A Highly Parallel Area Efficient S-Box Architecture for AES Byte-Substitution

Mostafa Abd-El-Barr and Altaf Al-Farhan

Abstract—The performance of the S-Box represents an important factor in the overall performance of the AES cryptography systems. It affects the speed, area, and the power consumption of the AES. In attempts to improve the performance of the S-box byte substitution a number of techniques were presented in the literature. In this paper, we classify the S-box byte substitution optimization techniques as those based on hardware, software, and combined hardware/software. We then move on to propose a new highly parallel and area-efficient S-box architecture for AES byte substitution. We also conduct a performance analysis and comparison of the proposed architecture with those achieved by existing techniques. The comparison shows that the proposed architecture outperforms the existing techniques in terms of speed and area.

Index Terms—Cryptography, advanced encryption standard (AES), s-box byte substitution, parallel s-box architecture, efficient s-box.

I. INTRODUCTION

The National Institute of Standards and Technology (NIST) organized an open competition in 1997 to find a replacement for DES cryptosystem. The Rijndael cryptosystem, submitted by Joan Daemen and Vincent Rijmen was one such system [1]. The NIST adopted a slightly modified version of Rijndael as the new security standard. This standard is now known as the Advanced Encryption Standard (AES). It is a symmetric-key block cipher algorithm used to encrypt/decrypt data worldwide.

According to the AES data to be encrypted is divided into equally sized blocks each is called a state. The algorithm performs a series of mathematical operations on each state based on the Substitution-Permutation Network principle to produce the cipher text. The algorithm starts with an initial step to add the round key to the state. After that, the state enters the main loop which includes four repeated operations: sub bytes, shift rows, mix columns, and add round key. This is followed by a final iteration that excludes mix columns. There have been a number of improvements to enhance efficiency of the original AES in terms of area, delay, and power consumption [2]-[4]. Fig. 1 provides a simple illustration of the AES encryption process.

Among the four loop operations, the substitution of bytes is performed in what is known as the substitution box (S-Box). The S-Box performs a non-linear transformation on data by replacing each individual byte by a different byte. The main purpose of the byte substitution is to bring confusion to the data to be encrypted using the AES [5]. The replacement bytes can be obtained on the fly by determining the multiplicative inverse of a given state in finite field GF (256) followed by affine transformation in GF 2). Alternatively, the replacement bytes can be pre-calculated and stored in a look-up table (LUT) in the S-Box. It should be noted that the replacement bytes for encryption are different than the ones used for decryption.

Byte substitution is considered one of the most complex loop operations. Hence, a number of research efforts were devoted to the optimization of the byte substitution both in time, hardware complexity, and power consumption [6]-[15]. The research work reported in this paper concentrates on proposing a new method for designing a highly parallel area-efficient S-Box architecture for the AES cryptosystem.

The paper is organized as follows. In Section II, we provide some background material. In Section III, we provide a brief summary of the existing S-Box realization techniques. In Section IV, we preview the related work reported in the literature. In Section V, we introduce the proposed S-Box architecture and algorithm. In Section VI, we illustrate the CMOS implementation of the proposed architecture. In Section VII, we provide estimation of the delay and area needed for the proposed architecture. In Section VIII, we provide a comparison with existing techniques. Section IX provides a number of concluding remarks.

II. EXISTING S-BOX REALIZATION TECHNIQUES

Fig. 2 illustrates our classification of the S-Box byte substitution realization techniques.

A. Hardware Techniques

Hardware techniques are the ones that compute substitution...
values rather than reading them from a memory. The representation of the finite field elements has an impact on the hardware in terms of gates count and chip area. Polynomial and normal bases are the two popular representations. The most compact implementation known is the one proposed by Canright [9] in normal bases, which used sub-field arithmetic to compute AES S-Box. Canright has examined all possible choices of representations in each sub-field including purely polynomial, purely normal basis, and a combination of both. The total number of isomorphisms was found to be 432 different choices. The case that uses polynomial basis for all sub-fields is the one proposed by Satoh et al., [7]. The optimal representation, which has the minimum gate count, was obtained using an exhaustive tree-search algorithm. In each sub-field both inverse and multiplication operations are required. These mathematical operations were discussed in details for the most compact case. Further optimization was done by calculating a merged S-Box for both encryption and decryption. The merged S-Box is 20% more compact than separate computations of S-Box and its inverse. Canright's implementation is best suited for limited space applications, e.g. mobile applications. The technique can also be parallelized and pipelined for high throughput applications. The main drawback of this implementation is its relatively long critical path and high power consumption [10]. Alternatively, Nikova et al., [11] proposed an implementation using normal bases that differs in the level of decomposition. Both normal bases implementations [9] and [11] are compact, however the choice of which to use depends on the application and hardware technology. The polynomial representations by Satoh et al., [7], Rijmen [12], and Wolkerstorfer et al., [13] were less efficient compared to normal bases representations in terms of chip area.

The implementation that resulted in least power consumption according to [10] is the one proposed by Bertoni et al., [14]. The proposed solution uses a synthesis methodology composed of three main blocks: a multi-level decoder followed by a permutation block, which does the S-Box computation, and finally an encoder.

S-Box values. Techniques that are included under such approach vary from the totally serial (slow) to the totally parallel (fast) ones. In the totally serial approach each byte of the State is substituted by the corresponding S-Box byte one at a time in a serial manner. Consider, for example, the 128 bit AES case. In this case each of the 16 State bytes is replaced by the corresponding S-Box byte. Assume that the area required for the LUT is \( A \), and the time needed to substitute one byte is \( T \), then the total time needed to substitute all sixteen State bytes is \( 16T \). The other extreme is the totally parallel approach. In this case there are 16 S-Boxes each consists of 16 bytes such that all of the 16 bytes of the State are substituted by their respective corresponding S-Box bytes simultaneously. The total time needed to substitute all sixteen State bytes is \( T \), while the area required is \( 16A \).

The two extreme cases: the most restricted serial and the most flexible parallel ones are illustrated in Fig. 3. In between, there exist a number of possible intermediate cases. These intermediate cases exhibit tradeoffs between area and throughput.

### C. Hardware/Software Techniques

In this case use is made of pipelining (hardware technique) in building the S-box employing small substitution tables constructed using LUTs (software technique) [15]. The basic idea of the approach is that the original large truth-table of say 8-variable function is broken down into a set of smaller size multiplexer-switched truth-table of say 4-variable functions using the Shannon expansion. The smaller tables were then mapped into 4-LUT of Xilinx FPGA. The approach shows significant improvement in the overall throughput.

### III. RELATED WORK

The traditional basic lookup table implementations (hw-lut) are relatively fast and can achieve better performance with some modifications [10]. One way to reduce power consumption is to divide the 256 bytes S-Box into smaller tables with the aim being to reduce the switching activities. The use of smaller LUTs of different sizes ranging from 16 to 128 bytes was examined in [10]. However, only the result for the 16 bytes size was reported (sub16-lut). The solution proposed in [8] not only uses small LUTs, but also reduces execution time at the expense of doubling the chip area required.

Unlike the serial approach reported in [10], the approach in [8] explored the use of 32 parallel small S-Boxes of size 16 bytes each. The first group of LUT (16 LUTs) use the left-most 4 bits of the \( state \) byte to distinguish among 16 tables (Table 0 to Table F) and the right-most 4 bits of the \( state \) byte as the address to obtain the value in a given table. This is done in parallel for all the bytes of a state. If the four left most bits of two or more bytes were identical then a conflict occurs. Conflict is resolved by using the second group of 16 S-Box tables indexed by the right-most 4 bits of the State byte (tables 0' to Table F'). In that case, the left-most 4 bits of the byte are used as the address to obtain the value in a given table. These steps are repeated until all bytes are substituted. Based on this approach the average expected speed up is 8 times faster than the classical serial implementation.
The main drawback of the approach in [8] is the possibility of conflict in accessing a given table. In this case, the author suggested using a busy bit with each table such that if the busy bit is ON, then it means that the table is currently in use and cannot be accessed in the current cycle. This will lead to a slowdown in the rate at which the S-Box substitution is performed. In the next Section, we propose an S-Box architecture which overcomes the conflict problem in addition to achieving higher throughput.

For each byte in a state do:
1) Use the left-most two bits to select a group of four,
2) Use the next left-most bits to select a row within a group,
3) Use the next left-most bits to select a column—The selected column and the row identified in steps 2 above intersect at a point that identifies a specific 2×2 table,
4) Use the two right-most bits to lookup the table and obtain substitution value.

V. PROPOSED S-BOX CMOS IMPLEMENTATION

In order to choose one group out of four, a 2-to-4 decoder is used. Two more 2-to-4 decoders are required to choose the row and column within the selected group. We choose to use NAND decoders to comply with CMOS technology. The decoder circuit is shown in Fig. 5(a). Once decoding is done, the LUT to be used is known. The two right most bits of the byte processed are used as an index to the 2×2 LUT. These two bits are connected to the select lines of a 4-to-1 multiplexer having the table data as inputs and the S-Box substitution value as the output. All the 4-to-1 multiplexers implemented are constructed by three 2-to-1 multiplexers for simplicity (Refer to Fig. 5(b)). The following is an illustration of three different ways to implement 2-to-1 multiplexers:

A. Gate-Level Implementation (NAND-NAND) CMOS Implementation

Each CMOS NAND gate consists of four transistors. Three NAND gates and an inverter can be used as shown in Fig. 5(c) to perform the operation of a 2-to-1 MUX. For our 2×2 LUT we need a 4-to-1 MUX which can be constructed using three 2-to-1 multiplexers for simplicity (Refer to Fig. 5(b)).

B. Gate-Level Implementation (NAND-NAND) CMOS Implementation

Pass transistor logic can be used as shown in Fig. 5(d) to implement a 2-to-1 multiplexer. The inverter added at the output to retain logic level. The 4-to-1 multiplexer needed for S-Box LUT is constructed using three 2-to-1 multiplexers.

C. Transmission Gates Implementation

Transmission gates are simply switches which can act as a two-to-one multiplexer as shown in Fig. 5(e). In this case the number of transistors required is less than the former implementations. Again, 4-to-1 multiplexers are constructed out of 2-to-1 ones.

VI. DELAY AND AREA ESTIMATION

All results reported in this paper are based on using 0.35 µm CMOS technology of AMS Corp. [16]. Since the decoding part for our three proposed solutions is the same, the associated delay and area are consequently identical. Remember that four bytes of a state are processed in parallel. This implies that 72 NAND gates are required, as every byte needs three 2-to-4 decoders. Each 2-to-4 decoders requires six NAND gates assuming that inverters in implemented using NAND gates. Area is measured using gate equivalents (GE), where one gate equivalent corresponds to one NAND2 gate.
According to the CMOS library used one NAND2 gate has 0.1 ns delay. Given that decoding of the row and column are done in parallel, the total decoding time is 1.6 ns for one AES state.

(a) The first four bytes of a given state are processed.

(b) Selecting a group.

(c) Identifying the row, the column, and the LUT containing the substitution value

Fig. 4. An illustrative example of the proposed S-Box architecture.

For the first proposed implementation, three 2-to-1 multiplexers are needed for each byte. This means that we need twelve 2-to-1 multiplexers to process four bytes of a state, that is 48 GEs given that each 2-to-1 multiplexer is 4 GE. The delay for multiplexing is four times the delay of one 4-to-1 multiplexer, that is 2.4 ns.

For the second proposed implementation, two pass transistors which construct a 2-to-1 multiplexer can be estimated to consume an area equivalent to ½ of a NAND2 area. We need an inverter for the selector which has a size of 1 GE. In addition to two cascaded inverters at the output to retain the logic value. The two inverters consume the area of 2 GE. The sum is 3.5 GE for each 2-to-1 multiplexer yielding a total of 42 GE (12×3.5). The delay of one 2-to-1 multiplexer is the delay of the transistors plus the delay of the two inverters, that is 0.25 ns. Therefore, the total delay can be estimated as 2 ns (8×0.25).

In the third proposed implementation, each transmission gate consumes an area equivalent to ½ of a NAND2. The inverter for the selector consumes 1 GE. The sum is 2 GE per 2-to-1 multiplexer. This gives a total area of 42 GE (12×2). The critical path delay for the data of the 2-to-1 multiplexer is estimated as the delay of one two input NAND gate yielding a total of 0.8 ns (8×0.1). The delay and area estimations of the three possible implementations are summarized in Table I.

<table>
<thead>
<tr>
<th>Implementation</th>
<th>First Proposed Solution (NAND)</th>
<th>Second Proposed Solution (Pass Transistor)</th>
<th>Third Proposed Solution (Transmission Gate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoders Delay (ns)</td>
<td>1.6</td>
<td>1.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Multiplexers Delay (ns)</td>
<td>2.4</td>
<td>2</td>
<td>0.8</td>
</tr>
<tr>
<td>Total Delay (ns)</td>
<td>4</td>
<td>3.6</td>
<td>2.4</td>
</tr>
<tr>
<td>Decoders Area (GE)</td>
<td>72</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>Multiplexers Area (GE)</td>
<td>48</td>
<td>42</td>
<td>24</td>
</tr>
<tr>
<td>Total Area (GE)</td>
<td>120</td>
<td>114</td>
<td>96 has in positive-influence.</td>
</tr>
</tbody>
</table>
VII. COMPARISON

Based on the minimum values of area and delay obtained from [10], a comparison is made between various AES S-Box techniques (Refer to Table II).

<table>
<thead>
<tr>
<th>Implementation</th>
<th>Critical Path Delay (ns)</th>
<th>Area (GE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satoh [4]</td>
<td>9</td>
<td>360</td>
</tr>
<tr>
<td>Canright [3]</td>
<td>8</td>
<td>281.3</td>
</tr>
<tr>
<td>hw_lut [5]</td>
<td>3</td>
<td>1203.8</td>
</tr>
<tr>
<td>sub16_lut [5]</td>
<td>4</td>
<td>1912.5</td>
</tr>
<tr>
<td>Bertoni [12]</td>
<td>3</td>
<td>1608.8</td>
</tr>
<tr>
<td>Our Proposed 1</td>
<td>4</td>
<td>120</td>
</tr>
<tr>
<td>Our Proposed 2</td>
<td>3.6</td>
<td>114</td>
</tr>
<tr>
<td>Our Proposed 3</td>
<td>2.4</td>
<td>96</td>
</tr>
</tbody>
</table>

As can be seen from the table, when it comes to critical path delay, the third proposed implementation (using Transmission Gates) outperforms Bertoni's [12] and hw_lut [5] approaches by 20%. And when it comes to gate equivalent (GE) the third proposed implementation requires about one third the gate equivalent (GE) used by Canright's [3] compact implementation.

VIII. CONCLUSION

A highly parallel area efficient S-Box Architecture for AES Byte-Substitution is introduced in this paper. The proposed is modular, compact, and is efficient in terms of time and area. The design uses parallelism and can be pipelined to maximize throughput and reduce delay. The authors are currently exploring those objectives as future work.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support received from Kuwait University in the form of the funded research project WI 04/10.

REFERENCES