Efficient Implementation of AES S-box in LUT-6 FPGAs

Anane Nadjia
CDTA (Centre de Développement des Technologies Avancées) Algiers, Algeria

Anane Mohamed
ESI (Ecole nationale Supérieure d'Informatique)
Algiers, Algeria

***Abstract*—Advanced Encryption Standard (AES) is a symmetric cryptographic algorithm used for protecting data. Designing efficient hardware architecture for AES with small hardware resource usage is a challenge. AES uses different data transformations and the most expensive one, in terms of computational resources, is the SubBytes transformation which is carried out by a Look-Up-Table (LUT) named the S-box.**

In this paper, an efficient implementation of the S-box on LUTs-6 of an FPGA circuit of Virtex-5 is presented. This has reduced both occupied area and execution time, where the reading time of an S-Box is that of one slice.

***Keywords*— AES, SubBytes, S-Box, LUTs.**

I. INTRODUCTION

In this modern world, communication between each and every one is very important. Internet, satellite made the communication much easier than past decades. Even communication between each get easier, the security transmission of the message is very important today. Transmitting and receiving the secured data becomes tougher in nowadays.

Cryptography plays an important role in securing data. It enables to transmit it across insecure networks so that unauthorized persons cannot read it. The urgency of secure exchange of digital data resulted in different encryption algorithms: symmetric with a secret key and asymmetric with public and private keys. Symmetric algorithms are much faster than asymmetric ones as they use small keys (128 bits) such as AES (Advanced Encryption Standard) [1].

AES is a symmetric algorithm operating on a 128-bit block of data considered as a 4×4 matrix of bytes, named the "state". It operates in 10, 12, and 14 rounds depending on the size of the used key: 128, 192 or 256 bits respectively [2]. This key is used to generate sub-keys for rounds.

Figure 1, shows the AES 128-bit, where ten rounds are required. Each round contains four separate blocks: SubBytes, ShiftRows, MixColumns and AddRoundKey.

1. The AddRoundKey transformation involves a bitwise XOR operation between the state array and the resulting Round Key, which is derived from the initial key in the Key Expansion function.
2. The SubBytes transformation is a highly non-linear byte substitution, where each byte in the state array is replaced with another one from a lookup table called the S-Box.
3. The ShiftRows transformation is done by cyclically shifting the rows in the array with different offsets.
4. The MixColumns transformation is a column mixing operation, where bytes in the new column are a function of the 4 bytes of a column in the state array.

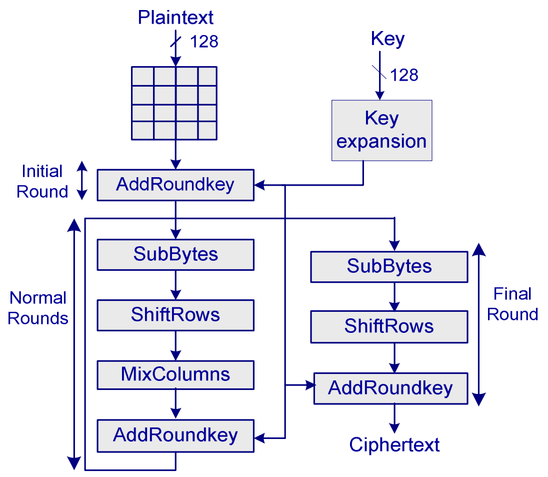


Fig. 1.AES transformations

The SubBytes transformation is the most time consuming operation of the AES round where for a parallel architecture of an AES design, several tables are required. For example, in a high-speed design of 128-bit AES [3], 16 S-box modules are needed for one round and a huge amount of hardware is required for the rest of the rounds. A hardware implementation of an S-Box on Virtex-5 LUTs permits to reduce both hardware complexity and execution time hence accelerates the AES encryption.

This paper is focusing on an efficient implementation of the AES SubBytes transformation based on reading the S-box table. This table is implemented on FPGA circuit of Virtex-5 using the 6-input Look-Up-Tables (LUT-6) combined with multiplexors [4] in order to reduce both occupied area and execution time, where the reading time of an S-Box is that of one slice.

The rest of this paper is organized as follows. Section 2 reviews the AES SubBytes transformation. Section 3 describes the AES parallel architecture. Section 4 details the S-Box architecture based on LUT-6. Section 5 presents the implementation results of this architecture on FPGA circuit of Virtex-5 and finally section 6 gives a conclusion.

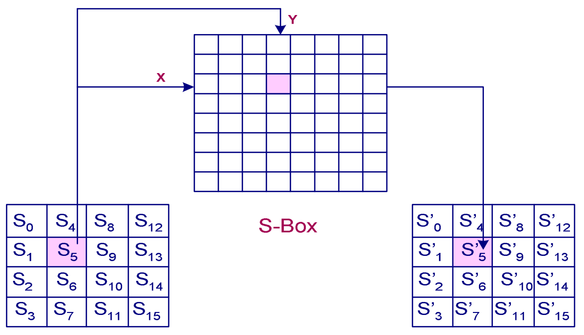
II. THE SUBBYTE TRANSFORMATION

AES defines a 16×16 matrix of byte values; called an S-Box, shown in Table 1.

TABLE 1. S-BOX TABLE

		Y															
		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
X	**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

The SubByte transformation is a nonlinear substitution that operates on individual bytes using a substitution table (S-Box), which contains a permutation of all 256 possible 8-bit values. Each individual byte of state is mapped into a new byte in the following way [5]:
The leftmost 4 bits MSB of the byte are used as a row value and the rightmost 4 bits LSB are used as column value. These row and column values serve as indexes into the S-box to select a unique 8-bit output value as shown in figure 2.



Y
x
S-Box
S_0 S_4 S_8 S_{12}
S_1 S_5 S_9 S_{13}
S_2 S_6 S_{10} S_{14}
S_3 S_7 S_{11} S_{15}
S'_0 S'_4 S'_8 S'_{12}
S'_1 S'_5 S'_9 S'_{13}
S'_2 S'_6 S'_{10} S'_{14}
S'_3 S'_7 S'_{11} S'_{15}

Fig. 2. S-Box Transformation

For example, the hexadecimal value {53} references row 5 and column 3 of the S-box, where their intersection contains the value {ED}. Accordingly, the value {53} is mapped into the value {ED} as shown in the S-Box of Table 1.

III. THE AES PARALLEL ARCHITECTURE

SubByte is composed of sixteen identical S-boxes for AES parallel architecture. This last allows the 128 bits of the state to enter in parallel by blocks of 32 bits or words w_i where i=1,..4, as shown in figure 3. Four S-boxes are required for the four words, and each word needs four other S-boxes to execute the SubByte transformation for the four bytes of the word. Each of these S-boxes can be implemented independently using a 256×8 bits look-up table, equivalent to 2048 bits or 2 kilobits.

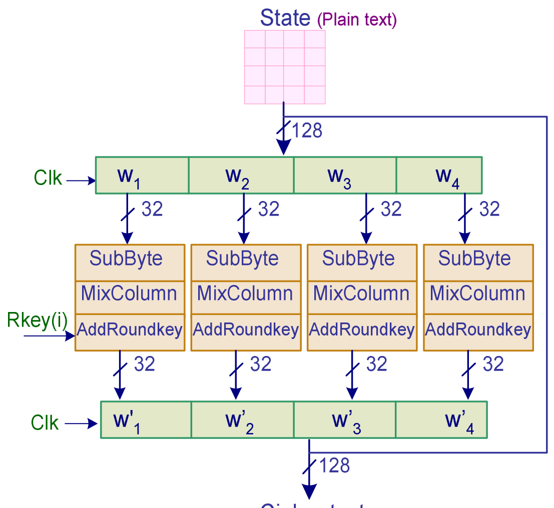


Fig. 3. Parallel architecture of an AES Round

The AES S-boxes can be implemented using SRAM or LUTs. Using an SRAM block of 18 Kbits, available on FPGA, for a memory of 2 Kbits presents a great waste of resources. However, the LUTs approach is more economical, simple to design and offers a shorter critical Path [6].
LUTs are one of several element types embedded in FPGAs. They are responsible for implementing the logic of a design. When combining LUTs with multiplexers, an FPGA can implement combinatorial and more complex logic functions. FPGAs use thousands of LUTs that can implement logic functions. A LUT is a memory affecting some values to other ones by avoiding computations of their corresponding functions which take long times.
The LUT approach is especially interesting on Virtex-5 devices [7].The main logic resources of those FPGAs are the CLB (Configurable Logic Bloc).Those CLBs are divided into two slices which are themselves composed by four logic-function generators implemented as 6-input LUTs. Each LUT possess 6 independent inputs and 2 independent outputs (O_5 and O_6) as shown on figure 4.

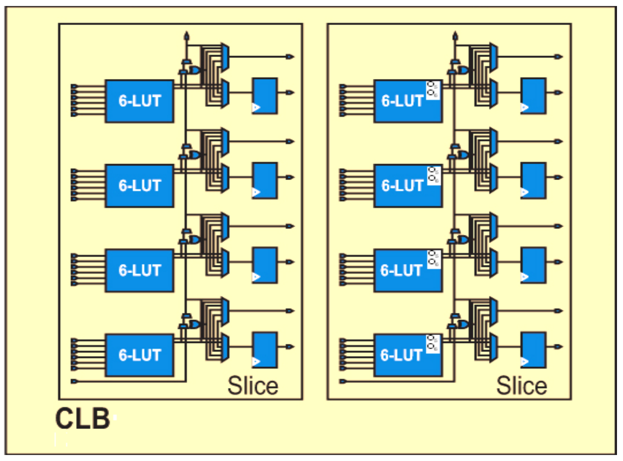


Fig. 4. CLB of Virtex-5

IV. S-BOX ARCHITECTURE BY LUT-6

The S-box has 8 bits input and 8 bits output and its design uses generation functions of eight variables [8]:

$$y_1 = f(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7),$$

represented by a LUT-8.

This last is composed by four LUT-6 combined with multiplexors which are available on Virtex-5 FPGA.

The LUT-6 is a memory of 26 addressed by 6 inputs corresponding to a truth table of 6×1.

Each bit of the S-Box memory will be considered as a generation function of 8 address inputs of the S-Box, which is the byte to be substituted. Then, the implementation of this function requires 4 LUTs-6 or one slice of Virtex-5 FPGA as represented on figure 5.

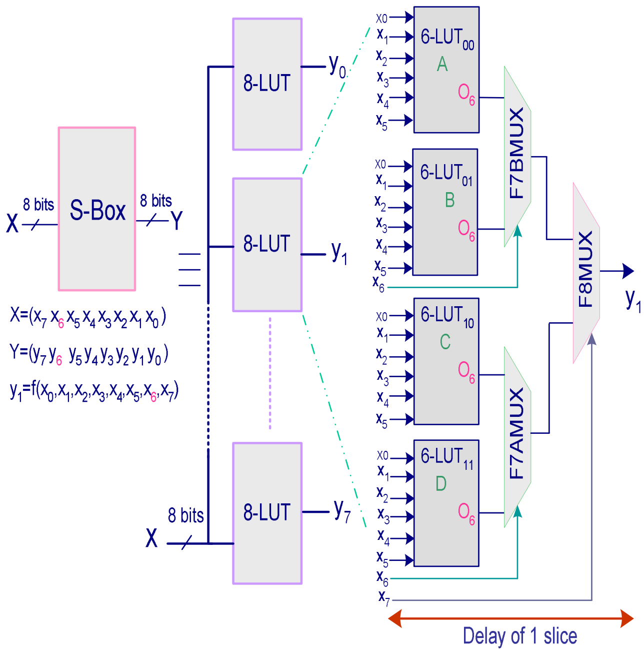


Fig. 5. AES S-Box with LUT-6

The four LUT-6 (A, B, C and D) are addressed by six inputs $(x_0, x_1, x_2, x_3, x_4, x_5)$. The input x_6 is used via 2:1 multiplexers: the (F7BMUX) to distinguish between LUT-A and LUT-B or the (F7AMUX) to distinguish between LUT-C and LUT-D. While x_7 is used via another 2:1 multiplexer, the (F8MUX), to select either the first block composed by LUTs-6 (A, B) or the second one composed by LUTs-6 (C, D).

A 256×8 bits S-box fits in 8×4 LUTs or 32 LUTs, representing 8 slices per S-Box

Each output of the 8 S-box columns represents an 8-bit to1-bit Boolean function and is stored as a 256-bit value denoted by $(a_{256}, a_{255}, \ldots, a_1)$.

The AES S-Box columns have to be split-up into 4 LUTs each, because one LUT of the FPGA can only store 64 bits [8]. An AES S-box column has eight input bits that we denote by $(x_7, x_6, \ldots, x_0)$. The straightforward example of how to distribute the 256 bit values $(a_{256}, a_{255}, \ldots, a_1)$ to 4 LUTs is illustrated in the following:

LUTA = $(a_{256}, \ldots, a_{193})$ sel. by $(x_6, x_7) = (0,0)$
LUTB = $(a_{192}, \ldots, a_{129})$ sel. by $(x_6, x_7) = (0,1)$
LUTC = $(a_{128}, \ldots, a_{65})$ sel. by $(x_6, x_7) = (1,0)$
LUTD = $(a_{64}, \ldots, a_1)$ sel. by $(x_6, x_7) = (1,1)$

V. IMPLEMENTATION RESULTS

The AES S-box architecture has been implemented on an FPGA circuit of Virtex-5, the x5vlx50t ff1136-3. The ISE 12.2 (Integrated Software Environment) Design suit and VHDL were used for the design of the architecture. The functional verification of the architecture has been realized with the ISim (ISE Simulator).

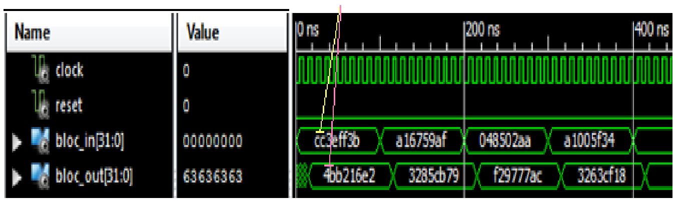
Figure 6 shows the schematic of the SubByte transformation applied to four 32-bit blocks at the inputs of the architecture given by:

"CC3EFF3B", "A16759AF", "048502AA", "A1005F34". These values have given at the outputs the following 32-bit blocks: "4BB216E2", "3285CB79", "F29777AC" and "3263CF18", where each byte of the input is replaced by its equivalent given in the S-Box table.

Bloc in
32
4 S-Boxes
Bloc out
32

Fig. 6. Schematic of SubByte transformation for a 32-bit word

The simulation results of these values are given in figure 7.



```
Name            Value      0 ns                       200 ns                     400 ns
clock           0
reset           0
bloc_in[31:0]   00000000   cc3eff3b   a16759af   048502aa   a1005f34
bloc_out[31:0]  63636363   4bb216e2   3285cb79   f29777ac   3263cf18
```

Fig. 7. Simulation Results of AES SubBytes

From this figure, we can verify that the value "CC" has been substituted by the value "4B" taken from Table 1.

The proposed implementation is based on optimization at the lowest level (slice level) of the transformation SubBytes. This optimization has resulted in high frequency operation combined with a small occupied area. The implementation results of the architecture have shown that a period of 2.27 ns is sufficient to read the S-BOX, which corresponds to a frequency of 440 MHz as presented in table 2.

TABLE 2. IMPLEMENTATION RESULTS

Frequency (MHz)	440
Period (ns)	2.27
Number of Luts-6	32
Occupied slices	8

VI. Conclusion

In this paper, we have implemented the architecture of the S-Box required for the AES SubBytes transformation on LUTs-6 of Virtex-5 FPGA. The implementation of one bit of the S-Box has required 4 LUTs-6 or one slice of the FPGA, whereas for an 8-bit S-Box, 8×4 LUTs are needed representing 8 slices. This solution has the significant advantage of both reducing the area requirements in LUTs and performing the S-box computation in a single clock cycle, thus reducing the latency.

References

[1] C. Paar, J. Pelzl, "Understanding Cryptography", Springer-Verlag 2010.

[2] N.F. Standard. Announcing the Advanced Encryption Standard, Federal Information Processing Standards Publication 197, 2001.

[3] N.Anane and M.Anane, "AES IP for for Hybrid Cryptosystem RSA-AES", the 12th International Multi-Conference on Signals Systems and Devices), "SSD'2015", Mahdia (Tunisia), 16-19 March 2015.

[4] P. Bulens, F. Standaert, J. Quisquater, P. Pellegrin, and G. Rouvroy, "Implementation of the aes-128 on virtex-5 fpgas," Progress in Cryptology–AFRICACRYPT 2008, pp. 16–26, 2008.

[5] Amruta Page, P. V. Sriniwas Shastry, "AES-128 Key Expansion with LUT and OTF S-Box", International Journal of Computer Technology and Electronics Engineering (IJCTEE),Volume 4, Issue 3, June 2014.

[6] K. Rahimunnisa, M. Priya Zach, S. Suresh Kumar, J.Jayakumar, "Architectural Optimization of AES Transformations and Key expansion", International Journal on Cryptography and Information Security (IJCIS),Vol.2, No.3, September 2012.

[7] Karim M. Abdellatif, R.Chotin-Avot, and H. Mehrez, "The Effect of S-box Design on Pipelined AES Using FPGAs", Colloque GDR SOC-SIP, Paris, France, pp. 1-4 (2012).

[8] Swierczynski, P, Pawel Swierczynski, Fyrbiak, M., Koppe, P. , Paar, C., "Trojans through Detecting and Weakening of Cryptographic Primitives", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Feb 2015.