Compact Hardware Implementation of the Block Cipher Camellia with Concurrent Error Detection

Huiju Cheng and Howard M. Heys
Electrical and Computer Engineering
Memorial University of Newfoundland
Email: {chenghuiju, howard}@engr.mun.ca

Abstract—**A compact hardware implementation of a block cipher is attractive for any low-cost embedded application like smart cards. In this paper, a compact hardware architecture for Camellia is investigated. In this architecture, encryption and key scheduling share the same datapath and a four s-box iterative structure is employed. In the hardware design of cryptographic algorithms, concurrent error detection (CED) techniques have been proposed not only to protect the encryption and decryption process from random faults but also from the intentionally injected faults by some attackers. In our design, we also investigate a multiple parity code based error detection scheme. In our CED scheme, all the components are protected and all single-bit faults and most multiple faults will be detected. We study the implementation of the compact architecture for an ASIC and an FPGA. The design requires 14.12K gates with a throughput of 143 Mbps based on 0.18-um CMOS standard cell library and 1052 slices with a throughput of 135 Mbps based on Xilinx Virtex-E v1000efg860 chip. For our concurrent error detection, the hardware overhead is about 83%.**

1. Introduction

Camellia is a 128-bit block cipher jointly developed by NTT and Mitsubishi Electric Corporation in 2000 [1]. It was chosen as a recommended algorithm by the NESSIE (New European Schemes for Signatures, Integrity and Encryption) project in 2003 [2] and was certified as the IETF (Internet Engineering Task Force) standard cipher for XML security URIs, SSL/TLS cipher suites and IPsec in 2005 [3][4][5].

Camellia supports a data size of 128 bits and the key size can be 128, 192, or 256 bits and therefore Camellia has a compatible interface with the Advanced Encryption Standard (AES) [1]. It is a Feistel cipher which only processes half the data block each round and the same datapath can be shared by both encryption and decryption. This Feistel structure provides Camellia with a feature of small hardware design. Low-cost embedded applications like smart cards and handheld wireless devices require low hardware complexity while high speed is not emphasized. Therefore, a compact and efficient hardware implementation of Camellia is attractive for such applications.

This paper is organized as follows: Section 2 briefly describes the structure of Camellia. Section 3 presents a compact S-box design using an inverter based on composite filed arithmetic operations in $GF(2^4)^n$ and it is compared with the straightforward combinational logic based S-box generated from look-up tables. Section 4 investigates a compact architecture for Camellia with a four S-box iterative structure. Section 5 is the Concurrent Error Detection scheme design for our compact architecture of Camellia. Section 6 is the hardware performance analysis for ASIC and FPGA implementations. Section 7 is the conclusion.

2. Structure of Camellia

Camellia requires 22-rounds of data processing composed of three main parts: an 18-round Feistel structure, two FL-function and FL^{-1}-function rounds inserted every 6 rounds, and two input/output whitenings [1]. Figure 1 shows the entire encryption process using 128-bit keys. In the first and last round, the 128-bit data block is XORed with 128-bit round keys. Before the data block is fed to the Feistel network, it is separated into two 64-bit data blocks. The left half goes into the F function together with the 64-bit round key and the output of the F function is XORed with the right half block. At the end of each round, the right and left half block will be exchanged. In the F function, the input 64-bit data is first XORed with the 64-bit round key and then grouped into eight 8-bit data blocks. All of them are separately input to eight S-boxes.

In Camellia, four types of S-boxes are applied and each one consists of a multiplicative inversion and affine transformations. A linear 64-bit permutation follows the nonlinear substitution of S-boxes. The FL and FL^{-1} functions inserted every 6 rounds are used to provide non-regularity between the rounds so that the security of the cipher is increased and these two functions are similarly constructed by logical operations including AND, OR, XOR, and rotations.

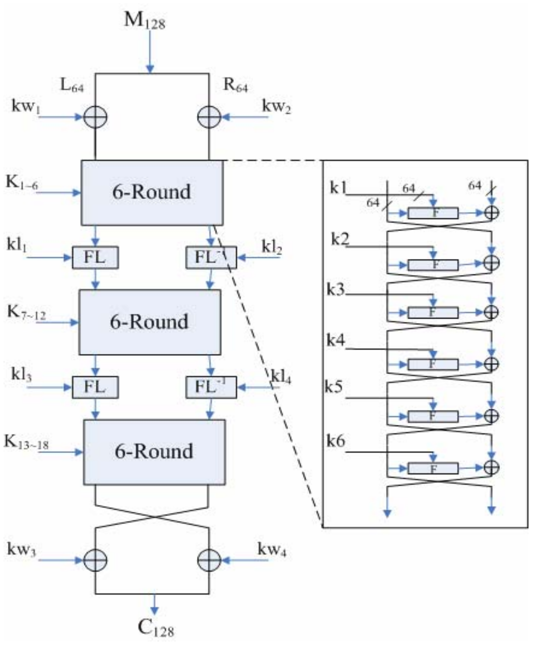

Fig. 1 Encryption process of Camellia [1]

The decryption of Camellia is performed in the same way as encryption except the subkeys should be in a reverse order. The key scheduler shares part of the process with encryption and has additional rotations of the subkeys.

3. Camellia S-box Implementations

The S-box is a major component in the hardware implementation. We have investigated two approaches to implement the S-boxes. One approach is to generate the straightforward combinational logic from the lookup tables described in the Camellia specification [2]. Another approach is a more compact S-box design using an inverter based on composite field arithmetic operations in GF($(2^4)^n$) [6]. The S-boxes of Camellia consist of multiplicative inversion on the Galois field GF(2^8) and affine transformations. We implement the inverter over the subfield GF(2^4) by using irreducible polynomials which are recommended in [7].

In S-box S1, linear affine transformations are performed before and after the multiplicative inversion and the other three S-boxes are related to S1. S2 and S3 are constructed from a 1-bit left rotation and 1-bit right rotation of S1's output, respectively. S4 is implemented as a 1-bit left rotation of S1's input. Table 1 shows the performance comparisons between the four S-boxes implemented based on the two above approaches. For the hardware implementation, a 0.18-um CMOS standard cell library was used and one gate is equivalent to a 2-input NAND gate. We can see the look-up table method is faster than the composite field implementation, but the size is much larger. Hence, in our compact implementation of Camellia, the second approach is preferred.

Table1. S-box performance comparison

Component	Method	Size (gates)	Delay (ns)
S1-S4	GF(2^4)	288	6.9
S1	Table	683	4.12
S2	Table	690	4.25
S3	Table	691	3.82
S4	Table	684	3.92

4. Compact Camellia architecture

A big component of Camellia is the 8 S-boxes used in the 64-bit F function. We reduce the 8 S-boxes into 4 S-boxes with the approach presented in [8]. As shown in Figure 2, the F32 function has four S-boxes (S1, S4, S3, S2) in a twisted order. The 64-bit input data is also divided into two 4-byte blocks (x8, x7, x6, x5) and (x1, x4, x3, x2). These two blocks are fed to the S-boxes after the key addition to generate two output blocks (z8, z7, z6, z5) and (z1, z4, z3, z2). The twisted order of S-boxes and input data is used to satisfy the original 64-bit F function of Camellia where the inputs (x8, x7, x6, x5, x4, x3, x2, x1) should be corresponding to eight S-boxes in the order of (S1, S4, S3, S2, S4, S3, S2, S1). Following the S-boxes is the 64-bit P function which is only composed of XORs.

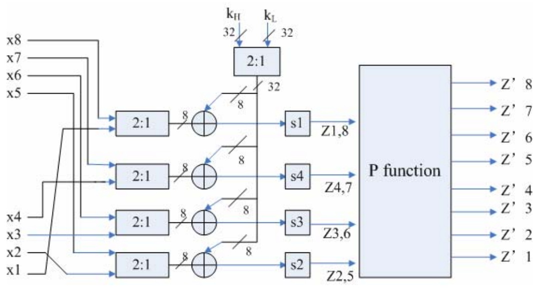

Fig. 2 F32 function with four S-boxes [8]

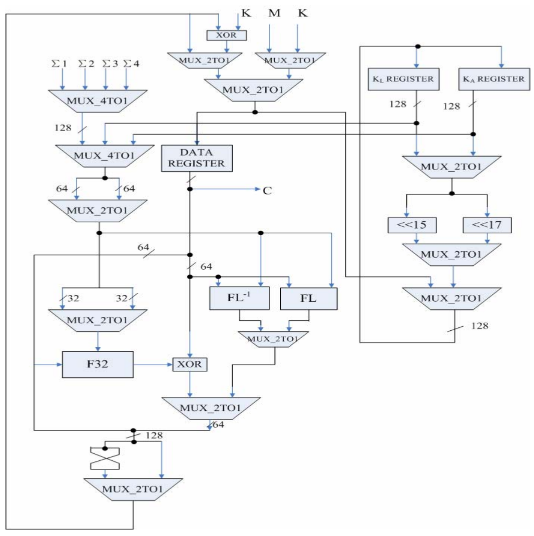

Fig. 3 Datapath of compact architecture

Figure 3 shows the compact datapath for both the encryption process and the key scheduler. The 64-bit F function is divided into two 32-bit F32 functions with four S-boxes. The first round 64-bit output of the P function generated from the first 4-byte block is XORed with the second round output of the P function generated from the second 4-byte input data so that the final result of the original 64-bit F function is obtained. The hardware is reduced while the throughput is also decreased due to the use of the F32 function. Other major components are the FL and FL^{-1} functions. These two functions are separately implemented, even though their structures are very similar to each other, because the merged hardware will have additional relatively costly multiplexers. In the key scheduler, the secret key K_L is stored in register "K_L" and the intermediate key K_A is generated by the shared datapath of the encryption process. During the encryption process, the subkeys are generated by the 15-bit left rotation or 17-bit left rotation of K_L and K_A.

5. Concurrent Error Detection Scheme Design for Camellia

Although the digital hardware circuits are quite reliable today, faults could randomly occur or be induced intentionally for some cryptographic attacks. Concurrent error detection (CED) techniques have been proposed [9][10] not only to protect the encryption and decryption process from random faults, but also from the intentionally injected faults by attackers. In [11], redundancy-based concurrent error detection techniques have been proposed in two approaches: hardware and time redundancy. Another commonly used CED technique is based on the parity code [10][12] and this approach costs low hardware overhead. In our compact hardware implementation of Camellia, a multiple parity code based error detection scheme is investigated and the parity is generated for each byte of data. For the operations of the cipher, the parity of the output is predicted from the input data and is compared with the parity of actual output. This parity-based scheme is applied to all the linear components like multiplexers and XOR gates. When it comes to the non-linear component such as the S-box, the CED approach is based on the straightforward duplication of the component and resulting direct comparison of both outputs. Hence, any type or number of faults in the S-boxes could be detected.

In this CED scheme for both encryption process and key scheduler, all the components are protected and all single-bit faults and most multiple faults will be detected. As long as one byte of data is affected by an odd number of errors, the multiple faults could be detected. As a result, our CED scheme has a high fault coverage.

Figure 4 shows the F32 function with CED of Camellia. The registers m0, m1, m2, m3, m4, m5, m6, and m7 are the eight byte inputs of the F function and p0, p1, p2, p3, p4, p5, p6, and p7 are the eight parities of corresponding bytes. Figure 5 shows the FL component with CED. The registers m0, m1, m2, and m3 are the left four bytes of the input data of FL function.

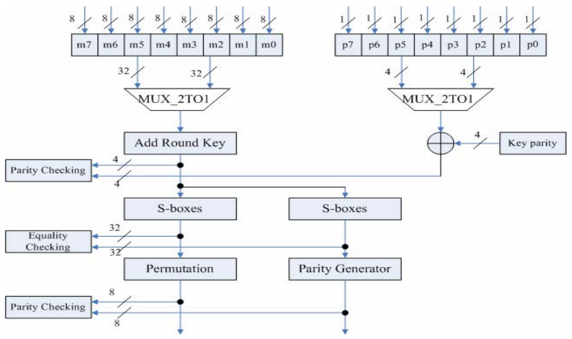

Fig. 4 F32 Function with CED

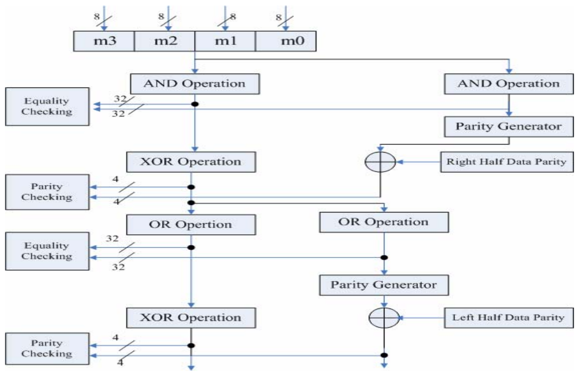

Fig. 5 FL Function with CED

In our compact implementation of Camellia, the multiplexers and registers are the other simple basic components and they cost a large percentage of the whole hardware resource. Therefore, CED is also applied to protect these components. For multiplexers, no modification is needed for the prediction of the input parities since the selected data will not change. Byte-based parities will be generated for each input of the multiplexer and are then sent into another smaller multiplexer. Finally, the selected parities will be compared with the parities of the actual output of the multiplexer. For the register, one parity bit for each byte is attached within the register. For example, one 128-bit input and 128-bit output register will be extended into a 128-bit input and 144-bit output register which includes 16-bit predicted parities. The parity checking is performed between the predicted parities and the actual parities. For both registers and multiplexers, all the single-bit faults and most multiple faults would be detected based on the parity code CED scheme.

In our whole CED scheme for Camellia, 22 checking points are inserted and all the components are under protection with high fault coverage at the cost of large amount of hardware overhead which includes the parity prediction circuit and duplicated components such as S-boxes. Since the encryption process and error detection can be performed at the same time, there is no notable error detection delay in our CED scheme designed for Camellia.

6. Hardware Performance in ASIC and FPGA

Based on the compact architecture of the encryption process and key scheduler, a 0.18-um CMOS standard cell library and a Xilinx FPGA chip Virtex-E v1000efg860 [13] are applied in our hardware performance analysis. With the 0.18-um CMOS standard cell library, Synopsis has been applied as our synthesis tool. We find the datapath shared by the encryption process and key scheduler based on the S-boxes using GF(2^4) requires a size of 14.12K with a throughput of 143Mbps. In addition, 1052 slices with a throughput of 135 Mbps is required based on Xilinx Virtex-E v1000efg860 chip. After applying our CED scheme to the compact architecture of Camellia, the size of the datapath for the CMOS implementation rises to 26K gates resulting in a hardware overhead of 83%.

7. Conclusion

A compact implementation with a four S-box iterative structure of Camellia has been investigated in this paper and the performance is evaluated by using 0.18-um CMOS standard cell library and a Xilinx Virtex-E v1000efg860 chip. As the S-box is a major component of Camellia, the number of S-boxes applied in the datapath is reduced from eight to four for a more compact architecture. We have also investigated a multiple parity code based error detection scheme for our compact architecture of Camellia and all the components are protected with a high fault coverage.

References

[1] K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-Bit Block Ciphers Suitable for Multiple Platforms – Design and Analysis", *Lecture Notes in Computer Science*, Vol. 2012, pp. 39-56, 2001.

[2] K. Aoki, T. Ichikawa, M. Kansa, M. Matsui, S. Moriai, Nakajima, and T. Tokita, "Specification of Camellia - a 128-bit Block Cipher", http://www.cosic.esat.kuleuven.be/nessie/workshop/submissions, 2000.

[3] D. Eastlake, "Additional XML Security Uniform Resource Indentifiers (URIs)", RFC4051, 2005.

[4] S. Moriai, A. Kato, M. Kanda, "Addition of Camellia Cipher Suites to Transport Layer Security (TLS)", RFC4132, 2005.

[5] A. Kato, S. Moriai, M.Kanda, "The Camellia Cipher Algorithm and Its Use With IPsec", RFC4312, 2005.

[6] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC Implementation of the AES SBoxes", *Lecture Notes in Computer Science*, Vol. 2271, pp. 67-78, 2002.

[7] A. Satoh, S. Morioka, "Unified Hardware Architecture for 128-Bit Block Ciphers AES and Camellia", *CHES 2003*, *Lecture Notes in Computer Science*, Vol. 2779, Springer, 2003, pp. 304-318.

[8] A. Satoh, S. Morioka, " Hardware-focused Performance Comparison for the Standard Block Ciphers AES, Camellia, and Triple-DES", *Lecture Notes in Computer Science*, Vol. 2851, Springer 2003, pp. 252-266, 2003.

[9] R. Karri, K. Wu, P. Mishra, Y. Kim, "Fault-based side-channel cryptanalysis tolerant Rijndael symmetric block cipher architecture", *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'01),* 2001.

[10] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A Parity Code Based Fault Detection for an Implementation of the Advanced Encryption Standard", 2002 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2002), pp 51-59, November 2002.

[11] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard", IEEE Transactions on Computers, Vol. 52, No. 4, April 2003.

[12] K. Wu, R. Karri, G. Kouznetzov and M. Goessel, "Low Cost Concurrent Error Detection for the Advanced Encryption Standard", International Test Conference 2004 (ITC 2004), pp. 1242-1248, 2004.

[13] Xilinx: Virtex-E Data sheet,

http://www.xilinx.com.