High-Performance Concurrent Error Detection Scheme for AES Hardware

Akashi Satoh[1], Takeshi Sugawara[2], Naofumi Homma[2], and Takafumi Aoki[2]

[1] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology (AIST)
Sotokanda, Tokyo, Japan
akashi.satoh@aist.go.jp
[2] Graduate School of Information Sciences, Tohoku University
Sendai, Miyagi, Japan
{sugawara,homma}@aoki.ecei.tohoku.ac.jp,
aoki@ecei.tohoku.ac.jp

Abstract. This paper proposes an efficient concurrent error detection scheme for hardware implementation of the block cipher AES. The proposed scheme does not require an additional arithmetic unit, but simply divides the round function block into two sub-blocks and uses the sub-blocks alternately for encryption (or decryption) and error detection. The number of clock cycles is doubled, but the maximum operating frequency is increased owing to the shortened critical path of the sub-block. Therefore, the proposed scheme has a limited impact on hardware performance with respect to size and speed. AES hardware with the proposed scheme was designed and synthesized using a 90-nm CMOS standard cell library with size and speed optimization options. The compact and high-speed implementations achieved performances of 2.21 Gbps @ 16.1 Kgates and 3.21 Gbps @ 24.1 Kgates, respectively. In contrast, the performances of AES hardware without error detection were 1.66 Gbps @ 12.9 Kgates for the compact version and 4.22 Gbps @ 30.7 Kgates for the high-speed version. There is only a slight difference between the performances with and without error detection. The performance overhead caused by the error detection is evaluated at the optimal balance between size and speed and was estimated to be 14.5% at maximum. Conversely, the AES hardware with the proposed scheme had better performance in some cases. If pipeline operation is allowed, as in the CTR mode, throughputs can easily be boosted by further dividing the sub-blocks. Although the proposed error detection scheme was applied to AES in the present study, it can also be applied to other algorithms efficiently.

1 Introduction

The fault injection attack is a physical attack to obtain internal secret information from cryptographic modules by causing a malfunction in operating units or the sequencer logic using electrical noise injection on the power source or clock signal or by illuminating the module by an electronic beam. In 1996, Boneh,

E. Oswald and P. Rohatgi (Eds.): CHES 2008, LNCS 5154, pp. 100–112, 2008.

Demillo, and Lipton [1] proposed a fault injection attack against public key cryptosystems, and Biham and Shamir [2] extended this attack to symmetric key cryptosystems. Since then, research on the fault injection attack has been rapidly evolved [3-5], and several papers have proposed attacks on the standard block cipher AES [7-13].

On the other hand, several countermeasures that detect errors in processing have also been proposed [14-29]. Fig. 1 summarizes the conventional error detection schemes for block cipher hardware with a loop architecture that iteratively uses one round function block. The figures illustrate error detection schemes for encryption process, but the same schemes can be applied to decryption circuits and to implementations merging encryption and decryption datapaths.

In Fig. 1(a), the data in register RegX is processed by the round function block for encryption (Enc), and then an error detection code, such as a parity bit, is generated. The code is compared with an expected value output from another data path (Predict) [14, 17-20]. It is very easy to calculate the expected value for linear functions by using a small amount of hardware resources, and thus several studies have proposed error detection codes for the non-linear substitution function S-box [15, 16, 21-23]. The operation "Predict" is much simpler than "Enc" and usually outputs a smaller number of bits, and thus it is impossible to detect all of the error patterns. Therefore, the trade-off between overhead of the additional circuit, "Predict", and the error detection ratio should be considered carefully.

In Fig. 1(b), two encryption operations for the same data in the register RegX are performed by duplicated round function blocks, and the results are compared [24]. The architecture of Fig. 1(c) has encryption and decryption datapaths, and the data in RegX is encrypted and soon decrypted. The result is then compared with the original data in RegY [24, 25]. These two schemes have a disadvantage in that the hardware size is almost double compared to that of the circuit without error detection.

The scheme of Fig. 1(d) encrypts the same data twice using one round function block and two results are compared [26, 27]. In Fig. 1(e), the round function block supports both encryption and decryption, and confirms that encrypted data can be decrypted correctly. This scheme can also be applied efficiently to the round function $F(x)$ with the characteristic of $x = F(F(x))$ [28]. The drawback of these schemes is that twice as many clock cycles are required.

Fig. 1(f) is similar to Fig. 1(d), where two encryptions are performed to confirm that the same encrypted data are generated, but the round function block is divided into two sub-blocks and encryption and error detection (another encryption) are performed simultaneously in each sub-block [29]. Hardware size and the number of clock cycles are almost the same between these schemes, but the maximum operating frequency of Fig. 1(f) is much higher than any other scheme in Fig. 1 because the critical path (the round function block) is halved.

Fig. 1(f) is the best scheme in terms of circuit size and speed, but the use of the same datapath for two encryptions (one of which is for error detection) causes a major problem. When an attacker injects an electron beam to cryptographic

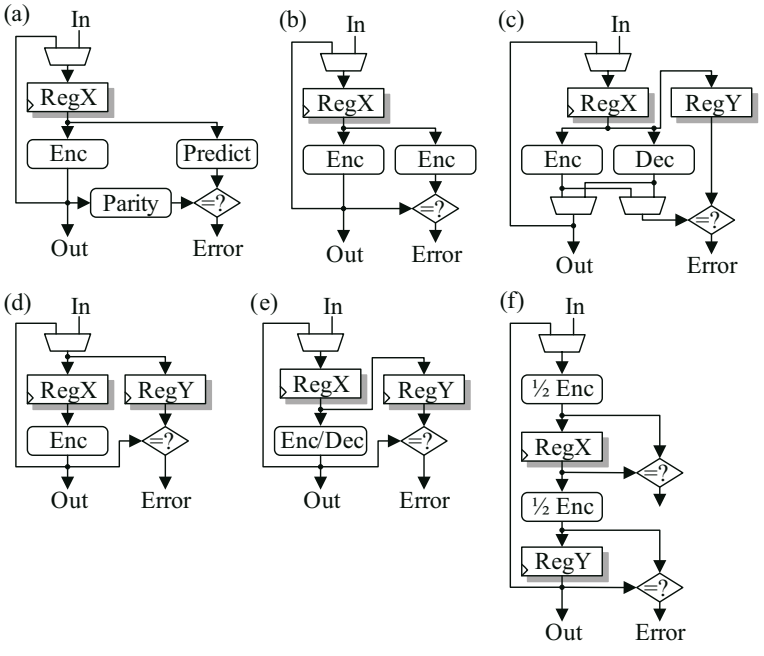

Fig. 1. Conventional error detection schemes (Encryption)

circuit, it is very difficult to control the beam precisely in order to make an error in only one clock period. In contrast, it is incomparably easy to keep the beam on during certain periods and to keep the circuit in failure. In this case, the same error occurs repeatedly and thus the scheme of Fig. 1(f) that repeats the same encryption twice for data checking cannot detect the error. The beam might cause different types of errors in each cycle, but defects on transistor devices and metal interconnections in LSI chips always make the same error, and thus the scheme of Fig. 1(f) is unworkable for these static errors.

In order to solve these problems, this paper proposes a new error detection scheme that performs encryption (or decryption) and error detection simultaneously in different operating blocks with limited impact on hardware size and speed. AES hardware using the proposed scheme is designed and synthesized using an ASIC library, and the effectiveness of the scheme is evaluated.

2 Proposed Error Detection Scheme

2.1 Normal AES Circuit

Fig. 2 shows a block diagram of an AES circuit using a loop-architecture based on the compact implementation proposed in references [30] and [31], which does not support error detection feature. A 128-bit input is encrypted (or decrypted)

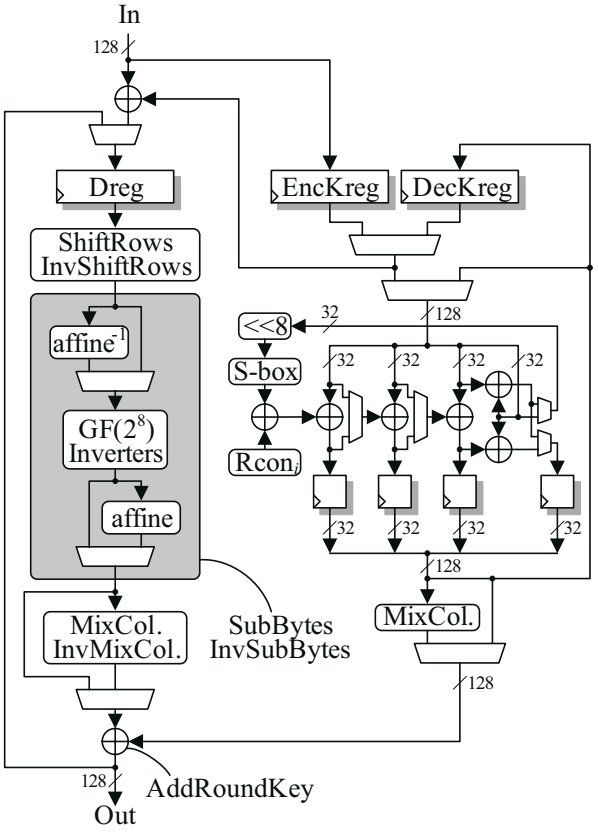

Fig. 2. Normal AES circuit

with a 128-bit secret key in 10 clock cycles. The encryption and decryption paths are merged by sharing $GF(2^8)$ inverters in S-boxes and common terms between the permutation functions MixColumns and InvMixColumns. The circuit size is almost halved in comparison to an implementation with two different datapaths for encryption and decryption. In order to merge the datapaths, the location of AddRoundKey and InvMixColumns (shown as InvMixCol. in Fig. 2) is switched from the original order. Then, the MixColumns function block is placed at the output of the key scheduler on the right in Fig. 2 to compensate the side effect. In the next section, the proposed error detection scheme is explained in contrast with this normal architecture.

2.2 AES Circuit with the Proposed Scheme

The proposed scheme uses a datapath that supports both encryption and decryption, which is similar to that shown in Fig. 2, and divides the merged round function block into pre- and post-blocks. Then, one of the blocks is used for

encryption (or decryption), and another block is used for decryption (or encryption) for error detection. Fig. 3 shows the outline of the proposed scheme in the encryption mode. Decryption can be carried out in a similar way. SR and ISR denote ShiftRows and InvShiftRows, respectively, SB and ISB denote SubBytes and InvSubBytes, respectively, and MX and IMX denote MixColumns and InvMixColumns, respectively. In Fig. 3(b), the order of ISB and ISR is switched to share components between the encryption (Enc.) and decryption (Dec.) flows of Fig. 3(a). Then SR and ISR are merged, and SB and ISR are merged, and a half round function block, BlockS, is composed. The permutation functions MX and IMX are also merged and compose another half round block, BlockM, with two 128-bit XORs (AddRoundKey). These two blocks are used alternately for encryption (or decryption) and error detection, as shown in Fig. 3(c), and each round of Round1, $\cdots$, Round10 in Fig. 3(a) is processed in two clock cycles as Round1X, Round1Y, $\cdots$, Round10X, Round10Y. The number of operating cycles is doubled, but the maximum operating frequency is boosted because the critical path of the round function block is divided into two sub-blocks. Therefore, this has a minor impact on the operating speed. It is also possible to increase the operating frequency of the normal AES circuit in Fig. 1 by dividing the round function block. However, it is only efficient for the Electric Code Book (ECB) and Counter (CTR) modes that can process 128-bit data blocks independently but cannot increase the speed for feedback modes, such as Cipher Block Chaining (CBC). When speed performance with the CTR is the first priority, the proposed architecture can also respond to this requirement by increasing the number of pipeline stages from 2 to $2n$. For example, it is easy to perform two encryptions (or decryptions) and two decryption (or encryption) as error detections by dividing sub-block BlockS and BlockM into two smaller sub-blocks each.

In Fig. 3(c), the XOR output from Round0 is processed by the SR and SB functions of BlockS in the clock cycle Round 1X, and the result is fed to BlockS and BlockM. In the following cycle Round 1Y, the inverse operation of Round1X is performed by BlockS, and the result is compared with the input to BlockS in the previous cycle Round1X for error detection. At the same time, the MX and XOR (AddRoundKey) operations are executed by BlockM to continue the encryption process. In the next cycle Round2X, BlockS performs the following encryption process, and BlockM checks the previous result. In a similar manner, the remainder of the encryption and error detection operations are executed by BlockS and BlockM interchangeably. The same round function blocks are used for encryption and error detection, but these operations are different, and thus static errors caused by defects in LSIs can be detected, while the scheme of Fig. 1(f) cannot find the errors, where the same operation is executed twice by the same function block for encryption and error detection.

Fig. 4 shows the datapath architecture of the AES circuit using the proposed error detection scheme. This architecture does not switch the order of AddRoundKey and InvMixColumns to share the XOR gates for AddRoundKey, as in Fig. 2. The critical path of the round function block in Fig. 2 is shortened by

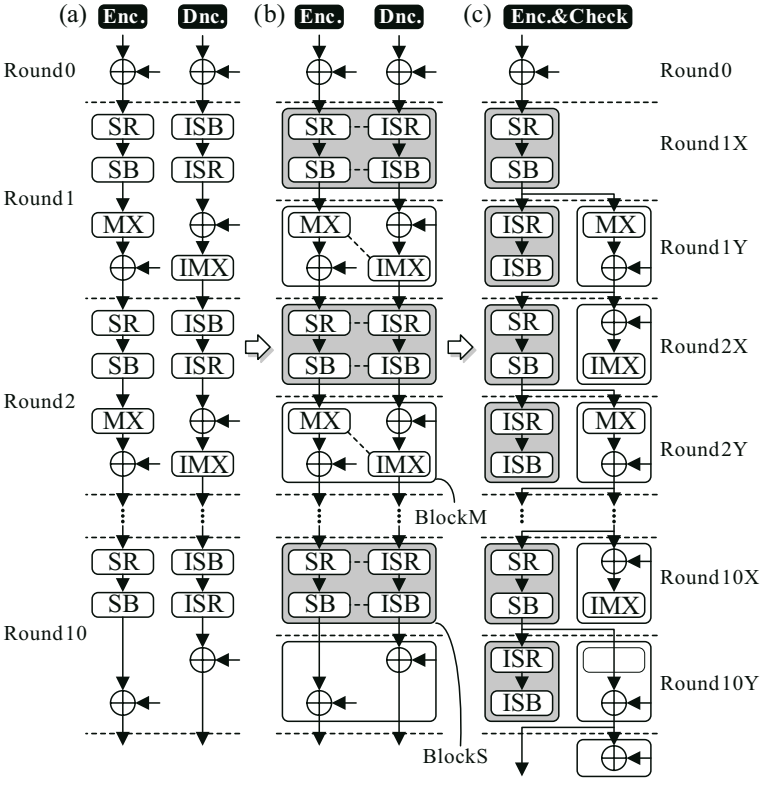

Fig. 3. Proposed error detection scheme for AES (Encryption)

sharing the XOR gate, but the additional MixColumns block is required at the output port of the key scheduler. In contrast, when the proposed scheme that divides the round function block into two sub-blocks was applied, implementations without sharing the XOR gates showed better performance in balance between size and speed. When the signal delay time for the round function block is shortened by the division, the key scheduler becomes the critical path. Therefore, the scheduler is also divided in two by inserting a register and uses two clocks to generate one round key. In Fig. 4, the datapaths of the round function block and the key scheduler are divided at the end of S-boxes for simplicity, but pipeline registers are actually placed inside the S-boxes in order to balance the signal delay times before and after the registers.

Even if the round function block works correctly, the key scheduler can also be attacked [11, 12, 13], or malfunction in a control counter may output intermediate data soon after the first round key is XORed without waiting for the completion of 10-round operations [3]. In order to prevent this, the key scheduler in Fig. 4 compares the round-key generated in the round key register with the pre-calculated keys in the key registers DecKreg or EncKreg in the final round of encryption or decryption, respectively. The register DecKreg holds the first

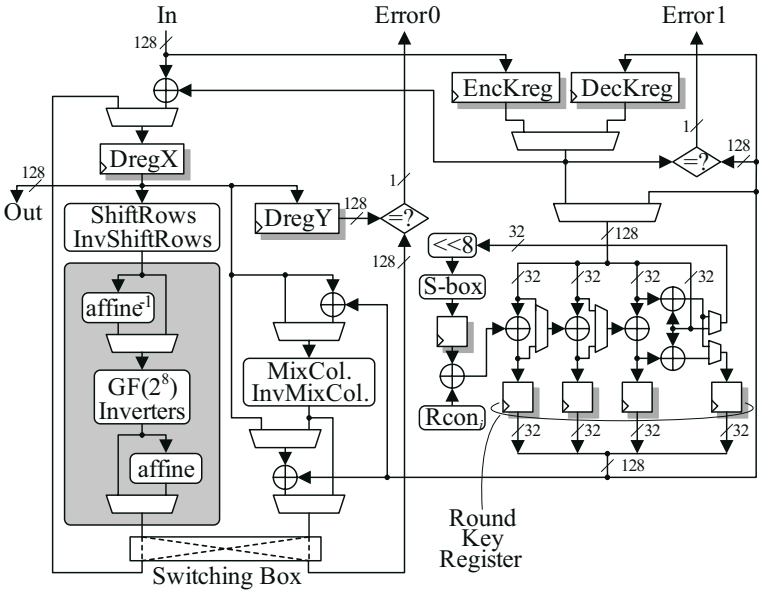

Fig. 4. AES circuit with the proposed error detection scheme

round-key for decryption that is the last round-key for encryption, and the register EncKreg holds the first round-key for encryption that is the last round-key of decryption. Even if an attacker can flip a few bits in the control counter to skip the round operations, it is impossible to control the unknown 128-bit round-key to match the final value.

2.3 Example Operation

Fig. 5 shows the example encryption process of the AES circuit with the proposed error detection scheme. It is assumed that the initial key for decryption K10 has been calculated from the initial key K0 for encryption, and the keys K0 and K10 are stored in registers EncReg and DecReg, respectively. In Fig. 5(a), a plaintext input XORed with the initial key K0 has been stored in the data register DregX as D0, and the first half of the round function (Shiftrows and SubBytes) is applied to the data D0, and then the result D1X is fed back to the register DregX. At the same time, the data D0 is transferred to the register DregY for error detection, and the key register generates the first round-key K1 from K0.

In Fig. 5(b), the datapath for encryption in Fig. 5(a) is used for decryption as error detection. The data D1X in DregX is processed by InvShiftRows and InvSubBytes, and the result is compared with the data D0 in RegY. In the other data path, the last half of the round function, MixColumns and AddroundKey (XOR with the round-key K1), is applied to the data D1X, and the result of the first round function is obtained in DregX as D1.

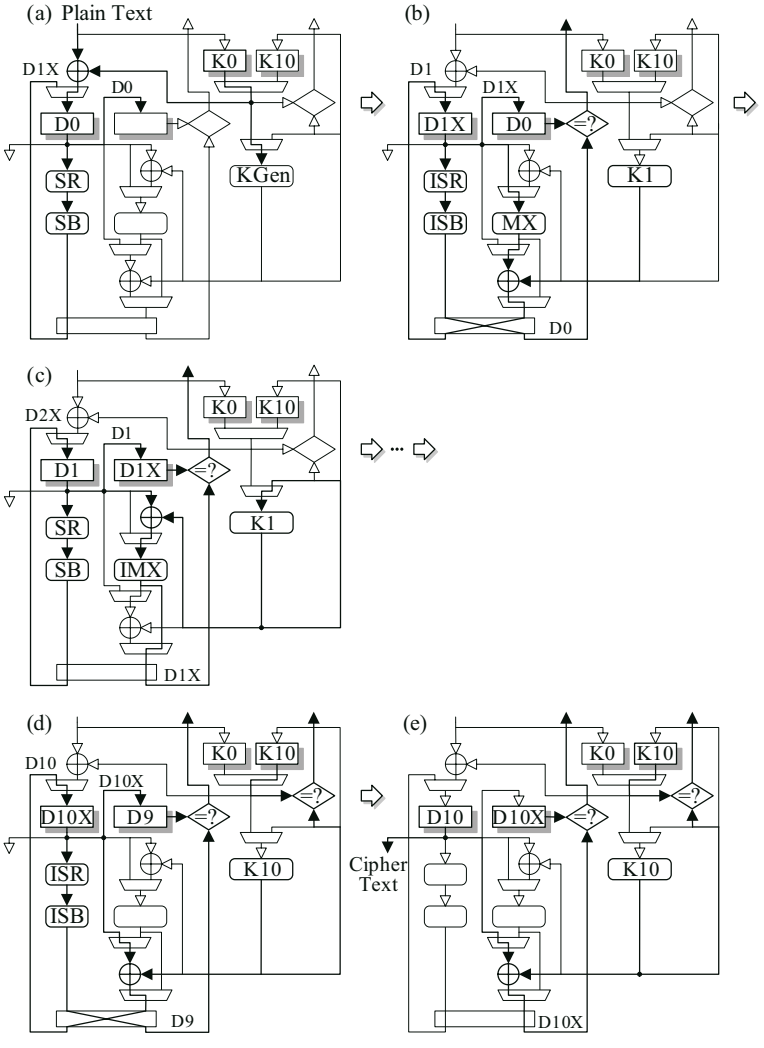

Fig. 5. Example operation of the proposed AES circuit

In Fig. 5(c), the same encryption datapath of Fig. 5(a) encrypts the data D1 in DregX to D2X, and the datapath on the right (InvMixColumns and XOR) decrypts the same data D1 to D1X for error detection. The output from the right datapath is then compared with the data in RegY. In a similar manner, the encryption and error detection process are continuously performed.

Fig. 5(d) shows the operation in the final round, where AddRoundKey with the 10-th round key K10 for encryption, and InvShiftRows and InvSubBytes are performed for error detection. As shown in Fig. 3(c), the MixColumns block is bypassed for this final round. In order to check whether the sequencer logic and key scheduler worked correctly and all 10 rounds are processed without skip,

the final round key generated in the round key register is compared with the pre-calculated key K10 in EncKreg. The ciphertext D10 can be output in this cycle, but it is output in the next cycle of Fig. 5(e) after confirming that D10 can return to D10X. The next plaintext cannot be input before this final check and thus requires 21 clocks, 20 clocks (= 10 rounds × 2 clocks) + one additional clock for the final check, to encrypt (or decrypt) one data block.

3 ASIC Performance Comparison

Table 1 shows a comparison of the performance between the AES circuits with and without the proposed error detection scheme, as shown in Figs. 2 and 4. The designs were synthesized by a Synopsis Design Compiler using a 90-nm CMOS standard cell library. In addition to size and speed optimizations, implementations that achieve the highest hardware efficiency, defined as throughput per gate, are shown.

The signal delay time for the round function block is approximately halved by using the proposed scheme, but the maximum operating frequency is not doubled because of the setup and hold times of the inserted register. In addition, the proposed scheme requires an additional clock cycle and additional hardware resources for error detection. Therefore, simple prediction may indicate that the proposed scheme is slower and larger than the simple AES circuit without the error detection scheme. However, the throughputs of compact implementations are 2.21 Gbps with 16.1 Kgates for the proposed scheme and 1.66 Gbps with 12.9 Kbps for the simple architecture. Thus, the proposed scheme is faster. Moreover, the gate counts of the high-speed versions are 24.1 Kgates with 3.21 Gbps for the proposed scheme and 30.7 Kgates with 4.22 Gbps for the simple architecture. Thus, the proposed scheme is smaller. This is because the longer combinatorial logic path in the round function block of the simple architecture causes wide variations in logic synthesis. The range of gate counts and throughputs in Table 1 are ×2 for the simple architecture, while this range is within ×1.5 for the proposed scheme. To achieve compact implementation, it is important to reuse gate logic, even though the critical path becomes longer, and to use smaller cells, even though their drivability is lower. On the other hand, parallel processing without

Table 1. Hardware performance comparison

Architecture	Clock Cycles	Size (gates)	Maximum Frequency (MHz)	Throughput (Mbps)	Hardware Efficiency (Kbps/gate)	Optimization
Proposed (Fig. 4)	21	16,099	362.32	2,208.40	137.18	Size
		17,087	406.50	2,477.70	145.01	Efficiency
		24,114	526.32	3,208.00	133.04	Speed
Normal (Fig. 2)	10	12,949	129.37	1,655.90	127.88	Size
		20,003	265.25	3,395.20	169.48	Efficiency
		30,708	330.03	4,224.40	137.57	Speed

(90-nm CMOS, 1 gate = 2-input NAND, worst condition).

sharing gate logic and use of large cells with higher drivability are efficient for high-speed implementation. This means that smaller circuits become slower and faster circuits become larger. Therefore, the simple implementation with a wide range of synthesis optimization had smaller but "slower" performance for the compact implementation, and the high-speed version is faster but "larger" than the proposed architecture.

The results indicate that total hardware performance cannot be determined by simply measuring gate counts and throughput, and thus the performance overhead caused by the error detection circuit cannot be evaluated either. Therefore, as the criterion, we use the balance between hardware size and operating speed, that is, the hardware efficiency is defined as the throughput per gate. However, the hardware efficiency still varies somewhat depending on the synthesis constraints. Consequently, the optimal balance between size and speed, i.e., the highest hardware efficiency, was chosen as the score of the hardware performance. To investigate the highest hardware efficiency, logic synthesis was repeated several times by changing the constraints. Then, the proposed AES architecture and the simple AES architecture achieved efficiencies of 145.0 Kbps/gate (= 2.48 Gbps/17.1 Kgates) and 169.5 Kbps/gate (= 3.40 Gbps/20.0 Kgates), respectively. The efficiency of the proposed scheme is 85.5% compared to the simple architecture, and thus we can say that the performance overhead of the error detection scheme is at most 14.5%. Meanwhile, in many cases, the AES circuit with the error detection showed better performances. These results clearly demonstrate the advantage of the proposed scheme.

4 Conclusion

This paper proposed an error detection scheme for the AES circuit, and evaluated its performance using a 90-nm standard cell library. The scheme divides a round function block into two sub-blocks and uses them alternatively for encryption (or decryption) and error detection. Therefore, no extra calculation block is needed, even though only a pipeline register, a selector and a comparator are added. The number of operating cycles is doubled, but the operating frequency is boosted because the round function block in the critical path is halved. Therefore, the scheme has only a minor impact on hardware performance.

Logic synthesis was repeated by changing the optimization conditions, and the AES circuit with the proposed scheme achieved a range of 16.1 $\sim$ 24.1 Kgates for hardware size and 2.21 $\sim$ 3.21 Gbps for throughput. Those of the simple architecture without error detection are 12.9 $\sim$ 30.7 Kgates and 1.66 $\sim$ 4.22 Gbps. The simple implementation has a longer combinatorial logic path, which leads to a wider range of performance optimization. These different ranges make it difficult to compare the performance between the proposed and simple architectures. Therefore, the highest hardware efficiency (throughput/gate), which gives the optimal balance between hardware speed and size was chosen for the performance comparison. The hardware efficiencies are 145.0 kbps/gate for the proposed scheme and 169.4 Kbps/gate for the simple implementation, and thus

the performance overhead due to the error detection is only 14.5%. In addition, the AES circuit with the proposed scheme had better performance than the simple implementation depending on the constraints of logic synthesis.

Although the round function block was divided by 2 in the above implementations, it should be possible to increase the number of pipeline stages to 4, 6, and 8, in which half of the stages are used for encryption and the other half are used for error detection. This is a very efficient way to achieve a much higher throughput when pipeline operation, such as that for the CTR mode, is possible. The proposed scheme does not depend greatly on the algorithm, and thus it can also be applied to hardware implementations of several coding algorithms, as well as cryptographic hardware. As a result, the proposed error detection scheme has significant advantages in both efficiency and versatility.

We have developed experimental ASIC and FPGA boards called SASEBO (Side-channel Attack Standard Evaluation BOard) and have distributed these boards to research institutes in an attempt to contribute to establish standard evaluation criteria and test requirements for cryptographic modules against physical analysis attacks including fault injection attacks. A cryptographic ASIC chip with countermeasures is currently under development, and the AES circuit proposed in this paper will be implemented on the chip. Detailed technical information and specifications about the experimental chip and the boards will be disclosed on the Website of the SASEBO project [32].

References

1. Boneh, D., Demillio, R., Liotin, R.: On the Importance of Checking Crypto-graphic Protocols for Fault. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 37–51. Springer, Heidelberg (1997)
2. Biham, E., Shamir, A.: Differential Fault Analysis of Secret Key Cryptosys-tems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 513–525. Springer, Heidelberg (1997)
3. Anderson, R., Kuhn, M.: Low Cost Attacks on Tamper Resistant Devices. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 125–136. Springer, Heidelberg (1998)
4. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attack. IACR ePrint archive, Report 2004/100 (2004)
5. Giraud, G., Thiebeauld, H.: A Survey on Fault Attacks. In: Proc. Sixth Smart Card Research and Advanced Application IFIP Conf. (CARDIS 2004), pp. 159–176 (August 2004)
6. National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES) FIPS Publication 197 (November 2001), `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`
7. Blömer, J., Seifert, J.-P.: Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
8. Blömer, J., Krummel, V.: Fault Based Collision Attacks on AES. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 106–120. Springer, Heidelberg (2006)

9. Piret, G., Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, With Application to the AES avd Khazad. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 77–88. Springer, Heidelberg (2003)
10. Dusart, P., Letourneux, G., Vivolo, O.: Differential Fault Analysis on AES. Cryptology ePrint Archive, Report2003/010 (2003), http://eprint.iacr.org/2003/010.pdf
11. Chen, C.-N., Yen, S.-M.: Differential Fault Analysis on AES Key Schedule and Some Countermeasures. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 118–129. Springer, Heidelberg (2003)
12. Giraud, C.: DFA on AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 27–41. Springer, Heidelberg (2005)
13. Takahashi, J., Fukunaga, T., Yamakoshi, K.: DFA Mechanism on the AES Key Schedule. In: Proc. Fault Diagnosis and Tolerance in Cryptography (FDTC 2007), pp. 62–72 (September 2007)
14. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P., Piuri, V.: Error Analysis and Detection Procedures for a Hardware Implementation of the Advanced Encryption Standard. IEEE Trans. Comp. Special Issue on Cryptographic Hardware and Embedded Software 52(4), 492–505 (2003)
15. Bertoni, G., Breveglieri, L., Koren, I., Maistri, P.: An Efficient Hardware-Based Fault Diagnosis Scheme for AES Performances and Cost. In: Proc. the 19th IEEE Int. Sym. Defect and Fault Tolerance in VLSI Systems (DFT 2004), pp. 130–138 (October 2004)
16. Breveglieri, L., Koren, I., Maistri, P.: Incorporating Error Detection and Online Reconfiguration into a Regular Architecture for the Advanced Encryption Standard. In: Proc. the 20th IEEE Int. Sym. Defect and Fault Tolerance in VLSI Systems (*DFT 2005*), pp. 72–80 (October 2005)
17. Karri, R., Kuznetsov, G., Gossel, M.: Parity-Based Concurrent Error Detection of Substitution-Permutation Network Block Ciphers. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 113–124. Springer, Heidelberg (2003)
18. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Robust Protection against Fault-Injection Attacks on Smart Cards Implementing the Advanced Encryption Standard. In: Proc. 2004 International Conference on Dependable Systems and Networks (*DSN 2004*), pp. 93–101 (July 2004)
19. Karpovsky, M., Kulikowski, K.J., Taubin, A.: Differential Fault Analysis Attack Resistant Architectures for the Advanced Encryption Standard. In: Proc. Sixth Smart Card Research and Advanced Application IFIP Conference (CARDIS 2004), pp. 177–192 (August 2004)
20. Yen, C.-H., Wu, B.-F.: Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard. IEEE Trans. Comp. 55(6), 720–731 (2006)
21. Wu, K., Karri, R., Kuznetsov, G., Goessel, M.: Low Cost Concurrent Error Detection for the Advanced Encryption Standard. In: Proc. The 2004 Int. Test Conf., pp. 1242–1248 (October 2004)
22. Kermani, M.M., Masoleh, A.R.: Parity-Based Fault Detection Architecture of S-box for Advanced Encryption Standard. In: Proc. the 21st IEEE Int. Symp. De-fect and Fault-Tolerance in VLSI Systems (DFT 2006), pp. 572–580 (December 2006)
23. Kermani, M.M., Masoleh, A.R.: A Structure-independent Approach for Fault Detection Hardware Implementations of the Advanced Encryption Standard. In: Proc. Fault Diagnosis and Tolerance in Cryptography (FDTC 2007), pp. 47–53 (September 2007)

24. Wu, K., Mishra, P., Karri, R.: Concurrent Error Detection of Fault-Based Side-Channel Cryptanalysis of 128-Bit RC6 Block Cipher. Special Issue on Defect and Fault Tolerance in VLSI Systems. Microelectronics Journal 34(1), 31–39 (2003)
25. Karri, R., Wu, K., Mishra, P., Kim, Y.: Concurrent Error Detection Schemes for Fault-Based Side-Channel Cryptanalysis of Symmetric Block Ciphers. IEEE Trans. CAD of Integrated Circuits and Systems 21(12), 1509–1517 (2002)
26. Anghel, L., Nicolaidis, M.: Cost Reduction and Evaluation of a Temporary Faults Detecting Technique. In: Proc. Design Automation and Test in Europe (DATE 2000), pp. 591–597 (March 2000)
27. Maistri, P., Vanhauwaert, P., Leveugle, R.: A Novel Double-Data-Rate AES Architecture Resistant against Fault Injection. In: Proc. Fault Diagnosis and Tolerance in Cryptography (FDTC 2007), pp. 54–61 (September 2007)
28. Joshi, N., Wu, K., Karri, R.: Concurrent Error Detection Schemes for Involution Ciphers. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 400–412. Springer, Heidelberg (2004)
29. Malkin, T.G., Standaert, F.-X., Yung, M.: A Comparative Cost/Security Analysis of Fault Attack Countermeasures. In: Breveglieri, L., Koren, I., Naccache, D., Seifert, J.-P. (eds.) FDTC 2006. LNCS, vol. 4236, pp. 159–172. Springer, Heidelberg (2006)
30. Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A Compact Rijndael Hardware Architecture with S-box Optimization. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 239–254. Springer, Heidelberg (2001)
31. Chodowiec, P., Gaj, K.: Very Compact FPGA Implementation of the AES Algorithm. In: D.Walter, C., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 319–333. Springer, Heidelberg (2003)
32. Side-channel Attack Standard Evaluation Board (SASEBO), `http://www.rcis.aist.go.jp/special/SASEBO/`