Implementation of SMS4 Block Cipher on FPGA

Yier Jin
Zhejiang University
Institute of VLSI Design
Hangzhou, China
jinye@vlsi.zju.edu.cn

Haibin Shen
Zhejiang University
Institute of VLSI Design
Hangzhou, China
shb@vlsi.zju.edu.cn

Rongquan You
Zhejiang University
Institute of VLSI Design
Hangzhou, China
yourq@vlsi.zju.edu.cn

Abstract

This paper describes two encryption designs of Chinese wireless local area network block cipher standard – SMS4 algorithm. Then these two designs are implemented on Xilinx Virtex-4 FPGA devices. The first (pipelined) design is optimized in order to reduce time delay and the encryption core has a throughput of 24Gb/s. The second (folded) design is optimized in order to minimize area coverage and the encryption core only needs 380 CLB slices with throughput of 740Mb/s. There are no known hardware implementations of an encryption SMS4 designs.

1. Introduction

In February 2006, the Office of State Commercial Cryptography Administrator(OSCCA) in China released symmetric-key encryption standard of wireless local area network(WLAN)—SMS4 Block Cipher.

In this paper, two architectures and their FPGA implementations are proposed, the folded architecture and the pipelined architecture. Both designs include encryption process and key expansion schedule of SMS4 algorithm. The folded architecture is optimized in order to achieve small cover area. The pipelined architecture is optimized to acquire small processing time. These designs are implemented using Xilinx ISE 7.1i software on Virtex-4 FPGA devices.

Pipelined design can improve the encryption speed significantly as well as the throughput with outputs in every clock cycle. But this design need considerable memory and in virtex-4 family LX series there are up to 336 Block RAMs (BRAMs) which can fulfill demand of large memory request. On the other hand, folded architecture does not require as much memory as that of pipelined architecture, but speed is still a key characteristic of performance. So both designs are implemented on the Virtex-4 XC4VLX100-10C-ES device which can provide high operation speed and as much as 240 BRAMs.

Because this block cipher standard of WLAN is just released, the authors are not aware of any other SMS4 hardware designs proposed to date.

The rest of paper is organized as follows: Section 2 introduces the encryption/decryption processes of SMS4 algorithm. Two designs of SMS4 encryption with different architectures and their implementations on FPGA are described in Section 3. Section 4 presents the performances of these two designs. Finally, the conclusions are drawn in Section 5.

2. SMS4 Algorithm

SMS4 algorithm is a symmetric-key cipher, in which both the sender and the receiver use a single key for encryption and decryption. The data block length and the key length are both fixed to 128 bits. Furthermore, SMS4 algorithm is an iterative algorithm and we call each iteration a round. The total round number, N_r, is 32, i.e., the plain text transformed to cipher text needs to go through 32 iterations. The 128 bits data block is divided into 4 words each with 32-bit length. Each word is mapped to a state, the input data block is mapped to the first four states and all the internal iterations operated on former states to generate new states. There are total 36 states which are denoted by $X_i, 0 \leq i < 36$. Figure 1 shows the block diagram of the SMS4 encryption and the equivalent decryption structures.

In the encryption process of SMS4 algorithm, each round consists four transformations: the MixExStates, the AddRoundKey, the SubBytes and the MixShftStates. After 32 rounds' iterations, an extra Reorder stage is added. The MixExStates transformation performs two additions over GF(2) (note that all addition in this paper operate over GF(2)) on former three states to get a temporary variable TMP_1 as follow:

$$TMP_1 = X_{i+1} \oplus X_{i+2} \oplus X_{i+3} \quad (1)$$

1-4244-0463-0/06/$20.00

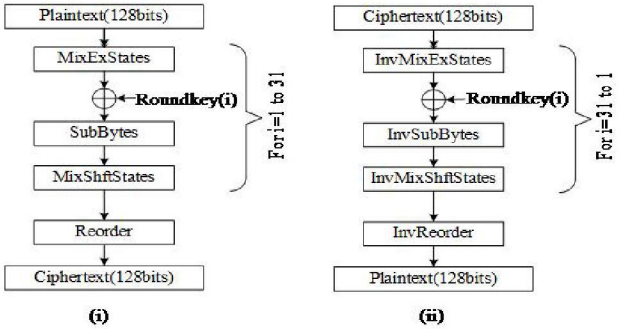

Figure 1. The SMS4 algorithm. (i) Encryption structure. (ii) Equivalent decryption structure

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	d6	90	e9	fe	cc	e1	3d	b7	16	b6	14	c2	28	fb	2c	05
1	2b	67	9a	76	2a	be	04	c3	aa	44	13	26	49	86	06	99
2	9c	42	50	f4	91	ef	98	7a	33	54	0b	43	ed	cf	ac	62
3	e4	b3	1c	a9	c9	08	e8	95	80	df	94	fa	75	8f	3f	a6
4	47	07	a7	fc	f3	73	17	ba	83	59	3c	19	e6	85	4f	a8
5	68	6b	81	b2	71	64	da	8b	f8	eb	0f	4b	70	56	9d	35
6	1e	24	0e	5e	63	58	d1	a2	25	22	7c	3b	01	21	78	87
7	d4	00	46	57	9f	d3	27	52	4c	36	02	e7	a0	c4	c8	9e
8	ea	bf	8a	d2	40	c7	38	b5	a3	f7	f2	ce	f9	61	15	a1
9	e0	ae	5d	a4	9b	34	1a	55	ad	93	32	30	f5	8c	b1	e3
a	1d	f6	e2	2e	82	66	ca	60	c0	29	23	ab	0d	53	4e	6f
b	d5	db	37	45	de	fd	8e	2f	03	ff	6a	72	6d	6c	5b	51
c	8d	1b	af	92	bb	dd	bc	7f	11	d9	5c	41	1f	10	5a	d8
d	0a	c1	31	88	a5	cd	7b	bd	2d	74	d0	12	b8	e5	b4	b0
e	89	69	97	4a	0c	96	77	7e	65	b9	f1	09	c5	6e	c6	84
f	18	f0	7d	ec	3a	dc	4d	20	79	ee	5f	3e	d7	cb	39	48

Figure 2. The S-box used in SubBytes transformation

in which TMP_1 is a 32-bit length word. Then the result is added with the ith roundkey. SubBytes transformation is a nonlinear byte substitution that operates independently on each byte of state X_i through a substitution table(S-box). The S-box can be listed as a 16×16 array which is showed in figure 2. The outputs of S-box are combined in one word—TMP_2—with the same sequence of inputs and MixShftStates transformation is to shift TMP_2 several fixed parameters, then adds all results together. Here we define $\lll i$ as shifting 32-bit length word to left in a loop. MixShftStates transformation can be expressed in (2).

$$X_{i+4} = TMP_2 \oplus (TMP_2 \lll 2) \oplus (TMP_2 \lll 10) \oplus (TMP_2 \lll 18) \oplus (TMP_2 \lll 24) \oplus X_i \quad (2)$$

After 32 iterations operated on 128-bit plaintext, we will get all the 36 states of $X_i, 0 \leq i < 36$. Finally, the Reorder transformation just combines the last four states, X_{32} to X_{35}, in a reverse sequence to generate cipher text.

$$ciphertext = (X_{35}, X_{34}, X_{33}, X_{32}) \quad (3)$$

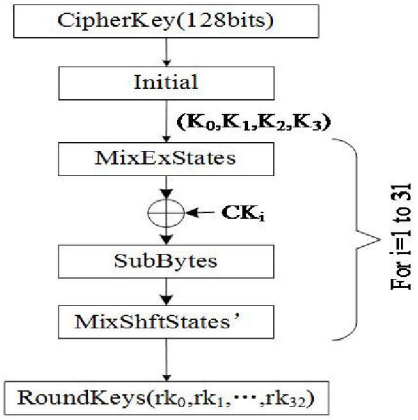

Figure 3. The SMS4 algorithm key expansion structure

The transformations of decryption process perform the inverse of the corresponding transformations in encryption process. However, they can share the same structure only by reverse the order in using roundkeys showed in figure 1. Furthermore, the S-box should be rebuilt because it is reverse to that in encryption.

In SMS4 algorithm, the key expansion schedule generates a total of thirty-two words named roundkeys $(rk_0, rk_1, \ldots, rk_{31})$. The process of Key expansion can be divided into 32 rounds which have the same structure as those in encryption process except for MixShftStates transformation. Figure 3 shows the total structure of key expansion schedule.

The 128-bit cipher key denoted as MK, which can be divided into 4 words named MK_0, MK_1, MK_2, MK_3, is operated through Initial transformation to form the generators of roundkeys. The Initial transformation is showed as

$$(K_0, K_1, K_2, K_3) = (MK_0 \oplus FK_0, MK_1 \oplus FK_1, MK_2 \oplus FK_2, MK_3 \oplus FK_3) \quad (4)$$

where $FK_i, 0 \leq i < 4$ present system parameters and are $(a3b1bac6)$, $(56aa3350)$, $(677d9197)$, $(b27022dc)$ in hexadecimal presentation, respectively. CK are fixed parameters which contains 32 words denoted as $CK_i, 0 \leq i < 32$. Each CK_i can be divided into four byte as $CK_i = (ck_{i,0}, ck_{i,1}, ck_{i,2}, ck_{i,3}) \in GF(2^8)^4$ and all bytes are computed as follow.

$$ck_{i,j} = (4i + j) \times 7 (mod\ 256) \quad (5)$$

Assume the output of SubBytes transformation is TMP, the MixShftStates' is presented as

$$rk_i = K_{i+4} = TMP \oplus (TMP \lll 13) \oplus (TMP \lll 23) \oplus K_i \quad (6)$$

In the key expansion process, each round produces a roundkey and after 32 rounds, all roundkeys are generated for encryption and/or decryption.

3. Implementation of SMS4 Algorithm

Because SMS4 algorithm is a block cipher, all four operation modes including the Electronic Codebook (ECB) mode, the Cipher Block Chaining (CBC) mode , the Cipher Feedback (CFB) mode, and the Output Feedback (OFB) mode, which are initially designed for the Federal Data Encryption Standard (DES) can be used [2]. In this paper, SMS4 algorithm implementations are based on ECB mode for its operation can be pipelined easily although this mode is less secure than other modes.

An additional novel characteristic of both designs is that the encryption process and key expansion are synchronous, which means roundkeys are generated at the same time of enciphering process. This merit confirms that the key can be changed arbitrarily and operations in which a batch of blocks may be encrypted for each of a number of differently keyed concurrent channels without loss in throughput.

The SMS4 designs described in this paper are coded using Verilog HDL. The pipelined structure is fully pipelined: both encryption design and key expansion schedule having thirty-two pipeline stages. While in folded structure, only one round design is presented and the outputs are available after several clock cycles of inputs.

3.1. Design of Pipelined Core

In the design of pipelined encryption structure, memory requirement is the main consideration because official document[1] only provides content of SMS4 S-box without any details about how to generate this S-box (although this may be released soon), so look-up table (LUT) is an efficient method to do SubBytes transformation. Since the SubBytes is operated on each byte individually, each round showed in Figure 1 needs four 8-bit$\times$8-bit LUTs. In FPGA, there are two types of RAM: distributed RAMs and RAM blocks (BRAMs). Distributed RAM suits for small data reservation while BRAMs are used for large data preservation. The LUT should be mapped on BRAM for its large data content and totally 128 LUTs are required in the pipelined encryption core. And the time cost in LUT is a large part of critical path, so these RAMs in FPGA should run as fast as possible. Based on two aspects — delay and capacity, we utilize Xilinx Virtex-4 FPGA family for implementation as they contain devices with up to 336 BRAMs running at frequency up to 500MHz.

In Virtex-4 family, each block RAM stores 18K bits of data with two symmetrical and totally independent ports.

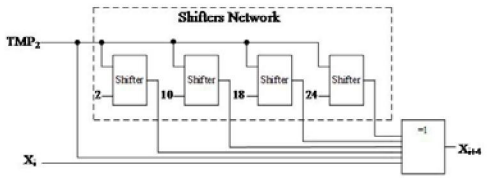

Figure 4. The construction of Shifters Network and structure of MixShftStates transformation

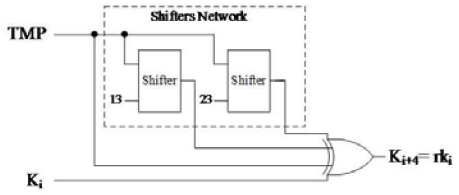

Figure 5. The structure of MixShftStates' transformation of key expansion

One BRAM can be configured into two single port 256$\times$8-bit RAMs, thus, two BRAMs are used in each round. From [3], if the write enable signal of BRAM is driven to low(1'b0), data stored in the BRAM is not affected on the write clock and one BRAM can be utilized as two LUTs.

The MixExStates transformation can be achieved only through two-word length XOR gates. The MixShftStates transformation is implemented by XORing the outputs of shifters network. These shifters are left-looped shifters of two input ports: one is data input and the other indicates shift times. The shifter network is constructed according to Equation (2) and illustrated in figure 4.

The Reorder transformation is hardwired easily with no logic involved of minimal time delays. In the key expansion schedule, MixExStates and SubBytes transformations are of the same structure with those in encryption design. The MixShftStates' transformation is showed in figure 5.

The Initial transformation implements word-length XORs on inputs with system parameters—FK. Furthermore, extra thirty-two 32-bit length registers should be added as the output of each key expansion round to store all the roundkeys. The key expansion structure is placed in parallel with encryption structure of one round earlier, so round key can be used immediately after it is generated for encryption. This structure is very efficient in the field where blocks are encrypted with variable keys.

From the above, in pipelined SMS4 algorithm design, totally 128 two-port BRAMs are used, 64 BRAMs are used in encryption structure and another 64 BRAMs are used in key

expansion schedule. Based on the memory consideration, we choose Virtex-4 XC4VLX100-10C-ES device which has 240 BRAMs and the BRAM usage rate is about 50%.

3.2. Design of Folded Core

The encryption part of folded core includes one round which is designed in Section 3.1 with some peripheral circuits, and a controller. And the main part of this controller is a counter, which will give control signals according to counter number. A sequential register file is used to store internal variables as well as the final results with the write enable signals from the controller. The key expansion schedule is of the similar structure. Both encryption structure and key expansion schedule operate synchronously.

For the sake of area, only eight LUTs are used in folded design, thus totally four BRAMs are utilized.

4. Performance Analysis

The SMS4 designs are implemented on Virtex-4 FPGA devices using Xilinx ISE v7.1 for downloading, ModelSim SE v6.0a for simulation and Synplify Pro v8.1 for synthesizing. In the pipelined design, enciphered data blocks are outputted each clock cycle after an initial delay of several clock cycles. The critical path of pipelined design contains LUT delay and combinational logic gates delay. The pipelined core which is implemented on the Xilinx XC4VLX100-10C-ES FPGA device, utilizes 9500 CLB slices(19%) and 386 IOB pins(50%). And 128 BRAMs are used out of total 240 BRAMs(53%). The frequency of this system achieves 190MHz with a high throughput of 24.32Gb/s. From the above mentioned data, this pipelined design makes efficient use of FPGA device we choose.

In the folded design, cipher text blocks can be accepted every 36 clock cycles – 4 cycles for buffering and other 32 cycles for transformation. Only 98 IOB pins, 380 CLB slices and 4 BRAMs are used. The maximal throughput is 740Mb/s.

The specifications of these two designs are listed in Table 1 including three aspects: area, frequency and throughput. Considering these three issues, the pipelined design of high throughput can be implemented in speed-critical occasion especially in real-time use such as ATM encryption. And the folded design can be implemented in area-critical uses for its small area cost—smart card, for example.

It should be noticed that the throughputs of both designs listed in Table 1 are under the condition that each data block has its own unique cipher key. If all the data blocks are encrypted in the same key, the throughput will be even larger than that in Table 1. And in our designs, we do not optimize the algorithm specifically to the requirement of FPGA devices in order to improve performance. However, this would make migrating to other devices easily.

Table 1. The performances of pipelined and folded designs

	Device	Area (CLB slices)	Frequency (MHz)	Throughput
folded architecture	XC4VLX100	380	185	740Mb/s
pipelined architecture	XC4VLX100	9500	190	24.32Gb/s

5. Conclusion

Two different designs and FPGA implementations of SMS4 algorithm are presented in this paper. To generic, these two encryption designs are the only hardware SMS4 encryption designs, reported to date. The proposed pipelined structure which divides the encryption precess into thirty-two pipeline stages, has very small encryption time delay and high throughput. And the proposed folded architecture has small gates number. When implemented, the pipelined encryption design can perform at a data-rate of 24Gbits/sec. And the folded design only needs 4 BRAMs and 380 CLB slices so it can be embedded in nearly all devices of Xilinx FPGA series. Future work will be concentrated on sub-pipelined structure to get higher encryption speed and larger throughput. SMS4 algorithm, the first cryptography algorithm released by Chinese government, is the block cipher encryption standard of WLAN in China. The designs proposed in this paper provide flexible choices both for area-critical and speed-critical situations.

References

[1] http://www.oscca.gov.cn, 2006.

[2] Douglas R. Stinson, *Cryptography: Theory and Practice, Second Edition*, Chapman & Hall/CRC, New York, 2002.

[3] Xilinx Virtex-4 User Guide. http://www.xilinx.com, 2005.

[4] M. McLoone, and J. V. McCanny, "High Performance Single-Chip FPGA Rijndeal Algorithm Implementations," *CHES 2001*, pp. 68-80, 2001.

[5] P. Chodowiec, and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," *CHES 2003, LNCS 2779*, pp. 319-333, 2003.

[6] X. Zhang, and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 12, Issue 9, pp. 957-967, 2004.

[7] K. Stevens, and O. A. Mohamed, "Single-chip FPGA implementation of a pipelined, memory-based AES Rijndael encryption design," *Canadian Conference on Electrical and Computer Engineering*, pp. 1296 - 1299, 2005.