"EFFICEINT ENCRYPTION AND DECRYPTION SYSTEM: FPGA Implementation of Cryptographic Algorithms using Verilog"

A Project Report

Submitted by:

MD JANNATUL NAYEM (2014166)

SUMAN SUTRADHAR (2014008)

ASHRAFUJJAMAN RAKY (2014165)

Under the Supervision of

PROF. SRIMANTA BAISHYA

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, SILCHAR

SILCHAR, ASSAM (INDIA)-788010

MAY - 2024

DECLARATION

I hereby declare that the project entitled **“Efficient Encryption and Decryption System: FPGA Implementation of Cryptographic Algorithms using Verilog”** submitted for the B. Tech. (ECE) degree is our original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

MD Jannatul Nayem (2014166)	Suman Sutradhar (2014008)	Ashrafujjaman Raky (2014165)

Place:

Date:

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, ASSAM, INDIA

CERTIFICATE

This is to certify that the project titled **“Efficient Encryption and Decryption System: FPGA Implementation of Cryptographic Algorithms using Verilog”** is the bonafide work carried out by MD Jannatul Nayem (2014166), Suman Sutradhar (2014008) and Ashrafujjaman Raky (2014165), student of B.Tech in ECE of National
Institute of Technology, Silchar (An Institute of National Importance under MHRD, Govt. of India), Silchar, Assam, India. During the academic year 2023-24, in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (Electronics and Communication Engineering) and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

Prof. Srimanta Baishya
(Supervisor)

Place:

Date:

NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR, ASSAM, INDIA

ABSTRACT

The "internet of things" (IoT), a well-known technology, is made up of numerous embedded devices, sensors, and other things connected to the Internet. There are several applications that use it. Because of how quickly this technology has developed, it currently accounts for a sizable fraction of current research interests. Most Internet of Things (IoT) devices are built with the capability of gathering various types of data from a variety of sources and transmitting it digitally. Data security is the most important problem with IoT technology since it significantly affects the privacy of important data. In this study, a novel, lightweight, resource-constrained 16-bit, 32-bit, 64-bit and 128-bit symmetric key encryption technique have been suggested for IoT devices for the efficient encryption and decryption process that can offer data security at the sensing level. This algorithm increased security by combining the structural benefits of the substitution-permutation network (SPN) and the Feistel structure. In this project, we have designed the encryption and decryption modules for 16-bit, 32-bit, 64-bit and 128-bit and simulated all the modules on Xilinx Vivado software. We are planning to evaluate the proposed method on the NEXYS 4 DDR FPGA (Artix-7) trainer kit, we are using Xilinx Vivado EDA tool.

ACKNOWLEDGEMENT

The essence of a successful project lies in proper guidance of the guide. We express our deepest appreciation to **Prof. Srimanta Baishya** sir, our B.Tech project supervisor, for his invaluable suggestions, guidance, and feedback. His support not only encouraged us but also aided in identifying and correcting errors.

We would like to express our special thanks to **Mr. Sourav Nath** sir for his valuable guidance.

We would also like to acknowledge the systematic guidelines provided by the Department of Electronics and Communication Engineering regarding the presentation and the sincerity of all the students to excel in it. We may also extend our thanks to our team itself for working as a team with constructive setbacks and opinions and overcoming the difficulties that came.

Lastly, special thanks to the panellists for their constructive comments on our work and for showing us the right direction to improve our project furthermore.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

Cryptography is a method by which we can secure the transmissions of confidential data, private information etc. Cryptography means converting any data into random form (encrypted data) that is not decodable to everyone using a secret key. We need an efficient cryptography that can deal with the data size, device power and cost of computing devices to a minimal level. So, while designing a cryptography algorithm dedicated to any small computing device, the main motive should be made it lightweight in every aspect such as memory usage, chip size, power consumption etc.

1.1 Problem Identification

IoT helps the modern world gradually become smarter through a variety of difficult application areas, such as smart healthcare, smart agriculture, smart energy grids, home and building automation, smart traffic, etc. In this new data environment, IoT is expanding quickly, but due to lax security and privacy standards, it may not be able to do so sustainably. In order to increase IoT device layer security and ensure the preservation of crucial sensed data, it is important to implement more security mechanisms into the hardware and software of the device. Applying the idea of cryptography to the physical constraints of IoT devices may be useful in this case. Data integrity can be preserved, and hacker data snooping prevented by using cryptographic algorithms to encrypt data at IoT devices before delivering it to back-end systems (cloud servers).

1.2 Project Overview

In this project, we have implemented a symmetric cryptographic approach based on the LRBC cryptographic algorithm. Our design involves 16-bit, 32-bit, 64-bit and 128-bit symmetric encryption and decryption techniques. For 16-bit encryption process, it takes a 16-bit plaintext and a 16-bit key as the inputs and generates 16-bit encrypted data known as ciphertext. This encryption process contains 24 rounds. The last round output is considered as the main ciphertext or encrypted text. On the other hand, during the decryption process, the 16-bit ciphertext and the same 16-bit key are used as the inputs and produces 16-bit decrypted text which is considered as the original message. The decryption process also consists of 24 rounds. The last round output is considered as the decrypted text. A similar process happened in the case of 32-bit, 64-bit and 128-bit.

In case of 32-bits encryption process, 32-bits plaintext (message) and 32-bits key are used as the inputs for the encryption process, and it generates the ciphertext (encrypted text) of 32-bits. During the decryption process, 32-bits encrypted text (ciphertext) which is produced from the encryption module and the same key of 32-bits are used as the inputs and it produces 32-bits decrypted text which is our original plaintext or message.

In case of 64-bits encryption process, it takes 64-bits plaintext (message) and 64-bits key as the inputs and generates the ciphertext (encrypted text) of 64-bits. Again, in the decryption process, 64-bits ciphertext (encrypted text) and the same key of 64-bits are used as the inputs, and it generates the decrypted text or plaintext or message of 64-bits length.

And in case of 128-bits encryption process, 128-bits plaintext (message) and 128-bits key are used as the inputs for the encryption process, and it generates the ciphertext (encrypted text) of 128-bits. During the decryption process, 128-bits encrypted text (ciphertext) which is produced from the encryption module and the same key of 128-bits are used as the inputs and it produces 128-bits decrypted text which is our original plaintext or message.

1.3 Hardware Specification

We have used Nexys 4 DDR Artix-7 FPGA Board for the hardware implementation of proposed 16-bit, 32-bit, 64-bit and 128-bit cryptographic algorithm.

1.4 Software Specification

For writing verilog codes, simulation of waveforms and verification of results, we have used Xilinx Vivado software. We have used Virtual Input/Output (VIO) module to take the inputs and to display the output waveform, we have used the Integrated Logic Analyzer (ILA) module.

1.5 Terminologies

- **Encryption:** Encryption is the process of converting plain text (message) to a cipher text (encrypted text).
- **Decryption:** Decryption is the process of conversion of cipher text (encrypted text) to plain text (original message).
- **Ciphertext:** The ciphertext is the encrypted text generated as the output after the encryption process. It requires plain text and the key.
- **Key:** Any text which is used to encrypt the plain text or to make the message random or in other form

Cryptography is primarily classified into two categories. The first one is symmetric key cryptography and the second is asymmetric key cryptography. In today's world, it is necessary to ensure the security and privacy of information that enables secure online transactions, protecting sensitive data stored in the database, ensures confidentiality of the communication. However, in this project, we are following the concept of symmetric key encryption and decryption method.

1.6 Symmetric Key Cryptography

In this symmetric encryption technique, the sender uses a key to encrypt the data and convert into encrypted form or ciphertext. On the other hand, during decryption process, the receiver uses the same key which is used for encryption to decrypt the encrypted data. The most used symmetric key algorithm is the Data Encryption Standards (DES). DES encryption algorithm is mainly used in banking applications where personal information needs to be secured. Another popular symmetric key algorithm is Advanced Encryption Standard (AES). AES is widely used today, and it is much stronger than DES and Triple DES.

1.7 Asymmetric Key Cryptography

In case of asymmetric encryption technique, both the sender and receiver have two different keys. These are public key and private key. The sender uses receiver's public key to encrypt the data. This public key is accessible to everyone. On the other hand, in case of decryption process, the receiver uses its own private key to decrypt the encrypted data. The sender uses the receiver's public key to encrypt the data and covert into encrypted text also known as ciphertext, and only that receiver whose public key is used for encryption can decrypt the data. The most popular asymmetric key system is the RSA (Rivest-Shamir-Adleman) algorithm.

CHAPTER 2: LITERATURE REVIEW

2.1 Existing System

Cryptographic algorithms are classified in two types based on the key used. Those are symmetric and asymmetric cryptography algorithms. In case of symmetric cryptography, same key is used for encryption as well as decryption processes. Algorithms like AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are the most popular symmetric encryption algorithms. The sender encrypts the data using a secret key and the receiver uses the same secret key to decrypt the transmitted data (original data).

On the other hand, in case of asymmetric cryptography, two different keys are used. These are a public key and a private key. The public key is accessible to everyone, and the sender encrypts the data using this public key. But the private key is kept secret and only the authorized receiver has the access to private key. Only authorized receiver can decrypt the encrypted data using private key. RSA and Elliptic Curve Cryptography (ECC) are commonly used asymmetric encryption algorithms.

Table 2.1 Comparison between existing cryptographic algorithms

Serial No.	Algorithm	Type of Cryptography	Key Size	Block Size	No. of Round	Feature
1.	LRBC	Symmetric	16 - bits	16 - bits	24	Light weight cryptography
2.	Data Encryption Standards (DES)	Symmetric	64 - bits	64 - bits	16	Not strong enough
3.	Advanced Encryption Standard (AES)	Symmetric	128, 192, 256 bits	128 - bits	10, 12, 14	Good secure
4.	RSA (Rivest-Shamir-Adleman)	Asymmetric	1024 to 4096 bits	128 - bits	1	Excellent security but low speed

2.2 Proposed System

Here, we have followed the approach of symmetric cryptography based on the 16-bit encryption technique known as the LRBC cryptographic algorithm. We have designed and implemented a 16-bit, 32-bit, 64-bit and 128-bit symmetric cryptography encryption and decryption technique in our architecture. In case of 16-bit technique, 16-bit plaintext and a 16-bit key are entered into the encryption process, which results in the creation of encrypted data known as ciphertext of 16-bit. This algorithm has 24 cycles for the encryption and decryption procedures. Using the same key used for encryption, the ciphertext is fed into the decryption phase, where the output is decrypted text that is exactly the same as the original plaintext.

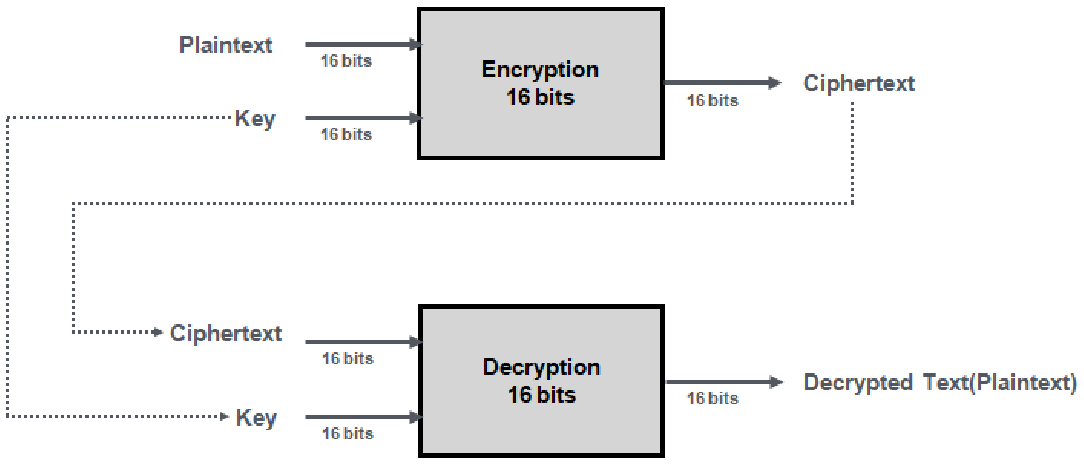


Figure 2.1 Block diagram of LRBC algorithm for 16-bits data

We have used 4x1 multiplexer so that only one operation is active and remaining operations are stopped. Depending upon the mode (select line), we can select which operation to be active. If mode is 00, then only 16-bit operation is active and remaining operations are turned off. If mode is 01, then only 32-bit operation is active and remaining operations are turned off. Similarly, if mode is 10, then only 64-bits operation works and for mode = 11, only 128-bits operation is active remaining operations are disabled. This is how, our designed module consumes less power and less area.

2.3 Feasibility Study

Secure data must be protected from unwanted access using cryptographic techniques. To make sure that only authorized people have access to read the data, these techniques are employed to encrypt and decode it. Various cryptographic algorithms exist, each with unique advantages and disadvantages. The performance of a cryptographic algorithm is an important consideration, especially for applications where data needs to be encrypted and decrypted quickly. The most important factors affecting the performance of a cryptographic algorithm are:

- Key size: The length of the key used by the algorithm. Longer keys typically provide better security, but they also require more computational power to encrypt and decrypt data.
- Block size: The size of the input data block that the algorithm operates on. Larger block sizes can improve performance, but they also increase the amount of memory required.
- Algorithm complexity: The complexity of the algorithm's mathematical operations. More complex algorithms typically provide better security, but they also require more computational power etc.

However, this algorithm can be used to securely transmit the data any IoT devices, sensors, and other objects connected to the Internet. In the field of military applications, we can use this algorithm to protect confidential information. In the medical field, we can secure biometrics. Not only that, we can use this cryptography technology for banking security such as ATM services, secure online transactions and more.

As we have already mentioned that we have used 4x1 multiplexer so that only one operation is active and remaining operations are stopped. Depending upon the mode (select line), we can select which operation to be active. If mode is 00, then only 16-bit operation is active and remaining operations are disabled. So, at a time only one type of operation is active. Also, in this project we have implemented both encryption and decryption processes on the same module. But in the real world, we will use the encryption module on the transmitter (sender) and decryption module will be used on the receiver. This is how we can make our encryption and decryption techniques more efficient.

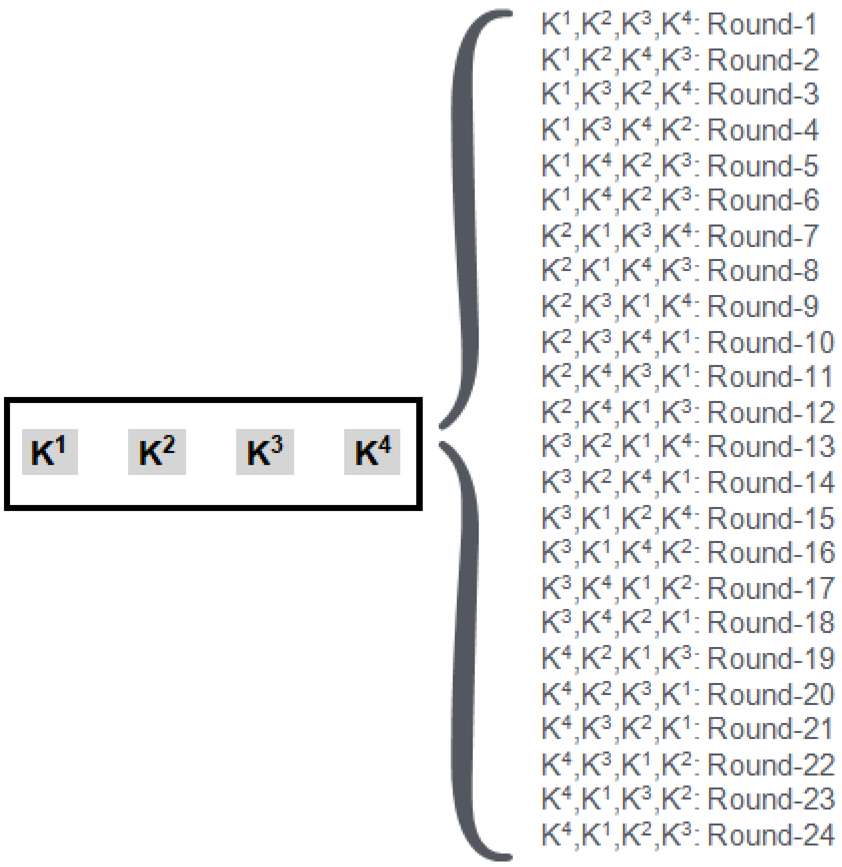
CHAPTER 3: SYSTEM ANALYSIS & DESIGN

3.1 Requirement Specification

In this project, we have written verilog code for both the encryption and decryption process. For this we have used the Xilinx Vivado EDA tool. In this software, we have written the verilog code, then verified using simulation. For hardware implementation, we have used Nexys 4 DDR Artix-7 FPGA board. To take input, we have used Virtual Input/Output (VIO) and to display the output waveform, we have used the Integrated Logic Analyzer (ILA). Both of these are available on Xilinx Vivado software.

3.2 Key Description

In the proposed approach for 32-bits encryption-decryption technique, four numbers of keys having 8-bit each (K^1, K^2, K^3 and K^4) have been used for both the encryption and decryption processes. Because a lightweight encryption is desired in this circumstance, a basic yet powerful combination of keys has been considered. There are 24 different combinations from the four keys listed above that will be used during the round operation of the encryption/decryption process. As a result, it ensures that it is resistant to similar key attacks. Figure: 3.1 depicts the key combination design.



K^1 K^2 K^3 K^4
K^1,K^2,K^3,K^4: Round-1
K^1,K^2,K^4,K^3: Round-2
K^1,K^3,K^2,K^4: Round-3
K^1,K^3,K^4,K^2: Round-4
K^1,K^4,K^2,K^3: Round-5
K^1,K^4,K^2,K^3: Round-6
K^2,K^1,K^3,K^4: Round-7
K^2,K^1,K^4,K^3: Round-8
K^2,K^3,K^1,K^4: Round-9
K^2,K^3,K^4,K^1: Round-10
K^2,K^4,K^3,K^1: Round-11
K^2,K^4,K^1,K^3: Round-12
K^3,K^2,K^1,K^4: Round-13
K^3,K^2,K^4,K^1: Round-14
K^3,K^1,K^2,K^4: Round-15
K^3,K^1,K^4,K^2: Round-16
K^3,K^4,K^1,K^2: Round-17
K^3,K^4,K^2,K^1: Round-18
K^4,K^2,K^1,K^3: Round-19
K^4,K^2,K^3,K^1: Round-20
K^4,K^3,K^2,K^1: Round-21
K^4,K^3,K^1,K^2: Round-22
K^4,K^1,K^3,K^2: Round-23
K^4,K^1,K^2,K^3: Round-24

Figure 3.1: 8-bit key combinations

For 16-bits operation Key is of 16-bits and K^1, K^2, K^3, K^4 all are of 4 bits.

For 32-bits operation Key is of 32-bits and K^1, K^2, K^3, K^4 all are of 8 bits.

For 64-bits operation Key is of 64-bits and K^1, K^2, K^3, K^4 all are of 16 bits.

For 128-bits operation Key is of 128-bits and K^1, K^2, K^3, K^4 all are of 32 bits.

3.3 Flowcharts

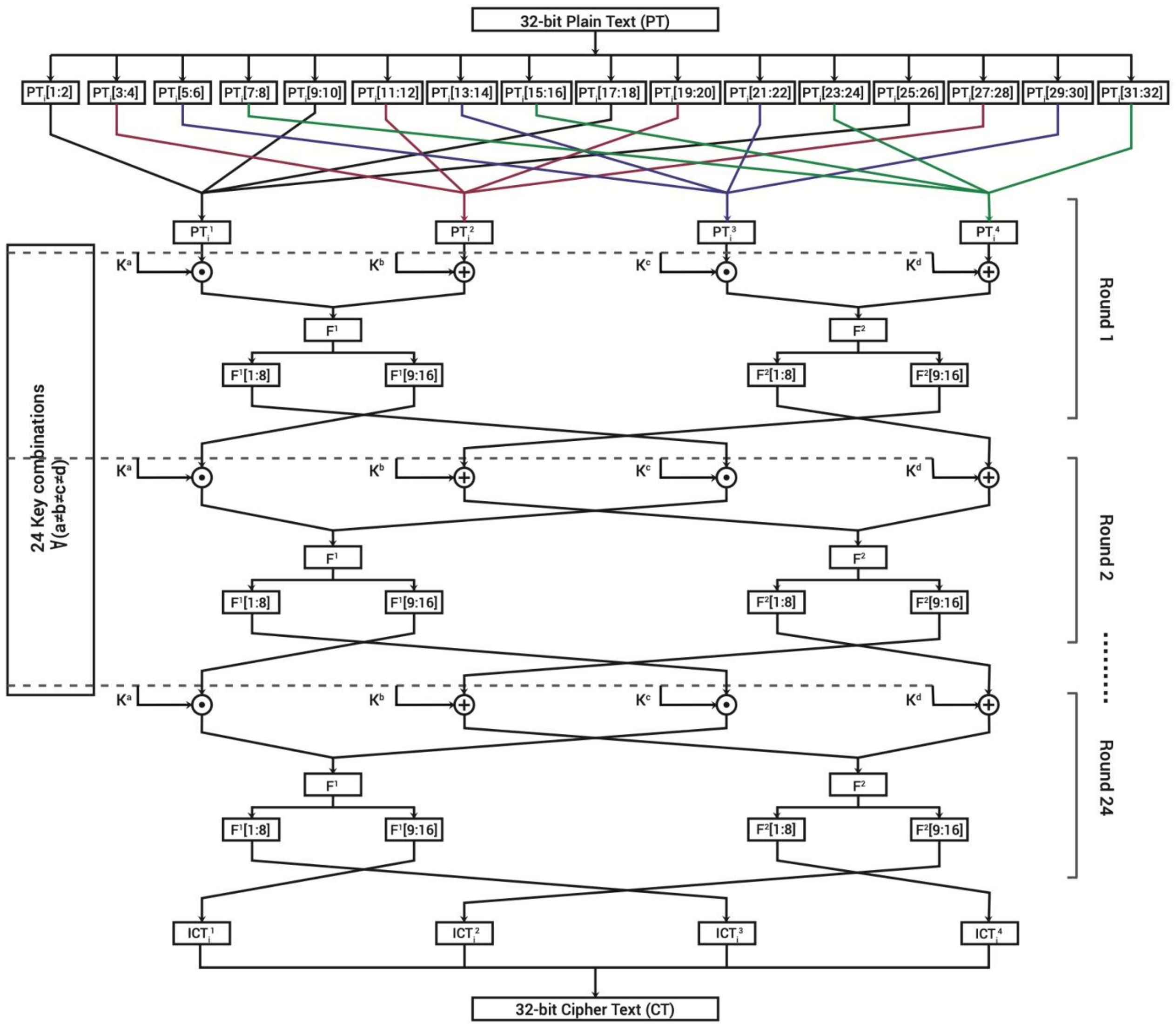

32-bit Plain Text (PT)
PT_i[1:2] PT_i[3:4] PT_i[5:6] PT_i[7:8] PT_i[9:10] PT_i[11:12] PT_i[13:14] PT_i[15:16] PT_i[17:18] PT_i[19:20] PT_i[21:22] PT_i[23:24] PT_i[25:26] PT_i[27:28] PT_i[29:30] PT_i[31:32]
PT_i^1 PT_i^2 PT_i^3 PT_i^4
K^a K^b K^c K^d
F^1 F^2
F^1[1:8] F^1[9:16] F^2[1:8] F^2[9:16]
24 Key combinations ∀(a≠b≠c≠d)
Round 1
Round 2
Round 24
ICT_i^1 ICT_i^2 ICT_i^3 ICT_i^4
32-bit Cipher Text (CT)

Figure 3.2: Flow diagram of encryption process for 32-bits using LRBC algorithm

3.4 Algorithms and Design Steps

Algorithm 1 shows the LRBC encryption technique for 32-bits data, whereas Figure: 3.2 is the graphical representation for encryption process. Encryption method repeats the techniques such as round transposition, F-function, and so on for 24 rounds. Each round generates intermediate ciphertexts (ICT), which are used as input for the next round. The final ciphertext (CT) is determined by the results of the 24th round.

3.4.1 Algorithm 1:

Input: 32-bit plaintext (PT), Four no's of 8-bit keys (K^a, K^b, K^c, K^d)

Output: 32-bit ciphertext (CT)

1. Read the byte values from the input file called plaintext (PT) and extract the byte values.

2. PT divided into 'n' number of blocks of 32 bits each (PT_i).

3. Initialize 'r' with value '1'.

4. Each PT_i is further sub-divided into 4 equal length parts $PT_j^k \Big| \begin{array}{l} 1 \le k \le 4 \\ 1 \le i \le n \end{array}$ as,

$$PT_i^1 = PT_i[1] \,||\, PT_i[2] \,||\, PT_i[9] \,||\, PT_i[10] \,||\, PT_i[17] \,||\, PT_i[18] \,||\, PT_i[25] \,||\, PT_i[26]$$

$$PT_i^2 = PT_i[3] \,||\, PT_i[4] \,||\, PT_i[11] \,||\, PT_i[12] \,||\, PT_i[19] \,||\, PT_i[20] \,||\, PT_i[27] \,||\, PT_i[28]$$

$$PT_i^3 = PT_i[5] \,||\, PT_i[6] \,||\, PT_i[13] \,||\, PT_i[14] \,||\, PT_i[21] \,||\, PT_i[22] \,||\, PT_i[29] \,||\, PT_i[30]$$

$$PT_i^4 = PT_i[7] \,||\, PT_i[8] \,||\, PT_i|15] \,||\, PT_i[16] \,||\, PT_i[23] \,||\, PT_i[24] \,||\, PT_i[31] \,||\, PT_i[32]$$

5. Compute intermediate round cipher blocks as,

$$IC_i^1 = PT_i^1 \odot K^a \;\Big|\; \begin{array}{l} 1 \le a \le 4 \\ a \ne b \ne c \ne d \end{array}$$

$$\left.\begin{array}{l} IC_i^2 = PT_i^2 \oplus K^b \\ IC_i^3 = PT_i^3 \odot K^c \\ IC_i^4 = PT_i^4 \oplus K^d \end{array}\right|\; \begin{array}{l} 1 \le b \le 4 \\ 1 \le c \le 4 \\ 1 \le d \le 4 \\ a \ne b \ne c \ne d \end{array}$$

6. Generate F-Function as,

$$F_i^1 = F\_Function(IC_i^1, IC_i^3);$$

$$F_i^2 = F\_Function(IC_i^2, IC_i^4)$$

```
7. Generate input for next round as,
        $PT_i^1 = F_i^1[9:16]; PT_i^2 = F_i^2[9:16]$
        $PT_i^3 = F_i^1[1:8]; PT_i^4 = F_i^2[1:8]$
        $r = r+1$
8. If ($r < 24$)
        Go to step 5.
9. Else
        Go to step 10.
10. $ICT_i^k \Big|_{\substack{1 \le k \le 4 \\ 1 \le i \le n}} = PT_i^k \Big|_{\substack{1 \le k \le 4 \\ 1 \le i \le n}}$
11. Generate Final Cipher as,
        $CT = ICT_i^1 \,||\, ICT_i^2 \,||\, ICT_i^3 \,||\, CT_i^4$
```

3.4.2 Algorithm 2: F-Function

Input: Intermediate cipher blocks $IC_i^1, IC_i^2, IC_i^3, IC_i^4$.

Output: 32 - bit ciphertext

```
1. S-box computation:
        $IS_i^1 = IC_i^1 \odot IC_i^3$
        $IS_i^2 = IC_i^1 \oplus 1$
        $IS_i^3 = IC_i^2 \odot IC_i^4$
        $IS_i^4 = IC_i^2 \oplus 0$
2. P-box computation:
        $P_i^1 = IS_i^1[1] || IS_i^2[8] || IS_i^1[2] || IS_i^2[7] || IS_i^1[3] || IS_i^2[6] || IS_i^1[4] || IS_i^2[5]$
        $P_i^2 = IS_i^1[5] || IS_i^2[4] || IS_i^1[6] || IS_i^2[3] || IS_i^1[7] || IS_i^2[2] || IS_i^1[8] || IS_i^2[1]$
        $P_i^3 = IS_i^3[1] || IS_i^4[8] || IS_i^3[2] || IS_i^4[7] || IS_i^3[3] || IS_i^4[6] || IS_i^3[4] || IS_i^4[5]$
        $P_i^4 = IS_i^3[5] || IS_i^4[4] || IS_i^3[6] || IS_i^4[3] || IS_i^3[7] || IS_i^4[2] || IS_i^3[8] || IS_i^4[1]$
3. L-box computation:
        $T_i[1] = (P_i^1[1] \oplus P_i^3[8]);$    $X_i[1] = (P_i^1[1] \odot 0);$
        $T_i[2] = (P_i^1[2] \odot P_i^3[7]);$    $X_i[2] = (P_i^1[2] \oplus 1);$
        $T_i[3] = (P_i^1[3] \oplus P_i^3[6]);$    $X_i[3] = (P_i^2[1] \odot 0);$
        $T_i[4] = (P_i^1[4] \odot P_i^3[5]);$    $X_i[4] = (P_i^2[2] \oplus 1);$
        $T_i[5] = (P_i^1[5] \oplus P_i^3[4]);$      $X_i[5] = (P_i^1[3] \odot 0);$
        $T_i[6] = (P_i^1[6] \odot P_i^3[3]);$      $X_i[6] = (P_i^1[4] \oplus 1);$
        $T_i[7] = (P_i^1[7] \oplus P_i^3[2]);$      $X_i[7] = (P_i^2[3] \odot 0);$
        $T_i[8] = (P_i^1[8] \odot P_i^3[1]);$      $X_i[8] = (P_i^2[4] \oplus 1);$
        $T_i[9] = (P_i^2[1] \oplus P_i^4[8]);$      $X_i[9] = (P_i^1[5] \odot 0);$
        $T_i[10] = (P_i^2[2] \odot P_i^4[7]);$    $X_i[10] = (P_i^1[6] \oplus 1);$
        $T_i[11] = (P_i^2[3] \oplus P_i^4[6]);$    $X_i[11] = (P_i^2[5] \odot 0);$
        $T_i[12] = (P_i^2[4] \odot P_i^4[5]);$    $X_i[12] = (P_i^2[6] \oplus 1);$
        $T_i[13] = (P_i^2[5] \oplus P_i^4[4]);$    $X_i[13] = (P_i^1[7] \odot 0);$
        $T_i[14] = (P_i^2[6] \odot P_i^4[3]);$    $X_i[14] = (P_i^1[8] \oplus 1);$
        $T_i[15] = (P_i^2[7] \oplus P_i^4[2]);$    $X_i[15] = (P_i^2[7] \odot 0);$
        $T_i[16] = (P_i^2[8] \odot P_i^4[1]);$    $X_i[16] = (P_i^2[8] \oplus 1);$

        $F_i(1) = T_i[1] \,||\, X_i[8] \,||\, T_i[2] \,||\, X_i[7] \,||\, T_i[3] \,||\, X_i[6] \,||\, T_i[4] \,||\, X_i[5]$
                $||\, T_i[5] \,||\, X_i[4] \,||\, T_i[6] \,||\, X_i[3] \,||\, T_i[7] \,||\, X_i[2] \,||\, T_i[8] \,||\, X_i[1]$

        $F_i(2) = T_i[9] \,||\, X_i[16] \,||\, T_i[10] \,||\, X_i[15] \,||\, T_i[11] \,||\, X_i[14] \,||\, T_i[12] \,||\, X_i[13]$
                $||\, T_i[13] \,||\, X_i[12] \,||\, T_i[14] \,||\, X_i[11] \,||\, T_i[15] \,||\, X_i[10] \,||\, T_i[16] \,||\, X_i[9]$

        $F = F_i(1) \,||\, F_i(2)$
4.  End.
```

3.5 F-Function Generation

According to this algorithm, each round has two F-functions consisting of three types of computing boxes, namely S-box, P-box, and L-box. Under each F-function, the calculated output of the S-box is given to the input of the P-box and similarly the calculated output of the P-box is applied as the input of the L-box. The computations performed by the F-function have been illustrated in Algorithm 2.

3.6 Decryption Process

It is undoubtedly important to have a successful decryption process when designing a strong encryption process. It also consists of 24 rounds. The decryption process of this algorithm is designed in such a way that it is exactly similar to encryption technology but in reverse order. Since we are using symmetric key cryptography, thus same key is used for decryption process i.e, total 24 different combinations from the four keys having 8-bit each (K^1, K^2, K^3 and K^4) for 24 rounds in case of 32-bits technique.

3.7 Types of Operation

This encryption and decryption techniques consist of 4 types of operations. These are 16-bit operation, 32-bit operation, 64-bit operation and 128-bit operation.

Here, we have used 4x1 multiplexer to implement our design. It has 2 select lines which collectively act as the mode of operation selection. If mode = 00, then 16-bit operation is selected, and remaining operations are turned off. If mode = 01, then 32-bit operation is selected, and remaining operations are turned disabled. Again, if mode = 10, then 64-bit operation is active and remaining operations are turned off. And if mode = 11, then 128-bit operation is selected, and remaining operations are disabled. Since at a time only one operation is active, thus our designed module consumes less power and less area.

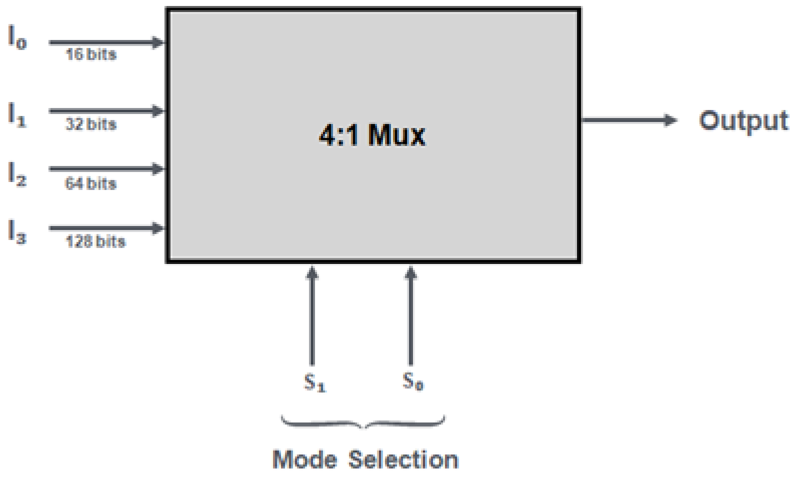


Figure 3.3: 4x1 Multiplexer for mode selection

Mode Selection

S1	S0	Output
0	**0**	**I_0**
0	**1**	**I_1**
1	**0**	**I_2**
1	**1**	**I_3**

Figure 3.4: Truth Table of 4x1 Multiplexer

3.7.1 16-bit Operation

In the encryption process, it takes 16-bits plaintext (message) and 16-bits key as the inputs and generates the ciphertext (encrypted text) of 16-bits. As we have mentioned that encryption process has total 24 rounds. The output produced from the last round is considered as the ciphertext (encrypted text).

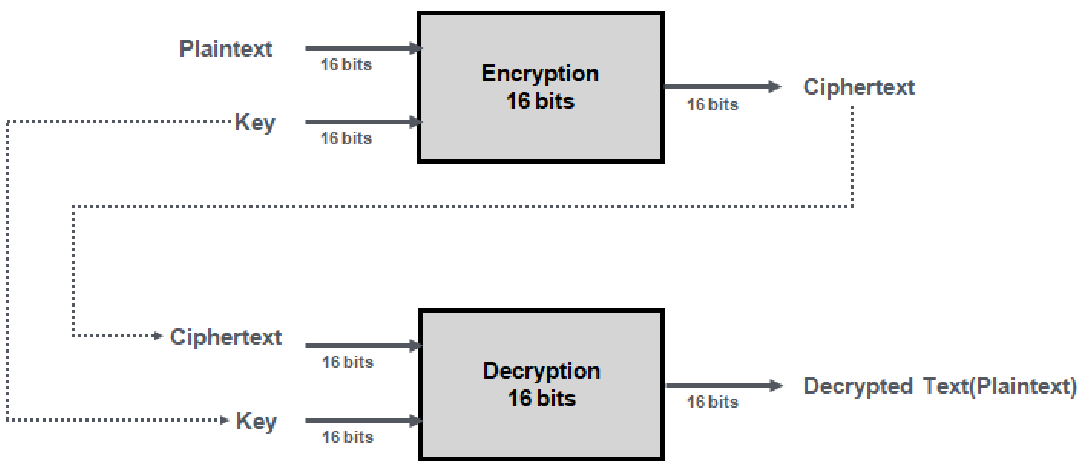


Figure 3.5: Block diagram of encryption and decryption processes for 16-bits

Again, in the decryption process, 16-bits encrypted text (ciphertext) and the same key of 16-bits are used as the inputs, and it generates 16-bits decrypted text which is same as the original plaintext.

3.7.2 32-bit Operation

Here 32-bits plaintext (message) and 32-bits key are used as the inputs for the encryption process, and it generates the ciphertext (encrypted text) of 32-bits.

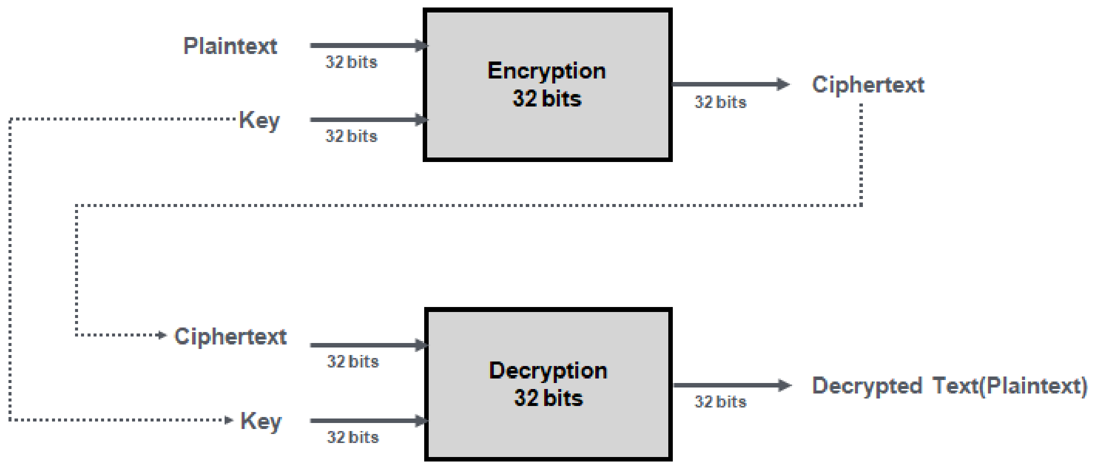


Figure 3.6: Block diagram of encryption and decryption processes for 32-bits

During the decryption process, 32-bits encrypted text (ciphertext) which is produced from the encryption module and the same key of 32-bits are used as the inputs and it produces 32-bits decrypted text which is our original plaintext or message.

3.7.3 64-bit Operation

In the encryption process, it takes 64-bits plaintext (message) and 64-bits key as the inputs and generates the ciphertext (encrypted text) of 64-bits.

Again, in the decryption process, 64-bits ciphertext (encrypted text) and the same key of 64-bits are used as the inputs, and it generates the decrypted text or plaintext or message of 64-bits length.

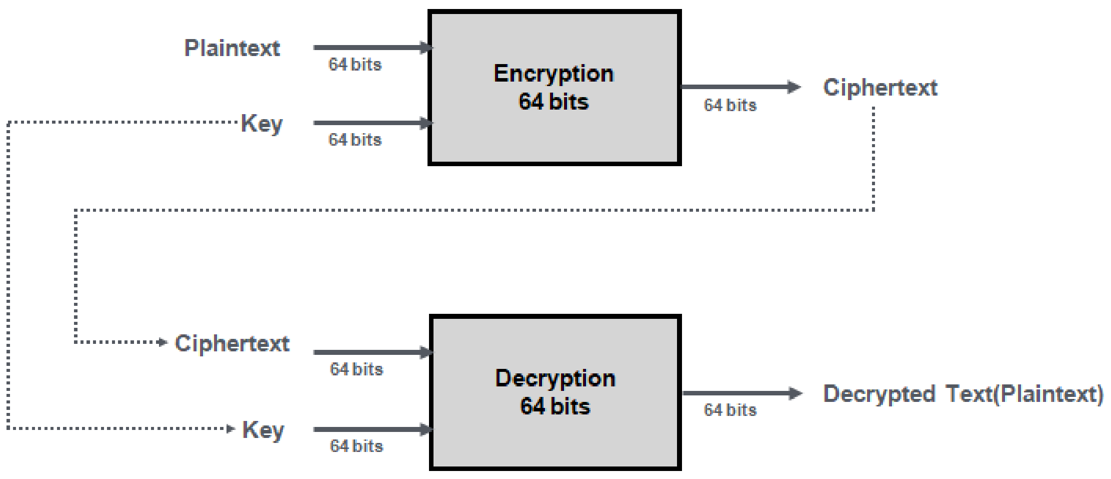


Figure 3.7: Block diagram of encryption and decryption processes for 64-bits

3.7.4 128-bit Operation

Here 128-bits plaintext (message) and 128-bits key are used as the inputs for the encryption process, and it generates the ciphertext (encrypted text) of 128-bits.

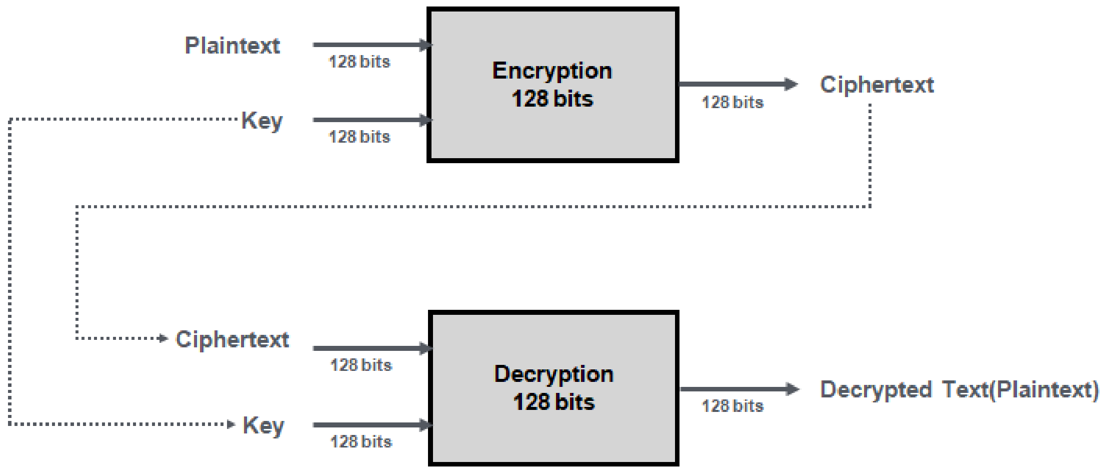


Figure 3.8: Block diagram of encryption and decryption processes for 128-bits

During the decryption process, 128-bits encrypted text (ciphertext) which is produced from the encryption module and the same key of 128-bits are used as the inputs and it produces 128-bits decrypted text which is our original plaintext or message.

3.8 Testing Process

We can test this algorithm in two ways. The first is on the Xilinx Vivado software by simply looking at the input and output waveforms which is called simulation. This process does not require an FPGA board. And the second way is that first we must synthesize the Verilog code, then apply it to the FPGA board and with the help of different input and output ports we can easily verify the results. However, we have used Virtual Input/Output (VIO) for taking the inputs up to 128 bits and to display the output waveform as well, we have used the Integrated Logic Analyzer (ILA). Both are available on Xilinx Vivado software.

CHAPTER 4:   RESULTS

4.1 Simulation and Testing

We have designed a module for encryption and decryption which consists of 4 sub-modules 16-bit, 32-bit, 64-bit and 128-bit using Verilog hardware description language. We have simulated and tested this symmetric cryptographic algorithm for various inputs and successfully verified the results by observing the waveforms on Xilinx Vivado software. Finally, we have implemented this encryption and decryption process on Nexys 4 DDR Artix-7 FPGA board and verified the results successfully.

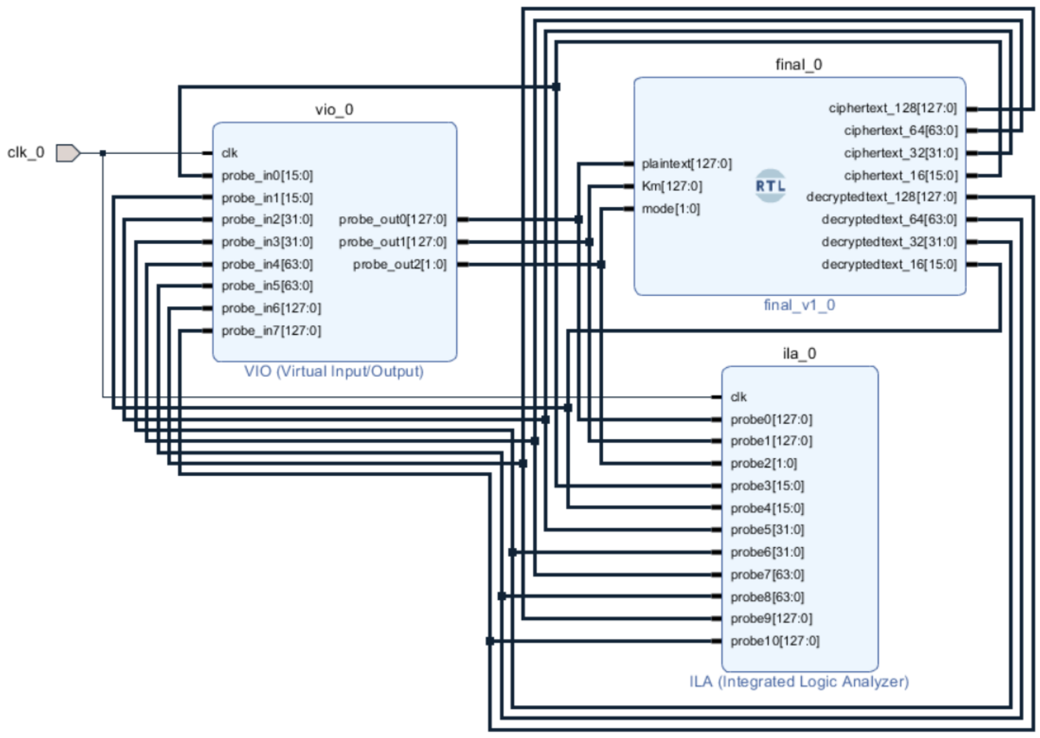


Figure 4.1: Block level design using VIO and ILA modules.

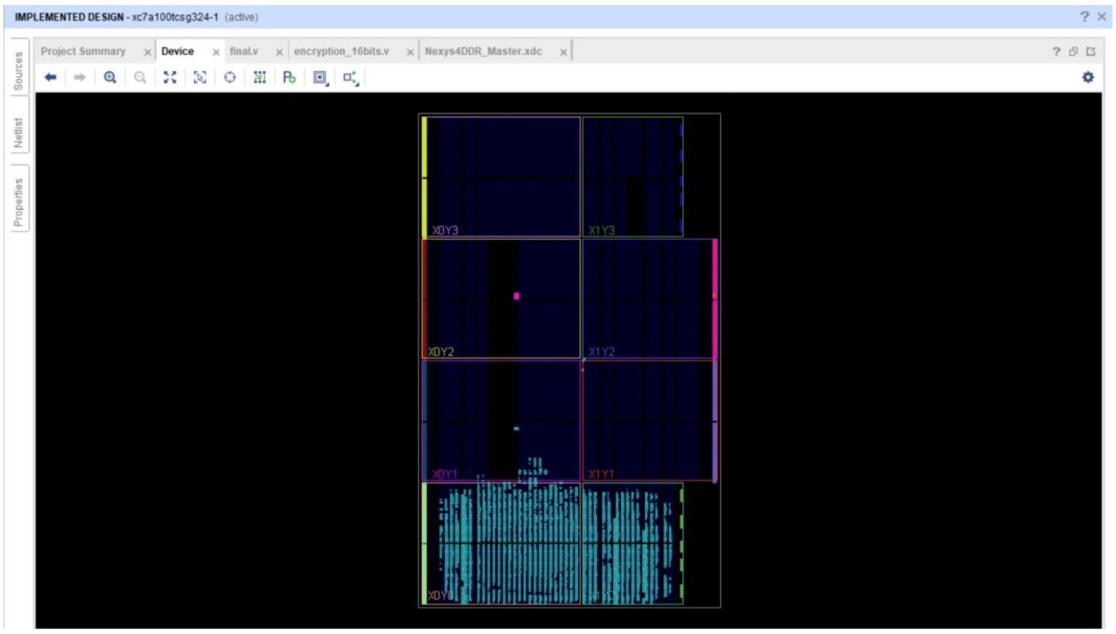
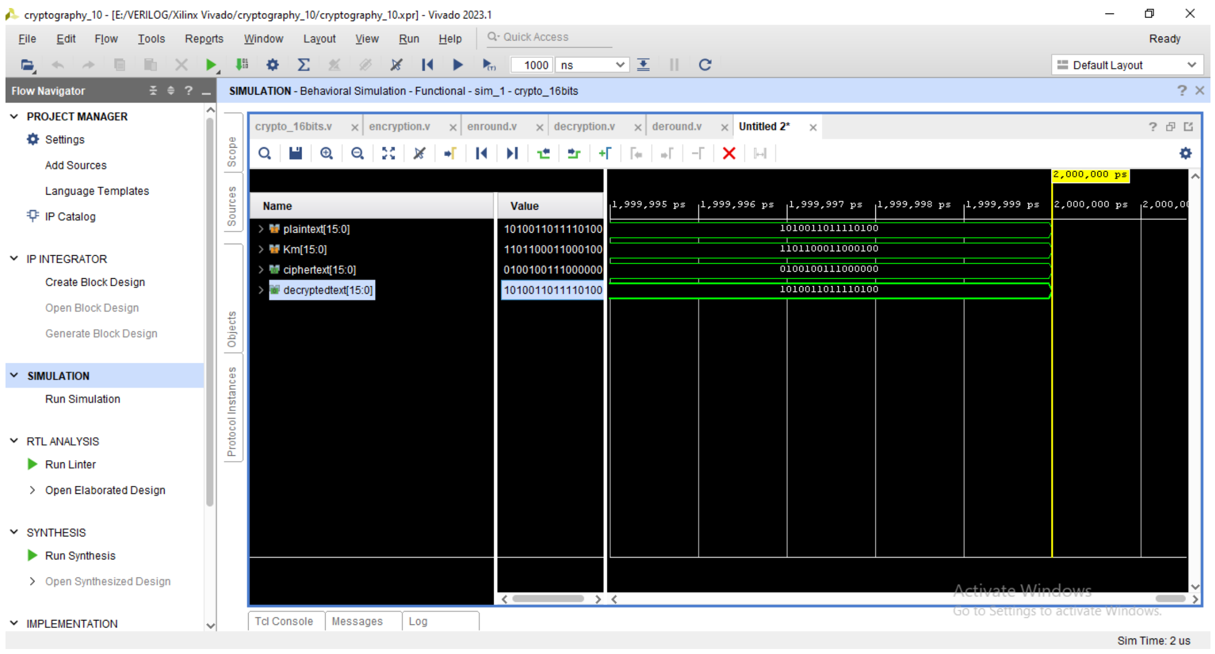


Figure 4.2: Physical design for both encryption and decryption

4.2 Simulated Outputs

Case – I:

Here, mode (S_1 S_0) = 00, then 16-bit operation is selected.



```
Name                 Value
plaintext[15:0]      1010011011110100
Km[15:0]             1101100011000100
ciphertext[15:0]     0100100111000000
decryptedtext[15:0]  1010011011110100
```

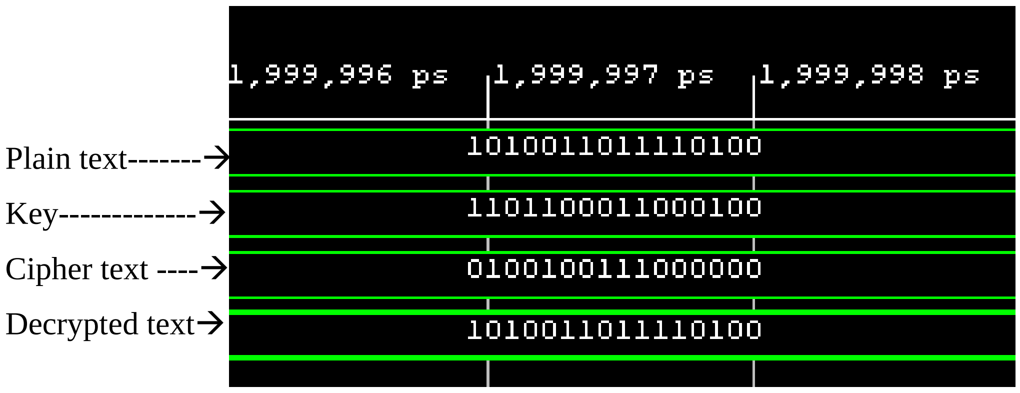
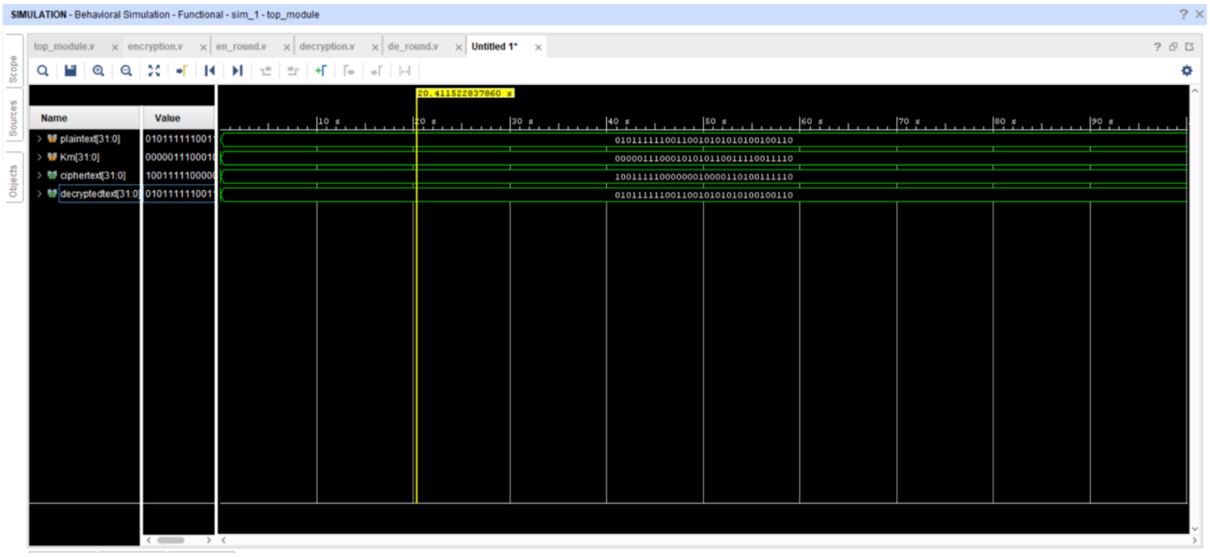


Figure 4.3: Simulated output for encryption and decryption processes for 16-bits data

Case – II:

Here, mode (S_1 S_0) = 01, then 32-bit operation is selected.

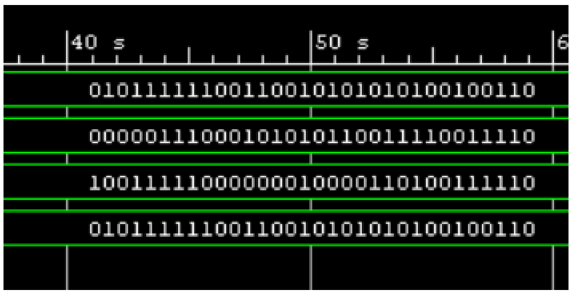

SIMULATION - Behavioral Simulation - Functional - sim_1 - top_module

top_module.v | encryption.v | en_round.v | decryption.v | de_round.v | Untitled 1*

20.411522837860 s

Name	Value
plaintext[31:0]	010111111001
Km[31:0]	000001110001
ciphertext[31:0]	100111110000
decryptedtext[31:0]	010111111001

```
01011111110011001010101010100100110
00000111000101010110011110011110
10011111000000001000011010011110
01011111110011001010101010100100110
```



Plain text-------→ 01011111110011001010101010100100110

Key-------------→ 00000111000101010110011110011110

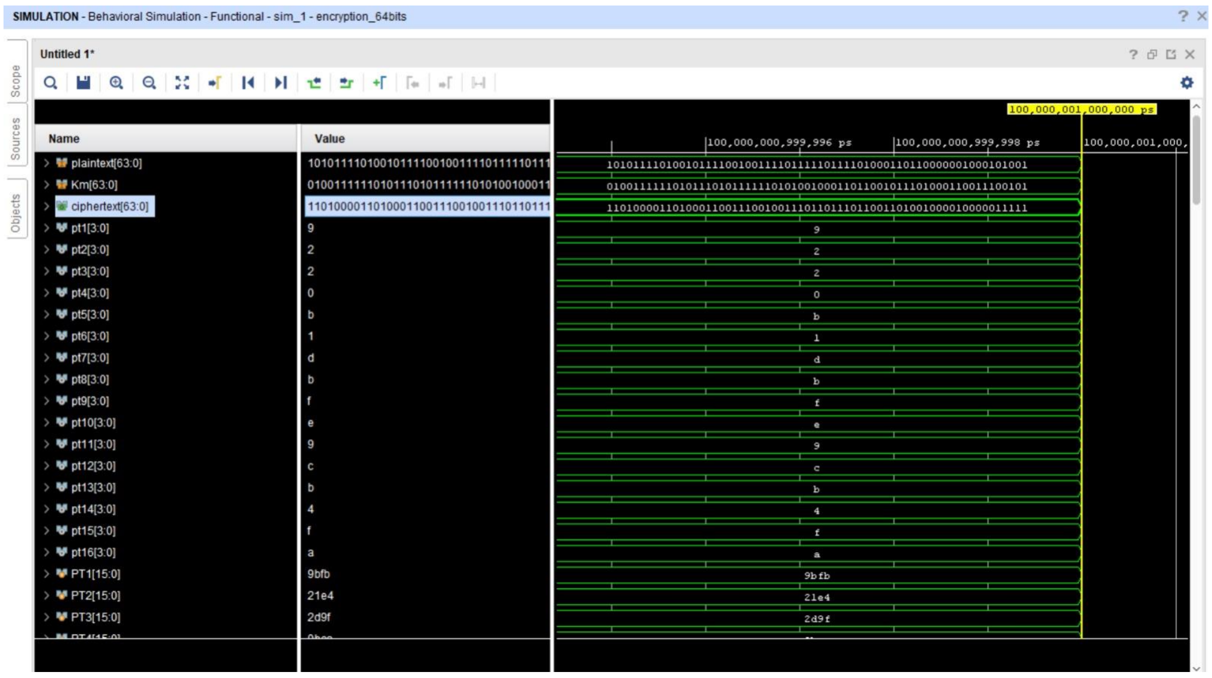
Cipher text-----→ 10011111000000001000011010011110

Decrypted text-→ 01011111110011001010101010100100110

Figure 4.4: Simulated output for encryption and decryption process for 32-bits data

Case – III:

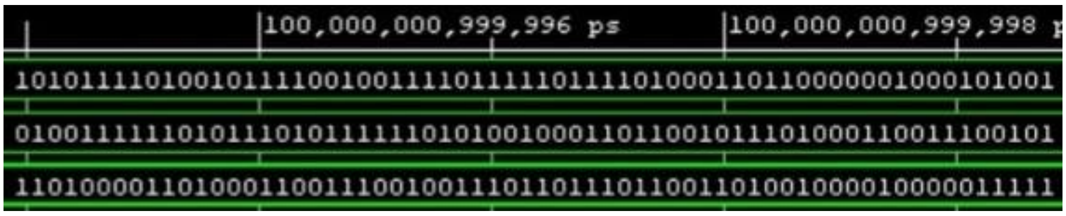
Here, mode (S_1 S_0) = 10, then 64-bit operation is selected.



```
SIMULATION - Behavioral Simulation - Functional - sim_1 - encryption_64bits
Untitled 1*

Name              Value
plaintext[63:0]   10101111010010111100100111101111110111
Km[63:0]          01001111110101110101111110101001000011
ciphertext[63:0]  11010000110100011001110010011101101111
pt1[3:0]          9
pt2[3:0]          2
pt3[3:0]          2
pt4[3:0]          0
pt5[3:0]          b
pt6[3:0]          1
pt7[3:0]          d
pt8[3:0]          b
pt9[3:0]          f
pt10[3:0]         e
pt11[3:0]         9
pt12[3:0]         c
pt13[3:0]         b
pt14[3:0]         4
pt15[3:0]         f
pt16[3:0]         a
PT1[15:0]         9bfb
PT2[15:0]         21e4
PT3[15:0]         2d9f

100,000,001,000,000 ps
|100,000,000,999,996 ps   |100,000,000,999,998 ps   100,000,001,000,
1010111101001011110010011110111111011110100011011000000010001010 01
0100111111010111010111111010100100011011001011101000110011100101
1101000011010001100111001001110110111011001101001000010000011111
```



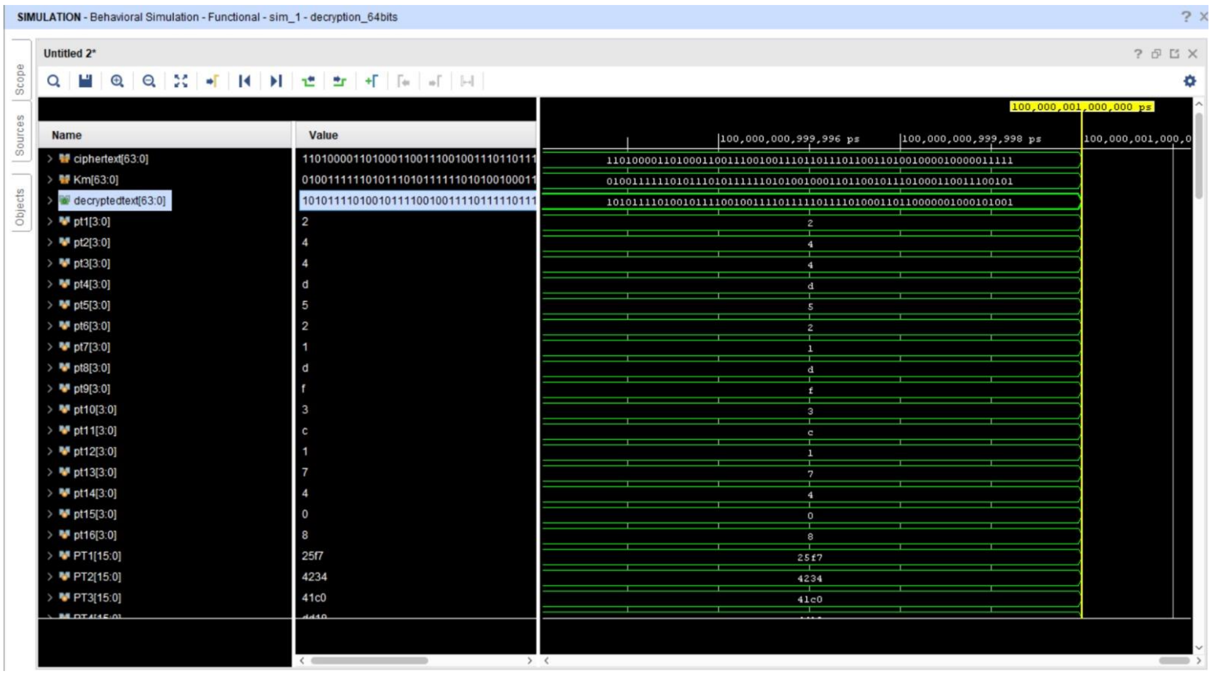
```
|100,000,000,999,996 ps   |100,000,000,999,998
```

Plain text 1010111101001011110010011110111111011110100011011000000010001010 01

Key 0100111111010111010111111010100100011011001011101000110011100101

Cipher text 1101000011010001100111001001110110111011001101001000010000011111

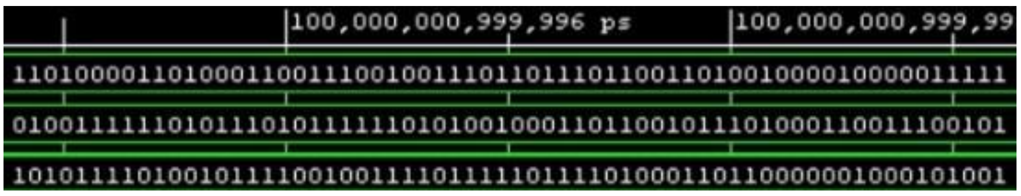
Figure 4.5: Simulated output for encryption process for 64-bits data



```
SIMULATION - Behavioral Simulation - Functional - sim_1 - decryption_64bits
Untitled 2*

Name                Value
ciphertext[63:0]    1101000011010001100111001001110110111
Km[63:0]            0100111111010111010111111010100100011
decryptedtext[63:0] 1010111101001011110010011110111110111
pt1[3:0]            2
pt2[3:0]            4
pt3[3:0]            4
pt4[3:0]            d
pt5[3:0]            5
pt6[3:0]            2
pt7[3:0]            1
pt8[3:0]            d
pt9[3:0]            f
pt10[3:0]           3
pt11[3:0]           c
pt12[3:0]           1
pt13[3:0]           7
pt14[3:0]           4
pt15[3:0]           0
pt16[3:0]           8
PT1[15:0]           25f7
PT2[15:0]           4234
PT3[15:0]           41c0

100,000,001,000,000 ps
|100,000,000,999,996 ps   |100,000,000,999,998 ps   100,000,001,000,0
1101000011010001100111001001110110111011001101001000010000011111
0100111111010111010111111010100100011011001011101000110011100101
1010111101001011110010011110111111011110100011011000000001000101001
```



```
|100,000,000,999,996 ps   |100,000,000,999,99
```

Cipher text 1101000011010001100111001001110110111011001101001000010000011111

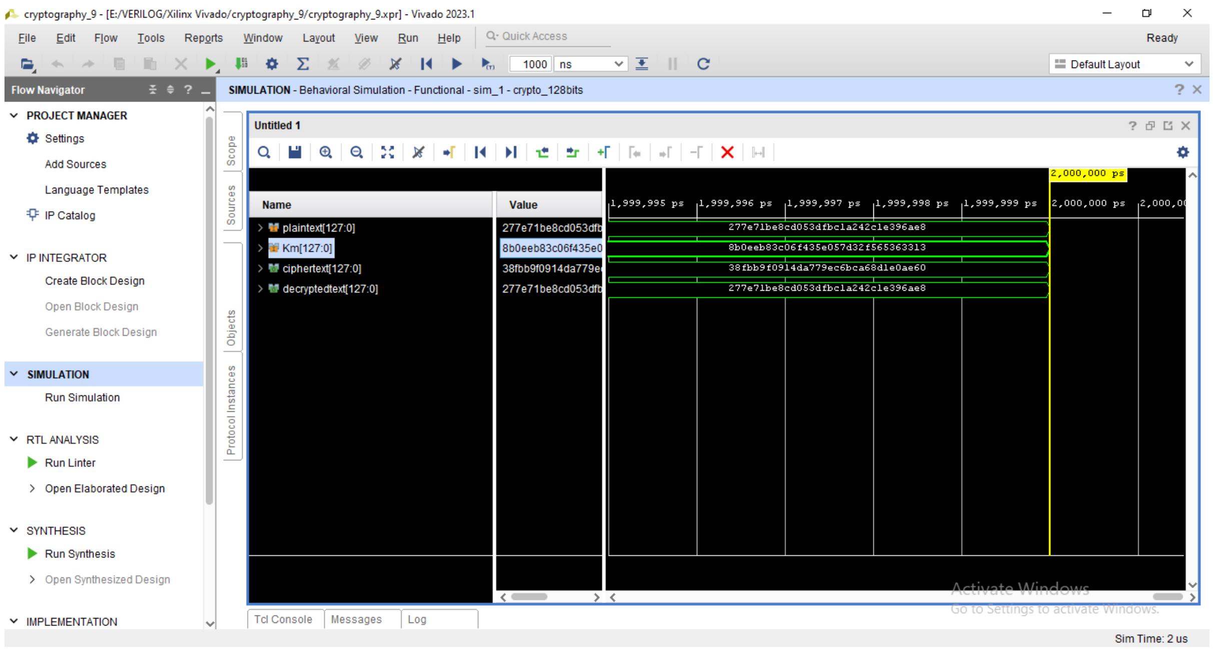
Key 0100111111010111010111111010100100011011001011101000110011100101

Decrypted text 1010111101001011110010011110111111011110100011011000000001000101001

Figure 4.6: Simulated output for decryption process for 64-bits data

Case – IV:

Here, mode (S_1 S_0) = 11, then 128-bit operation is selected.

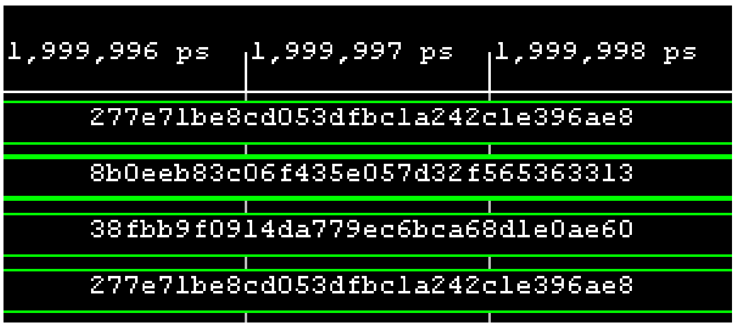


cryptography_9 - [E:/VERILOG/Xilinx Vivado/cryptography_9/cryptography_9.xpr] - Vivado 2023.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - crypto_128bits

Name	Value
plaintext[127:0]	277e71be8cd053dfbc1a242c1e396ae8
Km[127:0]	8b0eeb83c06f435e057d32f565363313
ciphertext[127:0]	38fbb9f0914da779ec6bca68d1e0ae60
decryptedtext[127:0]	277e71be8cd053dfbc1a242c1e396ae8

Sim Time: 2 us



Plain text-------→ 277e71be8cd053dfbc1a242c1e396ae8

Key--------------→ 8b0eeb83c06f435e057d32f565363313

Cipher text-----→ 38fbb9f0914da779ec6bca68d1e0ae60

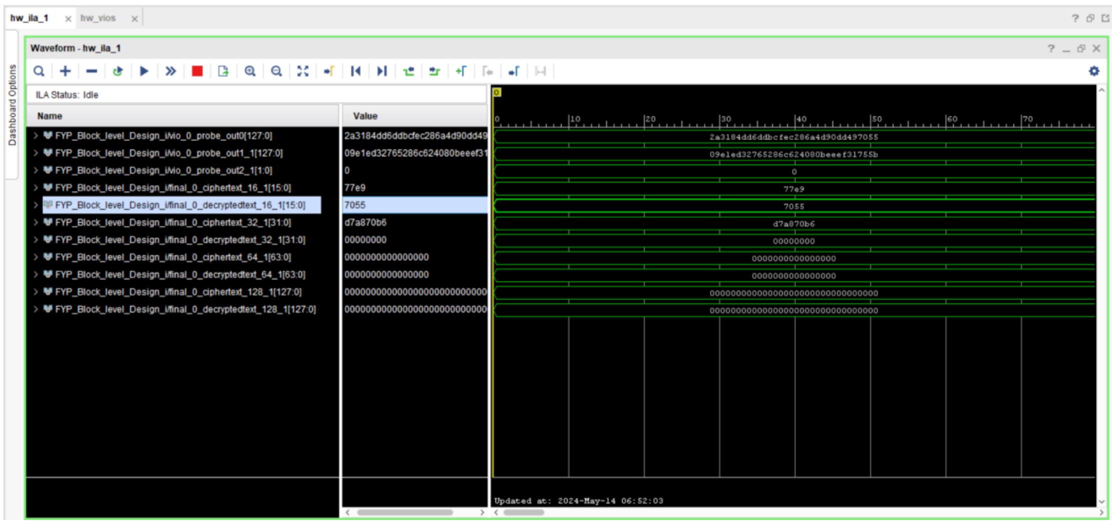
Decrypted text-→ 277e71be8cd053dfbc1a242c1e396ae8

Figure 4.7: Simulated output for encryption and decryption process for 128-bits data

4.3 Implemented Outputs using FPGA

Case – I:

Here, mode (S_1 S_0) = 00, then 16-bit operation is selected.



hw_ila_1 | hw_vios

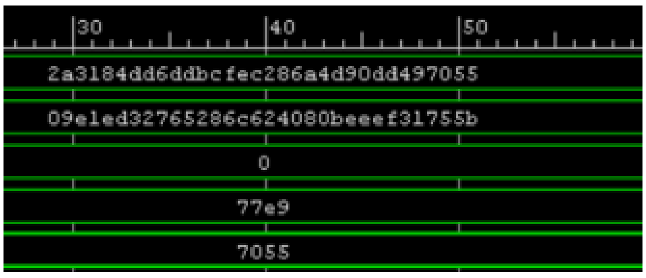
Waveform - hw_ila_1

ILA Status: Idle

Name	Value
FYP_Block_level_Design_i/vio_0_probe_out0[127:0]	2a3184dd6ddbcfec286a4d90dd49
FYP_Block_level_Design_i/vio_0_probe_out1_1[127:0]	09e1ed32765286c624080beeef31
FYP_Block_level_Design_i/vio_0_probe_out2_1[1:0]	0
FYP_Block_level_Design_i/final_0_ciphertext_16_1[15:0]	77e9
FYP_Block_level_Design_i/final_0_decryptedtext_16_1[15:0]	7055
FYP_Block_level_Design_i/final_0_ciphertext_32_1[31:0]	d7a870b6
FYP_Block_level_Design_i/final_0_decryptedtext_32_1[31:0]	00000000
FYP_Block_level_Design_i/final_0_ciphertext_64_1[63:0]	0000000000000000
FYP_Block_level_Design_i/final_0_decryptedtext_64_1[63:0]	0000000000000000
FYP_Block_level_Design_i/final_0_ciphertext_128_1[127:0]	0000000000000000000000000000
FYP_Block_level_Design_i/final_0_decryptedtext_128_1[127:0]	0000000000000000000000000000

```
2a3184dd6ddbcfec286a4d90dd497055
09e1ed32765286c624080beeef31755b
0
77e9
7055
d7a870b6
00000000
0000000000000000
0000000000000000
00000000000000000000000000000000
00000000000000000000000000000000

Updated at: 2024-May-14 06:52:03
```

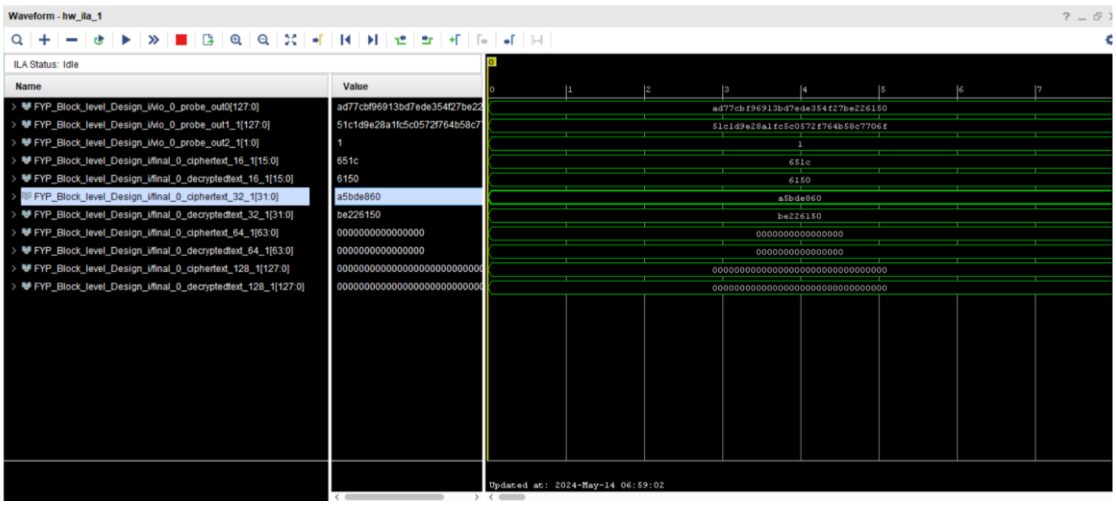


Plain text	2a3184dd6ddbcfec286a4d90dd497055
Key	09e1ed32765286c624080beeef31755b
Mode	0
Cipher text	77e9
Decrypted text	7055

Figure 4.8: FPGA implemented output for 16-bits data.

Case – II:

Here, mode (S_1 S_0) = 01, then 32-bit operation is selected.



Waveform - hw_ila_1

ILA Status: Idle

Name	Value	
FYP_Block_level_Design_i/vio_0_probe_out0[127:0]	ad77cbf96913bd7ede354f27be22	ad77cbf96913bd7ede354f27be226150
FYP_Block_level_Design_i/vio_0_probe_out1_1[127:0]	51c1d9e28a1fc5c0572f764b58c7	51c1d9e28a1fc5c0572f764b58c7706f
FYP_Block_level_Design_i/vio_0_probe_out2_1[1:0]	1	1
FYP_Block_level_Design_i/final_0_ciphertext_16_1[15:0]	651c	651c
FYP_Block_level_Design_i/final_0_decryptedtext_16_1[15:0]	6150	6150
FYP_Block_level_Design_i/final_0_ciphertext_32_1[31:0]	a5bde860	a5bde860
FYP_Block_level_Design_i/final_0_decryptedtext_32_1[31:0]	be226150	be226150
FYP_Block_level_Design_i/final_0_ciphertext_64_1[63:0]	0000000000000000	0000000000000000
FYP_Block_level_Design_i/final_0_decryptedtext_64_1[63:0]	0000000000000000	0000000000000000
FYP_Block_level_Design_i/final_0_ciphertext_128_1[127:0]	0000000000000000000000000000	00000000000000000000000000000000
FYP_Block_level_Design_i/final_0_decryptedtext_128_1[127:0]	0000000000000000000000000000	00000000000000000000000000000000

Updated at: 2024-May-14 06:59:02

Plain text

Key

Mode

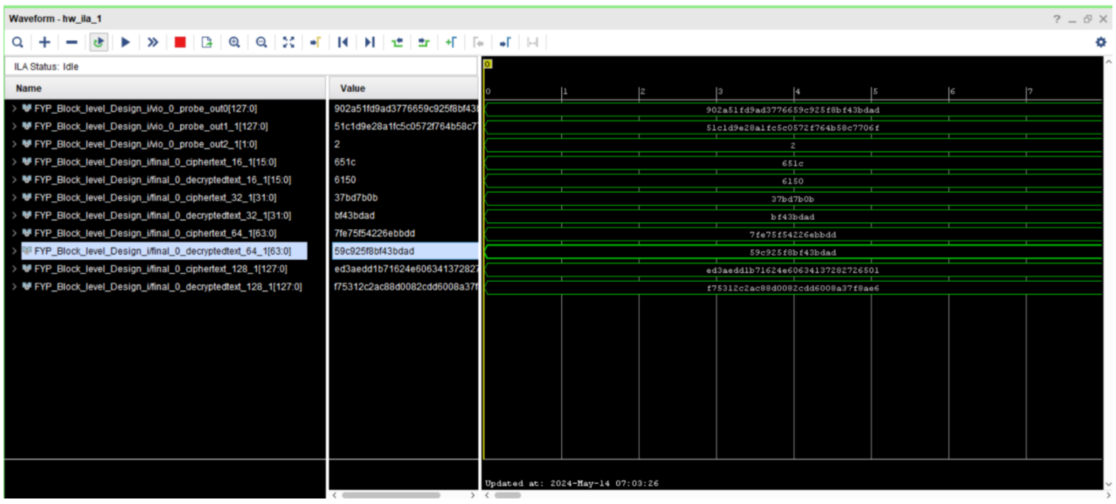
Cipher text.

Decrypted text

Figure 4.9: FPGA implemented output for 32-bits data

Case – III:

Here, mode (S_1 S_0) = 10, then 64-bit operation is selected.

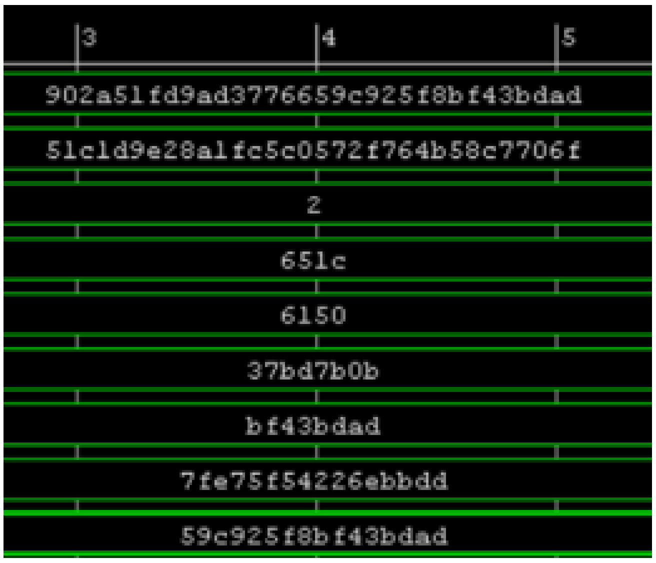


Waveform - hw_ila_1

ILA Status: Idle

Name	Value
FYP_Block_level_Design_i/vio_0_probe_out0[127:0]	902a51fd9ad3776659c925f8bf43bdad
FYP_Block_level_Design_i/vio_0_probe_out1_1[127:0]	51c1d9e28a1fc5c0572f764b58c7706f
FYP_Block_level_Design_i/vio_0_probe_out2_1[1:0]	2
FYP_Block_level_Design_i/final_0_ciphertext_16_1[15:0]	651c
FYP_Block_level_Design_i/final_0_decryptedtext_16_1[15:0]	6150
FYP_Block_level_Design_i/final_0_ciphertext_32_1[31:0]	37bd7b0b
FYP_Block_level_Design_i/final_0_decryptedtext_32_1[31:0]	bf43bdad
FYP_Block_level_Design_i/final_0_ciphertext_64_1[63:0]	7fe75f54226ebbdd
FYP_Block_level_Design_i/final_0_decryptedtext_64_1[63:0]	59c925f8bf43bdad
FYP_Block_level_Design_i/final_0_ciphertext_128_1[127:0]	ed3aedd1b71624e606341372827 26501
FYP_Block_level_Design_i/final_0_decryptedtext_128_1[127:0]	f75312c2ac88d0082cdd6008a37f8ae6

Updated at: 2024-May-14 07:03:26



Plain text 902a51fd9ad3776659c925f8bf43bdad

Key 51c1d9e28a1fc5c0572f764b58c7706f

Mode 2

651c

6150

37bd7b0b

bf43bdad

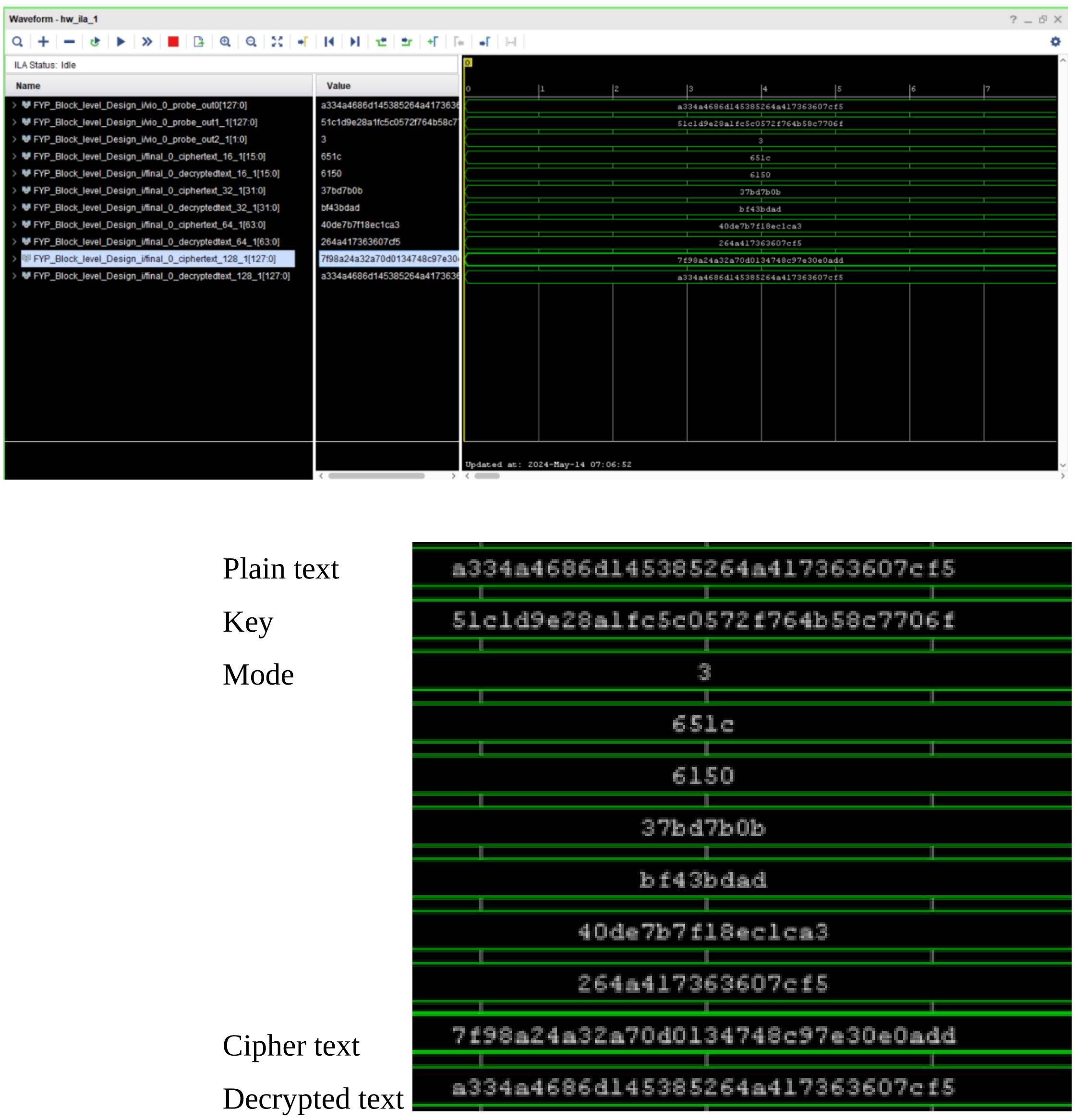
Cipher text 7fe75f54226ebbdd

Decrypted text 59c925f8bf43bdad

Figure 4.10: FPGA implemented output for 64-bits data

Case – IV:

Here, mode (S_1 S_0) = 11, then 128-bit operation is selected.



```
Name                                                    Value
FYP_Block_level_Design_i/vio_0_probe_out0[127:0]        a334a4686d145385264a4173636
FYP_Block_level_Design_i/vio_0_probe_out1_1[127:0]      51c1d9e28a1fc5c0572f764b58c7
FYP_Block_level_Design_i/vio_0_probe_out2_1[1:0]        3
FYP_Block_level_Design_i/final_0_ciphertext_16_1[15:0]  651c
FYP_Block_level_Design_i/final_0_decryptedtext_16_1[15:0]  6150
FYP_Block_level_Design_i/final_0_ciphertext_32_1[31:0]  37bd7b0b
FYP_Block_level_Design_i/final_0_decryptedtext_32_1[31:0]  bf43bdad
FYP_Block_level_Design_i/final_0_ciphertext_64_1[63:0]  40de7b7f18ec1ca3
FYP_Block_level_Design_i/final_0_decryptedtext_64_1[63:0]  264a417363607cf5
FYP_Block_level_Design_i/final_0_ciphertext_128_1[127:0]  7f98a24a32a70d0134748c97e30
FYP_Block_level_Design_i/final_0_decryptedtext_128_1[127:0]  a334a4686d145385264a4173636

a334a4686d145385264a417363607cf5
51c1d9e28a1fc5c0572f764b58c7706f
3
651c
6150
37bd7b0b
bf43bdad
40de7b7f18ec1ca3
264a417363607cf5
7f98a24a32a70d0134748c97e30e0add
a334a4686d145385264a417363607cf5

Updated at: 2024-May-14 07:06:52
```

Figure 4.11: FPGA implemented output for 128-bits data

4.4 Different Test Cases

4.4.1 Test Cases for 16-bit data

- **Test case 1:**

  Message: 1010011011110100

  Key: 1101100011000100

  Cipher text: 0100100111000000

  Decrypted text: 1010011011110100

- **Test case 2:**

  Message: 0100100001111101

  Key: 1001011101010010

  Cipher text: 1010100001111001

  Decrypted text: 0100100001111101

4.4.2 Test Cases for 32-bit data

- **Test case 1:**

  Message: 01011111100110010101010100100110

  Key: 00000111000101010110011110011110

  Cipher text: 10011111000000010000110100111110

  Decrypted text: 01011111100110010101010100100110

- **Test case 2:**

  Message: 11011100110011100101110100001000

  Key: 11000010111110001011001001010011

  Cipher text: 00010110000101001000111110110100

  Decrypted text: 11011100110011100101110100001000

4.4.3 Test Cases for 64-bit data

- **Test case 1:**

  Message: 1010111101001011110010011110111110111101000110110000001000101001

  Key: 0100111111010111010111111010100100011011001011101000110011100101

  Cipher text:

  1101000011010001100111001001110110111011001101001000010000011111

Decrypted text:

1010111101001011110010011110111110111101000110110000001000101001

4.4.4 Test Cases for 128-bit data

- **Test case 1:**

  Message: 277E71BE8CD053DFBC1A242C1E396AE8

  Key: 8B0EEB83C06F435E057D32F565363313

  Cipher text: 38FBB9F0914DA779EC6BCA68D1E0AE60

  Decrypted text: 277E71BE8CD053DFBC1A242C1E396AE8

- **Test case 2:**

  Message: 7EDAE43CFCB1B7F537A34584107FC033

  Key: 8A6E7724C31E8FE1EDF964A2412BB7F9

  Cipher text: 5FA20D2AE6D74D79FAF2B68492C89B3

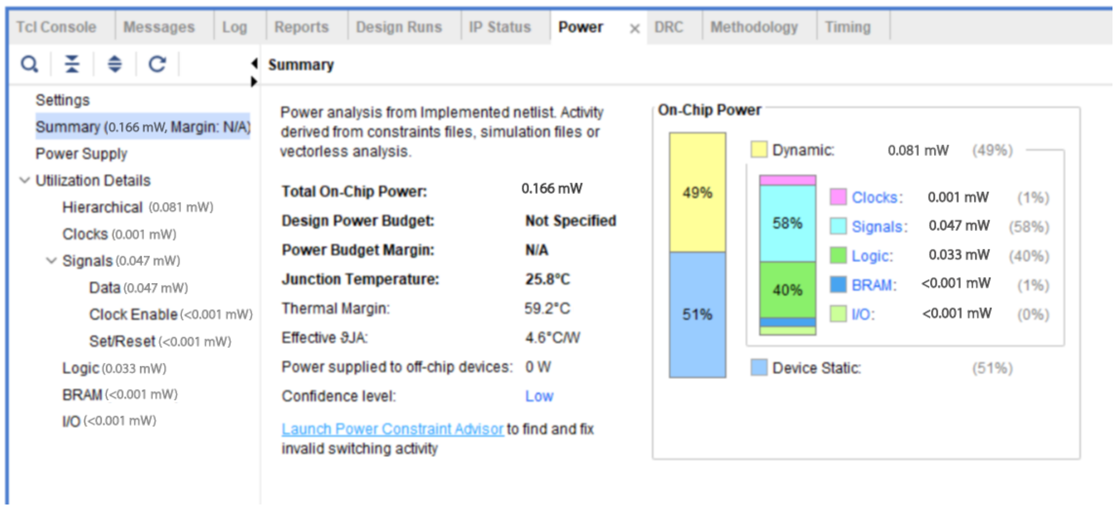
  Decrypted text: 7EDAE43CFCB1B7F537A34584107FC033

4.5 Analysis of Different Parameters

4.5.1 Area Report

```
****************************************
Report: area
Design: top_module_2
Version: L-2016.03
Date : Tue May 14 11:02:29 2024
****************************************
Library(s) Used: tcbn65gplusbwp12ttc_ccs (File:
/home/smart/Desktop/nayem/CCS/tcbn65gplusbwp12ttc_ccs.db)
Number of ports:                  4736
Number of nets:                   6826
Number of cells:                  2202
Number of combinational cells:         2154
Number of sequential cells:              0
Number of macros/black boxes:            0
Number of buf/inv:                     138
Number of references:                    2
Combinational area:            13578.240289
Buf/Inv area:                  198.720008
Noncombinational area:             0.000000
Macro/Black Box area:              0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)
Total cell area:              0.01357824 mm^2
Total area:              undefined
```

4.5.2 Power Report



Summary

Settings
Summary (0.166 mW, Margin: N/A)
Power Supply
Utilization Details
- Hierarchical (0.081 mW)
- Clocks (0.001 mW)
- Signals (0.047 mW)
  - Data (0.047 mW)
  - Clock Enable (<0.001 mW)
  - Set/Reset (<0.001 mW)
- Logic (0.033 mW)
- BRAM (<0.001 mW)
- I/O (<0.001 mW)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	0.166 mW
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25.8°C**
Thermal Margin:	59.2°C
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

Dynamic:		0.081 mW	(49%)
	Clocks:	0.001 mW	(1%)
	Signals:	0.047 mW	(58%)
	Logic:	0.033 mW	(40%)
	BRAM:	<0.001 mW	(1%)
	I/O:	<0.001 mW	(0%)
Device Static:			(51%)

Figure 4.12: Power report from Xilinx Vivado software

Total Power for both encryption and decryption: 0.166 mW

Average Power for encryption or decryption: 0.083 mW

Average Power for each operation: 0.02075 mW

After performing power analysis considering TSMC65 nm technology, we found that total power consumption is 0.166 mW because this system acts as transceiver i.e, it can transmit the data as well as receive the data. This algorithm consists of total 24 rounds for encryption and 24 rounds for decryption i.e, it has total 48 rounds. If we implement either encryption or decryption on any hardware, then its power consumption will be reduced to half that becomes 0.083 mW. And the average power consumption for each operation is 0.02075 mW since only single operation is active for particular mode and remaining operations are disabled.

CHAPTER 5: FUTURE PLAN OF WORK

So far, we have only designed a symmetric key cryptography algorithm for 16-bits, 32-bits, 64-bits and 128-bits. Also, we have written the Verilog codes and checked the result by observing the RTL analysis on Xilinx Vivado EDA tool for the encryption and decryption processes for 16-bits, 32-bits, 64-bits and 128-bits. We have already completed the design, testing and verification of encryption and decryption process. We have implemented this symmetric key cryptography on a Field Programmable Gate Array (FPGA) board and verified the result by the process of encryption and decryption. In the future, we aim to further improve this algorithm. For example, if the message (plaintext) has a low number of bits, we will follow 16-bits encryption and decryption. For medium size of message, we will use 32-bit encryption and decryption, and for large size of message, we will use 64-bit encryption and decryption and for very large size of message, we will use 128-bit encryption. In the future, we will try to apply this efficient cryptography technique in real world devices such as cameras in ATMs, various IoT sensors, in the field of biomedical, and also in the military field.

CHAPTER 6: CONCLUSIONS

In this project, we have designed an efficient symmetric cryptographic algorithm for 16-bits, 32-bits, 64-bits and 128-bits. In case of 32-bits technique, during the encryption process, it takes 32-bit plaintext (message) and 32-bit key as inputs and generates 32-bit encrypted data known as ciphertext. This ciphertext and the same key is used in the decryption process to get the original message. We have designed this algorithm using Verilog hardware description language and tested this system for different inputs. We have implemented these encryption and decryption techniques on Nexys 4 DDR Artix-7 FPGA board and verified the result successfully for all cases 16-bits, 32-bits, 64-bits and 128-bits data. We have used 4x1 multiplexer so that only one operation is active and remaining operations are stopped. Depending upon the mode (select line), we can select which operation to be active. If mode is 00, then only 16-bit operation is active and remaining operations are turned off. If mode is 01, then only 32-bit operation is active and remaining operations are turned off. Similarly, if mode is 10, then only 64-bits operation works and for mode = 11, only 128-bits operation is active remaining operations are disabled. This is how, our designed module consumes less power and less area. During the hardware implementation, we have used Virtual Input/Output (VIO) to take the inputs and to display the output waveform, we have used the Integrated Logic Analyzer (ILA). After performing area and power analysis considering TSMC65 nm technology, we found that total power consumption is 0.166 mW and total cell area consumption is 0.01357824 mm^2 because this system acts as transceiver i.e, it can transmit the data as well as receive the data. This algorithm consists of total 24 rounds for encryption and 24 rounds for decryption i.e, it has total 48 rounds. If we implement either encryption or decryption on any hardware, then its power consumption will be reduced to half that becomes 0.083 mW and it will take cell area of 0.00678912 mm^2.

REFERENCES

[1] A. Biswas, , "LRBC: a lightweight block cipher design for resource constrained IoT devices," Journal of Ambient Intelligence and Humanized Computing, January, 2020.

[2] Abhishek Guru, Asha Ambhikar, "A Study of Cryptography to Protect Data from Cyber-crimes," National Conference on Resent Trends in Cyber Security At: Raipur, April-June, 2020.

[3] Mohammed Abdulhameed Al-Shabi, "A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security," International Journal of Scientific and Research Publications 9(3):p8779, March, 2019.

[4] Asmita Poojari, Nagesh HR, Kiran Kumar V G, Shantharama Rai C, " Implementation of lightweight cryptographic algorithms in FPGA," 2017 International Conference on Circuits, Controls, and Communications (CCUBE), December, 2017.

[5] G.R.K.Prasad, T.Vivek, B.Phani Rohith, Y.Yashwanth, "Verilog implementation on cryptography encryption and decryption of 8 bit data using ECC algorithm," Journal of Advanced Research in Dynamical and Control Systems 9(14):2711-2719, January, 2017.

[6] Surekha, Sridhar.K, "FPGA Based Cryptography for Internet Security," International Journal of Computer Science and Mobile Computing, October, 2015.

[7] P. T. Thasneem Salim , T. Vigneswaran, "FPGA Implementation of Hiding Information using Cryptography," Indian Journal of Science and Technology, Vol 8(19), DOI: 10.17485/ijst/2015/v8i19/76853, August, 2015.