Chapter

High-Speed Area-Efficient Implementation of AES Algorithm on Reconfigurable Platform

Altaf O. Mulani and Pradeep B. Mane

Abstract

Nowadays, digital information is very easy to process, but it allows unauthorized users to access to this information. To protect this information from unauthorized access, cryptography is one of the most powerful and commonly used techniques. There are various cryptographic algorithms out of which advanced encryption standard (AES) is one of the most frequently used symmetric key cryptographic algorithms. The main objective of this chapter is to implement fast, secure, and area-efficient AES algorithm on a reconfigurable platform. In this chapter, AES algorithm is designed using Xilinx system generator, implemented on Nexys-4 DDR FPGA development board and simulated using MATLAB Simulink. Synthesis results show that the implementation consumes 121 slice registers, and its maximum operating frequency is 1102.536 MHz. Throughput achieved by this implementation is 14.1125 Gbps.

Keywords: cryptography, AES, FPGA, VLSI, system generator

1. Introduction

NIST has started a development process of FIPS for AES algorithm stating that this is the replacement for data encryption standard (DES) algorithm. Alternatively, this algorithm is also known as Rijndael algorithm. Rijndael algorithm has the advantages like resistance against all recognized attacks, code and speed compactness, and simple design. Cryptography is a process in which the information to be sent is added with secret key so as to transmit the data securely at the destination. There are two types of cryptography based on the type of key applied: symmetric key cryptography and asymmetric key cryptography. In symmetric key cryptography, equal key is utilized for encryption as well as decryption, whereas in asymmetric key cryptography, different keys are required in encryption and decryption. AES algorithm is selected for implementation because it is secure and its components and design principles are completely specified. AES is a symmetric key block cipher. The design of AES algorithm is based on linear transformation. Due to the use of Rijndael algorithm, different block and key sizes can be selected which was not possible in DES algorithm. Block and key size can be selected from 128/160/192/224/256 bits and need not be the same. According to AES standard, this algorithm can only accept 128 bits of block, and key size can be selected from 128/192/256 bits. Based on the key size, the number of rounds will vary. For example, if key size is 128, 192, or 256, then the number of rounds will be 10, 12, and 14, respectively. The structure of AES algorithm is shown in **Figure 1**. In this chapter,

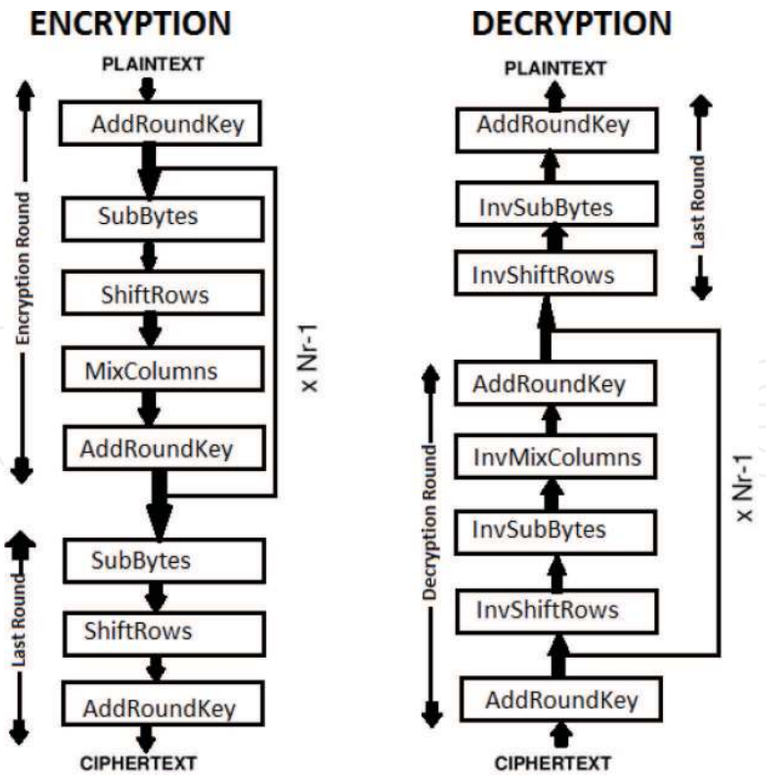

Figure 1.
Structure of AES algorithm.

this algorithm is designed with 128 bits of block size and key size, respectively, that is, AES generates cipher text of 128 bits for 128 bits of plaintext. After the initial round, plaintext processes through ten rounds. Each round contains processes like byte substitution, shift rows, mix columns, and add round key.

1.1 Byte substitution

The 16 input bytes are substituted by using fixed lookup table known as s-box. **Figure 2** shows s-box of AES algorithm. This s-box consists of all possible combinations of 8-bit sequence. The resulting new 16 bytes are organized in a matrix having four rows and four columns.

Figure 3 shows byte substitution stage in AES algorithm.

	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 2.
S-box of AES algorithm.

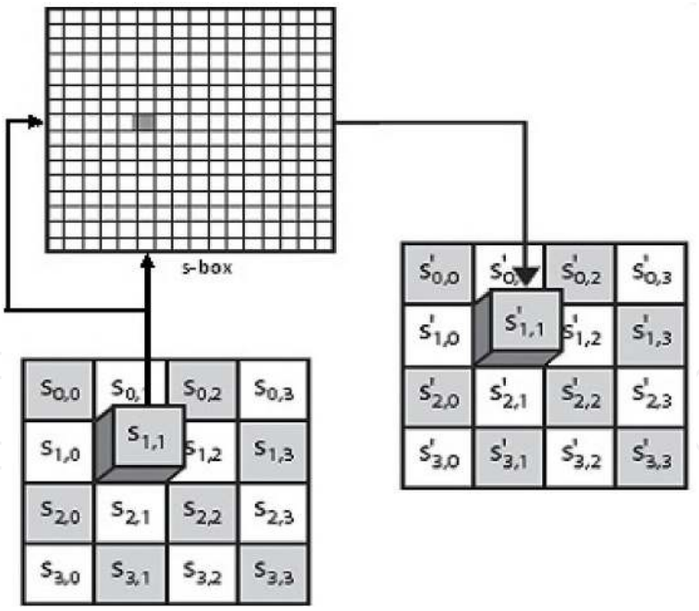

Figure 3.
Byte substitution stage.

1.2 Shift row

Each row from the matrix generated from the byte substitution is cyclically shifted to the left. Any entry that is dropped off is reinserted to the right side. The first row is kept as it is, the second row is shifted by one-byte position to the left, the third row is shifted by two-byte position to the left, and the fourth row is shifted by three-byte position to the left. The resultant matrix consists of same 16 bytes but at different position. **Figure 4** shows shift row stage in AES algorithm.

1.3 Mix column

Each column of four bytes is now transformed using special arithmetical function of Galois field (GF) 28. This function takes four bytes of the column as input and

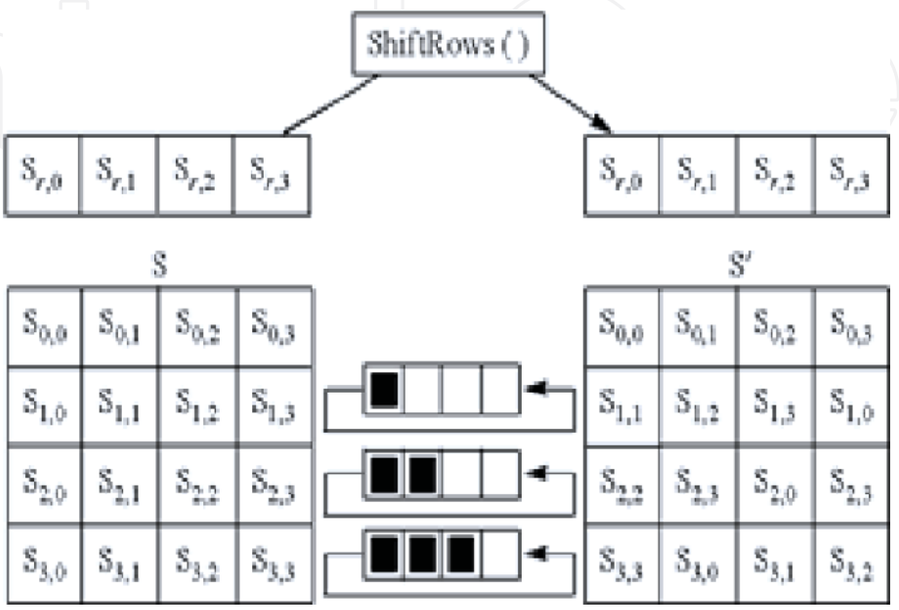

Figure 4.
Shift row stage.

outputs completely new four bytes that replaces the original four bytes. **Figure 5** shows mix column stage in AES algorithm.

1.4 Add round key

The 16 bytes of the resultant matrix generated from mix column stage are then considered as 128 bits. In add round key stage, 128 bits of state are bitwise EX-ORed with 128 bits of round key. If this result belongs to the last round, then the output is cipher text else the resulting 128 bits is considered as 16 bytes, and another round is started with new byte substitution process. This is a column-wise operation between four bytes of state column and one word of round key. In the last round, there is no mix column step. **Figure 6** shows add round key stage in AES algorithm.

Decryption of cipher text, generated from AES encryption, contains all the stages in encryption but in reverse order. AES decryption starts with inverse initial round. The remaining nine rounds in decryption consist of processes like add round key, inverse shift rows, inverse byte substitution, and inverse mix columns.

Add round key: Add round key has its own inverse function since XOR functions its own inverse and the round keys should be selected in reverse order.

Inverse shift rows: Inverse shift rows functions exactly in the same way as shift row stage but in opposite direction. The first row is kept as it is, the second row is shifted by one-byte position to the right, the third row is shifted by two-byte position to the right, and the fourth row is shifted by three-byte position to the right. The resultant matrix consists of same 16 bytes but at different position. **Figure 7** shows inverse shift row stage in AES algorithm.

Inverse byte substitution: Inverse byte substitution is done using predefined substitution table known as inverse s-box. **Figure 8** shows inverse s-box in AES algorithm.

Inverse mix column: Transformation in inverse mix column is done using polynomials of degree less than 4 over Galois field (GF) 28 in which coefficients are the elements from the column of the state.

The rest of the chapter is organized as follows:

Section 2 presents the survey based on the various kinds of implementation of AES algorithm on reconfigurable platform. In Section 3, implementation of AES algorithm using the proposed approach is discussed. In Section 4, experimental results achieved using the proposed method along with the comparative analysis with existing methods are discussed.

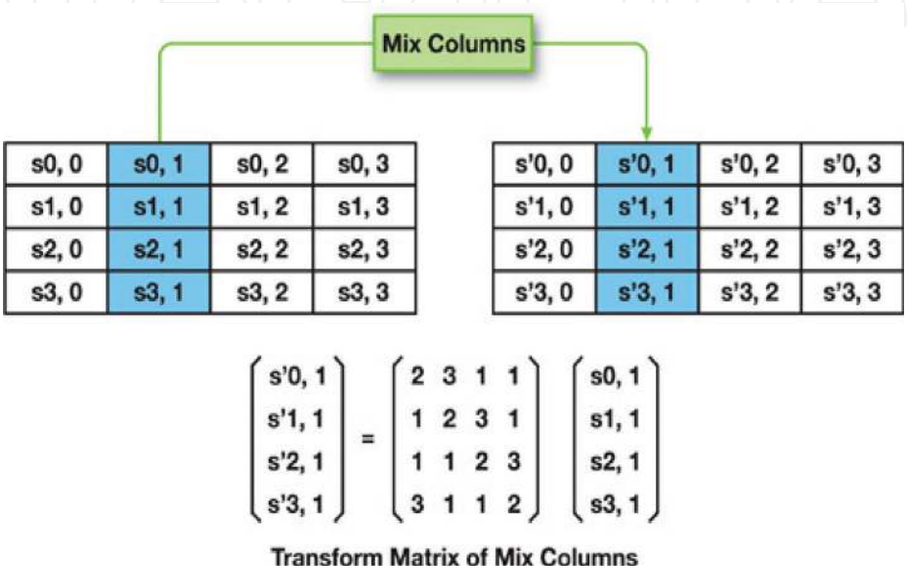

Figure 5.
Mix column stage.

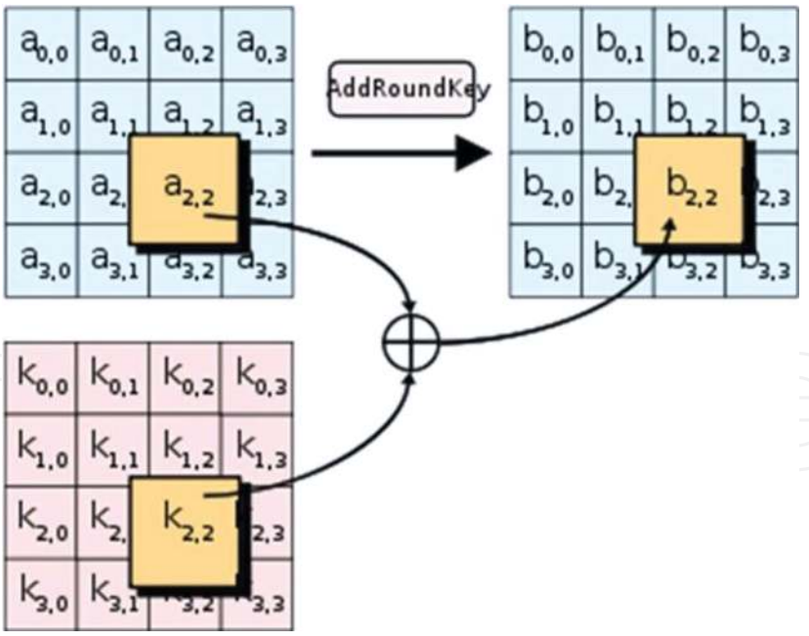

Figure 6.
Add round key stage.

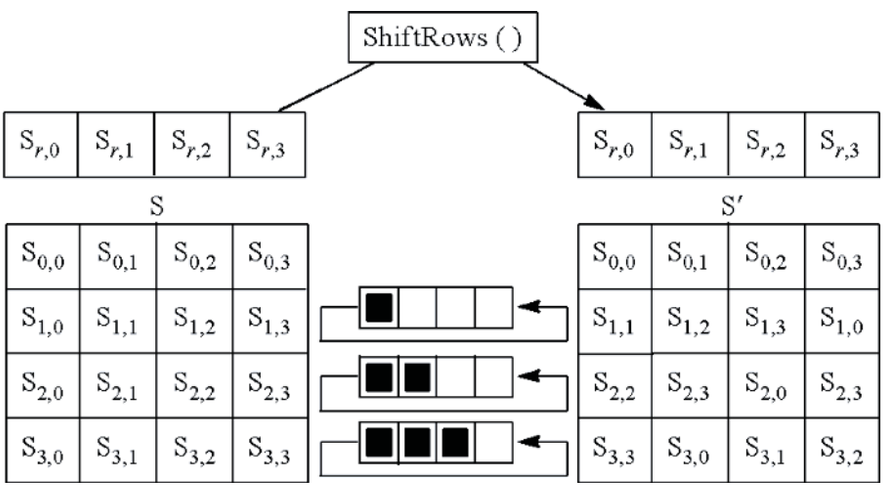

Figure 7.
Inverse shift row.

	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	82	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	89	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	84	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	87	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	38	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 8.
Inverse S-box of AES algorithm.

2. Literature survey

In this section, focus is given on the work done by various researchers on FPGA-based implementation of AES algorithm. There are various researchers which have either concentrated on area optimization or speed optimization. Mulani and Mane [1] discussed integrating of DWT and AES algorithm for implementation of watermarking on FPGA. The design was implemented on xc6vcx75t-2ff484, and it utilizes 2117 slices at maximum operating frequency of 228.064 MHz. Ratheesh and Narayanan [2] proposed implementation of AES algorithm with low-power MUX LUT-based s-box on FPGA. This design achieved total power distribution of 0.55 W. Agarwal et al. [3] suggested implementation of AES algorithm using Verilog on Spartan-3E FPGA. This design utilizes 1464 slices. Farooq and Faisal Aslam [4] discussed implementation of AES algorithm on FPGA device using five different techniques which are suitable for area critical applications and speed critical applications. This design was implemented on Spartan-6 FPGA device, and it utilizes 161 slices at maximum operating frequency which is 886.64 MHz. The throughput of this system is 113.5 Gbps. Sai Srinivas and Akramuddin [5] proposed less complex hardware implementation of AES Rijndael algorithm on Xilinx Virtex-7 XC7VX90T FPGA. In the proposed design, synthesis tool was set to optimize speed, area, and power. Mathur and Bansode [6] proposed a cryptosystem, which is a combination of AES algorithm and ECC. This is a hybrid encryption scheme and the key size is 192 bits and there are 12 numbers of iterations in this system. Kalaiselvi and Mangalam [7] proposed a low-power and high-throughput FPGA implementation of AES algorithm using key expansion technique. This design accepts key size of 256 bits for both encryption and decryption. This design utilizes 5493 slices, and its maximum operating frequency is 277.4 MHz. The throughput of this system is 0.06 Gbps. Deshpande et al. [8] suggested BRAM-based and FPGA-based implementation of AES algorithm. Due to the use of BRAMs for implementing s-box, this design utilizes less number of slices. The design was implemented on XC3S1400AN and it utilizes 3376 slices. Ibrahim [9] presented FPGA implementation of AES encryption core that is suitable for limited resource-limited applications. This design was implemented on Spartan-3, and it utilizes 150 slices at maximum operating frequency of 90 MHz. Khose and Raut [10] proposed implementation of AES algorithm on FPGA in order to achieve high speed of data processing and also to reduce time for generating key. This design utilizes 201 slices and 2 BRAMs at maximum operating frequency of 70 MHz. Mulani and Mane [11] proposed FPGA implementation of DES algorithm. The design was implemented on XC2S200, and it utilizes 2118 slices and 97 IOBs. Yewale Minal and Sayyad [12] proposed implementation of AES encryption using VHSIC hardware description language VHDL) and decryption using Visual Basic. With this approach, 1403 slices are utilized at maximum operating frequency of 160.875 MHz, and it has a throughput of 2.059 Gbps. Deshpande et al. [13] discussed FPGA-based optimized architecture that utilizes less area. This design was intended for plaintext of 128 bits and key of 128 bits. Tonde and Dhande [14] discussed FPGA-based implementation of AES algorithm using iterative looping approach for 128 bits of block and key size. Varhade and Kasat [15] proposed a FPGA-based AES algorithm, which utilizes 1746 logic elements and 32,768 memory bits. This design was synthesized on Cyclone-II using Altera. Wadi and Zainal [16] proposed some modifications like decreasing number of rounds and replacing S-box with new s-box to reduce hardware requirements in order to enhance the performance of AES algorithm in terms of time ciphering and pattern appearance. Wang et al. [17] suggested high-speed implementation of AES algorithm on FPGA to transmit the data securely using pipelining and parallel processing methods. Shylashree et al. [18] focused on various novel FPGA architectures of AES algorithm. Borkar et al. [19] proposed iterative design approach for FPGA implementation

of AES algorithm using VHDL. This design utilizes 1853 slices, and its operating frequency is 140.390 MHz. Deshpande et al. [20] presented very low complexity FPGA-based architecture for integrated AES encryptor and decryptor. This design is synthesized on Spartan-3 XC3S400 FPGA. Kaur and Vig [21] suggested an efficient implementation of AES algorithm on FPGA in which multiple rounds are processed simultaneously. Due to this implementation, speed is increased but it increases area. This design utilizes 6279 slices and 5 BRAMs, and its operating frequency is 119.954 MHz. Samanta [22] proposed fast and efficient reconfigurable platform-based implementation of AES algorithm using pipelining. This design utilizes 1051 slices and 11 BRAMs, and its operating frequency is 76.699 MHz. Good and Benaissa [23] discussed hardware implementation of fastest and slowest AES algorithm which utilizes 16,693 slices at maximum operating frequency of 184.8 MHz.

From the literature survey, it is clear that many researchers have either worked on optimizing the area or speed. Few researchers have concentrated on optimizing the speed as well as area. Implementation of AES algorithm, which is optimized in speed as well as area, is discussed in this chapter.

3. Implementation of AES algorithm

The proposed design is implemented with the aim to achieve both area and speed optimization. In the proposed design, keys for each round are initially generated by using MATLAB code, and then those keys are used in the design. Due to this approach, the design occupies less number of slices, and also the speed is faster than the normal approach. The design is implemented using Xilinx system generator. **Figure 9** shows Xilinx system generator-based model for AES algorithm.

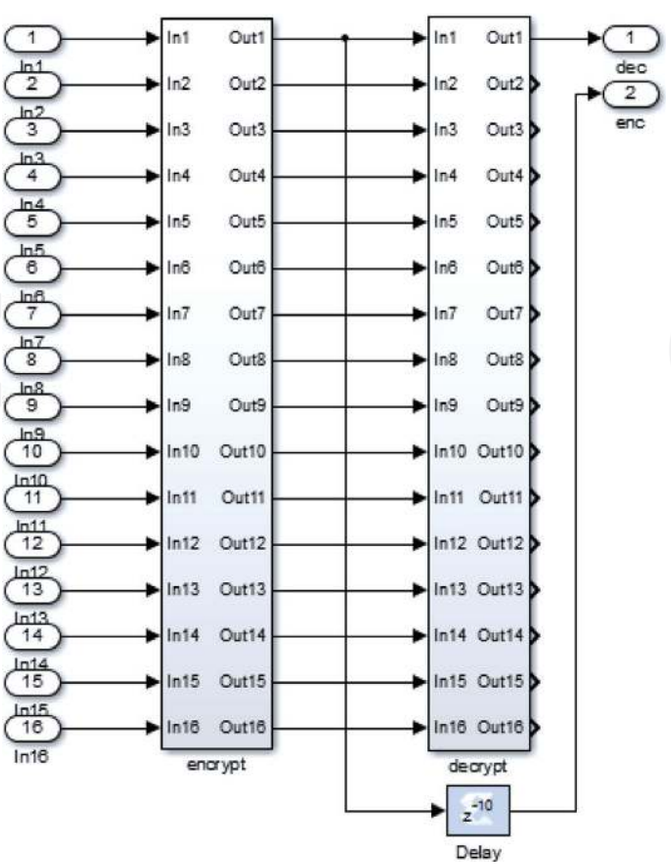

Figure 9.
System generator model for AES algorithm.

3.1 AES encryption

A plaintext of 128-bit is processed through 10 rounds. Each round contains processes like byte substitution, shift rows, mix columns, and add round key. As keys are generated using MATLAB code, only remaining system generator-based models like byte substitution, shift rows, and mix columns are discussed in this section.

Round function is one of the important processes in AES algorithm. **Figure 10** shows system generator-based model for implementing round0 function.

Round function consists of s-box, shift row, and mix column as shown in **Figure 11**.

Figure 12 shows implementation of s-box.

Figure 13 shows implementation of shift row.

Figure 14 shows implementation of mix column.

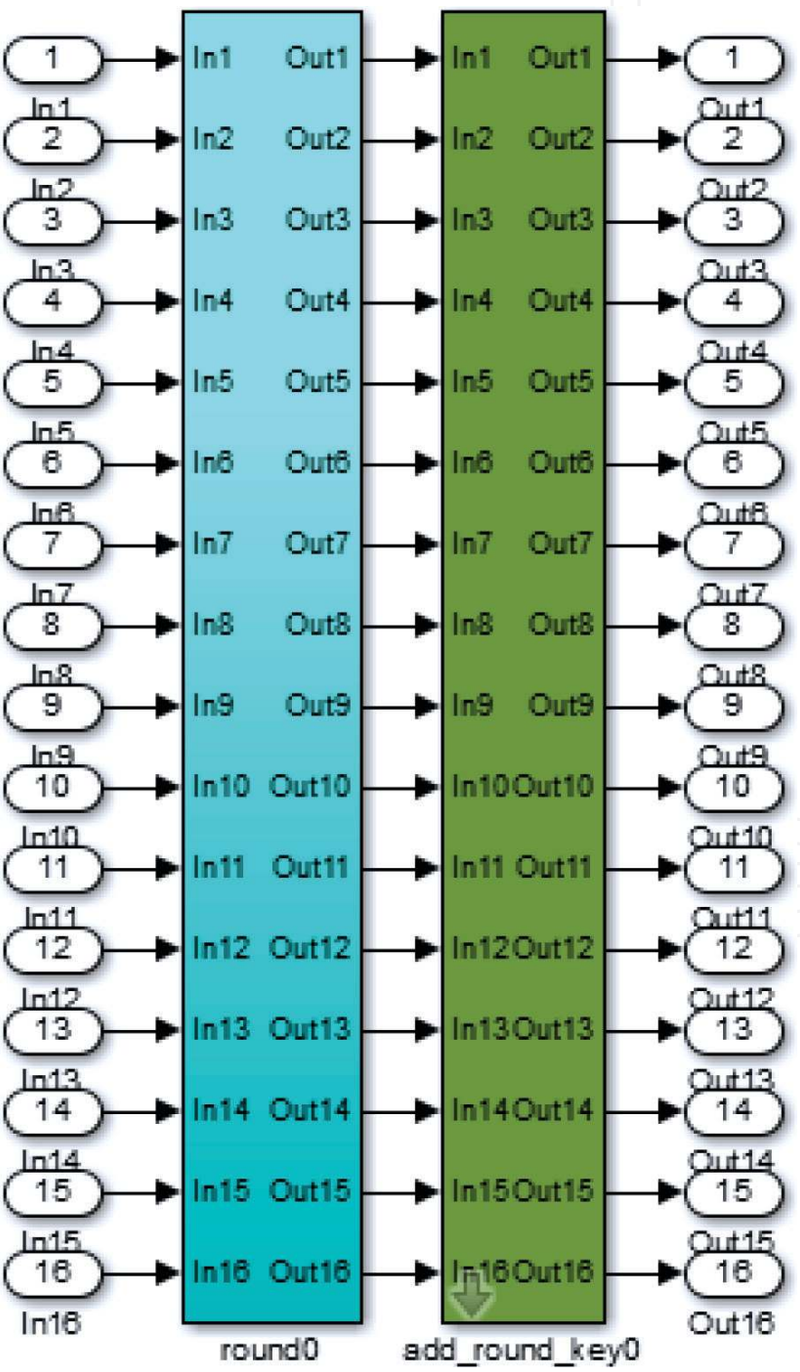

Figure 10.
System generator-based model of round function.

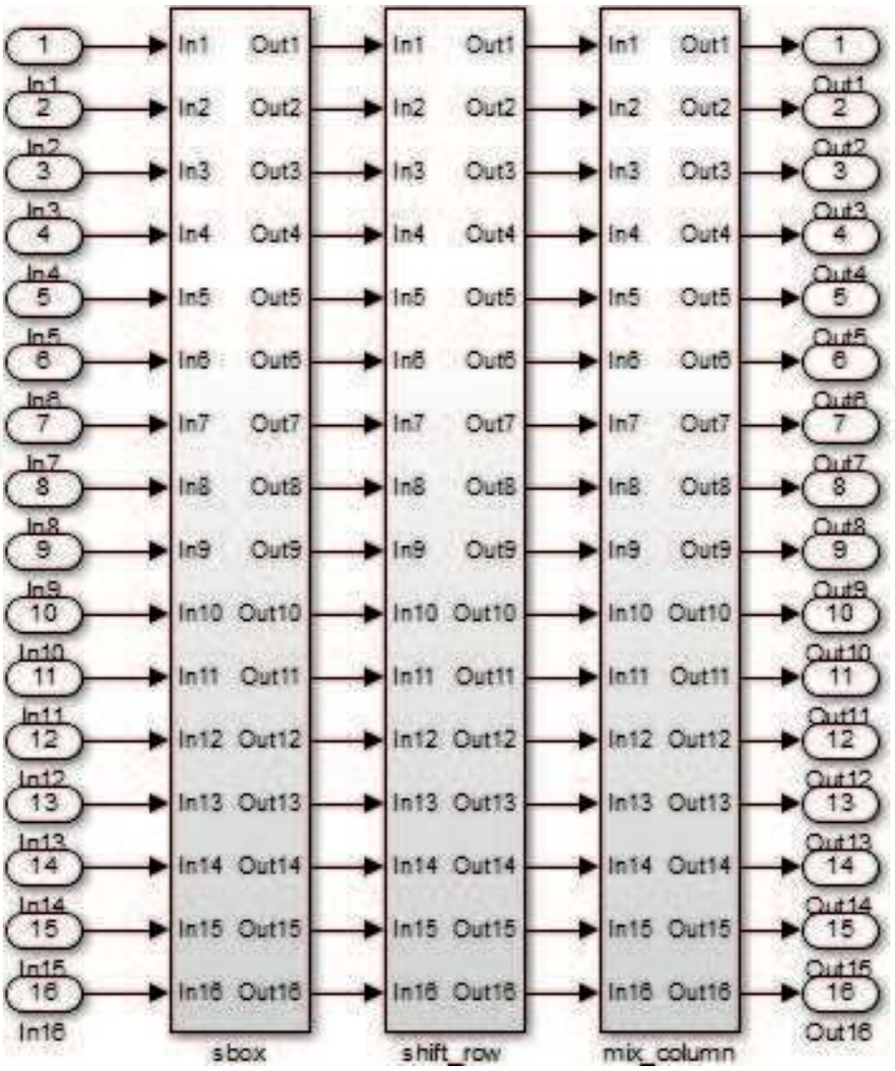

Figure 11.
Roundo.

Mix column consists of group_1, group_2, group_3, and group_4. **Figure 15** shows implementation of group. Further each group consists of four multiplication blocks such as mul_blk, mul_blk1, mul_blk2, and mul_blk3. **Figure 16** shows implementation of multiplication block.

3.2 AES decryption

A cipher text of 128-bits is processed through 10 inverse rounds. Each round contains processes like inverse byte substitution, inverse shift rows, inverse mix columns, and add round key.

Figure 17 shows implementation of inverse round function.

Inverse round function consists of inverse s-box, inverse shift row, and inverse mix column as shown in **Figure 18**.

Figure 19 shows implementation of inverse mix column.

Inverse mix column consists of four groups, i.e., group_1, group_2, group_3, and group_4. **Figure 20** shows implementation of group. Each group consists of multiplication blocks like mul_blk, mul_blk1, mul_blk2, and mul_blk3. **Figure 21** shows implementation of multiplication block.

Each multiplication block consists of three multipliers mul_2, mul_4, and mul_8 and EX-OR operations. **Figure 22** shows implementation of multipliers.

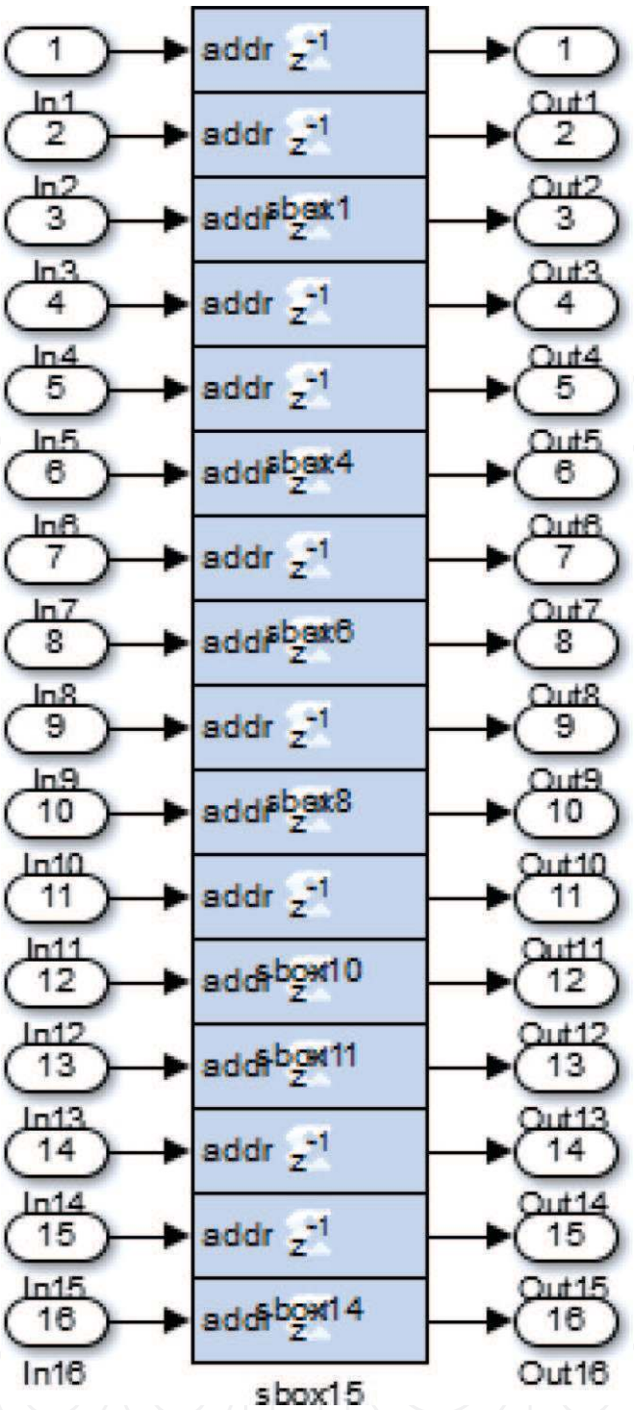

Figure 12.
Implementation of s-box.

Figure 23 shows implementation of inverse shift row.
Figure 24 shows implementation of inverse s-box.

3.3 Tools utilized

3.3.1 Software utilized

For implementing the proposed design, MATLAB 2013a and Xilinx ISE Design Suite are used. MATLAB is used for generating the keys and also to get the results in terms of images, whereas Xilinx ISE Design Suite is used to get the synthesis result, RTL schematic, and throughput of this implementation.

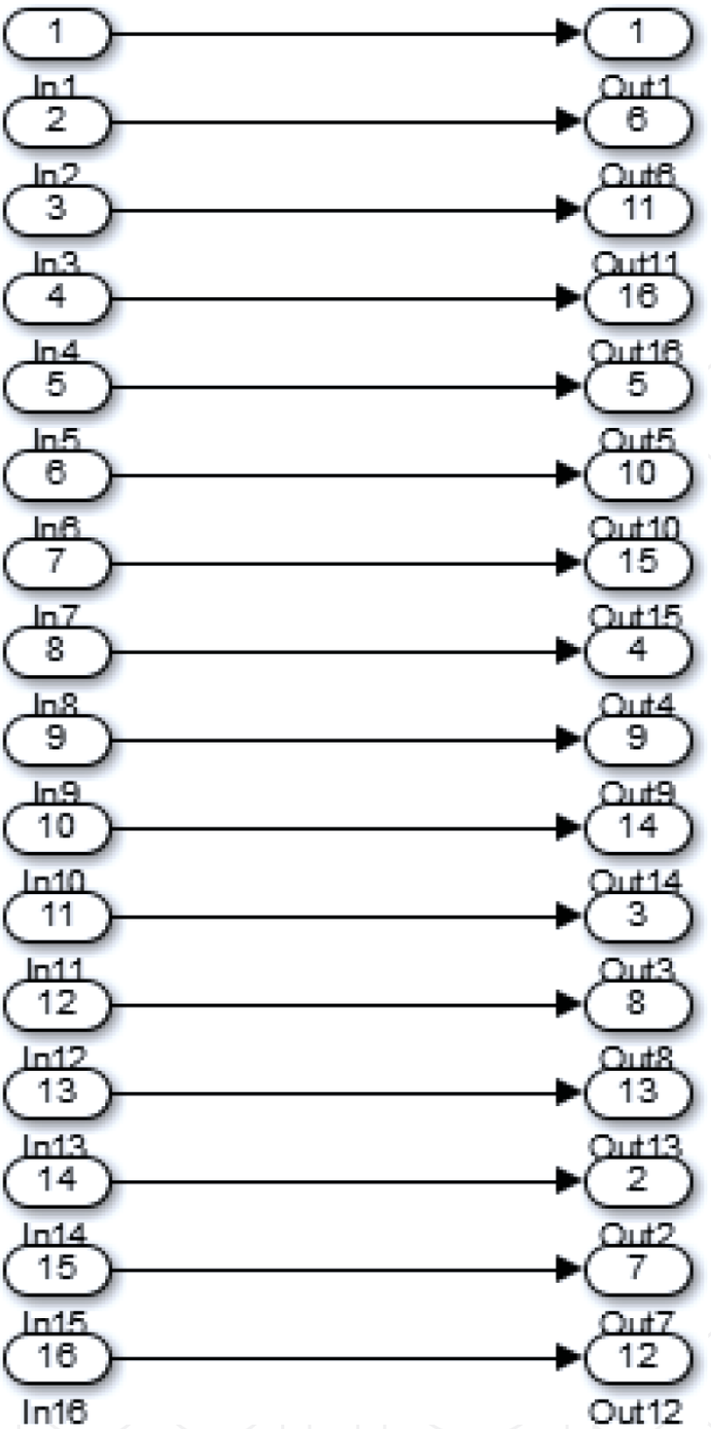

Figure 13.
Implementation of shift row.

3.3.2 Hardware utilized

Nexys-4 DDR development board is used for implementation. This board has the following features:

a. Xilinx Artix-7 FPGA XC7A100T-1CSG324C

b. 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops

c. 4860 Kbits of fast block RAM

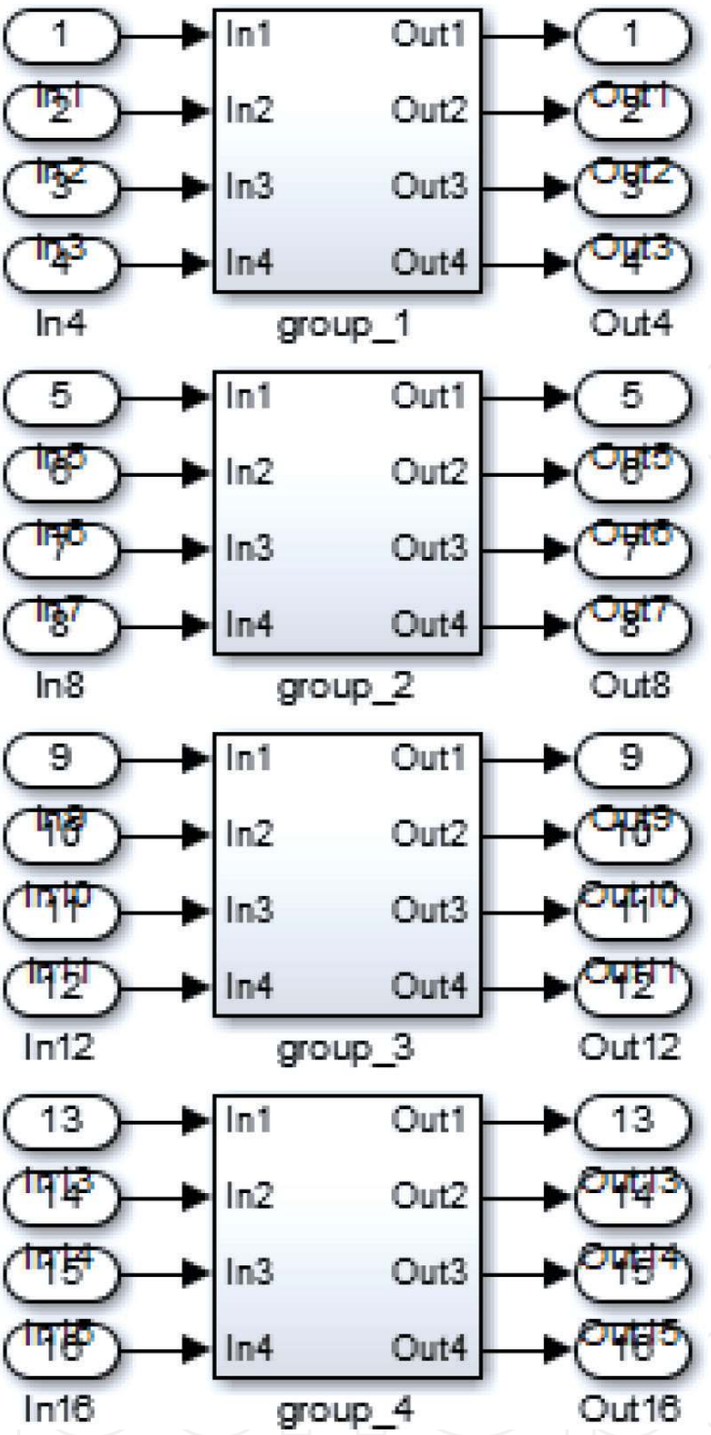

Figure 14.
Implementation of mix column.

d. Six clock management tiles, each with phase-locked loop (PLL)

e. 240 DSP slices

f. Internal clock speeds exceeding 450 MHz

g. On-chip analog-to-digital converter (XADC)

h. 128 MiB DDR2

i. Serial Flash

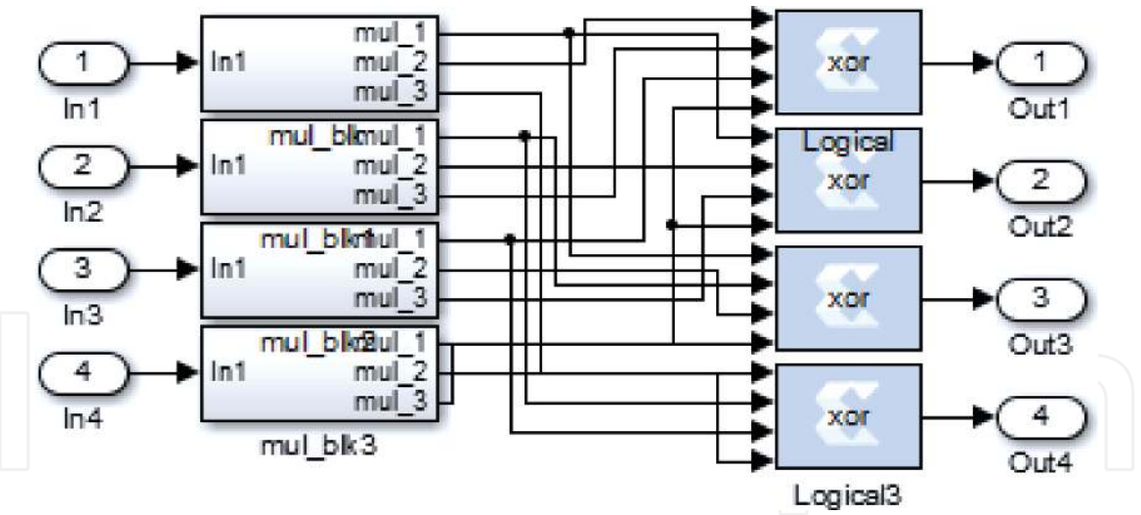

Figure 15.
Implementation of group.

j. Digilent USB-JTAG port for FPGA programming and communication

k. MicroSD card connector

l. Ships with rugged plastic case and USB cable

m. USB-UART Bridge

n. 10/100 Ethernet PHY

o. PWM audio output

p. 3-axis accelerometer

q. 16 user switches

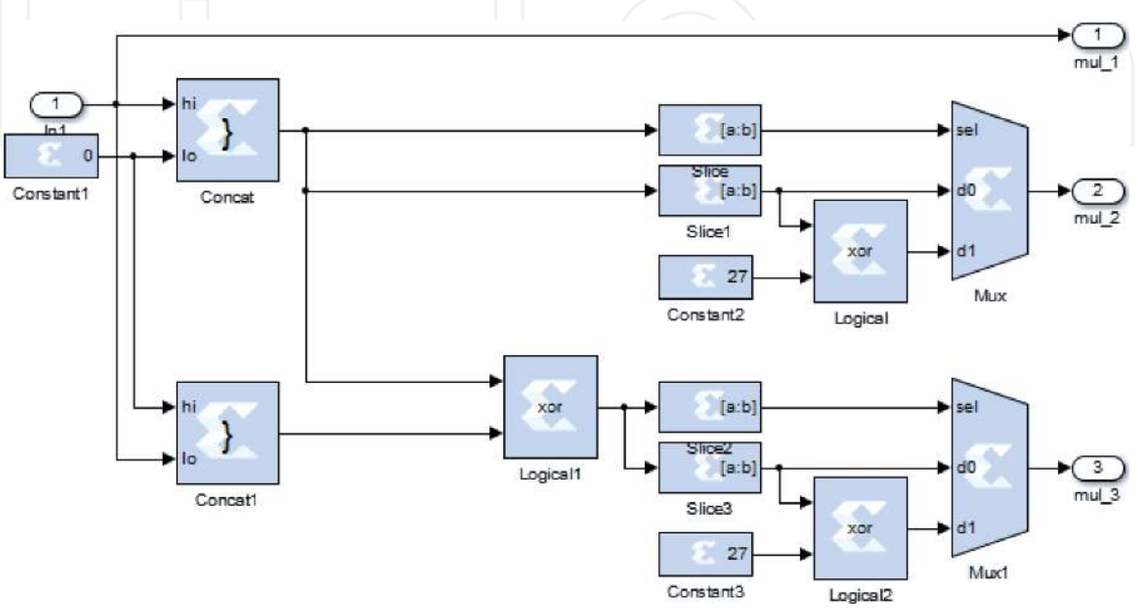

Figure 16.
Implementation of multiplication block.

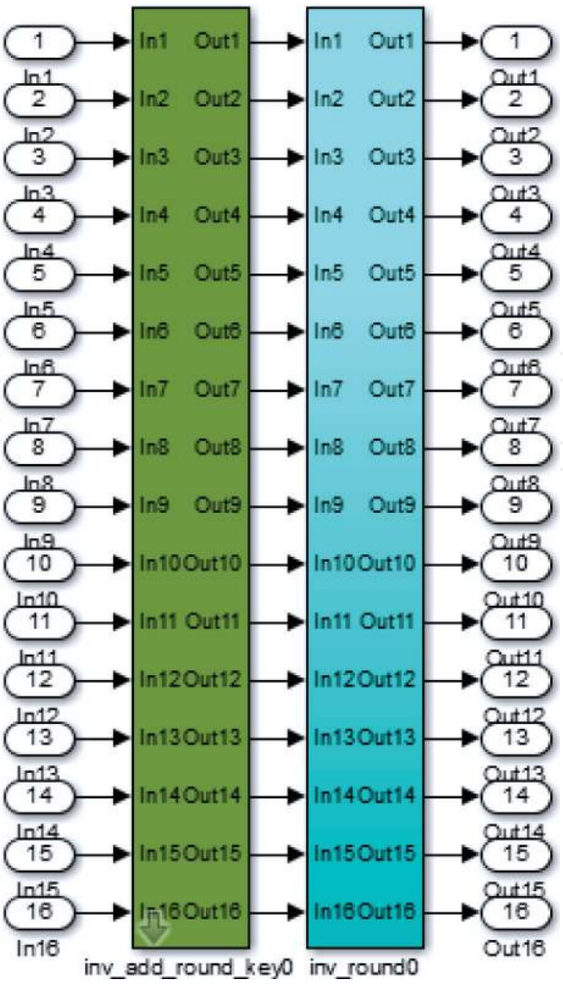

Figure 17.
System generator-based model of inverse round function.

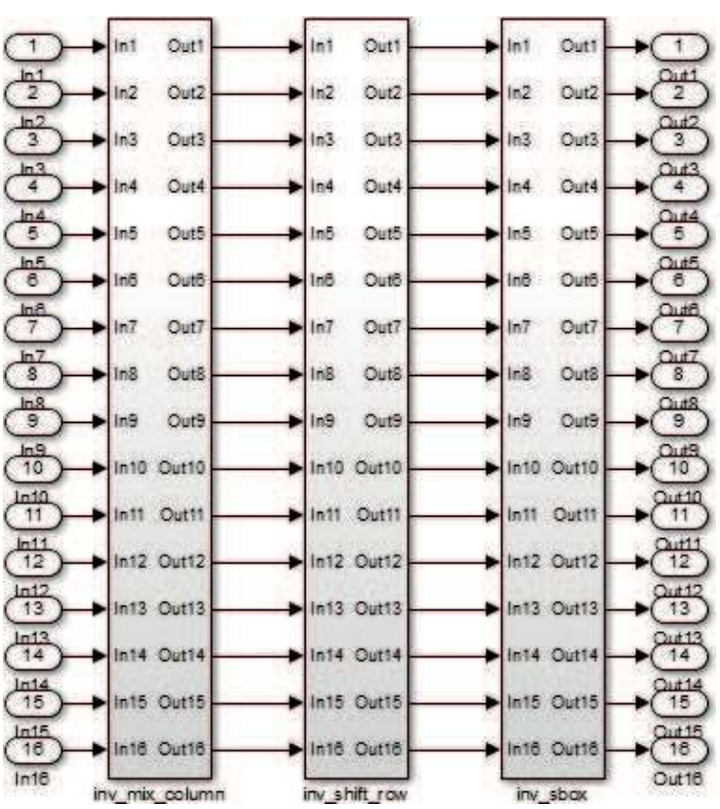

Figure 18.
Inverse roundo.

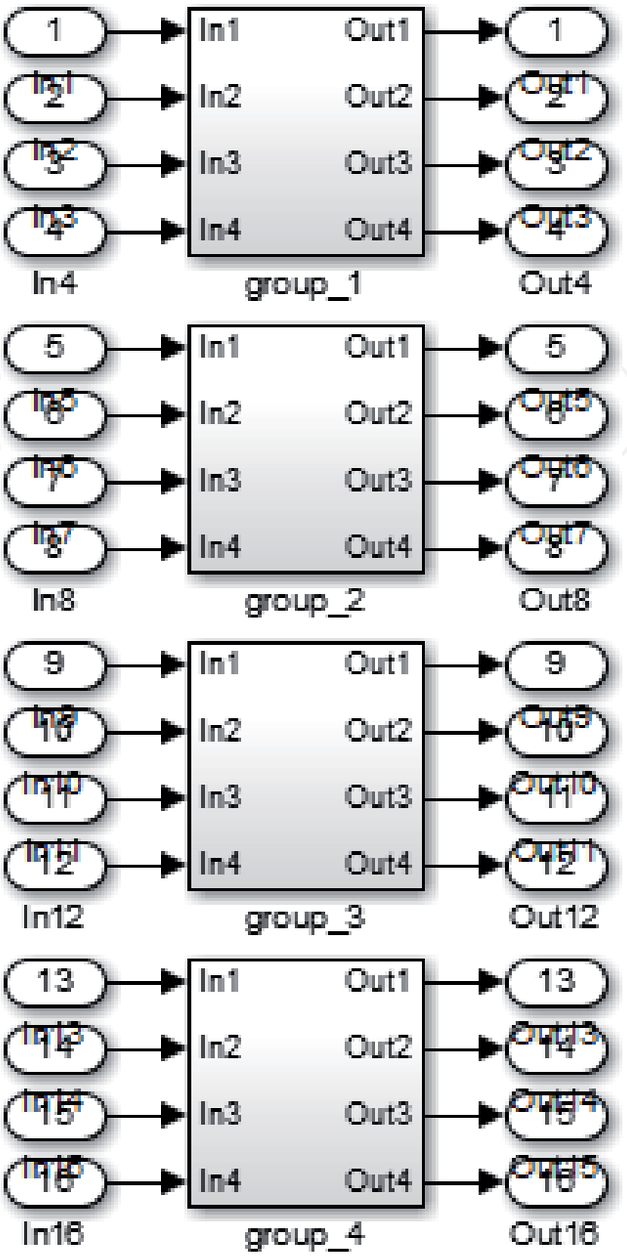

Figure 19.
Inverse mix column.

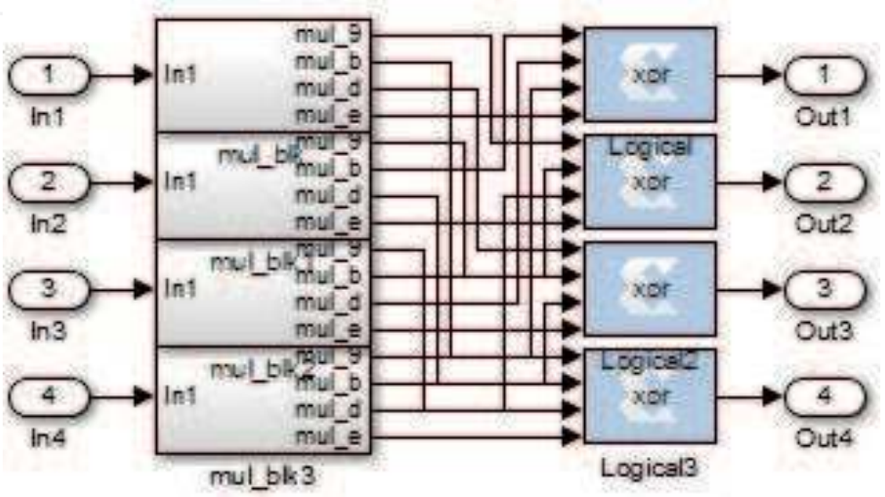

Figure 20.
Implementation of group.

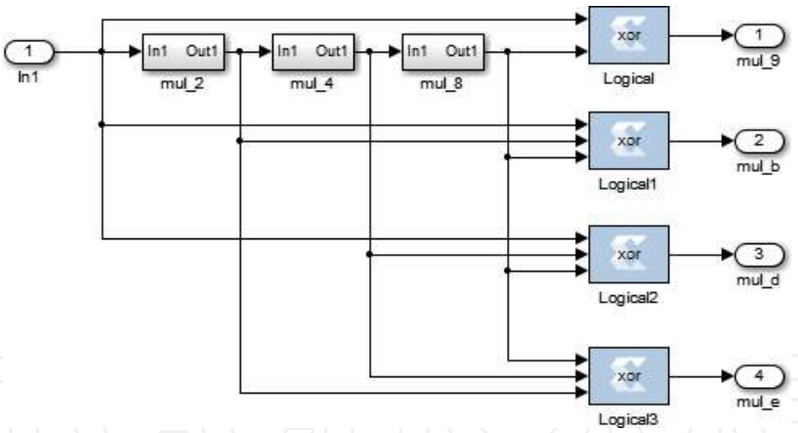

Figure 21.
Implementation of multiplication block.

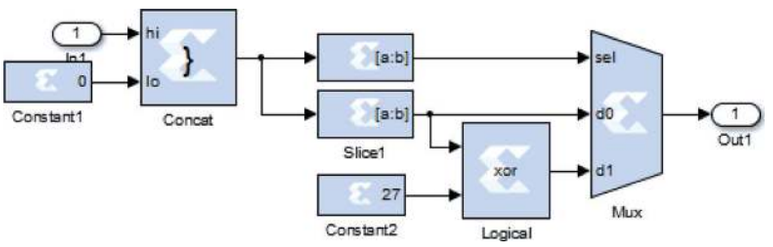

Figure 22.
Implementation of multipliers.

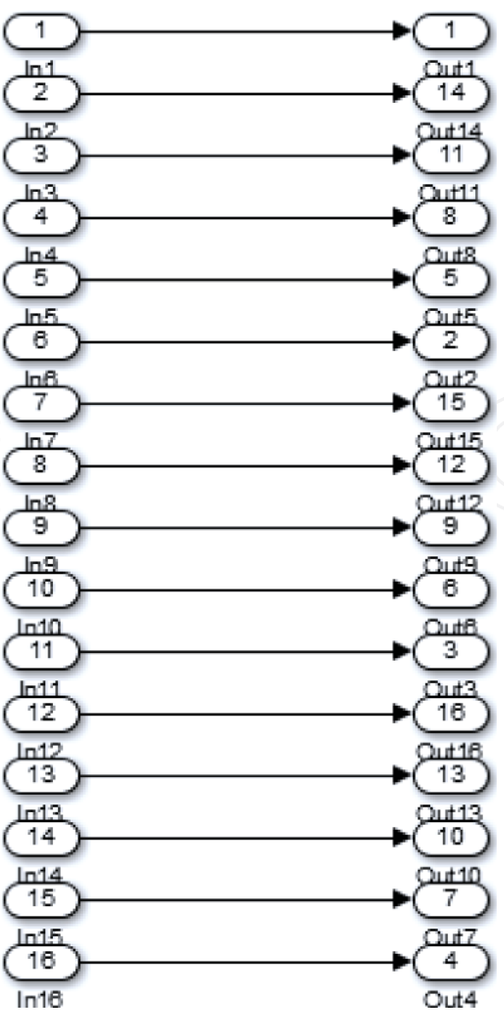

Figure 23.
Implementation of inverse shift row.

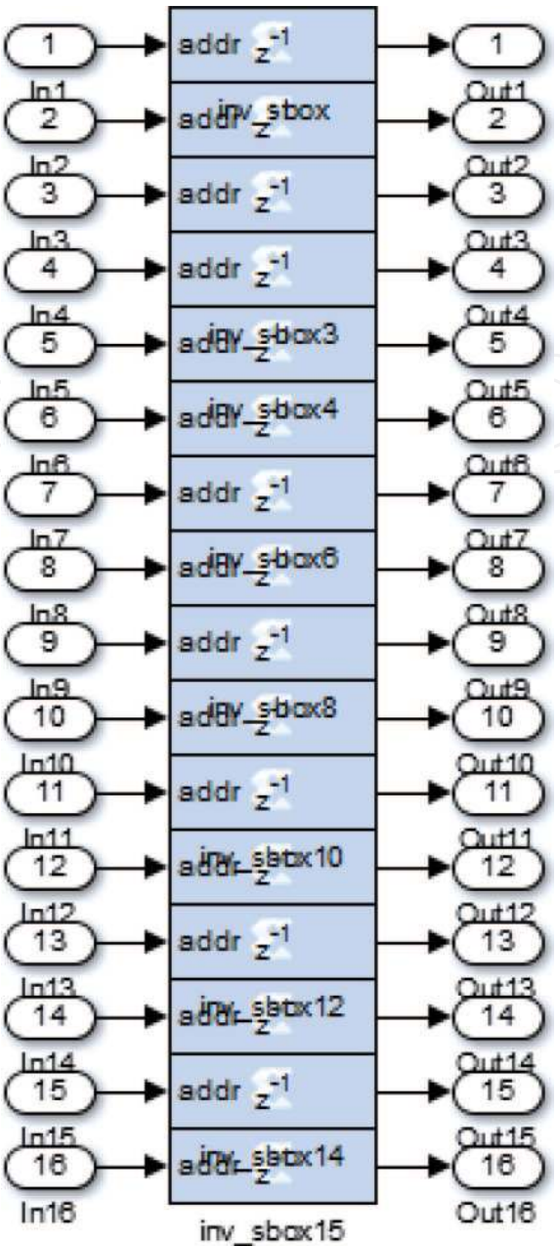

Figure 24.
Implementation of inverse s-box.

r. 16 user LEDs

s. Two tri-color LEDs

t. PDM microphone

u. Temperature sensor

v. Two 4-digit 7-segment displays

w. USB HID Host for mice, keyboards, and memory sticks

x. PMOD for XADC signals

y. 12-bit VGA output

z. Four PMOD ports

4. Experimental results

4.1 RTL schematic

Figure 25 shows detailed RTL schematic of the proposed implementation of AES algorithm.

4.2 Synthesis result

The design is synthesized using Xilinx XST synthesizer. In the proposed design, an optimized and synthesizable very high speed integrated circuit (VHSIC) hardware description language (VHDL) code for the implementation of image as well as 128-bit data encryption is developed so as to utilize less area and increase the speed. **Table 1** shows design utilization summary of the proposed design.

From the synthesis results of the proposed design, it is clear that this system utilizes only 121 slice registers, and its maximum operating frequency is 1102.536 MHz. The throughput of the system is calculated using the following formula:

$$(\text{Throughput}) \text{ of the system} = \frac{128 \text{ bits} \times \text{Clock frequency}}{\text{Cycles per Encrypted block}} \quad (1)$$

By substituting the values in Eq. (1), throughput of the systems is 14.1125 Gbps.

4.3 Simulation result

Figure 26 shows simulation result when an image is applied as an input.

4.4 Performance analysis

Performance analysis is a must to compare the performance of the proposed implementation with existing methods. The performance is compared on the basis of area and operating frequency. Till date various researchers have worked on FPGA-based implementations of AES algorithm; some of them have optimized speed and

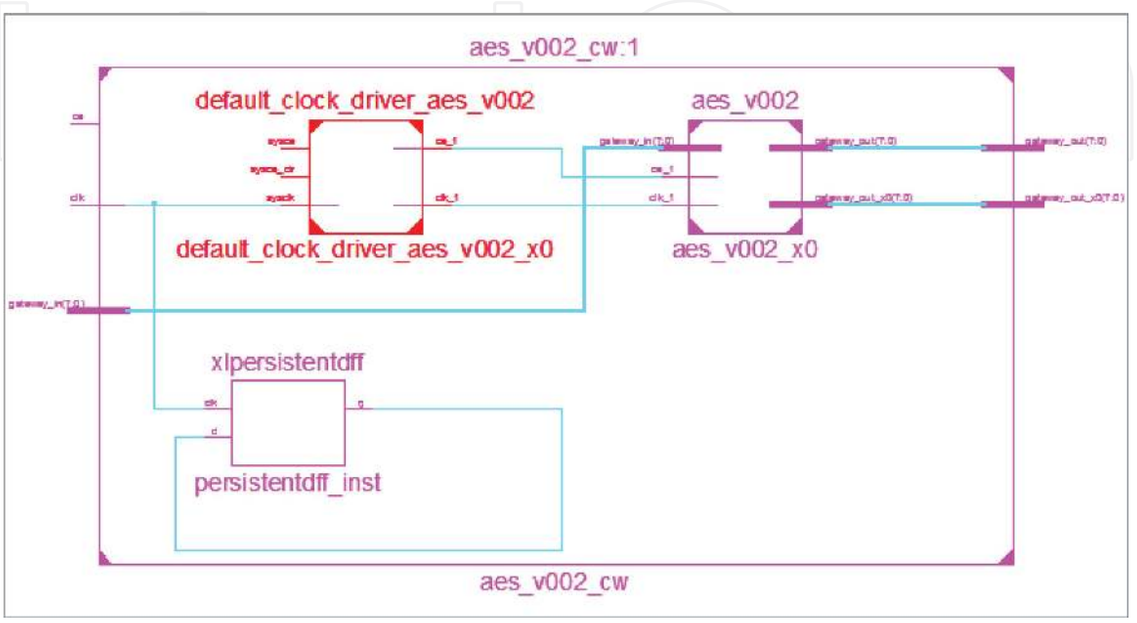

Figure 25.
Detailed RTL schematic of AES algorithm.

Design utilization summary			
Logic utilization	**Used**	**Available**	**% utilization**
Number of slice registers	121	126,800	0.00095
Number of slice LUTs	4782	63,400	7
Number of bonded IOBs	25	210	11

Table 1.
Design utilization summary.

Figure 26.
Simulation result (a) Original image, (b) Encrypted image, and (c) Decrypted image.

Sr. No.	**Authors**	**Slices**	**Operating freq. (MHz)**
1	Proposed work	121	1102.536
2	[3]	1464	—
3	[4]	161	886.64
4	[7]	5493	277.4
5	[8]	3376	—
6	[9]	150	90
7	[10]	201	70
8	[12]	1403	160.875
9	[15]	1746	—
10	[19]	1853	140.390
11	[21]	6279	119.954

Table 2.
Performance comparison of the proposed system with previous work.

some have optimized area. In the proposed system, both area and speed are optimized. **Table 2** shows performance comparison of the proposed system with previous work.

5. Conclusion

In this chapter, fast, area-efficient, and secure implementation of AES algorithm on FPGA is suggested. As per the literature survey, it is clear that Farooq and Faisal Aslam [4] achieved better performance in terms of speed, whereas Ibrahim [9] achieved better performance in terms of area. In this design, due to better Xilinx system generator-based design, the system is optimized, and it utilizes only 121 slice registers at maximum operating frequency of 1102.536 MHz. Also, throughput of the proposed system is 14.1125 Gbps.

Conflict of interest

There is no conflict of interest.

Acronyms and abbreviations

AES	advanced encryption standard
DDR	double data rate
DES	data encryption standard
FPGA	field-programmable gate array
Gbps	gigabits per second
MHz	megahertz
VHDL	VHSIC Hardware Description Language
VHSIC	very high speed integrated circuit

Author details

Altaf O. Mulani* and Pradeep B. Mane
AISSMS Institute of Information Technology, Pune, Maharashtra, India

*Address all correspondence to: aksaltaaf@gmail.com

References

[1] Mulani AO, Mane PB. Watermarking and cryptography based image authentication on reconfigurable platform. Bulletin of Electrical Engineering and Informatics. June 2017;**6**(2):181-187

[2] Ratheesh T, Narayanan S. FPGA based implementation of AES encryption and decryption with low power multiplexer LUT based S-box. IOSR Journal of Electronics and Communication Engineering. April 2017;**12**(2):57-61

[3] Agarwal A, Singh G, Sharma N. Implementation of AES algorithm. International Journal of Engineering Research and Science (IJOER). April 2016;**2**(4):112-116

[4] Farooq U, Faisal Aslam M. Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA. Journal of King Saud University-Computer and Information Sciences. March 2016;**29**(3):295-302

[5] Sai Srinivas NS, Akramuddin Md. FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption. In: IEEE International Conference on Electrical, Electronics and Optimization Techniques; March 2016

[6] Mathur N, Bansode R. AES based text encryption using 12 rounds with dynamic key selection. In: International Conference on Communication, Computing and Virtualization. Elsevier; 2016

[7] Kalaiselvi K, Mangalam H. Power efficient and high performance VLSI architecture for AES algorithm. Journal of Electrical Systems and Information Technology. September 2015;**2**(2):178-183

[8] Deshpande HS, Karande KJ, Mulani AO. Area optimized implementation of AES algorithm on FPGA. In: IEEE International Conference on Communications and Signal Processing (ICCSP); April 2015

[9] Ibrahim A. FPGA based hardware implementation of compact AES encryption hardware core. WSEAS Transactions on Circuits and Systems. 2015;**14**:364-371

[10] Khose PN, Raut VG. Implementation of AES algorithm on FPGA for low area consumption. In: IEEE International Conference on Pervasive Computing (ICPC); January 2015

[11] Mulani AO, Mane PB. Area optimization of cryptographic algorithm on less dense reconfigurable platform. In: IEEE International Conference on Smart Structures and Systems (ICSSS); October 2014

[12] Yewale Minal J, Sayyad MA. Implementation of AES on FPGA. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP). October 2014;**4**(5):65-69

[13] Deshpande HS, Karande KJ, Mulani AO. Efficient implementation of AES algorithm on FPGA. In: IEEE International Conference on Communications and Signal Processing (ICCSP); April 2014

[14] Tonde AR, Dhande AP. Review paper on FPGA based implementation of advanced encryption standard (AES) algorithm. International Journal of Advanced Research in Computer and Communication Engineering. January 2014;**3**(1):4878-4880

[15] Varhade SA, Kasat NN. Implementation of AES algorithm using FPGA and its performance analysis. International Journal of Science and Research. May 2013;**4**(5):2484-2492

[16] Wadi SM, Zainal N. Rapid encryption method based on AES algorithm for Grey scale HD image encryption. In: International Conference on Electrical Engineering and Informatics. Elsevier; 2013

[17] Wang W, Chen J, Xu F. An implementation of AES algorithm based on FPGA. In: IEEE International Conference on Fuzzy Systems and Knowledge Discovery; May 2012

[18] Shylashree N, Bhat N, Shridhar V. FPGA implementations of advanced encryption standard: A survey. International Journal of Advances in Engineering and Technology (IJAET). May 2012;**3**(2):265-285

[19] Borkar AM, Kshirsagar RV, Vyawahare MV. FPGA implementation of AES algorithm. In: IEEE International Conference on Electronics Computer Technology (ICECT);April 2011

[20] Deshpande AM, Deshpande MS, Kayatanavar DN. FPGA implementation of AES encryption and decryption. In: IEEE International Conference on Control, Automation, Communication and Energy Conservation; June 2009

[21] Swinder K, Vig R. Efficient implementation of AES algorithm in FPGA device. In: IEEE International Conference on Computational Intelligence and Multimedia Applications; December 2007

[22] Samanta S. FPGA Implementation of AES Encryption and Decryption. Surat: Sardar Vallabhbhai National Institute of Technology; 2007

[23] Good T, Benaissa M. AES on FPGA from fastest to smallest. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded systems. Springer; 2005