IMPLEMENTATION OF ENCRYPTION/DECRYPTION OF DATA USING AES ALGORITHM

A PROJECT REPORT

Submitted by

AYSWARYA. D	**71204106007**
RAGINI. S	**71204106038**
SARANYA. M.K.S	**71204106044**

In partial fulfillment for the award of the degree

Of

BACHELOR OF ENGINEERING

in

ELECTRONICS AND COMMUNICATION ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"IMPLEMENTATION OF ENCRYPTION/DECRYPTION OF DATA USING AES ALGORITHM"** is the bonafide work of "**AYSWARYA. D, RAGINI. S** and **SARANYA. M.K.S** " who carried out the project work under my supervision.

SIGNATURE

Dr. (Mrs.) Rajeswari Mariappan

HEAD OF THE DEPARTMENT

Electronics and Communication Engineering,
Kumaraguru College of Technology,
Chinnavedampatti Post,
Coimbatore-641 006.

16.04.08

SIGNATURE

Ms.A.Vasuki

SUPERVISOR

Assistant Professor,
Electronics and Communication Engineering,
Kumaraguru College of Technolo
Chinnavedampatti Post,
Coimbatore-641 006.

The candidates with University Register Nos: 71204106007, 71204106038 an 71204106044 was examined by us in the project viva-voce examination held o 21.4.2008.

21/4/08

INTERNAL EXAMINER

21/04/08

EXTERNAL EXAMINER

ABSTRACT

Due to the growing need for accessibility to huge amount of data, integrity of data has become essential. AES algorithm is one of the most secure ways to protect data when compared to DES and Triple DES. This project aims at implementing AES algorithm on Field Programmable Gate Array. The project comprises of three modules namely, encryption, decryption and key expansion. Encryption using AES is an iterative process involving multiple passes, by the end of which the input data evolves into a ciphertext. As AES is a symmetric block cipher, decryption involves the same steps as in encryption albeit in the reverse order. One of the steps in these passes is matrix multiplication where we make use of a pre-defined Galois polynomial of a higher order to maintain the size of the resultant matrix. Each pass of encryption and decryption involves the use of a unique key. To facilitate this, the input key is expanded into a larger one in the key expansion module.

The work done includes coding in VHDL, followed by simulation using Modelsim and synthesis using Xilinx ISE. Implementation at software level has been completed on VIRTEX series of Xilinx family.

ACKNOWLEDGEMENT

Our sincere thanks to **Dr. JOSEPH V THANIKAL, M.E., Ph.D., PDF., CEPIT** Principal, Kumaraguru College of Technology, Coimbatore, for the care and pains he had always taken to back us and to ensure that our labs are well equipped with all the required facilities and good maintenance.

We are obliged to thank **Dr. (Mrs.) RAJESWARI MARIAPPAN, M.E., Ph.D., B.Tech.Ed., FIE., MISTE,** Head of the Department, Department of Electronics and Communication Engineering, for the support and encouragement, she has provided us throughout our course.

We are grateful to our faculty guide **Ms.A.Vasuki M.E.,** Assistant Professor, Department of Electronics and Communication Engineering, who gave us a head start, directed and corrected our mistakes with her utmost patience in each stage of our project.

We extend our gratitude to **Mr.S.Govindaraju M.E.,** Professor, Department of Electronics and Communication Engineering, for having been with us providing his valuable suggestions and encouragement.

We would also like to thank **Mr.M.Meiyazhagan,** for providing his assistance in the lab during the course of our project.

Finally, we thank all the teaching and non-teaching staffs of our department for their kind support in our endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DES	Data Encryption Standard
AES	Advanced Encryption Standard
NIST	National Institute of Standards and Technology
GF	Galois Field
HEX	Hexadecimal value

CHAPTER 1
INTRODUCTION

1.1 NEED FOR CRYPTOGRAPHY

In today's electronic age, the importance of cryptography in securing electronic data transactions is unquestionable. Every day, users electronically generate and communicate a large volume of information with others. This information includes medical, financial and legal files, Internet banking, phone conversations and other e-commerce transactions. By using various cryptography techniques, people and organizations can secure electronic information to protect economic interest, prevent fraud and guarantee the privacy of individuals.

Cryptography is a set of protocols, algorithms and techniques providing security for digital information by encoding and decoding the information using a specific key. The cryptographic key controls both the encryption and decryption processes. The two basic objectives of cryptography are:

- **Privacy** to prevent the unauthorized disclosure of data.
- **Authenticity** to prevent the unauthorized modification of data.

Cryptography is the science and study of creating and using systems for communicating in secrecy via communication channels that are not secure. The Cipher Process has been illustrated in Fig 1.1. A sender maintains secrecy by transforming data, known as plaintext into an unintelligible form, known as ciphertext, in a process known as encryption or encipherment. The receiver recovers the original plaintext using the

inverse process of converting ciphertext into plaintext and this procedure is known as decryption or decipherment.

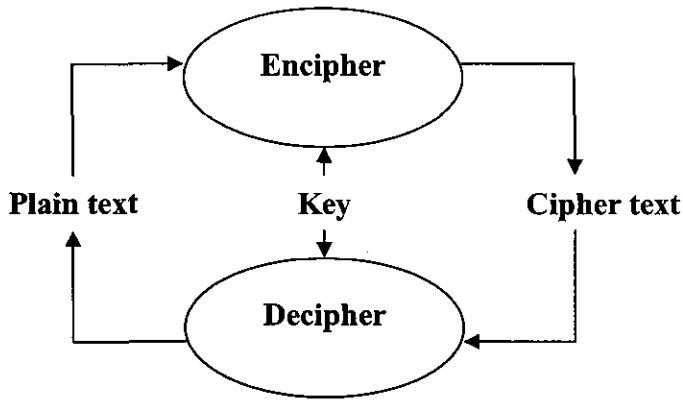


Fig 1.1 Cipher Process

1.2 ENCRYPTION TECHNIQUES

Encryption techniques use complicated algorithms to transform digital information from plaintext to ciphertext. Every time the encryption key is changed, the ciphertext will be different, although the algorithm stays the same. The relationship between the encryption and decryption keys classifies the encryption methods into two distinct categories: symmetric and asymmetric encryption.

1.2.1 Symmetric Encryption

In Symmetric Encryption, the encryption and decryption keys are the same. This method is faster and easier to implement than asymmetric encryption, since the sender and the receiver use the same key to transmit and receive information. In addition, the key sizes are smaller in symmetric encryption compared to asymmetric algorithms. However, for private

exchanging of the key between the sender and receiver, both parties have to agree and trust on a communication medium to exchange the key.

Examples of some common Private Key Encryption algorithms are:

- International Data Encryption Algorithm
- Fast Data Encipherment Algorithm
- Data Encryption Standard
- Triple DES

Symmetric encryption is also divided into two groups: Block and Stream ciphers. Block ciphers work on blocks of data and are commonly used to encrypt the documents, while Stream ciphers operate on bits of data.

1.2.2 Asymmetric Encryption

Asymmetric encryption is a method where the encryption and decryption keys are different. These systems are also called Public Key Encryption Systems, since the encryption key does not have to be a secret. The sender can publish the encryption key and anyone can encrypt messages going to the specific user. However, only the receiver can decrypt the message, since the decryption key cannot be generated with the knowledge of the encryption key. This method is slower and requires more computational power than symmetric encryption.

Examples of some common Public Key Encryption algorithms are:

- Rivest-Shamir-Adelman
- Diffie-Hellman

1.3 HACKER ATTACKS AND CRYPTANALYSIS

Cryptanalysis is the science and study of breaking ciphers. A cipher can be broken if it is possible to determine the plaintext or key from the ciphertext, or if someone can determine the key from plaintext-ciphertext pairs. The attempt to find the key in encryption algorithms by brute force requires a very long time. Longer the key length, more immune is the key against brute force attacks. A cryptanalysis attack is an attack by an intruder trying to discover the contents of an encrypted message or the secret key by means other than a straightforward random attack. The three methods of attack are:

- Ciphertext-only
- Known-plaintext
- Chosen-plaintext

Triple DES with 168-bit key length has become more popular. The Triple DES algorithm provides high security and has been proven to be immune to hacker attacks. However, it seems to have the following drawbacks:

- The algorithm is sluggish.
- It uses 64-bit block size. For efficiency and security, a larger block size is desirable.

Comparatively, AES offers a higher level of security, more efficiency and less computational complexity. Therefore it is fast replacing its encryption predecessors.

1.4 GALOIS FIELD - AN OVERVIEW

Galois field is a set that contains a finite number of elements and is ideally suited for Hardware implementation. A finite field has the property that the arithmetic operations on field elements always have a result belonging to the field.

1.4.1 Properties and Importance of Galois Field

A number field has the following properties:

- Both addition and multiplication operation satisfy the commutative, associative and distributive laws.
- Closure: Adding or multiplying elements always yields finite elements as results.

The identity element for addition is zero and for multiplication it is one.

1.4.2 Importance of Galois Field

- It involves operations on a finite field of numbers and there is no concept of irrational numbers or infinity.
- In hardware implementation, it reduces the logical elements needed to implement the addition operations using simple XOR and AND gates.

1.4.3 The Galois Field Elements

A Galois Field consists of a set of elements (numbers). Galois Fields are setup by initially defining the size of the field, which means how many elements will exist within the field and also the values those elements will possess. The values that the elements will possess are defined by a polynomial known as primitive polynomial of the Galois field.

The elements are based on a primitive element, denoted as α and take the values:

$$0, \alpha^0, \alpha^1, \alpha^2, \ldots\ldots\ldots\ldots \alpha^{n-1}$$

To form a set of elements where $n=2^m-1$, this field is then known as GF (2^m). The value of α is usually chosen to be 2. Having chosen α, the higher powers can then be obtained by multiplying by α at each step. The operations upon elements are accomplished via bit wise operations such as XOR, AND, OR logic.

1.4.4 Extended Galois Field (2^m)

Non-binary fields (2^m) are extensions of the binary field GF (2)={0,1} and have more than two elements. The additional elements in the extended field cannot be 0 or 1, since all of the elements must be unique, so a new symbol α is used to represent the other elements in the field. Each non-zero element can be represented by a power of α.

An element of GF (2^m) is defined as the root of a primitive polynomial p(x) of degree m. In general Extended Galois fields of class GF (2^m) possess 2^m elements, where m is the symbol size of the element in bits. The GF chosen is GF (256) where m=8. In GF (2^m) all elements besides the zero element can be represented in two alternative ways:

- Binary form.
- Exponential form as α^p.

Computationally the Galois Fields are quite attractive as all operations do not produce carry, involve no rounding issues and word length is always constant. Due to these appealing features, Galois Field finds a variety of

applications ranging from error control codes, cryptography and switching theory to DSP.

Having briefed the need and significance of cryptography, this report unfurls the implementation of AES algorithm in the following chapters.

CHAPTER 2

AES-AN OVERVIEW

The AES was published by National Institute of Standards and Technology in 2001. AES is a symmetric block cipher that is intended to replace DES as the approved standard for a wide range of applications.

2.1 ORIGIN OF AES

In 1999, NIST issued a new version of its DES standard that indicated that DES should only be used for legacy systems and that Triple DES be used. Because of its drawbacks, 3DES was not found to be a reasonable candidate for long-term use. This led to NIST's call for proposals for a new AES.

The new algorithm is expected to have security strength equal to or better than Triple DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with block length of 128 bits and support for key lengths of 128, 192 and 256 bits.

In a first round of evaluation, 15 proposed algorithms were accepted. A second round narrowed the field to 5 algorithms. NIST selected Rijndael as the proposed AES algorithm in November 2001. The two researchers who developed and submitted Rijndael were Dr. Joan Daemen and Dr.Vincent Rijmen from Belgium.

2.2 AES EVALUATION CRITERIA

NIST used three categories of criteria to evaluate potential candidates for the new algorithm.

2.2.1 Security

This refers to the effort required to crypt analyze an algorithm and the emphasis in the evaluation was on the practicality of the attack.

- **Actual security:** Compared to other submitted algorithms.
- **Randomness:** The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
- **Soundness:** Of the mathematical basis for the algorithm's security.

2.2.2 Cost

NIST intends AES to be practical in a wide range of applications. Accordingly, AES must have high computational efficiency, so as to be usable in high-speed applications, such as broadband links.

- **Licensing requirements:** NIST intends that when the AES is issued, the algorithm specified in the AES shall be available on a worldwide, non-exclusive, royalty-free basis.
- **Computational efficiency:** The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination; more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis. Computational efficiency essentially refers to the speed of

the algorithm. Public comments on each algorithm's efficiency will also be taken into consideration by NIST.

- **Memory requirements:** The memory required for both hardware and software implementations of the algorithm will also be considered during the evaluation process. Round 1 analysis by NIST will focus primarily on software implementations; more attention will be paid to hardware implementations during Round 2. Memory requirements will include such factors as gate counts for hardware implementations and code size and RAM requirements for software implementations.

2.2.3 Algorithm and Implementation Characteristics

This category includes a variety of considerations, including flexibility; suitability for a variety of hardware and software implementations and simplicity, which will make an analysis of security more straightforward.

- **Flexibility:** Candidate algorithms with greater flexibility will meet the needs of more users than less flexible ones. However, some extremes of functionality are of little practical application (e.g., extremely short key lengths); for those cases, preference will not be given. Some examples of flexibility may include the following:
    i. The algorithm can accommodate additional key and block sizes (e.g., 64-bit block sizes, key sizes other than those specified in the Minimum Acceptability Requirements section).
    ii. The algorithm can be implemented securely and efficiently in a wide variety of platform and applications (e.g., 8-bit processors, ATM networks, voice and satellite communications etc).

iii. The algorithm can be implemented as a stream cipher, message authentication code generator, pseudorandom number generator, hashing algorithm etc.

- **Hardware and software suitability:** A candidate algorithm shall not be restrictive in the sense that it can only be implemented in hardware. If one can also implement the algorithm efficiently in firmware, then this will be an advantage in the area of flexibility.
- **Simplicity:** A candidate algorithm shall be judged according to relative simplicity of design.

2.3 NIST EVALUATION OF RIJNDAEL

- **General Security:** Rijndael has no known security attacks and uses S-boxes as nonlinear components. It appears to have an adequate security margin, but has received some criticism suggesting that its mathematical structure may lead to attacks. On the other hand, the simple structure may have facilitated its security analysis during the timeframe of the AES development process.
- **Software Implementation:** Rijndael performs encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms and DSPs. However, there is a decrease in performance with the higher key sizes because of the increased number of rounds that are performed.
- **Restricted-Space Environments:** In general, Rijndael is very well suited for restricted-space environments where either encryption or decryption is implemented (but not both). It has very low RAM and ROM requirements. A drawback is that ROM requirements will

increase if both encryption and decryption are implemented simultaneously, although it appears to remain suitable for these environments. The key schedule for decryption is separate from encryption.

- **Attacks on Implementations:** The operations used by Rijndael are among the easiest to defend against power and timing attacks. The use of masking techniques to provide Rijndael with some defense against these attacks does not cause significant performance degradation relative to the other finalists and its RAM requirements remains reasonable. Rijndael appears to gain a major speed advantage over its competitors when such protections are considered.
- **Encryption Vs Decryption:** The encryption and decryption functions in Rijndael differ. One FPGA study reports that the implementation of both encryption and decryption takes about 60% more space than the implementation of encryption alone. Rijndael's speed does not vary significantly between encryption and decryption, although the key setup performance is slower for decryption than encryption.
- **Other Versatility and Flexibility:** Rijndael fully supports block sizes and key sizes of 128 bits, 192 bits and 256 bits, in any combination. In principle, the Rijndael structure can accommodate any block sizes and key sizes that are multiples of 32 as well as changes in number of passes that are specified.

2.4 AES PARAMETERS

A number of AES parameters that depend on the key length are described in Table 2.1.

Table 2.1 AES Parameters

Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Thus Rijndael was designed to have the following characteristics:

- Resistance against all known attacks
- Speed and code compactness on a wide range of platforms
- Design simplicity

The next chapter explains the steps involved in AES algorithm.

CHAPTER 3

AES ALGORITHM-ENCRYPTION AND DECRYPTION

AES is an iterated symmetric block cipher, which means that:

- AES works by repeating the same defined steps multiple times.
- AES is a secret key encryption algorithm.
- AES operates on a fixed number of bytes.

AES as well as most encryption algorithms are reversible. This means that almost the same steps are performed to complete both encryption and decryption in reverse order. The algorithm operates on bytes, which makes it simpler to implement. The symmetric key that is used is expanded into individual sub keys, one sub key for each pass. This process is called Key Expansion, which has been described later in the text.

Block diagram of the process flow is shown in Fig 3.1. As shown, the input data of 128 bits is encrypted using an expanded key (1408 bits) and is stored in a file. This is then decrypted to recover the original 128 bits.

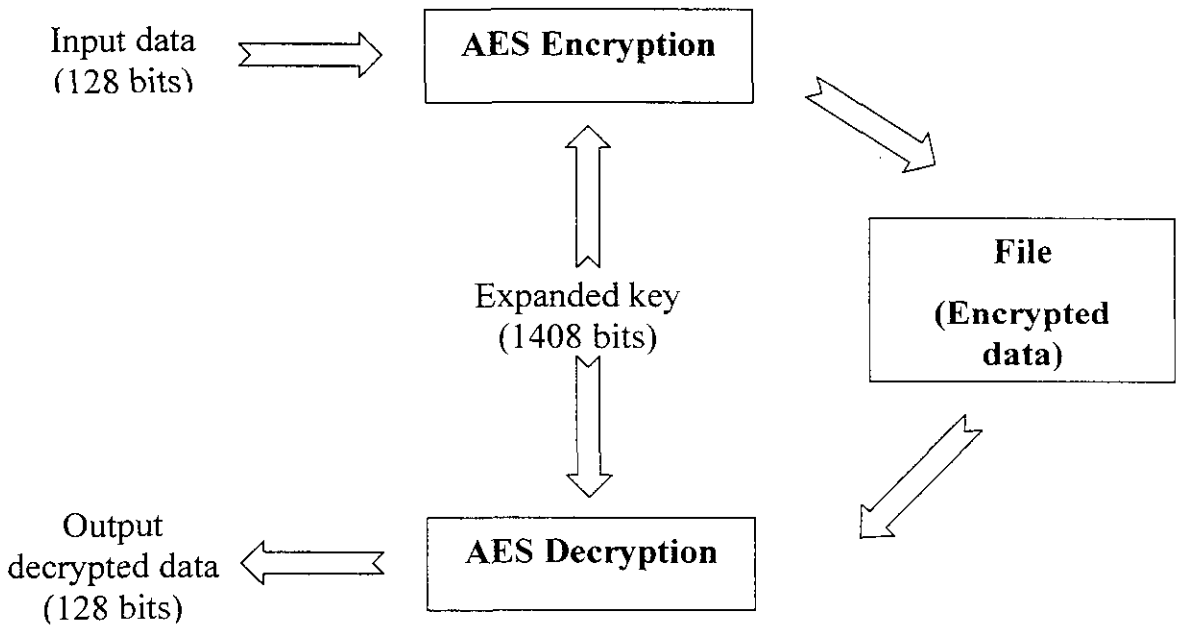


Fig 3.1 Process flow

The operations of AES can easily be broken down into the following functions:

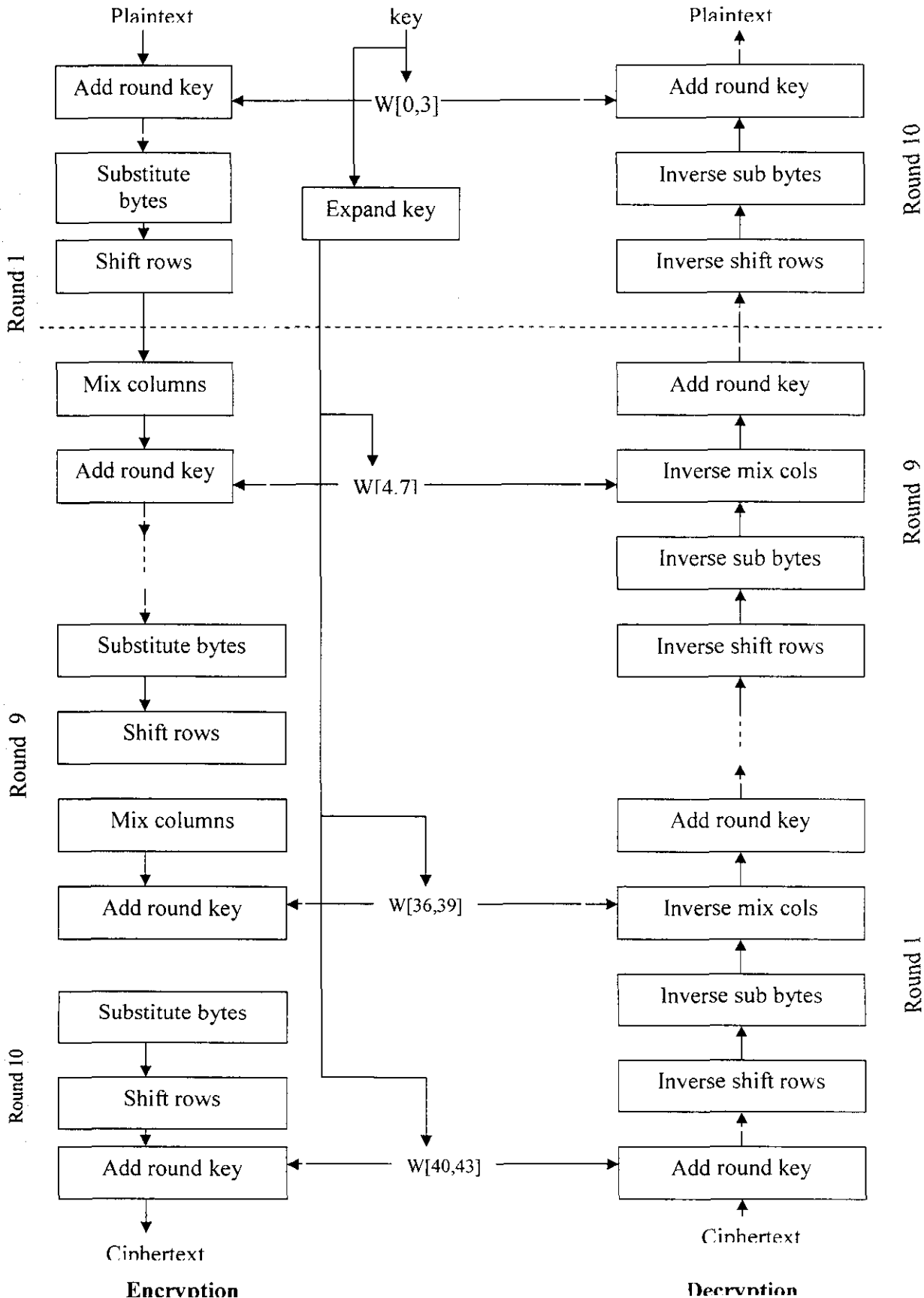
- Add Round Key
- Byte Sub
- Shift Row
- Mix Column

An iteration of the above steps is called a round. For a block size of 16-bytes and key size of 16 bytes the amount of round is 10. The number of passes of the algorithm depends on the key size as depicted in Table 3.1.

Table 3.1 Passes Involved

Key size (bytes)	Block size (bytes)	Rounds
16	16	10
24	16	12
32	16	14

The steps involved and the way in which the expanded key has been used in each pass of the encryption and decryption processes have been illustrated in Fig 3.2.


Plaintext
key
Add round key
W[0,3]
Substitute bytes
Expand key
Shift rows
Round 1
Mix columns
Add round key
W[4,7]
Substitute bytes
Shift rows
Round 9
Mix columns
Add round key
W[36,39]
Substitute bytes
Shift rows
Round 10
Add round key
W[40,43]
Ciphertext
Encryption
Plaintext
Add round key
Inverse sub bytes
Inverse shift rows
Round 10
Add round key
Inverse mix cols
Inverse sub bytes
Inverse shift rows
Round 9
Add round key
Inverse mix cols
Inverse sub bytes
Inverse shift rows
Round 1
Add round key
Ciphertext
Decryption

3.1 AES ENCRYPTION

AES encryption cipher using a 16-byte key is illustrated in Table 3.2. The term state below refers to the present condition of the block under consideration.

Table 3.2 Illustration of Encryption passes

Round	Function
-	Add Round Key (State)
0	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
1	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
2	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
3	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
4	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
5	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
6	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
7	Add Round Key (Mix Column (Shift Row (Byte Sub (State))))
8	Add Round Key (Mix Column(Shift Row (Byte Sub (State))))
9	Add Round Key (Shift Row (Byte Sub(State)))

3.1.1 AES Cipher Functions

The steps of the encryption process are elaborated below.

3.1.1.1 Add Round Key

Each of the 16 bytes of the state is XORed against each of the 16 bytes of a portion of the expanded key for the current round. The Expanded Key bytes are never reused. So once the first 16 bytes are XORed against the first 16 bytes of the expanded key then the expanded key bytes 1-16 are never used again. The next time the Add Round Key function is called bytes 17-32 are XORed against the state. The first time Add Round Key gets executed as shown in Fig 3.3.

e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
y	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Fig 3.3 First Pass of Add round key

The second time Add Round Key is executed as shown in Fig 3.4 and so on for each round of execution.

e	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR	XOR
	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Fig 3.4 Second Pass of Add round key

3.1.1.2 Byte Sub

In this step, a pre-defined S-box value is used to replace the state resulting from the Add Round operation. For example, HEX 19 would get replaced with HEX D4. AES S-Box Lookup table is shown below:

Table 3.3 S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

3.1.1.3 Shift Row

In this operation, circular left shift is performed on each row of the state matrix. This is not a bit wise shift. The circular shift just moves each byte one space over. A byte that was in the second position may end up in the first position after the shift. The circular part of it specifies that the byte in the first position shifted one space will end up in the last position of the same row. Consider that the bytes in the State matrix range from 1 through 16.So the matrix will be

$$\begin{pmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{pmatrix}$$

Each row is then moved over (shifted) 1, 2 or 3 spaces over to the left, depending on the row of the state. First row is never shifted.

The following table shows how the individual bytes are first arranged in the table and then moved over (shifted).

$$\begin{pmatrix} 01 & 05 & 09 & 13 \\ 02 & 06 & 10 & 14 \\ 03 & 07 & 11 & 15 \\ 04 & 08 & 12 & 16 \end{pmatrix} \Rightarrow \begin{pmatrix} 01 & 05 & 09 & 13 \\ 06 & 10 & 14 & 02 \\ 11 & 15 & 03 & 07 \\ 16 & 04 & 08 & 12 \end{pmatrix}$$

3.1.1.4 Mix Column

This step comprises of two parts. The first explains which parts of the state are multiplied against which parts of the matrix. The second will explain how this multiplication is implemented over what's called a Galois Field.

- **Matrix Multiplication**

  The state is arranged into a 4x4 matrix and XOR operation is performed against the predefined multiplication matrix (similar to matrix multiplication).

16 byte state

$$\begin{pmatrix} b1 & b5 & b9 & b13 \\ b2 & b6 & b10 & b14 \\ b3 & b7 & b11 & b15 \\ b4 & b8 & b12 & b16 \end{pmatrix}$$

multiplication matrix

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

The first resultant byte is calculated by multiplying 4 values of the state column against the corresponding 4 values of the first row of the matrix. The result of each multiplication is then XORed to produce 1 byte.

b1 = (b1 * 2) XOR (b2*3) XOR (b3*1) XOR (b4*1)

This procedure is repeated until there are no more elements.

- **Galois Field Multiplication**

  The resulting 16-bit product of each multiplication undergoes modulo operation with the binary equivalent of a Galois polynomial of order 8 ($x^8+ x^4+ x^3+ x^2+1$).

3.2 AES DECRYPTION

AES encryption cipher using a 16-byte key is illustrated in Table 3.4.

Table 3.4 Illustration of Decryption passes

Round	Function
-	Add Round Key(State)
0	Add Round Key (Byte Sub(Shift Row(State)))
1	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
2	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
3	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
4	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
5	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
6	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
7	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
8	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))
9	Add Round Key (Byte Sub(Shift Row(Inverse Mix Column(State))))

3.2.1 AES Decipher Functions

The steps of the decryption process are elaborated below.

3.2.1.1 Add Round Key

During decryption the encryption procedure is reversed. Therefore the state is first XORed against the last 16 bytes of the expanded key and then the second last 16 bytes and so on.

3.2.2.2 Inverse Mix Column

The same procedure as discussed in encryption is followed here with an exception that the multiplication matrix is changed to

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

3.2.2.3 Shift Row

This step is just the reverse of encryption Shift row except that, circular right shift replaces circular left shift.

$$\begin{bmatrix} 01 & 05 & 09 & 13 \\ 02 & 06 & 10 & 14 \\ 03 & 07 & 11 & 15 \\ 04 & 08 & 12 & 16 \end{bmatrix} \Rightarrow \begin{bmatrix} 01 & 05 & 09 & 13 \\ 14 & 02 & 06 & 10 \\ 11 & 15 & 03 & 07 \\ 08 & 12 & 16 & 04 \end{bmatrix}$$

3.2.2.4 *Byte Sub*

During decryption each value in the state is replaced with the corresponding inverse SBOX element. For example HEX D4 would get replaced with HEX 19. Inverse of SBOX Lookup Table is shown in Table 3.5.

Table 3.5 Inverse S-Box Lookup Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

The operations discussed above are performed for 10 passes, where an encrypted data (16 bytes) is obtained as an output for encryption process. It is provided as the input to the decryption process and the original input message is obtained.

CHAPTER 4

KEY EXPANSION

4.1 AN OVERVIEW

Key Expansion is the process of expanding the input 128-bit key into a key of size 1408 bits. The expanded key is used in the Add Round function that was described earlier. Each time the Add Round function is called, a different part of the expanded key is XORed against the state. In order for this to work the Expanded Key must be large enough so that it can provide key material every time the Add Round Key function is executed. The Add Round function gets called for each pass as well as one extra time at the beginning of the algorithm. Therefore the size of the expanded key will always be equal to:

16 * (number of passes + 1)

16 in the above function is actually the size of the block in bytes. The expanded key size depends on the input key size as shown in Table 4.1.

Table 4.1 Expanded Key Size

Key Size (bytes)	Block Size (bytes)	Expanded Key (bytes)
16	16	176
24	16	208
32	16	240

The key expansion routine executes a maximum of four consecutive operations as follows:

- Rotate word
- Sub word
- Rcon
- XOR

The key expansion order is depicted in Fig 4.1. The first bytes of the expanded key are always equal to the key. If the key is 16 bytes long the first 16 bytes of the expanded key will be the same as the original key. Each pass adds 4 bytes to the expanded key. With the exception of the first pass every other pass takes the previous pass's 4 bytes as input, operates on it and returns 4 bytes.

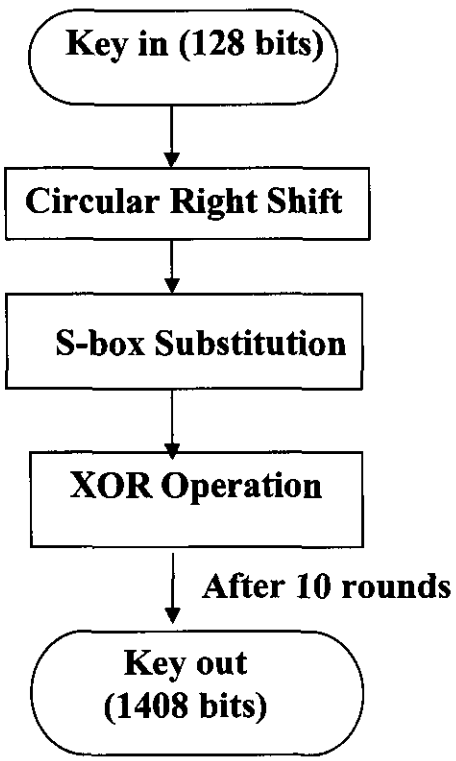


Fig 4.1 Key expansion process

4.2 AES KEY EXPANSION FUNCTIONS

The various key expansion steps have been elaborated in their order below.

4.2.1 Rotate Word

This does a circular left shift on 4 bytes similar to the Shift Row Function.

4.2.2 Sub Word

This step applies the S-box value substitution as described in bytes sub function to each of the 4 bytes in the argument.

4.2.3 Rcon

This function returns a 4-byte value based on the values shown below. Each pass gets the Rcon value named after it.

For example, Rcon(0) = 01000000 return (each Rcon bit expanded into it's 4-bit binary equivalent) "0000 0001 0000 0000 0000 0000 0000 0000".

```
Rcon(0)  = 01000000
Rcon(1)  = 02000000
Rcon(2)  = 04000000
Rcon(3)  = 08000000
Rcon(4)  = 10000000
Rcon(5)  = 20000000
Rcon(6)  = 40000000
Rcon(7)  = 80000000
Rcon(8)  = 1B000000
Rcon(9)  = 36000000
Rcon(10) = 6C000000
Rcon(11) = D8000000
Rcon(12) = AB000000
Rcon(13) = 4D000000
Rcon(14) = 9A000000
```

4.2.4 XOR

The byte that is obtained in the sub word step is XORed with the Rcon value returned in the previous step. The resultant is then XORed with the next 32-bits of the input key (which follow the ones used in the previous pass).

This key expansion process provides a unique set of key bytes for each pass of encryption/decryption and hence acts as an effective firewall against cryptanalysis.

CHAPTER 5

RESULTS AND DISCUSSION

The simulation results for Key Expansion module, Encryption and Decryption module are presented in this chapter. Code for the various modules has been written in VHDL and simulated using the MODELSIM 6.2c simulator. The synthesis part has been carried out using Xilinx 8.2i and implementation has been done at software level on XC2VP70 device of VIRTEX 2P family.

5.1 SIMULATION RESULT FOR KEY EXPANSION

The simulation result for key expansion module is shown in Fig 5.1. Here, the given 16-byte key is expanded into 176-byte key. The values 1407,1406,.....0 represent the expanded key bits.

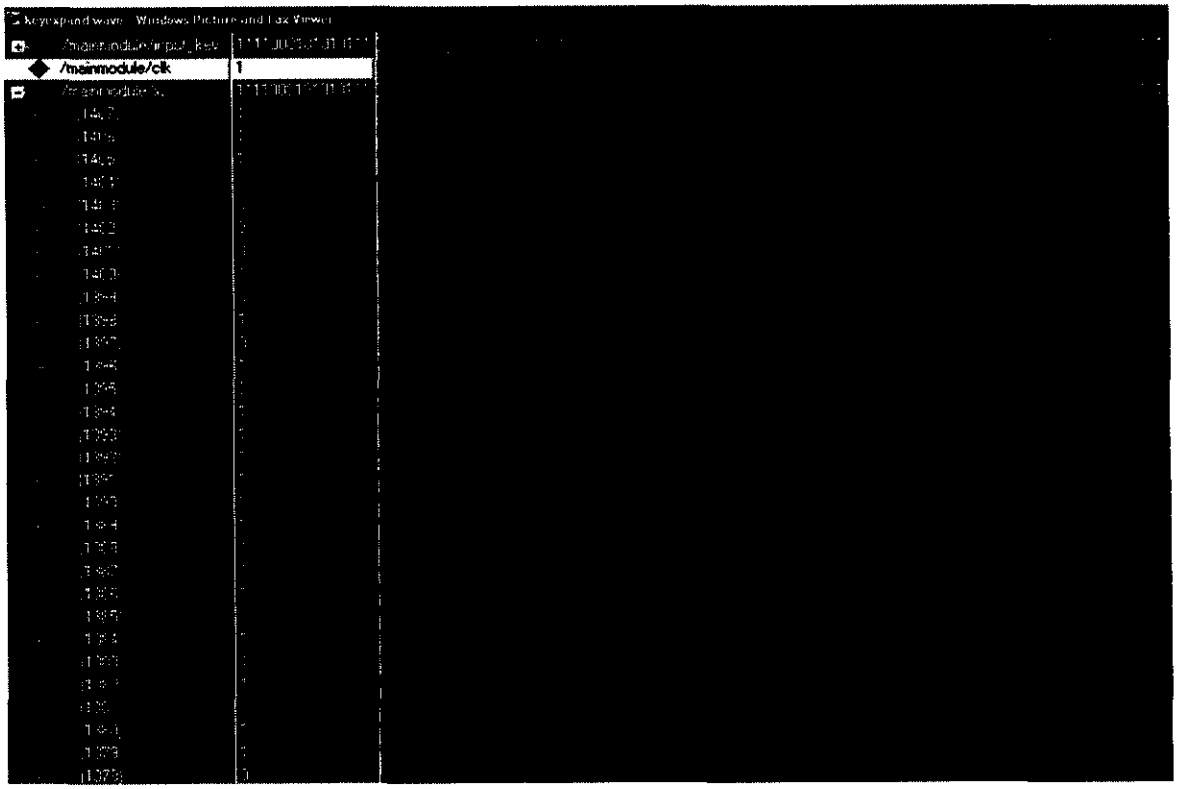


Fig 5.1 Simulation performance of Key expansion

5.2 SIMULATION RESULT FOR ENCRYPTION

The simulation result for encryption is shown in Fig 5.2. The given 16-byte input data is encrypted into a 16-byte cipher. The encryption module is designed in such a way that it works when the control input is 1.

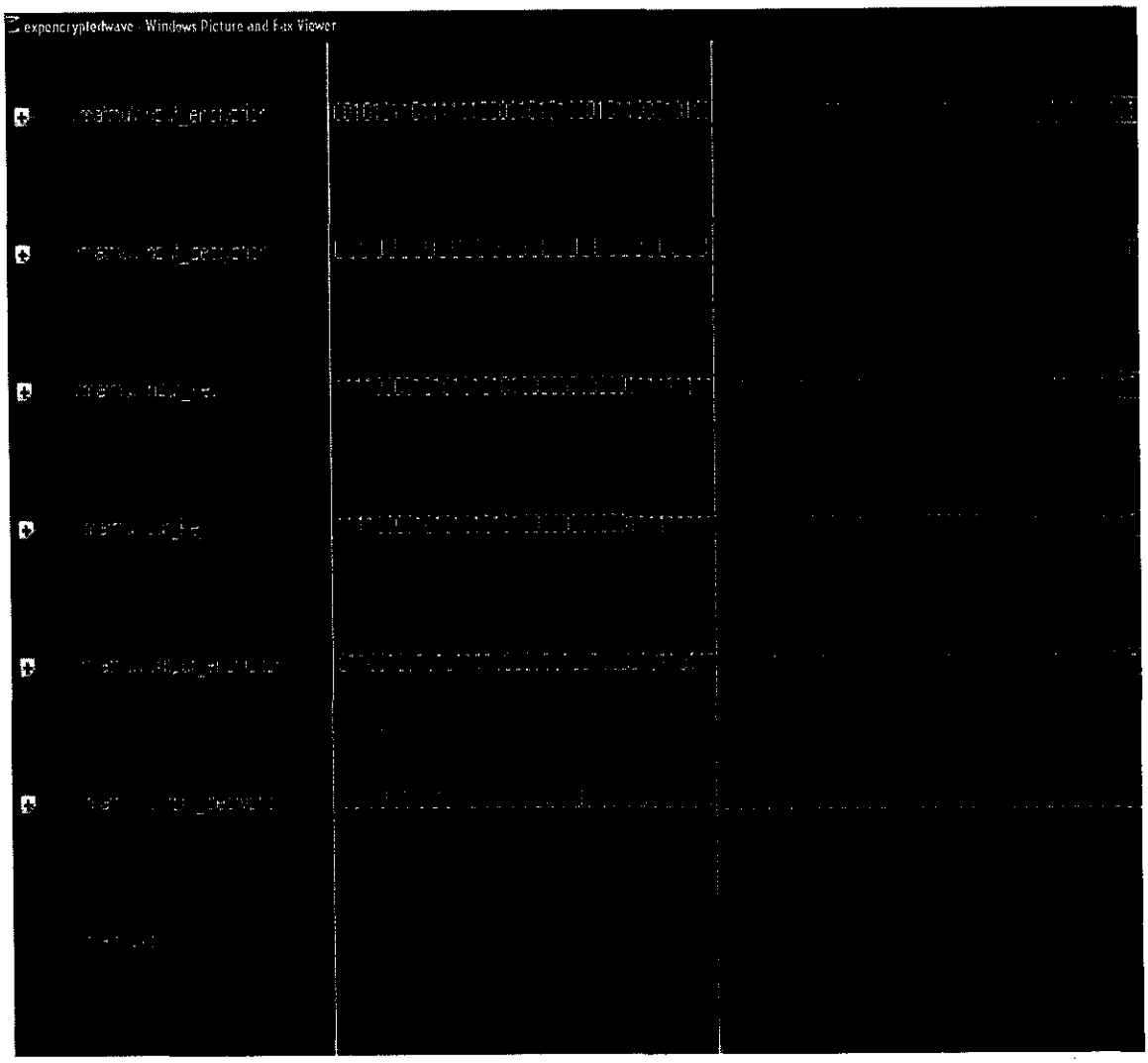


Fig 5.2 Simulation result for Encryption

5.3 SIMULATION RESULT FOR DECRYPTION

The simulation result for decryption is shown in Fig 5.3. The given 16-byte cipher is deciphered into the original 16-byte data. Decryption module is designed to work when the control input value is 0.

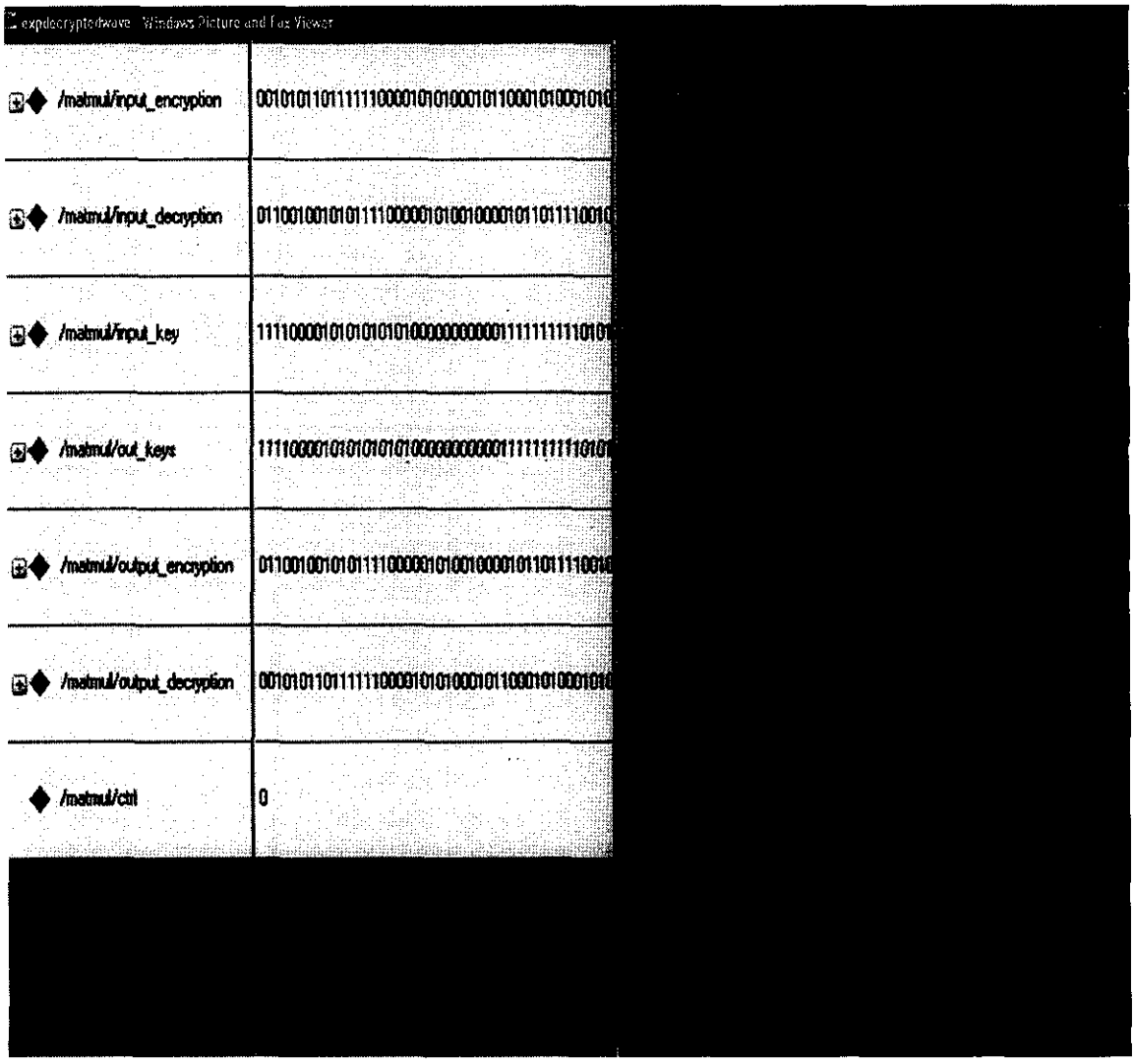


Fig 5.3 Simulation result for Decryption

5.4 DESIGN SUMMARY OF ENCRYPTION

The design summary of encryption is shown in Fig 5.4. This indicates the number of IOB latches, input LUTs and the percentage the device resources are used.

ENCRYPTION Project Status			
Project File:	encryption.ise	Current State:	Placed and Routed
Module Name:	matmul	• Errors:	No Errors
Target Device:	xc2vp70-6ff1704	• Warnings:	3 Warnings (0 new, 0 filtered)
Product Version:	ISE 8.2i	• Updated:	Sat Mar 15 10:34:40 2008

ENCRYPTION Partition Summary
No partition information was found.

Current Errors
No Errors Found

Current Warnings
Synthesis Warnings
WARNING:Xst:737: - Found 128-bit latch for signal <$rmux0249 12>.
Map Warnings
WARNING:LIT:243: - Logical network N48921 has no load.
WARNING:LIT:395: - The above warning message base_net_load_rule is repeated 1 more times for the following (max. 5 shown): N48931 To see the details of these warning messages, please use the detail switch.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	30,059	66,176	45%	
Logic Distribution				
Number of occupied Slices	15,127	33,088	45%	
Number of Slices containing only related logic	15,127	15,127	100%	
Number of Slices containing unrelated logic	0	15,127	0%	
Total Number 4 input LUTs	30,059	66,176	45%	
Number of bonded IOBs	385	996	38%	
IOB Latches	128			
Number of PPC405s	0	2	0%	
Number of GCLKs	1	16	6%	
Number of GTs	0	20	0%	
Number of GT10s	0	0	0%	

Total equivalent gate count for design	256,351
Additional JTAG gate count for IOBs	18,480

Performance Summary			
Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Failing Constraints
All Constraints Were Met

Clock Report					
Clock Net	Resource	Locked	Fanout	Net Skew(ns)	Max Delay(ns)
CTRL_BUFGP	BUFGMUX7S	No	128	0.384	

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sat Mar 15 09:48:01 2008	0	1 Warning (0 new, 0 filtered)	0
Translation Report	Current	Sat Mar 15 09:48:59 2008	0	0	0
Map Report	Current	Sat Mar 15 09:51:58 2008	0	2 Warnings (0 new, 0 filtered)	3 Infos (0 new, 0 filtered)
Place and Route Report	Current	Sat Mar 15 10:30:43 2008	0	0	1 Info (0 new, 0 filtered)
Static Timing Report	Current	Sat Mar 15 10:34:18 2008	0	0	2 Infos (0 new, 0 filtered)
Bitgen Report					

Secondary Reports		
Report Name	Status	Generated
Xplorer Report		

Fig 5.4 Design Summary of Encryption

5.5 DESIGN SUMMARY OF DECRYPTION

Similar to encryption, this summary provides information on the percentage of use of device resources, number of IOB latches, input LUTs and so on.

DECRYPTIONCODE Project Status			
Project File:	decryptioncode.ise	Current State:	Placed and Routed
Module Name:	matmul	• Errors:	No Errors
Target Device:	xc2vp70-6ff1704	• Warnings:	2 Warnings (2 new, 0 filtered)
Product Version:	ISE 8.2i	• Updated:	Sat Mar 15 13:44:03 2008

DECRYPTIONCODE Partition Summary
No partition information was found.

Current Errors
No Errors Found

Current Warnings
Map Warnings
WARNING:LIT:243: - Logical network N99651 has no load.
WARNING:LIT:395: - The above warning message base_net_load_rule is repeated 1 more times for the following (max. 5 shown): N99661 To see the details of these warning messages, please use the -detail switch.

Device Utilization Summary				
Logic Utilization	Used	Available	Utilization	Note(s
Number of 4 input LUTs	33,789	66,176	51%	
Logic Distribution				
Number of occupied Slices	17,017	33,088	51%	
Number of Slices containing only related logic	17,017	17,017	100%	
Number of Slices containing unrelated logic	0	17,017	0%	
Total Number 4 input LUTs	33,791	66,176	51%	
Number used as logic	33,789			
Number used as a route-thru	2			
Number of bonded IOBs	386	996	38%	
IOB Flip Flops	128			
Number of PPC405s	0	2	0%	
Number of GCLKs	1	16	6%	
Number of GTs	0	20	0%	
Number of GT10s	0	0	0%	

Total equivalent gate count for design	276,202
Additional JTAG gate count for IOBs	18,528

Performance Summary			
Final Timing Score:	0	Pinout Data:	Pinout Report
Routing Results:	All Signals Completely Routed	Clock Data:	Clock Report
Timing Constraints:	All Constraints Met		

Failing Constraints
All Constraints Were Met

Clock Report					
Clock Net	Resource	Locked	Fanout	Net Skew(ns)	Max Delay(ns)
clk_BUFGP	BUFGMUX7S	No	128	0.494	

Detailed Reports					
Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sat Mar 15 12:11:10 2008	0	0	0
Translation Report	Current	Sat Mar 15 12:13:44 2008	0	0	0
Map Report	Current	Sat Mar 15 12:17:58 2008	0	2 Warnings (2 new, 0 filtered)	3 Infos (3 new, 0 filtered)
Place and Route Report	Current	Sat Mar 15 13:38:53 2008	0	0	1 Info (1 new, 0 filtered)
Static Timing Report	Current	Sat Mar 15 13:43:39 2008	0	0	2 Infos (2 new, 0 filtered)
Bitgen Report					

Secondary Reports		
Report Name	Status	Generated
Xplorer Report		

Fig 5.5 Design Summary of Decryption

5.6 SYNTHESIS REPORT OF ENCRYPTION

1) Synthesis Options Summary
2) HDL Compilation
3) Design Hierarchy Analysis
4) HDL Analysis
5) HDL Synthesis
   5.1) HDL Synthesis Report
6) Advanced HDL Synthesis
   6.1) Advanced HDL Synthesis Report
7) Low Level Synthesis
8) Partition Report
9) Final Report
   9.1) Device utilization summary
   9.2) Timing report

Synthesis Options Summary

Source Parameters

Input File Name	: "matmul.prj"
Input Format	: mixed
Ignore Synthesis Constraint File	: NO

Target Parameters

Output File Name	: "matmul"
Output Format	: NGC
Target Device	: xc2vp70-6-ff1704

Source Options

Top Module Name	: matmul
Automatic FSM Extraction	: YES
FSM Encoding Algorithm	: Auto
FSM Style	: lut
RAM Extraction	: Yes
RAM Style	: Auto
ROM Extraction	: Yes
Mux Style	: Auto
Decoder Extraction	: YES

Priority Encoder Extraction	: YES
Shift Register Extraction	: YES
Logical Shifter Extraction	: YES
XOR Collapsing	: YES
ROM Style	: Auto
Mux Extraction	: YES
Resource Sharing	: YES
Multiplier Style	: auto
Automatic Register Balancing	: No

Target Options

Add IO Buffers	: YES
Global Maximum Fanout	: 500
Add Generic Clock Buffer(BUFG)	: 16
Register Duplication	: YES
Slice Packing	: YES
Pack IO Registers into IOBs	: auto
Equivalent register Removal	: YES

General Options

Optimization Goal	: Speed
Optimization Effort	: 1
Keep Hierarchy	: NO
RTL Output	: Yes
Global Optimization	: AllClockNets
Write Timing Constraints	: NO
Hierarchy Separator	: /
Bus Delimiter	: <>
Case Specifier	: maintain
Slice Utilization Ratio	: 100
Slice Utilization Ratio Delta	: 5

Other Options

lso	: matmul.lso
Read Cores	: YES
cross_clock_analysis	: NO
verilog2001	: YES
safe_implementation	: No
Optimize Instantiated Primitives	: NO

tristate2logic	: Yes
use_clock_enable	: Yes
use_sync_set	: Yes
use_sync_reset	: Yes

HDL Synthesis Report

Macro Statistics	
# ROMs	: 200
256x8-bit ROM	: 200
# Latches	: 1
128-bit latch	: 1
# Xors	: 2501
1-bit xor2	: 1280
1-bit xor3	: 32
1-bit xor4	: 1152
128-bit xor2	: 11
32-bit xor2	: 26

Advanced HDL Synthesis Report

Macro Statistics	
# ROMs	: 200
256x8-bit ROM	: 200
# Latches	: 1
128-bit latch	: 1
# Xors	: 2501
1-bit xor2	: 1280
1-bit xor3	: 32
1-bit xor4	: 1152
128-bit xor2	: 11
32-bit xor2	: 26

Final Report

Final Results

RTL Top Level Output File Name	: matmul.ngr
Top Level Output File Name	: matmul
Output Format	: NGC
Optimization Goal	: Speed

Keep Hierarchy : NO

Design Statistics

IOs : 385

Cell Usage

# BELS	: 55177
# LUT2	: 383
# LUT3	: 370
# LUT4	: 29306
# MUXF5	: 13918
# MUXF6	: 6400
# MUXF7	: 3200
# MUXF8	: 1600
# FlipFlops/Latches	: 128
# LD	: 128
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 384
# IBUF	: 256
# OBUF	: 128

Device utilization summary

Selected Device : 2vp70ff1704-6

Number of Slices	: 15196 out of 33088	45%
Number of 4 input LUTs	: 30059 out of 66176	45%
Number of Ios	: 385	
Number of bonded IOBs	: 385 out of 996	38%
IOB Flip Flops	: 128	
Number of GCLKs	: 1 out of 16	6%

Total memory usage is 574880 kilobytes

5.7 SYNTHESIS REPORT OF DECRYPTION

1) Synthesis Options Summary
2) HDL Compilation
3) Design Hierarchy Analysis
4) HDL Analysis
5) HDL Synthesis
   5.1) HDL Synthesis Report
6) Advanced HDL Synthesis
   6.1) Advanced HDL Synthesis Report
7) Low Level Synthesis
8) Partition Report
9) Final Report
   9.1) Device utilization summary
   9.2) Timing report

Synthesis options summary

Source Parameters

Input File Name	: "matmul.prj"
Input Format	: mixed
Ignore Synthesis Constraint File	: NO

Target Parameters

Output File Name	: "matmul"
Output Format	: NGC
Target Device	: xc2vp70-6-ff1704

Source Options

Top Module Name	: matmul
Automatic FSM Extraction	: YES
FSM Encoding Algorithm	: Auto
FSM Style	: lut
RAM Extraction	: Yes
RAM Style	: Auto
ROM Extraction	: Yes
Mux Style	: Auto
Decoder Extraction	: YES

Priority Encoder Extraction	: YES
Shift Register Extraction	: YES
Logical Shifter Extraction	: YES
XOR Collapsing	: YES
ROM Style	: Auto
Mux Extraction	: YES
Resource Sharing	: YES
Multiplier Style	: auto
Automatic Register Balancing	: No

Target Options

Add IO Buffers	: YES
Global Maximum Fanout	: 500
Add Generic Clock Buffer(BUFG)	: 16
Register Duplication	: YES
Slice Packing	: YES
Pack IO Registers into IOBs	: auto
Equivalent register Removal	: YES

General Options

Optimization Goal	: Area
Optimization Effort	: 2
Keep Hierarchy	: NO
RTL Output	: Yes
Global Optimization	: AllClockNets
Write Timing Constraints	: YES
Hierarchy Separator	: /
Bus Delimiter	: <>
Case Specifier	: maintain
Slice Utilization Ratio	: 100
Slice Utilization Ratio Delta	: 5

Other Options

lso	: matmul.lso
Read Cores	: YES
cross_clock_analysis	: NO
verilog2001	: YES
safe_implementation	: No
Optimize Instantiated Primitives	: NO

tristate2logic	: Yes
use_clock_enable	: Yes
use_sync_set	: Yes
use_sync_reset	: Yes

HDL Synthesis Report

Macro Statistics	
# ROMs	: 200
256x8-bit ROM	: 200
# Registers	: 1
128-bit register	: 1
# Xors	: 2501
1-bit xor2	: 1280
1-bit xor3	: 32
1-bit xor4	: 1152
128-bit xor2	: 11
32-bit xor2	: 26

Advanced HDL Synthesis Report

Macro Statistics	
# ROMs	: 200
256x8-bit ROM	: 200
# Registers	: 128
Flip-Flops	: 128
# Xors	: 2501
1-bit xor2	: 1280
1-bit xor3	: 32
1-bit xor4	: 1152
128-bit xor2	: 11
32-bit xor2	: 26

Final Report

Final Results

RTL Top Level Output File Name	: matmul.ngr
Top Level Output File Name	: matmul
Output Format	: NGC
Optimization Goal	: Area

Keep Hierarchy : NO

Design Statistics

IOs : 386

Cell Usage

# BELS	: 57937
# INV	: 1
# LUT2	: 931
# LUT3	: 978
# LUT4	: 31880
# MUXF5	: 12947
# MUXF6	: 6400
# MUXF7	: 3200
# MUXF8	: 1600
# FlipFlops/Latches	: 128
# FDE	: 128
# Clock Buffers	: 1
# BUFGP	: 1
# IO Buffers	: 385
# IBUF	: 257
# OBUF	: 128

Device utilization summary

Selected Device : 2vp70ff1704-6

Number of Slices	: 17495 out of 33088	52%
Number of 4 input LUTs	: 33790 out of 66176	51%
Number of Ios	: 386	
Number of bonded IOBs	: 386 out of 996	38%
IOB Flip Flops	: 128	
Number of GCLKs	: 1 out of 16	6%

Total memory usage is 1321952 kilobytes

5.8 RTL SCHEMATIC OF ENCRYPTION

The RTL schematic of encryption and its enlarged version is shown in Fig 5.6.1, Fig 5.6.2 and in Fig 5.6.3. This indicates that when an input data (128-bits), input key (128-bits) and the control signal are provided to the module, an output-enciphered data of 128-bits is obtained.

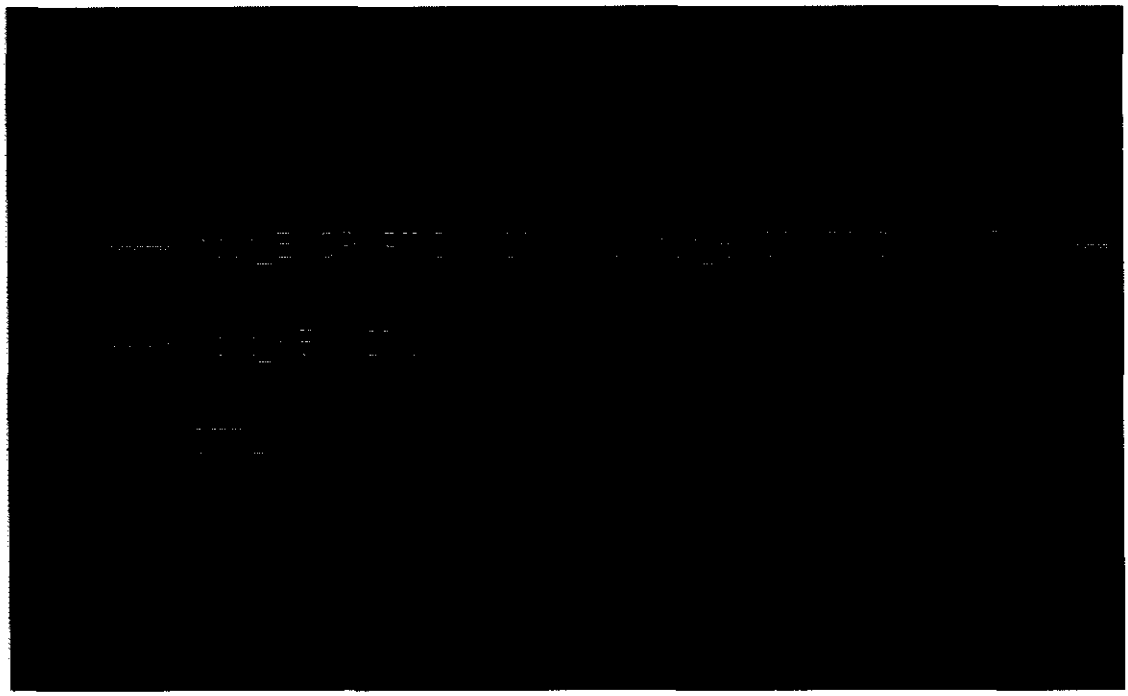


Fig 5.6 RTL Schematic of Encryption

Fig 5.7 Enlarged RTL schematic-1 of Encryption

Fig 5.8 Enlarged RTL schematic-2 of Encryption

5.9 RTL SCHEMATIC OF DECRYPTION

The RTL schematic of decryption and its enlarged version is shown in Fig 5.7.1, Fig 5.7.2 and in Fig 5.7.3. This indicates that when an input enciphered data (128-bits), input key (128-bits), control and clock signal are provided to the module, an output-deciphered data of 128-bits is obtained.

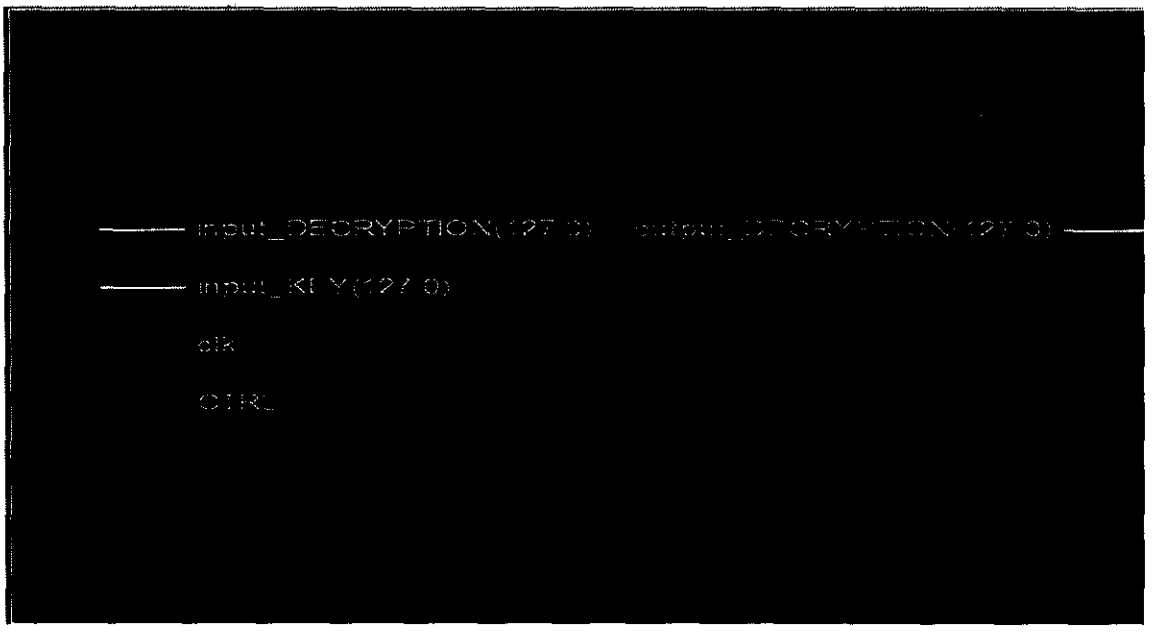


Fig 5.9 RTL schematic of Decryption

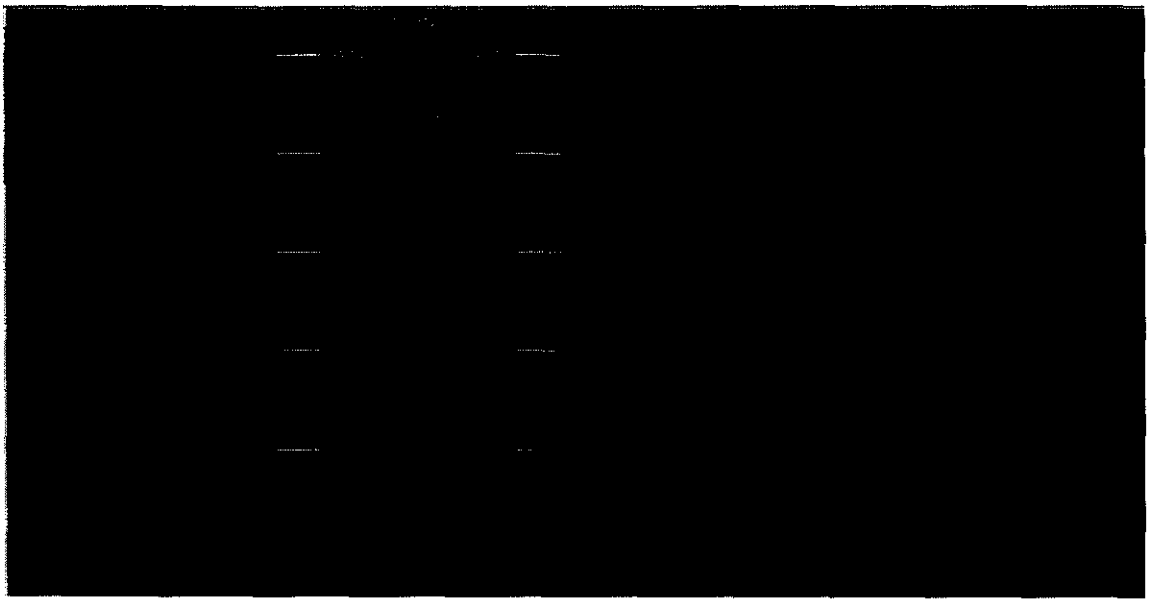


Fig 5.10 Enlarged RTL schematic-1 of Decryption

Fig 5.11 Enlarged RTL schematic-2 of Decryption

Thus detailed analyses of the use of device resources and the simulation results have been showcased in this chapter.

CHAPTER 6
CONCLUSION

The current project has been implemented at software level. It has focused on providing a solution to the difficulties faced by the most recent standards such as Triple DES. The drawbacks of Triple DES are easy cryptanalysis and sluggish performance. Implementation of AES has rendered cryptanalysis difficult, thus providing more security.

Possible future enhancements could be reading a file, be it text or image and encrypting the same, extension of the 16-byte input key to 24 bytes or more. Implementation can also be done in a real-time environment.

REFERENCES

1. Charles P.Pfleeger, Shari Lawrence Pfleeger (2004) 'Security in Computing' , Pearson education, Third edition.

2. Douglas Perry (1998) 'VHDL', McGraw-Hill International Editions, Third edition.

3. Volnei A.Pedroni (2005) 'Circuit design with VHDL', Prentice hall of India Pvt Ltd.

4. Wenbo Mao (2004) 'Modern Cryptography, Theory and Practice', Pearson Education, Third edition.

5. William Stallings (2003) 'Cryptography and Network security, Principles and Practices', Pearson Education, Third edition.