A

DISSERTATION-II REPORT

On

DESIGN OF ADVANCED CRYPTOGRAPHIC SYSTEM FOR DATA ENCRYPTION AND DECRYPTION USING

Submitted in the partial fulfillment of the requirement for the award of degree

MASTER OF TECHNOLOGY

in

Electronics and Communication Engineering

Submitted by

Dasari Srinivasa Rao (11301621)

Under the guidance of

Mr. Jeripothula Balakrishna
Assistant Professor
Department of Electronics & Communication Engineering

Department of Electronics & Communication Engineering

Lovely Professional University

Phagwara–144411, Punjab (India)

CERTIFICATE

This is to certify that **Dasari Srinivasa Rao** bearing Regd No: 11301621 has completed objective formulation of thesis titled, **"Design of advanced cryptographic system for data encryption and decryption using FPGA"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the thesis has ever been submitted for any other degree at any University.

Mr. Jeripothula Balakrishna
Assistant Professor
Electronics and Communication Engineering.
Lovely Professional University
Phagwara, Punjab.

DECLARATION

I hereby declare that the dissertation-II report entitled "**Design of advanced cryptographic system for data encryption and decryption using FPGA**" is an authentic record of my own work carried out as the requirements for the award of degree of **Master of Technology in VLSI Design at Lovely Professional University, Phagwara** under the guidance of **Mr.Jeripothula Balakrishna**, Assistant Professor, Department of Electronics and Communication Engineering.

Dated:

Dasari Srinivasa Rao

Regd.No: 11301621

ACKNOWLEDGEMENT

I would like to thanks to my guide Assistant Professor Mr.Jeripothula Balakrishna for his guidance, supporting, and encouragement and also his creative thinking in given idea. The idea comes from him became my inspiration to still continue my project will successfully. Still even he busy with him own work, he still spend his time with me to complete the project. I also would like to express my thanks to Professor Mr. Bhupinder Verma (DEAN) and Cherry Bhargava (HOD) for their support, advice and also suggestion in my project.

I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life and also support from all my beloved people give me more strength to complete the project.

TABLE OF CONTENTS

ABSTRACT ..... **i**

CHAPTER 1 INTRODUCTION ..... **……1**

1.1. Motivation **1**

1.2. Reason for implementing AES on hardware **2**

CHAPTER 2 CRYPTOGRAPHY **5**

2.1. Basic Description **5**

2.2. Purpose of Cryptography **5**

2.3. Classification of Cryptography **6**
2.3.1. Symmetric (Private) Key Cryptography **7**
2.3.2.Asymmetric (Public) Key Cryptography **7**

2.4. History of Rijndael AES algorithm **8**
2.4.1. Requirements and evaluation criteria **8**
2.4.2.Evaluation Process **8**

CHAPTER 3 RIJNDAELAES ALGORITHEM **10**

3.1. Basic Algorithm Description **10**

3.2. AES Encryption **10**
3.2.1. Representation of 128bit Key and Input information **11**
3.2.2. Substitution Bytes Transformation **12**
3.2.3. Shift Rows Transformation **13**
3.2.4. Mix Column Transformation **13**
3.2.5.Add Round Key Transformation **14**

3.3. AES Decryption **15**
3.3.1. Add Round Key Transformation **15**
3.3.2. Inverse Shift Rows Transformation **15**
3.3.3. Inverse Sub Bytes transformation **15**
3.3.4. Inv Mix Columns Transformation **16**

3.4. Key Expansion Algorithm **16**

3.5. AES Implementation Stages..... 18

CHAPTER 4 LITERATURE REVIEW..... 21

4.1. Different Hardware AES Implementations in FPGA..... 21
4.1.1. Literature review conclusion..... 26

4.2.Design Methodologies..... 29
4.2.1. Design Science Research Methodology..... 29
4.2.2. Hardware Design Approaches..... 32
4.2.3. Field Programmable Gate Arrays (FPGAs)..... 33
4.2.4. Verilog Hardware Description Language..... 34
4.2.5.Factors of hardware design..... 36

4.3. Research Objective..... 37

CHAPTER 5 DESIGN AND IMPLENTATION..... 38

5.1. Proposed AES Architecture..... 38
5.1.1. Block diagram of Key Generation Unit..... 39
5.1.2. Block diagram of encryption unit:..... 40

5.2. T-box Tables derivation..... 41
5.2.1. Round functions computation:..... 43

5.3. Encryption and Decryption in T-box architecture:..... 43
5.3.1. Last round in Encryption..... 43
5.3.2. Last round inDecryption..... 44

5.4. Modified T-box table used in the implementation..... 45

CHAPTER 6 RESULTS AND DISCUSSION..... 47

6.1. Outcomes of proposed AES Algorithm..... 47

6.2. Simulation Results..... 49

6.3. Performance Parameters..... 50

6.4. Power consumption report for encryption..... 52

6.4. Power consumption report for decryption..... 53

FUTURE SCOPE..... 54

CONCLUSION 55

REFERENCES..... 56

ABSTRACT

These days, internet and network applications are increasing rapidly across the world. The rapid development in the networking technology leads to interchanging of the data very drastically. Hence it is more susceptible to duplicating of data and redistributed by attackers or hackers. Therefore transferring data securely over a communication medium is considerably important in several applications. Consequently data security has become a significant issue in the modern world. This increasing need of information security in computer networks led to the growth of several cryptographic algorithms. Advanced Encryption Standard (AES) is most popular symmetric/private cryptographic algorithm used for encryption and decryption of information. This algorithm plays a vital role in developing the new cryptographic standards, since AES is a block cipher that has been analyzed broadly. Hardware implementations of cryptographic algorithms are physically protected than software implementations because outside attackers cannot alter them. In order to attain higher performance in today's overloaded communication networks, hardware implementations are a wise selection in terms of enhanced speed and high reliability. This work uses efficient hardware implementations, such as S-box and T-box architectures for the Advanced Encryption Standard (AES) algorithm along with secret data-integrity module on Xilinx Artix-7 FPGA families. In order to attain higher speed and smaller area, The Sub-Byte, Inverse Sub-Byte, Mix-Column and Inverse Mix-Column operations are considered as Look-up-Tables (LUTs) as well as Read-Only-Memories (ROMs). This design uses an iterative looping approach with data-block and key size of 128 bits, Look-up-Table implementation of T-box as BRAMs rather than Look-up-table implementation of S-box and other combinational implementations as per the specifications given by the NIST. It will give higher Field Programmable Gate Arrays (FPGA) (Throughput/Area) efficiency comparing to earlier AES implementations. This design approach also provides data-integrity module to check whether data is modified by hacker or not. The function of this module varies according to the sender /receiver agreement but it must be kept secret. Power consumption has been measured by Xilinx Power analysis tool in ISE web pack 14.7 design suite.

CHAPTER 1
INTRODUCTION

1.1. Motivation

In the modern world, Cryptography is not only limited to defence applications but also essential in other applications such as E-commerce, Electronic-mail etc. Cryptography has a major function in embedded systems design. As the number of devices and applications which transmit and receive data are growing rapidly, the information transfer rates are becoming higher. In several applications, this data requires a secured connection which is usually achieved by cryptography. Cryptographic algorithms are Data Encryption Standard algorithm (DES), Triple Data Encryption Standard algorithm (3DES) and Advanced Encryption Standard algorithm (AES) are standardized by National Institute of Standards and Technology (NIST). Many researchers and hackers are constantly trying to crack these algorithms by using brute force, side channel and other attacks. Some attacks were successful as it was the situation for the Data Encryption Standard (DES) in 1993, where the published cryptanalysis attack could crack the DES. The Advanced Encryption Standard (AES) is considered these days as one of the robust published cryptographic algorithms. It was then adopted by the NIST (National Institute for Standards and Technology) after the breakdown of the Data Encryption Standard (DES). In addition, it is used in various applications such as in RFID readers, ATMs, cellular phones and web browsers etc. Due to the significance of the AES algorithm and the various applications that it has, the major concern of this work will be presenting new efficient hardware implementations such as S-box, T-box architectures along with data-integrity for this algorithm. Hardware implementations for the AES algorithm vary according to the particular application. While some applications require extremely high throughput as in the e-commerce servers, other applications require medium throughput as in designs used for cellular phones. Some others require very small area implementations to be used in low power application as in RFID readers. Numerous hardware designs were suggested for the AES algorithm. Some of these designs targeted high speed applications as in the loop unrolled 128 bits designs, while others targeted medium and low area implementations in the designs as each application requires the AES to have different speeds and area.

1.2. Reason for implementing AES on hardware

The AES algorithm can be efficiently implemented in both hardware and software. Generally the required speed of encryption and cost of implementation are the most important factors determining the selection of technology. Software implementations are designed and written in programming languages such as C, Java and C++ etc., and are developed to run on Digital Signal Processors (DSPs), General Purpose Processors (GPPs) and smart cards. These processors offer sufficient power to satisfy the requirements of individual clients, so most of the existing implementations of cryptography reside in the software. On the other hand software implementations are very economical, but they offer a less physical protection and the slowest method. Due to increasing needs for high speed, high level secure communications joint with physical protection, hardware implementation of cryptographic algorithms designed. Hardware implementations are the only way to attain speeds beyond the DSP-processors or General-purpose processors. Hardware implementations are designed and written in hard ware description languages such as Verilog, VHDL or using schematic level design. There exist two major implementation strategies for hardware designs: Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs).

An FPGA implementation is an intermediate solution between application specific integrated circuits (ASICs) and general purpose processors (GPPs). It has advantages over both ASICs and GPPs. It gives a quicker hardware solution than a GPP. Furthermore, it has a wider applications than ASICs because its configuring software make use of the broad range of functionality supported by the reprogrammable device. An FPGA implementation is beneficial as compared to ASIC and software implementations, in terms of flexibility, cost-effectiveness, scalability and adaptability. The design of cryptographic algorithm on reconfigurable devices offers many advantages such as upgradability, algorithm agility and modification etc. Algorithm agility means the capability of the algorithm to choose from a different type of ciphers on per session basis. This requirement comes from the fact that present day security protocols are defined to be independent of the algorithm. Algorithm upgradability means the method of upgrading an existing algorithm in the application due to different reasons like expiration of standard and formation of new standard. The Algorithm modification implies changes in the algorithm implementation based on the necessity and capability of the host platforms. Even though all these changes can also be made in ASIC implementation but they require high cost and takes

more time to market because runtime reconfiguration not possible. The quality of a hardware implementation is measured in terms of throughput, hardware requirement cost, and latency. The implemented hardware must be optimized for cost or area, should offer low latency which reduces the time to encrypt/decrypt single block of information, and should gives high throughput by encrypting/decrypting many blocks simultaneously.

Moreover software encryption speed is depends on the host computer and it will impact total system performance. In addition, Software encryption also has the problem that, when installed, the encryption round keys are stored in gadget memory that can be accessed by other components of the mobile device. Hardware encryption is faster than software encryption since it has dedicated onboard security chip for processing encryption/decryption, key handling and key generation. The hardware encryption does not have an effect on system performance because it does not consume CPU cycles.

Table 1.1 Characteristic features of design of cryptographic algorithms in ASICs, FPGAs and Software (GPPs)

Parameter	**ASIC**	**FPGA**	**Software (GPPs)**
Speed of operation	Very high	High	Moderately high
Implementation cost	High cost	Moderate cost	low-cost
Implementation cycle	More time taken	Moderate time taken	Less time taken
implementation Tools	High cost	low-cost	low-cost
Upgrades and Maintenance	Moderate cost	low-cost	low-cost
Key security	High	moderately high	less
Algorithm Agility	Does not provide	Does provide	Does provide

Every type of design has its pros and cons. Their basic features are shown in table 1.1. Software implementations are an attractive choice when speed of encryption is not a main concern and expected level of security does not have to be high. It means that the importance of the protected information is low compared to the effort required for breaking security mechanisms. The more

secure and faster solutions are required, the more vital role is played by hardware implementations. When taking into account hardware solutions, FPGAs become more and more attractive. If we assume an attacker has no physical access to the device, then FPGA designs can be as secure as ASICs.

CHAPTER 2

CRYPTOGRAPHY

2.1. Basic Description

The word cryptography originated from the two Greek words, namely Crypto means hidden or secret and Graphy means writing. So cryptography is the art and science of creating secret codes. It is protecting the data by converting it into a non-recognizable format in which a message can be hidden for the reader and only the desired recipient will be able to translate it into original message. In cryptography, when the data in its original structure is known as plain text, means information can be read and understood without any special measure. The scrambled information is known as cipher text. The process of converting plain text into cipher text using a key is known as encryption. Conversely, the method of translating cipher text into its original plaintext using a key is known as decryption. A system that offers both encryption as well as decryption is called cryptosystem as shown in fig 1.1

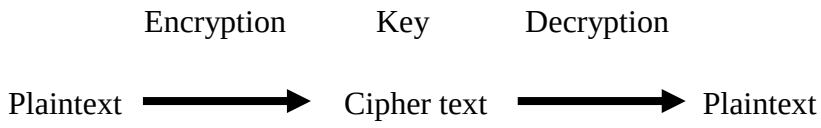

Fig.2.1 Basic Cryptosystem

2.2. Purpose of Cryptography

Cryptography offers various security goals guarantee the privacy of information and for non-alteration of information etc. It is widely used nowadays due to the great security advantages. Various security goals of cryptography are:

Authentication: - The data received by any system has to verify the identity of the transmitter to check whether the data is coming from authorized person/party or not.

Confidentiality: - Data in the computer has to be accessed and is transmitted only by the authorized party or person and not by anybody else.

Integrity: - Modify the transmitted data only by the authorized party or person. Nobody in between the transmitter and receiver are authorized to alter the given information.

Non-Repudiation: - Either the transmitter or the receiver of the information should not be able to deny the transmission.

Availability: - The data transmitted and stored by an association needs to be accessible to authorized entities**.** Data is of no use if it is not available.

2.3 Classification of Cryptography

Cryptography algorithms can be classified into two main categories namely Symmetric (Private) and Asymmetric (Public) key Cryptography.

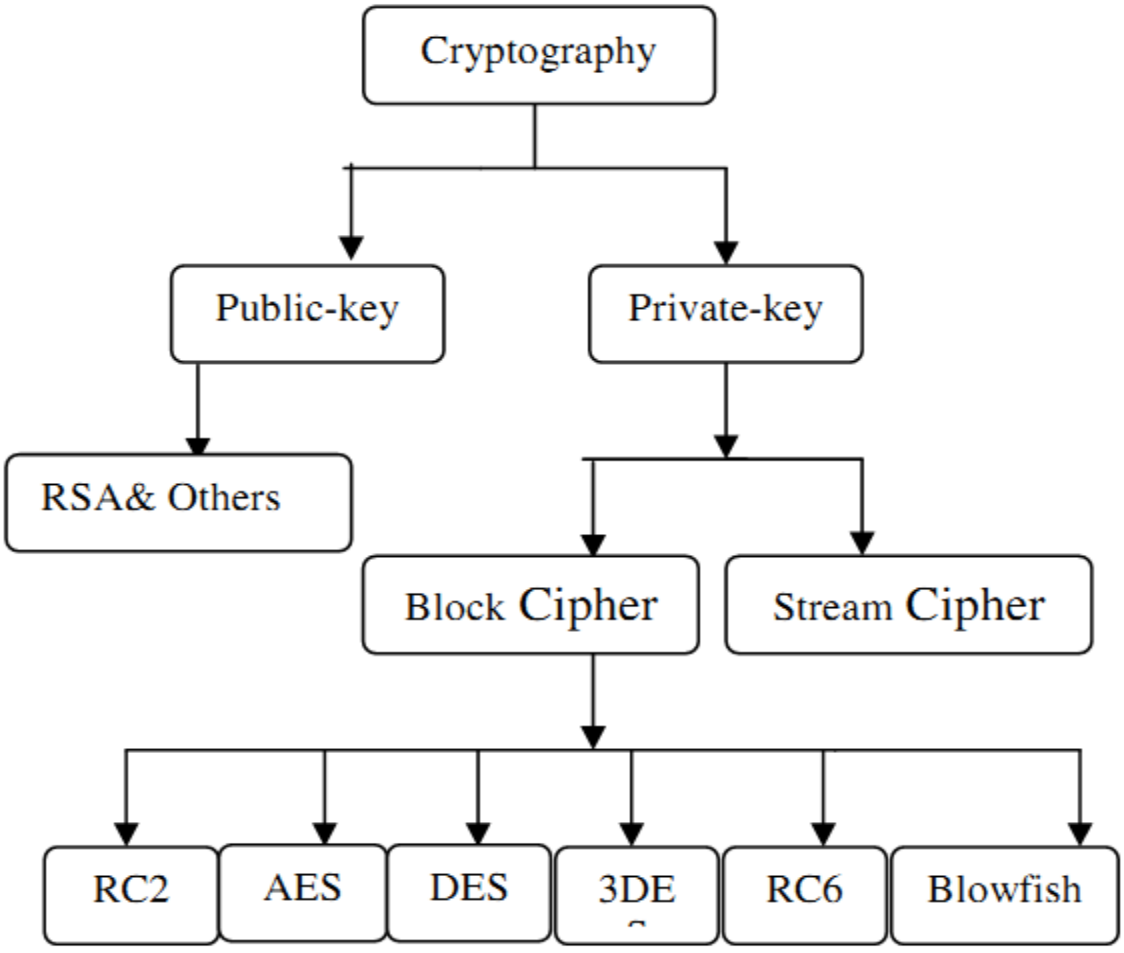

Fig.2.2 Classification of Cryptography

2.3.1. Symmetric (Private) Key Cryptography

Symmetric key cryptography is also known as secret-key, shared-key, and single-key or private-key cryptography. Because it uses a shared key for encryption as well as decryption of the message. Receiver and sender have to share the private key in the starting, after that they can start to encrypt and decrypt messages between them with that key. There are several private key algorithms such as DES, TRIPLE DES, RC4, AES, RC6, and BLOWFISH etc. Shared key cryptography schemes are usually categorized as stream ciphers and block ciphers. Stream ciphers encrypt the data bit by bit. A block cipher scheme operates on a block of information at a time using the similar key on each and every block. The advantages of private key cryptography are: simple, fast, uses fewer computer resources and it prevents wide-ranging information security compromise. While the disadvantages are: Requirement for secure channel for private key exchange, too many keys and authentication of information cannot be guaranteed

2.3.2. Asymmetric (Public) Key Cryptography

Asymmetric key cryptography makes use of two different keys: a public key and a secret key. The public key is made available to the public and is used to encrypt information by any person who desires to transmit a message to other person that the key belongs to. The private key is kept secret and is used to decrypt received information. The RSA algorithm is an example of public key cryptography system. The advantages of public key cryptography are: Convenience, data authentication, Detection of tampering and non-repudiation. While the disadvantages are Asymmetric keys should be authenticated, Slow, Requires additional computer resources, Widespread security compromise is achievable and Loss of private key may be irreparable.

As far which of them is more protected, there are differing views. Most of the experts believe that private key cryptography is more secure. And few others consider the asymmetric key cryptography is better. If possible, both of them are used together to get advantage of their benefits. In secret key cryptography AES algorithm is more secure than other cryptographic algorithms.

2.4. History of Rijndael AES algorithm

Data Encryption Standard (DES) was worldwide frequently used block cipher during 1970-1993. DES encrypts information in 64 bit block size and uses a 56 bit key. 56 bit key offers approximately $2^{\wedge 56}$ possible combinations. Although it seems to be large but according to nowadays computing power it is not enough and susceptible to brute force attack. So, Data Encryption Standard (DES) could not keep up with development in technology and it is no longer suitable for security. Since DES was used widely at that time, the immediate solution was to develop 3DES which is secure enough for most of the applications. 3DES is implemented by applying DES three times in succession. 3DES with three different cipher keys (e.g. K1, K2 and K3) has an effective key size of 168 bits . An extra variation is called two key (K1 and K3 are equal) 3DES reduces the effective key length to 112 bits which is not much secure as 3DES having three different keys. But it is highly inefficient, particularly in software implementations. After that the NIST (National Institute for Standards and Technology) has recognized requirement for new standard and started the process of developing an Advanced Encryption Standard (AES). The most important NIST's target was to build up an algorithm, which offers security at least equivalent to 3DES, and considerably efficient in software and hardware implementations in different platforms.

2.4.1. Requirements and evaluation criteria

NIST announced an official call for contestant algorithms in 1997. The minimum acceptable capabilities were: The algorithm must design or implement symmetric (private or secret) key cryptography, the given algorithm should be a block cipher and the contestant algorithm shall be able to supporting block-key combinations with sizes of 128-128, 128-192 and 128-256. All candidate algorithms were evaluated with respect to following three criteria: Security, Cost and implementation characteristics like flexibility, hardware/software suitability and simplicity.

2.4.2. Evaluation Process

The method of evaluating best AES algorithms has been divided into number of rounds. The initial round was mainly focus on the evaluation of algorithms depends on the cryptanalysis carried out by public and the efficiency of software implementations on a different type of platforms. Several algorithms were initially presented by researchers from 12 different countries.

Fifteen, algorithms were chosen from the first phase of submissions. After the one year research, five algorithms were selected as AES finalists from second phase of selection process. They are MARS, SERPENT, RIJNDAEL, TWOFISH and RC6. The conclusion was that the five contestants illustrated similar type of characteristics. Finally in October 2000, NIST declared that the Rijndael algorithm was the winner of the contest. The Rijndael algorithm was selected because it offers better security, efficiency, flexibility, performance and implementation ability as compared to other algorithms. The Advanced Encryption Standard (AES) is designed for the encryption or decryption of digital electronic information. This algorithm is depends on the Rijndael cipher developed by two Belgian cryptographers, namely Joan Daemen and Vincent Rijmen. Rijndael AES is a group of ciphers with different combinations of block and key sizes. AES algorithm became effective as a federal government standard (FIPS) on May 26, 2002 after approved by the Secretary of Commerce. AES has been adopted by the U.S. government and is now used worldwide.

Table 2.1: Comparative studies of different cryptographic techniques

PARAMETER	AES	3DES	DES	RC2	BLOWFISH	RC6
Key size (bits)	128, 192 or 256	168 or 112	56	8-128 or 64	32-448	128, 192 or 256
Cipher type	Symmetric block cipher	Symmetric block cipher	Symmetric block cipher	Symmetric cipher	Symmetric cipher	Symmetric cipher
Data size (bits)	128, 192 or 256	64	64	64	64 bits	128
Developed	2000	1978	1977	1987	1993	1998
Security	Considered secure	Secure but slow process	inadequate	Vulnerable	Vulnerable	Vulnerable
Possible keys	2^{128}, 2^{192} or $2^{\wedge 256}$	2^{112} or 2^{168}	2^{56}	2^{64} or 2^{128}	2^{32} or 2^{448}	2^{128}, 2^{192} or 2^{256}
Rounds	10(128bits), 12(192bits), 14(256bits)	48	16	16 of type mixing,2 of mashing	16	20

CHAPTER 3

RIJNDAEL AES ALGOTITHM

3.1. Basic Algorithm Description

The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption unit converts information to a non-readable format called cipher-text. Decryption of the cipher-text converts the information back into its original form, which is called plain-text. The AES algorithm operates on a 128-bit block of information and run in number of loop iterations. The number of iterations (N) of a loop, N can be 10**,** 12, or 14 based on the key size. The key size is 128, 192 or 256 bits respectively. Here, this AES algorithm uses 128 bit key size.

3.2. AES Encryption

The Rijndael AES algorithm (128bit key) consists of ten iterations of encryption. First the 128-bit key is expanded into eleven so-called round keys, each of them 128 bits in length. The initial and final iterations are differing from other iterations in that there is an additional Add Round Key operation at the starting of the initial round. Means first round key is added (XOR) with the plain text. After that nine identically structured iterations to follow. Each of these nine iterations consists of the following transformations:

- Substitute bytes
- Shift rows
- Mix columns
- Add round key

The final iteration is similar to the iterations one to nine, but the Mix columns transformation is not performed. In the following sections, these four transformations are explained. Figure 3.1 shows the flow diagram of AES encryption.

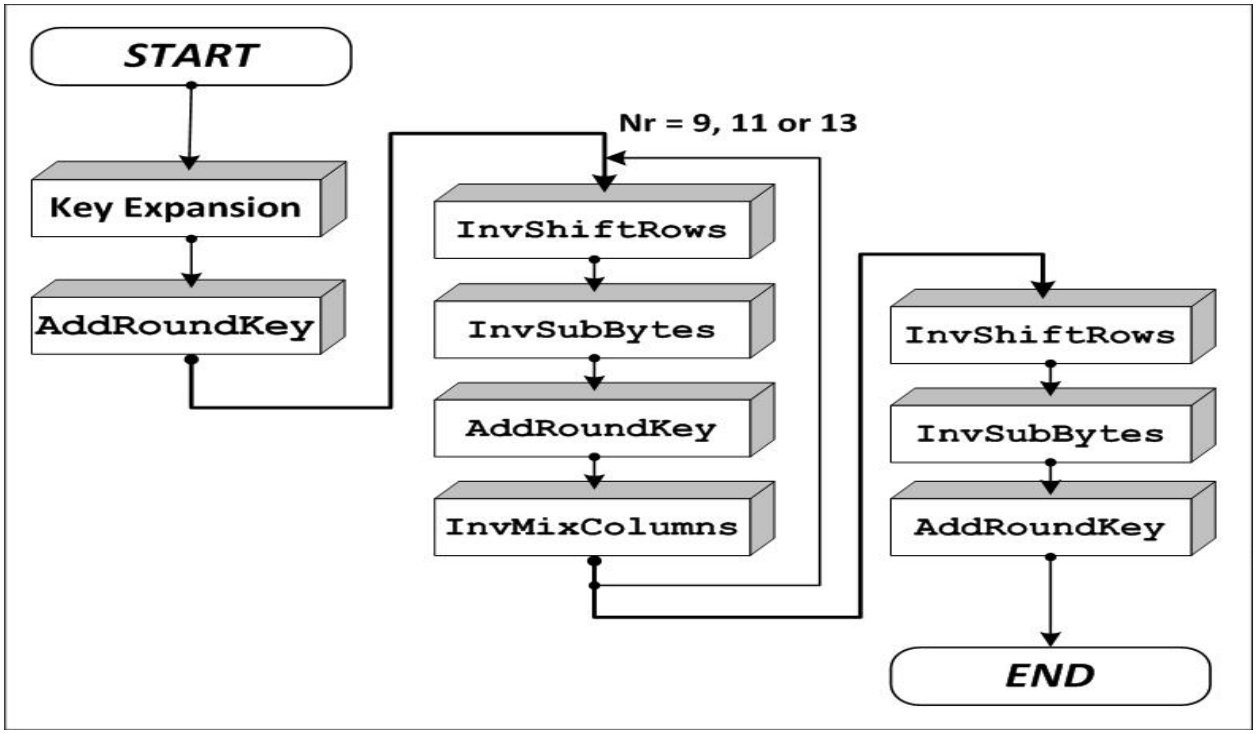

Fig.3.1 AES encryption

3.2.1. Representation of 128bit Key and Input information

Both the input information and the key are represented in a 4x4 matrix of 16bytes (128bits). How the 128-bit key and input information are distributed into the byte matrices as shown in Fig. 3.1.1.

The State

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

The Key

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}

Fig. 3.1.1 Representation of 128bit Key and Input Data

3.2.2. Substitution Bytes Transformation

The Sub bytes transformation is a non-linear byte substitution, operating on all state bytes independently. This is the major concern for the security of the AES. There are different ways of interpreting the Sub bytes operation in this design; one is combinational implementation, it uses Galois-field (composite field) arithmetic for on-the-fly calculation of sub byte values. While the other technique is look up table approach, this is done using a once-pre calculated substitution table called S-box. That S-box contains 256 numbers (from 0 to 255) and their corresponding resulting values as shown in table 3.1. The 16 bytes of the state (the input data) are substituted by the corresponding values found in the table. This is most efficient method than directly implementing the multiplicative inverse operation followed by affine transformation. This approach avoids complexity of hardware implementation.

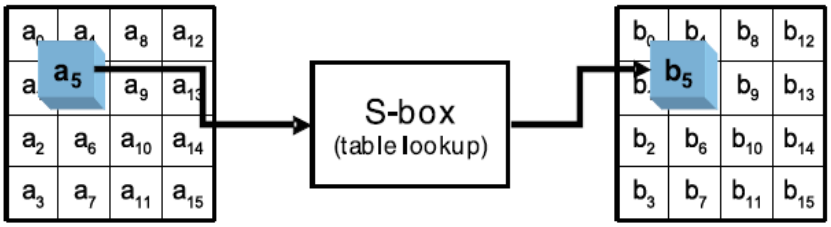

Fig. 3.1.2 Substitution Bytes Transformation

Table 3.1: Look-up-Table For Substitution

		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	1	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	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
	4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	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	6	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	8
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	3	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

3.2.3. Shift Rows Transformation

In this operation the rows of the state (4x4 byte input information) are cyclically left shifted with respect to offset of the row. Row 0 is not shifted, row 1 is cyclically shifted one byte of information to the left, row 2 is cyclically shifted two bytes of information to the left and row 3 is cyclically shifted three bytes of information to the left. The operation of Shift rows as shown in Fig. 3.1.3.

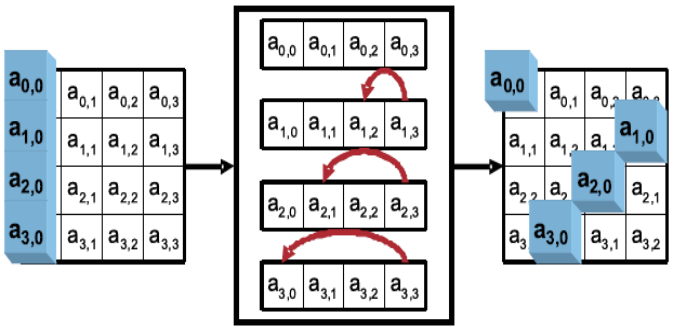

Fig. 3.1.3 Shift Rows Transformation

3.2.4. Mix Column Transformation

Mix column transformation is perhaps the most complex operation from a software implementation point of view. Contrast to the Shift rows operations, which operate on rows in the 4x4 state matrices, the Mix column operation process on the columns. In this, only a matrix multiplication required to be executed. To make this operation reversible, the normal addition and multiplication are not used. In AES, Galois field arithmetic operations are used. In Galois field arithmetic, an addition corresponds to an XOR operation between the operands and a multiplication is to a more complex equivalent. The mix column transformation takes a 4x4 byte matrix as input, and performs a matrix multiplication for each column with a constant vector.

In Mix Column transformation, the columns of the state are considered as polynomials over GF (2^8) and multiplied by modulo $X^4 + 1$ with a fixed polynomial c(X). This fixed polynomial defined by NIST, given by: $c(X) = \{03\} X^3 + \{01\} X^2 + \{01\} X + \{02\}$. The working principle of Mix column transformation as shown in Figure 3.1.4.

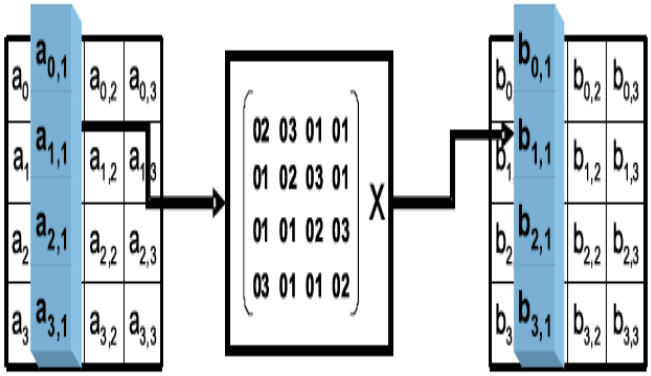

Fig. 3.1.4 Mix Column Transformation

3.2.5. Add Round Key Transformation

In the Add Round Key transformation, a Round Key is added to the State which is resulted from the operation of the Mix Column transformation. This transformation can be done by using simple bitwise XOR operation. The Round Key of each round is extracted from the main key using the Key Expansion Module. The encryption/decryption algorithm required eleven 128-bit Round Keys, which are denoted by Round Key[0] , Round Key[1] ,…,Round Key[9],Round Key[10]. In the initial round of encryption, Round Key [0] is added (XOR) to the plain text, where as in initial round decryption, Round Key [10] is added (XOR) to the cipher-text. The operating principle of Add Round Key transformation as shown in Fig. 3.1.5.

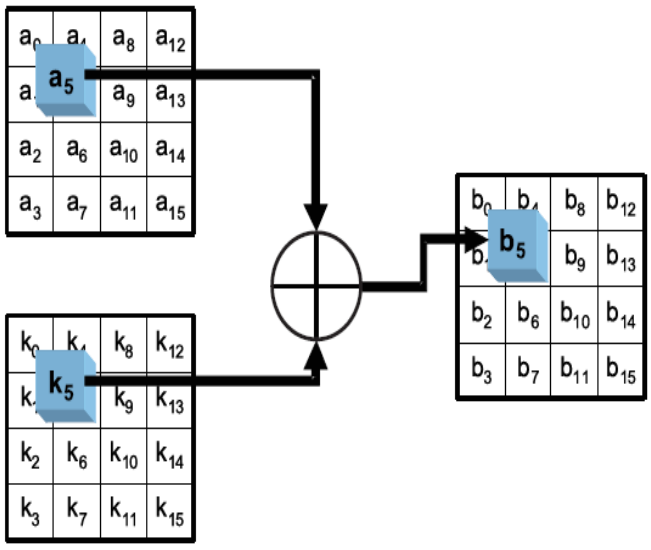

Fig. 3.1.5 Add Round Key Transformation

3.3. AES Decryption

Decryption of information is a reverse of encryption in which inverse round transformations are used to compute original plain text from the given cipher text. In order to get original information, it uses inverse round transformations in reverse order.

The Rijandael decryption consists of four inverse operations of encryption.They are

1. Inverse Substitution
2. Inverse Shift Row
3. Inverse Mix Column
4. Add Round Key Transformation

3.3.1. Add Round Key Transformation

Add Round Key Transformation is its own inverse function because the XOR operation is its own inverse. However, the round keys have to be selected in reverse order.

3.3.2. Inverse Shift Rows Transformation

Inv Shift Rows exactly functions the same as Shift Rows, 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.

3.3.3. Inverse Sub Bytes transformation

The Inv Sub Bytes transformation is done by two techniques. One is using a once-pre calculated substitution table called Inv S-box. That Inv S-box table contains 256 numbers (from 0 to 255) and their corresponding values. Inv S-box is presented in Table II. Another technique is on-the-fly calculations for inverse sub byte values using Galois field over 2^8.This transformation exactly inverse to the sub byte transformation. For example in this transformation ‘0’ is mapped to ‘52’, where as in sub byte transformation ‘52’ is substituted to ‘0’

Table 3.2: Lookup Table for Inverse Substitution

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
X	0	52	9	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	8	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	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
	7	d0	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	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	1a
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	7	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	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

3.3.4. Inv Mix Columns Transformation

This transformation exactly same as Mix Column transformation except constant polynomial. In the Inverse Mix Column transformation, the columns of the state are considered as polynomials over GF (2^8) and multiplied by modulo $X^4 + 1$ with a fixed polynomial d(X). This fixed polynomial defined by NIST, given by: $d(X) = \{0B\} X^3 + \{0D\} X^2 + \{09\} X + \{0E\}$, where {0B}, {0D}, {09}, {0E} denotes hexadecimal values.

3.4. Key Expansion Algorithm

The key expansion operation is fundamental for the AES algorithm, the key is changed with the key expansion for each round. The altered key is used as input to next iteration. The changed key is referred to as the key schedule because there is one unique key for each iteration. A top level description of the key expansion algorithm as shown in Fig. 3.2.

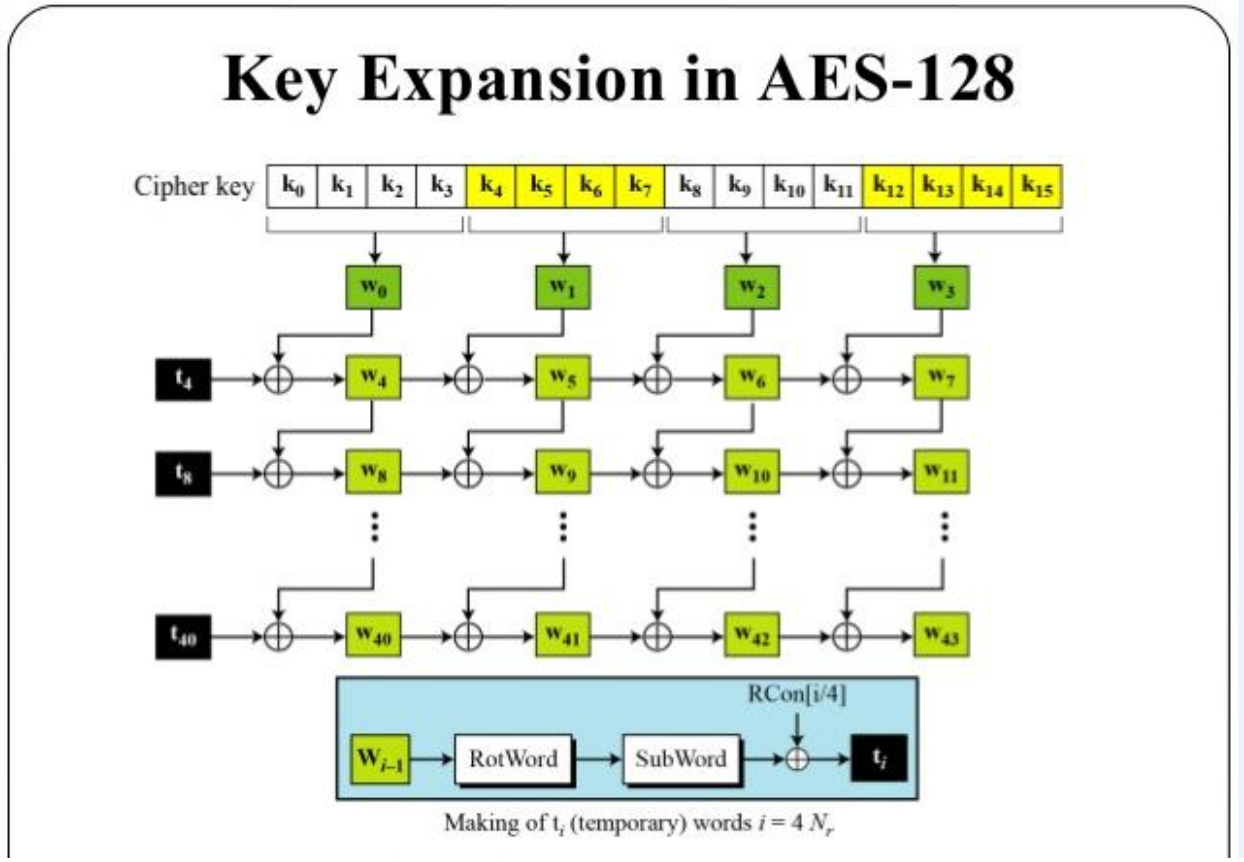

Fig. 3.2 Key Expansion in AES

The key consists of 16 bytes; however the key expansion processes the key as four sets of four bytes. The first operation is applying the SBOX to the last column four bytes. The next step is a byte left Rotation. After that the round constant (RCON) is XOR with the last four bytes. Round constant (RCON) which is a simple lookup table with 10 entries, each having 4 bytes constant. The result is four new bytes for the next key schedule. The remaining 12 bytes of next key schedule is obtained by simple XOR operation between different set of 4 bytes in the previous key schedule as shown in Fig. 3.2. The content of the RCON look up table is specified by National Institute for Standards and Technology. It is calculated by using equation.

$$RCON = 2^{round} \bmod (2^8+2^4+2^3+2+1).$$

Table 3.3: RCON look up table

j	RCON [j]
0	**32'h01000000**
1	**32'h02000000**
2	**32'h04000000**
3	**32'h08000000**
4	**32'h10000000**
5	**32'h20000000**
6	**32'h40000000**
7	**32'h80000000**
8	**32'h1b000000**
9	**32'h36000000**

3.5. AES Implementation Stages

Since the publication of the AES (Rijndael) algorithm in 2001, different hardware implementations were proposed for this algorithm. Most of these implementations have targeted the AES with 128-bit key length. This key length is considered to be enough for the majority of the commercial applications, where using larger key lengths is considered as waste of resources as it need larger area implementations with longer processing time. Key lengths of 256 bit and 192 bits are used mainly in top secret military applications to guarantee the maximum level of information security.

AES implementations stages can be divided into three major types based on data-path width. The first type come up with 8-bit data–path width as implemented in targeting for small area structural design. The second type comes with the 32-bit data -path structures which process every state array column or row collectively and targeting a normal throughput applications.

The third stage comes with the implementation of 128-bit loop unrolled structural designs which targets extremely high speed applications. Generally, implementations with 8-bit and 32- bit data path widths uses looping architectures. Fig.3.3.1 shows the looping architecture of one stage of AES encryption/decryption with a feedback at the ending. In this approach the information will go through this stage until finishing the required amount of rounds which is determined according to length of the given key. This AES stage could be simply an encryption block or an encryption with decryption block and it comprises the hardware implementation for the four different AES operations: Byte Substitution, Shift Rows, Mix Column and Add Round Key operations. For an extremely high speed applications which is designed as complete 128 bit data -path width, the throughput can be increased ideally N times with the implementation of loop unrolled structures. In this structural design, duplication of the N similar AES stages is implemented in series. In AES 128 bits key size structural design, replicas of 10 AES rounds are required to complete the encryption or decryption of the information. In order to get benefit from the loop unrolled structural design, a pipelining stage is implemented at the end of each and every AES stage which permits entering new information at each and every clock cycle, so, all AES stage will be operating simultaneously. The loop unrolled architecture with pipelining techniques as shown in Fig. 3.3.2.

In sub-pipelining techniques, instead of applying pipelining stage at the end of each AES stage, it is divided into certain number of pipelining stages inside each AES stage. This technique increases the throughput, certain number of times as compared to throughput achieved by using normal pipelining techniques. The loop unrolled structural design with sub-pipelining techniques as shown in Fig. 3.3.3.

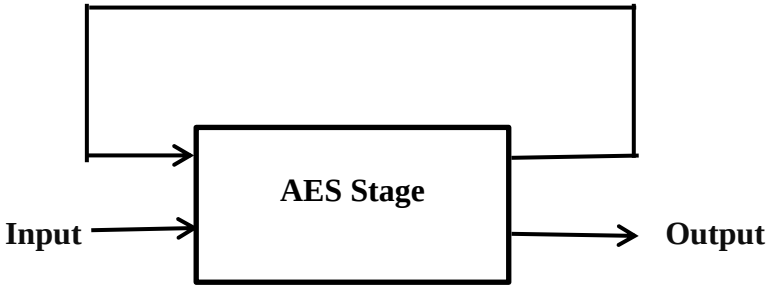

Fig. 3.3.1 Loop rolling architecture.

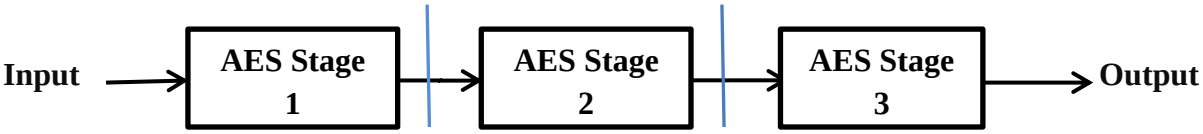

Fig. 3.3.2 Loop-unrolled architecture with pipelining.

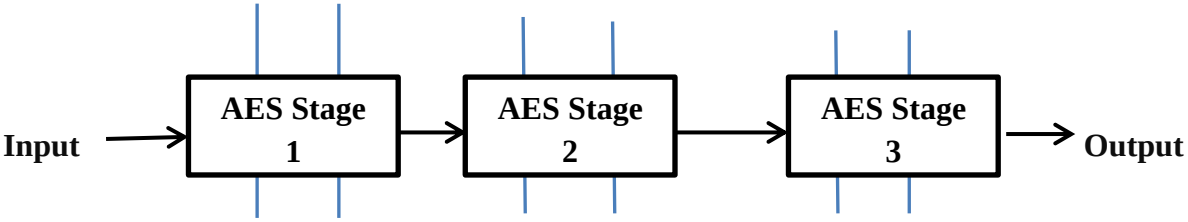

Fig. 3.3.3 Loop-unrolled architecture with sub-pipelining.

The loop unrolled sub-pipelined AES implementations which offers tens of gigabytes of throughput are used in numerous applications such as in e-commerce as highly traffic servers. Substitution Box (SBOX) implementation is a major concern in the AES hardware structural design. Two main techniques where proposed for the design of the S-BOX. The first technique is by pre storing the S-BOX elements in block RAMs (BRAMs), BRAMs is an FPGA memory block of RAM which can be used to store information. The design has used the BRAMs to offers high speed loop -unrolled structural design.

The second technique uses the Galois-field on-the-fly calculation of S-BOX, and implemented as high speed loop unrolled sub-pipelined AES structural design. As pipelining cannot suitable for BRAM since it is a memory block. Using BRAM in the implementation of S-BOX will limit the number of sub-pipelining stages used in the design. Furthermore implementation using BRAMs typically requires larger area than the Galois-field arithmetic structural designs.

CHAPTER 4

LITERATURE REVIEW

The literature review is divided into two sections: Different hardware AES implementations in FPGA and Design Methodologies to achieve required Goals. The first section is studied in order to find out how efficient AES algorithms have been implemented in FPGAs so far. For that it presents various AES implementations using different techniques. The second section gives different design methodologies to achieve our desired goals. This section also suggests the hardware and languages required for the efficient implementation of AES algorithm.

4.1. Different Hardware AES Implementations in FPGA

- In 2013, Yerlikaya and Akman were published an article where they compare performance between FPGA and CPU for the encryption of the data. The CPU implementation was in C programming while the FPGA implementation was in VHDL. The encryption algorithm has 128 bit block length AES with a key size of 128 bit. The comparison was based on simulations results; the encryption processing time was 390ns for the FPGA and 11000 ns for the CPU. This article is relevant for our study since it provide an empirical example of the superior performance of the FPGA versus a CPU.

- In 2013, Kumar and Sharma have improved the latency in the AES kernel by using an enhanced VLSI implementation. The Sub Bytes, which are part of the S-box in the AES encryption has been implemented in logic instead of placing them in ROM (BRAM), since the access time for the CLB's are much lower compared to BRAM access, then is the latency decreased. The designed was implemented in a Xilinx Virtex-II device and the simulation shows that the latency can be reduced by 0.6-0.9 ns, which is roughly the penalty for accessing the BRAM. While the FPGA technology used are rather outdated, the study shows, that latency improvement can be achieved by reducing the access time for the parameters of the s-box

- Dogan and Saldamli studied in 2012 the design techniques for FPGA AES encryption to achieve low power consumption. The designed minimize the power by reusing

calculations block in the AES kernel such as the S-box. Instead of performing true parallel calculations, input the reused kernel was time slot multiplexed, thus utilizing the periods where the blocks are idle. The design was targeted to a Xilinx Spartan-3 XC6SLX150L. The designed was running at low speed, 20 MHz and the throughput was low. The conclusion was that proposed design technique did lower the power consumption drastically. However no absolute power numbers was published

- In 2011 did a team consisting of Hongying Liua, Ying Zhoub, Yibo Fanc, Yukiyasu Tsunood and Satoshi Go to study how to increase the security the FPGA implementation, by considering the possibility of side channel in form of differential power analysis by using advance randomization where they able to hide data dependent encryption in the power spectrum. The performance of the implementation was 2.56Gbit/s.

- Jason Van Dyken and Jose G. Delgado-Frias investigated in 2010 how encryption strength and power consumption was related in FPGA implementation of AES. The study showed how to lower the power consumption of the encryption with minimum effect on performance. They were able to lower the power consumption with 66% while only lowering the encryption strength with 27%. The target device was a Xilinx Virtex-II Pro

- The implementation of AES encryption and decryption in a FPGA was done in 2010 by Yogesh Kumar and Prashant Purohit, they have implemented a parallel 128 Bit AES in a Xilinx Spartan 3 device. The focus of their work was the achieving high hashing speed in a low-cost device.

- The AES encryption was implemented in a FPGA in sequential and parallel architectures in 2003 by Nazar A. Saqib, Francisco Rodriguez Henriquez and Arturo Diaz-Pirez. The aim of the research was to compare sequential and parallel architectures in respect to area and speed. In sequential architecture did the implementation occupy 2744 CLB slices while the parallel architecture occupied 2136 CLB slices and not used any BRAM for this design. The sequential architecture was encrypting at 0.259Gbit/s while the parallel architecture was encrypting at 2.868Gbit/s. The target device was Xilinx Virtex E.

- In 2013, High performance FPGA implementation of AES was done by Abhijith.P.S, Mallika Srivastava and Aparna Mishra. They implemented AES-128 algorithm in vertex-5 FPGA. With the designing of all the AES transformations as LUTs and ROMs, the proposed design achieves a throughput of 3.74Gbps, latency 10 clock cycles and thereby utilizing only 1% of slices in the targeted FPGA.

- The fully pipelined AES implemented was implemented by Hodjat and Verbauwhede in 2004. They managed to fit the algorithm into one Virtex-II Pro FPGA. The latency for the algorithm was only 31 clock cycles and they achieved an encryption rate of 21.54Gbits/s. The implementation used 84 BRAMs and 5177 CLB slices, giving an efficiency of 4.2 Mb/slice if the BRAM usage is not taken into consideration

- Dur-e-Shahwar Kundi, Arshad Aziz and Nasar Ikram published a paper in March 2010 describing the resource efficient implementation of T-boxes in AES. The design was implemented on a Xilinx virtex-5 FPGA and operating at a clock frequency of 251.421MHz. This implementation saves 50% of the memory requirement (BRAMs) as compared other T-Box AES implantations, which aims to target 3Gbps to 32Gbps throughput applications.

- In June 2012, O.Prasanthi and M. Subba Reddy were developed an iterative AES 128 bit algorithm (written in VHDL) with Vertix-6 FPGA and verified using Xilinx 8.1 simulator and thereby achieves a throughput of 352Mbps for both encryption and decryption.

- L. Thulasimani and M.Madheswaran jointly published a paper in 2010 which presents FPGA implementation of Rijndael AES algorithm for Reconfigurable mobile terminals. They implemented complete algorithm (AES-128, AES-192 and AES-256) in the same hardware. It is designed in vertex-2 FPGA and requires 12 clock cycles for AES-128. As result 666.7Mbps of throughput is achieved and utilizes 2943 CLB slices.

- Secured high throughput AES algorithm was implemented by Manjesh.K.N and R.K. Karunavathi in may 2013. They presented a hardware implementation of pipeline AES architecture which includes both encryption and decryption and also gives an idea of restricting the number of pipelining stages in the design. The design is modelled using verilog HDL and simulated with the help of ISE 9.2. Synthesis is done by using RTL compiler v11.2. For 9 pipelining stages, this design gives a throughput of 800Mbps, uses 84268 cells, dynamic and leakage power dissipations are 341.88 mw, 27.043uw respectively and 20 clock cycles are required for the complete encryption.

- In 2010 Jason, Van Dyken and G. Delgado-Frias have published a paper describing the different FPGA schemes for minimizing the power-throughput trade-off in AES Algorithm. Their study is focus on how to lower the power consumption of an FPGA-based encryption scheme with minimum effect on performance. Three novel FPGA schemes are introduced and evaluated. These schemes are compared in terms of architectural and performance differences, as well as the power consumption rates. The results show that the proposed schemes are able to reduce the logic and signal power by 60% and 27% respectively on a vertex-2 pro FPGA while maintaining a high level of throughput.

- The Advanced Encryption Standard Implemented on FPGA by a Sujatha, Hiremath and M.S.Suma. This paper talks of AES 128 bit block and 128 bit cipher key and is implemented on Spartan 3 FPGA .Synthesis results in the use of 2511 slices, 712 Flip flops, and 4805- 4 input Look Up Tables. The design target is optimization of speed and cost. Maximum Frequency: 116.485MHz.

- In 2010, Nabihah Ahmad, Rezaul Hasan and Warsuzarina Mat Jubadi have designed AES S-Box using combinational logic Optimization. The proposed work employs a combinational logic design of S-Box implemented in Virtex II FPGA chip. The architecture employs a Boolean simplification of the truth table of the logic function with the aim of reducing the delay. The S-Box is designed using basic gates such as AND

gate, NOT gate, OR gate and multiplexer. Theoretically, the design reduces the overall delay and efficiently for applications with high-speed performance. The proposed S-Box gives another option for hardware implementation other than composite field to represent Sub-byte transformation. It reduces the complexities of hardware by avoiding the use of multiplicative inverse in Galois field. As compared to the LUT and composite field, the S-Box resulted in smaller area with medium delay.

- In 2013, High-performance and Balanced Method of Hardware Implementation for AES published by Xiaotao Zhang, Hui Li and Shouwen Yang. Optimized and synthesizable verilog HDL code is developed for the realization to achieve 2.33Gbits/s throughput and both AES-128 encryption and decryption process are verified using Quartus II 10.0 software from Altera and simulated by ModelSim SE 6.1f. 5037 slices for encryption and 5049 slices for decryption.

- In 2008, FPGA implementations of high throughput sequential and fully pipelined AES algorithm designed by Chih-Peng Fan and Jun-Kui Hwang. In this Sequential AES uses XC2V3000-6 FPGA for Enc/Dec slices 7617 latency 11 frequency 75.3 MHz throughput 0.876Gbits/s while Our Fully pipelined AES uses XC2V3000-6 FPGA for Enc/Dec slices 139357 (10 chips) latency 51 frequency 222.2 MHz throughput 28.4Gbits/s.

- Hoang Trang and Nguyen Van Loi have implemented an efficient FPGA implementation of 128 bit block and 128 bit key AES algorithm has been presented in this paper. The design is implemented on Altera using APEX20KC FPGA which is based on high performance architecture. The proposed design is implemented based on the iterative approach for cryptographic algorithms. Our architecture is found to be better in terms of latency, throughput as well as area. The design is tested with the sample vectors provided by FIPS 197 [2]. The algorithm achieves a low latency and the throughput reaches the value of 1054Mbit/sec for encryption and 615Mbit/sec for decryption.

- In 2010, Tanzilur Rahman, Shengyi Pan and Qi Zhang have studied hardware implementation of a high throughput 128- bits Advanced Encryption Standard (AES) algorithm on a single chip of Xilinx Spartan III XC3S1000 FPGA has been presented. The bus width of the architecture is 32 bit. Pipelining method has been used in this design in order to achieve a higher speed. Sub-Bytes method has been implemented using both composite field method and fixed Rom for further analysis and comparison of performance. Through a perfect combination of different methods of S-Box and key Expansion, a notable speed has been achieved in the range of 1.11Gbps to 3.22Gbps. Sub pipelined sub bytes requires 6605 slices and achieve 3.22 throughput. For rom basesd 6.279 ns, whereas composite field requires 22ns Combinational delay.

- In July 2012, Samir El Adib and Naoufal Raissouni published a paper on AES Encryption Algorithm Hardware Implementation: Throughput and Area Comparison of 128, 192 and 256-bits Key. In the present paper, throughput and area of 128, 192 and 256-bits AES have been measured in a hardware implementation. Results show that; key size has an almost-linear impact on throughput whereas it has an exponential positive relation with area. In terms of area 192-bits and 256-bits AES hardware design in this paper require about 21.31% and 48.51%, respectively, more area than 128-bits AES design. In the hardware perspective, bigger key size also means bigger area and small throughput. Some companies that employ ultra-high security in their systems may look for a key size bigger than 128-bit.

4.1.1. Literature review conclusion

The literature review has shown that there is no standard FPGA implementation of AES. Each Implementation was aimed to achieve different goals. The review has revealed that preferred key length for the researcher was 128 bit, instead of the more secure AES-192 and AES-256 bit. The most common goal for the researchers was to achieve as high throughput as possible. The highest throughput was achieved by Abhijith.P.S, Mallika Srivastava and Aparna Mishra. They implemented AES-128 algorithm in vertex-5 FPGA. They were able to reach 3.74Gbps on Virtex-5, and thereby utilizing only 1% of slices in the targeted FPGA.

The closest competing implementation was done by Nazar A. Saqib, Francisco Rodriguez Henriquez and Arturo Diaz-Pirez. They used a parallel architecture. They achieved a throughput of 2.868Gbps, with the use of 2136 slices. However, they used a Virtex E FPGA technology .They did not publish the latency for the core, nevertheless the design is comparable with the single pipeline architecture and we can therefore expect that the latency is approximately 20 clock cycles. Another closest design is done by Dur-e-Shahwar Kundi, Arshad Aziz and Nasar Ikram in March 2010 describing the resource efficient implementation of T-boxes in AES. The design was implemented on a Xilinx virtex-5 FPGA and operating at a clock frequency of 251.421MHz. This implementation saves 50% of the memory requirement (BRAMs) as compared other T-Box AES implantations, which aims to target 3Gbps to 32Gbps throughput applications.

The literature review has revealed a knowledge gap so far there has been little focus on power consumption. The reason could be that the published designs are only conceptual and the problem of reducing power consumption is left out for further research. Dogan and Saldamli had a study aimed directly at power reduction 0. However Dogan and Saldamli measured power on a low throughput design on older FPGA technology. There were no absolute measurements for the power consumption and the results cannot be transferred to new die technology since the ratio between static and dynamic current has changed drastically. Manjesh.K.N and R.K. Karunavathi did an effort to estimate power consumption using the Xilinx XPower analyzer tool, yet the XPower only provides a rough estimate which enables the hardware designer to dimension the power supply. Moreover, the estimate was not done at the target clock frequency, which only adds uncertainty to the estimate.

The conclusion of the literature review is that there are some excellent design ideas for AESFPGA implementations such as the Parallel sub-pipelined architecture or T-box approach (Efficient use of BRAMs) which have excellent performance on a modern FPGA technology, both in respect to throughput, efficiency, latency and power consumption the performance summary at 100MHz for the articles is listed below. NA means not available.

Table 4.1: Literature review performance summary

Author	Throughput (Gbps)	Area(slices)	Latency (clock cycles)	Power performance(Gb/Ws)
Hongying Liua, Ying Zhoub and Yibo Fanc	2.56	NA	NA	NA
Nazar, Henriquez and Pirez	2.87	2136	NA	NA
Abhijith, mallika and Aparna	3.74	NA	10	NA
Prasanthi and subba reddy	0.352	NA	NA	NA
Thulasimani and madhswaran	667.7	2943	12	NA
Tanzilar, Rahaman and Shengyi	3.22	6605	NA	NA
Zhang, Hui li and Shouwen	2.33	5049	NA	NA

4.2. Design Methodologies

The research method used in this thesis is Design Science Research Methodology. The Design Science Research Methodology is a relatively new method; it was first published in a journal article in 2007. The method is developed specifically for the field of Information Security and covers the gap between interpretive research in the field of information security and the discipline of engineering.

The objective of the project is to efficiently implement Rijndael AES algorithm. We need the knowledge from Information Security to choose and evaluate the encryption algorithm but we also need the discipline of engineering to implement and test the effectiveness of the chosen encryptions techniques, for this type of problem Design Science Research Methodology is an obvious choice. Other methods could also be chosen, but since this study aims to improve a current implementation of AES, is it difficult to predict what design changes that would produce good results. Therefore an iterative process like Design Science Research Methodology is an effective approach.

4.2.1. Design Science Research Methodology

The design science research methodology based on the journal article "A Design Science Research Methodology for Information Systems Research". The method uses six activities which are normally executed in a sequence. However the method is not constrain the researcher to start at the first activity. Further, the method is an iterative process which means the result of a given activity determines if the researcher goes forward to the next activity or back to the previous activity and uses the new knowledge as input. The flow of the method is shown in Fig. 4.1.

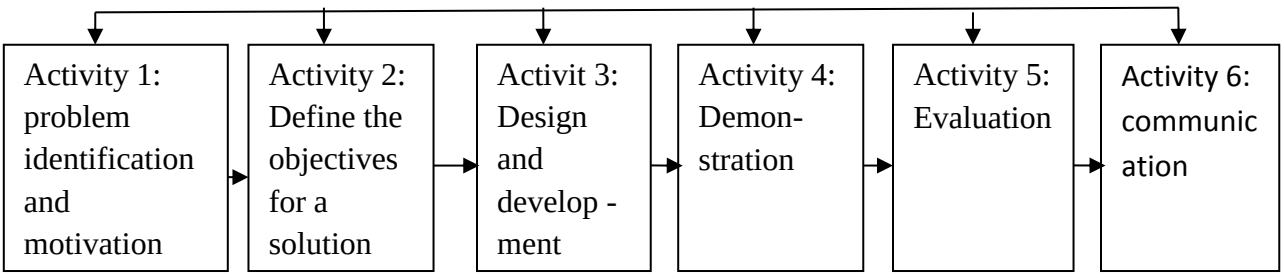

Fig. 4.1 Design Science research methodology

The research process in this project follows the principals of Design Science Research Methodology. The Design Science Research Methodology allows us to start at any activity, however this project is problem centered and therefore the best activity to start with is number 1. The input to the project research process was the project idea, which was to create a research platform to solve problems with the privacy of data.

Activity 1: Problem identification and motivation

The project idea was synthesized into project proposal. The problem to be addressed was how to implement encryption of data in hardware. The motivation for the project is that there is a need for real-time encryption in security terminals with power consumptions. The planning for project was also done in this activity, you can argue that the planning should be done in activity 2, since it is hard to plan are project where the objectives for the solutions are not fully known. However due to limited timeframe of project, the overall planning has to be done before you know how to create the solutions. This is not much different from the discipline of engineering, the project time frame is often fixed long time before the engineers know how to solve a problem.

Activity 2: Define the objectives for a solution

The creation of State-of-the-art has been the driver for this activity. The key component is the literature review, since it is fundamental for defining the objectives for solution that can generate new knowledge. The state-of-the-art has also served another important role, it has shown that encryption in FPGA has been done before; therefore it has little scientific value simply to repeat what others have done before, instead our solution is based on reusing an AES encryption algorithm already implemented and tested. Part of this activity has been to contact the researchers who have already implemented AES encryption in FPGA and retrieve their source code.

Activity 3: Design and development

The activity has a number of steps which is typical for FPGA design:

1. System design for the FPGA: The system design defines all modules in the FPGA and how they are interconnected.

2. Detailed design: The functionality for all the modules was described and the interface between all modules was defined.

3. Verilog/VHDL code writing: The Verilog/VHDL code was written based on the detailed design. Further, a test bench for each module was designed. There was also designed a test bench for the overall FPGA design.

4. Simulation: All modules are simulated and verified against the detail design specification. As well as the entire FPGA was simulated.

5. Implementation: The implementation of the FPGA is primarily a tool driven task, the actual synthesis and PAR (place and route) is done by the development tools. However the tools operate based on the input from the designer, the primary input is the VHDL code and the UCF file (user constraint file). The output is a bit file that can be loaded into a FPGA.

After the last step the creation of the artefact is complete. Another output from this activity was the user guide, which is a manual how to control the FPGA from a user perspective. The user guide is often referred to as the programmer's reference, since the user of a FPGA is typically a piece of software that controls the FPGA. The design and development activity also output "how-to knowledge", since the process of designing and implementing a solution gives the researcher knowledge on how to use the artefact.

Activity 4: Demonstration

The artefact (The FPGA design) was used to perform the following measurements:

- Verification
- Throughput rate
- Latency
- Data integrity
- Power consumption.

The artefact provides a complete research platform. The test data is loaded into the FPGA through the host interface. The encryption is initiated by the start command and the results are read out from the registers in the analysis module. Throughput rate, latency, data integrity is all

measured by the analyze module. However the FPGA is not able to measure its own power consumptions, this was done external with an oscilloscope and a current clamp probe.

Activity 5: Evaluation

The measurements from the previous activity were used to determine if the objectives for the project had been reached. This activity shows the effectiveness of the solutions, and the result was used to improve the design. The design science research methodology allows us to do iterations; therefore the knowledge from this activity was used to perform another iteration of the design and development, in respect to optimize throughput, latency and power consumption. This does not mean that the whole design and development was redone; only adjustment and optimizations were performed. At each of the iterations the Demonstration activity was redone, since a new design gave new results. The output from this activity is new knowledge to the field of information security.

Activity 6: Communication

This is out of the scope for this project, nevertheless the results from this studies could be considered for publication in a scientific journal.

4.2.2. Hardware Design Approaches

There are two fundamental approaches to hardware design: high-level as well as low-level designs. The high-level structural design, which is language-dependent, tends to be better because of practical considerations. The consequence of high-level implementations are relatively easy to construct, however may not be as optimal as compared to schematic based (low level) design methodology. All of the five hardware implementations of AES finalists were language based high-level structural designs. In the majority cases, the given performance statistics were the end result of device simulation, rather than measurements carried out on the original devices. So the tools used for designing and simulating ASICs and FPGAs are must be reliable and mature. However, the tools use conventional design assumptions and rules. Therefore the achievable clock rates of actual devices may exceeds the prediction of the device simulations sometimes. One more consideration is the design goals.

Achievable goals include: Minimum area, maximum throughput with unlimited area, maximum throughput within a predefined area and maximum efficiency, as measured by throughput per area. In general, the objective determines the design approaches; different goals frequently produce mismatched design choices. The area minimization as well as speed maximization is simultaneously not achievable. The maximization of speed may have an effect on efficiency. For example, complete loop unrolling may maximize throughput except increase the required area and decrease efficiency. However, in the non-feedback mode of operation, pipelining may maximize throughput but maintain efficiency essentially constant or increase efficiency.

4.2.3. Field Programmable Gate Arrays (FPGAs)

A Field Programmable Gate Array is an IC (integrated circuit) consists of a large two-dimensional array of tiny computing units which can be reconfigurable. Information can be routed in the array, vertically or horizontally. Changing links in between the blocks can result re-routing. This kind of hardware has the advantages in terms of flexibility, low implementation cost, and low production cost for relatively low capacity devices. In general, there is a possibility to re-program an FPGA to toggle from one cryptographic algorithm to other algorithm, from key setup to decryption or encryption. Reconfiguration requires a fraction of a second. However, the flexibility advantages are trade off against speeds lower than those achievable by non-reprogrammable hardware devices such as ASICs. On the other hand, FPGAs can attain speeds significantly better than software implementation. The smallest computing units of a Field Programmable Gate Arrays are called CLBs (Configurable Logic Blocks). Reconfiguration alters the function of the CLBs and the links between them. A CLB normally consisting of lookup tables (LUTs) and flip-flops (FFs). The LUTs are programmed as either a minimum number of combinational logic gates or a tiny RAM. An FPGA may also include fixed block RAMs which can be used as either LUTs or memory blocks as shown in Fig. 4.2. On the other hand, there is a significant dissimilarity between FPGAs, and the use of fixed block RAMs may have an effect on universality as well as portability of the outcomes. And also RAM has slower access time than Configurable Logic Blocks.

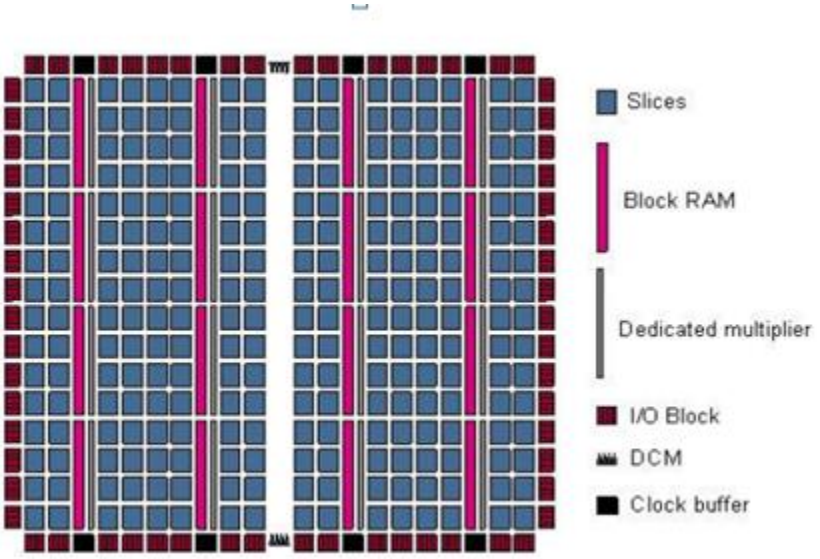

Fig. 4.2 FPGA Architecture

4.2.4. Verilog Hardware Description Language:

Verilog Hardware Description Languages is one of the two main HDL languages used in IC designs. The other hardware description language is VHDL. Hardware Description Language permits the design to be simulated in the design cycle earlier, in order to rectify faults or experimentation with different architectures. Implementations depicted in Hardware Description Language are independent of the technology, very easy to debug and design. And these type of designs typically more clear than schematic, predominantly for bulky circuits. In verilog the designs can be described in four abstraction levels; 1.Algorithmic level (analogous to c language). 2. Register transfer level (RTL uses registers linked by Boolean expressions). 3. Gate level (pre defined AND, OR, NOT etc. gates). 4. Switch level (uses MOS transistors like PMOS and NMOS). These VHDL and Verilog HDL languages differ from software programming languages since they include ways of describing the sensitivity (signal strengths) and propagation time. The designers of Verilog need a language with syntax analogous to the C programming language, which was already extensively used in engineering software development. Verilog is case-sensitive like C and has a basic pre-processor (less sophisticated than that of ANSI C/C++). Its control flow keywords (if/else, for, while, case, etc.) are equivalent, and its operator precedence is compatible with C language. Syntax differences include: required bit-widths for variable declarations, separation of procedural blocks (Verilog

uses begin/end as a substitute to curly braces {}), and several other slight differences. Verilog requires that variables be given a specific size. In C language these sizes are assumed from the 'type' of the variable (for instance an integer type size may be 8 bits). Verilog consists two types of assignment operators; a non-blocking assignment operator (<=), and a blocking assignment operator (=). The non-blocking assignment permits designer to illustrate a state-machine update without need to declare and use temporary storage variables. As these concepts are part of Verilog HDL language semantics, designer could quickly write descriptions of bulky circuits in a comparatively compact and brief form. At the time of Verilog's introduction (1984), Verilog represented a remarkable productivity enhancement for circuit designers who were already using graphical schematic based capture software and particularly written software programs to document and simulate electronic circuits

A Verilog HDL design consists hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared output, input, and bidirectional ports. Inside, a module can have any combination of the following: net/variable declarations (wire, register, integer, etc.), sequential and concurrent statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed within the begin and end blocks and executed in sequential manner inside the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language. Verilog has a concept of 'wire' consists of both signal values (“1, 0, floating (z), undefined (x) ") and signal strengths (strong, weak, etc.). This system allows abstract modeling of common signal lines, where multiple sources drive a common net. When a wire has multiple drivers, the wire's (readable) value is determined by a function of the source drivers and their strengths. A subset of statements in the Verilog language is synthesizable. Verilog modules be conventional to a synthesizable coding style, known as RTL (register-transfer level), can be physically realized by synthesis software. Synthesis software algorithmically transforms the Verilog source code into a net list, a logically equivalent description consists only of elementary logic primitives (AND, OR, NOT, flip-flops, etc.) that are accessible in a specific FPGA or VLSI technology. Further manipulations to the net list eventually lead to a circuit manufacture blueprint (such as a photo mask set for an ASIC or a bit stream file for an FPGA).

4.2.4 Factors of hardware design:

Throughput: It is defined as the amount of bits processed per unit time after the process has vanished through initialization, and generally expressed in Mbps or Gbps. It is represented by character R_t.

Latency: It is defined as the number of clock cycles required process one complete one block of information. This is the time between an instant when a chunk of data enters into the encryption unit, and an instant when it is come out of the same encryption unit. It is represented by $T_{L.}$

Area: Area illustrated the dimension of the circuit. There are different ways of expressing area based on the technology. In the ASIC technology, area is stated in terms of a die [um^2], or in terms of the amount of logic gates or transistors. In the FPGA technology, the size of the circuit is stated in terms of the number of CLBs.

Work size: It is defined as the quantity of data to be encrypted, it measured in bits and denoted by the letter W_s.

Processing time: It is defined by three parameters namely throughput, work size and the latency. The processing time is represented $t_{process}$ and is calculated by the given formula

$$\mathbf{t_{process} = W_s/R_t + T_L}$$

Efficiency: The efficiency of an AES implementation using FPGA is defined as the ratio of throughput to the area of the given FPGA. The area is equal to number of FPGA slices. Efficiency is measured in Mbps/slice. It is represented by the symbol η_E.

$$\mathbf{\eta_E = R/area}$$

Power Performance: The power performance for an algorithm measured in terms of throughput per Watt, units are (Gbps/w), and it is represented by $\mathbf{P_p}$.

4.3. Research Objective

The most important objectives of AES hardware implementation are:

1. Introduce new mathematical techniques for the AES algorithm design which reduces cost of the hardware implementation.
2. Increase throughput of the system by parallel processing of the data, using pipelining techniques and also by efficient utilization of BRAMs present in the target device.
3. Reduce area of the design by sharing or merging the recurring operational blocks and also by relocating the different modules present in the design.
4. Study how to efficiently implement AES in Xilinx Artix-7 architecture.
5. Analyze throughput rate, power consumption, Latency and data integrity parameters to test the performance of the AES design

CHAPTER 5

DESIGN AND IMPLEMENTATION

5.1. Proposed AES Architecture

In order to get maximum speed and lesser area by mapping all the four round transformations of AES to LUTs, ROMs and Block RAMs. And use different techniques to provide necessary Data-integrity. Therefore the proposed AES architecture has four modules 1.Key Generation Module 2.Encryption Module 3.Decryption Module. 4. Data-integrity module. Fig. 5.1 represents AES architecture

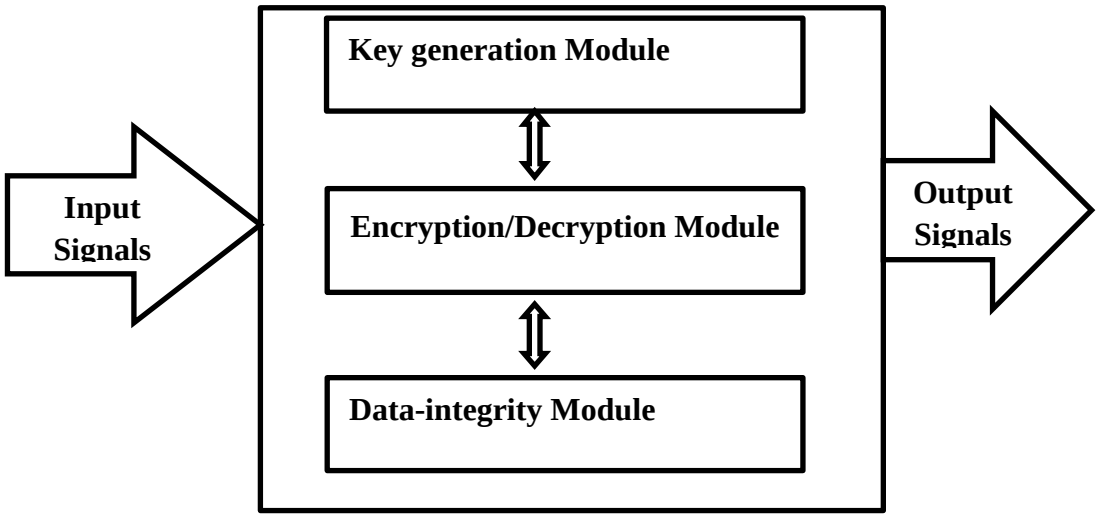

Fig. 5.1 AES architecture

The AES Architecture contains key generation module and data-integrity module are common units for both encryption and decryption. Key generation module gives necessary key expansion for both encryption and decryption functions. The key generation module consists of key register of 128 bits, S-Box and XOR gates for bitwise XOR operation. It is designed to produce round keys on each positive edge of the clock, when it is enabled. The key generation architecture does not require any hardware for shift operation and the port mapping between key register and S-Box is done according to the required shift. Hence this offers the advantage in area. Also in this bits are rearranged on data path from register to S-Box and the round constant required for each rounds are stored in ROM and retrieved on each clock. The function of data integrity module is different for both encryption and decryption. This module provides necessary data-integrity by adding some additional logic to the cipher text during encryption process and sends it to the

decryption unit. Where as in decryption process data-integrity module extract the additional logic from the cipher text and verifies whether the given data is modified by attacker or not. If the data is modified it shows a message to receiver that original information is altered ,otherwise it passes the cipher text to the decryption unit to get plain text from the given cipher text.

5.1.1. Block diagram of Key Generation Unit

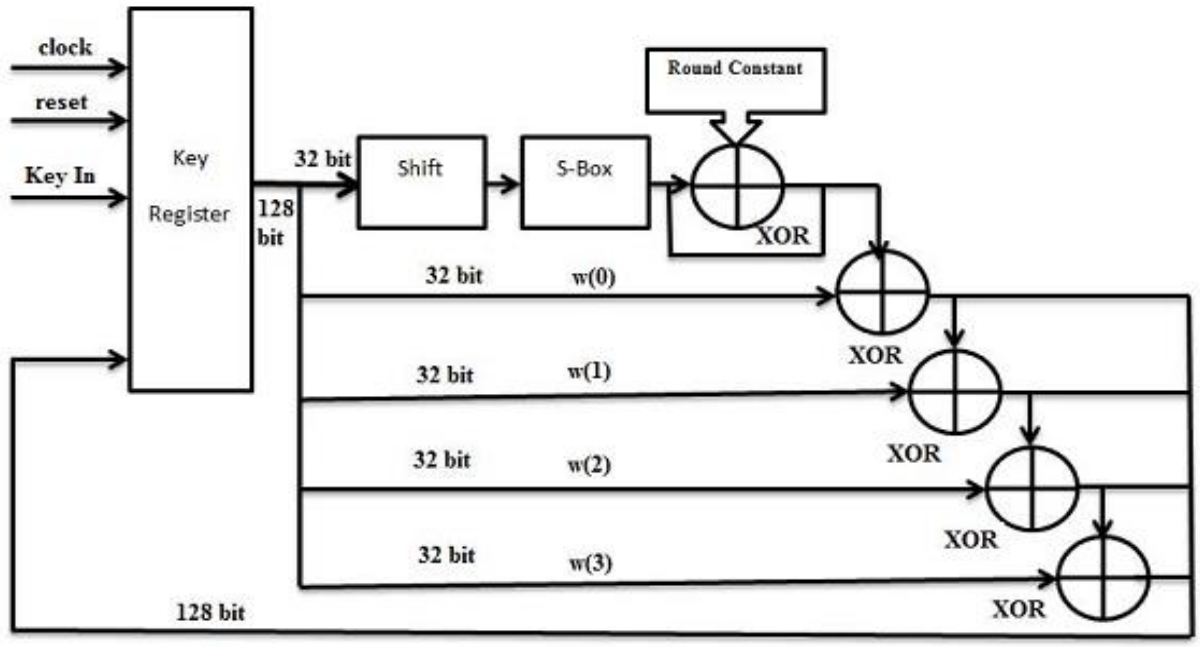

Fig.5.1.1 Key generation unit

The encryption module takes 128 bit text to be encrypted and receives round key from key generation module to do each round of encryption. Fig. 6(a) represents the proposed encryption module. Start, stop mix, terminate are control signal produced by the control unit. The "done' signal is provided to indicate that encryption is done. Architecture is as shown in Fig. 6. In the proposed work for reducing the hardware of entire architecture, the control unit of encryption module is not designed separately. The control unit of key generation module which is a 4-bit counter is designed to control the entire functioning of encryption module. The sharing of control unit by both encryption and round key generation gives unique advantage of reduction in hardware as compared to other implementations.

5.1.2. Block diagram of encryption unit:

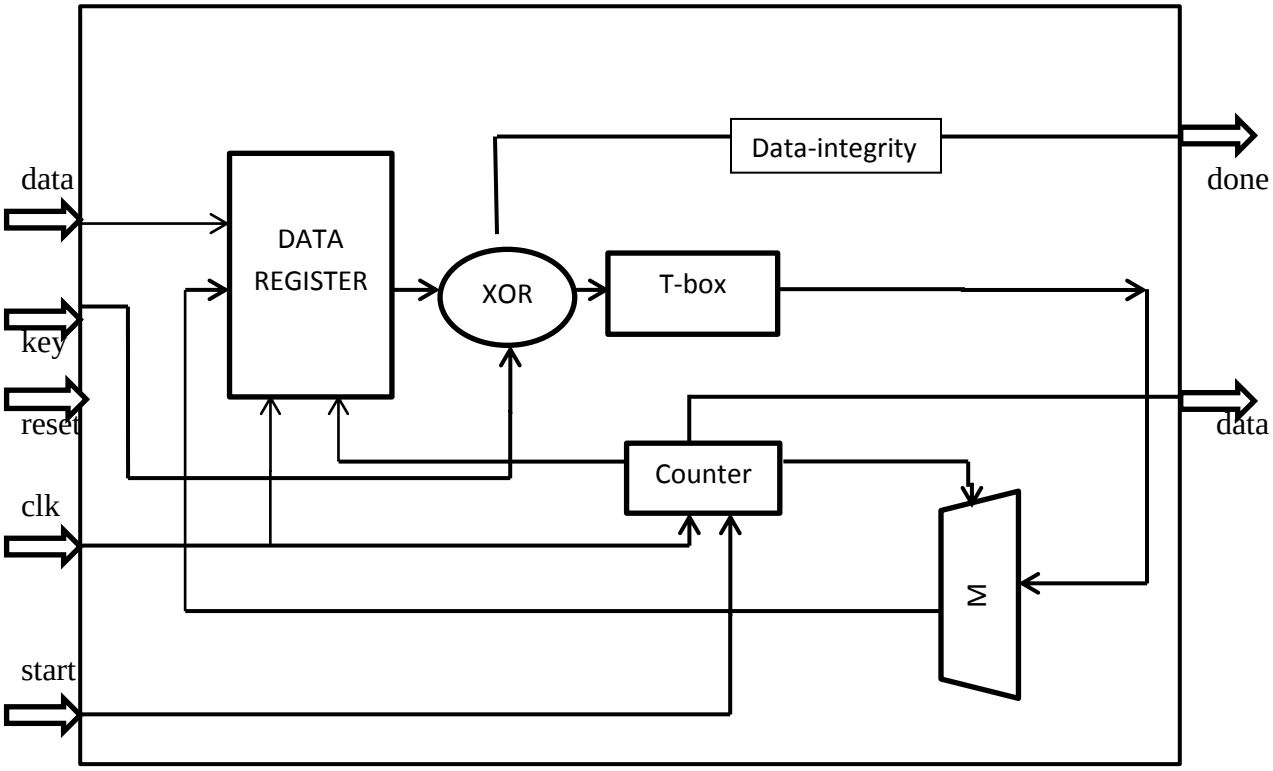

Fig. 5.2 Encryption module

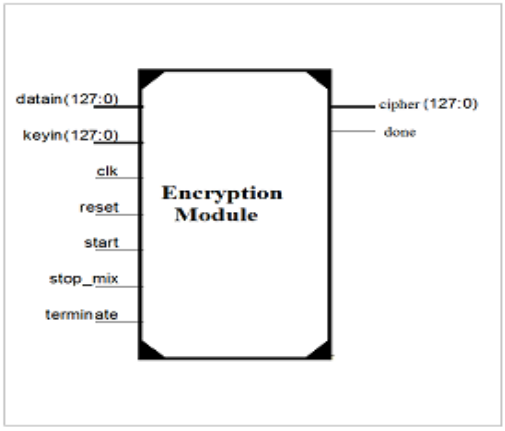

Fig.5.2.1 Encryption module

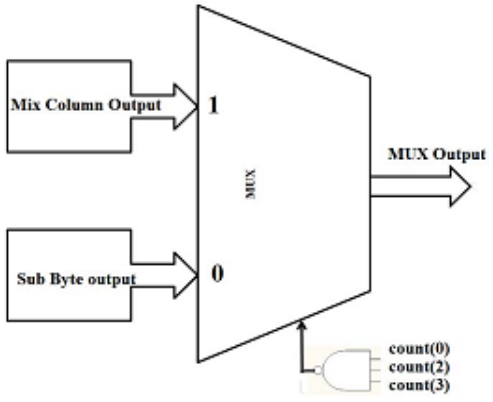

Fig.5.2.2 Nand gate to eliminate last round

NAND gate shown in fig. 5.2(b) and the 4-bit counter (Controller) are used to set and reset selection line of Multiplexer. For count one to ten the selection line will be in set condition and multiplexer will pass Mix Column output. However on last round, count will be eleven so

selection line will reset and pass Sub Byte output. Shift Row operation is designed in such a way that it does not take any hardware. After Round Key operation data is given to S-Box with required shift by port mapping the signal according to required shift in verilog HDL description of the design. Since there is no hardware for Shift Row operation design gets the advantage of area, power and speed. Architecture of Decryption module is same as encryption module with all complimentary functions of encryption module. In this proposed work, the T-box approach allows the computation of the entire iteration of AES using only table look-ups called T-boxes and XOR operations. These pre-computed T-box look up tables represent the combined operation of the sub bytes and the mix column transformations. Compared to the 8x8 bits wide S-box look up tables, the T-box tables are of the size of 8 x 32 bits. Therefore memory requirement is more than the S-Box approach. But T-box based implementation decreases the computation time require for two operations likely substitute byte and mix columns. This implementation gives higher throughput for the design by significantly decreasing delay in data path. As a result the proposed design takes lesser number of slices when compared with other combinational technique proposed. The mathematical description described below explains how T-box tables and the corresponding AES round operations are obtained:

5.2 T-box Tables derivation

State Input is

S_0	S_4	S_8	S_{12}
S_1	S_5	S_9	S_{13}
S_2	S_6	S_{10}	S_{14}
S_3	S_7	S_{11}	S_{15}

Mix column encryption operation to the first column of state input

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{pmatrix} =$$

$$\begin{pmatrix} 02*S_0 + 03*S_1 + 01*S_2 + 01*S_3 \\ 01*S_0 + 02*S_1 + 03*S_2 + 01*S_3 \\ 01*S_0 + 01*S_1 + 02*S_2 + 03*S_3 \\ 03*S_0 + 01*S_1 + 01*S_2 + 02*S_3 \end{pmatrix}$$

S4, S8, S12 First row elements → T_0

S5, S9, S13 Second Row → T_1

S6, S10, S14 Third Row → T_2

S7, S11, S15 Fourth Row → T_3

T-box and inverse T-box tables are as shown in Fig. 5.3

$$T_0[a] = \begin{pmatrix} 2.S[a] \\ S[a] \\ S[a] \\ 3.S[a] \end{pmatrix} \quad T_1[a] = \begin{pmatrix} 3.S[a] \\ 2.S[a] \\ S[a] \\ S[a] \end{pmatrix} \quad T_2[a] = \begin{pmatrix} S[a] \\ 3.S[a] \\ 2.S[a] \\ S[a] \end{pmatrix} \quad T_3[a] = \begin{pmatrix} S[a] \\ S[a] \\ 3.S[a] \\ 2.S[a] \end{pmatrix}$$

$$T_0^{-1}[a] = \begin{pmatrix} 0e.S^{-1}[a] \\ 09.S^{-1}[a] \\ 0d.S^{-1}[a] \\ 0b.S^{-1}[a] \end{pmatrix} \quad T_1^{-1}[a] = \begin{pmatrix} 0b.S^{-1}[a] \\ 0e.S^{-1}[a] \\ 09.S^{-1}[a] \\ 0d.S^{-1}[a] \end{pmatrix} \quad T_0^{-1}[a] = \begin{pmatrix} 0d.S^{-1}[a] \\ 0b.S^{-1}[a] \\ 0e.S^{-1}[a] \\ 09.S^{-1}[a] \end{pmatrix} \quad T_0^{-1}[a] = \begin{pmatrix} 09.S^{-1}[a] \\ 0d.S^{-1}[a] \\ 0b.S^{-1}[a] \\ 0e.S^{-1}[a] \end{pmatrix}$$

Fig. 5.3 T-box and inverse T-box

In the above resultant matrix representation each column has fixed constants except that state inputs are updated with next row elements for the following iteration. First row elements of state input are always multiplied with first column constants of the multiplication matrix. Similarly, the second row elements with second column of matrix and so on. The first row of the state input uses table T_0, the second row uses the table T_1, the third row uses the table T_2 and the fourth row uses the table T_3.

5.2.1. Round functions computation:

These are 4 tables with 256 4-byte word entries and make up for 4KByte of total space. Using these tables, the round transformation can be expressed as:

$$e_j = \mathrm{T}_0\left[a_{0,j}\right] \oplus \mathrm{T}_1\left[a_{1,j-C1}\right] \oplus \mathrm{T}_2\left[a_{2,j-C2}\right] \oplus \mathrm{T}_3\left[a_{3,j-C3}\right]$$

Hence, a table-lookup implementation with 4 Kbytes of tables takes only 4 table lookups and 4 EXORs per column per round. It can be seen that $T_i[a]$ = Rot Byte ($T_{i-1}[a]$). At the cost of 3 additional rotations per round per column, the table-lookup implementation can be realised with only one table, i.e., with a total table size of 1KByte. We have

$$e_j = \mathrm{T}_0\left[b_{0,j}\right] \oplus \mathrm{Rotbyte}(\mathrm{T}_0\left[b_{1,j-C1}\right] \oplus \mathrm{Rotbyte}(\mathrm{T}_0\left[b_{2,j-C2}\right] \oplus \mathrm{Rotbyte}(\mathrm{T}_0\left[b_{3,j-C3}\right])$$

The code-size (relevant in applets) can be kept small by including code to generate the tables instead of the tables themselves. In the final round, there is no Mix Column operation. This boils down to the fact that the S table must be used instead of the T tables. The need for additional tables can be suppressed by extracting the S table from the T tables by masking while executing the final round.

5.3. Encryption and Decryption in T-box architecture:

Round outputs of encryption and decryption iterations are computed from the look-up tables. Last round in both the cases is performed in a special way.

5.3.1. Last round in Encryption

Since mix columns operation has to be eliminated in the last round, this round is treated differently. In this round, sub byte values are needed instead of T-box outputs. Additional memory space is not needed to implement an additional sub byte table, as one byte outputs of the sub byte transformation can be extracted from the four-byte outputs of the T-box transformation corresponding to the same one-byte input as:

$$S[a]=byte(1,T_0[a])=byte(2,T_1[a])=byte(3,T_2[a])=byte(0,T_3[a])$$

5.3.2. Last round in Decryption

In the last round of decryption, the Inv Mix columns operation is not executed. As a result, the outputs of the Inv sub bytes transformation $S^{-1}[a]$ must be computed. Given the value of$T_0^{-1}[a]$, $S^{-1}[a]$ can be computed as follows:

$S^{-1}[a] = 0E^{-1}.\text{byte}(0, T_0^{-1}[a]) = 09^{-1}.\text{byte}(1, T_0^{-1}[a]) = 0D^{-1}.\text{byte}(2, T_0^{-1}[a]) = 0B^{-1}.\text{byte}(3, T_0^{-1}[a])$

Where byte (n, X) represents the n^{th} byte of a variable X. Based on the above equations each bit of $S^{-1}[a]$ can be computed using four different equations, each giving the same value.

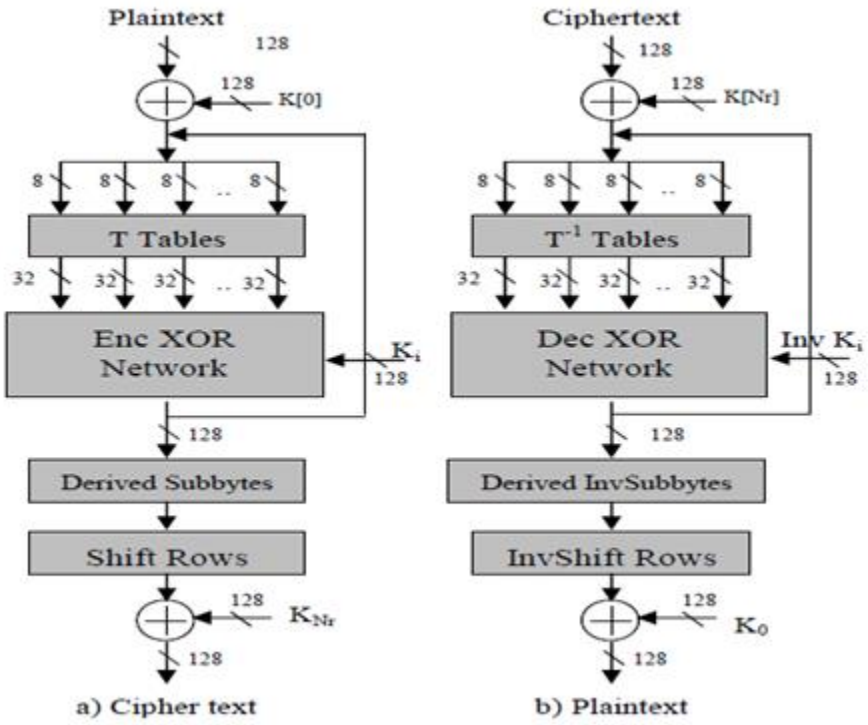

Fig. 5.4 Structure of T-box AES architecture

Decryption algorithm shown in figure.2 for the T-box differs from that of the standard decryption. Except for the first and the last round keys, inverse round keys are used in the intermediate iterations in this decryption algorithm. Round modification for decryption is shown in figure 3.

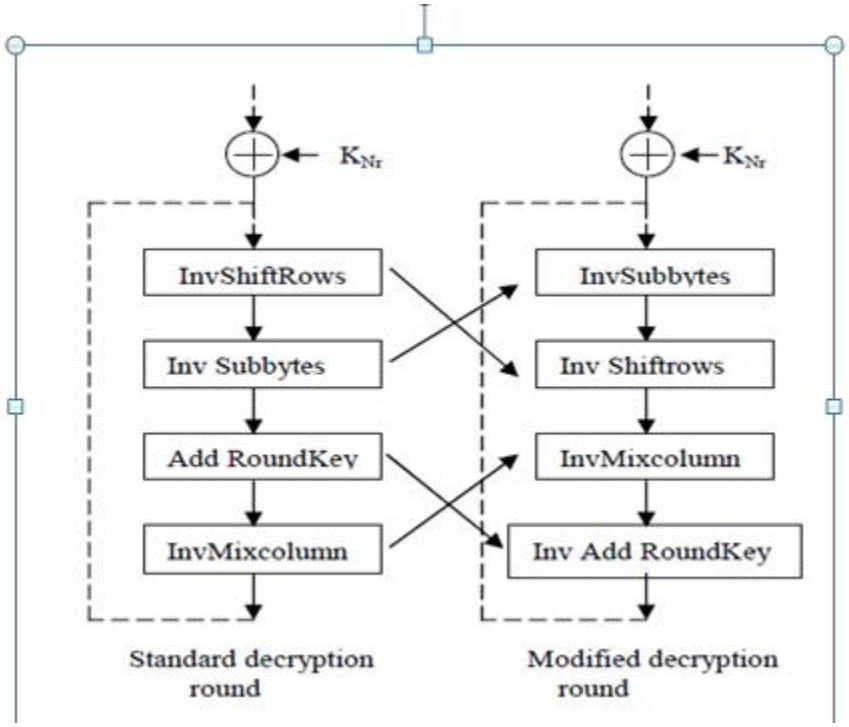

Fig. 5.5 Modified Decryption Algotithm

The operations Inverse Shift Rows and Inverse Sub bytes can be swapped without affecting the result and since Inverse Mix column is a linear transformation, the following equation is valid.

Inverse Mix column (d $\oplus$ K) = Inverse Mix Column (d) $\oplus$ Inverse Mix Column (K)

The reason for round modification in the case of T-box decryption is because the T-box outputs include sub bytes and mix column operations. This modification is very similar to the encryption round operation sequence and hence this method is appropriate for efficient implementation of this architecture.

5.4 Modified T-box table used in the implementation

In this architecture, input data is fed through the multiplexer and it is stored in the register and then passed through a one round combinational logic, the result of the combinational logic is fed back to the circuit through the multiplexer and stored in the register. This architecture can only encrypt one block of data at a time and the number of clock cycles necessary to encrypt a single block of data is equal to the total number of cipher rounds. The critical path is located in the decryption circuit and includes T-boxes lookups, XOR network and Inv Mix column for the

round key. This architecture also requires 11, 13, and 15 clock cycles in order to process one block of data for 128, 192, and 256-bit keys respectively. In the T-Box approach, each byte is multiplied with 2, 1, 1 and 3. A new approach was developed and presented to store the T-boxes. Thus, the T- Box tables are in the form of 32-bits. The multiplication with 1 over GF (2^8) is done only once due to which the T-Box tables are stored in the form of 24 bits . T-Box tables are stored in the form of 24 bits as shown below.

$$T[a] = \begin{pmatrix} S[a] \\ 02.S[a] \\ 0\ 3.S[a] \end{pmatrix}$$

In our implementation of AES algorithm modified T-box approach is adopted for encryption and decryption both. As discussed above 24-bit T-boxes are used for encryption. For decryption we arranged inverse T-boxes in such a manner so that it consists of 24-bit instead of 32-bit. Hence, it reduces memory required for both encryption and decryption

CHAPTER 6

RESULTS AND DISCUSSION

This chapter describes the evaluation of the results obtained from this design. The purpose is to evaluate if the project objectives has been achieved. The evaluation is also used to identify if we were able to generate new scientific knowledge.

6.1. Outcomes of proposed AES Algorithm

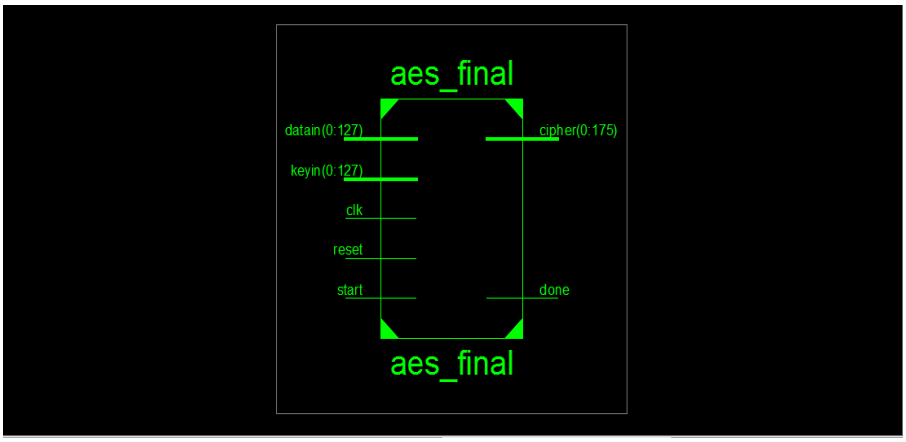

Fig. 6.1 Encryption Module

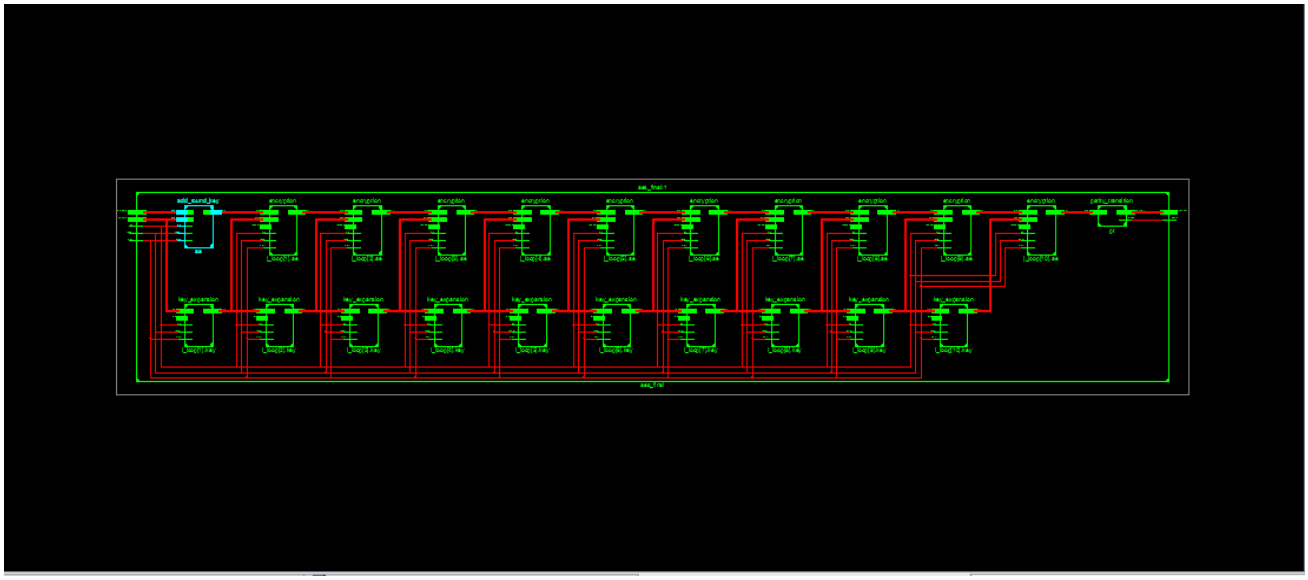

Fig. 6.2 Encryption RTL Schematic

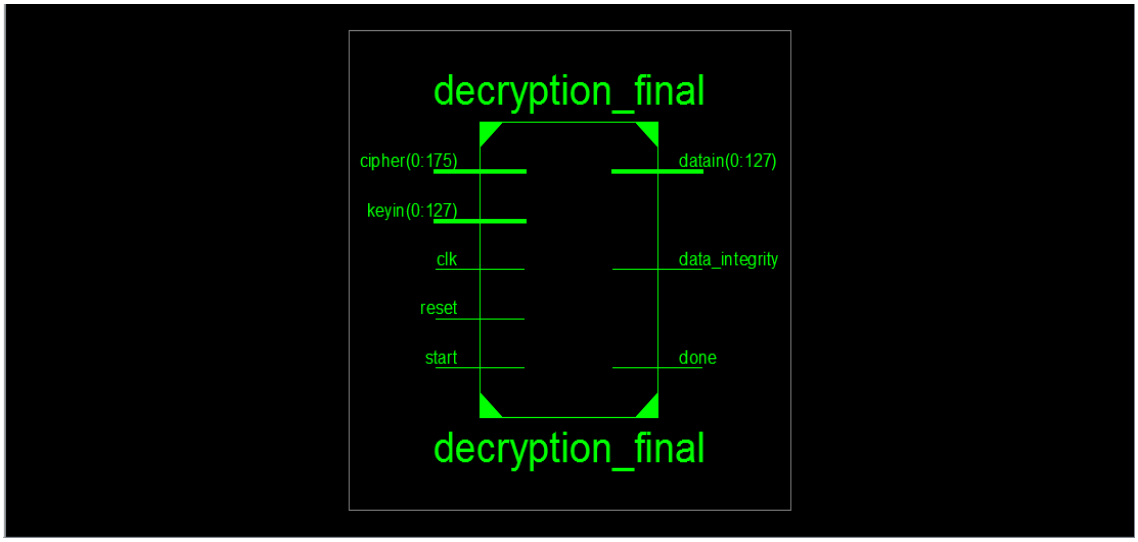

Fig. 6.3 Decryption Module

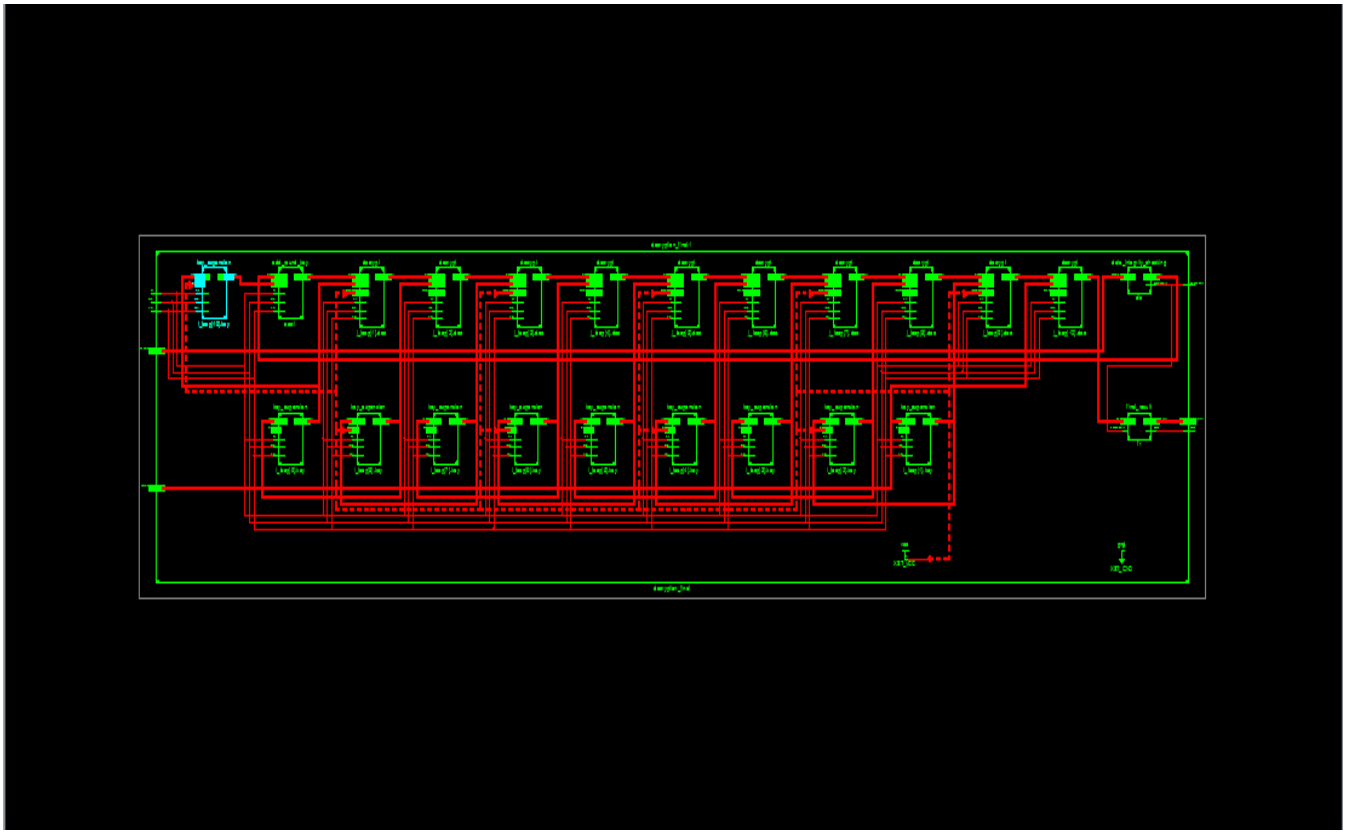

Fig. 6.4 Decryption RTL Schematic

6.2. Simulation Results

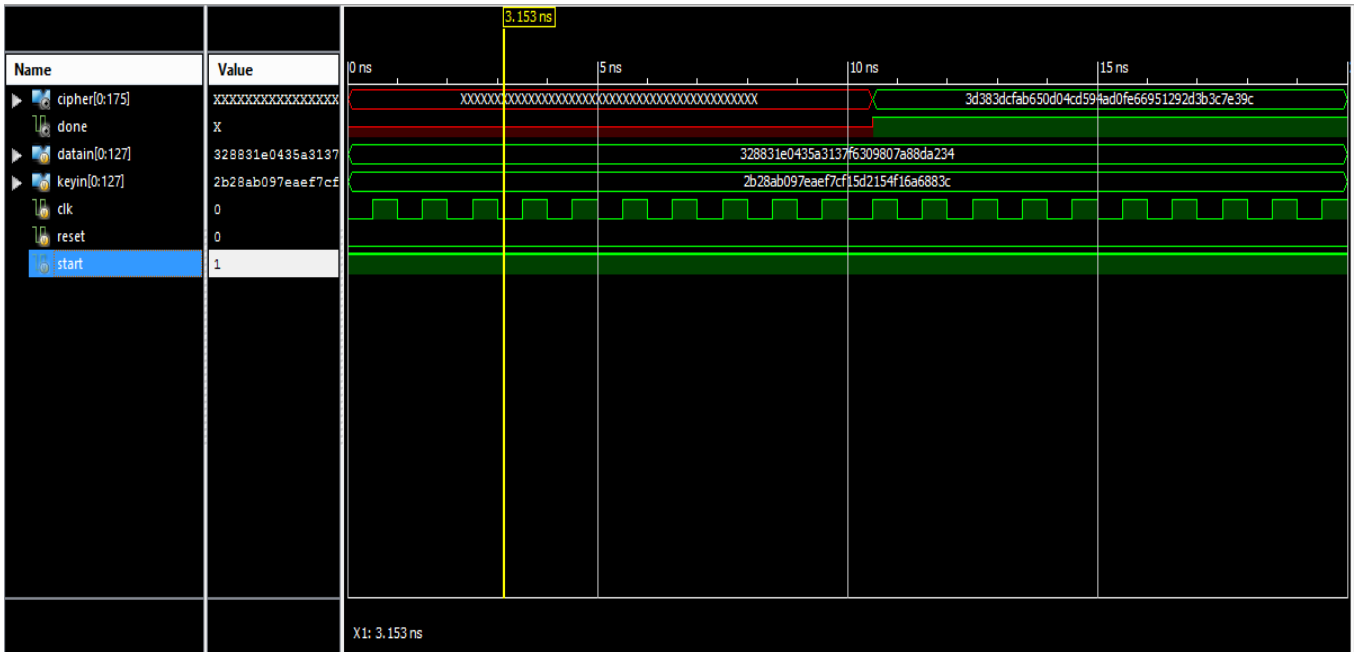

Fig. 6.5 AES Encryption Simulation Waveform

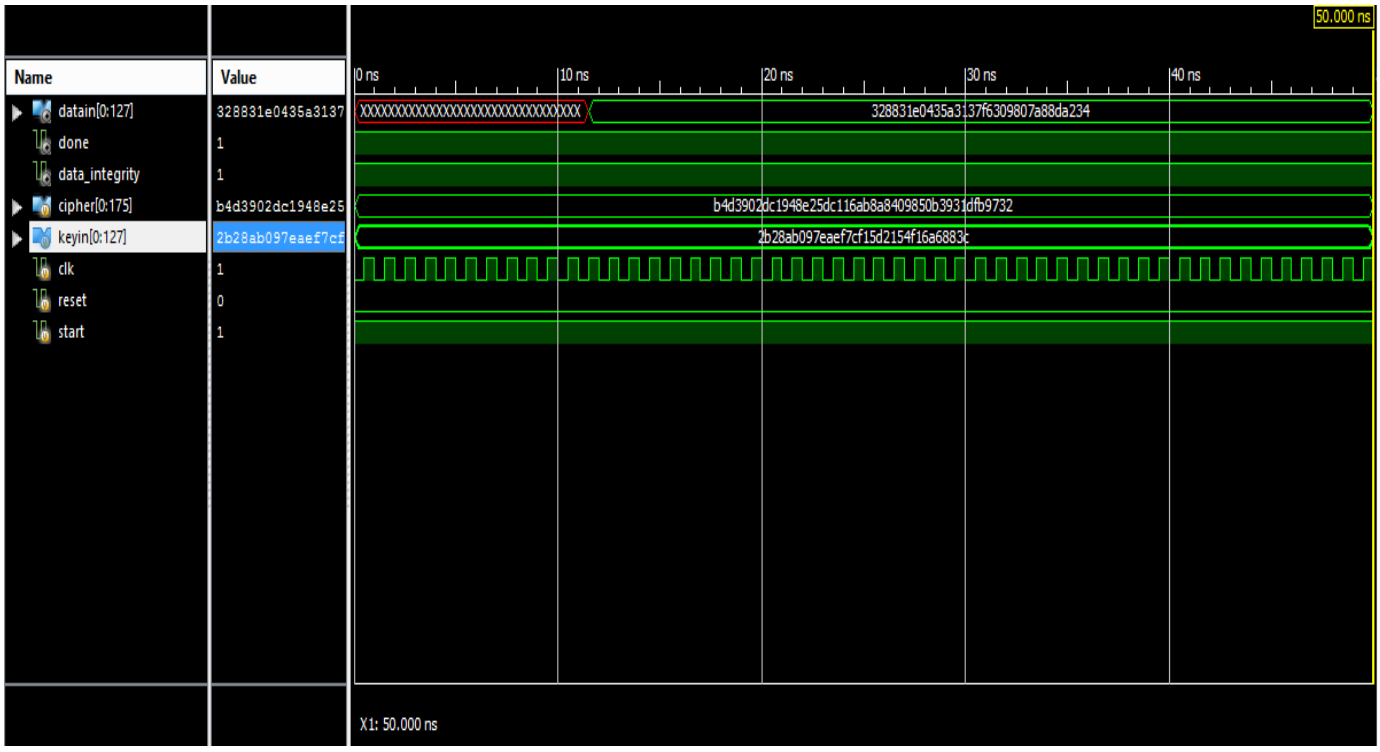

Fig. 6.6 AES Decryption Simulation Waveform

6.3. Performance Parameters

1. Verification: This design is verified by using Xilinx 14.7 ISIM simulator tool. Simulation results for both encryption and decryption process mentioned above.

2. Latency: It is equal to number of clock cycles required to process one block of information. For encryption process as well as decryption process this design requires latency of 10 clock cycles respectively. Whereas previous design requires 10 clock cycles for encryption and 20 clock cycles for decryption. So, there is an improvement in the latency for decryption of data

3. Throughput: Throughput for the both encryption and decryption is calculated by the given formula.

Throughput = (Block Size * Clock Frequency) / (Latency)

Block size =128 bits, Clock frequency =335.818MHz and Latency =10 clock cycles.

All three parameters are equal for both encryption and decryption. Then achieved throughput also same. Throughput =4.2984Gbps. Whereas for previous design, throughput=3.74Gbps for encryption and 1.87Gbps for decryption, therefore proposed design achieves a throughput greater than already reported design.

4. Area: In FPGA technology, the area required for the implementation is equal to the number CLB slices used in that design. This design uses pair of 1407 slices for encryption and decryption respectively.

5. Data-integrity: The proposed design providing data-integrity by adding some additional logic, so that there is an improvement in security provided by AES. While the previous designs does not provide any data-integrity. This is an added advantage for this design

6. Power consumption: It is measured by using Xilinx 14.7 XPower analyzer tool for the given Clock frequency.

The performance of this AES algorithm design using S-box and T-box approach in Artix-7 FPGA summarized in table 6.1.

Table 6.1: comparison of performance parameters of AES algorithm design using S-box and T-box approach

Parameter	Encryption (Using S-box)	Encryption (Using T-box)	Decryption (Using S-box)	Decryption (Using T-box)
Latency (in clock cycles)	10	10	20	10
Throughput (in Gbps)	3.74	4.2984	1.87	3.2984
Data-integrity	No	Yes	No	Yes
Power – consumption	more	less	more	less
verification	done	done	done	done

Table 6.2: Device Utilization Summary for encryption

Device Utilization Summary (estimated values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slice Registers	1408	269200	0%
Number of Slice LUTs	3909	134600	2%
Number of fully used LUT-FF pairs	1407	3910	35%
Number of bonded IOBs	436	500	87%
Number of Block RAM/FIFO	89	365	24%
Number of BUFG/BUFGCTRLs	1	32	3%

Table 6.3: Device Utilization Summary for decryption

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	1280	269200	0%
Number of Slice LUTs	8668	134600	6%
Number of fully used LUT-FF pairs	1279	8669	14%
Number of bonded IOBs	437	500	87%
Number of Block RAM/FIFO	126	365	34%
Number of BUFG/BUFGCTRLs	1	32	3%

6.4. Power consumption report for encryption using Xilinx 14.7 Xpower analyzer at 100MHz clock frequency

Device	
Family	Artix7
Part	xc7a200t
Package	ffg1156
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.009	1	---	---
Logic	0.061	2996	134600	2
Signals	0.113	8369	---	---
BRAMs	0.091	*	*	*
IOs	0.066	436	500	87
Leakage	0.117			
Total	0.457			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.5	84.3	25.7

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	0.305	0.270	0.035
Vccaux	1.800	0.024	0.004	0.020
Vcco18	1.800	0.036	0.031	0.005
Vccbram	1.000	0.009	0.007	0.002
Vccadc	1.710	0.020	0.000	0.020

	Total	Dynamic	Quiescent
Supply Power (W)	0.457	0.340	0.117

6.4. Power consumption report for decryption using Xilinx 14.7 Xpower analyzer at 100MHz clock frequency

Device	
Family	Artix7
Part	xc7a200t
Package	ffg1156
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.013	1	---	---
Logic	0.323	6655	134600	5
Signals	0.527	14108	---	---
BRAMs	0.128	*	*	*
IOs	0.078	437	500	87
Leakage	0.120			
Total	1.188			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	1.5	83.3	26.7

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.000	1.023	0.986	0.037
Vccaux	1.800	0.025	0.004	0.020
Vcco18	1.800	0.041	0.036	0.005
Vccbram	1.000	0.013	0.010	0.003
Vccadc	1.710	0.020	0.000	0.020

	Total	Dynamic	Quiescent
Supply Power (W)	1.188	1.069	0.120

FUTURESCOPE

The implementation results are shows that 128 bit AES is considered to be a very secure encryption method. However, this design has some deterministic nature of mathematical Operation. For example, a specific encryption key and a specific input vector always result in the same corresponding output vector. Therefore this type of AES design is not well suited for biometric image applications for several reasons: like shapes are not fully hidden in the picture. For that reason a new design and development is needed to address this problem. By using four modes: Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter (CTR), we can make the FPGA applicable for a biometric image application From hardware perspective, the CBC mode is the cheapest to implement. The principle is to form a chain, the output from the AES encryption is mixed (XOR) with the input vector for the next output calculation. The output vector will therefore not only be depended on the encryption key but also on previous outputs. For the first input vector is there no previous output vector, instead an initialization vector is used. This vector can be randomly generated for each block of data. The initialization vector is not secret; it can be attached to header of the image or transmitted separate. Server and Client can also generate the initialization vector locally in a pseudorandom way. The approach is to redesign the FPGA to support AES in CBC mode for higher level security. The other key sizes (AES-192 and AES-256) used mainly for top level security applications.

CONCLUSION

This dissertation report reveals that AES is the better algorithm in terms of performance and security. Although its power consumption is on higher side but it is way less than 3 DES (which not suitable for all platforms). Only DES has less power consumption than AES but on security front DES is the most vulnerable and can be easily broken by brute force attack in merely fifteen hours. The two different architectures (S-box and T-box) implemented for AES 128bit algorithm have different performance characteristics. The T-box implementation outperforms the S-box implementation, giving excellent performance in terms of high throughput, low latency and less power consumption. Also the hardware complexity of T-box implementation is less as compared to S-box approach. According to the results obtained, there is an improvement in performance for T-box implementation since the T-box tables for encryption and decryption are implemented as BRAMs and not as logic in the target Artix-7 FPGA. However, by ensuring that the T-box tables are stored in BRAMs, the desired performance can be achieved with the T-box implementations. This T-box approach achieves a throughput of 4.2984Gbps for both encryption and decryption with latency of 10 clock cycles. Therefore the proposed architecture (using T-box tables) achieves the speed is higher than the already reported systems(using S-box technique), hence the proposed design serves as the best high speed encryption algorithm and is thus suitable for various applications. Power calculations are done by using Xilinx 14.7 X-Power analyzer tool. Moreover with the addition of data-integrity module, the proposed design can offer better security than previous designs. The performance of these architectures can further be improved by using sub-pipelining techniques in the existing designs and converting the S-box look up tables to pure combinational logic. And we can redesign this FPGA implementation to support AES in CBC mode or using different key lengths (AES-192 and AES-256) for higher level security like biometric image applications.

REFERENCES

[1] Abhijith.P.S, Mallika Srivasatava, Aparana Mishra "High Performance Hardware Implementation of AES Using Minimal Resources", IEEE International conference on Intelligent Systems and Signal Processing(ISSP), Allahabad, India, 2013, pp. 338-343.

[2] FIPS-197, NIST - National Institute of Standards and Technology, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," 2001.

[3] M. Goswami and S. Kannojiya, "High Performance FPGA Implementation of AES Algorithm with 128-Bit Keys," IEEE International Conference Ahdvances Computing Comm., vol. 1, Himarpur, India, 2011, pp. 281-286. .

[4] http : //Advanced Encryption Standard-Wikipedia , the free encyclopedia.html.

[5] J. Daeme and V. Rijmen, "AES proposal: Rijndael," NIST AES Proposal, June 1998.

[6] W.Stallings,"Cryptography and network security principles and practice," Pearson edition2009, pp. 135-160.

[7] Mandal, B.K. Bhattacharyya, S.K. Bandyopadhyay, "Designing and Performance Analysis of a Proposed Symmetric Cryptography Algorithm," Communication Systems and Network Technologies (CSNT), pp.453 – 461, 2013.

[8] S. Kaur and R. Vig, "Efficient Implementation of AES Algorithm in FPGA Device," International Conference on Computational Intelligence and Multimedia Applications, 2007, pp. 179 –187.

[9] LeinHarn, Hung-Yu Lin, "A cryptographic key generation scheme for multilevel data security," Computers & Security vol. 9, Issue 6, 1990, pp. 539-546.

[10] P. Chodowiec and K.Gaj. "Very compact FPGA implementation of the AES algorithm". In Proc.5th International, workshop on Cryptographic Hardware and Embedded systems (CHES 2003), pages 319-333, cologne,Germany,Sept,8-10,2003.

[11] Jason Van Dyken, Jose G. Delgado-Frias, "FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm", Journal of Systems Architecture 56 (2013), pages 116-123.

[12] O.Prasanthi, M.Subba Reddy, "Enhanced AES Algorithm", International Journal of Computer Applications in Engineering Sciences vol. II, ISSUE II, June 2012, pp. 114-118.

[13] Dur-e-Shahwar Kundi, Arshad Aziz, Nasar Ikram, "Resource efficient implementation of T-boxes in AES on virtex-5 FPGA", Information Processing Letters 110(2010), pages 373-377.

[14] Manjesh.K.N, R.K. Karunavathi, "Secured High throughput implementation of AES Algorithm", International Journal of Advanced Research in Computer Science and Software Engineering vol 3, Issue 5, May 2015, pages 1193-1198.

[15] L. Thulasimani, M.Madheswaran, "Design And Implementation of Reconfigurable Rijndael Encryption Algorithms For Reconfigurable Mobile Terminals", International Journal on Computer Science and Engineering, vol. 02, No. 04, 2010, pages 1003-1011.

[16] FPGA and ASIC Implementations of AES, by Kris Gaj and Pawel Chodowiec (to be published as a chapter of a textbook on cryptographic Engineering.)

[17] Viktor Fischer and Milos Drutarovsky, " Two methods of Rijndael Implementation in reconfigurable hardware" CHES 2001.

[18] J. Daemen and V. Rijmen. The design of Rijndael: AES – The Advanced Encryption Standard. Number ISBN 3-540-42580-2. Springer-Verlag, 2002.

[19] Ga¨el Rouvroy, Franc¸ois-Xavier Standaert, Jean-Jacques Quisquater and Jean-Didier Legat. "Compact and Efficient Encryption/Decryption Module for FPGA Implementation of the AES Rijndael Very Well Suited for Small Embedded Applications", in Proc. of the International Conference on Information Technology, Coding and Computing (ITCC'04) ,Las Vegas, Nevada, USA, Vol. 2 , April 2004, pp.583-587.

[20] P. Bulens, F. Standaert, J. Quisquater, P. Pellegrin, G. Rouvroy, "Implementation of the AES-128 on Virtex-5 FPGAs", in: AFRICACRYPT 2008,in: Lecture Notes in Computer Science, Vol. 5023, Springer, 2008.

[21] Shastry P.V.S., Somani N., Gadre A., Vispute B. “Rolled Architecture Based Implementation of AES Using T-Box”, in Proc. of IEEE 55th International Midwest Symposium on Circuits and Systems, Boise, ID, USA, 5-8 Aug. 2012, pp.626-630.

[22] Dur-e-Shahwar Kundi, Arshad Aziz, Nasar Ikram, “ Resource efficient implementation of T-Boxes in AES on Virtex 5 FPGA”, Information Processing Letters 110 (2010) 373–377.

[23] Kris Gaj, Powel Chodowiec, “FPGA and ASIC Implementations of AES”, in Crytpographic Engineering, Springer, 2009, pp. 235-294.

[24] Chi-Wu Huang, Chi-Jeng Chang, Mao-Yuan Lin, Hung-Yun Tai. “Compact FPGA Implementation of 32-bits AES Algorithm Using Block RAM”, in Proc. of IEEE Region 10 Conference, Taipei, Taiwan, 2007, pp.1-4.

[25]. Chitu C, Chien D, Chang F., “ A Hardware Implementation in FPGA of the Rijndael Algorithm”, Circuit and system, vol 1, pp 507-10,2002.