JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

AES Algorithm Using VHDL

Asawari Ujjainkar, Prof. A.B. Kharate
Department of Electronics and Tele-communication
Amravati University, Amravati, India

Abstract: Advanced Encryption Standard (AES), a Federal Information Processing Standard (FIPS), is an approved cryptographic algorithm that is used to protect electronic data. The large and growing number of internet and wireless communication users has led to an increasing demand of security measures and devices for protecting the user data transmitted over the unsecured network so that unauthorized persons cannot access it. As we share the data through wireless network it should provide data confidentiality, integrity and authentication. The symmetric block cipher plays a major role in the bulk data encryption. One of the best existing symmetric security algorithms to provide data security is advanced encryption standard (AES). AES has the advantage of being implemented in both hardware and software. Implementation of the AES has lot of advantage such has increased throughput and better security level.

IndexTerms- AES, Encryption, Decryption, Security, FPGA, VHDL.

1. Introduction

In today's world most of the communication is done using electronic media. Data Security plays a vital role in such communication. Increasing need of data handle in Computer Network and Communication Technology capable to great mass of data and information need to be exchanged by public communication networks. So cryptography is constantly increasing sensitive data is more vulnerable from automated spying and high efficiency and safety of Data transmission Hence, there is a need to protect data from malicious attacks.

Each day millions of users generate and interchange large volumes of information in various fields, such as financial and legal files, medical reports and bank services via Internet. These and other examples of applications deserve a special treatment from the security point of view, not only in the transport of such information but also in its storage. In this sense, cryptography techniques are especially applicable. This implementation will be useful in wireless security like military communication and mobile telephony where there is a greater emphasis on the speed of communication

Cryptography is closely related to the disciplines of cryptology and cryptanalysis. Cryptography includes techniques such as microdots, merging words with images, and other ways to hide information in storage or transit. More generally, it is about constructing and analyzing protocols that block adversaries, various aspects in information security such as data confidentiality, data integrity, authentication, and non-repudiation are central to modern cryptography. Modern cryptography exists at the intersection of the disciplines of mathematics, computer science, and electrical engineering. Applications of cryptography include ATM cards, computer passwords, and electronic commerce. However, in today's computer-centric world, cryptography is most often associated with scrambling plaintext (ordinary text, sometimes referred to as clear text) into cipher text (a process called encryption), then back again (known as decryption). Individuals who practice this field are known as cryptographers. Modern cryptography concerns itself with the following four objectives:

1) **Confidentiality** (the information cannot be understood by anyone for whom it was unintended)
2) **Integrity** (the information cannot be altered in storage or transit between sender and intended receiver without the alteration being detected)
3) **Non-repudiation** (the creator/sender of the information cannot deny at a later stage his or her intentions in the creation or transmission of the information)
4) **Authentication** (the sender and receiver can confirm each other, identity and the origin/destination of the information).

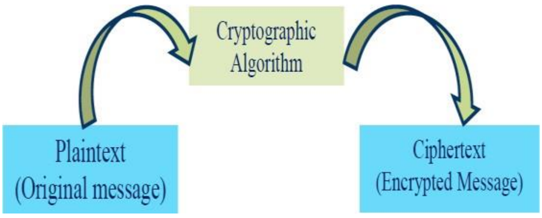


Figure 1.1: Basic step of Encryption in cryptography

The Advanced Encryption Standard which will be referred to as AES is the current industrial standard and has been in vogue since 2001. It is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. It is based on the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal which was evaluated by the NIST during the AES selection process. The AES encryption method finds extensive use in

the electronic and computational industry as most of the arithmetic operations that we generally use are not the ones used in here rather it's the ones which are highly electronic efficient and can be implemented using shift registers and exclusive OR gates which any processor is efficient in handling. This makes the complicated algorithm run very quickly and using minimal processor power and minimal hardware. AES has already received widespread use because of its high security, high performance in both hardware and software. Many implementations are done in software but it seems to be too slow for fast applications such as routers and some wireless communication systems. The various of AES hardware implementation architectures and optimizations have been suggested for different applications. 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. AES algorithm can resist any kinds of password attacks with a strong practicability in information security and reliability. AES can be implemented in software or hardware but, hardware implementation is more suitable for high speed applications in real time.

AES is founded on solid and well-published mathematical ground, and appears to resist all known attacks well. It has been published for a long time and has been the subject of intense scrutiny by researchers all over the world, thus indicating that in fact no back-door or known weakness exists. Enormous amounts of economic value and information is successfully protected by AES. It is endorsed by National Security Agency (NSA) and is the first open cipher which is easy to approach publicly.

On the current situation of researching at home and abroad, AES algorithm emphasizes its throughput using pipeline pattern. Its biggest advantage is to improve the system throughput, but there is a clear disadvantage that is at the cost of on-chip resources. And in accordance that AES algorithm is used in the low requirements of the terminal throughput at present, the high safety and cost-effective reduced AES system is designed and validated on the Altera, aiming at reduced hardware structure. The advantages in this system are high speed, high reliability, a smaller chip area, and high cost-effective. These will effectively promote the AES algorithm used in the terminal equipment's.

2. Related Work

Ai-Wen Luo, Qing-Ming Yi, Min Shi [1] presented for maintaining the speed of encryption, the pipelining technology is applied and the mode of data transmission is modified in this design so that the chip size can be reduced. AES encryption can be mainly divided into two parts: key schedule and round transformation. The improved structure is also divided into these two major processes. The initial key will be sent to the two modules: Key expansion and Key selection, while the plaintext is to be sent to the round transformation after the roundkey is selected. But the operand of data transmission is turned into a 32-bit unit. The initial round of encryption: The four packets of consecutive 32-bit plaintext (128 bits) have been put into the corresponding registers. Meanwhile, another four packets of consecutive 32-bit initial key (128 bits) have been put into other registers by the control of the enable clock signal. FPGA implementation of area-optimized AES algorithm which meets the actual application is proposed in this paper.

Yang Jun,Ding Jun Li, Na Guo Yixiong [2] presented the system aims at reduced hardware structure. Compared with the pipeline structure, it has less hardware resources and high cost-effective. And this system has high security and reliability. This AES system can be widely used in the terminal equipment's. AES encryption algorithm includes key expansion process and encryption process. The advantage of this design is the fact that we do not need to store the round key since they are currently calculated. BlockInput module is an interface module for data or key input. A simple controller is used here to identify that a complete of 128 bits data has been accepted. Controller module controls the Key Expander and Algori_Round block. Key Expander responsible to generate round key for every round from the initial key. Based on the algorithm specification, if we use 128-bit key, 10 rounds will be needed. Hence the Key Expander will generate 10 round keys. The implementation of Alg_Round has been designed this way that it can work as initial round, standard round, and final round. This system has the significant features such as less hardware resources, high speed, high reliability, high cost-effective.

Alia Arshad, Kanwal Aslam, Dur-e-Shahwar Kundi and Arshad Aziz [3] presented a resource efficient reconfigurable hardware implementation of Advance Encryption Standard (AES) algorithm using High Level Language (HLL) approach on Field Programmable Gate Array (FPGA) for rapid development. In this work use an approach to directly map the design described in a high level package i.e. System Generator on FPGA platforms. This approach is ideal for Encryption functions where the development of data-path architectures can easily be done to provide bit and cycle accurate models. It fills the gap between performance and flexibility by efficiently applying re-configurability. In order to attain a balance between the cost and time, an efficient method must be explored and implement for various combinations of hardware and software to realize algorithmic best solutions of different requisite [4],[5].

Wang Wei, Chen Jie, Xu Fei [6] presents to improve the safety of in data transmission. The mathematic principle, encryption process and logic structure of AES algorithm are introduced. So as to reach the propose of improving the system computing speed, the pipelining and parallel processing methods were used. The simulation results show that the high-speed AES encryption algorithm implemented correctly. Using the method of AES encryption the data could be protected effectively. In order to accomplish an encryption process, ten times of round must be iterative. This paper gives a design of AES encryption algorithm using pipeline structure and parallel processing. It is incompatible to implement the AES algorithm on hardware between the throughput and hardware resource. Different architecture should be selected according to the fields it is applied to. To make AES algorithm suitable to high-speed rate data application, we need to optimize the architecture. Meanwhile by sharing resource and eliminating common sub expression we can reduce the hardware resource utilization.

A. Amaar, I. Ashour and M. Shiple [7] presents a compact implementation of advanced encryption standard AES using different devices of FPGA technology. This implementation can be carried out through several trade-off between area and speed. Proposed architecture is implementing 128 bits data-path for both cipher key and plaintext. Proposed architecture is implementing 128 bits data-path for both cipher key and plaintext. The proposed algorithm tries to chopping the main block consuming the area SBOX, minimize in-between unwanted latches and shift registers to save area. Shift raw block is rejected and implemented by twisting the routing tracks. Mix column is implemented by combination gates. The proposed minimum area AES architecture which is described by VHDL is simulated using ModelSim to verify the functionality as a primer verification tool. Moreover, the proposed algorithm is synthesized and implemented (translate, fit, place and route) using Xilinx 6.2 [8],[9].

Adam J. Elbirt, W. Yip, B. Chetwynd, and C. Paar [10] presented the Advanced Encryption Algorithm includes efficiency testing of both hardware and software implementations of candidate algorithms. Reprogrammable devices such as field-programmable gate arrays (FPGAs) are highly attractive options for hardware implementations of encryption algorithms, as they provide cryptographic algorithm agility, physical security, and potentially much higher performance than software solutions. This contribution investigates the significance of FPGA

Implementations of the Advanced Encryption Standard candidate algorithms. Multiple architectural implementation options are explored for each algorithm. A strong focus is placed on high-throughput implementations, which are required to support security for current and future high bandwidth applications. Finally, the implementations of each algorithm will be compared in an effort to determine the most suitable candidate for hardware implementation within commercially available FPGAs. The core operations of the AES finalists were identified, and multiple architecture options were discussed [11].

William Stallings [12] AES is a block cipher intended to replace DES for commercial applications. It uses a 128-bit block size and a key size of 128, 192, or 256 bits. AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key. The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications. Compared to public-key ciphers such as RSA. The cipher takes a plaintext block size of 128 bits, or 16 bytes. The key length can be 16, 24, or 32 bytes (128, 192, or 256 bits). The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.

Alex Panato, Marcelo Barcelos, Ricardo Reis [13] This work presents an IP of the Rijndael encryption algorithm, the new Advanced Encryption Standard (AES) approved by the National Institute of Standards and Technology (NIST). The IP uses a VHDL description optimized to Altera devices. This Rijndael implementation runs its symmetric cipher algorithm using a key with 128 bits. This mode is called AES128. The paper presents the Rijndael basic structures, the AES128 architecture and results of throughput and device utilization in Altera devices.

Atul M. Borkar, Dr. R. V. Kshirsagar and Mrs. M. V. Vyawahare [14] presented the Advanced Encryption Standard can be programmed in software or built with pure hardware. However Field Programmable Gate Arrays (FPGAs) offer a quicker, more customizable solution. This research investigates the AES algorithm with regard to FPGA and the Very High Speed Integrated Circuit Hardware Description language (VHDL). Software is used for simulation and optimization of the synthesizable VHDL code. All the transformations of both Encryptions and Decryption are simulated using an iterative design approach in order to minimize the hardware consumption [15].

3. Proposed Work

3.1 Advanced Encryption Standard

The National Institute of Standards and Technology (NIST) announced that **Rajndael** pronounced as “**Rain Doll**” planned by two Belgium researchers Joan Daemen and Vincent Rijment was adopted as Advanced Encryption Standard (AES) for encryption and decryption of blocks of data. The draft is published in December 2001, under the name as FIPS-197 (Federal Information Processing Standard number 197). The criteria defined by selecting AES fall into three areas Security, Implementation and cost of the algorithm. The main emphasis was the security of the algorithm to focus on resistance of cryptanalysis attacks, implementation cost should be less so it can be used for small devices like smart cards. The AES algorithm is a private key block cipher. It encrypts data of block size 128 bits. It uses key sizes, 128 bits. AES uses three different types of round operations. One of the main features of AES is simplicity that is achieved by repeatedly combining substitution and permutation computations at different rounds. That is, AES encrypts/decrypts a 128-bit plaintext/cipher text by repeatedly applying the same round transformation a number of times depending on the key size. Advanced Encryption Standard (AES) algorithm not only for security but also for great speed. Advanced Encryption Standard not only assures security but also improves the performance in a variety of settings such as smartcards, hardware implementations etc. AES is federal information processing standard and there are currently no known non-brute-force direct attacks against AES.AES is strong enough to be certified for use by the US government for top secret information.

Features of AES Encryption Algorithm

- Advanced Encryption Standard (AES) algorithm works on the principle of Substitution Permutation network.

- AES doesn't use a Feistel network and is fast in both software and hardware.
- AES operates on a 4×4 matrix of bytes termed as a state
- The Advanced Encryption Standard cipher is specified as a number of repetitions of transformation sounds that convert the input plaintext into the final output of cipher text.
- Each round consists of several processing steps, including one that depends on the Encryption key.
- A set of reverse rounds are applied to transform cipher text back into the original plaintext using the same encryption key.

The AES is an iterative algorithm and uses four operations in different rounds, namely SubBytes, ShiftRows, MixColumns and Key Additions transformations as shown in fig.3.1 SubBytes transformation is done through S-box. S-box is the vital component in the AES architecture that decides the speed/throughput of the AES.

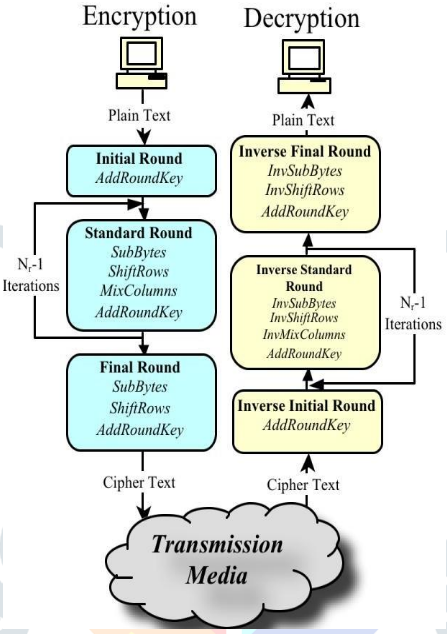


Figure 3.1 : AES Algorithm

3.2 AES Encryption

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 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. [13] The first and last rounds differ from other rounds in that there is an additional AddRoundKey transformation at the beginning of the first round and no MixCoulmns transformation is performed in the last round. Fig.(a) shows simple encryption process in which conversion of plain text to cipher text is done by using key. In this paper, we use the key length of 128 bits (AES-128) as a model for general explanation, AES encryption as shown in Fig. 3.2 consists of four operations as follows.

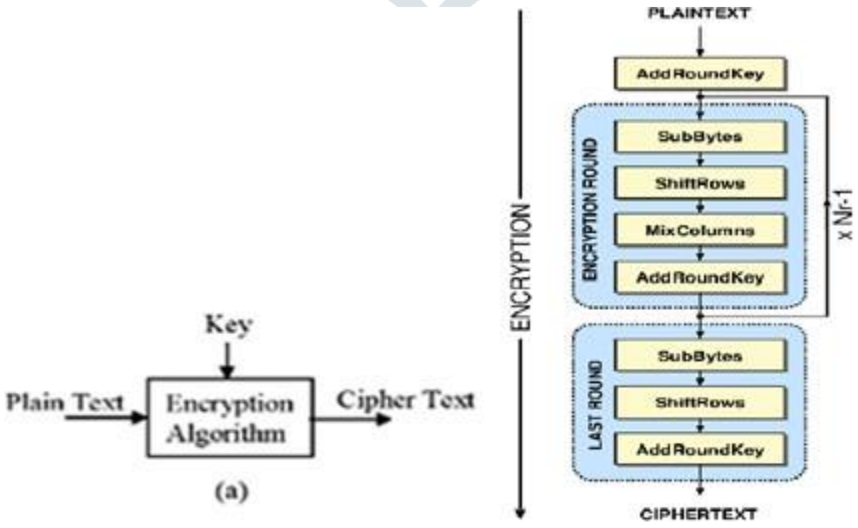


Figure 3.2: AES Encryption.

- **SubBytes Transformation:** SubBytes transformation is a non-linear byte substitution, operating on each of the state bytes independently. The SubBytes transformation is done using a once- precalculated substitution table called S-box. That S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values as shown in fig. This approach has the significant advantage of performing the S-box computation in a single clock cycle, thus reducing the latency and avoids complexity of hardware implementation as shown in fig.3.2.1.

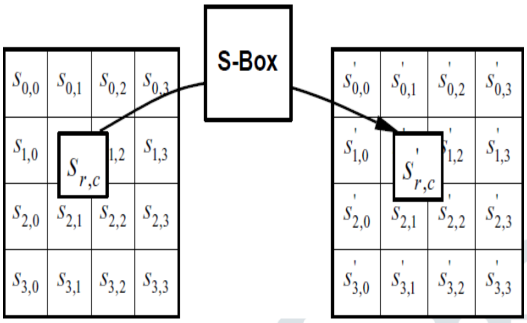

S-Box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{r,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{r,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 3.2.1: SubBytes Transformation

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

- **ShiftRows Transformation:** In ShiftRows transformation, the rows of the state are cyclically left shifted over different offsets. 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. Fig.3.2.2 shows ShiftRows Transformation.

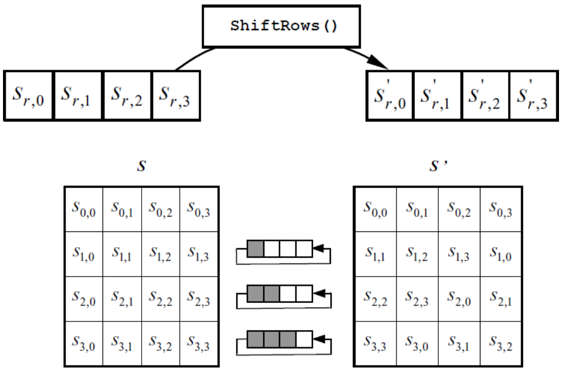

ShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
S
S'
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,1}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Figure 3.2.2: ShiftRows Transformation.

MixColumns Transformation : In MixColumns transformation, the columns of the state are considered as polynomials over GF (2^8) and multiplied by modulo x4 + 1 with a fixed polynomial c(x), given by: c(x)={03}x3 + {01}x2 + {01}x + {02}. Fig.3.2.3 shows MixColumn Transformation.

$$\begin{bmatrix} s'_{0,c}\\ s'_{1,c}\\ s'_{2,c}\\ s'_{3,c}\end{bmatrix} = \begin{bmatrix}02&03&01&01\\01&02&03&01\\01&01&02&03\\03&01&01&02\end{bmatrix}\begin{bmatrix} s_{0,c}\\ s_{1,c}\\ s_{2,c}\\ s_{3,c}\end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

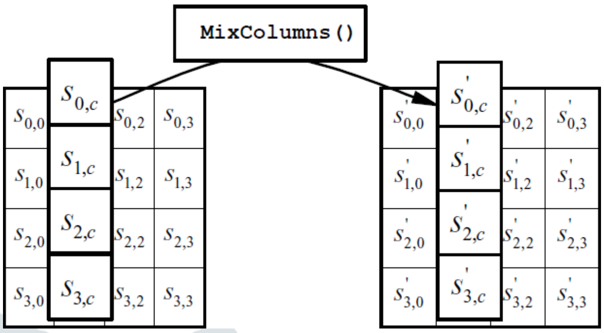

MixColumns()
$s_{0,0}$ $s_{0,c}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,2}$ $s'_{3,3}$

Fig. 3.2.3: MixColumn Transformation

- **AddRoundKey Transformation:** In the AddRoundKey transformation, a Round Key is added to the State - resulted from the operation of the MixColumns transformation - by a simple bitwise XOR operation. The RoundKey of each round is derived from the main key using the KeyExpansion algorithm. The encryption/ decryption algorithm needs eleven 128-bit RoundKey, which are denoted RoundKey [0] RoundKey [10].Fig. 3.2.4 shows AddRoundKey Transformation.

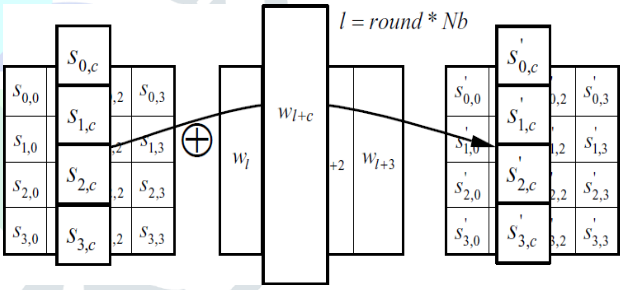
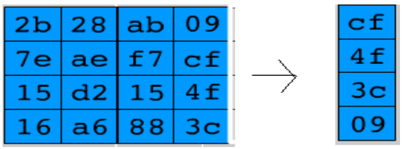
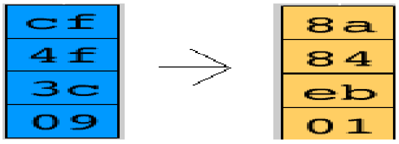

$l = round * Nb$
$s_{0,0}$ $s_{0,c}$ $s_{0,3}$
$s_{1,0}$ $s_{1,c}$ $s_{1,3}$
$s_{2,0}$ $s_{2,c}$ $s_{2,3}$
$s_{3,0}$ $s_{3,c}$ $s_{3,3}$
⊕
w_l
w_{l+c}
w_{l+3}
$s'_{0,0}$ $s'_{0,c}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,c}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,c}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,c}$ $s'_{3,3}$

Figure 3.2.4: AddRoundkey Transformation

- **Key Expansion:** The key expansion term describes the operation of generating all Round Keys from the original input key. The initial round key is original key in case of encryption and in case of decryption the last group of the generated by key expansion keys will be original keys. As mentioned earlier this initial round key will be added to the input firstly before starting the encryption or decryption iterations. The 128 bits key size, 10 groups of round keys will be generated with 16 bytes size. The round keys are generated word by word. The algorithm for generating the 10 rounds of the round key is as follows: The 4th column of the i-1 key is rotated such that each element is moved up one row.

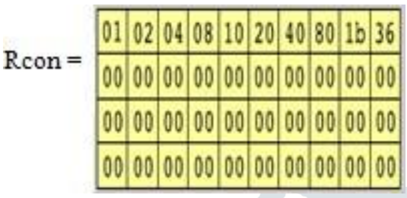


It then puts this result through a forwards Sub Box algorithm which replaces each 8 bits of the matrix with a corresponding 8-bit value from S-Box.



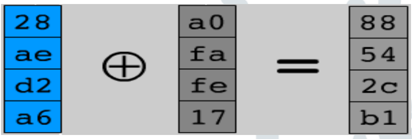
cf	→	8a
4f		84
3c		eb
09		01

To generate the first column of the ith key, this result is XOR-ed with the first column of the i-1th key as well as a constant (Row constant or Rcon) which is dependent on i.



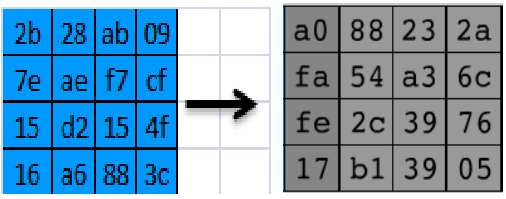
Rcon =	01	02	04	08	10	20	40	80	1b	36
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00
	00	00	00	00	00	00	00	00	00	00

The second column is generated by XOR-ing the 1st column of the ith key with the second column of the i- 1th key.



28	⊕	a0	=	88
ae		fa		54
d2		fe		2c
a6		17		b1

This continues iteratively for the other two columns in order to generate the entire ith key.



2b	28	ab	09			→	a0	88	23	2a
7e	ae	f7	cf				fa	54	a3	6c
15	d2	15	4f				fe	2c	39	76
16	a6	88	3c				17	b1	39	05

Additionally this entire process continues iteratively for generating all 10 keys. As a final note, all of these keys are stored statically once they have been computed initially as the ith key generated is require for the (10-i)th round of decryption.

3.3 AES Decryption

Decryption is a reverse of encryption which inverse round transformations to computes out the original plaintext of an encrypted cipher-text in reverse order. Fig (b) shows decryption process in which simple conversion of cipher text to plain text is done with the help of key. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively. AES decryption as shown in fig 3.3.

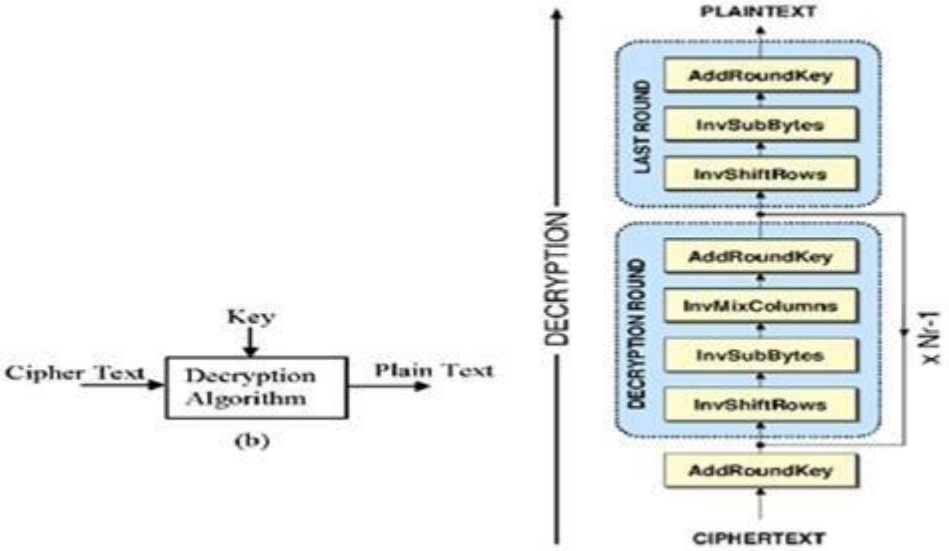
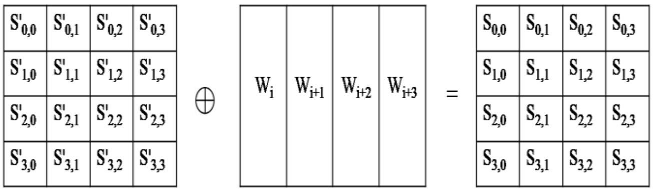


Figure 3.3: AES Decryption

- **AddRoundKey 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. Fig. 3.3.1 shows AddRoundkey Transformation.



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

Figure 3.3.1: AddRoundKey Transformation

- **InvMixColumn Transformation:** InvMixColumns transformation is done using polynomials of degree less than 4 over GF(2^8), which coefficients are the elements in the columns of the state, are multiplied modulo ($x4 + 1$) by a fixed polynomial $d(x) = \{0b\}x3 + \{0d\}x2 + \{09\}x + \{0e\}$, where {0*b*}, {0*d*}; {09}, {0*e*} denote hexadecimal values. Fig. 3.3.2 shows InvMixColumns Transformation.

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < Nb.$$

Figure 3.3.2: InvMixColumn Transformation

- **InvSubBytes Transformation:** The InvSubBytes transformation is done using a once precalculated substitution table called Inv S-box. That Inv S-box table contains 256 numbers (from 0 to 255) and their corresponding values is presented in Table II Fig.3.3.3 shows InvSubBytes Transformation.

128 Bit Plaintext (Input)	00112233445566778899aabbccddeeff
128 Bit Encryption Key (Input)	000102030405060708090a0b0c0d0e0f
128 Bit Cipher text (Output)	**69c4e0d86a7b0430d8cdb78070b4c55a**

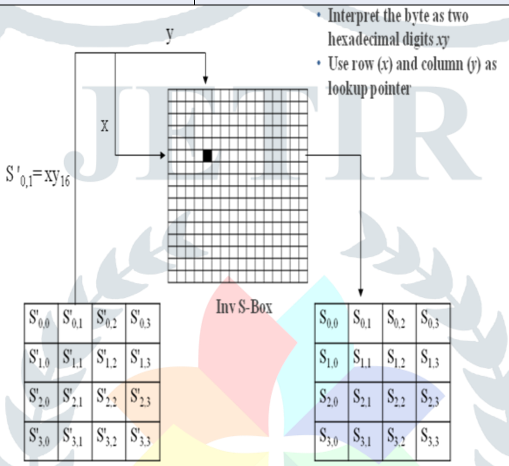


Figure 3.3.3: InvSubBytes Transformation

Table 2: Inv S- 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

- **InvShiftRows Transformation:** 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.Fig.3.3.4 shows InvShiftRows Transformation.

Figure 3.3.4: InvShiftRows Transformation

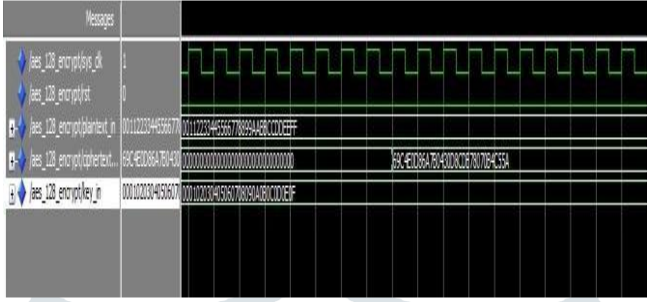
4. Result & Discussion

This chapter present the test environment and the experimental results of design modules. The objectives of this project are to design and implement the AES Algorithm to improve speed performance and throughput, reduction area. The implementation of AES Algorithm includes Encryption and Decryption in simulation Environment which was an interesting task to design that module.

4.1 Simulation of Encryption

The Table III shows the inputs and output of the Encryption. In AES Encryption 128 bit plain text and encryption key given as an input, and getting 128 bit cipher text as an output. The simulation waveform is shown in fig. 4.1.

Table 3: AES Encryption Plain text, Key, Cipher text.

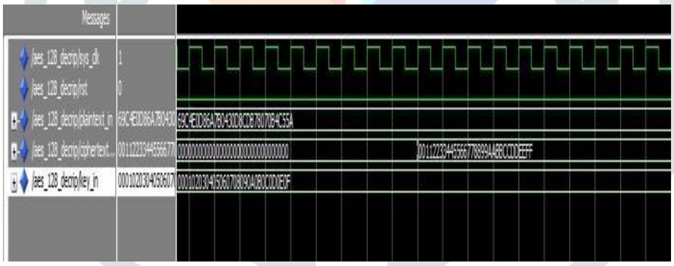


```
Messages
/aes_128_encrypt/sys_clk          1
/aes_128_encrypt/rst              0
/aes_128_encrypt/plaintext_in     00112233445566778   00112233445566778899AABBCCDDEEFF
/aes_128_encrypt/ciphertext...    69C4E0D86A7B0430    00000000000000000000000000000000   69C4E0D86A7B0430D8CDB78070B4C55A
/aes_128_encrypt/key_in           0001020304050607    000102030405060708090A0B0C0D0E0F
```

Figure 4.1: Simulation of Encryption

Table 5: AES Decryption Plain text , Key , Cipher text.

128 Bit Decryption Plaintext (Input)	69c4e0d86a7b0430d8cdb78070b4c55a
128 Bit Decryption Key (Input)	000102030405060708090a0b0c0d0e0f
128 Bit Cipher text (Output)	**00112233445566778899aabbccddeeff**

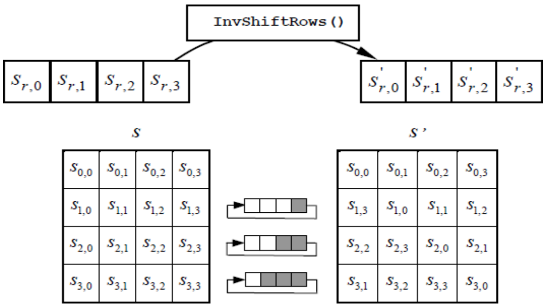


```
Messages
/aes_128_decrip/sys_clk           1
/aes_128_decrip/rst               0
/aes_128_decrip/plaintext_in      69C4E0D86A7B0430    69C4E0D86A7B0430D8CDB78070B4C55A
/aes_128_decrip/ciphertext...     00112233445566778   00000000000000000000000000000000   00112233445566778899AABBCCDDEEFF
/aes_128_decrip/key_in            0001020304050607    000102030405060708090A0B0C0D0E0F
```

Figure 4.2: Simulation of AES Decryption.

5. Applications

1) It can be used for security of Smart cards, wireless sensor networks, wireless mesh networks.
2) AES have high computational efficiency, so as to be usable in high speed applications, Such as broad band links.
3) AES is very well suited for restricted-space environments where either encryption or decryption is implemented. It has very low RAM and ROM requirements.
4) Web servers that need to handle many encryption sessions.
5) Any kind application where security is needed for our current cryptosystems.


InvShiftRows()
$s_{r,0}$ $s_{r,1}$ $s_{r,2}$ $s_{r,3}$
$s'_{r,0}$ $s'_{r,1}$ $s'_{r,2}$ $s'_{r,3}$
s
s'

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

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

6. Future Scope

Advanced Encryption Standard (AES) is the most secure symmetric encryption technique that has gained worldwide acceptance. The AES is an efficient cryptographic technique that includes generation of ciphers for encryption and inverse ciphers for decryption. Higher security and speed of encryption/decryption is ensured by operations like Sub Bytes (S-box). Sub Bytes and Key Scheduling. Extensive research has been conducted into development of S-box /Inv. S-Box and Mix Columns/Inv. Mix Columns on dedicated ASIC and FPGA to speed up the AES algorithm and to reduce circuit area. This is an attempt, to survey in detail, the work conducted in the aforesaid fields. The prime focus is on the FPGA implementations of optimized novel hardware architectures and algorithms. One could work on selection of a larger key size which would make the algorithm is more secure, and a larger input block to increase the throughput. The extra increase in area can however be tolerated. So such an algorithm with high level of security and high throughput can have ideal applications such as in multimedia communications. Furthermore, study of optimization approaches for the implementations supporting multiple key lengths and modes of operation have tremendous scope for future work.

7. Conclusion

The Advanced Encryption Standard algorithm is a symmetric block cipher that can process data blocks of 128 bits through the use of cipher keys with lengths of 128 bits. The usage of 128 bit cipher key to achieve the high security, because 128 bit cipher key is difficult to broken. AES algorithm can resist any kinds of password attacks with a strong practicability in information security and reliability. AES provides better security and has less implementation complexity, it has emerged as one of the strongest and most efficient algorithms in existence today. The AES algorithm can be efficiently implemented by software. Software implementations cost the smallest resources, but they offer a limited physical security and the slowest process.

References

[1] Ai-Wen Luo, Qing-Ming Yi, Min Shi, "Design and Implementation of Area-optimized AES Based on FPGA", 978-1-61284-109-0/11/2011 IEEE.

[2] Yang Jun Ding Jun Li Na Guo Yixiong "FPGA-based design and implementation of reduced AES algorithm," 978-0-7695-3972-0/2010 IEEE.

[3] Alia Arshad, Kanwal Aslam, Dur-e-Shahwar Kundi and Arshad Aziz, "FPGA Implementation of Advance Encryption Standard Using Xilinx System Generator", Asian Journal of Applied Sciences (ISSN: 2321 – 0893) Volume 02 – Issue 02, April 2014.

[4] M. Lukowiak, S. Radziszowski and J. Vallino and C. Wood, "Cybersecurity Education: Bridging the Gap Between Hardware and Software Domain.

[5] M. Hasamnis, P. Jambhulkar and S. Limaye, "Implementation of AES as a Custom", Advanced Computing: An International Journal (ACIJ), vol.3, No.4, July 2012.

[6] WANG Wei,CHEN Jie,XU Fei, "An Implementation of AES Algorithm Based on FPGA", 978-1-4673-0024- 7/2012 IEEE.

[7] A. Amaar, I. Ashour and M. Shiple "Design and Implementation A Compact AES Architecture for FPGA Technology", World Academy of Science, Engineering and Technology 59 2011.

[8] M. C. LIBERATORl and J. C. BONADERO "Aes-128
Cipher. Minimum Area, Low Cost FPGA Implementation".

[9] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Federal Information Processing Standards Publications – FIPS 197, http: //csrc.nist.gov/publications/fips/fips197/fips 197.pdf, November 2001.

[10] Adam J. Elbirt, W. Yip, B. Chetwynd, and C. Paar, " An FPGA-Based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalist ", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 9, NO. 4, AUGUST 2001.

[11] Presented at the Second Advanced Encryption Standard (AES) Conf., Rome, Italy, Mar. 1999.

[12] William Stallings, "Cryptography and Nework Security", Third Edition, Pearson Education, 2003.

[13] 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.

[14] 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 3rd.

[15] Marko Mali, Franc Novak and Anton Biasizzo "Hardware Implementation of AES Algorithm" –Journal of ELECTRICAL ENGINEERING, Vol. 56, No. 9-10, 2005, 265-269.