Scientific Research Publishing

Journal of Computer and Communications, 2024, 12, 228-246
https://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

A High Efficiency Hardware Implementation of S-Boxes Based on Composite Field for Advanced Encryption Standard

Yawen Wang, Sini Bin, Shikai Zhu, Xiaoting Hu[*]

College of Computer Science & Technology, Jiangsu Normal University, Xuzhou, China
Email: *hxt@jsnu.edu.cn

How to cite this paper: Wang, Y.W., Bin, S.N., Zhu, S.K. and Hu, X.T. (2024) A High Efficiency Hardware Implementation of S-Boxes Based on Composite Field for Advanced Encryption Standard. *Journal of Computer and Communications*, **12**, 228-246.
https://doi.org/10.4236/jcc.2024.124016

Received: March 11, 2024
Accepted: April 27, 2024
Published: April 30, 2024

Abstract

The SubBytes (S-box) transformation is the most crucial operation in the AES algorithm, significantly impacting the implementation performance of AES chips. To design a high-performance S-box, a segmented optimization implementation of the S-box is proposed based on the composite field inverse operation in this paper. This proposed S-box implementation is modeled using Verilog language and synthesized using Design Complier software under the premise of ensuring the correctness of the simulation result. The synthesis results show that, compared to several current S-box implementation schemes, the proposed implementation of the S-box significantly reduces the area overhead and critical path delay, then gets higher hardware efficiency. This provides strong support for realizing efficient and compact S-box ASIC designs.

Keywords

Advanced Encryption Standard (AES), S-Box, Tower Field, Hardware Implementation, Application Specific Integration Circuit (ASIC)

1. Introduction

Advanced Encryption Standard (AES) is one of the most important block encryption algorithms at present and has been widely applied in various fields, such as communication, network security, e-commerce, and so on. In this algorithm, the plaintext is fixed at 128 bits, while its key length can vary between 128,192 and 256 bits. This algorithm arranges the 128-bit plaintext into a 4 × 4 state matrix by byte, and then encrypts the plaintext through multiple iterations of the round function which includes SubBytes (S-box), ShiftRow, and MixColumnand AddRoundKey operations. The number of iterations of encryption is determined by the length of the key 10 iterations for the 128-bit key, 12 iterations for a 192-bit key, and 14 iterations for a 256-bit key. Among these operations in the AES algorithm, S-box transformation is the only nonlinear transformation that plays a crucial role in the security of the encryption algorithm [1]. It makes cryptanalysis more difficult by adding confusion and diffusion and ensuring the security of the AES algorithm. However, its complexity largely impacts AES implementation performance.

Researchers have focused on improving the performance of S-box implementation through two methods: the look-up table method (LUT) [2] [3] and the algebraic method [4] [5]. The LUT method can transform complex nonlinear operations into simple look-up table operations, reducing the complexity and difficulty of implementation, but it has drawbacks like high area overhead and vulnerability to attack. In the algebraic method, the S-box is implemented based on the logical expression between inputs and outputs. Compared with the LUT method, the algebraic method does not need to calculate and store 256 predefined values of S-Box in ROM, resulting in a smaller area overhead. However, this kind of method involves the complex computation of the multiplicative inversion in finite field GF(2^8), which leads to low processing speed.

To achieve a better tradeoff between area overhead and speed for S-box implementation, the research has explored various strategies. Wang *et al.* [6] modified the affine transformation part of the S-box by using a pipeline, and proposed an area optimized combinational logic S-box implementation of AES. QIN *et al.* [7] obtained the logic expressions of the S-box and inverse S-box in the AES algorithm through the improved Q-M simplifying method, which reduced the delay, area, and power of the circuit. However, due to the bottleneck of multiplication inversion operation, this kind of improvement is limited. To address the bottleneck, a method based on composite fields was proposed [8] [9]. Using this method, one can convert the multiplicative inversion in GF(2^8) to low-order fields such as GF(2^4), GF(2^2), and GF(2) through isomorphic mapping, thereby reducing the complexity of multiplicative inversion and improving the efficiency of S-box processes. However, the implementation performance of S-box based on composite fields is related to the selection of low-order fields and their representation basis. To address this issue, several scholars have conducted extensive research on the selection of isomorphic matrix or polynomial basis and have proposed various optimization schemes of S-box [10] [11] [12] [13]. R. Ueno *et al.* [14] used different polynomial basis and Galois field arithmetic to optimize the inversion circuit, and proposed an efficient and compact S-box architecture. Some structural optimization schemes have also been proposed to improve the performance of the S-box. A. Reyhani-Masoleh *et al.* [15] mapped the finite field GF(2^8) to tower fields and proposed the structure of S-box and inverse S-box combination. Y.-T. TENG *et al.* [16] adopted a pipeline architecture based on composite fields and combined S-box and inverse S-box to achieve optimization effects. In [17] [18], an optimization scheme of the S-box is proposed by changing the order and structure of affine transformation and multiplication inversion. In addition, the method of combining combinational logic simplification technique with composite fields is often used to optimize the S-box implementation. A. Nakashima *et al.* [19] optimized the isomorphic mapping logic by combining multiplicative and exponential offsets. S. -H LIN *et al.* [20] proposed a S-box structure of a seven-stage hardware pipeline system with high throughput based on composite fields, logic optimization technology and pipeline-flow technology.

In this paper, we propose a low-area, high-performance S-box ASIC (Application Specific Integration Circuit) implementation by combining composite fields, the Karnaugh map simplification technique, the Input Variable Bypass Technique (IVB) [21] and other combinational logic simplification techniques. The main contributions of this paper are as follows:

1) We merged the inverse isomorphism and affine transformation into one matrix transform, which simplifies the implementation of the S-box and reduces the area overhead.

2) Based on composite fields, we map the multiplicative inversion in GF(2^8), GF(2^4) and GF(2^2) to GF($(2^4)^2$), GF($(2^2)^2$) and GF(2) respectively. This approach reduces the complexity of multiplicative inversion.

3) We proposed a high efficiency of S-box architecture based on segmented optimization by combing different logic simplification techniques and exploring different segment combinations.

The organization of the remaining sections of this paper is as follows: Section 2 introduces basic theoretical background of this research topic in this paper. Section 3 describes the segmented optimization S-box structure proposed in this paper. Section 4 presents the logic derivation and optimization of each module in the proposed structure. Section 5 provides the experiment result and performance evaluation. Finally, in Section 6, this paper is concluded.

2. Preliminaries

2.1. GF(2^8) and Its Operations

GF(2^8) is a finite field with 256 elements. These elements can be represented in 8-bit binary or as polynomials of degrees less than 8 with coefficients in GF(2). Suppose an element $a \in \mathrm{GF}\left(2^8\right)$ and its binary form is $a_7a_6a_5a_4a_3a_2a_1a_0$, then its corresponding polynomial representation is shown in Equation (1).

$$a\left(x\right) = a_7x^7 + a_6x^6 + \cdots + a_1x + a_0 \quad \left(a_i \in \mathrm{GF}\left(2\right)\right) \tag{1}$$

The operations in GF(2^8) include addition/subtraction, multiplication, inversion, and other operations. The addition/subtraction operation here ($\oplus$) is a bitwise XOR operation which, in polynomial representation, corresponds to the XOR of coefficients of the same terms.

The multiplication operation is modular multiplication ($\otimes$). It involves two steps: polynomial multiplication and reduction modulo an irreducible polynomial $m(x)$. Let $a(x)$ and $b(x)$ be elements in GF(2^8) and $c(x)$ be their product. Then $c(x) = a(x) \times b(x) \bmod m(x)$.

The multiplicative inverse $a^{-1}(x)$ of $a(x)$ in GF(2^8) is defined as the element that satisfies $a(x) \times b(x) \bmod m(x) = 1$.

Note that, in the AES algorithm, irreducible polynomial in GF(2^8) is specified as $m(x) = x^8 + x^4 + x^3 + x + 1$.

2.2. The Fundamental Construction Principle of the S-Box Based on Composite Fields

The original S-box transformation in the AES algorithm consists of two parts: multiplicative inversion and affine transformation in GF(2^8). Its algebraic expression is shown as Equation (2):

$$S_b[x] = Ax^{-1} \oplus 63\text{H} \tag{2}$$

where x, $S_b[x] \in \text{GF}(2^8)$ are the 8-bit input and output of the S-box respectively. x^{-1} is the inversion of *x*, and *A* is affine transformation matrix.

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

As mentioned above, in the S-box transformation, the operation to find the inversion element of *x* is the most complex and time-consuming. Traditionally, the extended Euclidean algorithm is employed for inversion in GF(2^8), but its hardware implementation efficiency is not high. Therefore, the method of solving inversion based on composite fields was proposed. The key idea of this method is to transform the inversion operation in high-order fields into low-order fields, thus reducing the complexity of the inverse algorithm. Specifically, the implementation of the S-box based on composite fields can be divided into four steps:

1) Mapping an element *q* in GF(2^8) to q' in the composite field GF(2^4)^2 by using an isomorphic matrix so that q' can be represented as $bx + c$ in GF(2^4)^2 where $b, c \in \text{GF}(2^4)$.

2) Using Equation (3) to find the multiplicative inverse of q' in GF((2^4)^2) [22].

$$\begin{aligned} q'^{-1} &= (bx + c)^{-1} \\ &= b\left(b^2\lambda \oplus bc\beta \oplus c^2\right)^{-1} x + (c \oplus b\beta)\left(b^2\lambda \oplus bc\beta \oplus c^2\right)^{-1} \end{aligned} \tag{3}$$

3) Converting q'^{-1} in GF(2^4)^2 into q^{-1} in GF(2^8) by using the inverse of

isomorphic matrix.

4) Performing the operation $Aq^{-1} \oplus 63\text{H}$ to get final the output of S-box $S_b[q]$.

Note that β and λ in Equation (3) are the coefficients of the irreducible polynomial $x^2+\beta x+\lambda$ chosen for multiplication when mapping GF(2^8) to GF($(2^4)^2$).

3. Proposed S-Box Architecture Based on Segmented Optimization

This section presents a segmented optimized S-box architecture. As shown in Figure 1, this architecture consists of an isomorphic mapping module, an inversion module in composite fields and a merged module of inverse isomorphism and affine transformation. The functions of each module are described as follows:

The isomorphic mapping module: To convert an 8-bit element q (be regarded as a column vector) in GF(2^8) into q' in GF($(2^4)^2$) by multiplying q by the isomorphic matrix δ. That is $q' = \delta q$.

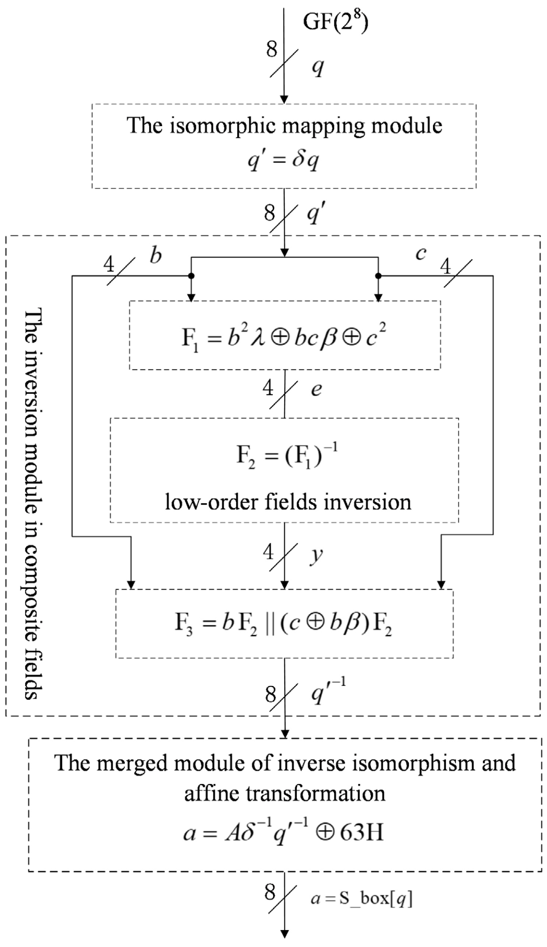

GF(2^8)
8
q
The isomorphic mapping module
$q' = \delta q$
8
q'
4
b
c
4
$\mathrm{F}_1 = b^2\lambda \oplus bc\beta \oplus c^2$
4
e
$\mathrm{F}_2 = (\mathrm{F}_1)^{-1}$
low-order fields inversion
4
y
$\mathrm{F}_3 = b\mathrm{F}_2 \,||\, (c \oplus b\beta)\mathrm{F}_2$
8
q'^{-1}
The inversion module in composite fields
The merged module of inverse isomorphism and affine transformation
$a = A\delta^{-1}q'^{-1} \oplus 63\text{H}$
8
$a = \text{S\_box}[q]$

Figure 1. Proposed S-box architecture based on segmented optimization.

The inversion module in composite fields: To calculate the inversion of q' in GF($(2^4)^2$). The inversion operation is completed by three steps F_1, F_2, F_3. All operations in submodules F_1, F_2, F_3 are 4-bit operations in the low-order field GF(2^4) which largely reduces the computational complexity.

The merged module of inverse isomorphism and affine transformation: To complete the operation $a = A\delta^{-1}q'^{-1} \oplus 63\text{H}$ equivalently by replacing $A\delta^{-1}$ with a merged matrix B and optimizing XOR constant operation where δ^{-1} represent the inverse isomorphism matrix.

4. Logic Derivation and Optimizations

In this section, we present the logic derivation for three modules and corresponding sub-segments in each module shown in Figure 1 in Section 3.

4.1. The Isomorphic Mapping Module

When calculating multiplicative inverses in GF(2^8) based on composite fields, the first step is to map the elements in GF(2^8) to the composite field GF($(2^4)^2$) using by multiplying an isomorphic matrix, but these isomorphic matrix is not fixed because the elements in GF($(2^4)^2$) can be generated by different irreducible polynomials. It implies that there exists diversity in the isomorphic mapping matrix from elements in GF(2^8) to GF($(2^4)^2$). To optimize the implementation performance of the isomorphic mapping module in S-box, it is necessary to carefully to select appropriate irreducible polynomials to get the best isomorphic matrix, thereby reducing critical path delay and area overhead. Refer to [22], we choose the isomorphic matrix δ as shown in Equation (4).

$$\delta = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{4}$$

Assuming that the input of the S-box is $q = [q_7, q_6, q_5, q_4, q_3, q_2, q_1, q_0]^{\text{T}}$, the expression for the output of the isomorphic mapping module is shown as Equation (5).

$$\delta \times q = q' = \begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ b_0 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{bmatrix} \tag{5}$$

where $b_3b_2b_1b_0$ represents the high 4 bits of output q', and $c_3c_2c_1c_0$ represents the low 4 bits of output q'.

4.2. The Inversion Module in Composite Fields

To obtain the multiplicative inverse of q' in GF($(2^4)^2$) expressed as Equation (3), we decompose Equation (3) into F_1, F_2 and F_3 as shown in Equation (6), (7) and (8), and use the recursive deduction method to get the logical expressions of F_1, F_2 and F_3.

$$F_1 = b^2\lambda \oplus bc\beta \oplus c^2 \tag{6}$$

$$F_2 = (F_1)^{-1} \tag{7}$$

$$F_3 = bF_2 \,||\, (c \oplus b\beta)F_2 \tag{8}$$

where the symbol "||" in Equation (8) represents concatenation.

In order to optimize the implementation, we select $\beta = 1$, $\lambda = 1100\,b$ to be the coefficients of the generating polynomial $m_1(x) = x^2 + \beta x + \lambda$ of the composite field GF($(2^4)^2$), then Equations (6), (7) and (8) are simplified to (9), (10), (11).

$$F_1 = b^2\lambda \oplus bc \oplus c^2 \tag{9}$$

$$F_2 = (F_1)^{-1} \tag{10}$$

$$F_3 = bF_2 \,||\, (c \oplus b)F_2 \tag{11}$$

Note that, when calculating inversion in composite fields, three types of irreducible polynomials are required to complete the operation conversion from the high-order field to the low-order field. To simplify the following expression, we list the three types of polynomials as shown in Table 1.

4.2.1. F_1 Module

Figure 2 shows the calculation process of F_1. Next, we will deduce and optimize the logic expression for each step in Figure 2.

Table 1. Three types of irreducible polynomials.

Composite Field	Irreducible Polynomial
GF(2^8) $\to$ GF($(2^4)^2$)	$m_1(x) = x^2 + x + \lambda$, where $\lambda = 1100b$
GF(2^4) $\to$ GF($(2^2)^2$)	$m_2(x) = x^2 + x + \varphi$, where $\varphi = 10b$
GF(2^2) $\to$ GF($(2)^2$)	$m_3(x) = x^2 + x + 1$

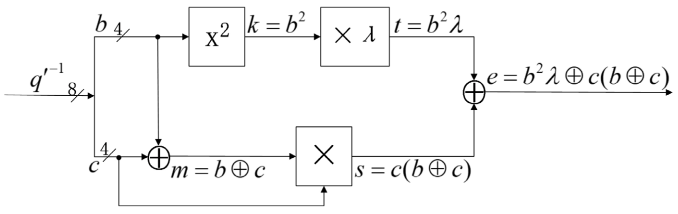


Figure 2. F_1 module logic operation diagram.

1) b^2 operation in GF(2^4)

As can be seen from **Figure 2**, we let $k = b^2 = k_3k_2k_1k_0 = (b_3b_2b_1b_0)^2$, then this operation can be expressed as the form in GF($(2^2)^2$) shown in Equation (12).

$$\begin{aligned} k = b^2 &= (b_H x + b_L)^2 \bmod m_2(x) \\ &= b_H^2 x + (b_H^2 \varphi \oplus b_L^2) \\ &= k_H x + k_L \end{aligned} \tag{12}$$

where $k_H = k_3k_2$, $k_L = k_1k_0$, $b_H = b_3b_2$, $b_L = b_1b_0$, $m_2(x)$ (see **Table 1**) is an irreducible generating polynomial for GF($(2^2)^2$). Obviously,

$$k_H = b_H^2;\ k_L = b_H^2 \varphi \oplus b_L^2. \tag{13}$$

Then one can further decompose the operation b_H^2 and b_L^2 in GF(2^2) to GF(2) to get Equation (14):

$$\begin{cases} b_H^2 = (b_3 x + b_2)^2 \bmod m_3(x) = b_3 x + (b_3 \oplus b_2) \\ b_L^2 = (b_1 x + b_0)^2 \bmod m_3(x) = b_1 x + (b_1 \oplus b_0) \end{cases} \tag{14}$$

where $m_3(x)$ (see **Table 1**) is an irreducible generating polynomial in GF(2).

By combining Equation (13), (14), we can obtain Equation (15).

$$\begin{cases} k_H = b_3 x + (b_3 \oplus b_2) = k_3 x + k_2 \\ k_L = (b_1 \oplus b_2) x + (b_0 \oplus b_1 \oplus b_3) = k_1 x + k_0 \end{cases} \tag{15}$$

According to Equation (15), we can get the expression of k as shown in equation (16).

$$\begin{cases} k_3 = b_3 \\ k_2 = b_2 \oplus b_3 \\ k_1 = b_1 \oplus b_2 \\ k_0 = b_0 \oplus b_1 \oplus b_3 \end{cases} \tag{16}$$

2) $t = b^2\lambda$ operation

As mentioned above, in order to optimize the expression, we choose $\lambda = \{1100\}_2$ and let

$$t = b^2\lambda = k\lambda = t_3t_2t_1t_0 = (k_3k_2k_1k_0) \otimes \lambda.$$

Then, the polynomial representation of t can be written as Equation (17).

$$\begin{aligned} t &= (k_H x + k_L)(\lambda_H x + \lambda_L) \bmod m_2(x) \\ &= (k_H\lambda_H x^2 + k_H\lambda_L x + k_L\lambda_H x + k_L\lambda_L) \bmod m_2(x) \\ &= (k_L\lambda_H \oplus k_H\lambda_H) x + k_H\lambda_H\varphi \\ &= t_H x + t_L \end{aligned} \tag{17}$$

where $t_H = t_3t_2$, $t_L = t_1t_0$, $\lambda_H = 11$, $\lambda_L = 00$.

Using similar derivation method used for k, t_H and t_L can be expressed as Equation (18).

$$\begin{cases} t_H = k_L\lambda_H \oplus k_H\lambda_H \\ \quad = \left(\left(k_1x+k_0\right)\left(x+1\right)+\left(k_3x+k_2\right)\left(x+1\right)\right)\operatorname{mod} m_3\left(x\right) \\ \quad = \left(k_0 \oplus k_2\right)x+\left(k_0 \oplus k_1 \oplus k_2 \oplus k_3\right) \\ \quad = t_3x+t_2 \\ t_L = k_H\lambda_H\varphi \\ \quad = \left[\left(k_3x+k_2\right)\left(x+1\right)x\right]\operatorname{mod} m_3\left(x\right) \\ \quad = k_3x+k_2 \\ \quad = t_1x+t_0 \end{cases} \tag{18}$$

Then we can get the expression of t as Equation (19).

$$\begin{cases} t_3 = k_0 \oplus k_2 \\ t_2 = k_0 \oplus k_1 \oplus k_2 \oplus k_3 \\ t_1 = k_3 \\ t_0 = k_2 \end{cases} \tag{19}$$

3) $s = c\left(b \oplus c\right)$ operation

For s, we let $m_i = b_i \oplus c_i$, then the expression of s can be written as Equation (20).

$$\begin{cases} s_3 = c_3m_2 \oplus c_2m_3 \oplus c_3m_0 \oplus c_0m_3 \oplus c_2m_1 \oplus c_1m_2 \oplus c_3m_3 \oplus c_3m_1 \oplus c_1m_3 \\ s_2 = c_2m_2 \oplus c_2m_0 \oplus c_0m_2 \oplus c_3m_3 \oplus c_3m_1 \oplus c_1m_3 \\ s_1 = c_1m_0 \oplus c_0m_1 \oplus c_2m_2 \oplus c_1m_1 \oplus c_3m_2 \oplus c_2m_3 \\ s_0 = c_0m_0 \oplus c_1m_1 \oplus c_3m_2 \oplus c_2m_3 \oplus c_3m_3 \end{cases} \tag{20}$$

By combining Equations (16), (19), (20) and simplifying them, we can obtain the logic expression of e as Equation (21).

$$\begin{cases} e_3 = b_0\overline{c_3} \oplus b_1\overline{c_2} \oplus b_2\overline{c_1} \oplus c_3b_2 \oplus c_2b_3 \oplus c_0b_3 \oplus c_3\overline{b_3} \oplus c_3b_1 \oplus c_1b_3 \\ e_2 = \overline{c_2}b_0 \oplus \overline{c_1}b_3 \oplus c_2\overline{b_2} \oplus c_0b_2 \oplus c_3\overline{b_3} \oplus c_3b_1 \\ e_1 = c_1b_0 \oplus c_0b_1 \oplus c_2\overline{b_2} \oplus c_1\overline{b_1} \oplus c_3b_2 \oplus \overline{c_2}b_3 \\ e_0 = c_0\overline{b_0} \oplus c_1\overline{b_1} \oplus \overline{c_3}b_2 \oplus \overline{c_2}b_3 \oplus c_3\overline{b_3} \end{cases} \tag{21}$$

4.2.2. F₂ Module

The F₂ module is used to calculate the inversion in GF(2^4). To derive and simplify the output expression of F₂, we first utilize the idea of tower fields to partition the operation in F₂ into three parts: G₁, G₂ and G₃, as depicted in **Figure 3**. Subsequently, after obtaining the logic expression of G₁, G₂ and G₃, we merge them into an expression. Finally, we employ Karnaugh map simplification technique, input variable bypass technique [21] and other logic simplification technique to further simplify and obtain the output of F₂.

Specifically, according to the idea of tower fields, the G₁, G₂ and G₃ can be expressed as Equation (22).

$$\begin{cases} \mathrm{G}_1 = e_H^2\varphi \oplus e_L\left(e_H \oplus e_L\right) \\ \mathrm{G}_2 = \left(\mathrm{G}_1\right)^{-1} \\ \mathrm{G}_3 = e_H\mathrm{G}_2 \,\|\, \left(e_H \oplus e_L\right)\mathrm{G}_2 \end{cases} \tag{22}$$

where $e_H = e_3e_2$, $e_L = e_1e_0$.

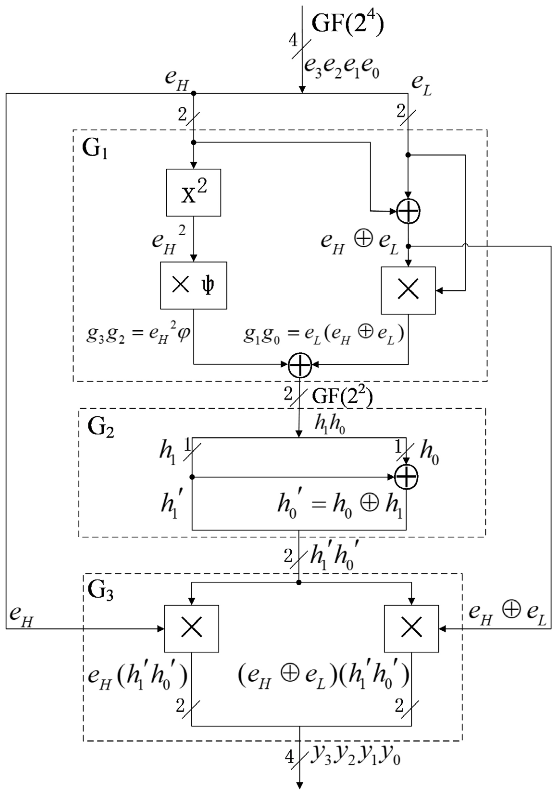


Figure 3. F₂ module structure diagram.

For G₁, we first get Equation (23), then obtain the expression of G₁ as shown in equation (24) by combining Equations (22) and (23) and simplifying them.

$$\begin{cases} e_H^2 = \left(e_3, e_2 \oplus e_3\right) \\ e_H^2 \varphi = \left(e_2, e_3\right) \\ e_L\left(e_H \oplus e_L\right) = \left(e_1\overline{e_3} \oplus e_0e_3 \oplus e_1e_2, e_1\overline{e_3} \oplus e_0\overline{e_2}\right) \end{cases} \tag{23}$$

$$\begin{aligned} \mathrm{G}_1 &= \left(h_1, h_0\right) \\ &= \left(e_1\overline{e_3} \oplus e_0e_3 \oplus e_2\overline{e_1}, e_3 \oplus e_1\overline{e_3} \oplus e_0\overline{e_2}\right) \end{aligned} \tag{24}$$

For G₂, we directly use the simplification method of Karnaugh map to obtain its output expression as shown in Equation (25).

$$\begin{cases} h_1' = h_1 \\ h_0' = h_1 \oplus h_0 \end{cases} \tag{25}$$

Finally, by combining with Equation (24) and (25) and G₃ in Equation (22) and simplifying the expression combined, we obtain the expression of F₂ as shown in Equation (26).

$$\begin{cases} y_3 = \overline{e_0}e_3 \oplus e_1e_2\overline{e_3} \oplus \overline{e_1}e_2 \\ y_2 = e_0\overline{e_2}e_3 \oplus e_1e_2e_3 \oplus \overline{e_1}e_2 \\ y_1 = e_1 \oplus e_0e_1\overline{e_2} \oplus e_3 \oplus e_1e_2\overline{e_3} \oplus \overline{e_0}\,\overline{e_1}e_2 \oplus e_0e_1\overline{e_3} \\ y_0 = e_1e_2 \oplus e_1\overline{e_2}\,\overline{e_3} \oplus \overline{e_0}\,\overline{e_1}e_2 \oplus e_0e_1e_3 \oplus e_0\overline{e_2}\,\overline{e_3} \end{cases} \tag{26}$$

4.2.3. F3 Module

As shown in **Figure 1**, *b*, *c* and *y* are the inputs of F3, then we can derive to get the output expression of F3 $q_7'^{-1}q_6'^{-1}q_5'^{-1}q_4'^{-1}q_3'^{-1}q_2'^{-1}q_1'^{-1}q_0'^{-1}$ as shown in Equation (27).

$$\begin{cases} q_7'^{-1} = y_3(b_0 \oplus b_1 \oplus b_2 \oplus b_3) \oplus y_2(b_1 \oplus b_3) \oplus y_1(b_2 \oplus b_3) \oplus y_0 b_3 \\ q_6'^{-1} = y_3(b_1 \oplus b_3) \oplus y_2(b_0 \oplus b_2) \oplus y_1 b_3 \oplus y_0 b_2 \\ q_5'^{-1} = y_3 b_2 \oplus y_2(b_2 \oplus b_3) \oplus y_1(b_0 \oplus b_1) \oplus y_0 b_1 \\ q_4'^{-1} = y_3(b_2 \oplus b_3) \oplus y_2 b_3 \oplus y_1 b_1 \oplus y_0 b_0 \\ q_3'^{-1} = y_3(b_0 \oplus c_0 \oplus b_1 \oplus c_1 \oplus b_2 \oplus c_2 \oplus b_3 \oplus c_3) \\ \qquad \oplus y_2(b_1 \oplus c_1 \oplus b_3 \oplus c_3) \oplus y_1(b_2 \oplus c_2 \oplus b_3 \oplus c_3) \oplus y_0(b_3 \oplus c_3) \\ q_2'^{-1} = y_3(b_1 \oplus c_1 \oplus b_3 \oplus c_3) \oplus y_2(b_0 \oplus c_0 \oplus b_2 \oplus c_2) \\ \qquad \oplus y_1(b_3 \oplus c_3) \oplus y_0(b_2 \oplus c_2) \\ q_1'^{-1} = y_3(b_2 \oplus c_2) \oplus y_2(b_2 \oplus c_2 \oplus b_3 \oplus c_3) \oplus y_1(b_0 \oplus c_0 \oplus b_1 \oplus c_1) \\ \qquad \oplus y_0(b_1 \oplus c_1) \\ q_0'^{-1} = y_3(b_2 \oplus c_2 \oplus b_3 \oplus c_3) \oplus y_2(b_3 \oplus c_3) \oplus y_1(b_1 \oplus c_1) \oplus y_0(b_0 \oplus c_0) \end{cases} \tag{27}$$

To optimize the implementation of Equation (27) in the hardware circuit, we extract the common part of 8 sub-expressions of Equation (27) and define intermediate variable d_i as shown in Equation (28). Then the expression can be optimized into Equation (29).

$$\begin{cases} d_0 = b_0 \oplus b_1 \\ d_1 = b_0 \oplus b_2 \\ d_2 = b_1 \oplus b_3 \\ d_3 = b_2 \oplus b_3 \\ d_4 = b_3 \oplus c_3 \\ d_5 = b_2 \oplus c_2 \\ d_6 = b_1 \oplus c_1 \\ d_7 = b_0 \oplus c_0 \\ d_8 = d_0 \oplus d_3 \\ d_9 = d_4 \oplus d_5 \oplus d_6 \oplus d_7 \\ d_{10} = d_4 \oplus d_6 \\ d_{11} = d_4 \oplus d_5 \\ d_{12} = d_5 \oplus d_7 \\ d_{13} = d_6 \oplus d_7 \end{cases} \tag{28}$$

$$\begin{cases} q_7'^{-1} = y_3 d_8 \oplus y_2 d_2 \oplus y_1 d_3 \oplus y_0 b_3 \\ q_6'^{-1} = y_3 d_2 \oplus y_2 d_1 \oplus y_1 b_3 \oplus y_0 b_2 \\ q_5'^{-1} = y_3 b_2 \oplus y_2 d_3 \oplus y_1 d_0 \oplus y_0 b_1 \\ q_4'^{-1} = y_3 d_3 \oplus y_2 b_3 \oplus y_1 b_1 \oplus y_0 b_0 \\ q_3'^{-1} = y_3 d_9 \oplus y_2 d_{10} \oplus y_1 d_{11} \oplus y_0 d_4 \\ q_2'^{-1} = y_3 d_{10} \oplus y_2 d_{12} \oplus y_1 d_4 \oplus y_0 d_5 \\ q_1'^{-1} = y_3 d_5 \oplus y_2 d_{11} \oplus y_1 d_{13} \oplus y_0 d_6 \\ q_0'^{-1} = y_3 d_{11} \oplus y_2 d_4 \oplus y_1 d_6 \oplus y_0 d_7 \end{cases} \tag{29}$$

4.3. The Merged Module of Inverse Isomorphism and Affine Transformation

This module completes the operation $a = \left(A\delta^{-1}q'^{-1}\right) \oplus 63\text{H}$. For this module, we use two methods to optimize the logic to reduce the number of logic gates and shorten the critical path delay: 1) To merge the inverse isomorphic mapping and affine transformation into a matrix transformation. 2) To simplify the logic expression for XOR operation with 63H by using NOT gate to replace XOR.

Specifically, we let $B = A \times \delta^{-1}$, q'^{-1} and q to be the input and output of the merged module of inverse isomorphism and affine transformation respectively. Then the expression of the original expression $a = \left(A\delta^{-1}q'^{-1}\right) \oplus 63\text{H}$ can be reduced to Equation (30).

$$a = Bq'^{-1} \oplus 63\text{H} \tag{30}$$

where the inverse transformation matrix δ^{-1}, affine transformation matrix A and merged matrix B corresponds to Equations (31), (32) and (33) respectively.

$$\delta^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{31}$$

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{32}$$

$$B = A\delta^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{33}$$

Then by combining the Equations (30) and (33), we can get the expression of output of S-box a as shown in Equation (34).

$$\begin{cases} a_7 = q_7'^{-1} \oplus q_3'^{-1} \oplus q_2'^{-1} \oplus 0 \\ a_6 = q_7'^{-1} \oplus q_6'^{-1} \oplus q_5'^{-1} \oplus q_4'^{-1} \oplus 1 \\ a_5 = q_7'^{-1} \oplus q_2'^{-1} \oplus 1 \\ a_4 = q_7'^{-1} \oplus q_4'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \oplus 0 \\ a_3 = q_2'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \oplus 0 \\ a_2 = q_6'^{-1} \oplus q_5'^{-1} \oplus q_4'^{-1} \oplus q_3'^{-1} \oplus q_2'^{-1} \oplus q_0'^{-1} \oplus 0 \\ a_1 = q_7'^{-1} \oplus q_0'^{-1} \oplus 1 \\ a_0 = q_7'^{-1} \oplus q_6'^{-1} \oplus q_2'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \oplus 1 \end{cases} \tag{34}$$

Observing the expression in Equation (34), we find that expressions of a_6, a_5, a_1, a_0 includes constant term 1 and they are common term $q_7'^{-1}$. Thus we can use $\overline{q_7'^{-1}}$ to substitute the $q_7'^{-1} \oplus 1$ and convert Equation (34) into Equation (35), which simplify the expression and reduce the number of XOR gate.

$$\begin{cases} a_7 = q_7'^{-1} \oplus q_3'^{-1} \oplus q_2'^{-1} \\ a_6 = \overline{q_7'^{-1}} \oplus q_6'^{-1} \oplus q_5'^{-1} \oplus q_4'^{-1} \\ a_5 = \overline{q_7'^{-1}} \oplus q_2'^{-1} \\ a_4 = q_7'^{-1} \oplus q_4'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \\ a_3 = q_2'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \\ a_2 = q_6'^{-1} \oplus q_5'^{-1} \oplus q_4'^{-1} \oplus q_3'^{-1} \oplus q_2'^{-1} \oplus q_0'^{-1} \\ a_1 = \overline{q_7'^{-1}} \oplus q_0'^{-1} \\ a_0 = \overline{q_7'^{-1}} \oplus q_6'^{-1} \oplus q_2'^{-1} \oplus q_1'^{-1} \oplus q_0'^{-1} \end{cases} \tag{35}$$

4.4. Example

To test the correctness of the output of each segment, taking the input 11110000 as an example and referring to equations (5), (21), (26), (28), (29) and (35), we calculate and obtain the operation results of each segment in Figure 1 and the final S-box output as shown in Table 2. It is observed that the output of S-box of proposed S-box is consistent with the output of AES standard, which proves that the expression of each segment is correct.

5. Experiment Result and Performance Evaluation

In order to evaluate the hardware implementation performance of the proposed architecture, we modelled and simulated the proposed design using Verilog language and Modelsim respectively. Design Complier (DC) was employed to complete ASIC synthesis under TSMC 90nm CMOS Technology library.

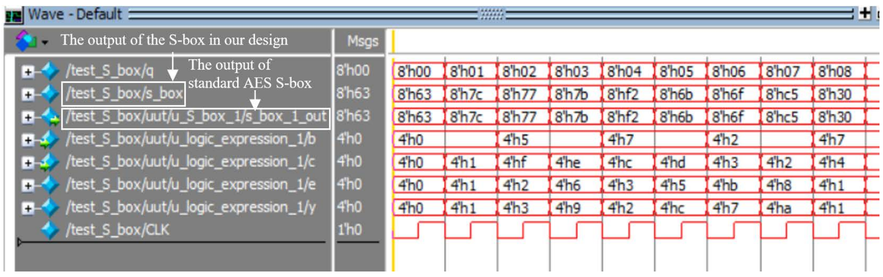
Table 2. The segmented calculation results of S-box in this paper and the comparison with AES standard output.

Step	q	q'	e	y	q'^{-1}	a (the output of S-box)
This work	11110000	01000001	1100	0101	00100111	10001100
Standard S-box	11110000	-	-	-	-	10001100

Figure 4 shows the simulation results of two kinds of S-box. In Figure 4, *s_box_1_out* represents the output result of the standard AES S-box, and *s_box* is that of the S-box proposed in this paper. As shown in Figure 4, the output of the S-box in this paper is consistent with that of the standard S-box for different inputs, which proves that the scheme in this paper is correct.

Table 3 shows the DC synthesis results and the theoretical estimation results of Critical Path Delay (CPD). Note that the delay equivalent relationship between logic gates used to estimate CPD and equivalent CPD in Table 3 refers to the standard in [13]. The specific equivalent relationship is listed in Table 4. Note that, in Table 4, XOR stands for two-input XOR gate, AND two-input AND gate, AND3X1 three-input AND gate, OR two-input OR gate, MUX two-input multiplexer.

According to the DC synthesis results in Table 3, we first compared the synthesis results of our proposed S-box with [16] synthesized using the same TSMC 90 nm CMOS standard cell library.


Wave - Default
The output of the S-box in our design
The output of standard AES S-box
Msgs

```
/test_S_box/q                            8'h00  8'h00 8'h01 8'h02 8'h03 8'h04 8'h05 8'h06 8'h07 8'h08
/test_S_box/s_box                        8'h63  8'h63 8'h7c 8'h77 8'h7b 8'hf2 8'h6b 8'h6f 8'hc5 8'h30
/test_S_box/uut/u_S_box_1/s_box_1_out    8'h63  8'h63 8'h7c 8'h77 8'h7b 8'hf2 8'h6b 8'h6f 8'hc5 8'h30
/test_S_box/uut/u_logic_expression_1/b   4'h0   4'h0        4'h5        4'h7        4'h2        4'h7
/test_S_box/uut/u_logic_expression_1/c   4'h0   4'h0  4'h1  4'hf  4'he  4'hc  4'hd  4'h3  4'h2  4'h4
/test_S_box/uut/u_logic_expression_1/e   4'h0   4'h0  4'h1  4'h2  4'h6  4'h3  4'h5  4'hb  4'h8  4'h1
/test_S_box/uut/u_logic_expression_1/y   4'h0   4'h0  4'h1  4'h3  4'h9  4'h2  4'hc  4'h7  4'ha  4'h1
/test_S_box/CLK                          1'h0
```

Figure 4. Modelsim simulation results of the proposed S-box.

Table 3. DC synthesis results of different S-boxes and theoretical estimation results of critical path delay.

S-box structure	Area (μm²)	Tech.	Frequency (Mhz)	Through-put (Gbps)	Hardware efficiency (Mbps/μm²)	The equivalent number of NAND (GE)	Critical path delay	Equivalent critical path delay	Time (ns)	GE*Time
our design	1593.24	TSMC 90 nm	1041.67	8.333	5.230	565	15 XOR + 2 AND + 1 AND3X1 + 2 INV	51 NAND + 4 INV	0.96	542.40
[16]	2769.48	TSMC 90 nm	1204.82	9.639	3.480	981	19 XOR + 3 AND + 1 OR + 2 MUX	65 NAND + 8 INV	0.83	814.23
[14] (compact)	-	TSMC 65 nm	328.95	2.632	-	249	-	-	3.04	756.96
[20]	832.13	TSMC 40 nm	1250.00	10.000	12.017	1223	-	-	0.8	978.40

Table 4. The delay equivalent relationship.

Logic gate type	Equivalent to the number of AND + INV
XOR	3 NAND
AND	1 NAND + 1 INV
AND3X1	4 NAND
OR	1 NAND
MUX	2 NAND + 2.5 INV

The results indicate that the S-box implementation proposed in this paper only needs 1593.24 μm^2 area overhead. In comparison with the implementation of the pipeline architecture proposed in [16], although the implementation in this paper reduces the frequency and throughput by approximately 13.54% and 13.55% respectively, the area overhead is reduced by about 42.47%, leading to an increase in hardware efficiency (*throughput/area*) by about 50.29%. Obviously, the scheme proposed in this paper offers significant advantages in terms of area overhead and hardware efficiency.

In addition, in terms of CPD, the S-box implementation proposed in this paper has a CPD of 15 XOR + 2 AND + 1 AND3X1 + 2 INV. The CPD is calculated based on Equations (5), (21), (26), (28), (29), (35) which is corresponding to five segments in **Figure 1** in Section 3. Specifically, the CPD is 3 XOR for Equation (5), 4 XOR + 1 AND + 1 INV for Equation (21), and 3 XOR + 1 AND3X1 + 1 INV for Equation (26) and (28) (calculated in parallel). Then the CPD of Equation (29) and (35) are 2 XOR + 1 AND and 3 XOR, respectively. Comparatively, the CPD in [16] is 19 XOR + 3 AND + 1 OR + 2 MUX. After being converted equivalently according to the relationship shown in **Table 4**, the CPD of this paper is 51 NAND + 4 INV, whereas that of [16] is 65 NAND + 8 INV. It is evident that the CPD of the S-box proposed in this paper saves the delay of 14 NAND and 4 INV logic gates compared with [16]. Even ignoring the difference in the number of INV logic gates, the CPD of the S-box in this paper is about 21.54% higher than that in [16], which is a relatively valuable improvement. However, it should be noted that the DC synthesis results in [16] show that its working clock frequency is higher than that in this paper. This is mainly because the synthesis results given in [16] are for the S-box of 5-level pipeline architecture in which its clock frequency is determined by the CPD of some sub-stage, but not entire S-box.

Secondly, we compared our proposed implementation with [14] and [20] which were synthesized using TSMC 65 nm and 40 nm CMOS standard cell library respectively. For the convenience of comparison with related data in [14], we calculated that the frequency ($frequency = 1/Time$) and throughput ($throughput = 8\ bit/Time$) in [14] are approximately 328.95 Mhz and 2.632 Gbps respectively based on the Time given in [14] and made a comparison with corresponding data in this paper. The comparison results show that the Time of the proposed implementation in this paper is reduced by approximately 68.42% compared to [14], which indicates that the speed of S-box implementation proposed in this paper significantly surpasses that reported in [14]. Furthermore, despite the larger area of the S-box proposed in this paper compared to [14], the S-box in this paper exhibits a lower area-time product (GE*Time) reduced by approximately 28.34%. It implied that our design achieves a superior balance between area and speed.

Reference [20] is the latest known paper focusing on S-box implementation. To facilitate comparison, we first calculated that the equivalent GE in [20] is 1223 two-input NAND Gates (the equivalent GE under TSMC40 nm technology library = area under TSMC40 nm/0.68 where 0.68 is the area of the two-input NAND in the TSMC 40 nm technology library). The comparison results show that the S-box implementation proposed in this paper can save GE and the product of GE and Time by approximately 53.80% and 44.56% respectively and improve 80.59% hardware efficiency compared to that in [20].

Overall, the S-box implementation proposed in this paper has obvious advantages over the existing schemes in terms of area overhead, critical path delay and hardware efficiency. It can provide favorable support for implementing efficient and compact AES S-box ASIC design.

6. Conclusions and Future Work

Aiming at the shortcomings of low processing speed and computational complexity in multiplicative inversion over the finite field, this paper explores the optimization of AES S-box and proposes a segmented optimized S-box architecture by using the idea of calculating inverse elements in tower fields and combining with Karnaugh map simplification and IVB technique. This architecture was divided into three modules: an isomorphic mapping module, an inversion module in composite fields and a merged module of inverse isomorphism and affine transformation. In this way, we simplify the logic of the S-box implementation.

In the isomorphic mapping module, based on the concept of composite fields, we map elements in the finite field to the composite field, thus reducing the complexity of calculations. In the inversion module in composite fields, we use the recursive method to optimize the inversion part. The experimental results indicate that compared with the traditional algebraic method, our design effectively reduces the computational complexity and area overhead. In the merged module of inverse isomorphism and affine transformation, we combine the matrix of inverse isomorphic mapping and affine transformation into one to further simplify the logic expression of the S-box.

In conclusion, the segmented optimized S-box architecture proposed in this paper has the characteristics of small area, low critical path delay and high hardware efficiency which is very valuable for area-limited applications.

While we have made some progress in the implementation of S-boxes, we only consider the forward AES S-box in this paper because many operations in the block encryption algorithm only involve encryption. The inverse S-box and the forward S-box have many similar parts, so how to design the low-area inverse S-box and how to combine the S-box and the inverse S-box is the work we need to study in the future. At the same time, although the AES encryption algorithm has higher security performance, the security of AES hardware implementation is seriously threatened by the development of hardware attack technology. Therefore, to improve the security of the AES encryption algorithm, adding fault detection on the basis of the proposed S-box architecture is also our future work.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Xu, Y.T., Lyu, Z.G., Huang, Y.G. and Li, X.Y. (2020) Optimization and Implementation of AES Algorithm Based on STM32 MCU. *Process Automation Instrumentation*, **41**, 56-60. https://chn.oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFD&dbname=CJFDLAST2020&filename=ZDYB202007013&uniplatform=OVERSEA&v=FBrazSYzB-bBRjeM4cilrp85xrbI9lm56klGQJewY_FfxAjpFv1g49uZwQfYYS3u

[2] Manjith, B.C. (2019) Improving Overall Parallelism in AES Accelerator Using BRAM and Multiple Input Blocks. 2019 *Innovations in Power and Advanced Computing Technologies* (*i-PACT*), Vellore, 22-23 March 2019, 1-5. https://doi.org/10.1109/i-PACT44901.2019.8960016

[3] Arul Murugan, C., Karthigaikumar, P. and Priya, S.S. (2020) FPGA Implementation of Hardware Architecture with AES Encryptor Using Sub-Pipelined S-Box Techniques for Compact Applications. *Automatika*, **61**, 682-693. https://doi.org/10.1080/00051144.2020.1816388

[4] Rachh, R.R. and Ananda Mohan, P.V. (2008) Implementation of AES S-Boxes Using Combinational Logic. 2008 *IEEE International Symposium on Circuits and Systems* (*ISCAS*), Seattle, 18-21 May 2008, 3294-3297. https://doi.org/10.1109/ISCAS.2008.4542162

[5] Ahmad, N., Rezaul Hasanand, R. and Jubadi, W.M. (2010) Design of AES S-Box Using Combinational Logic Optimization. 2010 *IEEE Symposium on Industrial Electronics & Applications* (*ISIEA*), Penang, 3-5 October 2010, 696-699. https://doi.org/10.1109/ISIEA.2010.5679375

[6] Wang, Q., Liang, J. and Qi, Y. (2010) The Area Optimized Implementation of S-box in AES Algorithm. *Chinese Journal of Electronics*, **38**, 939-942. https://kns.cnki.net/kcms2/article/abstract?v=smPsKIJgVaAXklosraIEqnytl0tPMLltpr9WZoEcUceHoSuINcBb4nLePRzxH-SSJdJ5qxypin17TJWVPQy99V8N47WNPrJaAIT7SexfzEdyNjgiXFuNLuhitAQEjLJSuxZ3wNLE8p8=&uniplatform=NZKPT&language=CHS

[7] Qin, X.C. and Li, S.G. (2014) An Expression Method to Implement S-Box and Inverse S-Box Substitution for AES Algorithm. *Microelectronics & Computer*, **31**, 112-115.

https://kns.cnki.net/kcms2/article/abstract?v=smPsKIJgVaAXklosraIEqnytl0tPMLltpr9WZoEcUceHoSuINcBb4nLePRzxH-SSJdJ5qxypin17TJWVPQy99V8N47WNPrJaAIT7SexfzEdyNjgiXFuNLuhitAQEjLJSuxZ3wNLE8p8=&uniplatform=NZKPT&language=CHS

[8] Satoh, A., Morioka, S., Takano, K. and Munetoh, S. (2001) A Compact Rijndael Hardware Architecture with S-Box Optimization. In: Boyd, C., Ed., *ASIACRYPT* 2001: *Advances in Cryptology—ASIACRYPT* 2001, Springer, Berlin, 239-254. https://doi.org/10.1007/3-540-45682-1_15

[9] Canright, D. (2005) A Very Compact S-Box for AES. In: Rao, J.R. and Sunar, B., Eds., *CHES* 2005: *Cryptographic Hardware and Embedded Systems—CHES* 2005, Springer, Berlin, 441-455. https://doi.org/10.1007/11545262_32

[10] Reyhani-Masoleh, A., Taha, M. and Ashmawy, D. (2018) Smashing the Implementation Records of AES S-Box. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, **2018**, 298-336. https://doi.org/10.46586/tches.v2018.i2.298-336

[11] Ashmawy, D. and Reyhani-Masoleh, A. (2021) A Faster Hardware Implementation of the AES S-Box. 2021 *IEEE* 28*th Symposium on Computer Arithmetic* (*ARITH*), Lyngby, 14-16 June 2021, 123-130. https://doi.org/10.1109/ARITH51176.2021.00034

[12] Qin, P.Y., Zhou, F., Wu, N. and Xian, F.C. (2021) A Compact Implementation of AES S-Box Based on Dual Basis. 2021 *IEEE* 4*th International Conference on Electronics Technology* (*ICET*), Chengdu, 7-10 May 2021, 118-122. https://doi.org/10.1109/ICET51757.2021.9451103

[13] Li, Y.J., Zhang, W.G., Ge, Y.D., Huang, Y.T. and Huo, S.S. (2023) Optimized Realization of AES-Like Algorithm S-Box. *Journal of Cryptologic Research*, **10**, 531-538. https://chn.oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFD&dbname=CJFDLAST2023&filename=MMXB202303007&uniplatform=OVERSEA&v=3krAWpsgW0pQFgxPqPGHNXUu9KOVM3LR-yRj4uPmzaytoGewPl1MMY0QmJe5iVlK

[14] Ueno, R., Homma, N., Nogami, Y. and Aoki, T. (2018) Highly Efficient *GF*(2^8) Inversion Circuit Based on Hybrid GF Representations. *Journal of Cryptographic Engineering*, **9**, 101-113. https://doi.org/10.1007/s13389-018-0187-8

[15] Reyhani-Masoleh, A., Taha, M. and Ashmawy, D. (2020) New Low-Area Designs for the AES Forward, Inverse and Combined S-Boxes. *IEEE Transactions on Computers*, **69**, 1757-1773. https://doi.org/10.1109/TC.2019.2922601

[16] Teng, Y.T., Chin, W.L., Chang, D.K., Chen, P.Y. and Chen, P.W. (2022) VLSI Architecture of S-Box with High Area Efficiency Based on Composite Field Arithmetic. *IEEE Access*, **10**, 2721-2728. https://doi.org/10.1109/ACCESS.2021.3139040

[17] Zhong, X.L. and Wu, X.C. (2023) Improved Scheme for AES S-Box and Its Hardware Design. *Application Research of Computers*, **40**, 3784-3788. https://chn.oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFD&dbname=CJFDLAST2024&filename=JSYJ202312041&uniplatform=OVERSEA&v=6V31X-ZMm4dw3aUv6LWSfloYqa4O0wQBBQfFsOG5q8ozVts-sEJtUdkPlBfClBfe

[18] Shen, X.C. and Han, M. (2018) Improved S-box Based on Strict Avalanche Distance Criterion. *Microelectronics & Computer*, **35**, 92-96. https://chn.oversea.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFD&dbname=CJFDLAST2018&filename=WXYJ201806020&uniplatform=OVERSEA&v=y8jwcNYZOk4NEvpQNzq689lZkTI8P8tEKPKjl4d94PoJ4RAsb8iS50lWFfAulS1X

[19] Nakashima, A., Ueno, R. and Homma, N. (2022) AES S-Box Hardware with Efficiency Improvement Based on Linear Mapping Optimization. *IEEE Transactions on Circuits and Systems II*: *Express Briefs*, **69**, 3978-3982.

https://doi.org/10.1109/TCSII.2022.3185632

[20] Lin, S.H., Lee, J.Y., Chuang, C.C., Lee, N.Y., Chen, P.Y. and Chin, W.L. (2023) Hardware Implementation of High-Throughput S-Box in AES for Information Security. *IEEE Access*, **11**, 59049-59058. https://doi.org/10.1109/ACCESS.2023.3284142

[21] Maity, H., Kundu, P., Bhowmik, A. and Barik, A.K. (2023) Input Variable Bypass or IVB Technique for Logic Functions Simplification. 2023 *IEEE Devices for Integrated Circuit* (*DevIC*), Kalyani, 7-8 April 2023, 1-4. https://doi.org/10.1109/DevIC57758.2023.10135020

[22] Mui, E.N., Custom, R. and Engineer, D. (2007) Practical Implementation of Rijndael S-Box Using Combinational Logic. Custom R&D Engineer Texco Enterprise Pvt. Ltd. http://www.geocities.ws/dariuskrail20/Practical_Implementation_of_Rijndael_S-Box_Using_Combinational_Logic.pdf