International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

An Efficient Design of Advanced Encryption Algorithm with FPGA

Soraisham Tarunjit Meitei[1], M. Rajmohan[2]

Department of Electronics and Communication Engineering, Hindustan University, Chennai – 603103, India

Abstract: ***A FPGA-based implementation of the Advanced Encryption Standard (AES) algorithm is presented in this paper. This implementation is performed using a reconfigurable 32-bit MicroBlaze processor embedded in the FPGA chip using RS232 to interface with PC to obtain a prototyped data encryption/decryption system. The iterative looping approach with block and key size of 128 bits, lookup table implementation of S-box will performed. Simulation results, data summary results will be carried out with previous reported designs.***

Keywords: AES, FPGA, encryption, decryption, Rijndael, block cipher.

1. Introduction

Encryption [1] is a common technique to uphold image security. Image and video encryption [2] are found in various fields that include internet communication, multimedia techniques, image processing systems, telecom security and defense application.

The Advanced Encryption Standard– AES, announced by the NIST (National Institute of Standards and Technology) [1] became a standard in 1997 for a symmetric cryptographic algorithm to be used to protect confidential data in the USA. The algorithm should meet a better secure and faster technique than the 3DES, using 128 bit cryptograph blocks using 128, 192 and 256 bit keys. The possibility of hardware implementation rather then software implementation gives a better efficient technique. In 2000, Rijndael [3] was chosen by a group of cryptographic experts to be the most reliable technique. This algorithm was designed by the Belgians Vincent Rijmen e Joan Daemen [1][3].

The software implementation can prove vulnerable to attacks by trespassers so hardware-based cryptography providing more secure platform [2] in authentication of users is suggested. This superior cryptographic work can provide an effective and efficient security cryptanalysis attacks, cyber-attacks and offline attacks.

This paper deals with an FPGA(Field programmable Gate Array) implementation using a 32-bit Microblaze embedded core processor for the AES encryption/decryption. The iterative looping approach with block and key size of 128 bits using the lookup table implementation of S-box [7] enhances our design. This method gives an ease in the encryption/decryption process resulting in a very low complexity architecture achieving low area as well as high throughput.

Section II describes the AES algorithm, Section III provides the hardware implementation of the AES design, Sections IV gives the results obtained and Section V gives the conclusion.

2. AES Rijndael

The AES algorithm [4][5] which is a symmetric block cipher includes both encryption and decryption of data and information. Encryption converts a root form of data called the plain-text to an unintelligible form called the cipher-text. The decryption operation does the reverse and makes the original plain-text to be recovered.

The AES encryption and decryption process can be observed using the encryption structure as in figure1 and figure 2 respectively. A state is generated in each stage of the design flow and the matrix of bytes that is processed between many stages, or rounds, gets modified in each stage. In the Rijndael algorithm [3], the size of the matrix depends on the block size being used. The matrix is composed of 4 lines and Nb columns. Nb represents the number of bits in the block, comprising the state. As our AES algorithm uses 128 bit blocks, the state will be defined by 4 lines and 4 columns.

The cipher key which is a secret cryptographic key is used by the key expansion module that generates a set of round keys having 4 rows and Nk columns. Nr gives the number of rounds and is a function of Nk and Nb. The key expansion technique, thus, generates a series of round keys.

Accordingly as we choose the key size to be 128, 192 or 256, Nr will be 10, 12 and 14 and Nk be 4, 6 and 8, respectively.

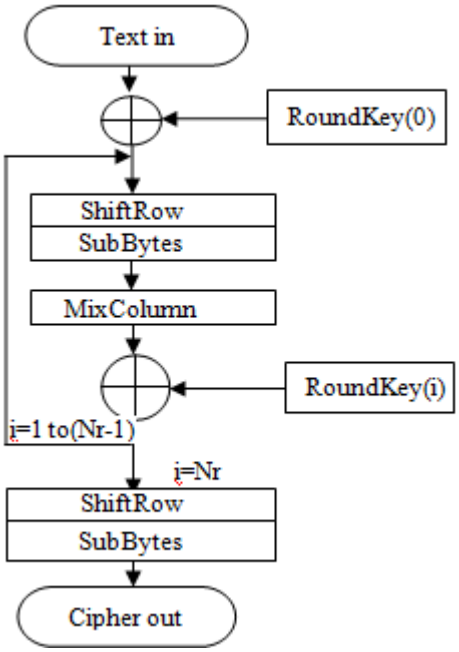

Figure 1: AES Encryption Structure

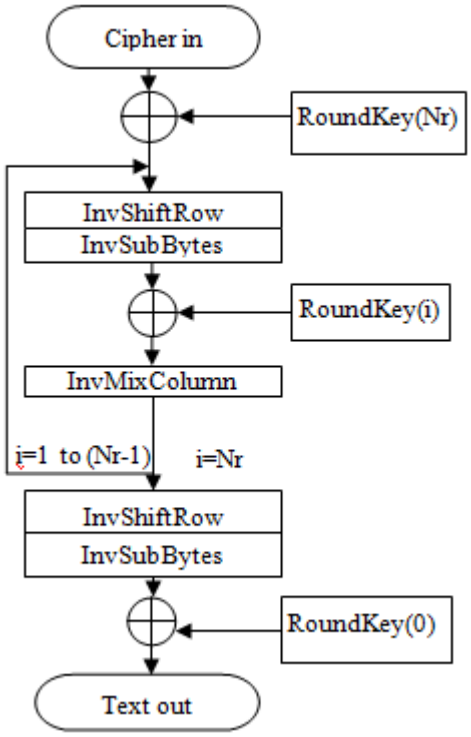

Figure 2: AES Decryption Structure

The encryption algorithm undergoes 4 different operations i.e. AddRoundKey, SubBytes, ShiftRows and MixColumns. In the last stage, the Mix Columns operation is suppressed. The decryption algorithm performs the respective inverse operations i.e. InvAddRoundKey, InvSubBytes, InvMixColumns and InvShiftRows. Here the InvMixColumns is suppressed on the last stage of decryption algorithm [3][5]. The encryption/decryption algorithm is explained by the individual operations as follows:

A. SubBytes

In this operation each state byte is replaced with the corresponding values as per the S-box as indicated in Table 1. The values of the rows and columns of the state are used to find the corresponding replacement values, which are present in hexadecimal format. For the inverse operation (decryption) i.e. InvSubBytes, an inverse S-Box(Table II) is used. As an example, the S-box [7] outputs e0 for the input value a0 (Table I - line a, column 0). On the same way, the inverse S-Box outputs a0 for the input value e0 (Table II - line e, Column 0). Figure 3 shows the operation of the SubBytes.

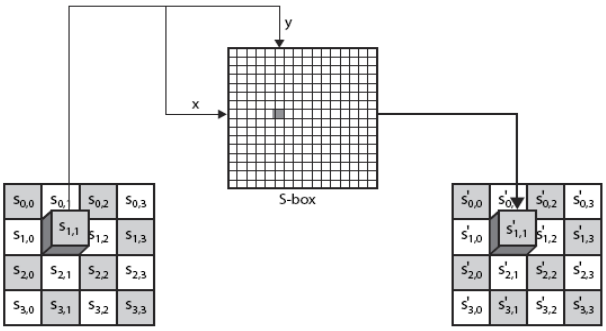

Figure 3: SubBytes

Table 1: S-BOX

		Y															
		0	1	2	3	4	5	6	7	8	9	A	b	c	d	e	f
X	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

B. ShiftRows

After the SubBytes operation the resulting matrix is shifted, and the original matrix is replaced by the new matrix. In this paper we use a left shift of one position in the 2nd row, two shifts in the 3rd row and three shifts in the 4th row. The first row is kept as it is. It is shown in figure 4.

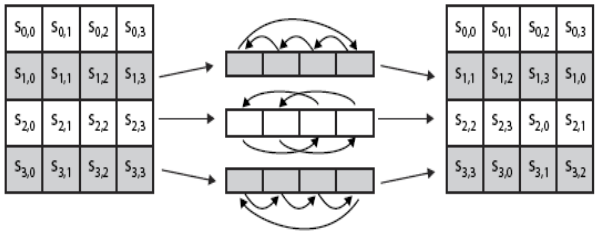

Figure 4: ShiftRows

In the inverse operation, performed by InvShiftRows, the corresponding rows shifted are to the right accordingly, in the decryption algorithm.

C. MixColumns

This transformation operates on the state column-by-column. The columns are treated as a four-term polynomial over GF(2^8)(Galois Field) multiplied with a fixed polynomial. The state bytes are treated as polynomials of Galois Field algebra GF(28) [11] in this operation. In the Figure 5, S depicts the initial state and S′ depicts the final state, after the operation has been carried out.

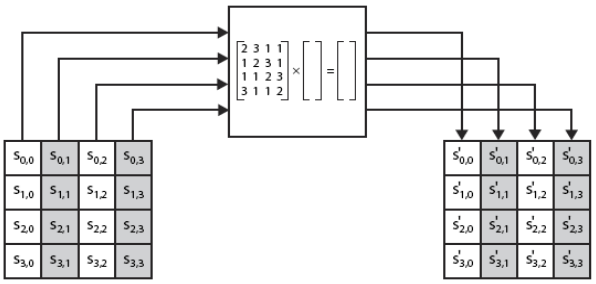

Figure 5: MixColumns

Table 2: InvS-BOX

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	F
X	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
	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

In the InvMixColumns, the multiplication uses the corresponding inverse matrix denoted by C′ for the original C matrix which was used in encryption. In the last round in both encryption and decryption, this operation does not take place.

$$C = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \qquad C' = \begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}$$

D. AddRoundKey

This operation is performed in encryption as well as decryption algorithm. XOR operation is performed by the state and the round key which is generated from Key expansion module. The key expansion technique will further be discussed in later part of the paper. The corresponding column-by-column for the state matrix and the round matrix is performed XOR operation and is substituted in the state matrix. The new byte $S'_{x,y}$ is given by $S_{x,y} \oplus K_{x,y}$. This can be seen in figure 6.

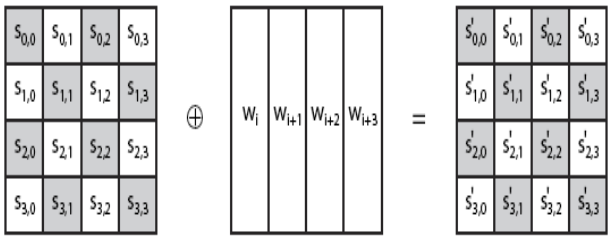

Figure 6: AddRoundKey

E. Key Expansion

The key expansion [1] is a routine used to generate a series of Round Keys from the Cipher Key. Cipher key defines the number of rounds in both, encryption and decryption algorithms. The expansion technique is as shown in figure 7. The expansion technique involves the use of three different operations. The first operation, RotWord, takes a four-byte word and performs a cyclic permutation. The second operation, SubWord takes a four-byte input word and applies an S-box to each of the four bytes to produce an output word. The third operation involves XOR operations.

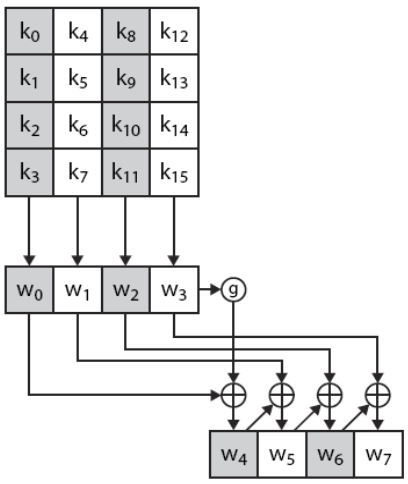

Figure 7: Key Expansion

3. Hardware Implementation

The AES design is implemented using a reconfigurable [6] 32-bit MicroBlaze processor embedded in the FPGA chip, by using a parallel JTAG interface from the PC(Personal Computer) to the FPGA and a serial RS232 to interface from FPGA to PC to obtain a prototyped data encryption/ decryption system. This Microblaze is said to be a soft core processor supporting around 900 LUTs, 32x32 general purpose registers that have separate instructions for accessing data and memory. It supports on-chip Block-RAM as well as external memory and follows RISC (Reduced Instruction Set Computers) architecture. The general block diagram of a Microblaze processor is as shown in figure 8.

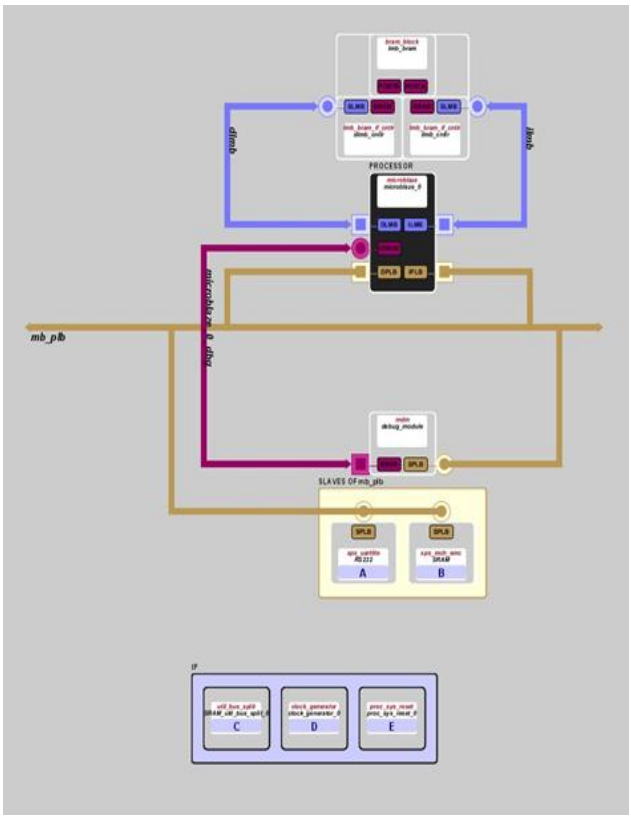

Figure 8: Microblaze processor

An EDK (Embedded Development Kit) tool, called Xilinx Platform Studio has been used to implement this design. VHDL is used for configuration of the hardware and Impulse C for software, which is used to drive the hardware design. Efficiency is observed using a Spartan-3 XC3S200 FPGA.

The use of EDK in the AES design enables the processor IP configuration in the database and lowers the complexity in architecture providing an ease in design platform as compared to the Xilinx ISE. The Block diagram for AES implementation in FPGA using EDK software is shown in figure 9 in parallel with Xilinx ISE design flow.

For SubByte lookup table having multiplications and additions (XORs) two 512×2 SRAMs (Static RAMs) is proposed in slice area for S-BOX.

All the modules were independently tested and characterized, and hence they can be used in any combination as desired, as per the application requirements. Tests were performed on each and every block using an encryption - decryption AES set of 128 bits that was assembled in a Xilinx Spartan-3 FPGA device.

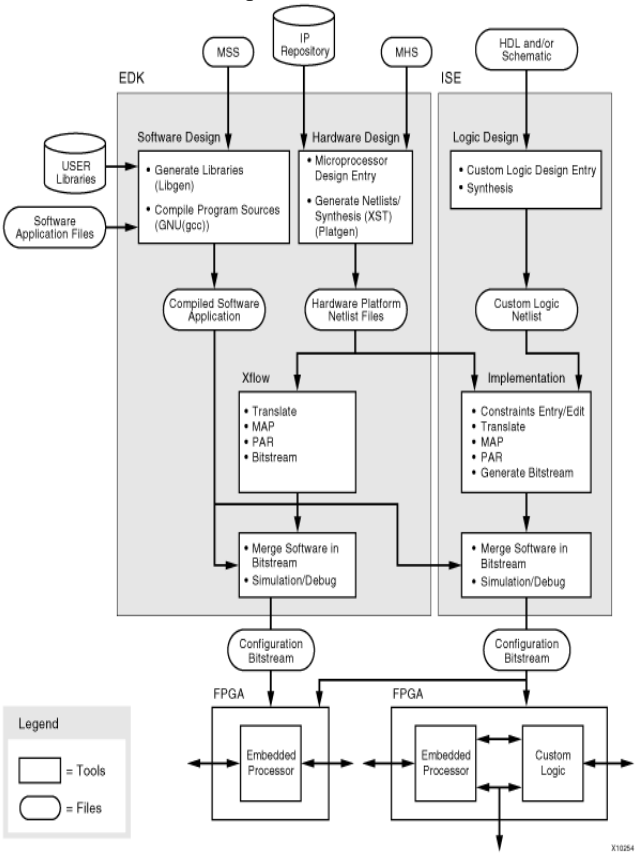

Figure 9: Block diagram for AES implementation in FPGA using EDK software

4. Results Analysis

Xilinx ISE version 10.1i was used for the design flow and the results quoted are from post place and route figures including all input and output delays. The new designs were coded in VHDL and validated using ISE simulator.

Using Xilinx 10.1i the encryption result is observed for:
Plaintext = 123456h
Key Value = 123h

The Cipher Text is found to be B1B10EDC18EE9518862CC9C963636240h
The simulation result is as shown in figure 10.

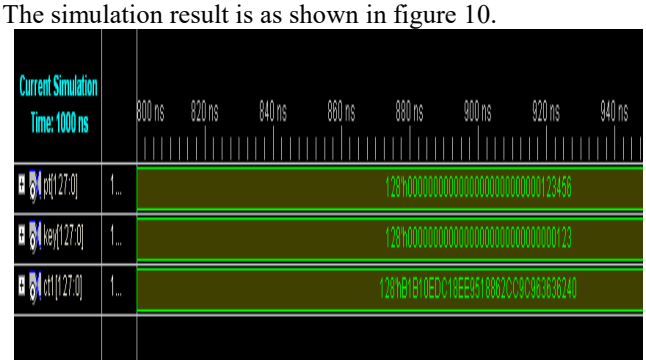

Figure 10: AES Encryption Simulation

Now the same tool the decryption value for the cipher text is observed.
Plaintext= B1B10EDC18EE9518862CC9C963636240h
Key Value = 123h
Cipher Text is found to be 123456h, which is our input value. The simulation result is as shown in figure 11.

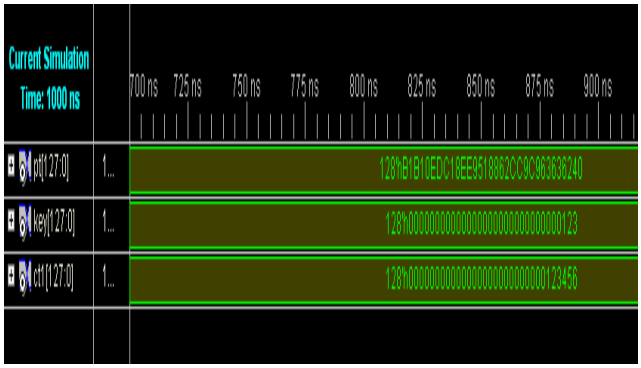

Figure 11: AES Decryption Simulation

The hardware implementation was performed using Microblaze soft core processor embedded in the Spartan-3 XC3S200 through Xilinx Platform Studio, an EDK(Embedded Development Kit) tool. All peripherals are implemented on the FPGA(Field Programmable Gate Array) fabric. The reference clock frequency and the processor bus clock frequency inputs were set to 50Mhz and the maximum frequency obtained was 72.495Mhz. Also the resulted device utilization data indicating the availability and consumed area is as shown in table III.

Table 3: Device utilization for Spartan-3 XC3S200

Resource type	Used	Available	Percent
Slices	712	1920	37
Slice Flip Flop	898	3840	23
4 input LUTs	1377	3840	35
IOs	2296	NA	NA
Bonded IOBs	0	173	0
MULT18x18s	3	12	25

Also the total power consumption is obtained as in table IV below

Table 4: Power Analysis

Name	Value (W)	Used	Total Available	Utilization
Clocks	0.00508	3	---	---
Logics	0.00756	2379	3840	62.0
Signals	0.01043	3431	---	---
I/Os	0.01298	30	97	30.9
BRAMs	0.00000	4	12	33.3
DCMs	0.03663	1	4	25.0
MULTs	0.00055	3	12	25.0
Total Power	0.11483			

5. Conclusion

The article presented a very efficient approach of encryption/decryption cryptography which can give various applications. The algorithm has been designed for a 128 bit block size which can be used with cipher keys length 128, 192 and 256 bits. The design is implemented using a soft core Microblaze processor with Xilinx Platform Studio, an EDK tool.

Xilinx Spartan-3 XC3S200 was chosen to conduct the performance comparison of our work with others [4][5]. Xilinx ISE 10.1 was the software used to run the synthesis, and Xilinx Platform Studio was used for hardware implementation. Area was greatly reduced and also an ease in hardware implementation process was experienced.This approach presented an efficient AES cryptography hardware structure can be customized to a wide range of applications.

References

[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] Vishnu, M.B. ; Tiong, S.K. ; Zaini, M. ; Koh,S.P. "Security enhancement of digital motion image transmission using hybrid AES-DES algorithm" 14th Asia-Pacific Conference on Communications, 2008. APCC 2008. Publication Year: 2008 , Page(s):1-5 Cited by: Papers (2).
[3] Daemen J., and Rijmen V, "The Design of Rijndael: AES-the Advanced Encryption Standard", Springer-Verlag, 2002.
[4] Mr. Atul M. Borkar, Dr. R. V. Kshirsagar and Mrs. M. V. Vyawahare, "FPGA Implementation of AES Algorithm", International Conference on Electronics Computer Technology (ICECT), pp. 401-405, 2011.
[5] Trang Hoang; Van Loi Nguyen, "An Efficient FPGA Implementation of the Advanced Encryption Algorithm" Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on Publication Year: 2012 , Page(s): 1-4 Cited by: Papers (3).
[6] Tessier, R., and Burleson, W., "Reconfigurable computing for digital signal processing: a survey", J.VLSI Signal Process, 28, (1-2), pp.7-27, 2001.
[7] Ahmad, N.; Hasan, R.; Jubadi, W.M; "Design of AES S-Box using combinational logic optimization", IEEE Symposium on Industrial Electronics & Applications(ISIEA), pp. 696-699, 2010.
[8] Alex Panato, Marcelo Barcelos, Ricardo Reis, "An IP of an Advanced Encryption Standard for Altera Devices", SBCCI 2002, pp. 197-202, Porto Alegre, Brazil, 9 and 14 September 2002.
[9] Otávio S. M. Gomes, Robson L. Moreno and Tales C. Pimenta, "A Fast Cryptography Pipelined Hardware developed in FPGA with VHDL" Ultra Modern Telecommunications and Control Systems and Workshop(ICUNT), 3rd International conference, 2011, Page(s): 1 – 6, 2011.
[10] J. Granado-Criado, M. Vega-Rodriguez, J. Sanchez-Perez, and J. Gomez-Pulido,"A New Methodology to Implement the AES Algorithm Using Partial and Dynamic Reconfiguration," Integration, the VLSI J., vol. 43, no. 1, pp. 72-80,2010.
[11] Klima, R. E., Sigmon, N., and Stitzinger, E. Applications of abstract algebra with Maple. CRC Press, Boca Raton, FL. 2000.
[12] Dur-e-Shahwar Kundi, Saleha Zaka, Qurat-Ul-Ain and Arshad Aziz, "A Compact AES Encryption Core on

Xilinx FPGA", 2nd IEEE International Conference on Computer, Control & Communication (IEEE IC4-2009) Karachi, Pakistan Vol:1 pp:1-4, 2009.

[13] 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.