Performance Improvement in Fault Detection Schemes for the Advanced Encryption Standard Using Composite Fields

M.GOPAL\(^1\), K.BALA\(^2\)

\(^1\)Research Scholar, Dept of ECE, SRTS, Kadapa, Andhra Pradesh, India, E-mail: Mallegopal.6@gmail.com.
\(^2\)Asst Prof, Dept of ECE, SRTS, Kadapa, Andhra Pradesh, India, E-mail: Kbalu443@gmail.com.

Abstract: The faults that accidently or maliciously occur in the hardware implementations of the Advanced Encryption Standard (AES) may cause erroneous encrypted/decrypted output. The use of appropriate fault detection schemes for the AES makes it robust to internal defects and fault attacks. In this paper, we present a lightweight concurrent fault detection scheme for the AES. In the proposed approach, the composite field S-box and inverse S-box are divided into blocks and the predicted parities of these blocks are obtained. Through exhaustive searches among all available composite fields, we have found the optimum solutions for the least overhead parity-based fault detection structures. Moreover, through our error injection simulations for one S-box (respectively inverse S-box), we show that the total error coverage of almost 100% for 16 Sboxes (respectively inverse S-boxes) can be achieved. Finally, it is shown that both the application-specific integrated circuit and field programmable gate-array implementations of the fault detection structures using the obtained optimum composite fields have better hardware and time complexities compared to their counterparts.

Keywords: AES, Composite Fields, Error Coverage, Fault Detection.

I. INTRODUCTION

The Advanced Encryption Standard (AES) has been lately accepted by NIST [2] as the symmetric key standard for encryption and decryption of blocks of data. In encryption, the AES accepts a plaintext input, which is limited to 128 bits, and a key that can be specified to be 128 (AES-128), 192 or 256 bits to generate the cipher text. In the AES-128, which is here-after referred to as the AES; the cipher text is generated after 10 rounds, where each encryption round (except for the final round) consists of four transformations. The four transformations in the AES encryption include Sub Bytes (implemented by 16 S-boxes), Shift Rows, Mix Columns, and Add Round Key. Furthermore, to obtain the original plaintext from the cipher text, the AES decryption algorithm is utilized. The decryption transformations are the reverse of the encryption ones[2]. Among the transformations in the AES, only the S-boxes in the encryption and the inverse S-boxes in the decryption are nonlinear. It is interesting to note that these transformations occupy much of the total AES encryption/decryption area [2]. Therefore, the fault detection schemes for their hardware implementations play an important role in making the standard robust to the internal and malicious faults.

There exist many schemes for detecting the faults in the hardware implementation of the AES, see for example [3]. Among them, the schemes presented in [3]–[8] are independent of the ways the AES S-box and inverse S-box are implemented in hardware. Moreover, there exist other fault detection schemes that are suitable for a specific implementation of the S-box and the inverse S-box. The approach in [9] and the one in [10] which is extended in [11] are based on using memories (ROMs) for the Sbox and the inverse S-box. Moreover, a fault tolerant scheme which is resistant to fault attacks is presented in [12]. To protect the combinational logic blocks used in the four transformations of the AES, either the parity-based scheme proposed in [11] or the duplication approach is implemented. Furthermore, to protect the memories used for storing the expanded key and the state matrix, either the Hamming or Reed–Solomon error correcting code is utilized. It is noted that our proposed fault detection approach is only applied to the composite field S-box and inverse S-box. Whereas, the scheme presented in [11] uses memories. Using ROMs may not be preferable for high performance AES implementations. Therefore, for applications requiring high performance, the S-box and the inverse S-box are implemented using logic gates in composite fields.

The schemes are suitable for the composite field implementation of the S-box and the inverse S-box. The approach is based on using the parity-based fault detection method for a specific S-box using composite field and polynomial basis for covering all the single faults. In the scheme of, the fault detection of the multiplicative inversion
of the S-box is considered for two specific composite fields. The transformation and affine matrices are excluded in this approach. Moreover, predicted parities have been used for the multiplicative inversion of a specific S-box using composite field and polynomial basis. Furthermore, the transformation matrices are also considered. Finally, in the parity-based approach, through exhaustive search among all the fault detection S-boxes utilizing five predicted parities using normal basis, the most compact one is obtained.

The contributions of this paper are as follows.

- We have presented low-cost parity-based fault detection scheme for the S-box and the inverse S-box using composite fields. In the presented approach, for increasing the error coverage, the predicted parities of the five blocks of

![Fig.1. The S-box (the inverse S-box) using composite fields and polynomial basis and their fault detection blocks.](image1)

for the transformation and affine matrices). It is interesting to note that the cost of our multi-bit parity prediction approach is lower than its counterparts which use single-bit parity. It also has higher error coverage than the approaches using single-bit parities. We have implemented both the proposed fault detection S-box and inverse S-box and other counterparts. Our both ASIC and FPGA implementation results show that compared to the approaches presented in the complexities of the proposed fault detection scheme are lower.

- Through exhaustive searches, we obtain the least area and delay overhead fault detection structures for the optimum composite fields using both polynomial basis and normal basis. While, only the S-box using normal basis has been considered.

- The proposed fault detection scheme is simulated and we show that the error coverage’s of close to 100% for 16 S-boxes (respectively inverse S-boxes) can be obtained.

- Finally, we have implemented the fault detection hardware structures of the AES using both 0.18- m CMOS technologies and on Xilinx Virtex-II Pro FPGA. It is shown that the fault detection scheme using the optimum polynomial and normal bases have lower complexities than those using other composite fields for both with and without fault detection capability.

**II DEVELOPMENT OF THE ADVANCED ENCRYPTION STANDARD**

In 1997, the National Institute of Standards and Technology (NIST) initiated a process to select a symmetric-key algorithm. In 1998, NIST announced the acceptance of fifteen candidate algorithms and requested the assistance of the cryptographic research community in analyzing the candidates. This analysis included an initial examination of the safety and efficiency characteristics for each algorithm. NIST reviewed the results of this preliminary research and selected five final candidates: MARS, RC6, Rijndael, Serpent and Two fish. One can find the comparison of these algorithms. Finally Rijndael algorithm was established among these finalists as the Advanced Encryption Standard. It is noted that before the acceptance of Rijndael algorithm, DES and its improved variant 3DES were used as symmetric key standards. DES has 16 rounds and encrypts and decrypts data in 64-bit blocks, using a 64-bit key. This can be compared to AES-128 which has 10 rounds where data’s are encrypted and decrypted in 128-bit blocks, using a 128-bit key. It is easy to find a comparison between Rijndael and DES and triple DES.

The five finalists in the AES contest mentioned above are iterated block Ciphers. It means that they specify a series of transformations (round) that is iterated a number of times on the data block to be encrypted or decrypted. Also, each finalist specifies a method for generating a series of keys
from the original user key. This method is called the key
schedule, and the generated keys are called round keys., the
very first and last cryptographic operations are some form of
mixing of round key with the data block are made for each
finalist. Such mixing prevents a challenger who does not
know the keys from even beginning to encrypt the plaintext
or decrypt the cipher text Rijndael has very low RAM and
ROM requirements and is very well suited to Restricted-
space environments where either encryption or decryption is
implemented. It is noted that Rijndael appears to be every
time a very good performer in both hardware and software
across a wide range of computing environments.

A. Encryption and Decryption of AES

The AES accepts a 128-bit plain text input. The key can
be specified to be 128 (AES-128), 192 or 256 bits. In AES-
128, the cipher text is generated after 10 rounds, where,
each round consists of four transformations except for the
final round which has three transformations. The reverse
procedure is used in decryption algorithm transforming the
cipher text to the original plain text. Fig.3 shows the
transformations and rounds in the encryption and decryption
of AES-128.

Fig.3. Encryption and decryption of AES

B. AES Rounds and Transformations

The four transformations of each round of the encryption
each transformation in every round of encryption/decryption
acts on its 128-bit input which is considered as a four by
four matrix, whose entries are eight bits called state. The
transformations in each round of encryption except for the
last round are as follows:

- **Sub Bytes:** The initial transformation in each Step is
  the bytes substitution, called Sub Bytes, which is
  implemented by 16 S-boxes. These S-boxes are non-
  Linear transformations which substitute the 128-bit
  input state with a 128-bit output state. In the S-box, each
  byte of the state (Ii in Fig.3) is substituted by a new
  byte (Bi in Fig.3). S-box will be explained clearly in the
  next section.

- **Shift Rows:** Shift Rows is the following transformation
  in which the four bytes of the rows of the input state are
  regularly shifted to the left and the first row remains
  unchanged as shown in the leftmost part of Fig.3. The
  number of left shifts for each row is equal to the
  number of rows. Denote rows as row is where, i, 0 ≤ i ≤
  3, is the row number. Then, for row 0 no shift, for row1
  one shift, for row2 two shifts and for row3 three shifts
  are required. • Mix Columns: The third transformation
  is Mix columns in which each entry in the outputs state
  are constructed by the multiplication of a column in the
  input state with a fixed polynomial over GF (2^8). The
  output states are obtained by multiplying the columns
  of the input state modulo x4 + 1 with the fixed
  polynomial of \(a(x) = (03) x3 + (01) x2 + (01) x + (02)\),
  where the coefficients are in hexadecimal form. The
  Mix columns matrix representation is shown in Fig1.
  As seen in figure, the output states are constructed by
  multiplying the entries of the input states by a fixed
  matrix whose entries are in the hexadecimal form.

- **Add Round Key:** The ultimate transformation is Add
  Round Key which XORs the input states with the key
  of that round, i.e., ki, 0 ≤ i ≤ 10. The AES key
  expansion unit in Fig.3 takes the 128-bit original key,
  k0, as input and produces a linear array of expanded
  keys, k1 to k10. Each key is added to the input by 128
  two-input XOR gates.

Among the four transformations in the encryption and
decryption of AES, only S-box for encryption and inverse
S-box for decryption are non-linear and composite
operations. Furthermore, not only is the S-box one of the
four round transformations, but it is also used in the key
expander unit which generates the keys used in the AES
rounds. Therefore, the implementations of these two
transformations affect the implementation of the whole AES
tremendously soon after in this process, the implementation
variations of the S-box and inverse S-box including the
composite field implementation.

III. S-BOX OPERATION

In cryptography, an S-Box (Substitution-box) is a basic
component of symmetric key algorithms which performs
substitution operations. In block ciphers, they are typically
used to unclear the relationship between the key and the
cipher text. In common, an S-Box takes some number of
input bits, m, and transforms them into some number of
output bits, n, where n may not be equal to m. An mxn S-
Box can be implemented as a lookup table with $2^m$ words of $n$ bits each. Fixed tables are normally used, as in the Advanced Encryption Standard (AES), but in some ciphers the tables are generated animatedly from the key (e.g. two fish encryption algorithms).

If exactly one bit error appears at the output state of the S-box and also in inverse S-box) the presented fault detection scheme is able to detect it and the error coverage is about 100%. This is because in this case, the error indication flag of the corresponding block alarms the fault. However, due to the technological constraints, single stuck-at fault may not be applicable for a mugger to gain more information. Thus, multiple bits will actually be flipped and hence multiple stuck-at errors are also considered in this paper covering both natural faults and fault attacks. For the calculation of the fault coverage for the multiple errors, $P_i$ defines as the possibility of error detection in block in Figs. 4 and 5. Then, the probability of not detecting the errors in block is $(1-P_i)$. For arbitrarily distributed errors in the S-box (respectively inverse S-box), this probability for each block is Independent of those of other blocks. Therefore, one can derive the equation for the error coverage of the randomly distributed errors as

$$EC\% = 100 \times \left(1 - \prod_{i=1}^{S} (1-P_i)\right)$$

(1)

Where $S$ is the set of the block numbers where the faults are injected. For randomly distributed errors, the error coverage for each block is $P_i \approx \frac{1}{2}$. The results of the error simulations are presented in Table I. As shown in the table, using five parity bits of the five blocks, the error coverage for random faults reaches 97%. Error.

IV. ASIC AND FPGA IMPLEMENTATIONS AND COMPARISONS

In this section, we compare the areas and the delays of the presented scheme with those of the previously reported ones in both application-specific integrated circuit (ASIC) and field programmable gate array (FPGA) implementations. We have implemented the S-boxes using memories and the ones presented (the hardware optimization, and which use polynomial basis representation in composite fields. We have also implemented the fault detection schemes proposed (both united and parity-based), which are based on the ROM-based implementation of the S-box. The results of the implementations for both original and fault detection scheme (FDS) in terms of delay and area have been tabulated in Tables I and II. As seen in these tables, the original structures are not divided into blocks and full optimization of the original entire architecture as a single block is performed in both ASIC and FPGA. This allows us to find the actual over-head of the presented fault detection scheme as compared to the original structures which are not divided into five blocks. We have used 0.18- m CMOS technology for the ASIC implementations. These architectures have been coded in VHDL as the design entry to the Synopsys Design Analyzer. The results are tabulated in Table I. Moreover, for the FPGA implementations in Table II, the Xilinx Virtex-II Pro FPGA (xc2vp2-7) is utilized in the Xilinx ISE version 9.1i. Furthermore, the synthesis is performed using the XST.
As seen in Tables I and II, we have implemented the fault detection schemes presented in [3] and [9] based on using redundant units for the S-box (united S-box). Furthermore, the fault detection scheme proposed in [11] is implemented. This scheme uses 512 9 memory cells to generate the predicted parity bit and the 8-bit output of the S-box [11]. One can obtain from Tables I and II that for both of these schemes, the area overhead is more than 100%. As mentioned in the introduction, the approach in [12] utilizes the scheme in [11] for protecting the combinational logic elements, whose implementation results are also shown in Tables I and II. Additionally, for certain AES implementations containing storage elements, one can use the error correcting code-based approach presented in [12] in addition to the proposed scheme in this paper to make a more reliable AES implementation. Moreover, the parity-based scheme in [9] which only realizes the multiplicative inversion using memories is implemented. As seen in these tables, we have also implemented the schemes. It is noted that the scheme for the multiplicative inversion and does not present the parity predictions for the transformation matrices. Moreover, we have applied the presented fault detection scheme to the S-boxes. As seen in bold faces in Tables I and II, with the error coverage of close to 100%, the presented low-complexity fault detection S-boxes are the most compact ones among the other S-boxes. The optimum S-box and inverse S-box using normal basis have the least hard-ware complexity with the fault detection scheme. Moreover, as seen in the tables, the optimum structures using composite fields and polynomial basis (PB1andPB2) have the least post place and route timing overhead among other schemes. It is noted that using sub-pipelining for the presented fault detection scheme in this paper, one can reach much faster hardware implementations of the composite field fault detection structures.

### TABLE I

ASIC Implementations of the Fault Detection Schemes for the S-Box (SB) and The Inverse S-Box Using 0.18- M CMOS Technology

<table>
<thead>
<tr>
<th>Operation</th>
<th>Architecture</th>
<th>Area (μm²) / Delay (ns)</th>
<th>Area (μm²) / Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-box</td>
<td>ROM (SB)</td>
<td>169 × 10³ / 5.4</td>
<td>344 × 10³ / 7.7</td>
</tr>
<tr>
<td></td>
<td>ROM (SB)</td>
<td>102 × 10³ / 5.4</td>
<td>576 × 10³ / 7.0</td>
</tr>
<tr>
<td></td>
<td>ROM (SB)</td>
<td>180 × 10³ / 5.8</td>
<td>191 × 10³ / 5.0</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[13] (mult. inv.)</td>
<td>5915 / 12.9</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[14]</td>
<td>5375 / 12.0</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[12]</td>
<td>5915 / 12.0</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[13]</td>
<td>5942 / 12.3</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[14]</td>
<td>5947 / 12.3</td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>[15]</td>
<td>5179 / 12.9</td>
</tr>
<tr>
<td></td>
<td>PB1</td>
<td>This work</td>
<td>3231 / 10.6</td>
</tr>
<tr>
<td></td>
<td>PB2</td>
<td>This work</td>
<td>3230 / 9.3</td>
</tr>
<tr>
<td>Inverse S-box</td>
<td>NB</td>
<td>This work</td>
<td>5107 / 13.3</td>
</tr>
<tr>
<td></td>
<td>PB1</td>
<td>This work</td>
<td>8228 / 10.9</td>
</tr>
<tr>
<td></td>
<td>PB2</td>
<td>This work</td>
<td>5374 / 9.4</td>
</tr>
</tbody>
</table>

### TABLE II

Xilinx Virtex-II Pro FPGA Implementations (Xc2vp2-7) of the Fault Detection Schemes for the S-Box (SB) and the Inverse S-Box

<table>
<thead>
<tr>
<th>Operation</th>
<th>Architecture</th>
<th>Slice / Delay (ns)</th>
<th>Slice / Delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S-box</td>
<td>ROM (SB)</td>
<td>69 / 3.235 / 420 / 5.396</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROM (SB)</td>
<td>69 / 3.235 / 420 / 5.396</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROM (SB)</td>
<td>88 / 3.734 / 300 / 6.370</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB (20)</td>
<td>35 / 9.375 / 44 / 9.869</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NB</td>
<td>31 / 7.286 / 40 / 7.465</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB1</td>
<td>31 / 7.286 / 40 / 7.465</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PB2</td>
<td>31 / 7.286 / 40 / 7.465</td>
<td></td>
</tr>
</tbody>
</table>

| Inverse S-box | NB | This work | 31 / 7.734 / 38 / 7.964 |
|               | PB1 | This work | 32 / 6.992 / 42 / 7.423 |
|               | PB2 | This work | 32 / 6.992 / 42 / 7.423 |

### TABLE III

ASIC Implementations of the Fault Detection Schemes of the AES Encryption Using 0.18- M MOS Technology

<table>
<thead>
<tr>
<th>AES encryption</th>
<th>Optimum S-box</th>
<th>Area (μm²)</th>
<th>S-boxes</th>
<th>All</th>
<th>Freq. (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original without fault detection</td>
<td>PB1</td>
<td>692781 (80%)</td>
<td>859471</td>
<td>79.4</td>
<td></td>
</tr>
<tr>
<td>PB2</td>
<td>704440 (80%)</td>
<td>871180</td>
<td>91.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NB</td>
<td>680590 (80%)</td>
<td>845126</td>
<td>77.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We have also implemented the AES encryption using the presented optimum S-boxes excluding the key expansion. Then, we have added the proposed scheme for Sub Bytes and Shift Rows considering that Shift Rows is the rewire from the output of Sub Bytes. The results are presented in Tables III and IV. As one can notice, the S-boxes occupy more than three fourths of the AES encryption. As shown in these tables, the most compact AES encryption with and without the fault detection scheme is for normal basis. Furthermore, the frequency degradation is negligible. Moreover, the original AES encryption for and the ones with fault detection for PB1andPB2 have the highest working frequencies. In addition, as seen in the tables, we have applied the presented scheme to Sub Bytes and Shift Rows and used the scheme in [11] for the other transformations.
V. CONCLUSION

In this paper, we have presented a high performance parity based concurrent fault detection scheme for the AES using the Sbox and the inverse S-box in composite fields. Using exhaustive searches, we have found the least complexity S-boxes and inverse S-boxes as well as their fault detection circuits. Our error simulation results show that very high error coverage for the presented scheme is obtained. Moreover, a number of fault detection schemes from the literature have been implemented on ASIC and FPGA and compared with the ones presented here. Our implementations show that the optimum S-boxes and the inverse S-boxes using normal basis are more compact than the ones using polynomial basis. However, the ones using polynomial basis result in the fastest implementations. We have also implemented the AES encryption using the proposed fault detection scheme. The results of the ASIC and FPGA mapping show that the costs of the presented scheme are reasonable with acceptable post place and route delays.

VI. REFERENCES


International Journal of Scientific Engineering and Technology Research
Volume.03, IssueNo.02, February-2014, Pages:0350-0355