Abd-El-Barr,. Int J Comput Softw Eng 2019, 4: 146
https://doi.org/10.15344/2456-4451/2019/146

International Journal of
Computer & Software Engineering

Review Article Open Access

Performance Assessment of an Efficient Search and Realization Technique of the S-Box in the AES Cryptosystem

Mostafa Abd-El-Barr

Department of Information Science (ISC), College of Computing Sciences and Engineering (CCSE), Kuwait University, Safat 13060, Kuwait

Abstract

The speed and area performance of the byte-substitution (S-Box) impose a correlated impact on the speed and area of the Advanced Encryption System (AES) cryptosystem. In this paper, we present an efficient realization technique of the S-Box in the AES cryptosystem. We start by classifying the S-Box byte-substitution design techniques. We then provide a brief coverage of the hardware, software, and hybrid S-Box realization techniques. We then present an efficient S-Box realization technique that replaces the search needed in the 16 ×16 S-Box by four simpler searches two of them can be conducted in parallel and in a way that reduces the final search into searching within a set consisting of only four 2×2 cells which in turn leads to a faster search strategy. We also provide a simple hardware to carry out the search. A performance comparison among five realization techniques is conducted. The comparison is based in terms of the area (A) and the delay (T). It is shown that the technique due to Bertoni is the fastest followed by the technique due to Abd-El-Barr. In terms of area, it was found out that the technique due to Abd-El-Barr consumes the least area followed by the technique due to Canright. The technique due to Abd-El-Barr achieves the best normalized $O(AT^2)$ where A is the area and T is the critical path delay.

Publication History:

Received: March 23, 2019
Accepted: May 23, 2019
Published: May 25, 2019

Keywords:

Cryptography, AES Cryptosystem, byte-substitution, S-Box, Performance comparison

Introduction

The Advanced Encryption Standard (AES) algorithm defines the underlining cryptosystem in state-of-the-art security systems in vast majority enterprises all over the world. The AES is a *symmetric key block cipher* cryptosystem in which the encryption and the decryption keys are the same. For surveys on AES, please refer to [1-3]. Researchers made numerous performance enhancements to the AES in terms of area, delay, and power consumption, see for example [4-6]. According to the AES data to be encrypted is divided into equally sized 4 ×4 blocks each is called a *state* as shown in Figure 1.

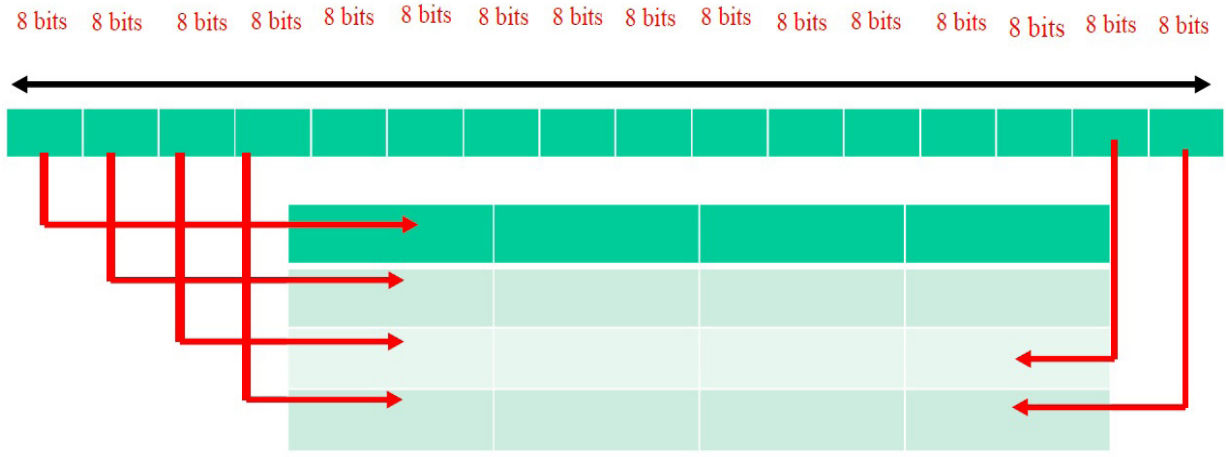


Figure 1: AES data 4×4 state.

The AES algorithm performs a series *(four)* of mathematical operations *(steps)* on each data state based on the *Substitution-Permutation* (also called the *Confusion-Diffusion*) principle in eight *rounds* to produce the *cipher text*. The AES algorithm starts with an initial step in which it adds the round key to the data state. The state then goes through a loop of four repeated operations: byte-substitution (S-Box) during which every byte is replaced by another one, using the Rijndael S-Box, shift rows (SR) during which every row in the 4x4 array (except the first one) is shifted cyclically a specific number of shift left, mix columns (MC) which is a linear transformation on the columns of the state, and add round key (ARK) during which each byte of the state is combined with a round key using a bitwise XOR. This is followed by a final iteration to produce the cipher text. Similar to encryption, the decryption of a cipher text starts with add round key (ARK) operation. However, the decryption operations are ordered as follows: inverse shift rows (ISR), inverse sub bytes (ISB), add round key (ARK), and inverse mix columns (IMC). The plain (decrypted) text is obtained after a final iteration that excludes inverse mix column operation. An overall illustration of the process is shown in Figure 2 [2]. It should be noted that Round key is different for each round and

*Corresponding Author: Prof. Mostafa Abd-El-Barr, Department of Information Science (ISC), College of Computing Sciences and Engineering (CCSE), Kuwait University, Safat 13060, Kuwait; E-mail: mostafa.abdelbarr@gmail.com

Citation: Abd-El-Barr M (2019) Performance Assessment of an Efficient Search and Realization Technique of the S-Box in the AES Cryptosystem. Int J Comput Softw Eng 4: 146. doi: https://doi.org/10.15344/2456-4451/2019/146

is derived from the Rijndael key schedule. It is also important to notice that Byte substitution (S-Box) is one of the complex operations in AES.

In Figure 3 we present our taxonomy for the S-Box different realization techniques.

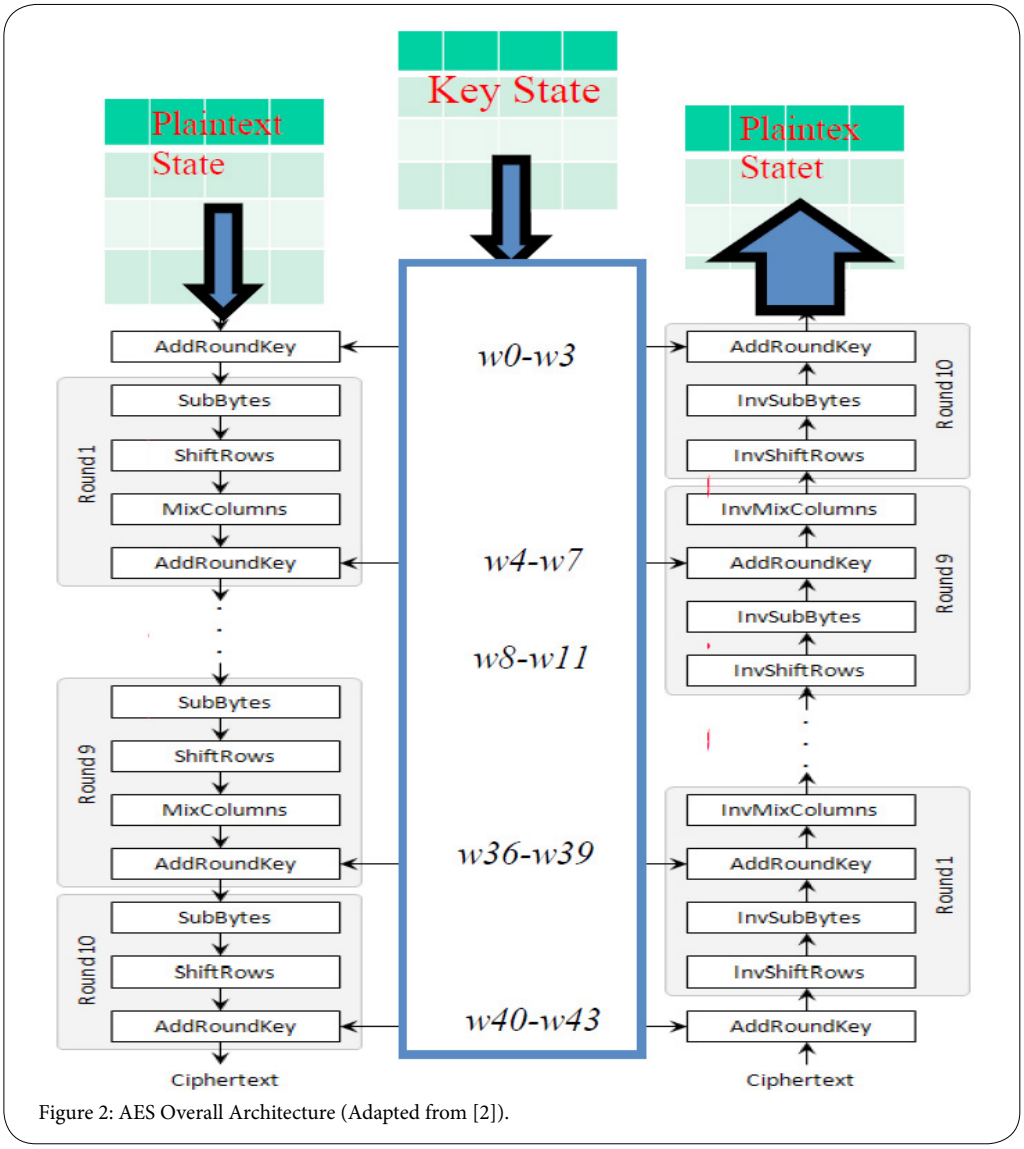


Figure 2: AES Overall Architecture (Adapted from [2]).

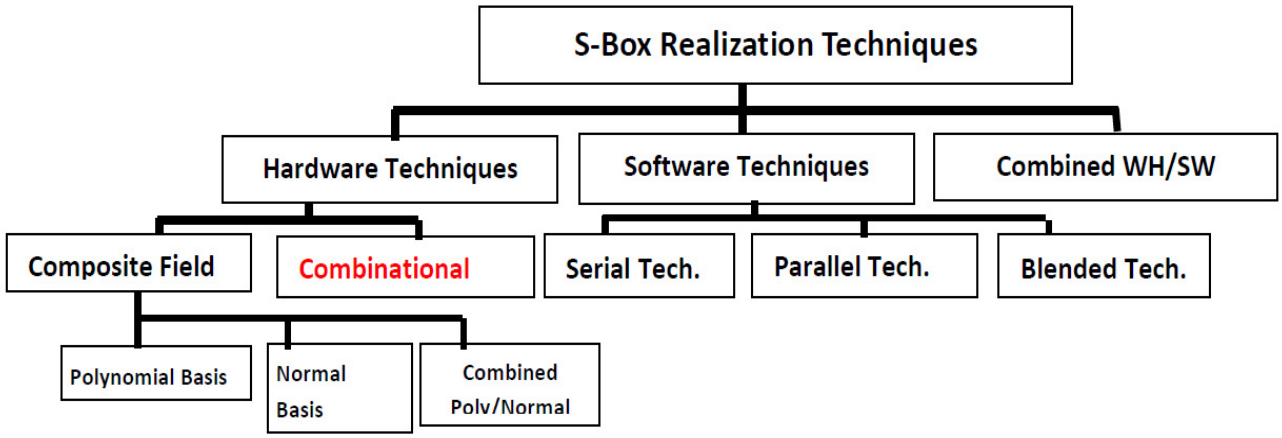


Figure 3: A taxonomy for S-Box realization techniques.

In this paper we cover a number of techniques used for realization of the S-Box. In Sections 2 to 4, we present respectively the hardware, software, and the hybrid S-Box realizations techniques. In Section 5, we present a technique that was introduced earlier by the author [10]. Section 6 presents performance comparison of the different techniques in terms of speed, area, and $O(AT^2)$ where A is the area and T is the critical path delay. We then present some concluding remarks.

Hardware S-Box Realization Techniques

The S-Box is a nonlinear transformation made by a substitution operation on each of the state independently. The Substitution Table (S-Box) is constructed using two transformations:

1. Multiplicative inverse

$$X = (X')^{-1} = \begin{cases} (X')^{256} & if\ X' \neq 0 \\ 0 & if\ X' = 0 \end{cases}$$

2. Affine (over GF(2)) transformation defined by

$$Y = L * X + '63' = \begin{bmatrix} 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \\ 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \end{bmatrix} \begin{bmatrix} x_7 \\ x_6 \\ x_5 \\ x_4 \\ x_3 \\ x_2 \\ x_1 \\ x_0 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Which translates to $Y=(x^4+x^3+x^2+x+1)*X+(x^6+x^5+x+1)\ mod\ (x^8+1)$

The multiplicative inverse is complex to perform in GF(2^8). *Composite field* (using sub-fields) arithmetic is usually used to simplify the computation, examples include the followings:

1. Degree-4: GF((2^4)^2), OR
2. Degree-2: GF(((2^2)^2)^2)←GF((2^2)^2)←GF(2^2). This leads to simplified hardware needed for performing the inversion and multiplication operations required for S-Box computation.

The resulting S-Box is shown in Figure 4.

The S-Box can be treated as a multi-input multi-output truth-table that need to be minimized. In this case, the S-Box realization is considered as 8-input, 8-output functions. Each function takes the form $Z_i = f(x_0, x_1, x_2, x_3, y_0, y_1, y_2, y_3)$ where the x_i identifies a given row in the S-Box, and the y_j identifies a given column in the S-Box. See Figure 5 (a).

If we extract the least significant bits of all S-Box bytes we end up with table Z0. The same idea applies to each of the remaining seven bits. The resulting eight slices of the S-Box is shown in Figure 6.

Software S-Box Realization Techniques

In the software realization of the S-Box, the substitution values are pre-computed and stored in what is called the "Look Up table (LUT)". See Figure 4. The substitution made through the S-Box is done such that the first right most 4 bits of the data are used to select a row (out of the 16 S-Box rows) and the last 4 bits are used to select a column (out of the 16 S-Box columns). The intersection of the row and the column identifies a cell whose content is the substitution byte. For example if

y / x	0	1	2	3	4	5	6	7	8	9	a	b	c	D	E	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	Ab	76
1	ca	82	c9	7d	fa	59	47	f0	Ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	Bc	b6	da	21	10	Ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	Ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 4: The S-Box.

the original data byte is {b9}≡{1011 1001} then the intersection of row 101 and column 1001 gives the substitution value, {56} in this case, see Figure 4. There are basically two main S-Box software realization Techniques: Series and Parallel See [4] and [5]. See Figure 7.

In the series technique, extraction of the S-Box byte is done one byte at a time while in the parallel scheme all 16 bytes are obtained simultaneously. As can be seen the series scheme is slow but requires the least amount of resources while the parallel scheme is the fastest but requires availability of 16 S-Boxes.

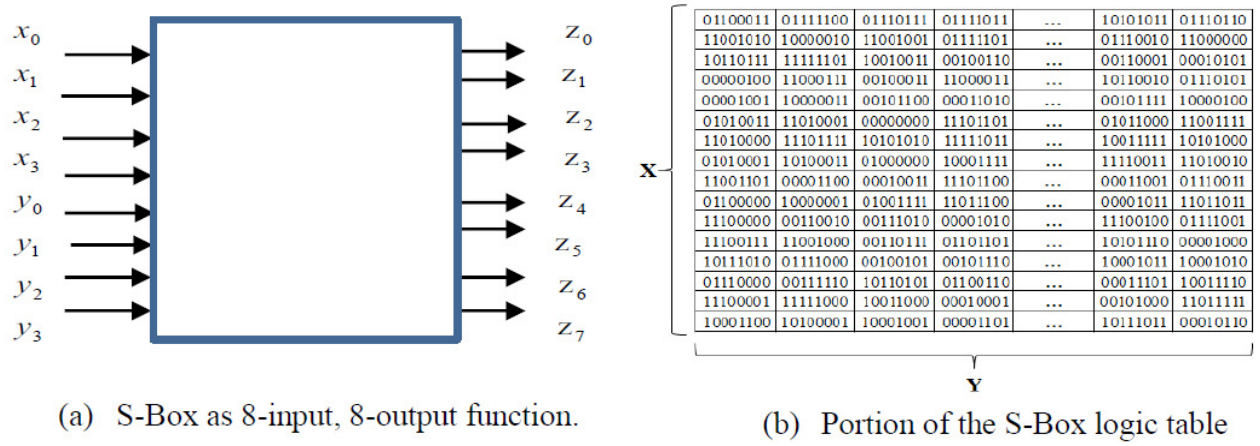


Figure 5: The S-Box.

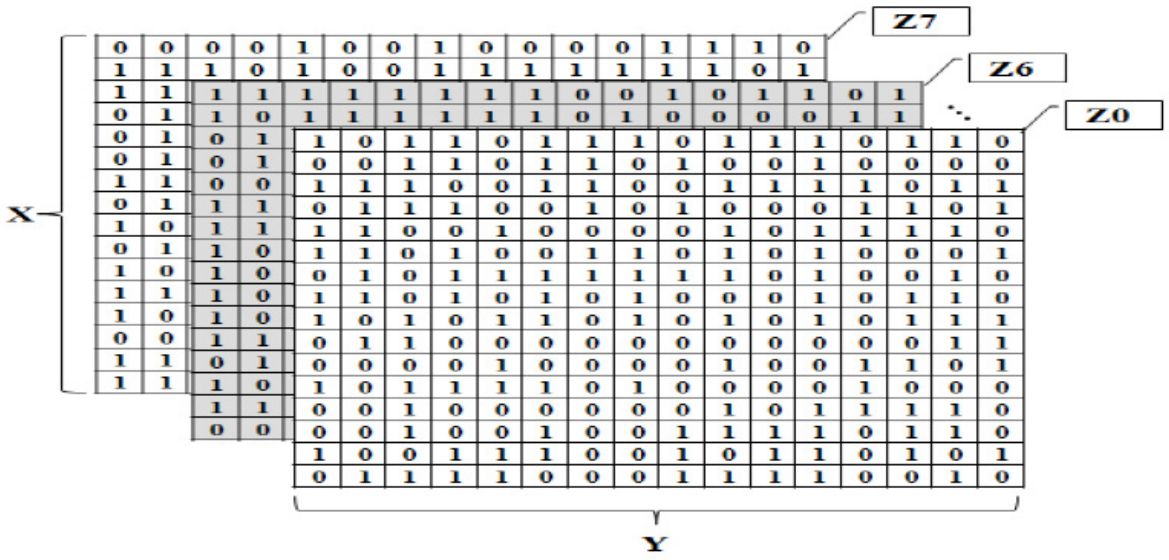


Figure 6: The S-Box sliced into eight boxes having single bit cells.

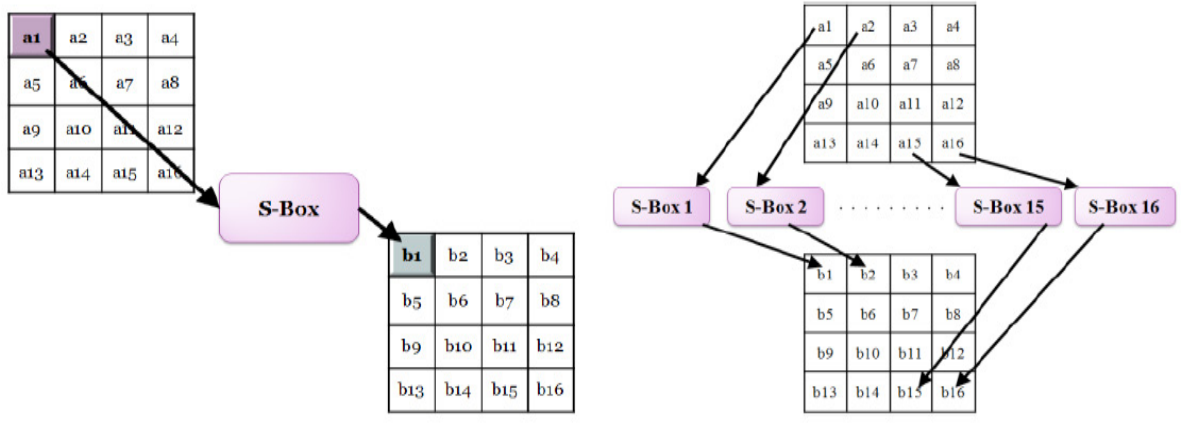


Figure 7: Two main S-Box software realization techniques.

Hybrid S-Box Realization Techniques

These techniques include LUT software implementations combined with hardware techniques. Pipelining (hardware technique) is used in building the S-Box by employing small substitution tables constructed through LUTs (software technique). The basic idea is that the original large truth-table (the S-Box in this case) of 8-variable functions is broken down into a set of smaller size multiplexer-switched truth-table of n-variable functions using the Shannon expression. The smaller tables are mapped into n-LUT of Xilinx FPGA. An example is shown in Figure 8 [9].

There three stages in this example. The first stage implements Shannon's decomposition using sixteen 4-LUTs, each having 64-bit string as an input. The results S_i, i {1,2,...,16} are latched in the registers of the slices. The second stage uses eight 4-LUTs; each performs the selection between two Sum of Products (SOPs), using combinations of the variables X_5 & X_6. The results P_j, j {1,2,...,8} are latched in the internal registers connected to the corresponding 4-LUTs. The third stage has inputs generated from the previous stage besides the two variables X_7 & X_8 which are employed for multiplexing. The results P_k, k {1,2,..., 4} are latched in the internal registers and connected to the corresponding 4-variable OR-function. The whole process can be expressed mathematically as follows:

$$F(X_1,X_2,X_3,X_4,X_5,X_6,X_7,X_8,) = Q_1 \vee Q_2 \vee Q_3 \vee Q \vee Q_4$$

$$= X_7(X_8P_1 \vee \bar{X}_8P_2) \vee \bar{X}_7(X_8P_3 \vee \bar{X}_8P_4) \vee X_7(X_8P_5 \vee \bar{X}_8P_6) \vee \bar{X}_7(X_8P_7 \vee \bar{X}_8P_8)$$

$$= X_7\begin{pmatrix} X_8X_5(X_6S_1 \vee \bar{X}_6S_2) \vee \\ \bar{X}_8\bar{X}_5(X_6S_3 \vee \bar{X}_6S_4) \end{pmatrix} \vee \bar{X}_7\begin{pmatrix} X_8X_5(X_6S_5 \vee \bar{X}_6S_6) \vee \\ \bar{X}_8\bar{X}_5(X_6S_7 \vee \bar{X}_6S_8) \end{pmatrix}$$

$$X_7\begin{pmatrix} X_8X_5(X_6S_9 \vee \bar{X}_6S_{10}) \vee \\ \bar{X}_8\bar{X}_5(X_6S_{11} \vee \bar{X}_6S_{12}) \end{pmatrix} \bar{X}_7\begin{pmatrix} X_8X_5(X_6S_{13} \vee \bar{X}_6S_{14}) \vee \\ \bar{X}_8\bar{X}_5(X_6S_{15} \vee \bar{X}_6S_{16}) \end{pmatrix}$$

A S-Box Realization Technique using 2×2 Cells

According to this S-Box realization technique byte substitution is performed by using a number of 2×2 tables that are *organized in groups*. Each group has 16 2×2 cells organized in a bigger table of *four rows* and *four columns*. The size of each group is 64 bytes, which is one fourth of the regular S-Box size. The small tables are selected based on row and column values. Each byte needs exactly four groups to cover all values of the original S-Box. If we consider the use of 4 groups, then, 4 bytes can be processed simultaneously [10].

The steps of the new search algorithm can be summarized as follows:

Start the substitution process from the left-most side of the input byte to be substituted and do the followings:

1. Use the first two left-most bits of the input byte to select a group of sixteen 2 × 2 cells (one slice)
2. Use the next two bits of the input byte to select four 2 ×2 cells within the selected group (one row).
3. Use the next two bits of the input byte to select four 2 × 2 cells within the selected group (one column).
4. The intersection of the selected row and the selected column produces a set consisting of four 2 × 2 cells.
5. Use the final two bits of the input byte to select the output (substitute) byte from among the four bytes.

End of the substitution process.

Example: Suppose that it is required to find the byte substitution for the hexadecimal byte {b9}. Figure 9 illustrates the four steps to perform. Notice that the substitute byte for {69} is {56} the same as obtained in Figure 4.

Figure 10 show the overall mapping technique using three 2-4 decoders and one 4-1 multiplexer for the above example. As can be seen, there are four steps. During the first step the two left-most bits are used to select one of the four groups (in this case it is Group #2 (binary 10)) is selected using 2-4 decoder. During the second step the next two bits from the left are used to select one of the four rows (in this case it is row #3 (binary 11)) is selected using 2-4 decoder. During the third step the next two bits from the left are used to select one of the four columns (in this case column #2 (binary 10)) is selected using 2-4 decoder. During the fourth step one of the four cells at the intersection of row #3 and column #2 is selected using the two most bits (in this case byte # 1 (binary 01)) is selected using the 4-1 MUX.

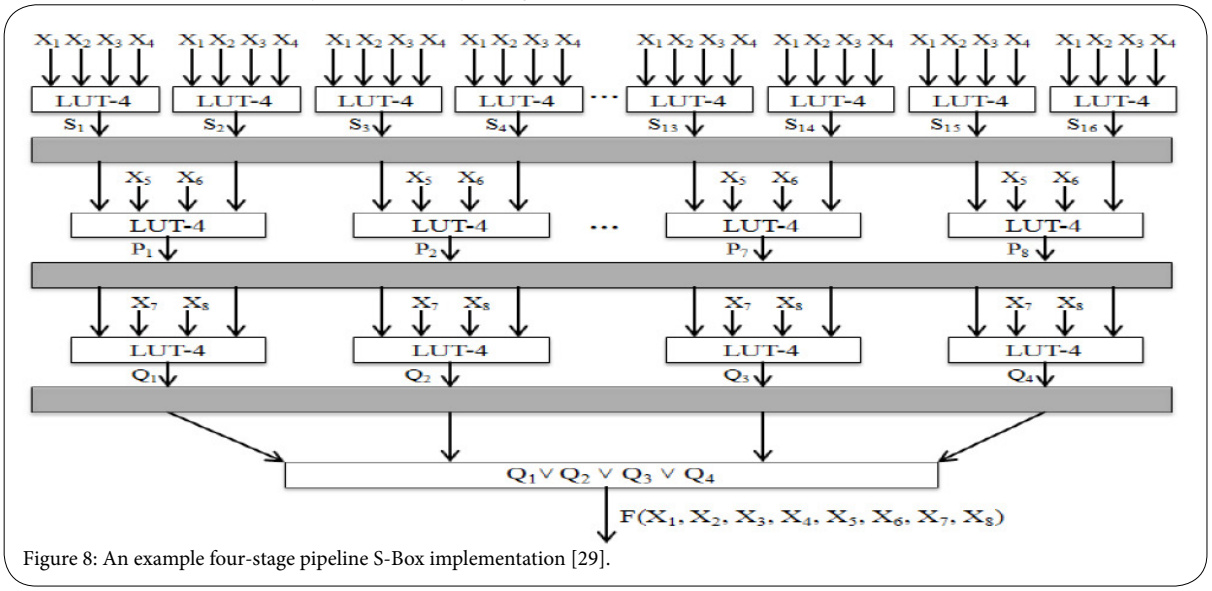


Figure 8: An example four-stage pipeline S-Box implementation [29].

As can be seen the selected byte is {56} which is the same as the one done in the example above.

Possible gate-level realizations of the 2-4decoders and the 4-1 MUX are shown in Figure 11.

The novelty of the work presented in this paper stems from the fact that it replaces the search in the 16 × 16 S-Box by a simpler four searches two of them are conducted in parallel such that the search is limited to a set consisting of four 2 × 2 cells which in turn leads to a faster search strategy. We have also provided a simple combinational hardware to carry out the search.

Performance Comparison

In this section we provide performance comparison among the different techniques used in the design and realization of the S-Box. In this performance comparison we use the minimum values for the area and delay published in [7] and we also use the CMOS (Complemented Metal Oxide Semiconductor) process characteristics published in [11]. Table 1 illustrates the comparison in terms of the area (measured in gate equivalent GE) and critical path delay (measured in ns).

The values presented in the table shows that there is a trade-off between area and the critical path delay. In particular, the following can be concluded:

1. The technique due to Bertoni [15] achieves the shortest critical path delay followed by the 2×2 Cells [10], while the Satoh technique [12] achieves the longest path delay.
2. The 2×2 cells technique achieves the smallest area followed by the technique due to Canright [14] while the technique due to Bertoni achieves the largest area.
3. The normalized AT^2 shows that the 2×2 cells achieves the best measure followed by the technique due to Wolkerstorfer [13].

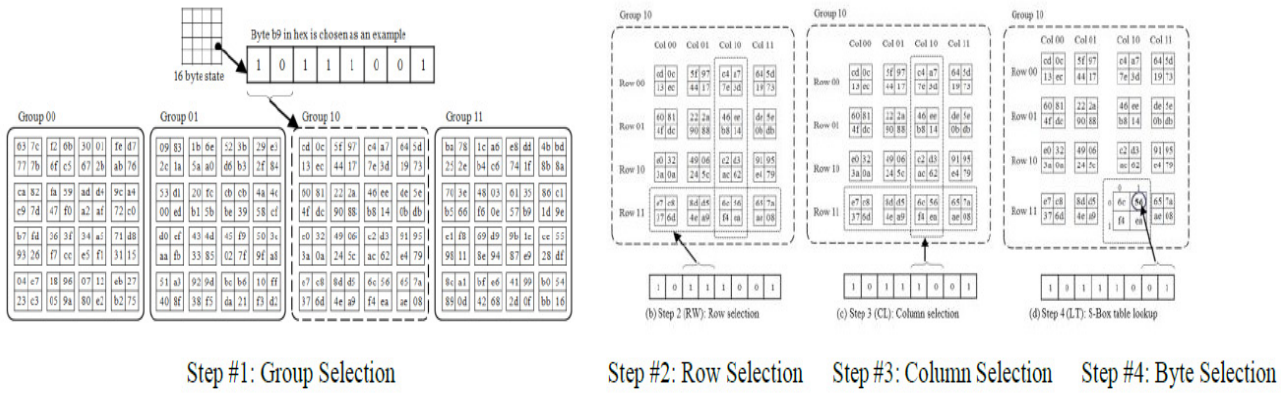


Figure 9: Illustration for byte substitution example.

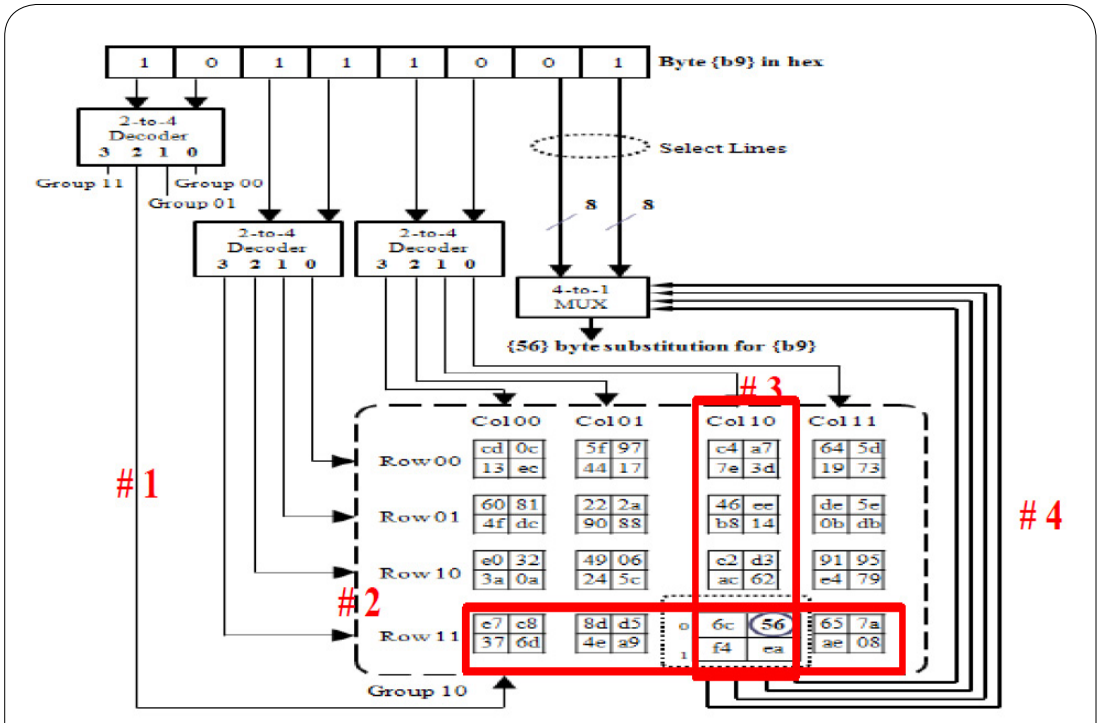
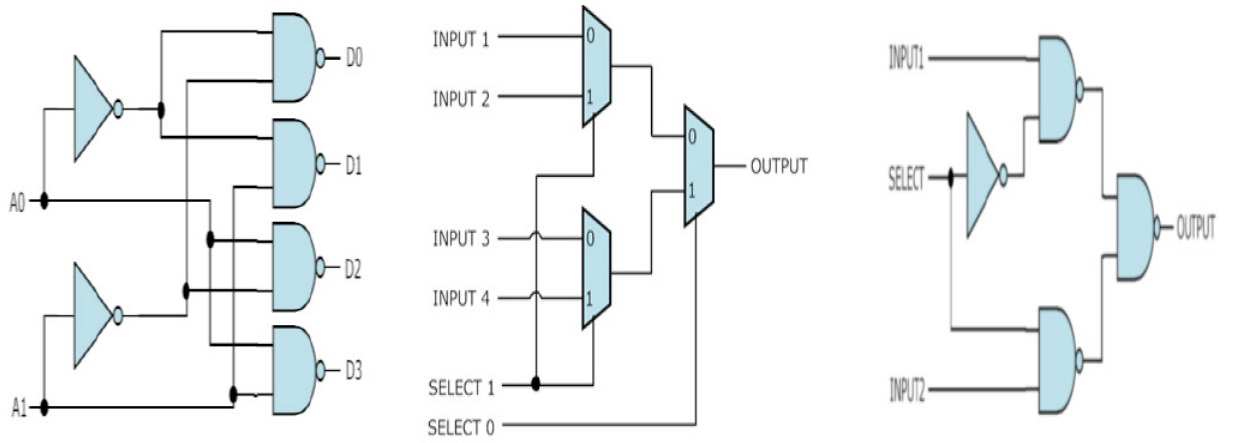


Figure 10: The substitution mapping process.



(a) 2-4 Decoder gate-level realization (b) 4-1 MUX realization using 2-1 MUXs (c) Gate-level realization of 2-1 MUX

Figure 11: Gate level realizations of the 2-4 decoder and 4-1 MUX.

Technique	Critical path delay (ns),T		AREA (GE), a		AT^2	
	Actual	Normalized	Actual	Normalized	Actual	Normalized
Satoh [12]	9	3.00	360.0	3.00	27	15.3
Wolkerstorfer [13]	8	2.66	382.5	3.19	22.57	12.8
Canright [14]	8	2.66	281.3	2.34	23.63	13.4
Bertoni [15]	3	1.00	1608.8	13.4	1608.8	909.9
2×2 Cells [10]	4	1.33	120.0	1.00	1.77(1)	1.0

Table 1: Performance comparison among the S-Box realization techniques.

Concluding Remarks

In this paper, we have presented a taxonomy of the AES S-Box realization techniques. They include hardware, software, and the combined Hardware/software techniques. A brief coverage of each technique and its sub-classes is then provided. The presentation showed that the hardware-based techniques are classified into composite and combinational techniques. The paper focuses on the combinational hardware sub-class and a comparison among five related techniques is conducted. The comparison is based on the performance of techniques in terms of speed, area and the $O(AT^2)$ where A is the area and T is the critical path delay. It was found that the technique due to Bertoni [15] has the shortest critical path delay while the technique due to Abd-El-Barr [10] achieves the smallest area and also has the least $O(AT^2)$.

Competing Interests

The author declare that there is no competing interests regarding the publication of this article.

References

1. Rao S, Mahto D, Khan DA (2017) A Survey on Advanced Encryption Standard. IJSR 6: 710-723.
2. Nechvatal J, Barker E, Bassham L, Burr W, Dworkin M, et al. (2001) Report on the Development of the Advanced Encryption Standard (AES). J Res Natl Inst Stand Technol 106: 511-577.
3. Padate R, Patel A (2014) Encryption and decryption of text using AES algorithm. International Journal of Emerging Technology and Advanced Engineering 4: 833-859.
4. Samiee H, Atani RE, Amindavar H (2011) A novel area-throughput optimized architecture for the AES algorithm. International Conference on Electronic Devices, Systems and Applications (ICEDSA).
5. Rahman T, Pan S, Zhang Q (2010) Design of a High Throughput 128-bit AES (Rijndael Block Cipher). International Multi Conference of Engineers and Computer Scientists.
6. Hodjat A, Verbauwhede I (2006) Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors. IEEE Transactions on Computers 55: 366-372.
7. Tillich S, Feldhofer M, Großschädl J (2006) Area, delay, and power characteristics of standard-cell implementations of the AES S-box. Embedded Computer Systems: Architectures, Modeling, and Simulation, Springer Verlag 4017: 457-466.
8. McLoone M, McCanny JV (2001) High performance single-chip FPGA Rijndael algorithm implementations, Cryptographic Hardware and Embedded Systems (CHES 2001). Springer Verlag 2162: 65-76.
9. Pipeline AES S-box Implementation Starting with Substitution Table, LC Engineers Inc., USA.
10. Abd-El-Barr M, Al-Farhan A (2014) A Highly Parallel Area Efficient S-Box Architecture for AES Byte-Substitution. International Journal of Engineering and Technology 6: 346-350.
11. Specialty Process 0.35µm CMOS Application Notes.
12. Satoh A, Morioka S, Takano T, Munetoh S (2001) A compact rijndael hardware architecture with S-box optimization. Advances in Cryptology - ASIACRYPT 2248: 239-254.
13. Wolkerstorfer J, Oswald E, Lamberger M (2002) An ASIC implementation of the AES SBoxes. Topics in Cryptology-CT-RSA 2002: 67-78.
14. Canright D (2005) A very compact S-Box for AES. Cryptographic Hardware and Embedded Systems-CHES. Springer Verlag.
15. Bertoni G, Macchetti M, Negri L (2004) Power-efficient ASIC Synthesis of Cryptographic Sboxes. ACM Creat Lakes symposium on VLSI.