Proposed Optimized Hardware Implementation for the S-box of the PRESENT Algorithm Using Combinational Logic Circuits

DOI: 10.54654/isj.v3i23.1070

Tran Quang Huy, Do Thi Bac*, Bui Duc Trinh, Le Thi Khanh Linh, Hoang Le Hieu Hao, Duong Phuc Phan

***Abstract*— The lightweight block cipher PRESENT has been standardized by ISO/IEC 29192-2:2012 and TCVN 12854-2:2020. It is a lightweight block cipher with a block size of 64 bits and key sizes of either 80 or 128 bits. For lightweight block ciphers commonly deployed in resource-constrained embedded and IoT devices, resource optimization is a top priority. The S-box, as the only nonlinear component, plays a crucial role in ensuring the security of the cryptographic algorithm by providing resistance against nonlinear and differential attacks. The S-box also consumes the most resources compared to other components of the algorithm, making the optimization of the S-box implementation essential for minimizing the overall resource usage of the algorithm. The S-box of the PRESENT algorithm is used in many other block cipher algorithms. By surveying existing research on PRESENT implementations and analyzing S-box deployment methods based on combinational logic circuits, this paper proposes new architectures for implementing S-boxes using the lowest resource-consuming logic gates, such as 2-input NAND gates, 2-input NOR gates, and NOT gates. The results demonstrate that the proposed methods achieve reduced resources compared to other designs.**

***Tóm tắt*— Thuật toán mã khối hạng nhẹ PRESENT đã được chuẩn hóa bởi ISO/IEC 29192-2:2012 và TCVN 12854-2:2020. Đây là một mã khối hạng nhẹ với kích thước khối là 64 bit và kích thước khóa là 80 hoặc 128 bit. Đối với các mã khối hạng nhẹ thường được triển khai trong các thiết bị nhúng và IoT, việc tối ưu hóa tài nguyên khi triển khai là ưu tiên hàng đầu. S-box là thành phần phi tuyến duy nhất, đóng vai trò quan trọng trong việc đảm bảo tính an toàn của thuật toán mật mã, giúp chống lại các tấn công phi tuyến và vi sai. S-box cũng chiếm lượng tài nguyên lớn nhất khi cài đặt thuật toán so với các thành phần khác, do đó việc tối ưu cài đặt S-box sẽ làm cho tài nguyên tổng thể của cả thuật toán được tối ưu nhất. S-box của thuật toán PRESENT cũng được sử dụng trong nhiều thuật toán mã khối khác. Thông qua việc khảo sát các nghiên cứu liên quan đến cài đặt thuật toán PRESENT, phân tích các phương pháp triển khai S-box dựa trên mạch logic tổ hợp, bài báo này đã đề xuất mới các kiến trúc cài đặt S-box sử dụng các cổng logic tiêu tốn tài nguyên thấp nhất như cổng NAND 2 đầu vào, cổng NOR 2 đầu vào và cổng NOT. Kết quả cho thấy các phương pháp được đề xuất giảm được lượng tài nguyên so với các thiết kế khác.**

Keywords*— *PRESENT; S-box; lightweight block cipher; combitional logic circuit.

Từ khóa*— *PRESENT; hộp thế; mật mã hạng nhẹ; mạch logic tổ hợp.

I. INTRODUCTION

The rapid development of embedded devices and the Internet of Things (IoT) is driving an increasing demand for data security in communication processes. These devices often have limited hardware resources, making the implementation of lightweight cryptographic algorithms essential to secure data without compromising performance. One of the notable lightweight cryptographic algorithms is

This manuscript was received on November 14, 2024. It was reviewed on December 9, 2024, revised on December 16, 2024 and accepted on December 18, 2024.

* Corresponding author

PRESENT, standardized in ISO/IEC 29192-2:2012, offering an optimal encryption solution for resource-constrained devices, especially in IoT applications [1].

Implementing lightweight cryptographic algorithms on hardware requires minimal resource usage and resilience against side-channel attacks [2, 3]. To meet this requirement, optimized circuit designs are needed to minimize area, reduce power consumption, and enhance resistance to attacks.

Among block cipher components, the S-box (Substitution box) is the most critical element, playing a vital role in providing nonlinearity to secure data against cryptographic attacks [4]. Consequently, optimizing S-box design is one of the primary challenges when implementing block ciphers on hardware [5]. The use of combinational logic circuits to implement the S-box is a promising solution to achieve area efficiency, reduce power consumption, and enhance security.

Based on these requirements, this study focuses on implementing the PRESENT cipher's S-box in the form of combinational logic circuits to achieve the optimal trade-off in area and increase resistance to CPA attack. Specifically, the proposed design will employ optimization techniques based on Karnaugh mapping and Boolean expressions to minimize the number of logic gates needed, thereby meeting the system performance criteria.

The main contributions of this paper are as follows:

- First, this paper surveys studies related to the implementation of the PRESENT algorithm and analyzes research on S-box implementation based on combinational logic circuits.

- Second, it proposes a new architecture for S-box design using logic gates with the lowest resource usage, specifically 2-input NAND, 2-input NOR, and NOT gates. The results show that the proposed design achieves the best performance compared to other studies.

The remaining content of this paper is as follows: Section 2 introduces the PRESENT algorithm, Section 3 studies and analyzes the results related to the S-box implementation, Section 4 proposes a new solution for S-box design, and finally, the conclusion.

II. BACKGROUND

PRESENT algorithm encrypts data in 64-bit blocks, making it suitable for applications with low computational power and security requirements. PRESENT supports two main key sizes: 80 bits and 128 bits, depending on the desired security level. It was proposed by Bogdanov et al. [1] in the year 2007 and released as the standard for lightweight cryptography under ISO/lEC29192–2.

The overall flow of the PRESENT algorithm can be observed in Figure 1.

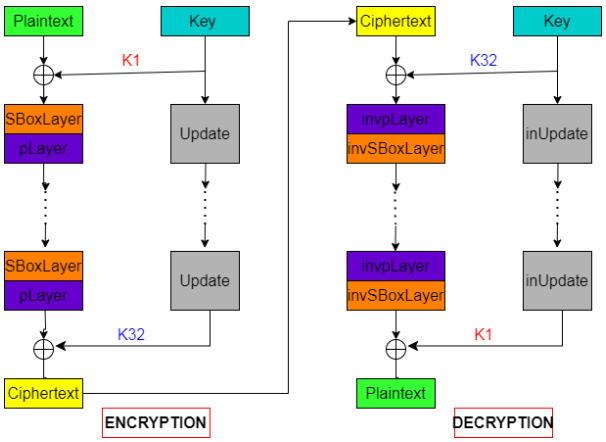

Figure 1. PRESENT Algorithm

The encryption process in the PRESENT algorithm consists of three main steps in each round: AddRoundKey, S-box layer, and Permutation layer. Below is a detailed description of each step:

- AddRoundKey: In this step, the current state of the data is XORed with the current round key. The round key is derived from the main key (master key) through the key scheduling process. The XOR with the round key creates a layer of key-based protection, ensuring that each encryption round is distinct, even if the input data remains unchanged.

- S-box layer: Following the AddRoundKey step, each nibble (4 bits) of the state is substituted with a corresponding value from the S-box table. This substitution layer

introduces non-linearity, an essential aspect of securing the algorithm against cryptographic attacks, such as linear and differential attacks. Each nibble of the state is replaced with a different value to increase complexity and obfuscate the data, making it more resistant to cryptanalysis.

- Permutation layer: After the S-box layer, each encryption round concludes with a permutation step. In this step, the bits of the state are rearranged according to a predefined order. Permutation diffuses minor changes in the input across the state, ensuring that every minor modification affects the entire state after several rounds. This high diffusion enhances PRESENT's resistance against attacks by spreading the data across all bits in the state.

The encryption process is repeated over 31 rounds to ensure adequate diffusion and non-linearity, followed by a final AddRoundKey step to complete the encryption process.

The decryption process in PRESENT reverses the encryption steps and includes inverse AddRoundKey, inverse S-box layer, and inverse Permutation layer. Specifically:

- In the inverse AddRoundKey step, the current round key is XORed with the current state to reverse the key's effect from the encryption process.

- Next, each nibble in the state is replaced with the corresponding value from the inverse S-box table, undoing the substitution performed during encryption.

- Finally, the inverse permutation layer restores the original bit order, reversing the bit rearrangements made during encryption. These inverse steps ensure accurate recovery of the original data, preserving the symmetry in the encryption and decryption processes.

The S-box in PRESENT is a non-linear lookup table where each nibble (4 bits) of the state is replaced with a corresponding value to introduce non-linearity into the algorithm. This is crucial for protecting data from cryptographic attacks such as linear and differential attacks, as the S-box creates significant output differences when there are minor changes in the input. The inverse S-box is used in the decryption process to reverse the changes from the encryption process, ensuring that data can be fully restored. The S-Box and inverse S-box are given in Table 1 and Table 2, respectively.

TABLE 1. S-BOX OF PRESENT

x	0	1	2	3	4	5	6	7
$S(x)$	C	5	6	B	9	0	A	D
x	8	9	A	B	C	D	E	F
$S(x)$	3	E	F	8	4	7	1	2

TABLE 2. INVERSE S-BOX OF PRESENT

x	0	1	2	3	4	5	6	7
$S^{-1}(x)$	5	E	F	8	C	1	2	D
x	8	9	A	B	C	D	E	F
$S^{-1}(x)$	B	4	6	3	0	7	9	A

The permutation in Table 3 provides data diffusion across multiple rounds, creating strong diffusion in the encrypted data. In the Permutation layer, the bits of the state are rearranged according to a specific order. This rearrangement ensures that minor changes in the input will affect the entire state after multiple encryption rounds, creating a powerful layer of security against cryptographic attacks. The inverse permutation in Table 4 is used during decryption to reverse the permutations applied during encryption. The inverse permutation process restores the original bit order, ensuring accurate recovery of the original input state.

TABLE 3. PERMUTATION OF PRESENT

i	0	1	2	3	4	5	6	7
$P(i)$	0	16	32	48	1	17	33	49
i	8	9	10	11	12	13	14	15
$P(i)$	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23
$P(i)$	4	20	36	52	5	21	37	53
i	24	25	26	27	28	29	30	31

$P(i)$	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39
$P(i)$	8	24	40	56	9	25	41	57
i	40	41	42	43	44	45	46	47
$P(i)$	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55
$P(i)$	12	28	44	60	13	29	45	61
i	56	57	58	59	60	61	62	63
$P(i)$	14	30	46	62	15	31	47	63

TABLE 4. INVERSE PERMUTATION OF PRESENT

i	0	1	2	3	4	5	6	7
$P^{-1}(i)$	0	4	8	12	16	20	24	28
i	8	9	10	11	12	13	14	15
$P^{-1}(i)$	32	36	40	44	48	52	56	60
i	16	17	18	19	20	21	22	23
$P^{-1}(i)$	1	5	9	13	17	21	25	29
i	24	25	26	27	28	29	30	31
$P^{-1}(i)$	33	37	41	45	49	53	57	61
i	32	33	34	35	36	37	38	39
$P^{-1}(i)$	2	6	10	14	18	22	26	30
i	40	41	42	43	44	45	46	47
$P^{-1}(i)$	34	38	42	46	50	54	58	62
i	48	49	50	51	52	53	54	55
$P^{-1}(i)$	3	7	11	15	19	23	27	31
i	56	57	58	59	60	61	62	63
$P^{-1}(i)$	35	39	43	47	51	55	59	63

The key schedule in PRESENT is designed to generate distinct round keys from the master key, thereby enhancing the algorithm's security. This key scheduling process generates a distinct round key for each encryption round, ensuring the security of the PRESENT algorithm. Illustrations of this process can clarify how the round keys are derived from the master key, facilitating secure and efficient implementation and verification.

III. RELATED WORKS

A traditional method for implementing the S-box in block ciphers, including the PRESENT cipher, is to use a lookup table (LUT). In this approach, input values are mapped directly to output values by referencing a pre-stored table. For the PRESENT S-box, the LUT contains predefined mappings for 4-bit input-output pairs, allowing fast access speed and easy implementation. Kavun et al. [6] proposed a RAM-based FPGA implementation of PRESENT. The authors explained that the memory is used in a ping-pong fashion to store the partially encrypted data instead of using registers. Storing the S-box in RAM as LUT, however, provided undesirable results due to complex control logic. The LUT approach can be a practical choice in scenarios where simplicity and ease of implementation are prioritized, and hardware resources are not highly constrained. However, in systems requiring high optimization, methods based on Boolean expressions or sub-expression sharing are often superior for achieving a balance between performance, area, and power consumption.

The study [7] focuses on optimizing the area and power consumption of both the S-box and inverse S-box of the PRESENT cipher. The authors employ sub-expression sharing and factorization techniques to reduce the number of logic gates and hardware area. They propose two optimization techniques, PD-I and PD-II, to achieve this goal:

PD-I: In this approach, the S-box is designed by transforming Boolean expressions and sharing common logic gates. Boolean transformations simplify the expressions in the S-box, utilizing a limited number of common logic gates. As a result, after optimization, the PD-I requires 14 2-input AND gates, 7 2-input OR gates, 5 2-input XOR gates, and 4 NOT gates. This method also significantly reduces power consumption due to the decrease in active gates.

PD-II: Building on PD-I, this approach further optimizes the design using a two-level logic structure, enabling the S-box to share more common components. The PD-II consumes, 20 2-input AND gates, 12 2-input OR gates, 2 2-input XOR gates, and 6 NOT gates. The sub-expression sharing and factorization methods in this study significantly reduce the area and power consumption of the S-box and inverse S-box, making them

particularly useful for embedded applications requiring maximum resource efficiency.

The research in [8] introduces a method of constructing the S-box using Boolean expressions instead of relying on LUTs or BRAM for storage. This approach conserves hardware area and allows for further optimization through expression analysis. The implementation process consists of two main steps: Use of Karnaugh Maps: The authors first use Karnaugh maps to minimize the Boolean expressions for each output bit of the S-box. Specifically, these expressions are presented in the sum of products (SOP) form, using AND and OR gates instead of traditional LUTs. This reduces the required hardware elements and enhances customization options. Factorization Analysis: After simplification using Karnaugh maps, the Boolean expressions are further optimized through factorization analysis. Common terms in the expressions are separated, thereby reducing the number of required logic gates. Before factorization, this design required 45 AND gates and 17 OR gates. After optimization, this number was reduced to 26 AND gates and 17 OR gates, significantly saving hardware area.

The final design only requires 62 slices when implemented on the Xilinx Virtex-5 XC5VLX50 FPGA, making it one of the most compact S-box designs for the PRESENT cipher. Notably, this design maintains high performance with a throughput of 51.32 Mbps at a maximum frequency of 236.574 MHz. At the standard frequency of 13.56 MHz, the throughput reaches 2.94 Mbps, meeting the security requirements for IoT and wireless sensor applications.

Enhanced S-box Structure: The minimized S-box only utilizes simple logic gates (AND, OR, NOT), without relying on LUT or memory, conserving hardware area and reducing delay. Loop Unrolling: This technique enhances throughput by replacing loops with duplicates of loop bodies, allowing parallel execution of encryption rounds and reducing processing time. Experimental results show that the new optimized S-box structure significantly reduces the number of logic gates and achieves low delay, enhancing throughput when implemented on FPGA.

The study by Rashidi et al in [9] optimizes the PRESENT S-box through Karnaugh mapping and logic expression minimization, reducing hardware resources and delay when implemented on FPGA. Karnaugh Mapping and Expression Minimization: The authors use Karnaugh maps to minimize the Boolean expressions for each output of the S-box, aiming to reduce the number of required logic gates. Karnaugh mapping helps identify groups of variables that can be combined and eliminates redundant elements, achieving an optimized structure. The results show that the S-box designed in this paper includes 7 XOR gates, 13 AND gates, 6 OR gates, and 4 NOT gates.

Optimization methods for the PRESENT cipher's S-box represent a critical area of research to meet the security needs of IoT devices and resource-constrained embedded systems. The methods of sub-expression sharing and factorization, Boolean structure, and logic optimization via Karnaugh maps have demonstrated outstanding effectiveness in conserving hardware area and reducing power consumption. Each method has unique advantages in PRESENT cipher optimization, and they hold potential for combination to create comprehensive, optimal designs in the future. This study also approaches logic function optimization after transforming the Boolean functions of S-boxes. However, after optimizing each function from the Karnaugh maps, the design will proceed with splitting to optimize the circuit while sharing common elements across all four Boolean functions of the S-box substitution box.

IV. Proposed Implementation Method

In this study, we propose various designs using low-area logic gates that are technology-independent, allowing for easier evaluation of hardware resources. Table 6 shows the gate area across different design technologies. All gates are converted to NAND-equivalent gates [10].

Designing logic circuits using NAND and NOR gates is an optimal approach in embedded

TABLE 5. S-BOX TABLE

Input (hexa)	x_3	x_2	x_1	x_0	f_3	f_2	f_1	f_0	**Output (hexa)**
0	0	0	0	0	1	1	0	0	0C
1	0	0	0	1	0	1	0	1	05
2	0	0	1	0	0	1	1	0	06
3	0	0	1	1	1	0	1	1	0B
4	0	1	0	0	1	0	0	1	09
5	0	1	0	1	0	0	0	0	00
6	0	1	1	0	1	0	1	0	0A
7	0	1	1	1	1	1	0	1	0D
8	1	0	0	0	0	0	1	1	03
9	1	0	0	1	1	1	1	0	0E
A	1	0	1	0	1	1	1	1	0F
B	1	0	1	1	1	0	0	0	08
C	1	1	0	0	0	1	0	0	04
D	1	1	0	1	0	1	1	1	07
E	1	1	1	0	0	0	0	1	01
F	1	1	1	1	0	0	1	0	02

systems and lightweight cryptography, where hardware area and power efficiency are critical factors. NAND and NOR gates are known as "universal gates," meaning they can be used to create any other logic operation, including AND, OR, and NOT. This provides flexibility in circuit design, allowing designers to optimize the S-box without requiring additional types of logic gates, thus reducing area and hardware resources.

TABLE 6. COMPARISONS OF SEVERAL STANDARD CELL LIBRARIES FOR TYPICAL COMBINATORIAL CELLS

Library	**Technology**	**NAND /NOR**	**NOT**	**XOR/ XNOR**	**AND /OR**
UMC	180nm	1.00	0.67	3.00	1.33
SXLB	130nm	1.00	0.75	2.25	1.25
TSMC	65nm	1.00	0.50	3.00	1.50
NandGate	45nm	1.00	0.67	2.00	1.33
NandGate	15nm	1.00	0.75	2.25	1.50

Using NAND and NOR gates in S-box design offers several specific benefits:

Area Optimization: In integrated circuits and FPGAs, NAND and NOR gates occupy less area and have simpler structures compared to complex gates. This helps reduce the overall area of the S-box, making it suitable for applications with hardware size constraints.

Power Efficiency: NAND and NOR gates consume less power when performing logic operations, thereby saving energy for the entire system. This is especially valuable in IoT applications and mobile devices.

Resistance to Attacks: By minimizing the number of gates and creating circuits with

$$f_3 = (\overline{x_3}\,\overline{x_1}\,\overline{x_0}) + (x_3\,\overline{x_2}\,x_0) + (x_3\,\overline{x_2}\,x_1) + (\overline{x_3}\,x_1\,x_0) + (\overline{x_3}\,x_2\,x_1),$$

$$f_2 = (\overline{x_3}\,x_2\,x_1x_0) + (\,\overline{x_2}\,x_1\,\overline{x_0}) + (x_3\,x_2\,\overline{x_1}\,) + (\overline{x_3}\,\overline{x_2}\,\overline{x_1}) + (\overline{x_2}\,\overline{x_1}\,x_0),$$

$$f_1 = (\overline{x_3}\,\overline{x_2}\,x_1) + (\overline{x_3}\,x_1\,\overline{x_0}) + (x_3\,x_2\,x_0) + (x_3\,\overline{x_2}\,\overline{x_1}) + (x_3\,\overline{x_2}\,\overline{x_0}),$$

$$f_0 = (\overline{x_3}\,\overline{x_2}\,x_0) + (\overline{x_3}\,x_2\,\overline{x_1}\,\overline{x_0}) + (\overline{x_3}\,x_1\,x_0) + (x_3\,\overline{x_2}\,\overline{x_0}) + (x_3\,x_2\,\overline{x_1}\,x_0) + (x_3\,x_1\,\overline{x_0}).$$

Figure 2. The simplified functions are represented as a sum of products SOP

consistent execution times, NAND and NOR-based S-box designs can reduce leakage of side-channel information, such as power consumption, thereby enhancing resistance to power analysis attacks.

Designing an S-box with NAND and NOR gates requires Boolean expression optimization, often achieved through Karnaugh maps and factorization techniques. This ensures that the logic expressions are maximally simplified, using as few logic gates as possible while maintaining the necessary nonlinearity and security properties of the S-box. The result is an efficient and compact S-box design, well-suited to the stringent requirements of lightweight cryptography in embedded and IoT devices.

$$f_3 = \overline{\overline{x_3}\,\overline{x_2}\,x_1}\;\overline{\overline{x_2}\,\overline{x_1}\,\overline{f_0}}\;\overline{x_2\,\overline{x_0}\,f_1}\;\overline{\overline{x_3}\,f_0}\;\overline{\overline{x_2}\,\overline{f_1}},$$

$$f_2 = \overline{\overline{x_2}\,x_1\,\overline{x_0}}\;\overline{x_3\,x_2\,\overline{x_1}}\;\overline{\overline{x_2}\,\overline{x_1}\,\overline{f_0}}\;\overline{x_0\,\overline{f_1}f_0},$$

$$f_1 = \overline{x_3\,\overline{\overline{x_2}\,\overline{x_1}}\;\overline{x_2\,x_0}}\;\overline{\overline{x_3}\,x_1\,f_0}\;\overline{\overline{x_2}\,x_1\,f_0},$$

$$f_0 = \overline{x_3\,\overline{\overline{x_0}\,\overline{x_2}\,\overline{x_1}}\;\overline{x_0\,x_2\,\overline{x_1}}}\;\overline{\overline{x_3}\,\overline{x_0}\,\overline{\overline{x_2}\,\overline{x_1}}}\;\overline{x_0\,x_2\,\overline{x_1}}.$$

Figure 3. Coordinate Boolean functions of the S-box are represented in terms of 2-input NAND logic gates

In this paper, we propose three combinational logic circuit designs for the PRESENT S-box. The designs utilize only a single type of 2-input NAND gate, a single type of 2-input NOR gate, and a combination design using three resource-efficient gate types: 2-input NAND, 2-input NOR, and NOT. The reason for choosing 2-input NAND and 2-input NOR gates is that they are equivalent to 1 GE and are independent of any technology. First, the S-box is represented as a table of output values based

on input values, as shown in Table 5. From this, using the Karnaugh map method, we simplify the Boolean functions of the S-box, as illustrated in Figure 2.

$$f_3 = \overline{x_3 + \overline{f_0}},$$

$$f_2 = \overline{\overline{f_0 + x_2 + x_1} + \overline{\overline{x_3 + \overline{x_2} + x_1}}},$$

$$f_1 = \overline{x_3 + \overline{x_1} + f_0},$$

$$f_0 = \overline{\overline{x_3} + \overline{\overline{x_2} + x_1 + x_0} + \overline{\overline{x_2} + x_1 + \overline{x_0}}} + \overline{x_3 + \overline{\overline{x_2} + x_1 + x_0} + \overline{\overline{x_2} + x_1 + \overline{x_0}}}.$$

Figure 4. Coordinate Boolean functions of the S-box are represented in terms of 2input NOR logic gates

In this paper, we follow the general principles outlined below for our proposed optimizations: Simplify each individual Boolean function.

- Evaluate the four Boolean functions collectively to identify common terms involving 2, 3, or 4 shared variables. These shared terms will be reused in the circuit.
- Re-transform the simplified functions using Boolean algebra transformations to create shared components. This process requires extensive experimentation.
- All Boolean functions can fundamentally be implemented using a single type of gate, such as NAND or NOR. This principle is widely applied in digital electronics and IC design.
- The transformed functions are rigorously evaluated and verified by testing all possible inputs and comparing the outputs.

$$f_3 = \overline{x_3\, x_2\, \overline{x_1}\, \overline{x_2 + x_1}\, \overline{f_0}\, \overline{x_2 + x_0}\, f_1 + \overline{x_3}\, f_0 + \overline{f_1} + x_2},$$

$$f_2 = \overline{x_3\, x_2\, \overline{x_1}\, \overline{f_0}\, \overline{x_2 + x_1}\, \overline{\overline{x_2}\, x_1 + x_0} + \overline{f_0} + \overline{\overline{f_1} x_0}},$$

$$f_1 = \overline{\overline{x_3}\, x_1 + f_0 + \overline{x_2 + x_1} + \overline{x_2} + \overline{x_1} + \overline{x_3}\, \overline{x_2}\, x_1\, f_0},$$

$$f_0 = \overline{(x_3 + \overline{x_2\, \overline{x_1}\, \overline{x_0}\, x_2\, \overline{x_1}\, x_0})\, \overline{x_3}\, \overline{x_2\, \overline{x_1}\, \overline{x_0}\, x_2\, \overline{x_1}\, x_0}}.$$

Figure 5. Coordinate Boolean functions of the S-box are represented in terms of 2-input NAND, 2-input NOR, and NOT logic gates

Applying the above principles, we propose the following recommendations:

- Proposed 1: The design utilizes only a single type of 2-input NAND gate. From the basic Boolean formulas of the S-box functions in Figure 2, the functions are transformed into logical equations consisting of 2-input NAND gates, as shown in Figure 3. The corresponding combinational logic circuit is shown in Figure 6. The results indicate that 53 2-input NAND gates are required, corresponding to 53 GE.
- Proposed 2: The design utilizes only a single type of 2-input NOR gate. From the basic Boolean formulas of the S-box functions in Figure 2, the functions are transformed into logical equations consisting of 2-input NOR gates, as shown in Figure 4. The corresponding combinational logic circuit is shown in Figure 7. The results indicate that 55 2-input NOR gates are required, corresponding to 55 GE.
- Proposed 3: The design utilizes a combination of three types of gates: 2-input NAND, 2-input NOR, and NOT gates. From the basic Boolean formulas of the S-box functions, the functions are transformed into logical equations consisting of 2-input NAND, 2-input NOR, and NOT gates. The formulas are shown in Figure 5, and the corresponding combinational logic circuit is shown in Figure 8. The results indicate that 24 2-input NAND gates, 14 2-input NOR gates, and 12 NOT gates are required. The corresponding number of GE depends on the technology, as NOT gates occupy different areas in different technologies. Typically, the NOT gate area occupies 1/2, 2/3, or 3/4 of 1 GE, as mentioned previously in Table 6.

The equations and circuits can be easily verified by substituting the input values of the S-box from 0000 to 1111 and checking the corresponding outputs. These values are exact matches to those in the S-box (Table 1) of the PRESENT algorithm.

Table 7 presents a comparison of different S-box designs in terms of area cost and gate equivalence. This table includes various designs, including three proposed designs (Proposed 1, Proposed 2, Proposed 3) and reference designs from works PD-I, PD-II in [7], and Design [8], Design [9]. The "Area Cost" section lists the number of different types of logic gates (AND, OR, NOT, NAND, NOR, XOR) used in each design. For instance, the Proposed 1 design uses

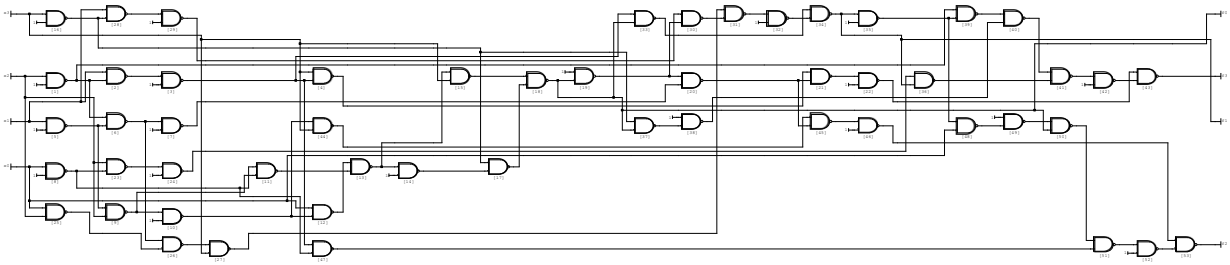

Figure 6. Proposed 1: Represent the S-box architecture as a circuit of 2-input NAND gates

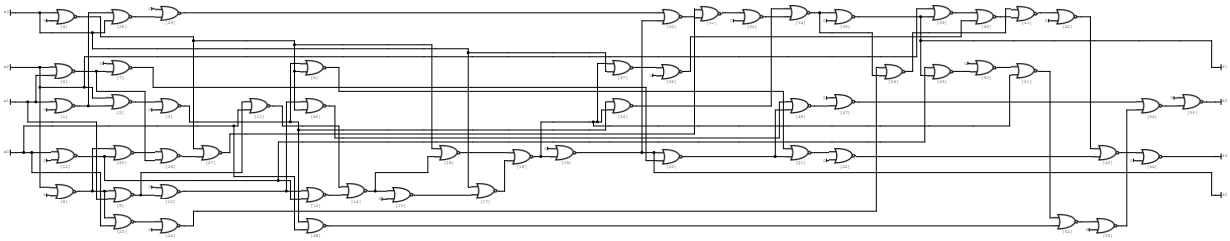

Figure 7. Proposed 2: Represent the S-box architecture as a circuit of 2-input NOT gates

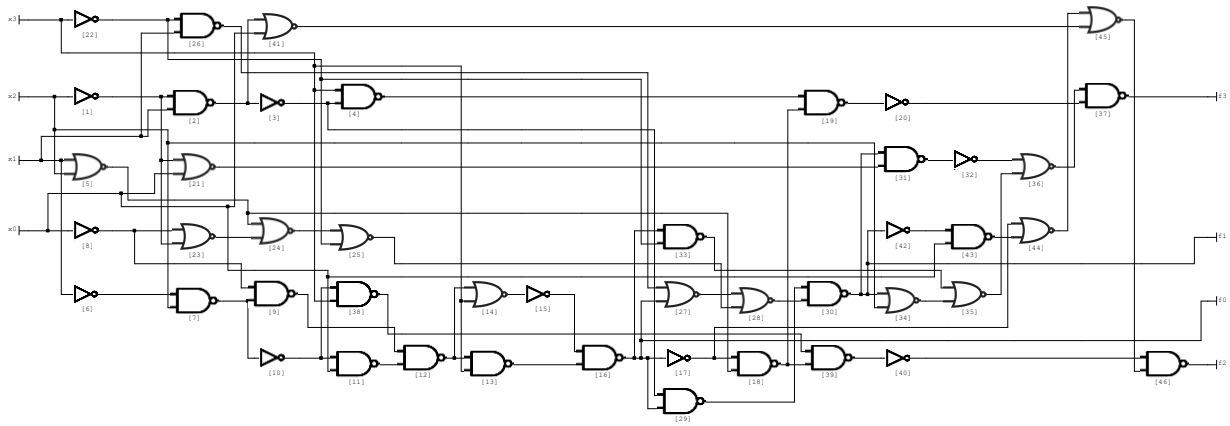

Figure 8. Proposed 3: Represent the S-box architecture as a circuit combining 2-input NAND, 2-input NOR and NOT gates

TABLE 7. COMPARISON OF S-BOX DESIGNS IN TERMS OF AREA COST

S-Boxes Design	Area cost						Gate Equivalence				
	AND	OR	NOT	NAND	NOR	XOR	UMC(180nm)	SxLib(130nm)	TSMC(65nm)	NandGate(45nm)	NandGate(15nm)
Proposed 1	0	0	0	53	0	0	53.00	53.00	53.00	53.00	53.00
Proposed 2	0	0	0	0	55	0	55.00	55.00	55.00	55.00	55.00
Proposed 3	0	0	12	20	14	0	42.04	43.00	40.00	42.04	43.00
PD-I [7]	14	7	4	0	0	5	45.61	40.50	48.50	40.61	45.75
PD-II [7]	20	12	6	0	2	0	52.58	49.00	57.00	50.58	57.00
Design [8]	17	26	4	0	0	0	62.53	59.25	69.50	62.53	70.50
Design [9]	13	6	4	0	0	7	48.95	42.50	51.50	41.95	47.25

only 53 NOT gates, while the Proposed 2 design uses 55 NOT gates. The "Gate Equivalence" section represents the gate count equivalence based on different manufacturing technologies, including UMC (180nm), SxLib (130nm), TSMC (65nm), and NAND gates at 45nm and 15nm sizes. The values in this column show the relative area cost for each design when implemented on different semiconductor technologies. For example, the Proposed 3 design has an equivalence of 43.00 gates on SxLib (130nm) technology and 40.00 gates on TSMC (65nm), indicating a slight difference in area cost depending on the technology used.

Based on Table 7, we have the following observations:

- Proposed 1 and Proposed 2 exhibit low area costs of 53 and 55 GE, respectively, across all platforms, including UMC (180nm), SxLib (130nm), TSMC (65nm), and NandGate (45nm and 15nm). This consistency in area cost across technology platforms is a distinct advantage.
- Overall, the design in Proposed 3 is the best in terms of resource efficiency compared to the other designs, particularly for three popular technologies such as UMC (180nm), TSMC (65nm), and NandGate (15nm). These improvements have the potential to enhance the performance of cryptographic systems in applications requiring high security but limited resources, such as IoT devices and mobile devices.

The proposed designs are detailed down to the level of logic gates, making them suitable for designing the S-box in particular and the PRESENT algorithm in general across various CHIP design technologies. Each technology has a different area cost for the gate equivalent (GE), leading to varying area costs depending on the chosen technology.

IV. Conclusion

In this work, we studied the lightweight block cipher algorithm PRESENT, standardized by ISO/IEC 29192-2:2012 and TCVN 12854-2:2020, for its applicability in embedded and IoT devices due to the need for efficient resource optimization. The S-box, as the only nonlinear component, plays a critical role in ensuring security against nonlinear and differential attacks, while also consuming the most resources during implementation. By analyzing existing S-box designs and proposing new combinational logic structures, we developed optimized S-box implementations using three designs with 2-input NAND gates, 2-input NOR gates, and NOT gates, achieving a significant reduction in area cost compared to previous designs. Our findings contribute to enhancing the implementation efficiency of the PRESENT algorithm in systems with limited resources, especially in IoT and embedded devices.

References

[1] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Proceedings of CHES 2007*, vol. 4727, pp. 450–466, 2007.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO'99, Lecture Notes in Computer Science*, vol. 1666. Springer, Berlin, Heidelberg, pp. 388–397, 1999.

[3] A. F. Echevarría, R. R. Aulet, and A. G. Gómez, "On some relations of sca-related properties of S-box under the hamming weight leakage mode," *Journal of Science and Technology on Information Security*, vol. 3, no. 17, pp. 3–9, 2023.

[4] D.P. Phan, N. H. Minh, D. B. Anh, K. D. N. Binh, T.T. Ha, H. T. Thuc, and P. C. Kha, "Construction of robust lightweight s-boxes using enhanced logistic and enhanced sine maps," IEEE Access, vol. 12, pp. 63976-63994, 2024.

[5] N. V. Long and L. D. Duc, "Đề xuất s-hộp có tính chất mật mã tốt cho hoán vị của hàm băm keccak," *Journal of Science and Technology on Information Security*, vol. 1, no. 11, pp. 32–45, 2020.

[6] E. B. Kavun and T. Yalcin, "Ram-based ultralightweight fpga implementation of present," in *2011 International Conference on Reconfigurable Computing and FPGAs*. IEEE, pp. 280–285, 2011.

[7] M. R and N. K. V, "Optimized implementation of s-box and inverse s-box for present lightweight block cipher," in *2023 2nd International Conference on Vision Towards Emerging Trends in Communication and Networking Technologies (ViTECoN)*, pp. 1–5, 2023.

[8] J. J. Tay, M. L. D. Wong, M. M. Wong, C. Zhang, and I. Hijazin, "Compact fpga implementation of present with boolean s-box," in *2015 6th Asia Symposium on Quality Electronic Design (ASQED)*, pp. 144–148, 2015.

[9] B. Rashidi, "Efficient and high-throughput application specific integrated circuit implementations of hight and present block ciphers," *IET Circuits, Devices & Systems*, vol. 13, no. 6, pp. 731–740, 2019.

[10] T. Peyrin, "Lightweight symmetric-key cryptography." *Suzdal, Russia: CTCRYPT 2018*, pp. 8-10, May 29th 2018.

ABOUT THE AUTHORS

Tran Quang Huy

Workplace: Thai Nguyen University of Information and Communication Technology (ICTU).
Email: tqhuy@ictu.edu.vn.
Education: He received the B.Sc. degrees in the Thai Nguyen University of Information and Communication Technology (ICTU), in 2013. In 2016, he earned an M.Sc. degree from the VNU University of Engineering and Technology. He is currently pursuing a Ph.D. degree in computer science at ICTU.
Recent research direction: cryptography, system security, and wireless networks.

Tên tác giả: Trần Quang Huy
Cơ quan công tác: Trường Đại học Công nghệ thông tin và Truyền thông Thái Nguyên.
Email: tqhuy@ictu.edu.vn.
Quá trình đào tạo: Tốt nghiệp đại học năm 2013 và thạc sĩ năm 2016. Hiện đang là nghiên cứu sinh tại Đại học công nghệ thông tin và truyền thông Thái Nguyên.
Hướng nghiên cứu hiện nay: Mật mã, bảo mật hệ thống, các mạng không dây.

Do Thi Bac

Workplace: Thai Nguyen University of Information and Communication Technology (ICTU).
Email: dtbac@ictu.edu.vn.
Education: Ph.D. from Le Quy Don Technical University in 2014.
Recent research direction: cryptography, communication, and network security.

Tên tác giả: **Đỗ Thị Bắc**
Cơ quan công tác: Trường Đại học Công nghệ thông tin và Truyền thông, Đại học Thái Nguyên.
Email: dtbac@ictu.edu.vn.
Quá trình đào tạo: Nhận bằng Tiến sĩ tại Đại học Lê Quý Đôn năm 2014.
Hướng nghiên cứu hiện nay: Mật mã, truyền thông, an ninh mạng.

Bui Duc Trinh

Workplace: Academy of Cryptography Techniques, Hanoi, Vietnam.
Email: buiductrinh@actvn.edu.vn.
Education: He received a B.Sc. in Cryptographic Engineering from the Academy of Cryptography Techniques (2002), an IT Engineering degree from Hanoi University of Science and Technology (2005), an M.Sc. in Cryptographic Engineering (2008), and a Ph.D. in Information Security from FSO Academy, Russia (2013).
Recent research direction: His research interests include cryptography, IoT design, and hardware security.

Tên tác giả: **Bùi Đức Trình**
Cơ quan công tác: Học viện Kỹ thuật mật mã.
Email: buiductrinh@actvn.edu.vn.
Quá trình đào tạo: Nhận bằng kỹ sư tại Học viện Kỹ thuật mật mã năm 2002, bằng kỹ sư Đại học Bách Khoa năm 2005. Tốt nghiệp thạc sĩ Học viện Kỹ thuật mật mã năm 2008, nhận bằng Tiến sĩ của Học viện FSO, Liên Bang Nga, năm 2013.
Hướng nghiên cứu hiện nay: Mật mã, IoT, bảo mật phần cứng.

Le Thi Khanh Linh

Workplace: currently a final-year student specializing in Embedded Systems and Automatic Control at the Academy of Cryptography Techniques.
Email: khanhlinh180102@gmail.com.
Recent research direction: Embedded System Security and IoT.

Tên tác giả: **Lê Thị Khánh Linh**
Cơ quan công tác: Học viện Kỹ thuật Mật mã.
Email: khanhlinh180102@gmail.com.
Quá trình đào tạo: Sinh viên năm 5, Học viện Kỹ thuật mật mã.
Hướng nghiên cứu hiện nay: Bảo mật hệ thống nhúng, IoT.

Hoang Le Hieu Hao

Workplace: currently a final-year student specializing in Embedded Systems and Automatic Control at the Academy of Cryptography Techniques.
Email: hoanghao18vm@gmail.com.
Recent research direction: Embedded System Security, IoT.

Tên tác giả: **Hoàng Lê Hiếu Hảo**
Cơ quan công tác: Học viện Kỹ thuật Mật mã.
Email: hoanghao18vm@gmail.com.
Quá trình đào tạo: Sinh viên năm 5, Học viện Kỹ thuật mật mã.
Hướng nghiên cứu hiện nay: Bảo mật hệ thống nhúng, IoT.

Duong Phuc Phan

Workplace: The University of Electro-Communications (UEC), Tokyo, Japan.
Email: duongphucphan@vlsilab.ee.uec.ac.jp
Education: He received the B.Sc. and M.S. degrees in 2011 and 2014, respectively. He is currently pursuing a Ph.D. degree in information and network engineering at The University of Electro-Communications (UEC), Tokyo, Japan.
Recent research direction: cryptography, embedded systems design, and hardware security.

Tên tác giả: **Dương Phúc Phần**
Cơ quan công tác: Đại học UEC, Nhật Bản.
Email: duongphucphan@vlsilab.ee.uec.ac.jp
Quá trình đào tạo: Nhận bằng đại học và thạc sĩ năm 2011 và 2014. Hiện nay là nghiên cứu sinh Tiến sĩ tại đại học UEC, Nhật Bản.
Hướng nghiên cứu hiện nay: Mật mã, bảo mật hệ thống nhúng, IoT.