Research Paper

Volume : 2 | Issue : 7 | July 2013 • ISSN No 2277 - 8179

Fast pipelined AES algorithm implemented on Xilinx FPGAs

Engineering

KEYWORDS : Cryptography, AES, FPGA, encryption, decryption.

Anusha D Sphoorthy Engineering College, Student, M. Tech, E.C.E Dept., Hyderabad, India

Swathi G Sphoorthy Engineering College, Asst .Prof., E.C.E Dept., Hyderabad, India

ABSTRACT

The Advanced Encryption Standard (AES) is a specification for the encryption of electronic data also called Rijndael. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data. In the proposed work we present an efficient cryptography hardware implementation and its improvement using pipelines. The algorithm was implemented in FPGA due to its flexibility and reconfiguration capability. A reconfigurable device is very convenient for a cryptography algorithm since it allows cheap and quick alterations. The implementation of pipelined cryptography hardware was used to improve performance in order to achieve higher throughput and greater parallelism. The VHDL description will be implemented on FPGAs. Modelsim Xilinx Edition will be used for functional simulation and verification of results. Xilinx ISE will be used for synthesis. The Xilinx's chipscope tool will be used for verifying the results on Spartan 3E FPGA.

I. INTRODUCTION

In 1997, the National Institute of Standards and Technology – NIST released a contest to choose a new symmetric cryptograph algorithm that would be called Advanced Encryption Standard – AES to be used to protect confidential data in the USA. The algorithm should meet few requirements such as copyright free, faster than the 3DES, cryptograph of 128 bit blocks using 128, 192 and 256 bit keys, possibility of hardware and software implementation, among others. In 2000, after analysis by cryptography experts, it was chosen the winner: Rijndael. The algorithm was created by the Belgians Vincent Rijmen Joan Daemen. Hardware-based cryptography is used for authentication of users and of software updates and installations. Software implementations can generally not be used for this, as the cryptographic keys are stored in the PC memory during execution, and are vulnerable to malicious codes. Hardware-based cryptography, when implemented in a secure manner, is demonstrably being superior to software-based encryption.

II. DESCRIPTION OF AES ALGORITHM

The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plain-text.

A. AES Encryption

The AES algorithm operates on a 128-bit block of data and executed Nr - 1 loop times. A loop is called a round and the number of iterations of a loop, Nr, can be 10, 12, or 14 depending on the key length. The key length is 128, 192 or 256 bits in length respectively. In order to better understand the AES structure it is necessary to know the definition of state in the algorithm. State is the matrix of bytes that is processed between many stages, or rounds, and therefore, it will be modified in each stage. In the Rijndael algorithm, the matrix size depends on the block size being used, composed of 4 lines and Nb columns. Here, Nb is the number of bits in the block, divided by 32. Since the AES algorithm uses 128 bit blocks, the state will be composed by 4 lines and 4 columns.

On the encryption algorithm, there will be 4 phases: AddRoundKey, SubBytes, ShiftRows and MixColumns.

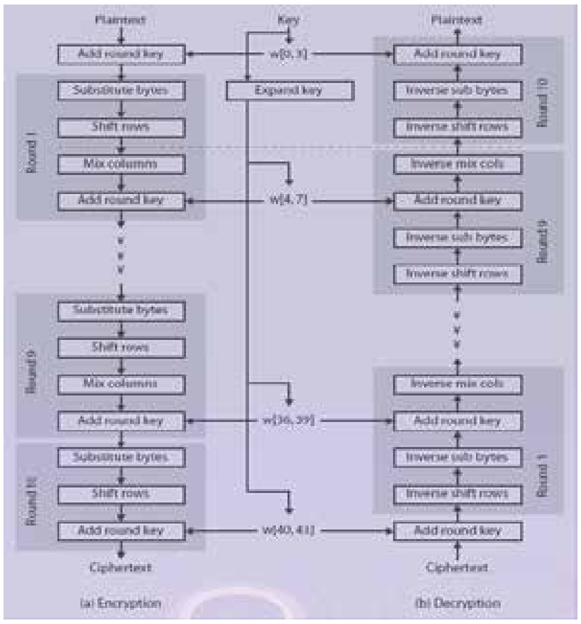
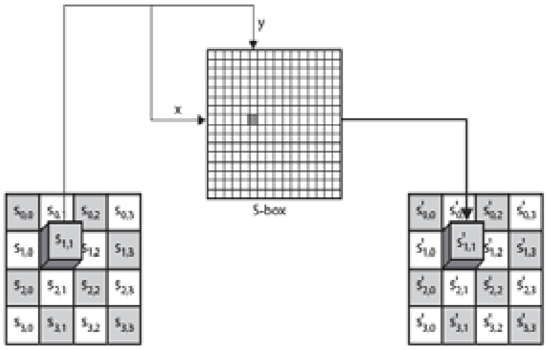


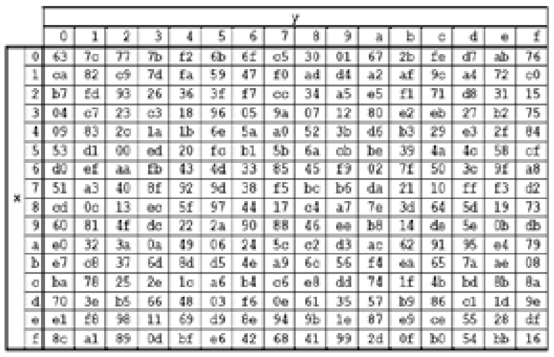
Figure 1. Advanced Encryption Standard

SubBytes Transformation

In Subbytes transformation, each state byte is replaced by another in the S-box, as indicated in Fig. 2. The replacement follows a matrix, where the first hexadecimal value corresponds to the line positioning, and the second hexadecimal value corresponds to the column positioning. As an example, the S-box outputs 24 for the input value A6 (Figure 3 – line A, column 6).

Figure 2. SubBytes operation process





		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
	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
x	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 3. S-box

ShiftRows Transformations
In ShiftRows transformation, the rows of the state are cyclically left shifted over different offsets, as indicated in Fig. 4. Row 0 is not shifted; row 1 is shifted one byte to the left; row 2 is shifted two bytes to the left and row 3 is shifted three bytes to the left.

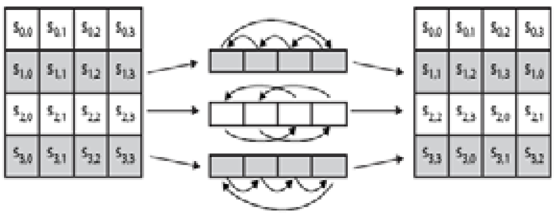

$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Figure 4. ShiftRows operation process

MixColumns Transformation
In MixColumns Transformation, the state bytes are treated as polynomials of Galois Field algebra. The operation can be represented as a matrix multiplication, as indicated in Fig. 5, where S is the initial state and S´ is the final state, after the operation.

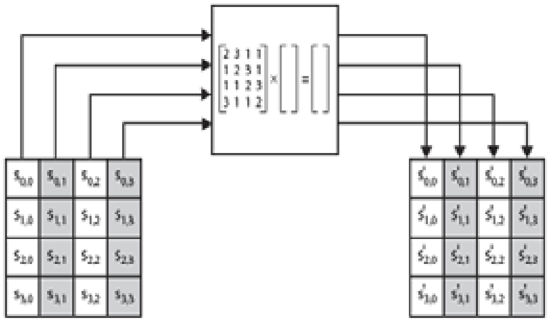

2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 5. MixColumns operation process

AddRound Key Transformation
In the AddRoundKey transformation, a Round Key is added to the State by a simple bitwise XOR operation, as indicated in Fig. 6. The RoundKey of each round is derived from the main key using the KeyExpansion algorithm.

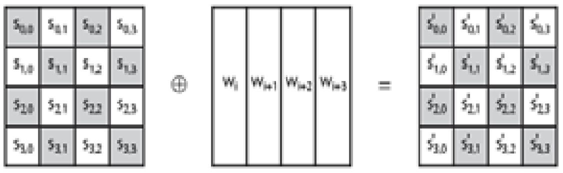
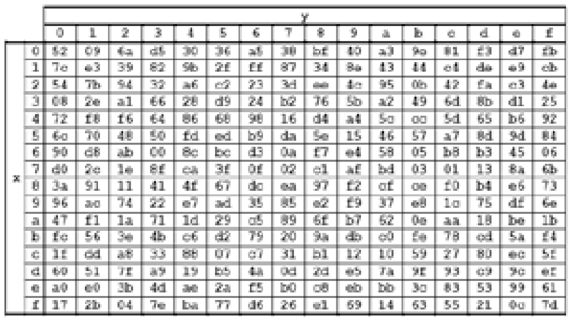

$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
⊕
W_i W_{i+1} W_{i+2} W_{i+3}
=
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 6. AddRound Key operation process

B. AES Decryption
Decryption is a reverse of encryption which inverse round transformations to compute out the original plaintext of an encrypted cipher-text in reverse order. The round transformation of decryption uses the functions AddRoundKey,InvMixColumns, InvShiftRows, and InvSubBytes successively.

InvSubBytes Transformation
The InvSubBytes Transformation is the inverse of the SubBytes Transformation in which the inverse S-Box, as indicated in Fig. 7, is applied to each byte of the State.



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

Figure 7. Inverse S-box

InvShiftRows Transformations
InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

InvMixColumns Transformation
The InvMixColumns is the inverse of MixColumns which consists of the multiplication using the inverse matrix. In the last round, on both the encryption and decryption algorithms, the MixColumns operation is suppressed.

AddRound Key Transformation
AddRoundKey is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order.

C. Key Expansion
The Key size defines the number of rounds in the encryption/decryption algorithm, and it also defines its expansion process. Basically, the Key Expansion operation consists of three operations, as presented in Fig. 8. The first operation, RotWord, makes a one byte circular shifting on the word. The second operation, SubWord replaces each byte of the input word according to the S-Box. The third operation consists of XOR operations, as indicated in Fig.8.

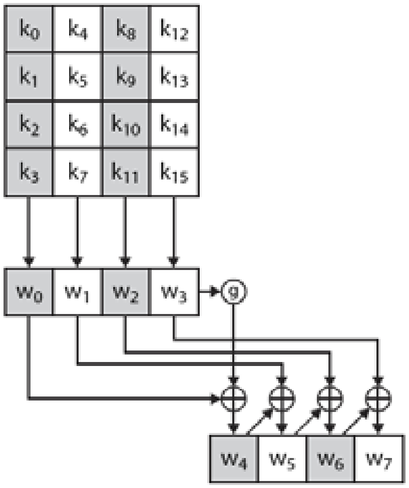

k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7

Figure 8. Key Expansion Operation Process

III. DESIGN IMPLEMENTATION

The AES top block diagram consists of control FSM, encryption, decryption, add round key, key expansion, as indicated in Fig. 9. Control FSM block controls the whole operation of AES. For encryption, input plain text and round key of 128 bit, data will be given by control unit and the output from the encryption will be given back to control unit.

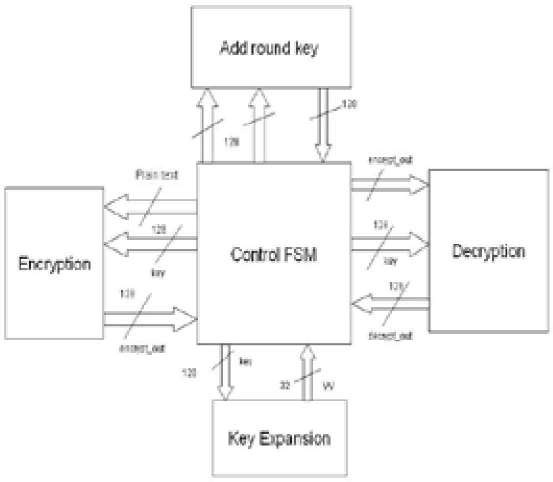


Figure 9. Architecture diagram of AES

Figure 10 shows the Encryption block with its I/Os. There are three inputs: the selector that chooses the embedded hardware according the rounds (first, last or other); the round keys in which each key is called; and the word of each phase of encryption that is used after the calculations (feedback). The output is the final encryption word.

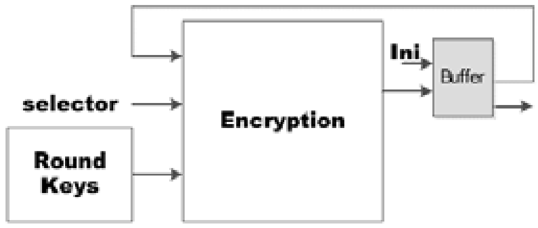


Figure 10. I/O's of Encryption block

The Encryption block architecture diagram of is presented in Figure 11. The Decryption Block follows the same scheme, with its own functions and procedures.

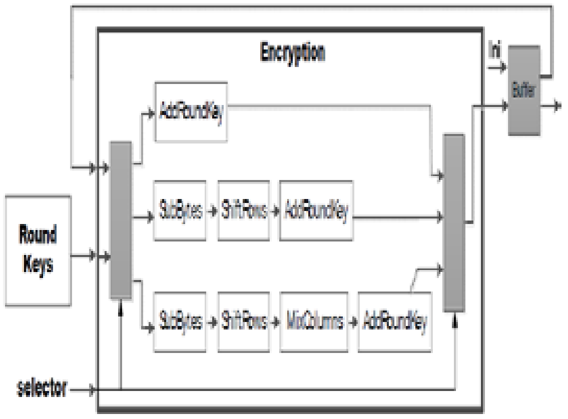
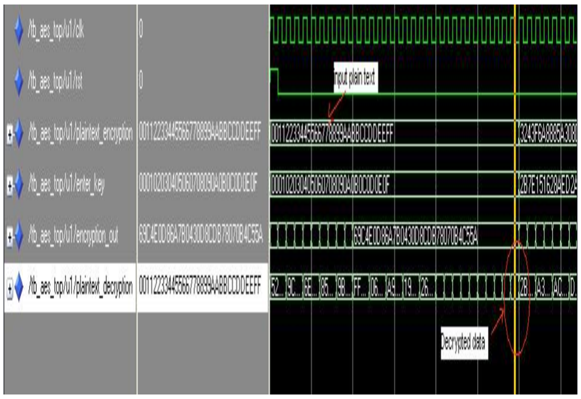


Figure 11. Architecture diagram of Encryption block

IV. SIMULATION RESULTS

The design has been coded by VHDL. Modelsim Xilinx Edition is used for functional simulation and verification of results. The results of simulating the encryption/decryption algorithm from the ModelSim simulator are shown in Fig.12.



```
/tb_aes_top/u1/clk                    0
/tb_aes_top/u1/rst                    0
/tb_aes_top/u1/plaintext_encryption   00112233445566778899AABBCCDDEEFF
/tb_aes_top/u1/enter_key              000102030405060708090A0B0C0D0E0F
/tb_aes_top/u1/encryption_out         69C4E0D86A7B0430D8CDB78070B4C55A
/tb_aes_top/u1/plaintext_decryption   00112233445566778899AABBCCDDEEFF
```

input plain text
00112233445566778899AABBCCDDEEFF
3243F6A8885A308
000102030405060708090A0B0C0D0E0F
2B7E151628AED2A
69C4E0D86A7B0430D8CDB78070B4C55A
52... 9C... 6E... 85... 9B... FF... 06... A9... 19... 26...
28... A3... AC... D...
Decrypted data

Figure 12. Simulation of AES encryption and decryption

The plaintext_encryption, enter_key are the 2 inputs of each 128-bit. The encryption_out will be given to decryption as an input, the plaintext_decryption will be the decrypted output as shown in Fig 12.

Chip Scope Results

The Xilinx's chipscope tool will be used for verifying the results on Spartan 3E FPGA (shown in Fig.10). Chip Scope is an embedded, software based logic analyzer. By inserting an "Integrated Controller Core" (ICON) and an "Integrated Logic Analyzer" (ILA) into the design and connecting them properly, the signals in the design can be monitored. ChipScope provides with a convenient software based interface for controlling the "integrated logic analyzer," including setting the triggering options and viewing the waveforms.

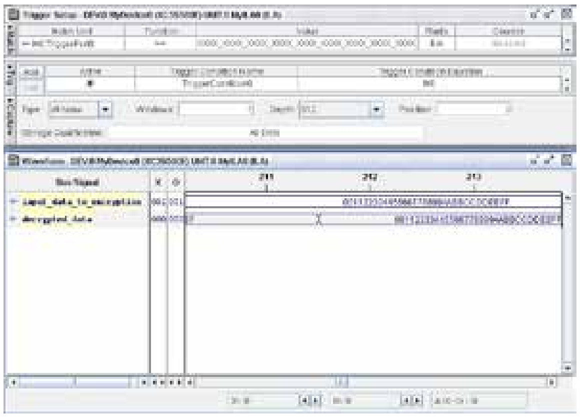


Figure 11. Chipscope results of AES encryption/decryption.

V.CONCLUSION

In this paper both the Encryption and Decryption sections of AES have been implemented on FPGA using Very High Speed Integrated Circuit Hardware Description Language (VHDL). A secure and effective Advanced Encryption Standard is established. The FPGA design will operate with an average encipher-decipher maximum frequency (including key expansion) of 409.333 MHZ. In place of S_BOX, SRAM memory blocks are used to reduce the number of slices and LUTs. The Modelsim simulator is used to simulate the design at various stages. Xilinx synthesis tool (XST) is used to synthesize the design for spartan3E family FPGA (XC3S500E). The circuit implementation is very efficient and can be customized to a wide range of applications.

Acknowledgement

I would like to articulate my profound gratitude and indebtedness to Dr. V.Chandra Mouli, Principal, Sphoorthy Engineering College., and Prof. Mr B.Sridhar, Head of the Department, Sphoorthy Engineering College, for guiding and encouraging me in all aspects.

I wish to extend my sincere thanks to all the faculty members of Sphoorthy Engineering College.

REFERENCE

[1] FIPS FIPS-197, Federal Information Processing Standards Publication FIPS-197, Advanced Encryption Standard (AES), http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf, 1999. | [2] DAEMEN, J. AND RIJMEN, V., The design of Rijndael: AES — The Advanced Encryption Standard. Springer-Verlag, 2002. | [3] GOMES, O. S. M.; PIMENTA, T. C.; MORENO, R. L., "A Highly Efficient FPGA Implementation", 2nd Latin America Symposium on Circuits and Systems(LASCAS-2011), February 2011. | [4] X. Zhang and K. K. Parhi, “High-speed VLSI architectures for the AES algorithm,” IEEE Trans. Very Large Scale Integer. (VLSI) Syst., vol. 12, no. 9, pp. 957–967, Sep. 2004. | [5] C. CHIEN, D. CHIEN, C. CHIEN, I. VERBAUWHEDE AND F. CHANG, "A hardware implementation in FPGA of the Rijndael algorithm", The 2002 45th Midwest Symp. Circuits and Systems (MWSCAS-2002), Vol. 1,4--.7 August 2002, pp. 507-509. | [6] E. J. SWANKOSKI, V. NARAYANAN, M. KANDEMIR AND M. J. IRWIN, "A parallel architecture for secure FPGA symmetric encryption", 18th Int. Parallel and Distributed Processing Symp. (IPDPS'04) - Workshop, Santa Fe, New Mexico, 26-.30 April 2004, p. 123. | [7] Chi-Jeng Chang “8-bit AES FPGA Implementation using Block RAM,” IEEE trans. Very Large Scale Integer. (VLSI) syst., Nov 2007. | [8] ARSHAD AZIZ AND NASSAR IKRAM, "Memory efficient implementation of AES S-boxes on FPGA", Journal of Circuits, Systems, and Computers, Vol. 16, No.4 (2007) 603--611. | [9] P. M. KOGGE. “The Architecture of Pipelined Computers”. McGraw-Hill Book Company, New York, NY, 1981.s |