Hardware Implementation of SM4 Encryption Algorithm with Hybrid Stochastic Computing

Yinjie Song, Hongge Li* and Yuhao Chen

College of Electronic Information Engineering, Beihang University
{yinjiesong, honggeli, zy1902608}@buaa.edu.cn

Abstract

Hybrid stochastic computing (HSC) combines binary number and stochastic number that uses the expectations of multibit streams to achieve stochastic computing (SC). In this paper, SM4 algorithm circuit is designed using HSC. An arithmetic expression about HSC and ADD operation in the Galois field GF(2^3) is derived and implemented with HSC-XOR (exclusive-or) instead of regular XOR operations in SM4. We designed SM4 hardware circuit based on HSC-XOR circuit to realize the encryption function of SM4 algorithm. The result of SM4 hardware with HSC indicates that SC breaks through the existing application scenarios and is available for cryptographic chip design. The experiments illustrate that the determinate computing of HSC is verified by SM4 encryption hardware.

Keywords: Hybrid stochastic computing, determinate computing, SM4, encryption hardware

1 Introduction

Stochastic computing (SC) has attracted wide attention in recent years because of its simple arithmetic circuit implementation, low power consumption, high energy and area efficiency, and high fault tolerance[1]. Being invented on purpose of intimating human brain's working process to change the way traditional digital computers work, SC is now widely applied in various application conditions such as image processing, machine learning, and communication encoding[2][3]. Different from the positional number system, SC numbers are encoded as bitstreams and represented by the

*Corresponding author: College of Electronic Information Engineering, Beihang University, Beijing, 100191, China, Email: honggeli@buaa.edu.cn probability of 1's in the stream. The arithmetic operations based on SC play a role in probability values on streams that differ from the way that classic binary computing works.

With a wider range of applications, traditional SC's drawbacks become obvious gradually. An evident disadvantage that traditional SC technique has is the imprecision of computation. The calculation error comes from the property that the results are estimates of probabilities carried by stochastic streams and the indelible correlations of input streams. This problem can only be alleviated by using proper linear feedback shift registers (LFSR) as pseudorandom number generators to generate SC streams previously in traditional SC methods. D. Jenson et al.[4] designed deterministic methods to implement accurate multiplication computation by extending the length of stochastic streams. Another urgent problem is the long calculation latency getting a relatively precise result need. H. Sim et al.[5] proposed a low-latency SC calculating method to reduce calculation time. Low-discrepancy sequences like the Sobol sequence are used to improve calculation precision and reduce stream length at the same time compared to normal LFSR-generated sequence[6]. However, these methods can only partially solve the above problems. In addition, the stochastic streams must be summed up to binary numbers after finite times of SC computing operations and re-generated to stochastic streams to do the rest calculations to prevent the loss of information, leading to more stochastic number generator (SNG) used and weakening the low hardware cost advantage of SC[7]. This problem still remains and seriously reduces the efficiency for large and complex circuit applications. In that case, Hybrid SC (HSC)[8][9][10] was designed on purpose of avoiding frequent conversion between binary numbers and stochastic bit streams, reducing the length of the sequence, and optimizing computation accuracy. It highly improves the energy and area efficiency and has been used on neural networks, polynomial fitting, image processing and Bernstein polynomial calculation.

Considering the advantages that HSC has over traditional SC, we believe that HSC has a potential application prospect in other areas like cryptography that traditional SC dose not. SM4 algorithm is a commercial block cipher algorithm designed by China[11]. It was approved as the national cryptography industry standard by the Office of State Commercial Cryptography Administration of China in 2012 and became an ISO/IEC standard in 2021. Being characterized by its high security, high computing efficiency, hardware friendliness, low cost, and high reliability, the SM4 algorithm is now used in Wireless Local Area Networks (WLAN) where high requirement for mobile internet security is needed. There are several designs of SM4 algorithm circuit proposed to optimize the hardware performance[12][13]. For cryptographic hardware implementations or chips that highly rely on their own fault tolerance, the reliability of the circuits needs full and serious consideration and thus many redundant hardware designs are proposed[14]. Displacing traditional computing method with HSC hopefully improves reliability against random faults or noises due to the reliability of HSC itself.

In this paper, an HSC-based SM4 algorithm circuit design called HSC-based SM4 is proposed and implemented successfully on Field Programmable Gate Array (FPGA). A novel HSC exclusive-or (HSC XOR) circuit is designed based on an arithmetic expression equivalent to the addition operation, or ADD operation in GF(2^3). Several XOR gates in SM4 computing architecture are replaced with HSC XOR circuits so that signals go through the encryption circuit as bit streams. The function of regular SM4 circuit is realized by the proposed HSC-based SM4 design. The successful implementation of the non-binary computing and reliable HSC-based SM4 circuit indicates that HSC is suitable for SM4 encryption circuit design and the determinate computing function of HSC is verified.

The remainder of this paper is organized as follows. In section 2, we introduce basic SC principles and computational elements, as well as the SM4 algorithm. In section 3, our proposed HSC XOR circuit and HSC-based SM4 circuit design based on the HSC XOR are described. We also discuss the performance of HSC-based SM4 circuit. Section 4 concludes this paper.

2 Background and Related Works

2.1 Stochastic Computing Theory

In stochastic computing, the probability of 1's presenting in a stochastic bit stream represents the value of an SC number. An SC bit stream with length N and $\sum_{i=0}^{N-1} X_i$ 1's encodes the value $\hat{p} = \hat{p}(X = 1) = \frac{\sum_{i=0}^{N-1} X_i}{N}$ in the unipolar form. A binary number A is encoded as a stochastic bit stream S_A through a comparator by comparing with random numbers generated from a random number generator (RNG). The RNG is mainly implemented by using LFSR in hardware designs. The probability of 1's in S_A is also considered as the expectation value of S_A. Thus, the calculation of numbers is converted to the computation of expectations of stream, that is, the operation on streams by SC calculation units is actually the operation on expectations.

$$P(S_{AB}) = E(S_{AB} = 1) = E(S_A = 1 \cap S_B = 1) = E(S_A = 1) \cdot E(S_B = 1) = P(S_A) \cdot P(S_B) \quad (1)$$

Where $E(\cdot)$ refers to the expectation of the stochastic bitstream. Thus, 2 LFSRs are needed to generate stochastic numbers S_A and S_B independently. The AND gate multiplier has the advantage of simple circuit design and low power consumption. Although the accuracy gets improved as streams get longer, it is mathematically inaccurate because the irrelevance of two input streams cannot get insured in the actual design, and input streams P(S_A) and P(S_B) satisfy the Bernoulli distribution so the result is also a random variable only the expectation of which is the correct value. In that case, it is not allowed for precise computing situations like encryption and decryption circuit design.

The HSC approach uses both binary number and SC bitstream coding to reach higher energy efficiency and accuracy. Compared to the classic SC system using a stochastic single bit stream to encode a value in the range [0,1], hybrid SC extends the concept of the stochastic bitstream to a stochastic multi-bit stream so that any value belongs to [0, 2^{n-1}] can get represented by an n-bit hybrid stochastic stream. Every single bitstream of n-bit stream can be considered as an n-bit binary number, so the value is encoded as the expectation of the binary number value of the hybrid logic stream. The hybrid SC is proved to be highly fault-tolerant, and operations based on it as multipliers, adders, and dividers work more precisely than any former designs.

2.2 SM4 Algorithm

In SM4, the key and plaintext with a length of 128 bits are considered as 4 words as each word has 32 bits. After 32 rounds of the key expansion algorithm, 32 round keys are obtained and sent to the encryption algorithm so that plaintexts get encrypted 32 rounds to attain the final ciphertext. The SM4 decryption process works the same way the encryption process does except for the reverse order of round keys. For simplicity, only the SM4 encryption algorithm is discussed in this paper. As shown in Fig. 1, SM4 consists of two similar parts, the key expansion round function F', and the encryption round function F. The encryption function F is determined by (2).

$$F(X_i, X_{i+1}, X_{i+2}, X_{i+3}, rk_i) = X_{\mathrm{i}} \oplus T(X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \oplus rk_i) \quad (2)$$

Where the round keys rk_i and intermediate round encryption values X_i are 32-bit words. The function T(·) is a combination of a linear transformation L(·) used to combine inputs by XOR after cyclic shifts of different bits, and a nonlinear transformation τ(·) consisting of 4 identical S-box, that is, T(·)=L(τ(·)). The S-box is a look-up table with 8-bit input and 8-bit output. Given a 32-bit input B, the function L(·) is defined by (3).

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

The key expansion round function F’ shares the same structure with F except for the linear transformation L’(·), that is, T’(·)=L’(τ(·)), as determined by (4) and (5).

$$F'(rk_i, rk_{i+1}, rk_{i+2}, rk_{i+3}, CK_i) = rk_\mathrm{i} \oplus T'(rk_{i+1} \oplus rk_{i+2} \oplus rk_{i+3} \oplus CK_i) \quad (4)$$

$$L'(B) = B \oplus (B <<< 13) \oplus (B <<< 23) \quad (5)$$

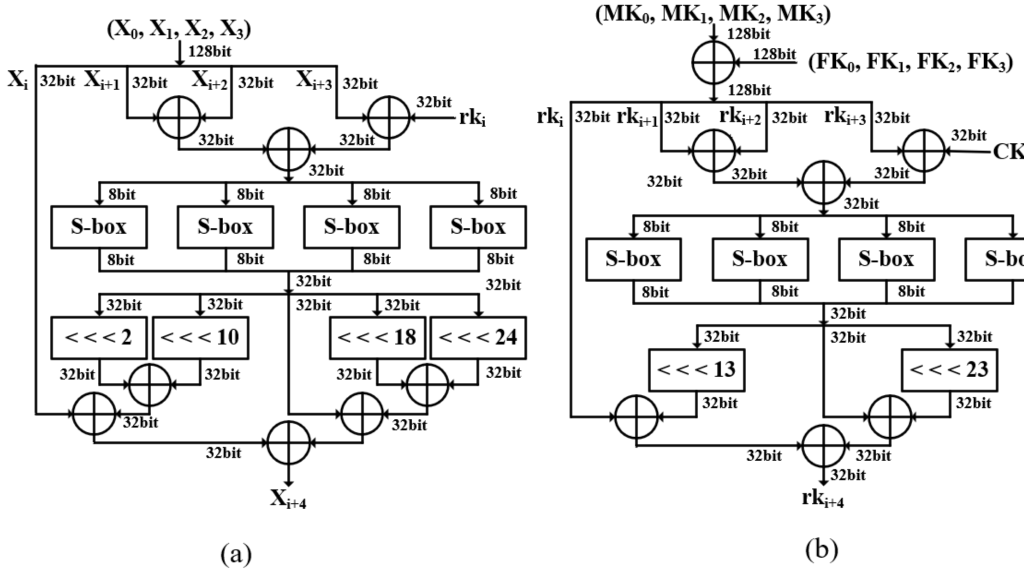


Figure 1: SM4 algorithm structure. (a) The structure of encryption round function F. (b) The structure of key expansion round function F’.

3 HSC-based SM4 Design

3.1 GF Exclusive-or Circuit with HSC

Besides the S-box lookup table and shift-left, XOR is the main calculation operation in the SM4 block cipher algorithm. For 32-bit XOR operator shown in Figure 1, considering $\mathrm{A} = (a_{31}a_{30} \dots a_1a_0)$, $\mathrm{B} = (b_{31}b_{30} \dots b_1b_0)$, a_i, $b_i \in \{0,1\}$, $i = 1,2,\dots,31$. The output is expressed as $C = A \oplus B = (c_{31}c_{30} \dots c_1c_0)$, where $c_i = a_i \oplus b_i$. It can also be considered as the addition in extension Galois Field GF(2^{32}). In that case, an arithmetic expression of A and B that equals the addition operation in GF(2^3) is derived and the deterministic HSC circuit based on the expression is designed.

In our design, 3 parallel XOR gates, or the addition in GF(2^3), is a basic processing element. For 3-bit binary numbers $X = (x_2x_1x_0)_2$ and $Y = (y_2y_1y_0)_2$ after XOR operation to get the binary number result $Z = (z_2z_1z_0)_2$, $z_i = x_i \oplus y_i$, $x_i, y_i, z_i \in \{0,1\}$, i=1,2,3, their values are expressed as $X = x_2 2^2 + x_1 2^1 + x_0 2^0$, $Y = y_2 2^2 + y_1 2^1 + y_0 2^0$, and $Z = z_2 2^2 + z_1 2^1 + z_0 2^0$. For every single bit, there exists the equation shown in (6),

$$z_i = x_i \oplus y_i = (x_i + y_i)\mathrm{mod}2 = |x_i - y_i| = (x_i - y_i)^2 = x_i^2 + y_i^2 - 2x_iy_i = x_i + y_i - 2x_iy_i \quad (6)$$

Thus, the value of Z can be expressed by x_i and y_i,

$$Z = 4z_2 + 2z_1 + z_0 = 4(x_2 + y_2 - 2x_2y_2) + 2(x_1 + y_1 - 2x_1y_1) + (x_0 + y_0 - 2x_0y_0)$$

$$= X + Y - 8x_2y_2 - 4x_1y_1 - 2x_0y_0$$

As Z is a 3-bit binary number, $Z \in \{0,1,2,\dots,6,7\}$, that is $Z = Z\,\mathrm{mod}\,8$. Thus, a simpler expression of the number Z is given:

$$\begin{aligned} \mathrm{Z} &= (\mathrm{X} + \mathrm{Y} - 8\mathrm{x}_2\mathrm{y}_2 - 4\mathrm{x}_1\mathrm{y}_1 - 2\mathrm{x}_0\mathrm{y}_0)mod8 \\ &= [(\mathrm{X} + \mathrm{Y} - 4\mathrm{x}_1\mathrm{y}_1 - 2\mathrm{x}_0\mathrm{y}_0)\mathrm{mod8} + (-8\mathrm{x}_2\mathrm{y}_2)\mathrm{mod8}]\mathrm{mod8} \\ &= (\mathrm{X} + \mathrm{Y} - 4\mathrm{x}_1\mathrm{y}_1 - 2\mathrm{x}_0\mathrm{y}_0)\mathrm{mod8} \end{aligned}$$

Considering the product XY:

$$\begin{aligned} (XY)\mathrm{mod8} &= [(4x_2 + 2x_1 + x_0)(4y_2 + 2y_1 + y_0)]\mathrm{mod8} \\ &= [16x_2y_2 + 8(x_2y_1 + x_1y_2) + 4(x_2y_0 + x_1y_1 + x_0y_2) + 2(x_1y_0 + x_0y_1) \\ &\quad + x_0y_0]\mathrm{mod8} \\ &= [4x_2y_0 + 2x_1y_0 + x_0y_0 + 4x_0y_2 + 2x_0y_1 + x_0y_0 + 4x_1y_1 - x_0y_0]\mathrm{mod8} \\ &= (Xy_0 + Yx_0 + 4x_1y_1 - x_0y_0)\mathrm{mod8} \end{aligned}$$

The term $4x_1y_1$ could be eliminated and the final arithmetic expression for Z in terms of X and Y is derived as shown in (7),

$$\begin{aligned} Z &= (X + Y - 4x_1y_1 - 2x_0y_0 - XY + XY)\mathrm{mod8} \\ &= [\mathrm{X} + \mathrm{Y} - 4\mathrm{x}_1\mathrm{y}_1 - 2\mathrm{x}_0\mathrm{y}_0 - \mathrm{X}\mathrm{y}_0 - Y\mathrm{x}_0 - 4\mathrm{x}_1\mathrm{y}_1 + \mathrm{x}_0\mathrm{y}_0 + XY]\mathrm{mod8} \\ &= \big((1 - \mathrm{y}_0)\mathrm{X} + (1 - \mathrm{x}_0)\mathrm{Y} - \mathrm{x}_0\mathrm{y}_0 + XY - 8\mathrm{x}_1\mathrm{y}_1\big)\mathrm{mod8} \\ &= [XY + (1 - \mathrm{x}_0)Y + (1 - \mathrm{y}_0)X - \mathrm{x}_0\mathrm{y}_0]\mathrm{mod8} \\ &= \mathrm{XY} + \frac{1}{2}(1 + (-1)^X)Y + \frac{1}{2}(1 + (-1)^Y)\mathrm{X} - \frac{1}{4}(1 - (-1)^X)(1 - (-1)^Y) \end{aligned} \tag{7}$$

However, this algebraic expression is not hardware friendly because it contains power terms that lead to complex SC calculation circuit design. While terms $\frac{1}{2}(1 + (-1)^X)$, $\frac{1}{2}(1 + (-1)^Y)$, $\frac{1}{2}(1 - (-1)^X)$, $\frac{1}{2}(1 - (-1)^Y)$ work as parity factors only, they are replaced with x_0 and y_0 to get a simpler form of arithmetic expression of Z shown in (8).

$$\mathrm{Z} = \mathrm{X} \oplus \mathrm{Y} = [XY + (1 - x_0)Y + (1 - y_0)X - x_0y_0]mod8 \tag{8}$$

It relies on multiplication, addition and subtraction so that it is realizable to be mapped to HSC as is shown in (9).

$$\begin{aligned} \mathrm{P}(s_Z) &= \mathrm{E}[s_Z] \\ &= X\mathrm{E}[s_Y] + (1 - x_0)\mathrm{E}[s_Y] + (1 - y_0)\mathrm{E}[s_X] - \mathrm{E}\big[s_{x_0y_0}\big] \\ &= X\mathrm{P}[s_Y] + (1 - x_0)\mathrm{P}[s_Y] + (1 - y_0)\mathrm{P}[s_X] - \mathrm{P}\big[s_{x_0y_0}\big] \end{aligned} \tag{9}$$

Notice that the mapping from (8) to an HSC expression is not unique, like the term XY can also be mapped to the equivalent $Y\mathrm{P}[s_X]$ which is implemented with almost the same circuit. The 3-bit HSC XOR circuit is designed based on (9) as is shown in Figure 2(a). LFSRs are used as random number generators which are compared to the input binary numbers X and Y to obtain stochastic streams s_X and s_Y.As proposed by H. Li et al.[8], a multiplexer (MUX) consisting of 3 paralleled AND gates implements the term $X\mathrm{P}[s_Y]$ as the multiplication of X and Y by encoding binary number Y to stochastic stream s_{XY} with weight X. The same design method applies on terms $(1 - x_0)\mathrm{P}[s_Y]$ and $(1 - y_0)\mathrm{P}[s_X]$ obtaining streams $s_{(1-x_0)Y}$ and $s_{(1-y_0)X}$ separately by using two 1-bit MUXs which equals to an AND gate. As x_0 and y_0 are 1-bit binary numbers, two NOT gates are used to calculate $1-x_0$ and $1-y_0$. The stochastic stream $s_{x_0y_0}$ is generated directly from x_0y_0 calculated by an AND gate from x_0 and y_0. Although streams $s_{(1-x_0)Y}$, $s_{(1-y_0)X}$ and $s_{x_0y_0}$ are formally 1-bit traditional stochastic streams, they are taken as hybrid stochastic streams and summed up with the 3-bit hybrid stream s_{XY} to get the result hybrid stream s_Z by two adders and one subtractor. The adders and subtractor are common 3-bit binary adders and subtractor rounding off the carry and return terms to implement modular 8 operation. A modular 8 counter is used as a stochastic bit stream to binary number converter in order to get the final binary number result Z As hybrid SC is used, signals flow forward as streams in the circuit as is shown in Figure 2(a), and formula (12) can be calculated precisely with one complete calculation clock cycles being 2^3. In that case, the hybrid SC achieves determinate computing which could not be implemented by traditional SC and can reach a higher calculation speed than any former SC method. The schematic and sequential diagram in Figure 2 also give a calculation example of $X=(100)_2=(4)_{10}$ and $Y=(110)_2=(6)_{10}$ go through the HSC XOR circuit to get $Z=(010)_2=(2)_{10}$.

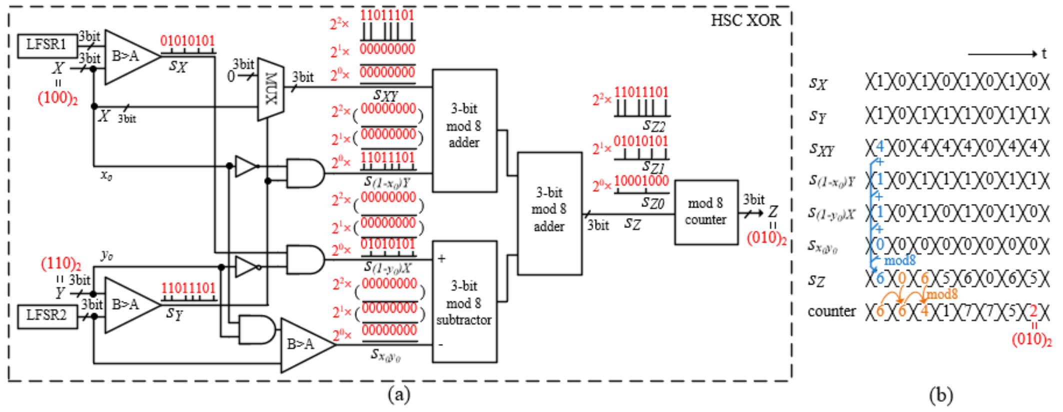


Figure 2: The 3-bit HSC XOR circuit design. (a) The circuit design and an example of the HSC XOR circuit working exhibiting inputs $X=(100)_2$ and $Y=(110)_2$ getting the result $Z=(010)_2$. (b) The sequential diagram of HSC signals of Figure (a).

Each 3-bit HSC XOR circuit costs 16 LUTs and 12 FFs on FPGA. As shown in Figure 3, a 32-bit HSC XOR circuit is implemented by combining eleven 3-bit HSC XOR circuits parallelly with the extra one bit set to 0 so it can be used on stochastic computing SM4 hardware design.

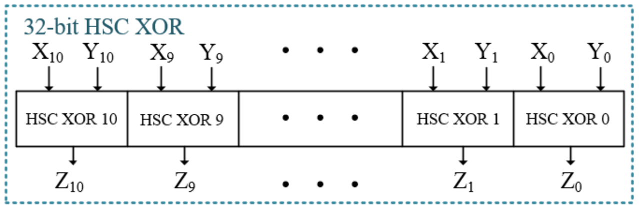

32-bit HSC XOR
X_{10} Y_{10} X_9 Y_9 X_1 Y_1 X_0 Y_0
HSC XOR 10
HSC XOR 9
HSC XOR 1
HSC XOR 0
Z_{10} Z_9 Z_1 Z_0

Figure 3: The 32-bit HSC XOR spliced by eleven HSC XORs in parallel.

3.2 HSC-based SM4 Circuit Implementations

In HSC-based SM4 circuit, the regular XOR operations before the S-box are replaced by the proposed HSC XOR considering aligning the latency of round function circuit and key expansion round function circuit, as is shown in Figure 4. In our design, signals before reaching the S-box are calculated in SC stream form. The new HSC key expansion round function F' circuit obtains the next round key rk_{i+1} based on the current round key rk_i and current round parameter CK_i, which costs 714 LUTs and 511 FFs on FPGA. The example from the SM4 Block Cipher Algorithm Standard is selected to give an exhibition of one round HSC key expansion process. In Figure 5, given round key $(rk_0, rk_1, rk_2\ rk_3)$ = (F1 21 86 F9 41 66 2B 61 5A 6A B1 9A 7B A9 20 77)$_{16}$, we have the next round key rk_4 = (36 73 60 F4)$_{16}$. The presence of 1 in signal *keytime* represents the complement of one round process. One running process of an HSC XOR is exhibited with signals *s_x*, *s_y*, *s_xy*, *s_nx0_y*, *s_ny0_x*, and *s_x0y0* showing SC numbers passing through as streams. Similarly, the HSC encryption round function F circuit obtains the next round ciphertext X_{i+1} from the current round ciphertext X_i and the current round key rk_i. With nearly the same circuit architecture, the HSC encryption circuit costs 714 LUTs and 511 FFs on FPGA. As shown in Figure 6, given current round ciphertext $(X_4, X_5, X_6\ X_7)$ = (27 FA D3 45 A1 8B 4C B2 11 C1 E2 2A CC 13 E2 EE)$_{16}$ and round key rk_4 = (36 73 60 F4)$_{16}$, the next round ciphertext X_8 = (F8 7C 5D B5)$_{16}$ is obtained.

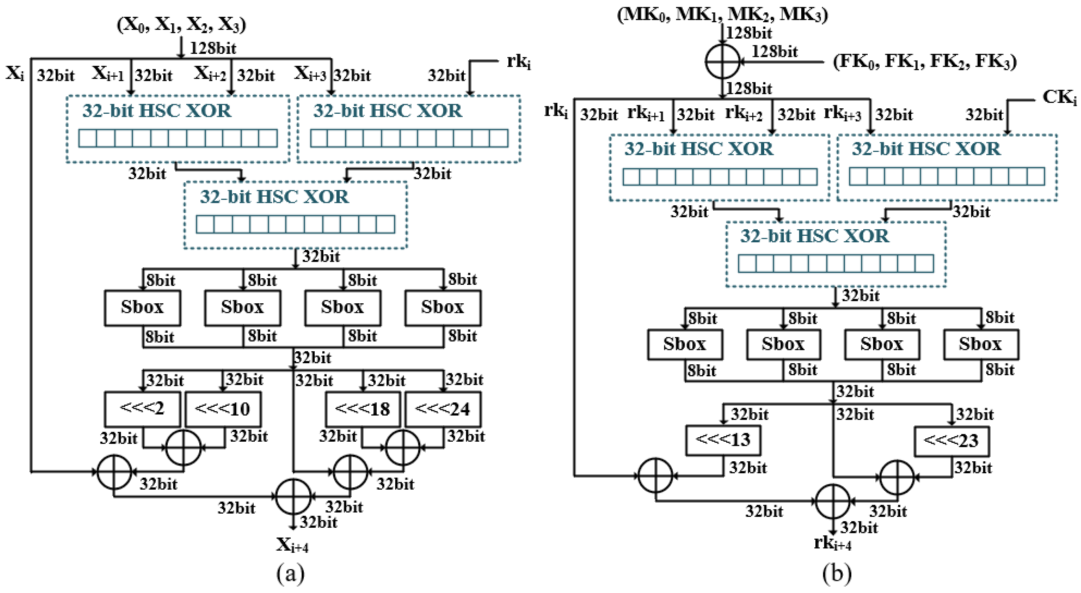
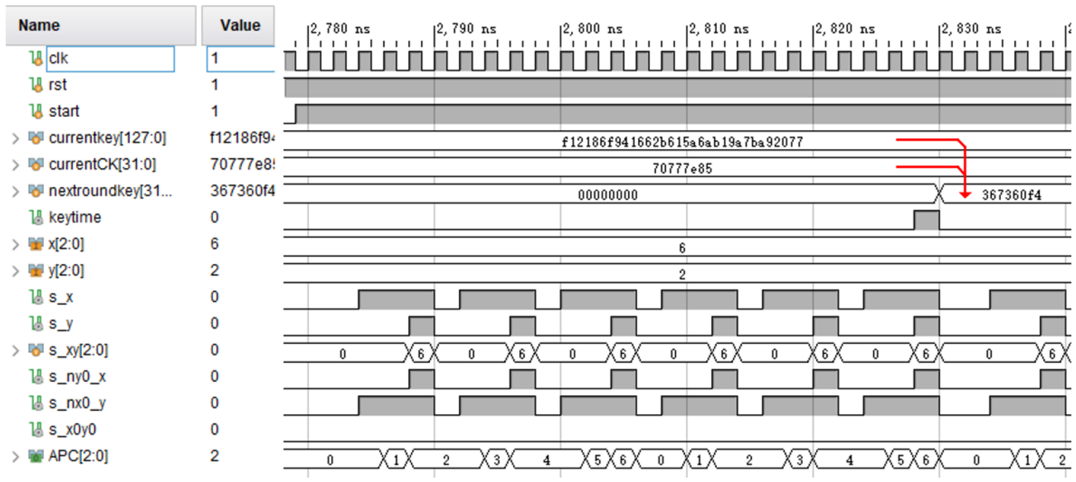


Figure 4: HSC-based SM4 design. (a) The structure of HSC-based SM4 encryption round function F. (b) The structure of HSC-based SM4 key expansion round function F'.



Name	Value
clk	1
rst	1
start	1
currentkey[127:0]	f12186f9
currentCK[31:0]	70777e8
nextroundkey[31...	367360f4
keytime	0
x[2:0]	6
y[2:0]	2
s_x	0
s_y	0
s_xy[2:0]	0
s_ny0_x	0
s_nx0_y	0
s_x0y0	0
APC[2:0]	2

Figure 5: Calculating process of HSC key expansion circuit.

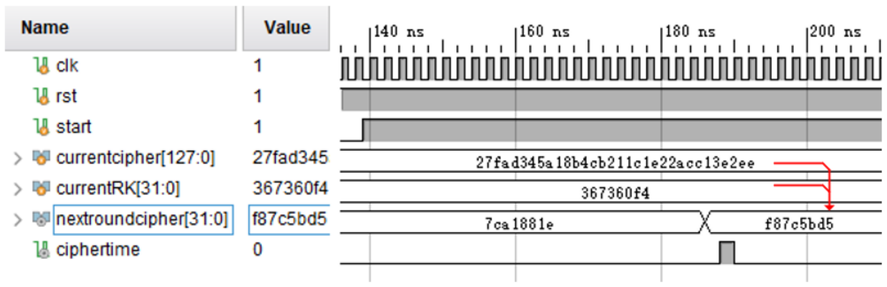


Figure 6: Calculating process of HSC encryption circuit.

From the architecture level, we use the scalar design in which there's one key expansion round function module and one encryption round function module in the calculation circuit. As shown in Figure 7, a Finite-State Machine (FSM) is included as the control module to ensure the plaintext and key run 32 times correctly through the circuit as it extends the one-round circuit to 32 rounds. Every round function circuit to implement F or F' includes three 32-bit HSC XOR circuits working in 2 independent steps, so the circuit can be time division multiplexed to encrypt two plaintexts with two keys separately and simultaneously. A complete encryption process example is given in Fig. 8 with the plaintexts and keys set to128'h 01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10 and 128'h FE DC BA 98 76 54 32 10 01 23 45 67 89 AB CD EF. The plaintexts and keys set method is based on the SM4 Block Cipher Algorithm Standard. The alternate encryption process and the results are shown as the verification of the function of the proposed HSC-based SM4 design.

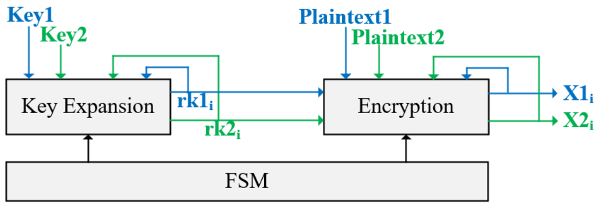


Figure 7: Scalar structure of HSC-based SM4.

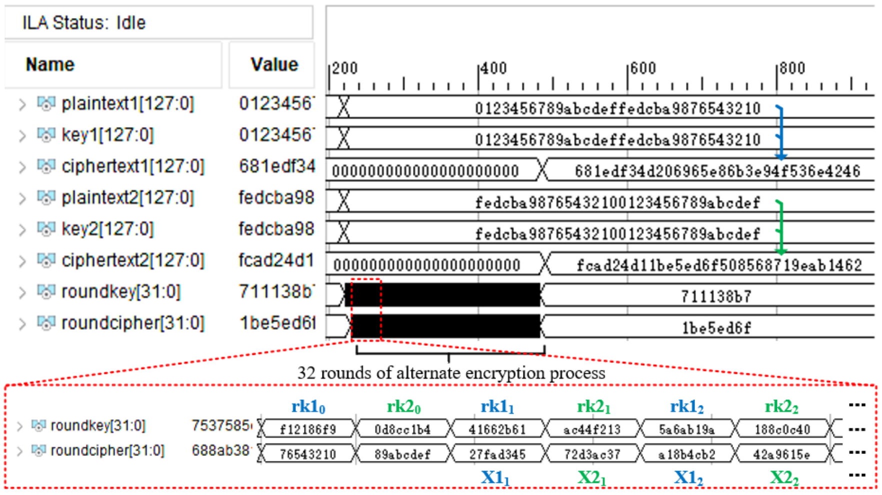

ILA Status: Idle
Name
Value
200
400
600
800
plaintext1[127:0] 0123456
key1[127:0] 0123456
ciphertext1[127:0] 681edf34
plaintext2[127:0] fedcba98
key2[127:0] fedcba98
ciphertext2[127:0] fcad24d1
roundkey[31:0] 711138b
roundcipher[31:0] 1be5ed6f
0123456789abcdeffedcba9876543210
0123456789abcdeffedcba9876543210
0000000000000000000000
681edf34d206965e86b3e94f536e4246
fedcba98765432100123456789abcdef
fedcba98765432100123456789abcdef
0000000000000000000000
fcad24d11be5ed6f508568719eab1462
711138b7
1be5ed6f
32 rounds of alternate encryption process
rk1₀ rk2₀ rk1₁ rk2₁ rk1₂ rk2₂ ...
roundkey[31:0] 7537585
f12186f9 0d8cc1b4 41662b61 ac44f213 5a6ab19a 188c0c40 ...
roundcipher[31:0] 688ab38
76543210 89abcdef 27fad345 72d3ac37 a18b4cb2 42a9615e ...
X1₁ X2₁ X1₂ X2₂ ...

Figure 8: The encryption process of HSC-based SM4 observed by the Integrated Logic Analyzer (ILA) using an example from the SM4 Block Cipher Algorithm Standard and a similar example to verify the function of the circuit.

The HSC-based SM4 design is implemented on Xilinx FPGA Vertex-7 VC709 and Kintex UltraScale+ XCKU5P KCU116 devices. The synthesis results of the designed circuit and the comparison with other designs with the same scalar structure are shown in Table 1. The whole HSC-based SM4 encryption circuit costs 1547 LUTs and 1328 FFs, with 538 clock cycles for the HSC-based SM4 to finish the encryption process. The maximum clock frequencies are 300 MHz and 560MHz separately, with a power consumption of 168mW and 236mW at the highest frequency without the cost of Mixed-Mode Clock Manager (MMCM) and Block RAM (BRAM), reaching a throughput of 142.7 Mbps and 266.4Mbps separately. The success of implementing HSC on the SM4 algorithm demonstrates that the deterministic HSC is available for SM4 encryption circuit design and similar block cipher circuit implementations for further applications.

Items		Results	
Methods		VC709	KCU116
Area	LUTs	1547	1547
	FFs	1328	1328
Maximum Frequency (MHz)		300	560
Clock Cycle		538	538
Power (W) (at Freq.$_{max}$)		0.168	0.236
Throughput (bps) ($\times 10^6$)		142.7	266.4
Energy Efficiency (bps/W) ($\times 10^9$)		0.849	1.129
Area Efficiency * ($\times 10^6$)		0.079	0.149

*Area Efficiency = Throughput$\times$LUT^{-1}

Table 1: Hardware Results of HSC-based SM4.

4 Conclusion

In this paper, a novel SM4 block cipher circuit based on hybrid stochastic computing is proposed. The equivalence of the addition operation in GF(2^3) and an arithmetic operation on the same strings if considered binary numbers is derived. The HSC XOR circuit is designed based on the arithmetic expression by using SC elements. Several XOR operations in the SM4 algorithm are implemented with HSC XOR circuit in HSC-based SM4 circuit design. The new HSC-based SM4 with new calculation method and fault tolerance fulfills the function of the original SM4 algorithm, verifies the determinate computation of HSC and demonstrate that HSC is available for the SM4 circuit design and other similar block cipher circuit designs.

References

[1] A. Alaghi and J. Hayes, (2013). Survey of Stochastic Computing. *ACM Transactions on Embedded Computing Systems*.

[2] B. R. Gaines (1969). Stochastic computing systems. *Adv. Inf. Syst. Sci., Boston, MA, USA: Springer*, pp. 37–172.

[3] Y. Liu; L. Liu; F. Lombardi and J., Han, (2019, Sep.). An energy-efficient and noise-tolerant recurrent neural network using stochastic computing. *IEEE Trans. Very Large Scale Integr. (VLSI)*, pp. 2213–2221.

[4] D. Jenson and M. Riedel, (2016). A deterministic approach to stochastic computation. *Proc. 35th Int. Conf. Comput.-Aided Design (ICCAD)*, (pp. 1–8).

[5] H. Sim and J., Lee, (2017). A new stochastic computing multiplier with application to deep convolutional neural networks. *Proc. 54th Annu. Design Autom. Conf. (DAC)*, (p. 29).
[6] S. Liu and J. Han, (2018, July). Toward Energy-Efficient Stochastic Circuits Using Parallel Sobol Sequences. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 1326-1339.
[7] S. A. Salehi, (2020, April). Low-Cost Stochastic Number Generators for Stochastic Computing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 992-1001.
[8] H. Li and Y. Chen, (2022), Hybrid Logic Computing of Binary and Stochastic. *IEEE Embedded Systems Letters*, pp. 171-174.
[9] Y. Chen and H. Li, (2022, May). Stochastic Computing Using Amplitude and Frequency Encoding. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, pp. 656-660.
[10] H. Li and Y. Chen, (2023). Hybrid Stochastic Number and its Neural Network Computation. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst., Early Access*.
[11] Office of State Commercial Cryptography Administration of China. (n.d.). Block Cipher for WLAN products-SMS4.
[12] S. Abed, R. Jaffal, B. Mohd and M. Alshayeji, (2021, Jan.). Performance evaluation of the SM4 cipher based on field-programmable gate array implementation. *IET Circuits, Devices & Systems*, pp. 121-135.
[13] Z. Guan, Y. Li, T. Shang, J. Liu, M. Sun and Y. Li, (2018). Implementation of SM4 on FPGA: Trade-Off Analysis between Area and Speed. *2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR), Shenyang, China*, pp. 192-197.
[14] S. Sheikhpour, A. Mahani and N. Bagheri, (2021). Reliable advanced encryption standard hardware implementation: 32- bit and 64-bit data-paths. *Microprocessors and Microsystems*.