Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 10, October 2019, pp. 2589-2596
DOI: 10.23940/ijpe.19.10.p4.25892596

Improving the Performance of Multi-Mode SM4 Block Cipher

Guangyong Hu* and Rui Chen

School of Computer and Software Engineering, Nanjing Institute of Industry Technology, Nanjing, 210023, China

Abstract

In this paper, a low-cost multi-mode hardware architecture is proposed for data confidentiality of resource-constrained IoT (Internet of things) devices. The proposed architecture adopts resource sharing technologies to reduce the number of s-boxes, eliminate memory requirement of expanded keys, and realize flexible reconfigurability of various operation modes. The FPGA implementation results show that only 1,326 LUTs and 1,000 registers are needed at 50MHz. Compared with related works, resource costs are reduced significantly.

Keywords: resource-constrained; internet of things; block cipher; SM4; FPGA

(Submitted on August 31, 2019; Revised on September 20, 2019; Accepted on September 25, 2019)

1. Introduction

In recent years, IoT (Internet of thing) devices have developed rapidly. It is estimated that by 2020, the number of global IoT devices will reach 26 billion, and the IoT market will exceed 1.9 trillion US dollars [1]. As more and more devices connect to the Internet, new opportunities grow to exploit potential security vulnerabilities [2]. IoT devices are low-cost while lacking enough resources to ensure security.

Security of IoT devices can be categorized into six domains: confidentiality, integrity, availability, non-repudiation, authorization, and privacy [3]. This paper will focus on confidentiality while maintaining flexibility. To meet the requirement of confidentiality, symmetric-key cryptography will be used. SM4, which is released by the Office of State Commercial Cipher Administration of China, can be considered a symmetric-key cryptography option for resource-constrained IoT devices. The following reasons can account for this [4]: 1) security characteristics of SM4 are equivalent to those of AES-128; 2) the encryption and decryption have the same structure; 3) the encryption and decryption have the same s-box; and 4) only four s-boxes (each with 256×8-bit) are needed in one round function, while 16 are needed for AES.

SM4 can be implemented with software or hardware. Due to the potential vulnerabilities in the IoT applications or operating system, implementation with software may be easily cracked. Therefore, it is necessary to implement the SM4 in hardware instead of software. Many related works have studied hardware architecture of SM4, but all of them are limited to a single operation mode. For example, references [5-10] only implemented the ECB mode, references [11-12] only implemented the CBC mode, reference [13]only implemented the CTR mode, and reference [14] only implemented the XTS mode. Different from existing works, a low-cost multi-mode hardware architecture of the SM4 algorithm is proposed to provide both flexibility and confidentiality. By fully exploiting the resources that can be shared in the algorithm, the electronic codebook mode (ECB), cipher block chaining mode (CBC), cipher feedback mode (CFB), output feedback mode (OFB), and counter mode (CTR) are supported.

2. Review of SM4 Algorithm

2.1. SM4 Encryption/Decryption

SM4 consists of encryption, decryption, and the key expansion algorithm. The data block and key length are both 128 bits.

* Corresponding author.
E-mail address: hugy@niit.edu.cn

The encryption and decryption algorithm and the key expansion algorithm use a 32-round nonlinear iterative structure. The structure of the encryption algorithm is exactly the same as that of the decryption algorithm, except that the order of the round keys of the two algorithms is reversed [4]. The SM4 encryption and decryption algorithm consists of an iterative process and a reverse transform. Taking the encryption algorithm as an example, if the plaintext is $(X_0, X_1, X_2, X_3) \in (Z_2^{32})^4$, ciphertext is $(Y_0, Y_1, Y_2, Y_3) \in (Z_2^{32})^4$, and round key is $rk_i \in Z_2^{32}, i = 0,1,2,\cdots,31$, then the iterative process of the SM4 encryption algorithm can be expressed as

$$X_{i+4} = X_i \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i), i = 0,1,2,\cdots,31 \quad (1)$$

In Equation (1), T(.)=L(τ(.)), where L(.) is a linear transform and $\tau(.)$ is a nonlinear transform. The nonlinear transform $\tau(.)$ can be defined as follows: set input $A = (a_0, a_1, a_2, a_3) \in (Z_2^8)^4$ and output $B = (b_0, b_1, b_2, b_3) \in (Z_2^8)^4$, then

$$B = (b_0, b_1, b_2, b_3) = \tau(A) = \big(S(a_0), S(a_1), S(a_2), S(a_3)\big) \quad (2)$$

Where S(.) is the s-box lookup function. The linear transform L(.) can be defined as follows: set input $B \in Z_2^{32}$ and output $C \in Z_2^{32}$, then

$$C = L(B) = B \oplus (B << 2) \oplus (B << 10) \oplus (B << 18) \oplus (B << 24) \quad (3)$$

After 32 rounds of iteration, (X_{32}, X_{33}, X_{34}, X_{35}) is generated, and then reverse transform is used to adjust the data order. Let the reverse transform function be R(.), then

$$(Y_0, Y_1, Y_2, Y_3) = R(X_{32}, X_{33}, X_{34}, X_{35}) = (X_{35}, X_{34}, X_{33}, X_{32}) \quad (4)$$

Where (Y_0, Y_1, Y_2, Y_3) is the final encryption result.

2.2. SM4 Key Expansion Algorithm

During the SM4 encryption/decryption iterative process, one round key generated by the key expansion algorithm is required for each iteration. Compared with the encryption/decryption algorithm, the key expansion algorithm has no reverse transform. It can be defined as follows: set user key $MK = (MK_0, MK_1, MK_2, MK_3)$, $MK_i \in Z_2^{32}, i = 0,1,2,3$, and $K_i \in Z_2^{32}, i = 0,1,\cdots,35$, then the round key is generated as

$$rk_i = K_{i+4} = K_i \oplus T'(K_{i+1} \oplus K_{i+2} \oplus K_{i+3} \oplus CK_i) \quad (5)$$

Where (K_0, K_1, K_2, K_3) = ($MK_0 \oplus FK_0$, $MK_1 \oplus FK_1$, $MK2 \oplus FK_2$, $MK_3 \oplus FK_3$), FK_0 = (A3B1BAC6), FK_1 = (56AA3350), FK_2 = (677D9197), and FK_3 = (B27022DC). Equation (5) is slightly different from Equation (1), that is, $T'(.) = L'(\tau(.))$ is different, which is defined as follows: set input $B \in Z_2^{32}$ and output $C \in Z_2^{32}$, then

$$C = L'(B) = B \oplus (B \ll 13) \oplus (B \ll 23) \quad (6)$$

The $CK_i = \big(ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3}\big) \in (Z_2^8)^4, i = 0,1,\cdots,31$ in Equation (5) is a set of constants, and its calculation method can be expressed as $ck_{i,j} = (4 \times i + j) \times 7 (mod\ 256)$.

2.3. Operation Modes of Block Cipher

There are five operation modes of the block cipher algorithm defined by reference [15], namely, electronic codebook mode (ECB), cipher block chaining mode (CBC), cipher feedback mode (CFB), output feedback mode (OFB), and counter mode (CTR). As shown in Figure 1, these five modes of operation can be summarized into three phases: the input processing phase, the encryption and decryption algorithm phase, and the output processing phase. Different operation modes can be realized by adjusting the input/output, as shown in Table 1. Taking the encryption algorithm in the CBC mode as an example, input processing needs to distinguish whether it is the first data block or not. If it is the first, the input data XORed with the IV (initialization vector) is used as an input of the encryption algorithm (input(0) ⊕ IV). If it is not the first, the input data is XORed with the output processing result of the previous data block and then sent to the encryption and decryption algorithm for processing (input(i) ⊕ output(i-1)).

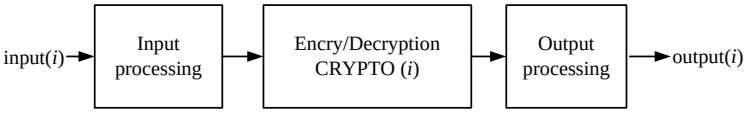

Figure 1. Operation flow of operation modes for block cipher

Table 1. Comparison of in/out processing method of various modes

Operation Modes	Input processing		Output processing	
	Encryption	Decryption	Encryption	Decryption
ECB	input(*i*)	input(*i*)	CRYPTO(*i*)	CRYPTO(*i*)
CBC	input(0) ⊕ IV	input(*i*)	CRYPTO(*i*)	IV ⊕ CRYPTO(0)
	input(*i*) ⊕ output(*i*-1)			input(*i*-1) ⊕ CRYPTO(*i*)
CFB	IV	IV	input(*i*) ⊕ CRYPTO(*i*)	input(*i*) ⊕ CRYPTO(*i*)
	output(*i*-1)	input(*i*-1)		
OFB	IV	IV	input(*i*) ⊕ CRYPTO(*i*)	input(*i*) ⊕ CRYPTO(*i*)
	CRYPTO(*i*-1)	CRYPTO(*i*-1)		
CTR	counter	counter	input(*i*) ⊕ CRYPTO(*i*)	input(*i*) ⊕ CRYPTO(*i*)

3. Methodology

In order to provide both flexibility and confidentiality for IoT devices, the following four basic ideas are considered:

1) Resource sharing: Comparing Equations (1) and (5), it can be found that only linear transforms are different. Thus, time-division multiplexed architecture can be considered to save resources.

2) Reduce the number of S-boxes: Each S-box stores 256×8 bits of data, which requires 1K-bit of storage space. If the number of S-box is reduced, the storage resources required for the S-box will be reduced.

3) Eliminate the storage space of the expanded keys: The key expansion algorithm generates 32 32-bit extended keys after 32 rounds of iteration, and if these keys are saved, 1K-bit storage space will be needed. However, if the key expansion algorithm and the encryption/decryption algorithm are alternately executed, the key expansion is performed first, and then the encryption/decryption algorithm is executed, thereby avoiding the preservation of the extended key.

4) Input processing and output processing can be configured: Comparing the five operation modes described in Table 1, it can be found that there are eight different input processing modes and four different output processing modes. If the operation mode is specified before the encryption and decryption algorithm is executed, and then the input processing and the output processing mode are configured according to the operation mode, multi-mode supporting can be realized.

4. Implementation

Based on the design ideas of Section 3, low-cost multi-mode hardware architecture is proposed in this section. In the following subsections, the design details of the proposed architecture will be introduced.

4.1. Architecture of Round Function

Figure 2 is the proposed architecture of the round function, and it is used to implement Equations (1)-(5). As shown in Figure 2, this architecture is composed of a nonlinear transform module, a linear transform module, and four sets of XOR gates. Switching between encryption/decryption or key expansion algorithms is mainly embodied in the linear transform module. As shown in Figure 3, shift and XOR operations required by Equations (3) and (6) can be realized by controlling MUXs by the mode signal.

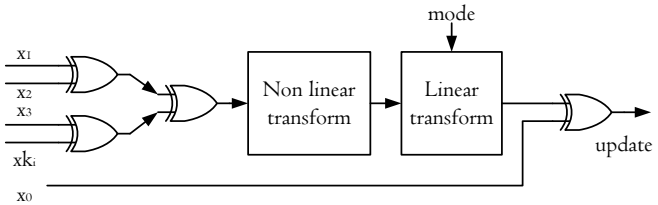

Figure 2. Architecture of round function

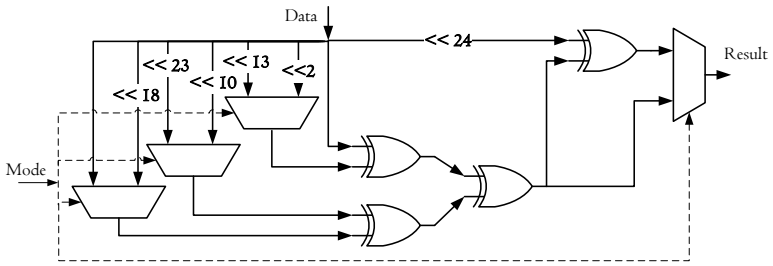

Figure 3. Architecture of multi-mode linear transform

In order to reduce the resource cost required for the s-box, a tradeoff between latency and area must be considered. As shown in Figure 4, a multi-cycle nonlinear transform architecture is proposed to realize Equation (2). This architecture consists of only one s-box and shift registers. The data path is eight bits, so it takes four clock cycles to complete the nonlinear transform for a 32-bit input.

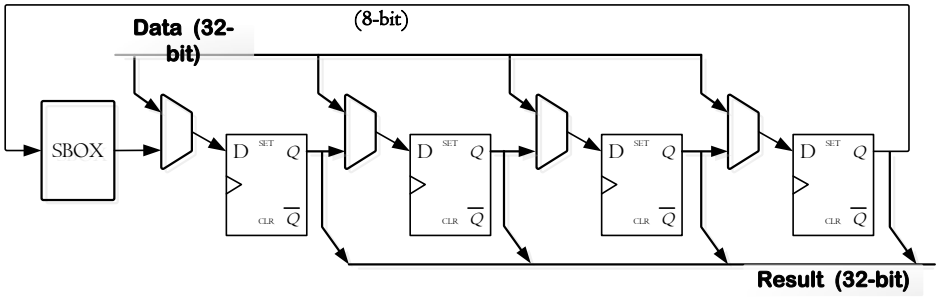

Figure 4. Architecture of multi-clock cycle non-linear transform

4.2. Architecture of SM4 Core

Figure 5 is the proposed architecture to realize the algorithms described in Sections 2.1 and 2.2. As shown in Figure 5, this architecture consists of a control state machine, round function, CK coefficient generation logic, 4×32-bit shift register for storing temporary encryption and decryption data (d0-d3), and a 4×32-bit shift register (k_0-k_3) for temporarily storing the extended key. With the architecture, encryption, decryption, and key expansion algorithms can be implemented, and there is no need to store expanded keys.

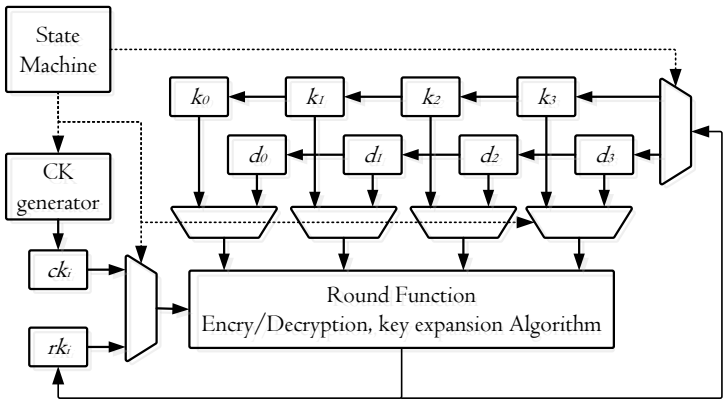

Figure 5. Architecture of SM4 core

In order to realize the resource sharing between encryption, decryption, and key expansion algorithm, a control state machine is proposed. As shown in Figure 6, the state machine contains two modes (encryption mode and decryption mode) and six states. The encryption mode alternately performs a key expansion algorithm and encryption algorithm. Since the round key required for the decryption algorithm is opposite to the round key order required by the encryption algorithm, and in order to avoid storing the expansion keys, it is necessary to perform decryption key preparation before alternately executing the key expansion algorithm and decryption algorithm. The so-called decryption key preparation means that the

32-round key expansion algorithm is iteratively executed to generate rk_{31}. State transition of the encryption and decryption mode is described in the following subsections.

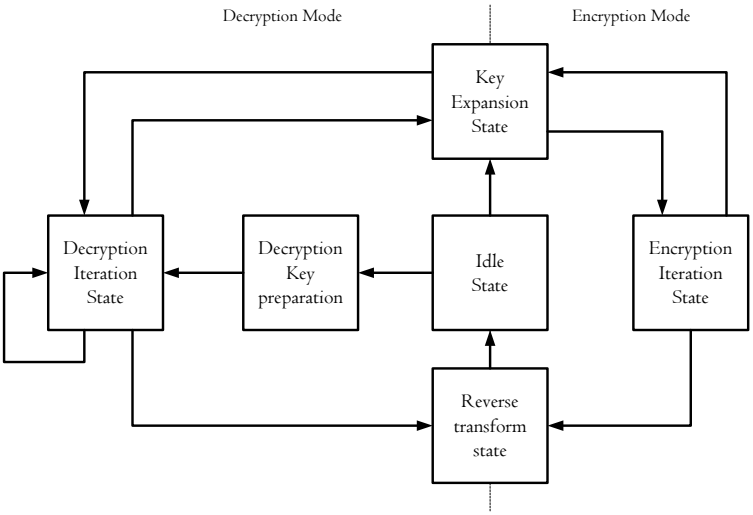

Figure 6. State transition of encryption/decryption algorithm

4.2.1. State Transition in Encryption Mode

1) In the idle state, when receiving a valid signal of a 128-bit data block, load the data block to d_0-d_3, and load the user key to k0-k3.

2) Enter the key expansion state, execute one round of key expansion algorithm, generate a 32-bit extended key, store it in the register, and push it into the shift register k_0-k_3.

3) Enter the encryption iteration state, execute one round of encryption algorithm, generate a 32-bit data, and push it into the shift register d_0-d_3.

4) The key expansion state and encryption state continue to alternate 31 times in turn, first the key expansion and then the encryption algorithm.

5) Enter the reverse transform state, and adjust the order of the shift registers d_0-d_3 according to Equation (4).

6) Return to idle mode and wait for the input of the next data block.

4.2.2. State Transition in Decryption Mode

1) In the idle state, when receiving a valid signal of a 128-bit data block, load the data block to d_0-d_3, and load the user key to k_0-k_3.

2) Enter the decryption key preparation state. After 32 round key expansion, four 32-bit expansion keys $(rk_{31}, rk_{30}, rk_{29}, rk_{28})$ are generated, and temporarily store them in the shift register k_0-k_3.

3) Enter the decryption iteration state, and sequentially perform four rounds of decryption algorithms with the four expanded keys generated in the above step.

4) Enter the key expansion state, execute one round of key expansion algorithm, and generate rk_{27}.

5) Return to the iterative state of the decryption and perform one round of the decryption algorithm.

6) The key expansion state and decryption iteration state continue to alternate 27 times in turn, first the key expansion and then the decryption algorithm.

7) Enter the reverse transform state, and adjust the order of the shift registers d0-d3 according to Equation (4).

8) Return to the idle mode and wait for the input of the next data block.

4.3. Overall Architecture

According to the comparison results of the different operation modes listed in Table 1, it can be found that there are eight different input modes and four different output modes. Since each data block operates in only one mode, in order to reduce resource overhead, it is not necessary to implement all input modes and output modes simultaneously. After careful study and comparison, it is found that through resource sharing, only four 128-bit data registers, one 128-bit user key register, and an 8-bit control register are needed to realize the encryption/decryption algorithm in five operation modes. Based on the above findings, the overall architecture shown in Figure 7 is proposed.

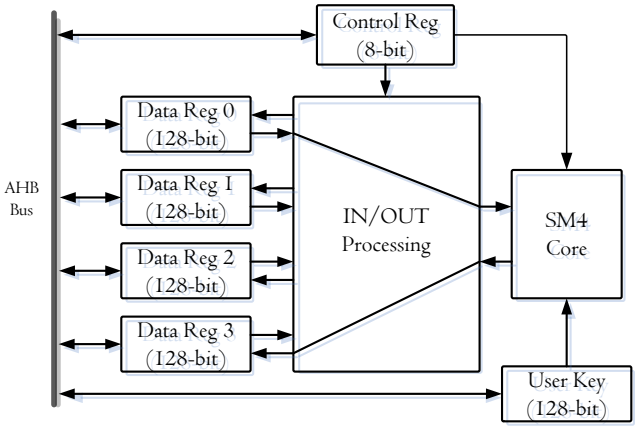

Figure 7. Overall architecture

As shown in Figure 7, for the purpose of module reusing, the proposed architecture is attached as a Slave to AHB bus. The input/output processing module in Figure 7 is configurable and used to generate input data and adjusts output data for the SM4 algorithm core in accordance with the operating mode. There are seven configurations for this module, and six of them are listed in Figure 8 (the configuration of ECB mode is shown in Figure 7).

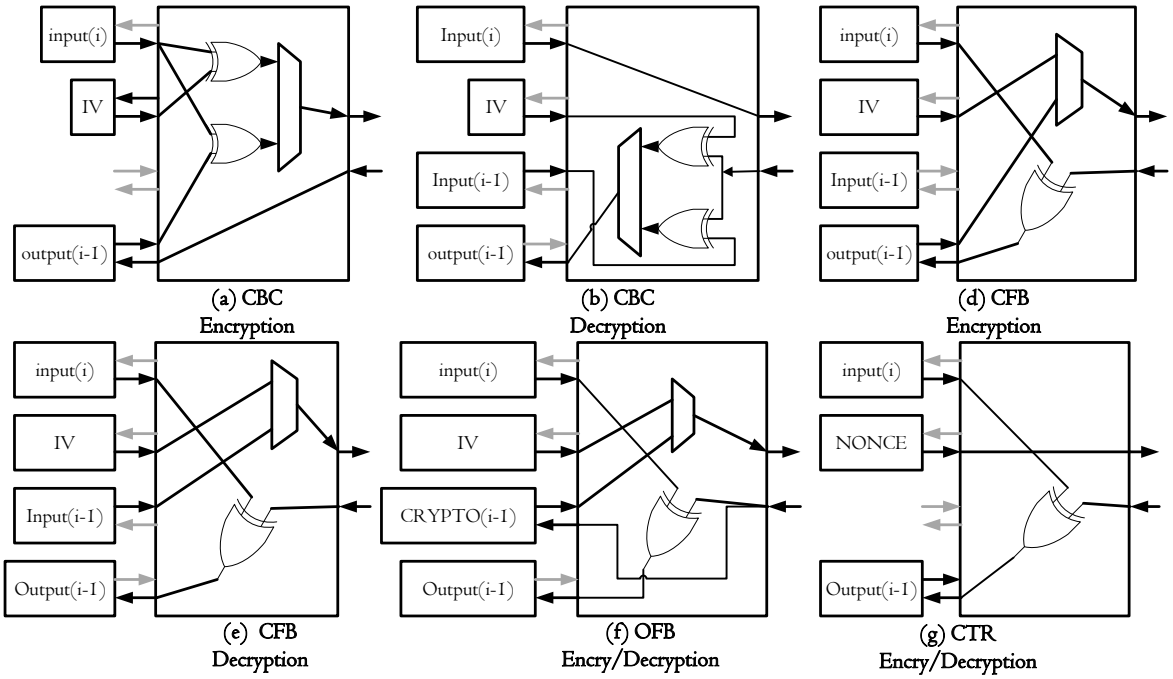

Figure 8. Different configurations of various operation modes

5. Experimental Results and Comparison

The proposed architecture is described with Verilog HDL, functional verified by Mentor Graphics ModelSim, and implemented on Xilinx ZYNQ-7 ZC702 FPGA (xc7z020clg484-1) and Altera Cyclone IV E (EP4CE115F29I8L) FPGA.

5.1. FPGA Implementation Results

Tables 2 and 3 give the Xilinx and Altera FPGA resource cost at 50MHz. As shown in Table 2, only 1,326 LUTs and 1,000 registers are required when implemented on Xilinx FPGA. As shown in Table 3, only 2,861 LEs are required when implemented on Altera FPGA.

Table 2. Resource cost of Xilinx FPGA

Module	LUT	Registers	Memory	Slice
Round Function	486	36	0	164
SM4 Core	1187	308	0	359
In/out Processing	139	692	0	95
Overall Architecture	1326	1000	0	454

Table 3. Resource cost of Altera FPGA

Module	LE	Register	Memory
Round Function	405	36	0
SM4 Core	1553	308	0
In/out Processing	139	692	0
Overall Architecture	2861	1000	0

5.2. Performance Analysis

According to the post-implementation simulation results, 384 clock cycles are needed to encrypt one 128-bit data block and 554 clock cycles for decryption. Therefore, using the proposed architecture at 50MHz, the throughput rate for the encryption algorithm can be achieved up to $128 \div 384 \times 50 = 16.67$Mbps, and $128 \div 554 \times 50 = 11.55$Mbps for decryption algorithm.

5.3. Performance Comparison

Table 4 shows the comparisons of different SM4 architectures. The performance is compared and discussed in terms of operation modes, throughput rate, technology, and resource overhead. As shown in Table 4, all of the references are limited to a single operation mode, but five different operation modes are supported in our proposed architecture. References [11-13] are aimed at improving the throughput rate of hardware architecture, not reducing the area cost of resource-constrained IoT devices. Thus, the throughput rate exceeds 500 Mbps, and the corresponding resource overhead is also large. References [7-10] are aimed at reducing the resource overhead and therefore achieved a resource overhead that is close to the proposed architecture. In reference [10], Xilinx FPGA was used to implement the SM4 with ECB mode. Reference [7-9] implemented SM4 with ECB mode on Altera FPGAs. Since references [7-10] only implemented the ECB mode, the SM4 core described in Section 4.2 can be considered the ECB mode. Thus, the resource overhead of the proposed SM4 core will be compared with them. According to Tables 2 and 3, the resource overhead is 100 slices less than reference [10], 151 LE less than reference [9], 332 LE less than reference [8], and 133 LE less than reference [7].

Table 4. Performance comparison with references

Ref.	Operation Mode	Throughput (Mbps)	Technology	Resource Cost
[9]	ECB	95	Altera FPGA	1704 LE
[10]	ECB	266.5	Xilinx FPGA	459 Slice
[11]	CBC	528.8	65nm	NA
[12]	CBC	2600	0.13μm	267361μm^2
[13]	CTR	14647	Xilinx FPGA	7423 ALMs
[7]	ECB	113.17	Altera FPGA	1686 LE
[8]	ECB	100	Altera FPGA	1885 LE
Ours	5 Modes	16.67 (Encryption)	Xilinx FPGA	454 Slice
Ours	5 Modes	11.55 (Decryption)	Altera FPGA	2861 LE

6. Conclusions

This paper proposed a low-cost multi-mode hardware architecture of SM4 to provide flexibility and confidentiality for resource-constrained IoT devices. By fully exploiting the key expansion, encryption, decryption algorithm, and resource sharing under different operation modes, the number of s-boxes is reduced, storage space for expanded keys is eliminated,

and configurable operation modes are achieved. Compared with existing designs, the resource overhead is significantly reduced. In the future, the method of area reduction for the SM4 algorithm will be further studied, and the SM4 IP core will be integrated into the system on chip.

References

1. Y. Q. Zhang, W. Zhou, and A. N. Peng, "Survey of Internet of Things Security," *Journal of Computer Research and Development*, Vol. 54, No. 10, pp. 2130-2143, 2017
2. K. Rose, S. Eldridge, and L. Chapin, "*The Internet of Things*: *An Overview Understanding the Issues and Challenges of a More Connected World*," The Internet Society (ISOC), 2015
3. S. Li, "Chapter 5 - Security Requirements in IoT Architecture," in *Securing the Internet of Things*, S. Li and L. D. Xu Eds. Boston: Syngress, pp. 97-108, 2017
4. S. Lv, B. Su, P. Wang, Y. Mao, and L. Huo, "Overview on SM4 Algorithm," *Journal of Information Security Research*, Vol. 2, No. 11, pp. 995-1007, 2016
5. W. Li, G. Bai, and X. Wu, "Hardware Implementation of SM4 based on Composite Filed S-box and It's Security Against Machine Learning Attack," in *Proceedings of 2018 IEEE International Conference on Electron Devices and Solid State Circuits* (*EDSSC*), pp. 1-2, 2018
6. Z. Guan, Y. Li, T. Shang, J. Liu, M. Sun, and Y. Li, "Implementation of SM4 on FPGA: Trade-off Analysis Between Area and Speed," in *Proceedings of 2018 IEEE International Conference on Intelligence and Safety for Robotics* (*ISR*), pp. 192-197, 2018
7. K. S. Zhu, Z. B. Dai, L. C. Zhang, W. Li, and W. M. Zhu, "Lightweight Implementation of SM4 for Internet of Things," *Application of Electronic Technique*, Vol. 42, No. 12, pp. 27-30, 2016
8. C. G. Wang, S. S. Qiao, and Y. Hei, "Low Complexity Implementation of Block Cipher SM4 Algorithm," *Computer Engineering*, Vol. 39, No. 7, pp. 177-180, 2013
9. C. G. Wang, S. S. Qiao, and Y. Hei, "Design of Low Complexity SM4 Block Cipher IP Core," *Science Technology and Engineering*, Vol. 13, No. 2, pp. 347-350, 2013
10. H. Liang, L. J. Wu, and X. M. Zhang, "Design and Implementation of SM4 Block Cipher based on Composite Field," *Microelectronics and Computer*, Vol. 32, No. 5, pp. 16-20, 2015
11. L. Y. Fan, M. Zhou, J. J. Luo, and H. L. Liu, "IC Design with Multiple Engines Running CBC Mode SM4 Algorithm," *Journal of Computer Research and Development*, Vol. 55, No. 6, pp. 1247-1253, 2018
12. T. S. Fu and S. G. Li, "A High-Throughput ASIC Implementation of SM4 Algorithm in CBC Mode," *Microelectronics and Computer*, Vol. 33, No. 10, pp. 13-18, 2016
13. Z. F. Wang and Z. J. Tang, "A High-Throughput ASIC Implementation of SM4 Algorithm in CTR Mode," *Chinese Journal of Electron Devices*, Vol. 42, No. 1, pp. 173-177, 2019
14. L. Zheng, C. Li, Z. Liu, L. Zhang, and C. Ma, "Implementation of High Throughput XTS-SM4 Module for Data Storage Devices," in *Proceedings of International Conference on Security and Privacy in Communication Systems*, pp. 271-290, 2018
15. M. Dworkin, "Recommendation for Block Cipher Modes of Operation. Methods and Techniques," *National Institute of Standards and Technology*, Vol. 5, No. 6, pp. 669-675, 2001