boolean function for cloud computing security platform. Int J Inf Comput Secur. 2019;11(2):145. doi:10.1504/ijics.2019.098201.
9. Zhu Y, Xing Z, Xue J, Li Z, Hu Y, Zhang Y, et al. Area-efficient parallel reconfigurable stream processor for symmetric cryptograph. IEEE Access. 2021;9:28377–92. doi:10.1109/ACCESS.2021.3057866.
10. Zhang ZR, Dai ZB, Liu YJ, Zhang XL. RA-NLBF: design method of reconfigurable operation unit for stream cipher non-linear Boolean function. J Comput Appl. 2023;43(11):3527–33. (In Chinese). doi:10.11772/j.issn.1001-9081.2022111690.
11. Chen L, Lai J, Tong J. An AND-LUT based hybrid FPGA architecture. J Semicond. 2007;3(03):398–403. (In Chinese).
12. De Micheli G. Logic synthesis for emerging technologies. In: Proceedings of the 2023 IEEE 15th International Conference on ASIC (ASICON); 2023 Oct 24–27; Nanjing, China. doi:10.1109/ASICON58565.2023.10396421.

13. Thümmler M, Rai S, Kumar A. Improving technology mapping for and-inverter-cones. In: Proceedings of the 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE); 2022 Mar 14–23; Antwerp, Belgium. doi:10.23919/DATE54114.2022.9774544.
14. Zhang Z, Shao K, Deng R, Wang X, Zhang Y, Wang M. PrivLSTM: a privacy-preserving LSTM inference framework by fusing encryption and network structure for multi-sourced data. Inf Fusion. 2026;127:103711. doi:10.1016/j.inffus.2025.103711.
15. Perlis UM, Wei Peng C, Bahari Jambek A, Perlis UM, Baskara Dass S, Teik Wah L, et al. Performance evaluation of RISC-V microcontroller system on FPGA: a study of the NEORV32 core. Int J Integr Eng. 2024;16(1):26. doi:10.30880/ijie.2024.16.01.026.
16. Zhou X, Cai G, Huang Z. Design and implementation of RISC-V extended instruction set supporting FPGA dynamic reconfiguration. Comput Eng. 2025;51(5):229–38. (In Chinese). doi:10.19678/j.issn.1000-3428.0069267.
17. Iida M, Amagasaki M, Okamoto Y, Zhao Q, Sueyoshi T. COGRE: a novel compact logic cell architecture for area minimization. IEICE Trans Inf Syst. 2012;95(2):294–302. doi:10.1587/transinf.E95.D.294.
18. Vieira MD, Ng SSH, Walter M, Wille R, Walus K, Ferreira RS, et al. Three-input NPN class gate library for atomic silicon quantum dots. IEEE Des Test. 2022;39(6):147–55. doi:10.1109/MDAT.2022.3189814.
19. Edwards CR. A special class of universal logic gates (ULG) and their evaluation under the Walsh transform. Int J Electron. 1978;44(1):49–59. doi:10.1080/00207217808900783.
20. Sudhanya P, Joy Vasantha Rani SP. Analysis of FPGA architecture with hybrid logic blocks based on ULG and LUT. J Circ Syst Comput. 2025;34(2):2550059. doi:10.1142/s0218126625500598.
21. Wu X, Hurst SL. A new universal logic gate (ULG3) based on the Reed-Muller canonic expansion. Int J Electron. 1981;51(6):747–62. doi:10.1080/00207218108901380.
22. Sarangi S, Baas B. DeepScaleTool: a tool for the accurate estimation of technology scaling in the deep-submicron era. In: Proceedings of the 2021 IEEE International Symposium on Circuits and Systems (ISCAS); 2021 May 22–28; Daegu, Republic of Korea. doi:10.1109/iscas51556.2021.9401196.