TWO BIT ERROR DETECTION AND SINGLE BIT CORRECTION FOR AES

G SARANYA DEVI
Department of Electronics and Communication, Sona College of Technology(Autonomous), Salem, India.
saranyagopal91@gmail.com

ABSTRACT

The Advanced Encryption Standard (AES) is the private-key cryptographic standard for transferring block of data securely. The faults that accidently occur in the hardware implementation of this standard may cause invalid encrypted/decrypted output that results in losing the original message and/or leaking the secret key information. In order to reach a fault detection scheme capable of detecting faults occurring in the hardware implementation of the AES, the structures of the S-box and its inverse have been modified so that single faults lead to either zero or odd number of erroneous output bits. Thus, such errors can be detected and corrected using SECDED.

Keyword: AES, S-box, secret key, fault detection, SECDED.

1. INTRODUCTION

Advanced Encryption Standard (AES) is a replacement for triple DES (3DES which not only has comparable security strength, but also achieve significant efficiency improvement for implementation on software or hardware. With the wide-spread of the AES applications, differential types of faults may occur from time to time. Several efforts were devoted into fault tolerance of the transformations and rounds in AES algorithm. Guido Bertoni et al presented a fault model for AES and analyzed the behavior of the AES algorithm in the presence of faults[1]. They also proposed a fault detection technique for a hardware implementation of the AES algorithm which is based on the parity codes [2]. Moreover, they developed an analytical error model for the parity-based EDC for the AES encryption algorithm and are capable of locating single-bit transient and permanent faults [3, 4]. Later, the same group further described the complete error model extended to include the Key Schedule (KS) part and presented the results of the software simulations of the model [5]. L. Berveglieri et al proposed an extension to an existing AES architecture to provide error detection and fault tolerance [7]. Kaijie Wu et al presented a low-cost concurrent checking method for the AES encryption algorithm by using parity checking which can detect faults during normal operation and deliberately injected faults [19].

Mark Karpovsky et al presented a method of protecting a hardware implementation of the AES against a side-channel attack known as Differential Fault Analysis attack [13]. Chih-Hsu Yen et al proposed several error detection schemes for AES which are based on the (n+1, n) cyclic redundancy check over GF (2^s) [20]. Luca Berveglieri et al presented an operation-centered approach to the incorporation of fault detection into cryptographic device implementation through the use of Error Detection Codes [8]. Ramesh Karri et al presented a fault-tolerant architecture for symmetric block ciphers which is based on a hardware pipeline for encryption and decryption [12]. P. Maistri et al presented the results of a validation campaign on an AES core protected with someerror detection mechanisms [15]. Mojtaba Valinataj et al combined and reinforced the parity prediction scheme with a partially distributed TMR to achieve more reliability against multiple simultaneous errors [18]. Other efforts focus on relevant fault detection field. L. Berveglieri et al presented suggestions for providing fault detection capabilities in recent block ciphers and came to the conclusion that the detection capability of any code depends on the type of the code, the frequency of checkpoints and the level of redundancy [6]. Ramesh Karri et al presented a technique to concurrently detect errors in block ciphers as well as a new encoding strategy [9].

As a summary, the parity bit check coding technique has been introduced and widely applied to the basic operations of AES. For this technique, the parity bit needs to be generated and checked for every individual AES operation which brings in considerable time and hardware overhead. The algorithm based tolerant (ABFT) technique is a general concept for designing efficient fault tolerant schemes based on structures of the algorithms [10, 14, 16]. The ABFT makes use of the computational nature of the targeted algorithm and poses a conceptual way to better create a fault tolerant version by altering the algorithm computation so that its output contains extra information for error detection and correction. It has relatively low overhead and no additional arithmetical logic unit is required.

However, there is a limitation that the ABFT approach cannot be applied to AES directly. The limitation is that it applies to certain type of algorithm usually only involved linear and/or logical XOR operations. Due to the security nature of the AES algorithm, some non-linear operation such as s-box byte substitution operation is...
required. For this reason, the ABFT can not apply to AES directly. In this paper, we present a novel fault detection technique which is based on error detecting codes, namely a suitably designed multiple parity bit code based on SECDED. This technique proves to be very efficient and has a rather low hardware overhead.

This paper is organized as follows. Section 2 briefly reviews the AES algorithm and introduces the notations for further discussion. Section 3 presents the methods for AES error detection and correction.

2 THE AES ALGORITHM AND RELEVANT NOTATIONS

2.1: The AES algorithm

The Rijndael encryption algorithm [14] consists of three procedures, namely Encryption, Decryption (i.e., the inverse of Encryption) and Key Schedule. NIST imposed an input block size of 128 bits with the most common version using a 128-bit-long key, but longer keys are allowed. The ciphered output is 128-bit. The architecture we propose is limited to 128-bit-long keys; however, longer keys can be easily supported by modifying the Key Unit and the Control Unit.

The encryption process consists of 10 iterative rounds executed after a pre-processing key-mixing phase, where the initial key is added (modulo 2) to the initial input. The intermediate result of the encryption process is stored in a 16-byte square matrix \( S \) called state. The encryption round is a chain of four byte-based transformations, both linear and non-linear, defined over the Galois Field \( GF(2^8) \). These transformations are:

- **SubBytes**: a byte-wise non-linear substitution, it can be computed on-the-fly or implemented by means of a lookup table;
- **ShiftRows**: a byte rotation of the rows of the state \( S \) using an offset that depends on the row itself;
- **MixColumns**: a linear algebraic transformation of each column of the state, over the Galois Field;
- **AddRoundKey**: a bit-wise addition (modulo 2) of the current round key, provided by the key schedule, to the state \( S \).

Each round key is also a 16-byte square matrix. The Key Schedule updates the matrix by using some of the encryption operations: in particular, each byte of the last row of the matrix is fed as input to the SubBytes operation. This result is later used to update each row of the matrix with a cascaded XOR-tree.

2.2: The reference architecture

The architecture we selected as a reference was presented in [13]. The authors described a highly regular architecture, which mimics the AES state: each byte of the state is computed and stored in a data cell, which communicates only with the cells in the same row or column. Additionally, each cell has an extra input for the round key, not shown in the figure to make the drawing clearer. The first clock cycles are used to load the plain text into the circuit by feeding the input from the side. After the load is complete, the encryption process starts. Fig. 1 shows a block diagram of an AES circuit using a loop-architecture based on the compact implementation, which does not support error detection feature. A 128-bit input is encrypted (or decrypted) 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. 1) is switched from the original order.

![Figure 1: AES circuit](image-url)
Then, the MixColumns function block is placed at the output of the key scheduler on the right to compensate the side effect. In the next section, the proposed error detection scheme is explained in contrast with this normal architecture. All four internal transformations of one AES round work on byte elements, and are rooted in the algebra of finite fields (Galois fields, GF) [16]. The finite fields of interest for AES are the binary fields, of type GF(2^n). The integer n (n = 1) identifies the number of bits used to represent the field elements. These two operations are executed by the XOR and AND logic gates, respectively. The basic operations of AES are defined over elements of the field GF(2^8), i.e., on byte elements of n = 8 bits each. For convenience, a byte can also be represented in binary or hexadecimal in addition to its polynomial presentation.

Sub-Bytes:
All bytes are processed separately. For every byte not equal to 0=(0,0,0,0,0,0,0,0) first the inverse in GF(2^8) is determined. m(x)=x8+x4+x+1 is used as the modular polynomial for GF(2^8). The byte 0 is mapped to 0. Then a linear affine transformation is applied. Very often Sub-Bytes is implemented using 16 copies of an 8-bit×8-bit ROM.

Shift-Rows:
The rows of the state are shifted cyclically byte-wise using a different offset for each row. Row 0 is not shifted, row 1 is cyclically shifted left 1 byte, row 2 is cyclically shifted left by 2 bytes and row 3 is cyclically shifted left 3 bytes.

Mix-Column:
The elements of the columns of the state are considered as the coefficients of polynomials of maximal degree 3. The coefficients are considered as elements of GF(2^8). These polynomials are multiplied modulo the polynomial x4+1 with a fixed polynomial c(x) = (03)x3+(01)x2+(01)x+(02). The coefficients of this polynomial given in hexadecimal representation are also elements of GF(2^8). Thus for instance the coefficient in hexadecimal representation is in binary representation (00000001)2 or x+1 in polynomial representation in GF(2^8). The Mix-Column operation on a column zT=[z0, z1, z2, z3]T of the state into the column uT=[u0, u1, u2, u3]T and of the vectors zT and uT as well as the multiplication and the addition are in GF(2^8). The polynomial x=x8+x4+x+1 is used as the modular polynomial.

Add-key:
Add-key operation is a bit-wise exclusive-or of the 128-bit round key with the 128-bit state.

3. AES ERROR DETECTION AND CORRECTION
The SECDED design described here is the combinational logic for data communication between the microprocessor and memory. The data bus from the processor is 16-bit wide data, while the data written to memory is a 22-bit data word. When data is read back from the memory device, the stored parity bits are compared with a newly created set of parity bits from the read data. The result of this comparison, called the syndrome, will indicate the incorrect bit position in a single data error. This design is a model of the Hamming code developed by R. Hamming. SECDED for N bits of data requires K parity bits to be stored with the data where:

\[ N \leq 2K - 1 - K \]

If the bits are numbered in sequence, those bit positions that represent powers of two are dedicated to parity bits. Table 1 illustrates how the 16-bit data word and parity bits are stored in memory.

<table>
<thead>
<tr>
<th>Bit Position</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Number</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Data/Parity Bit</td>
<td>D15</td>
<td>D14</td>
<td>D13</td>
<td>D12</td>
<td>D11</td>
<td>D10</td>
<td>D9</td>
<td>D8</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
<td>P3</td>
<td>P2</td>
<td>P1</td>
<td>P0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The parity bits P0-P4 are created for single error detection and correction and are created as follows:

P0 = D15 XOR D13 XOR D11 XOR D10 XOR D8 XOR D6 XOR D4 XOR D3 XOR D1 XOR D0
P1 = D13 XOR D12 XOR D10 XOR D9 XOR D6 XOR D5 XOR D3 XOR D2 XOR D0
P2 = D15 XOR D14 XOR D10 XOR D9 XOR D8 XOR D7 XOR D5 XOR D4 XOR D2 XOR D1
P3 = D10 XOR D9 XOR D8 XOR D7 XOR D6 XOR D5 XOR D3 XOR D4
P4 = D15 XOR D14 XOR D13 XOR D12 XOR D11

The elements of the columns of the state are considered as the coefficients of polynomials of maximal degree 3. The coefficients are considered as elements of GF(2^8). These polynomials are multiplied modulo the polynomial x4+1 with a fixed polynomial c(x) = (03)x3+(01)x2+(01)x+(02). The coefficients of this polynomial given in hexadecimal representation are also elements of GF(2^8). Thus for instance the coefficient in hexadecimal representation is in binary representation (00000001)2 or x+1 in polynomial representation in GF(2^8). The Mix-Column operation on a column zT=[z0, z1, z2, z3]T of the state into the column uT=[u0, u1, u2, u3]T and of the vectors zT and uT as well as the multiplication and the addition are in GF(2^8). The polynomial x=x8+x4+x+1 is used as the modular polynomial.
3.2 SECDED DESIGN
The "Error Detection" block generates the error_out[1:0] flag based on the syndrome and the overall parity created from the data in memory.

Figure 2: SECDED Block Diagram

The "Generate Parity Bits" block creates the parity bits to store with the processor data (u_data[15:0]) during a write cycle. In a read cycle, this block is also responsible for creating one of the inputs in generating the syndrome; this block creates the parity bits with the data word stored in memory. The error_out flag decodes to the states shown in Table 2.

Table 2: Error Detection Flags

<table>
<thead>
<tr>
<th>error_out[1:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No error has occurred.</td>
</tr>
<tr>
<td>01</td>
<td>Single error has been detected. Syndrome holds value of erroneous bit.</td>
</tr>
<tr>
<td>10</td>
<td>Double error has been detected. Not correctable.</td>
</tr>
<tr>
<td>11</td>
<td>Parity error has occurred. Correctable.</td>
</tr>
</tbody>
</table>

The Xilinx SpartanIII XC3S400 FPGA device is used to prototype the proposed scheme. Simulation is done by Modelsim PE version. Xilinx ISE synthesizes and implements the design. Very-High-Speed Integrated Circuit Hardware Description Language (VHDL) is chosen as the description language and top-level source type in Xilinx ISE.

The SECDED architecture is partitioned modules performing distinct functions, each of which synchronously cooperates with other modules by using linked signals. Assembling these components together, a ciphertext state and an error detection signal are obtained after each encryption process, which feedback the encrypted ciphertext and error detection result.

The program control module takes charge of the procedure and sends out time-sequential commands to the AES core module. The AES core is responsible for all the sub-operations in the rolling architecture. These suboperations include add round key, substitute bytes, shift rows, mix columns, error detection computation and result state comparisons. This module acts as an intermediary between the program control module, S-box module, and key-ram module. The S-box module is used for S-box table lookup for SubByte operation. It is a 16*16 ROM and each element is an 8-bit data. The key-ram module stores the 44 word round keys for the 11 rounds. In this implementation, round keys are pre-generated before the encryption and are stored in the key RAM in advance. A 16-bit RAM bus is used to transfer the two bytes round keys to AES core module.

Table 3 gives the specification of the proposed scheme generated by the Xilinx ISE. The comparison is performed between the original AES encryption and the proposed scheme.

Table 3: Comparison between the original AES and the proposed scheme

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Original</th>
<th>SECDED</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Slices</td>
<td>943</td>
<td>1254</td>
<td>32.9%</td>
</tr>
<tr>
<td># of Slice Flip Flops</td>
<td>289</td>
<td>433</td>
<td>49.8%</td>
</tr>
<tr>
<td># of 4 Input LUTs</td>
<td>1802</td>
<td>2376</td>
<td>31.9%</td>
</tr>
<tr>
<td>Clock Period (ns)</td>
<td>17,494</td>
<td>17,760</td>
<td>1.52%</td>
</tr>
</tbody>
</table>
3.3 PROPOSED SOLUTION TO THE PROBLEM

Although normal AES hardware designs which exist today are highly reliable, chances of soft error occurrence cannot be ruled out. This no doubt forms a major hurdle in the design of microelectronic circuits and systems. Designing a microelectronic chip is a very expensive task and excessive design costs are the greatest threat to continuation of the semiconductor industry’s growth. In order to contain this threat, the increasing gap between the complexity of new systems and the productivity of system design methods must be mitigated by developing new and more efficient design. In this paper we have attempted to formulate a design procedure which can always guarantee the functional correctness of the AES hardware that is designed. In this method error detection is done by validating the output block cipher during the hardware implementations. The hardware design of this AES block with single bit error correction was accomplished using VHDL and implemented on Xilinx FPGA. The modeling process utilized in this project is the bottom-up approach. This means that the leaf components in the design hierarchy were developed first and the higher-level modules were constructed by instantiating their subcomponents and connecting them with the internal signals. All the modules in the design hierarchy were modeled in behavioral style, but the root module consisted of data flow modeling as well to implement the four major cipher transformations.

The output is soft error hardened cipher text. This has been accomplished via two steps: I) according to a built-in reliability functions library, the designer specifies the coding techniques to be incorporated into the circuit; II) then, by using a specific fault injection technique, the designer estimates the circuit reliability. Both steps are performed in VHDL high-level description language. The first step of the approach is based on the incorporation of coding techniques into the original VHDL circuit description. The coding approaches considered are in the form of: (a) Hamming code plus one parity bit per storage element (single registers) to correct single errors and to detect double errors (SEC/DED); and (b) Two dimensional parity code to be applied to the columns and lines of embedded memory arrays.

CONCLUSION

To improve system reliability, a designer may wish to provide an automatic error detection and correction circuit. One such example is the data communicated from the microprocessor to peripheral memory devices. In this design, multiple parity bits are added to the data word upon a write to memory. With multiple parity bits, both single and double data errors can be detected upon reading the word from memory and correct single data errors. The Spartan 3e provides a 2-bit error output flag for the microprocessor to handle detected double errors. The SECDED design described here is the combinational logic for data communication between the microprocessor and memory. The data bus from the processor is 16-bit wide data, while the data written to memory is a 22-bit data word. When data is read back from the memory device, the stored parity bits are compared with a newly created set of parity bits from the read data. The result of this comparison, called the syndrome, will indicate the incorrect bit position in a single data error.

ACKNOWLEDGEMENTS

Behind every achievement lies an unfathomable sea of gratitude to those who actuated it, without them it would never have into existence. To them we lay the word of gratitude imprinted within us. I wish to express my sincere thanks to our respected Principal, Dr. V. JAYAPRAKASH, Ph.D, for all the blessing and help provided during the period of project work. I wish to express my sincere thanks Dr. K. R. KASHWAN, Ph.D, Professor and Head of the Department of Electronics and Communication Engineering (PG), for the continuous help over the period of project work and timely guidance unstrained attention and constant inspiration and creative ideas over the period of project work. Finally, I take this opportunity to extend our deep appreciation to my family and friends, for all that they meant to us during the crucial times of the completion of our project.

REFERENCES


