DPR Based AES/SM4 Encryption Highly Efficient Implementation

Qiangjia Bi, Ning Wu, Fang Zhou and Yasir

***Abstract*—This paper presents an efficient hardware implementation of Advance Encryption Standard (AES) and SM4 algorithms on Xilinx Virtex 7 FPGA by exploiting the feature of dynamic partial reconfiguration (DPR) to optimize S-Box on composite field arithmetic (CFA). We utilize delay aware common sub-expression elimination (DACSE) scheme to reduce overall area consumption by sharing the multiplicative inverse (MI) over GF(2^8) and eliminating the redundant circuitry. Our results reveal that hardware resources i.e. slices are minimized by 21.6%, and frequency is improved by 27.9%. In addition to that efficiency of our implementation is improved by 62.70 Mbps/slices which is higher than the previously proposed design.**

***Index Terms*—AES, SM4, Substitution Box(S-Box), DPR**

I. Introduction

THE Advanced Encryption Standard (AES) is widely used in various applications of information security as an encryption algorithm [1], [2]. SM4 is the first block cipher algorithm released by the Chinese National Cipher Management Committee Office and it is usually used in wireless security of local area network products [3]. Large number of products use these two algorithms to implement encryption of data. Most applications are using them independently; and without any optimization that leads to more on chip area consumption. Therefore, it is significant to optimize these algorithms used in the wireless sensor networks and other resource limited applications.

S-Box is a nonlinear function and the fundamental computing unit of AES and SM4, that occupies most of area and power consumption in circuits. For AES it occupies about 75% area of the round transformation [4]. Hence it is important to analyze the S-Box and optimize the logic.

The design and optimization of S-Box have been studied in detail. Shared memory scheme is used to design a reconfigurable S-Box for different cipher algorithms that provide more flexibility in [5], [6]. However, their design consumes more area and reduces the performance of the circuit because of the memory. It uses the additional circuit to implement the two encryption schemes but only one algorithm can be used in one instance of time in [7]. It costs additional area and increased power consumption.

Since both of the cipher algorithm AES and SM4 are using MI as a computing unit which is the most complex function for reducing the chip area of S-Box. Our design shares the multiplicative inverse (MI) over GF(2^8) based on dynamic partial reconfiguration (DPR) technology. In order to further improve implementation efficiency, composite field arithmetic (CFA) and delay aware common sub-expression elimination (DACSE) have been employed in our S-Box design.

The rest of the paper is organized as follows: Section II introduces the S-Box algorithm of AES and SM4. Section III describes the optimized implementation of S-Box circuit. Meanwhile, the detailed CFA, DACSE algorithm and DPR technology is discussed. Section IV shows the results and evaluation of our design along with comparison to other design schemes. Finally, Section V concludes this paper.

II. Review of S-Box Algorithm

The algebraic expression of S-Box of AES and SM4 is shown as (1) respectively [8], [9].

$$\begin{cases} Z(x) = I(x) \bullet M + V \\ S(x) = I(x \bullet A + C) \bullet A + C \end{cases} \tag{1}$$

Where $\boldsymbol{I}$ is the multiplicative inverse (MI) over GF(2^8) which can be reused by AES and SM4, $\boldsymbol{x}$ is the input of S-Box. $\boldsymbol{M}$ is the affine matrix and $\boldsymbol{V}$ is row vector of AES S-Box. $\boldsymbol{A}$ is the affine matrix and $\boldsymbol{C}$ is row vector of SM4 S-Box. $\boldsymbol{Z(x)}$ and $\boldsymbol{S(x)}$ are the output of S-Box of AES and SM4 respectively. M and V are shown in (2). A and C are shown in (3).

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad V = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \tag{2}$$

Manuscript received July 02, 2018; revised July 25, 2018. This work was supported by the National Science Foundation of China (No.61774086, No.61376025), the Fundamental Research Funds for the Central Universities (NS2017023) and the Natural Science Foundation of Jiangsu Province (BK20160806).

Q. B. is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China (e-mail: biqiangjia@nuaa.edu.cn).

N. W. is with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China (e-mail: wunee@nuaa.edu.cn).

F. Z and Yasir is with the College of Electronic and Information Engineering Nanjing University of Aeronautics and Astronautics, Nanjing, 211100, China.

$$A=\begin{bmatrix}1&1&1&0&0&1&0&1\\1&1&1&1&0&0&1&0\\0&1&1&1&1&0&0&1\\1&0&1&1&1&1&0&0\\0&1&0&1&1&1&1&0\\0&0&1&0&0&1&1&1\\1&0&0&1&0&1&1&1\\1&1&0&0&1&0&1&1\end{bmatrix},\quad C=\begin{bmatrix}1\\1\\0\\1\\0\\0\\1\\1\end{bmatrix} \tag{3}$$

III. Design and Optimization of MI

A. Using CFA to optimize MI over GF(2^8)

Since the calculation of the MI over GF(2^8) is very complex, we introduce CFA to reduce the hardware complexity by mapping the MI over GF(2^8) into composite field GF((2^4)^2). In order to achieve this transformation, we need to optimize the equation (1), that is expressed as:

$$\begin{cases}Z(\mathrm{x})=M\left(\delta^{-1}(\delta X)^{-1}\right)+V\\ \mathrm{S}(\mathrm{x})=A\left(T^{-1}\left(T(AX+C)\right)^{-1}\right)+C\end{cases} \tag{4}$$

Where $\boldsymbol{\delta}$, $\boldsymbol{T}$ is the mapping matrix and the $\boldsymbol{\delta^{-1}}$, $\boldsymbol{T^{-1}}$ is the inverse of mapping matrix. We can calculate them using MATLAB based on the irreducible polynomials.

The structure of S-Box computing process is shown as Fig.1. The structure on the upper half of Fig.1 is for AES S-Box while bottom half shows SM4 S-Box. The MI over GF((2^4)^2) in the middle is the MI which need to be optimized.

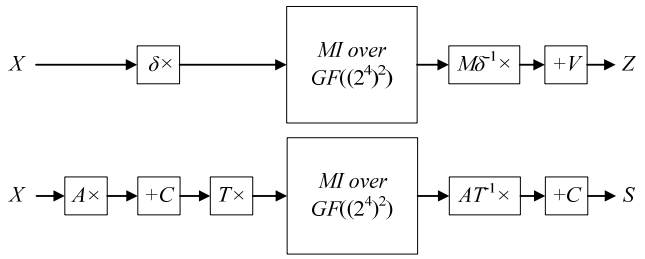

Fig. 1. S-Box computing process

In finite field, the irreducible polynomials of GF(2^8) of AES is (5). In the composite field GF((2^4)^2) using (6) as irreducible polynomials internally.

$$P_{28}(x)=x^8+x^4+x^3+x+1 \tag{5}$$

$$\begin{cases}GF\left(\left(2^4\right)^2\right):\ f_1(y)=y^2+y+\upsilon\\ GF\left(2^4\right):\ f_2(x)=x^4+x^3+x^2+x+1\end{cases} \tag{6}$$

Where $\boldsymbol{v}$={0010}$_2$. $\boldsymbol{v}$ is the coefficient of the GF(2^2) irreducible polynomial. Modification of this polynomial affects the design the circuit. For SM4, it has a little different irreducible polynomial in finite field GF(2^8). When it converts to composite field it still needs the irreducible polynomials (6).

Based on the first equation in (6), the MI over GF(2^8) is calculated in the following expression:

$$\begin{aligned}A^{-1}&=\left(A_h^{\,2}\upsilon+\left(A_h+A_l\right)A_l\right)^{-1}\left(A_h\gamma+\left(A_h+A_l\right)\right)\\&=B_l+B_h\gamma=B\end{aligned} \tag{7}$$

Where A can be expressed as A=A_h+γA_l, A_h, A_l ∈GF(2^4), B is the output of MI that can be expressed as B=B_h+γB_l, B_h, B_l ∈GF(2^4). The main architecture of MI over GF((2^4)^2)as shown in Fig.2.

The composite filed MI over GF((2^4)^2) needs three multiplication, two addition, one square multiply coefficient and one multiplicative inverse. All sub-calculation are 4 bit input and 4 bit output in finite field GF(2^4).

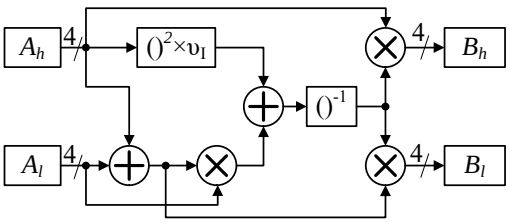

Fig. 2. MI over GF((2^4)^2)

Following section shows the calculation of multiplication, addition, square multiply coefficient and MI in GF(2^4).

a) Multiplication over GF(2^4)

Computing the multiplication, we assume that a and b are elements over GF(2^4). They can be described as:

$a=a_3\omega_3+a_2\omega_2+a_1\omega+a_0$, $b=b_3\omega_3+b_2\omega_2+b_1\omega+b_0$ (8)

where {a_3, a_2, a_1, a_0, b_3, b_2, b_1, b_0} ∈ GF(2). So the expression of multiplication can be shown in (9) and $c=a\times b \bmod(\mathrm{f}_2(\mathrm{x}))$.

$$c=\begin{cases}c_3=\left(a_3b_1+a_1b_3\right)+\left(a_3b_0+a_0b_3\right)+\left(a_2b_1+a_1b_2\right)+a_2b_2\\ c_2=\left(a_3b_1+a_1b_3\right)+\left(a_2b_0+a_0b_2\right)+a_2b_2+a_1b_1\\ c_1=\left(a_3b_1+a_1b_3\right)+\left(a_1b_0+a_0b_1\right)+a_3b_3+a_2b_2\\ c_0=\left(a_3b_2+a_2b_3\right)+\left(a_3b_1+a_1b_3\right)+a_2b_2+a_0b_0\end{cases} \tag{9}$$

As the set of equation (9) shows, the multiplication needs 25AND gates and 21 XOR gates.

b) Addition over GF(2^4)

In finite field, the addition is just XOR of the corresponding bit so the addition of GF(2^4) is relatively simple. It just needs 4 XOR gates, the expression is shown in (10).

$$d=a+b=\begin{cases}d_3=a_3b_3\\ d_2=a_2b_2\\ d_1=a_1b_1\\ d_0=a_0b_0\end{cases} \tag{10}$$

c) Constant Multiplied by Square over GF(2^4)

Based on the multiplication we calculated the square operation shown in (11).

$$h=a^2=\begin{cases}h_3=a_2\\ h_2=a_2+a_1\\ h_1=a_3+a_2\\ h_0=a_2+a_0\end{cases} \tag{11}$$

Because $\boldsymbol{v}$ is a constant {0010}$_2$. We calculate $\boldsymbol{v}$ with the square operation. And it cost nothing.

$$e=a^2*v=\begin{cases}e_3=a_1\\ e_2=a_3\\ e_1=a_0\\ e_0=a_2\end{cases} \tag{12}$$

d) Multiplicative Inverse over GF(2^4)

In finite field, it has $\alpha^{2^p-1}=1,\alpha\in GF\left(2^p\right)$. So in GF($2^4$), the calculation of multiplicative inverse is

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

$\alpha^{-1} = \alpha^{14}$. We transfer the calculation from multiplicative inverse to multiplication. The expression is shown in (13).

$$f = a^{-1} = \begin{cases} f_3 = a_3a_2a_0 + a_3a_1a_0 + a_3a_1 + a_2a_0 + a_2 + a_1 \\ f_3 = a_3a_2a_0 + a_2a_1a_0 + a_2a_0 + a_1a_0 + a_3 + a_1 \\ f_3 = a_3a_2a_1 + a_3a_1a_0 + a_2a_1a_0 + a_3a_0 + a_2a_1 \\ \qquad\qquad + a_2a_0 + a_1 \\ f_3 = a_3a_2a_1 + a_3a_2a_0 + a_3a_1 + a_2a_0 + a_1 + a_0 \end{cases} \quad (13)$$

It needs 27 AND gates and 21 XOR gates.

TABLE I
GATES CONSUMPTION AFTER CFA

Computing Unit	AND gate	XOR gate
Multiplication	25	21
Addition	0	4
Constant multiplied by square	0	0
MI over GF(2^4)	27	21
MI over GF(2^8)	102	92

MI over GF(2^8) includes 2 addition, 3 multiplication, 1 MI over GF(2^4) and 1 constant multiplied by square.

So after all the sub-calculation, we summarize the MI over GF(2^8) gate consumption in TABLE I.

B. Using DACSE to optimize the MI over GF(2^8)

Because those multiplication and MI over GF(2^4) are complex, we continue to optimize these two units based on DACSE considering the critical path delay and hardware consumption.

a) Multiplication over GF(2^4) optimized by DACSE

On (8), we extract the same computing elements as shown in (14).

$$\begin{cases} R_1 = a_3b_2 + a_2b_3 \\ R_2 = a_3b_1 + a_1b_3 \\ R_3 = a_3b_0 + a_0b_3 \\ R_4 = a_2b_1 + a_1b_2 \\ R_5 = a_2b_0 + a_0b_2 \\ R_6 = a_1b_0 + a_0b_1 \end{cases}, \begin{cases} r_3 = a_3b_3 \\ r_2 = a_2b_2 \\ r_1 = a_1b_1 \\ r_0 = a_0b_0 \end{cases} \quad (14)$$

So the product of a and b is shown:

$$c = a * b \bmod(\mathrm{f}_2(\mathrm{x})) = \begin{cases} c_3 = R_2 + R_3 + R_4 + r_2 \\ c_2 = R_2 + R_5 + r_2 + r_1 \\ c_1 = R_2 + R_6 + r_3 + r_2 \\ c_0 = R_1 + R_2 + r_2 + r_0 \end{cases} \quad (15)$$

Still we can find one computing unit expressed in (16).

$$S = R_2 + r_2 \quad (16)$$

So, after comparing and searching the final result is (17):

$$c = a * b \bmod(\mathrm{f}_2(\mathrm{x})) = \begin{cases} c_3 = S + R_3 + R_4 \\ c_2 = S + R_5 + r_1 \\ c_1 = S + R_6 + r_3 \\ c_0 = S + R_1 + r_0 \end{cases} \quad (17)$$

The overall hardware consumption of multiplication is 16 AND gates and 15 XOR gates and the critical path is $3T_{XOR}+1T_{AND}$.

b) Multiplicative inverse over GF(2^4) optimized by DACSE

Same way, the common part of multiplicative inverse is shown in (18) and the final expression is (19).

$$\begin{cases} R_1 = a_3a_2 \\ R_2 = a_3a_1 \\ R_3 = a_3a_0 \\ R_4 = a_2a_1 \\ R_5 = a_2b_0 \\ R_6 = a_1a_0 \end{cases}, \begin{cases} R_7 = R_1a_1 \\ R_8 = R_1a_0 \\ R_9 = a_3R_6 \\ R_{10} = a_2R_6 \end{cases}, \begin{cases} S_1 = a_1 + R_5 \\ S_2 = R_7 + S_1 \\ S_3 = R_8 + S_1 \end{cases} \quad (18)$$

$$f = a^{-1} = \begin{cases} f_3 = S_2 + a_2 + R_1 + R_8 \\ f_3 = S_2 + a_3 + R_5 + R_9 \\ f_3 = S_1 + R_2 + R_3 + R_8 + R_9 \\ f_3 = S_1 + a_0 + R_0 + R_7 \end{cases} \quad (19)$$

The overall hardware consumption of MI over GF(2^4) is 10 AND gates and 16 XOR gates and the critical path is $3T_{XOR}+2T_{AND}$.

After DACSE the gates utilization of GF(2^8) is shown in Table II.

TABLE II
GATES CONSUMPTION AFTER DACSE

Computing Unit	AND gate	XOR gate
Multiplication	25→16	21→15
Addition	0	4
Constant multiplied by square	0	0
MI over GF(2^4)	27→10	21→16
MI over GF(2^8)	102→58	92→69

The A→B means that the number of gates decrease from A to B.

C. Using DACSE to optimize the constant matrix over GF(2^8)

After calculating the MI over GF(2^8), we generate the mapping matrix and applied further optimizing by DACSE. The computing procedure of AES is like Fig.3.

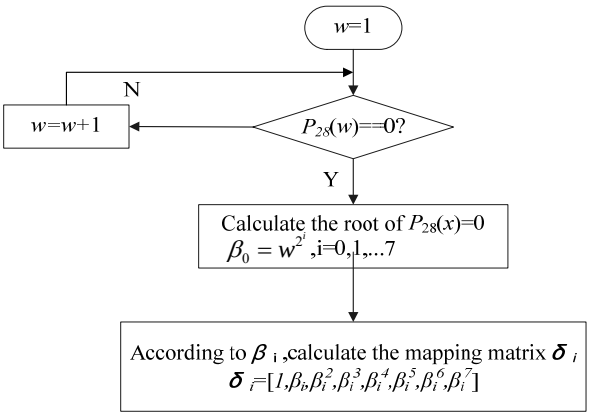

Fig. 3. AES calculation of mapping matrix

For example, based on the irreducible polynomial of AES over GF(2^8) shown in (5), we calculate the root β_i of $P_{28}(w)=0$. w is from 1 to 255, we use method of exhaustion to find β_i. Based on the equation (18), use β_i to generate the mapping matrix $\boldsymbol{\delta}$ and the $\boldsymbol{\delta^{-1}}$ is using $\boldsymbol{\delta^{-1}\delta = E}$ to generate. The mapping matrix calculation of SM4 is same like AES.

$$\begin{aligned} \beta_i &= \beta_0^{2^i}, (\mathrm{i} = 0 \sim 7) \\ \delta_i &= \left[1, \beta_i, \beta_i^{\,2}, \beta_i^{\,3}, \beta_i^{\,4}, \beta_i^{\,5}, \beta_i^{\,6}, \beta_i^{\,7}\right] \end{aligned} \quad (20)$$

And the related matrix of AES is shown in (21), for SM4 is (22).

$$\delta=\begin{bmatrix}1&1&0&1&0&0&1&0\\1&0&1&0&1&1&0&0\\1&1&0&1&1&1&1&0\\0&1&1&1&0&0&0&0\\0&0&0&0&0&1&0&0\\1&1&1&1&1&1&0&0\\1&1&1&0&0&0&0&0\\1&0&1&0&0&0&0&1\end{bmatrix},\delta^{-1}=\begin{bmatrix}1&0&1&1&0&1&0&0\\1&1&1&0&0&0&1&0\\0&1&0&1&0&1&0&0\\1&0&1&0&0&1&1&0\\1&0&1&0&1&0&0&0\\0&0&0&0&1&0&0&0\\0&1&1&1&0&0&0&0\\1&1&1&0&0&0&0&1\end{bmatrix} \quad (21)$$

$$T=\begin{bmatrix}0&1&1&1&1&1&0&0\\1&0&0&0&1&1&1&0\\1&0&1&0&1&1&0&0\\0&0&1&0&1&1&0&0\\1&1&1&0&0&0&1&0\\0&1&1&1&0&1&0&0\\0&1&1&0&0&1&1&0\\0&1&1&0&0&0&1&1\end{bmatrix},T^{-1}=\begin{bmatrix}0&0&1&1&0&0&0&0\\0&1&0&1&1&0&0&0\\1&0&1&0&1&1&1&0\\1&1&0&0&1&0&0&0\\1&0&0&0&0&1&0&0\\0&0&1&1&1&0&1&0\\1&1&0&0&1&1&1&0\\0&0&1&1&1&0&0&1\end{bmatrix} \quad (22)$$

Then we calculate the $\mathbf{M}\boldsymbol{\delta}^{-1}$ and $\mathbf{A}\boldsymbol{T}^{-1}$. This matrix multiplication has also been optimized by DACSE.

Finally, we have the gate consumption of those matrix multiplication as shown in Table III.

TABLE III
CONSTANT MATRIX CONSUMPTION AFTER DACSE

Computing Unit	AES	SM4
$\boldsymbol{\delta}\times$	11	/
$\boldsymbol{M\delta^{-1}}\times$	13	/
$\boldsymbol{A}\times$	/	21
$\boldsymbol{T}\times$	/	13
$\boldsymbol{AT^{-1}}\times$	/	13

The number in the table is the XOR gates number.

D. Using DPR Technology to optimize the MI over GF(2^8)

Dynamic Partial Reconfiguration is being used in many kind of security area [10]. Based on Fig.1, we use the optimized MI over GF(2^8) as the static part which always work by AES and SM4 whenever any of the algorithms is used. The other operations of AES S-Box including mapping matrix, affine matrix and raw vector are in dynamic 1 AES. The other operations of SM4 are dynamic 2 SM4 and the structure of our design is shown in Fig.3.

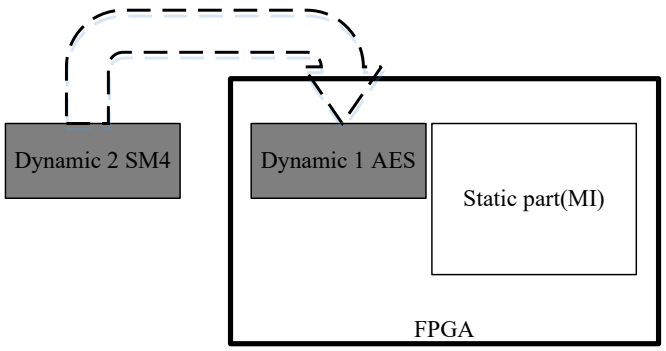

Fig. 3. S-Box computing process

In this design, we let the biggest unit be the static and small circuit to be the dynamic. In this way, we do not have to use the redundancy circuit which always works on chip like [7]. Further improvement circuit area, critical path and DPR computing process is shown in Fig.4.

After all calculation, the un-optimized and optimized design are shown in Table IV. And comparing with the normal design, our optimized design reduce 11XOR gate of mapping unit. Mapping inverse also reduce 7 XOR gate.

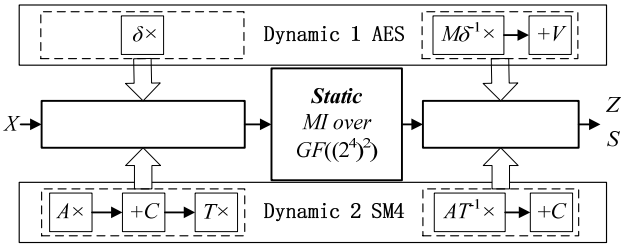

Fig. 4. DPR S-Box computing process

TABLE IV
THE GATE OPTIMIZATION COMPARISON

Computing Unit	Not Optimized		Optimized	
	XOR	AND	XOR	AND
Mapping	22	0	11	0
MI over GF(2^8)	102	92	58	69
Mapping inverse	20	0	13	0
Sum	144	92	82	69

IV. IMPLEMENT RESULTS AND ANALYSIS

In paper [11], [3], they optimized AES S-Box and SM4 S-Box on GF((((2^2)^2)^2)^2) and implemented on Spartan-3E FPGA. In order to compare the difference, we implement their design on Vritex-7 FPGA. As the table V shows, the 2 design is being optimized. But if directly add the slices number of two design. The slices should be 46.

TABLE V
THE COMPARISON OF DIFFERENT DESIGN

Performance Parameter	Ref.[11] AES -Box	Ref.[3] SM4 S-Box	Ours
LUT6	10	10	15
LUT5	16	26	7
LUT4	12	14	16
LUT3	6	5	6
LUT2	14	10	10
LUT1	0	10	2
Critical Path	9	12	9/12
Frequency/MHz	221.3	188.8	188.8

In paper [7], it design a redundancy S-Box circuit in ASIC. Because of the redundancy, it cost more area, power and increase the path delay. We have implemented that design in FPGA to compare that design with our design. After implementation, the comparison of our S-Box design and others design is shown in Table VI.

TABLE VI
THE COMPARISON OF DIFFERENT DESIGN

Performance Parameter	Ref.[7]	Ours
Static/LUT	120	39
Dynamic/LUT	0	12/21
Slices	37	29
Power/W	0.262	0.271
Frequency/MHz	163.93	188.8
Efficient/ (Mbps/slices)	35.44	52.1

As the Table VI shows, paper [7]'s design cost 120 LUT, but ours only cost 39 for static and at most 21 for dynamic. And the slices are reduced by 21.6%. And the frequency is improved 27.9%.

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. Conclusion

S-Box is the most important and complex unit of AES and SM4 encryption algorithm[5]. It is significant to do more researches on S-Box, especially on reducing the area, improving the speed and low power. CFA and DACSE schemes remarkably reduce the implementation area of the S-Box. Beside that, a new implementation of AES and SM4 S-Box based on Dynamic Partial Reconfiguration technology of Xilinx Virtex 7 is proposed in this paper. After comparing with the other designs, our design shows area reduction of 21.6% and frequency improvement is 27.9%.

References

[1] D. Canright, "A Very Compact S-Box for AES," *Preceddings of 7th International Workshop on Cyptograpic Hardware and Embedded Systems (CHES)*, pp. 441–455, 2005.

[2] D. Chen, G. Shou, Y. Hu, and Z. Guo, "Efficient Architecture and Implementations of AES GF (24)." *International Conference on Advanced Computer Theory & Engeineering*, pp. 295–298, 2010.

[3] XU Yan-hua, BAI Xue-fei, and GUO Li, "A new algorithm of S-box for hardware implementation of SMS4," *Journal of University of Science & Technology of China*, vol. 39, p. 1164-1170, 2009.

[4] Yong Zhang, Fang Zhou, and Yasir, "FPGA Based Highly Efficient AES Implementation," *WCECS*, vol. I, pp. 1–5, 2017.

[5] Y. Jinjiang, G. Wei, C. Peng, and Y. Jun, "An Area-Efficient Design of Reconfigurable S-box for Parallel Implementation of Block Ciphers," *IEICE Transactions on Fundamentals of Electronics Communications & Sciences*, vol. 10, pp. 1–10, 2016.

[6] W. Shan, X. Zhang, X. Fu, and P. Cao, "VLSI design of a reconfigurable S-box based on memory sharing method," *IEICE Electrn Express*, vol. 11, no. 1, pp. 1–6, 2014.

[7] L. Fu, X. Shen, L. Zhu, and J. Wang, "A new compact hardware architecture of S-Box for block ciphers AES and SM4," *IEICE Electronics Express*, vol. 7, no. 2, pp. 365–375, 2014.

[8] F. Liu *et al.*, "Analysis of the SMS4 Block Cipher.," *Information Security & Privacy, Australasian Conference, Acisp, Townsville, Australia, July*, 2007, pp. 158–170.

[9] W. E. I. B. dian, M. A. W. ping, W. X. mei, and X. an, "The algebraic expression for the AES Sbox," *Journal of Xidian University.*, 2003.

[10] Z. E. A. A. Ismaili and A. Moussa, "Self-Partial and Dynamic Reconfiguration Implementation for AES using FPGA," *International Journal of Computer Science Issues*, vol. 2, pp. 33–40, 2009.

[11] M. Wisdom and P. Lee, "An efficient implementation of a Fully Combinational Pipelined S-Box on FPGA," *2016 Conference of Basic Sciences and Engineering Studies (SGCAC)*, pp. 222–227, 2007.

ISBN: 978-988-14048-1-7
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)