IMPLEMENTATION OF REVERSIBLE SUBSTITUTION BOX USING LFSR-BASED DYNAMIC KEY IN ADVANCED ENCRYPTION STANDARD ALGORITHM

A PROJECT REPORT

Submitted by

BHAVAN KUMAR C	**190701013**
DEVA S	**190701018**
DHIVYAN K	**190701020**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SRI VENKATESWARA COLLEGE OF ENGINEERING

(An Autonomous Institution; Affiliated to Anna University, Chennai -600 025)

ANNA UNIVERSITY :: CHENNAI 600 025

JUNE 2023

SRI VENKATESWARA COLLEGE OF ENGINEERING

(An Autonomous Institution; Affiliated to Anna University, Chennai -600025)

ANNA UNIVERSITY, CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "**IMPLEMENTATION OF REVERSIBLE SUBSTITUTION BOX USING LFSR-BASED DYNAMIC KEY IN ADVANCED ENCRYPTION STANDARD ALGORITHM"** is the bonafide work of "**BHAVAN KUMAR C (190701013), DEVA S (190701018) and DHIVYAN K(190701020)"** who carried out the project under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S. MUTHUKUMAR	**Ms. B. SARALA**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Department of Electronics and	Assistant Professor
Communication Engineering	Department of Electronics and Communication Engineering

Submitted for the project viva-voce examination held on________________.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Data security has been a major concern as that of the faster processing of data. As the capability of data processing is being evolved, the attacks on these devices for data extraction have also been increasing daily. It is well known that the AES algorithm has been registered as the standard encryption model since 2001. The purpose of this work is to optimise the security of current crypto coprocessors with the help of a Linear Feedback Shift Register (LFSR). This project work offers a simple and innovative method for building dynamic and key dependant S-boxes utilising Multiplicative Inverse and Affine transformation. Here AES with plain text (128-bit) given to the S-box, the 128 bits or 16 bytes is converted into four subcomponents each with 4 bytes. Each byte or 8-bit is given to the multiplicative inverse and affine transform, The output of this above process will be 128-bit cipher text. The 128-bit in the LFSR is Xored with the output to become a dynamic s-box. The same will be continued for N rounds. Followed by this, reversible logic is applied. Reversible logic has numerous methods. Toffoli, one of the methods in reversible logic, is used to maximize speed and reduce energy consumption. The proposed model of generating dynamic and key depended on s-box is the alternative to the existing s-box. As a result, the efficiency of the s-box increases, and encryption becomes stronger.

ACKNOWLEDGMENT

We wish to express our sincere gratitude to **Dr.S.Ganesh Vaidyanathan,** Principal of Sri Venkateswara College of Engineering for providing an opportunity to do project work.

We express our sincere thanks to **Dr.S.Muthukumar**, Professor and Head of the Department, Department of Electronics and Communication, Sri Venkateswara College of Engineering, for encouraging us to work on this project.

We wish to express our heartfelt thanks to our project supervisor **Mrs.B.Sarala,** Assistant Professor, Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, for her immense and timely support given during this project work.

We sincerely thank our project coordinators **Mrs.K.S.Subhashini**, Assistant Professor and **Dr.R.Priyadharshini**, Assistant Professor, Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering for their valuable suggestions and support in this project work.

We would extend our thanks to our entire department faculty members and lab staffs for providing us facilities and helping us to understand various concepts.

BHAVAN KUMAR C

DEVA S

DHIVYAN K

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AES	Advanced Encryption Standard
LFSR	linear feedback shift register
S-BOX	Substitution-box
BytesSub	Substitution Bytes
InvBytesSub	Inverse Substitution Bytes
SPN	Substitution–permutation network
IEEE	Institute of Electrical and Electronics Engineers
RTL	Register Transfer Level
GF	Galois field
ISE	Integrated Synthesis Environment
FPGA	Field Programmable Gate Array
VLSI	Very Large Scale Integration

CHAPTER 1

INTRODUCTION

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm widely used for securing sensitive data. It was selected by the U.S. National Institute of Standards and Technology (NIST) in 2001 as a replacement for the aging Data Encryption Standard (DES). AES is considered secure and efficient, making it a popular choice for protecting data confidentiality in various applications. AES has been extensively analyzed and scrutinized by the cryptographic community, and no practical attacks have been found against its full strength variant. It has been adopted as the encryption standard by governments, organizations, and industries worldwide, providing a robust and trustworthy method for securing sensitive information. AES offers a balance between security and performance. Its design allows for efficient implementation on a wide range of platforms, including hardware and software.

AES encryption and decryption operations can be performed relatively quickly, making it suitable for applications that require high-speed data processing. One of the key strengths of AES is its resistance to various cryptographic attacks, including differential and linear cryptanalysis. The extensive scrutiny and analysis that AES has undergone contribute to its confidence as a secure encryption algorithm.

AES has widespread adoption and is used in a variety of applications, such as secure communications, data storage, virtual private networks (VPNs), secure messaging, and securing sensitive information in databases and financial systems. It is employed in popular encryption protocols and standards, including Transport Layer Security (TLS), secure shell (SSH), and Wi-Fi Protected Access (WPA2). The availability of AES implementations and libraries in different programming languages makes it accessible for developers to incorporate AES encryption into their applications. AES's versatility, security, and efficiency have made it a cornerstone of modern cryptographic systems, contributing to the protection of sensitive information in various domains.

1.1 OVERVIEW OF ADVANCED ENCRYPTION STANDARD

AES operates on fixed-size blocks of data, typically 128 bits, and supports three key sizes: 128, 192, and 256 bits. It uses a substitution-permutation network (SPN) structure, which provides a high level of security through repeated rounds of transformation. The algorithm consists of key expansion, initial round, multiple rounds, and a final round. During each round, AES applies a combination of byte-level substitution, shifting, mixing, and XOR operations to the data block using round keys derived from the original encryption key. These operations provide confusion and diffusion, making it difficult to discern patterns and relationships in the encrypted data.

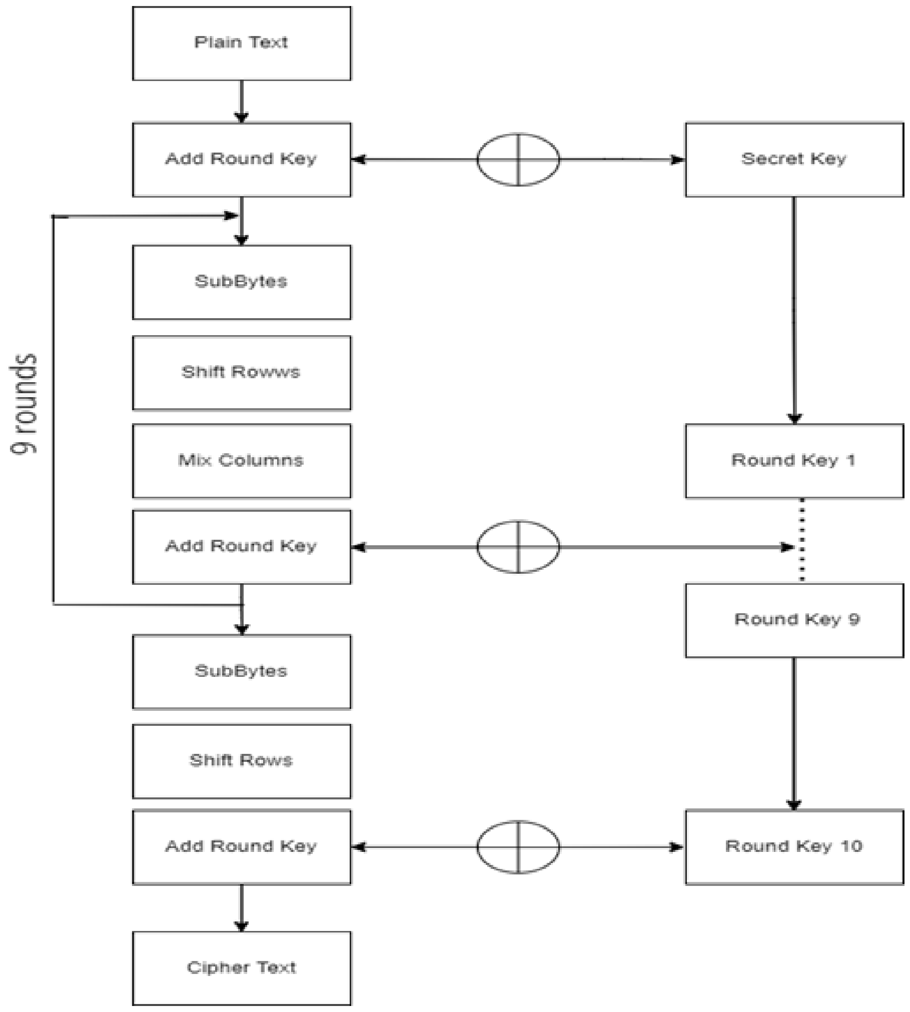


Figure 1.1. AES Block Diagram

AES consists of several key components:

1. Key Expansion: The original key is expanded to create a set of round keys, one for each encryption round.

2. Initial Round: The input block is XORed with the first round key.

3. Rounds: AES uses multiple rounds (10, 12, or 14 depending on the key size) to transform the data. Each round consists of four main operations applied to the data block: Sub Bytes, Shift Rows, Mix Columns and Add Round Key.

4. Sub Bytes: Byte-level substitution using a substitution box (S-box), which replaces each byte with a corresponding value from the S-box.

5. Shift Rows: Shifting the rows of the data block to create diffusion.

6. Mix Columns: Mixing the columns of the data block using matrix multiplication to provide diffusion across columns.

7. Add Round Key: Xoring the data block with the round key.

8. Final Round: Similar to the rounds but without the Mix Columns operation.

9. Output: The final transformed block is the encrypted data.

The S-box's responsibility is to reduce the algorithm's vulnerability to algebraic and differential attacks as well as to linear and differential cryptanalysis techniques. The S-box function must be invertible, have no fixed points, and meet the complexity condition. It must also execute quickly and be simple to implement. Examples of this include complimentary fixed points S(a)=a. All 256 8-bit values that could be used are permuted in S-box. The following is the mapping of each unique state byte into a new byte: The byte's four leftmost bits are used as a row value and its four rightmost bits as a column value. These column and row values act as indices in the S-box to choose a specific 8-bit output value. By applying the multiplicative inverse to the plain text in GF (28) and then applying an affine transformation to it, the ByteSub & InvByteSub transformation is calculated.

1.2 ORGANIZATION OF THE PROJECT

The purpose of this introduction chapter is to provide a general overview on the Advanced Encryption Standard. Literature survey has been discussed in Chapter 2. Realization of Substitution box using Rijndael algorithm is given in the Chapter 3. In Chapter 4, proposed design of Substitution box using reversible logic gates embedded with LFSR are discussed. In Chapter 5, RTL Schematics of isomorphic and inverse isomorphic mapping, affine transformation and multiplicative inverse are discussed. In Chapter 6, Simulations of the above mentioned module are discussed. In Chapter 7, Area, Power and Timing reports of various modules S-box and the comparison has been discussed. In the Chapter 8, the conclusion and future work of the project is given.

1.3 OBJECTIVE OF THE PROJECT

- To create a static S-box using Rijndael hardware algorithm.
- To improve the security of the S-box by converting the static S-box to a dynamic one, using Linear Feedback Shift Register.
- To reduce the power dissipation of the hardware by using reversible logic gates.
- To compare and analyze the existing and proposed S-box.

CHAPTER 2
LITERATURE SURVEY

The various data collected from the below literature survey is useful in various ways in developing this project. Literature survey helps in analyzing the design of S-Box using reversible logic gates and design of dynamic S-Box using 4-bit LFSR.

Aaron Barrera, Chu-wen Cheng and Dr. Sanjeev Kumar(International Conference on Data Intelligence and Security (ICDIS), 2020) proposed about the fast implementation of Rijndael substitution box for cryptographic AES. In this, we investigate various implementations for improving the hardware performance of the Rijndael S-box component of the AES in IEEE format algorithm in terms of delay and number of logic elements.[1]

Bahram Rashidi and Bahman Rashidi(International Research Journal on Computer Network and Information Security, January 2013) proposed about the implementation of an optimized and pipelined combinational logic Rijndael S-box on FPGA. Here, The complete data path of the S-box algorithm is simulated as a net list of AND, OR, NOT and XOR logic gates, also for increase in speed and maximum operation frequency used 4-stage pipeline in proposed method.[4]

Donghui Lee, Yongtae Kim (IEEE International Conference on Consumer Electronics, November 2021) presented a new light weight schedulerin AES using linear feedback shift register. Here, it improves the overall hardware efficiency, the proposed key scheduler exploits an LFSR based structure with XOR operation.[7]

D Srinivas, Boda Aruna, Ravi Boda(International Journal of latest Engineering Research and Application(IJLERA) Volume : 02 Issue:12 | December - 2017) proposed the implementation of Advanced encryption Standard using Reversible Logic Gate.[8]

Jia Jun Tay, M L Dennis Wong, Ming Ming Wong, Cishen Zhang, Ismat Hijazin (IEEE region 10 Conference | October 2018) proposed a compact composite AES S-box by deriving a new low multiplicative complexity GF (2^4) inversion circuit.[9]

Mummadi Swathi, Dr. Bhawana Rudra (IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) | February 2021) proposed the Implementation of Reversible Logic Gates with Quantum Gates in IEEE format. Here the complexity of the digital circuits is reduced by using reversible computing.[11]

Saravanan P, Kalpana P (Wireless Pers Communication Volume:07 |March 2018) proposed the novel reversible design of 128-bit Advanced Encryption Standard(AES) cryptographic algorithm for wireless sensor networks.[12]

Sushma D K , Manju Devi (International Journal of Engineering Research and Technology (IJERT) ,Volume:06, issue:13| 2018) proposed the design of S-box and INV S-box using Composite Field Arithmetic for AES algorithm. The proposed architecture which is combined implementing S-box and Inv s-box makes use of an enable pin to perform encryption and decryption in AES.[13]

The above way of predictive determining of proposals helped in innovatingkey results, areas of improvement & innovation and setting up benchmark standard of the project in various aspects of technology domain.

CHAPTER 3

REALIZATION OF S-BOX

3.1 S-BOX

An S-box (Substitution Box) is a fundamental component in symmetric key cryptography, used in various encryption algorithms such as the AES. It performs substitution operations on blocks of data during encryption or decryption.

The purpose of an S-box is to provide confusion and non-linearity, making the encryption process resistant to cryptanalysis attacks. It achieves this by replacing input bit patterns with corresponding output bit patterns based on a predefined substitution table.

Here are the key characteristics and properties of an S-box:

- Size: An S-box typically operates on fixed-size input blocks. For example, in AES, the S-box operates on 8-bit input values.

- Substitution Table: An S-box consists of a substitution table that maps input values to output values. The substitution table is typically designed to exhibit desirable cryptographic properties such as non-linearity, diffusion, and resistance to various attacks.

- Non-Linear Mapping: The substitution table in an S-box ensures a non-linear mapping between the input and output values. This non-linearity helps to prevent linear and differential cryptanalysis attacks.

- Confusion: The S-box adds confusion to the encryption process by making the relationship between the input and output values complex and non-trivial. This property helps to hide the statistical properties of the plaintext and increase the overall security of the algorithm.

- Fixed and Secret: The substitution table used in an S-box is fixed and predefined. It is carefully designed to resist known attacks and is kept secret to prevent attackers from exploiting any weaknesses.

- During the encryption or decryption process, each input block is substituted with the corresponding output block using the S-box. This substitution step adds a significant level of complexity and non-linearity to the cryptographic algorithm, contributing to its security.

The design and construction of S-boxes are critical aspects of cryptographic algorithm design, and extensive analysis and research go into creating S-boxes with strong security properties.

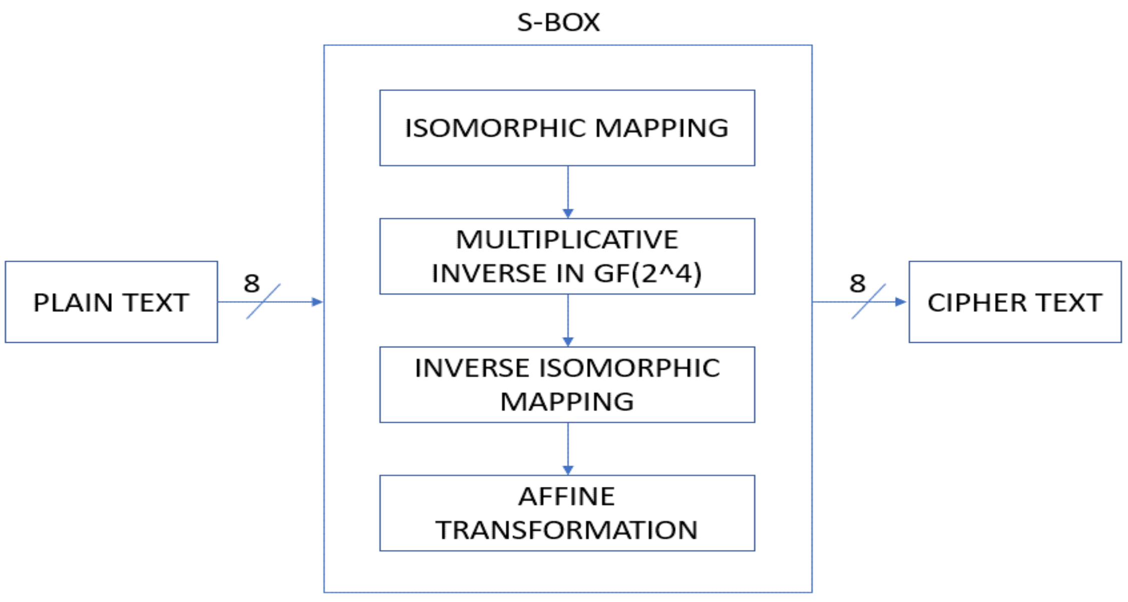


Figure 3.1 Existing S-Box Architecture

3.2 MULTIPLICATIVE INVERSE

The multiplicative inverse in the context of an S-box refers to finding the inverse element of a given input value with respect to multiplication within a finite field. In cryptographic algorithms, such as AES, S-boxes often operate on elements in a finite field to provide non-linearity and confusion.

To find the multiplicative inverse of an element in an S-box, the following steps can be taken:

- Represent the element as a binary vector. For example, if the S-box operates on 8-bit input values, convert the element to an 8-bit binary representation.

- Determine the irreducible polynomial that defines the finite field in which the S-box operates. This polynomial is typically chosen based on cryptographic properties and design considerations.

- Use a method such as the Extended Euclidean Algorithm to calculate the multiplicative inverse. This algorithm finds the coefficients of Bezout's identity, which represents the multiplicative inverse of the element.

- Perform modular reduction using the irreducible polynomial. The modular reduction step ensures that the resulting inverse element remains within the finite field.

- Convert the binary representation of the inverse element back to its original format, if necessary.

It's important to note that the existence of a multiplicative inverse depends on the specific element and the properties of the finite field. In some cases, an element may not have a multiplicative inverse, which can affect the security and functionality of the S-box.

The multiplicative inverse operation in an S-box is significant in cryptographic algorithms as it enables the process of decryption. During encryption, the S-box is applied to the input values, and during decryption, the inverse S-box is applied to the cipher text to retrieve the original plaintext. The calculation of the multiplicative inverse ensures the correct reversal of the encryption process.

3.2.1 Isomorphic Mapping

In the context of cryptography, an S-box is a component commonly used in symmetric key algorithms, such as the Advanced Encryption Standard. It performs substitution operations on input bit patterns to provide confusion and non-linearity in the cryptographic algorithm.

Isomorphic mapping, on the other hand, refers to a mapping or transformation that preserves certain properties of the original structure. In the case of S-boxes, isomorphic mappings are used to create equivalent S-boxes that maintain certain cryptographic properties while changing the specific values or arrangements of elements within the S-box.

The goal of an isomorphic mapping in the context of S-boxes is to create an equivalent S-box with different values, while preserving desirable properties like resistance against various cryptographic attacks. By applying an isomorphic mapping, the specific values of the S-box elements are changed, but the overall structure and behaviour of the S-box are maintained.

Isomorphic mappings can be used to achieve different objectives, such as enhancing security or optimizing hardware implementations. However, it's important to note that modifying S-boxes can have significant implications for the security of cryptographic algorithms. Any changes to the S-box must undergo rigorous analysis and testing to ensure they do not introduce vulnerabilities or weaken the security of the algorithm.

The below-mentioned are the equations for isomorphic mapping:

$O_7 = q_7 \oplus q_5$ ……………………………….(3.1)

$O_6 = q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1$ …….(3.2)

$O_5 = q_7 \oplus q_5 \oplus q_3 \oplus q_2$ …………………..(3.3)

$O_4 = q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1$ ……………(3.4)

$O_3 = q_7 \oplus q_6 \oplus q_2 \oplus q_1$ …………………...(3.5)

$O_2 = q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1$ ……………(3.6)

$O_1 = q_6 \oplus q_4 \oplus q_1$ …………………………..(3.7)

$O_0 = q_6 \oplus q_1 \oplus q_0$ …………………………..(3.8)

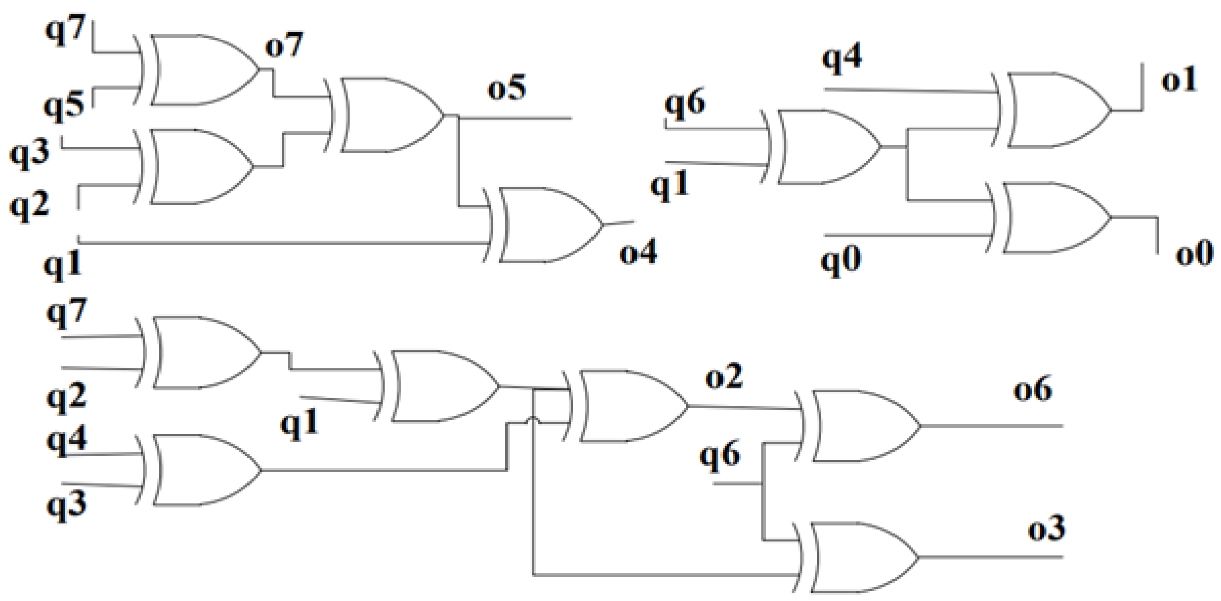


Figure 3.2 Isomorphic mapping logic diagram

3.2.2 Inverse Isomorphic Mapping

Inverse isomorphic mapping, in the context of S-boxes, refers to the process of reversing the changes applied through an isomorphic mapping to retrieve the original S-box. It involves transforming an S-box with modified values or arrangements back to its original form while preserving the desired properties.

The purpose of an inverse isomorphic mapping can vary depending on the specific scenario. For example, it might be necessary to revert changes made to an S-box during analysis or testing, or to reconstruct the original S-box after applying optimizations or transformations.

Performing an inverse isomorphic mapping typically requires knowledge of the original mapping function or transformation used to modify the S-box. By applying the inverse of that mapping function, the modified S-box can be transformed back to its original representation.

It's important to note that inverse isomorphic mapping is only feasible if the original mapping function is known and reversible. If the mapping function is irreversible or its details are not available, it might not be possible to reconstruct the original S-box accurately.

In cryptographic applications, it is essential to maintain the security and integrity of the S-box. Therefore, any modifications, including isomorphic mapping and its inverse, should be thoroughly analyzed, tested, and validated to ensure that they do not introduce vulnerabilities or compromise the security of the cryptographic algorithm.

The below-mentioned are the equations for inverse isomorphic mapping:

$O_7 = q_7 \oplus q_6 \oplus q_5 \oplus q_1$ ……………(3.9)

$O_6 = q_6 \oplus q_2$ ………………………..(3.10)

$O_5 = q_6 \oplus q_5 \oplus q_1$ ………………...(3.11)

$O_4 = q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1$ …..(3.12)

$O_3 = q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1$ …..(3.13)

$O_2 = q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1$ …..(3.14)

$O_1 = q_5 \oplus q_4$ ………………………..(3.15)

$O_0 = q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0$ …..(3.16)

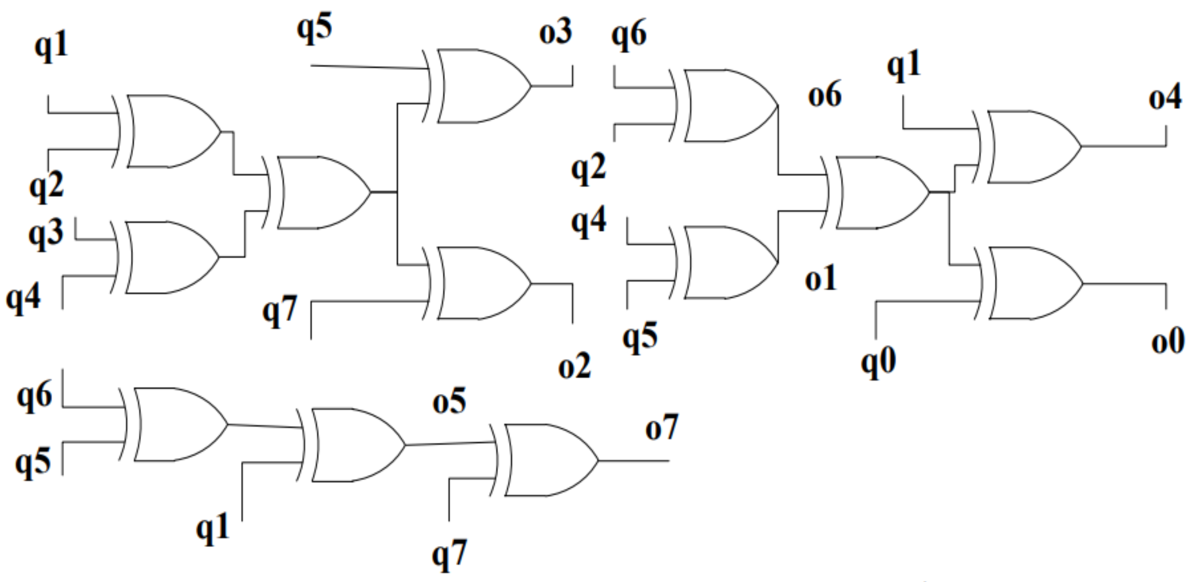


Figure 3.3 Inverse Isomorphic mapping logic diagram

3.2.3 Squarer

The term "squarer" in the context of an S-box refers to the operation of finding the square of an input element within a finite field. In cryptographic algorithms, S-boxes often perform various operations to provide non-linearity and confusion, including squaring.

To compute the square of an element in an S-box, the following steps can be followed:

- Represent the element as a binary vector. For example, if the S-box operates on 8-bit input values, convert the element to an 8-bit binary representation.
- Determine the irreducible polynomial that defines the finite field in which the S-box operates. This polynomial is typically chosen based on cryptographic properties and design considerations.

- Perform polynomial multiplication of the element with itself. This multiplication can be achieved using standard polynomial multiplication techniques, where the binary representation of the element is treated as a polynomial.
- Perform modular reduction using the irreducible polynomial. The modular reduction step ensures that the resulting squared element remains within the finite field.
- Convert the binary representation of the squared element back to its original format, if necessary.
- The squaring operation in an S-box introduces additional non-linearity and confusion, as it involves multiplying an element by itself. This operation enhances the security properties of the S-box and the cryptographic algorithm as a whole.
- It's important to note that the specific implementation of the squaring operation within an S-box can vary depending on the cryptographic algorithm in use. The squaring operation is often tailored to meet the desired cryptographic properties and security requirements of the algorithm.

The below-mentioned are the equations for squarer:

$O_3 = i_3$ ………………(3.17)

$O_2 = i_3 \oplus i_2$ ………..(3.18)

$O_1 = i_2 \oplus i_l$ ………...(3.19)

$O_0 = i_3 \oplus i_1 \oplus i_0$ ….(3.20)

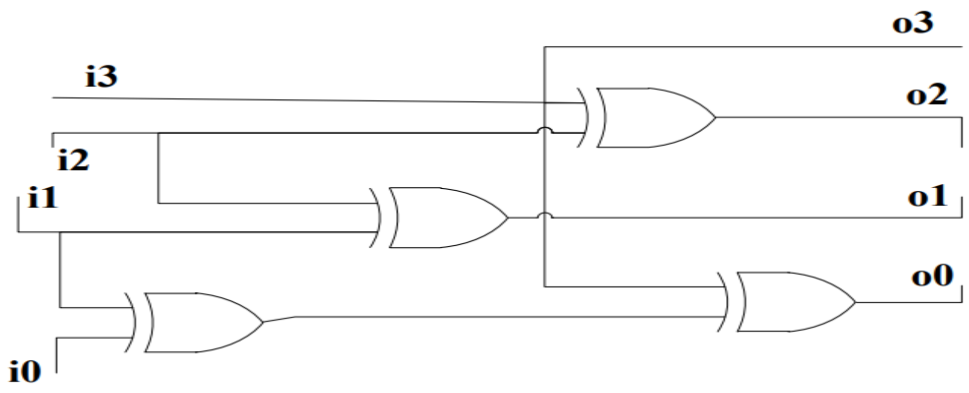


Figure 3.4 Squarer logic diagram

3.2.4 Multiplication in GF(2^4)

Multiplication in GF(2^4), which stands for the Galois Field with 2^4elements, within an S-box involves performing multiplication operations on elements represented as binary vectors with four binary inputs and four binary outputs. In cryptographic algorithms like AES, S-boxes often operate on elements in GF(2^4) to introduce non-linearity and confusion.

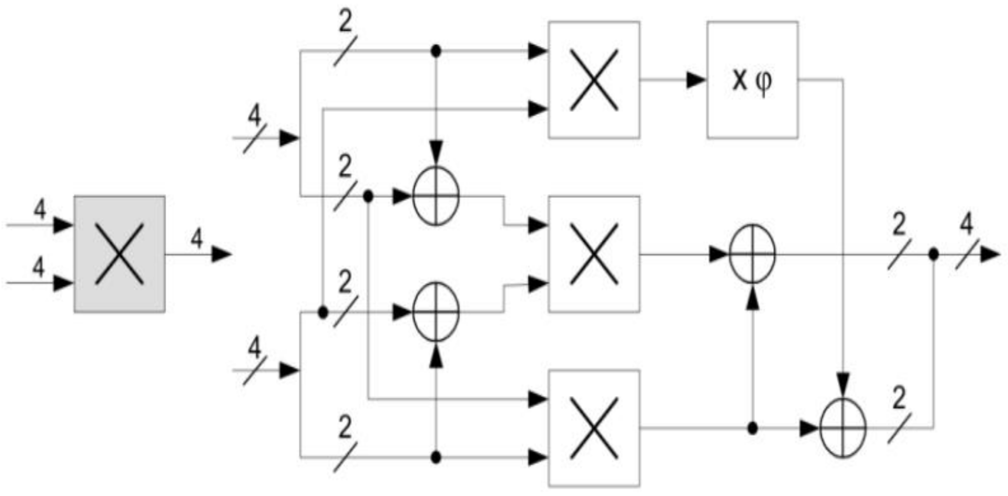


Figure 3.5 Multiplication in GF(2^4) Block diagram

3.2.5 Multiplication in GF(2^2)

In GF(2^2), addition and subtraction are performed using the bitwise XOR operation. Additionally, the value 0 is treated as the additive identity (0 + A = A), and the value 1 is treated as the multiplicative identity (1 * A = A) in the field.

Performing multiplication in GF(2^2) within an S-box involves working with finite field arithmetic, including modular reduction, to maintain the 2-bit representation. These operations contribute to the non-linearity and confusion properties of the S-box, enhancing the security of the cryptographic algorithm.

The below-mentioned are the equations for multiplication in GF(2^2):

$k_1 = q_1 w_1 \oplus q_0 w_1 \oplus q_1 w_0$ ....(3.21)

$k_0 = q_1 w_1 \oplus q_0 w_0$ ....(3.22)

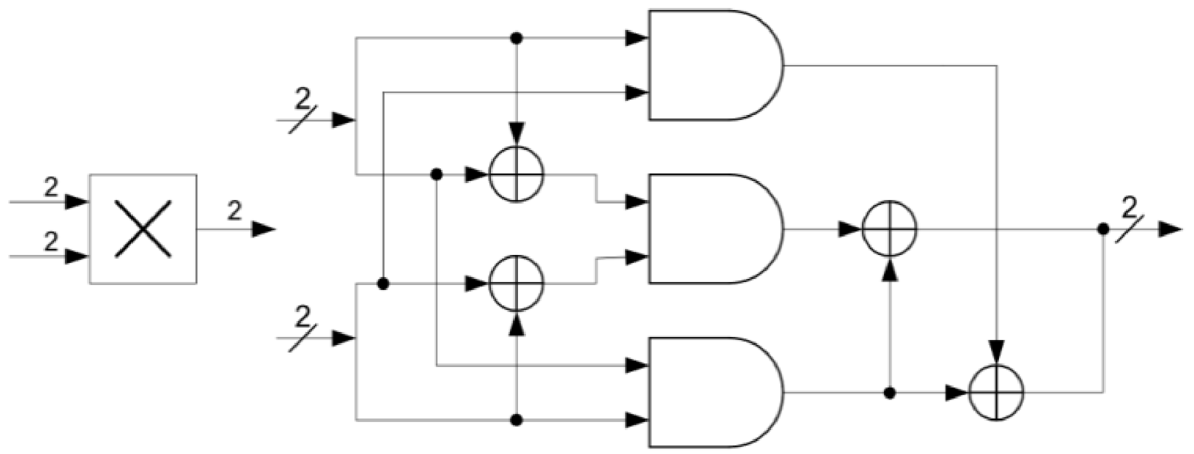


Figure 3.6 Multiplication in GF(2^2) logic diagram

3.2.6 Multiplication with constant X(ϕ)

The multiplication with constant X(ϕ) is one of the module used in the multiplication in GF(2^4). In multiplication with constant X(ϕ) there are two binary inputs and two binary outputs.

The below-mentioned are the equations for multiplication with constant X(ϕ):

$k_1 = q_1 \oplus q_0$ ……..(3.23)

$k_0 = q_1$ …………….(3.24)

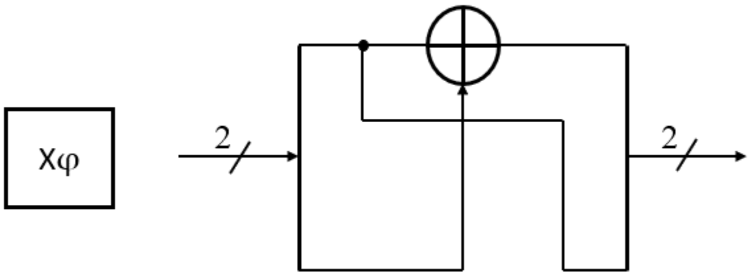


Figure 3.7 Multiplication with constant X(ϕ) Block diagram

3.2.7 Multiplication with constant X(λ)

The multiplication with constant X(λ) is one of the module used in the multiplicative inverse GF(2^8). In multiplication with constant X(λ) there are four binary inputs and four binary outputs.

The below-mentioned are the equations for multiplication with constant X(λ):

$O_3 = i_2 \oplus i_0$ ……………………..(3.25)

$O_2 = i_3 \oplus i_2 \oplus i_1 \oplus i_0$ ……….(3.26)

$O_1 = i_3$ ………………………….(3.27)

$O_0 = i_2$ ………………………….(3.28)

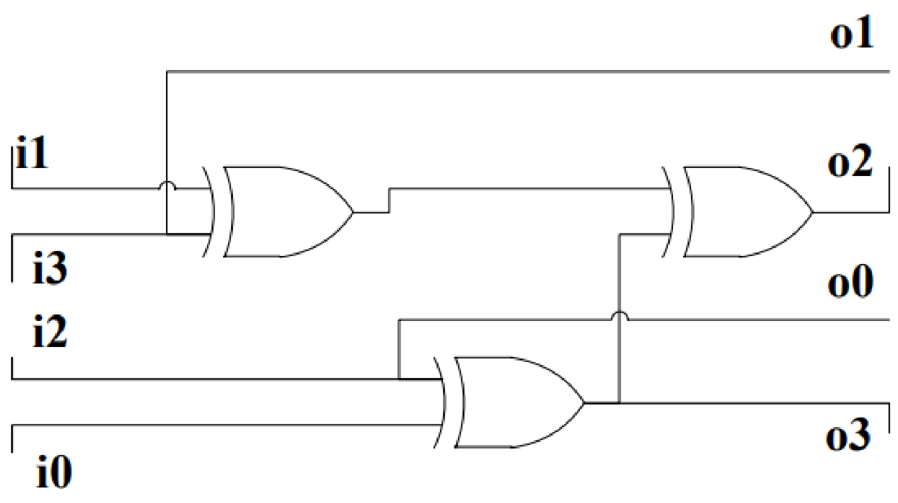


Figure 3.8 Multiplication with constant X(λ) logic diagram

3.2.8 Multiplication inversion in GF(2^4)

The multiplication inversion is one of the module used in the multiplicative inverse GF(2^8). In multiplication inversion in GF(2^4) there are four binary inputs and four binary outputs.

The below-mentioned are the equations for multiplication inversion in GF(2^4):

$$q_3^{-1}=q_3 \oplus q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_2 \quad \text{......................................(3.29)}$$

$$q_2^{-1}=q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_0 \oplus q_2 \oplus q_1 q_2 \quad \text{..............................(3.30)}$$

$$q_1^{-1}=q_3 q_2 q_1 \oplus q_3 q_1 q_0 \oplus q_2 q_0 \oplus q_2 \oplus q_1 \oplus q_3 \quad \text{...........................(3.31)}$$

$$q_0^{-1}=q_3 q_2 q_1 \oplus q_3 q_2 q_0 \oplus q_3 q_1 \oplus q_3 q_1 q_0 \oplus q_3 q_0 \oplus q_2 q_1 q_0 \oplus q_2 q_1 \oplus q_2 \oplus q_1 \oplus q_0 \quad (3.32)$$

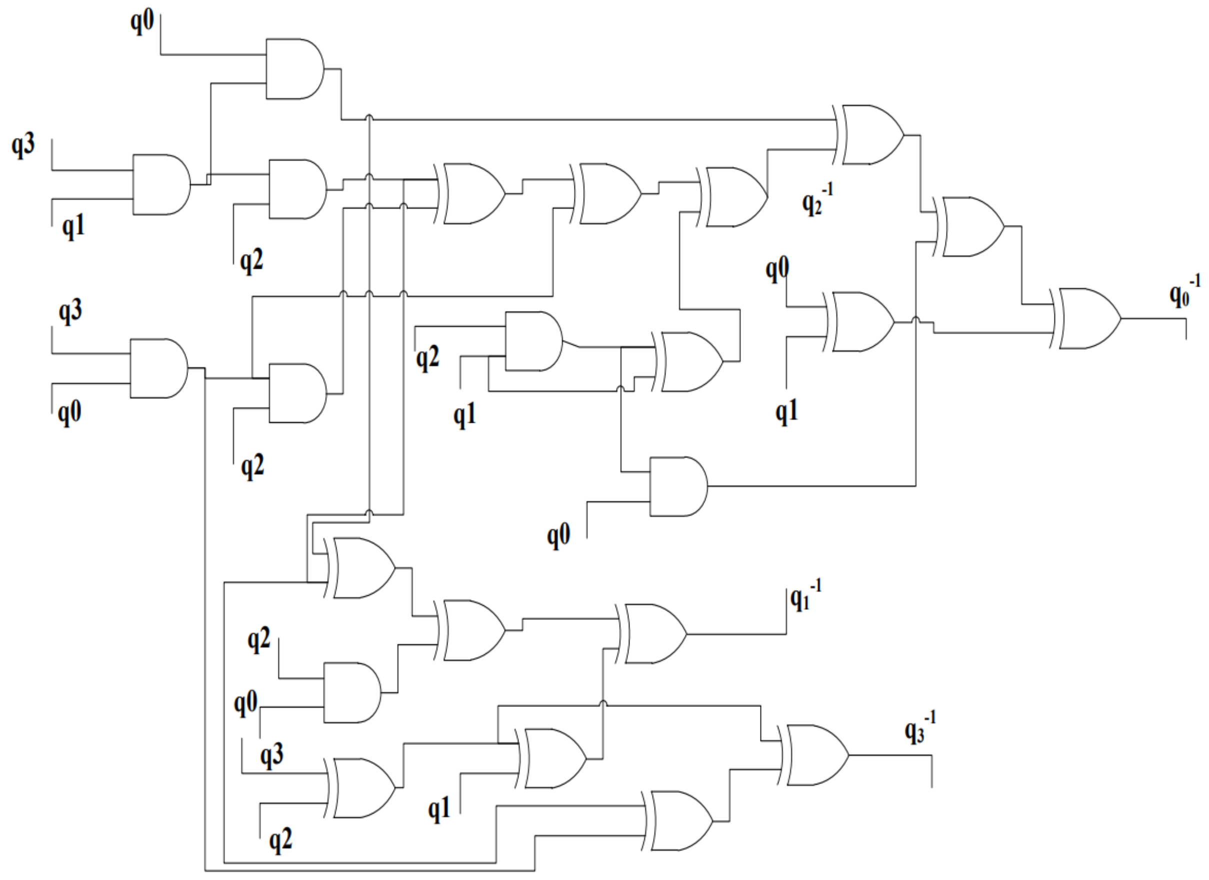


Figure 3.9 Multiplication inversion in GF(2^4) logic diagram

3.3 AFFINE TRANSFORM

An affine transform is a linear transformation followed by a translation. In the context of cryptography, an affine transform is often applied to elements in a Substitution Box (S-box) to increase their resistance against cryptographic attacks.

An S-box is a crucial component in symmetric key algorithms, such as the Advanced Encryption Standard. It is responsible for performing substitution operations on blocks of data during encryption or decryption. The S-box operates on fixed-size input blocks and replaces them with corresponding output blocks according to a predefined substitution table.

To incorporate an affine transform into an S-box, each element in the S-box is first represented as a binary vector. Let's consider an example where the S-box takes 4-bit input values and produces 4-bit output values. The binary representation of an input value, x, is given as x = x3x2x1x0, where x3 is the most significant bit and x0 is the least significant bit.

The affine transform is typically defined by the following equation:

y = (Ax + b) mod 2 ……………………(3.33)

where y is the transformed output, x is the input, A is a fixed binary matrix, b is a fixed binary vector, and mod 2 indicates the operation is performed modulo 2 (i.e., bitwise XOR).

The binary matrix A and vector b are selected to achieve desirable cryptographic properties, such as confusion and diffusion. They are typically chosen based on cryptographic analysis and are kept secret to enhance the security of the algorithm.

To apply the affine transform to an S-box, the following steps are performed:

- Convert the input value x into a binary vector x3x2x1x0.
- Apply the matrix multiplication Ax.
- Perform a bitwise XOR between the resulting vector and the vector b.
- The resulting vector is the transformed output y.
- Convert the binary vector y into a decimal value to obtain the output from the S-box.
- By incorporating an affine transform, the S-box exhibits improved resistance against attacks such as linear and differential cryptanalysis. The specific choice of the matrix A and vector b plays a crucial role in achieving this enhanced security.

The below-mentioned are the equations for Affine Transform:

$O_7 = a_3 \oplus a_4 \oplus a_5 \oplus a_6 \oplus a_7$ …………………..(3.34)

$O_6 = \sim a_2 \oplus\sim a_3 \oplus\sim a_4 \oplus\sim a_5 \oplus \sim a_1$ ………..(3.35)

$O_5 = \sim a_1 \oplus\sim a_2 \oplus\sim a_3 \oplus\sim a_4 \oplus \sim a_5$ ………..(3.36)

$O_4 = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_4$ …………………..(3.37)

$O_3 = a_0 \oplus a_1 \oplus a_2 \oplus a_3 \oplus a_7$ …………………..(3.38)

$O_2 = a_0 \oplus a_1 \oplus a_2 \oplus a_6 \oplus a_7$ …………………..(3.39)

$O_1 = \sim a_0 \oplus\sim a_1 \oplus\sim a_5 \oplus\sim a_6 \oplus \sim a_7$ ………..(3.40)

$O_0 = \sim a_0 \oplus\sim a_4 \oplus\sim a_5 \oplus\sim a_6 \oplus \sim a_7$ ………..(3.41)

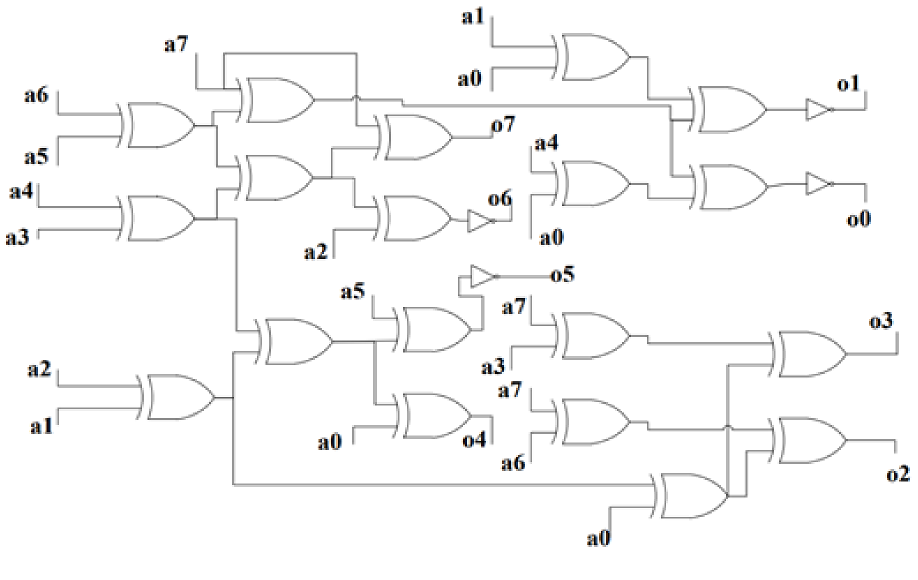


Figure 3.10 Affine transform logic diagram

CHAPTER 4

DYNAMIC S-BOX USING LFSR WITH REVERSIBLE LOGIC GATES

4.1 REVERSIBLE LOGIC GATES

Reversible logic gates, also known as reversible gates or quantum gates, are fundamental building blocks in reversible computing and quantum computing. Unlike classical logic gates, which are irreversible and result in information loss due to the destruction of input information, reversible logic gates preserve information by maintaining a one-to-one mapping between inputs and outputs.

In reversible logic, every output bit is uniquely determined by the combination of input bits, and vice versa. This reversibility property allows for the reconstruction of input information from the output, making reversible circuits ideal for applications where energy efficiency and information preservation are critical.

Reversible logic gates adhere to two important principles:

1. Reversibility: A reversible gate ensures that every unique input pattern maps to a unique output pattern, and every unique output pattern has a unique input pattern. This reversibility property is essential for information conservation.

2. Consistency: The number of input bits must be equal to the number of output bits for each reversible gate. This ensures a one-to-one mapping between inputs and outputs.

Several common reversible logic gates are used to construct reversible circuits. Let's discuss a few key reversible gates:

1. Toffoli Gate (also known as Controlled-Controlled-Not or CCNOT):

- The Toffoli gate has three inputs and three outputs.
- It performs a NOT (bit-flip) operation on the third output (target) if both the first and second inputs (controls) are set to 1. Otherwise, it leaves the target unchanged.

- The Toffoli gate is universal for classical reversible computing, meaning any reversible computation can be implemented using Toffoli gates alone.

2. Fredkin Gate (also known as Controlled-Swap or CSWAP):

- The Fredkin gate has three inputs and three outputs.

- It swaps the second and third outputs if the first input is set to 1. Otherwise, it leaves the outputs unchanged.

- The Fredkin gate is another universal reversible gate, and it can be used to implement any reversible computation.

3. Feynman Gate (also known as Controlled-Phase or CPHASE):

- The Feynman gate has two inputs and two outputs.

- It applies a phase shift of 180 degrees (π radians) to the second output if the first input is set to 1. Otherwise, it leaves the outputs unchanged.

- The Feynman gate is often used in quantum computing for creating entanglement and implementing quantum algorithms.

4. Peres Gate (also known as Controlled-Phase-Flip or CPHASEFLIP):

- The Peres gate has two inputs and two outputs.

- It performs a phase flip (bit-flip) operation on the second output if both inputs are set to 1. Otherwise, it leaves the outputs unchanged.

- The Peres gate is useful in reversible computing for creating reversible circuits that exhibit parity preservation.

These are just a few examples of reversible logic gates, and there are many other gates available with specific functionalities. Reversible logic gates play a crucial role in the design and implementation of reversible circuits, enabling information-preserving computations and providing the foundation for energy-efficient computing paradigms such as quantum computing and reversible computing.

5. NOT Gate:

- The NOT gate, also called the bit-flip gate, is a basic reversible gate.
- It has a single input and a single output.
- The NOT gate simply flips the input bit, changing a 0 to 1 or vice versa.

6. Controlled-Pauli Gates:

- Controlled-Pauli gates are reversible gates derived from Pauli gates used in quantum computing.
- They include Controlled-X (CNOT), Controlled-Y, and Controlled-Z gates.
- These gates have two inputs and two outputs.
- The Controlled-X gate applies a Pauli-X (bit-flip) operation to the second output if the first input is set to 1.

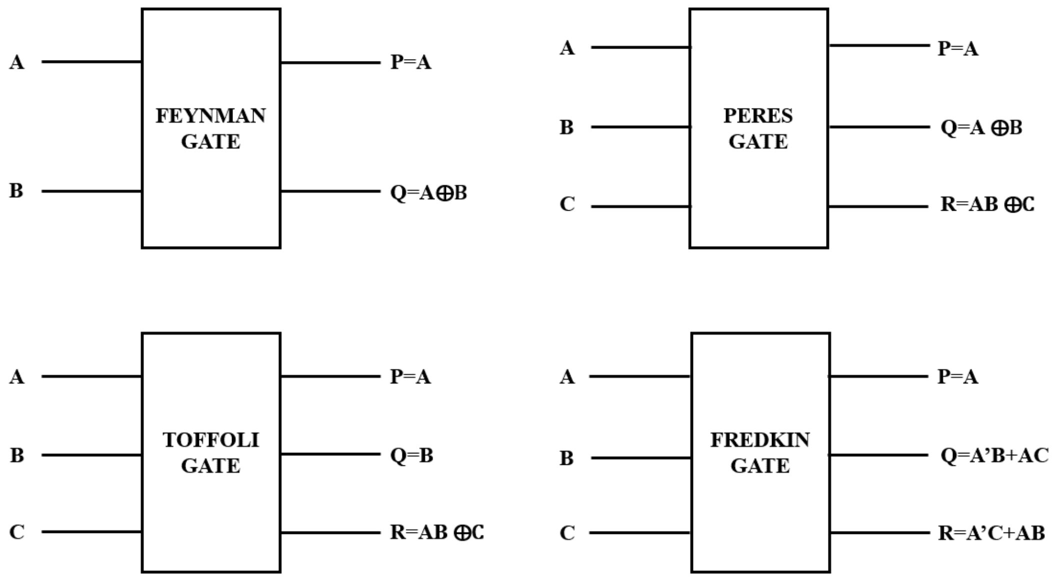


Figure 4.1 Reversible Logic Gates

4.1.1. Isomorphic Mapping

An 8-bit Isomorphic mapping logic diagram using feynman reversible logic gates is shown below.

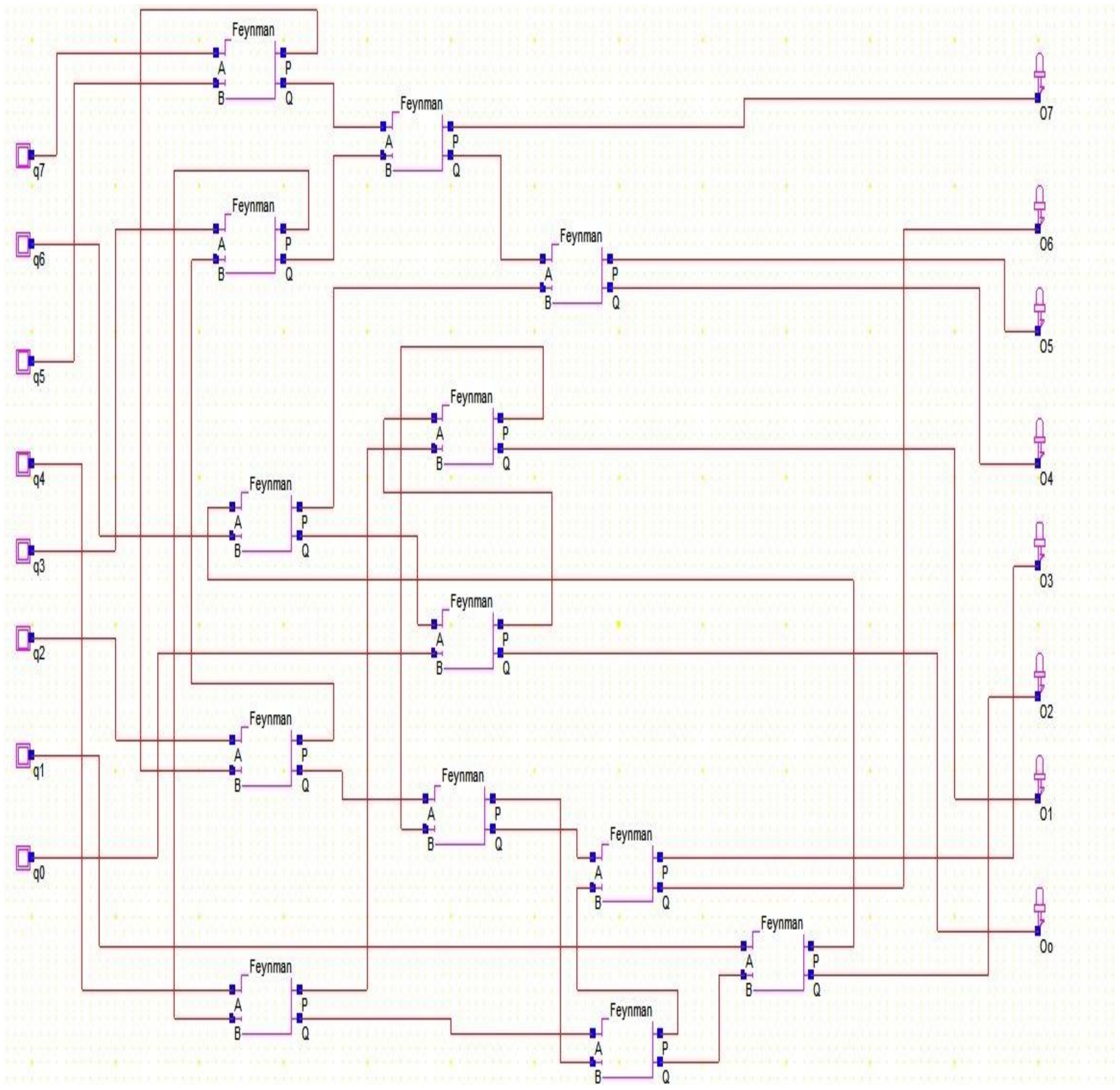


Figure 4.2 Isomorphic Mapping

4.1.2. Inverse Isomorphic Mapping

An 8-bit Inverse Isomorphic mapping logic diagram using feynman reversible logic gates is shown below.

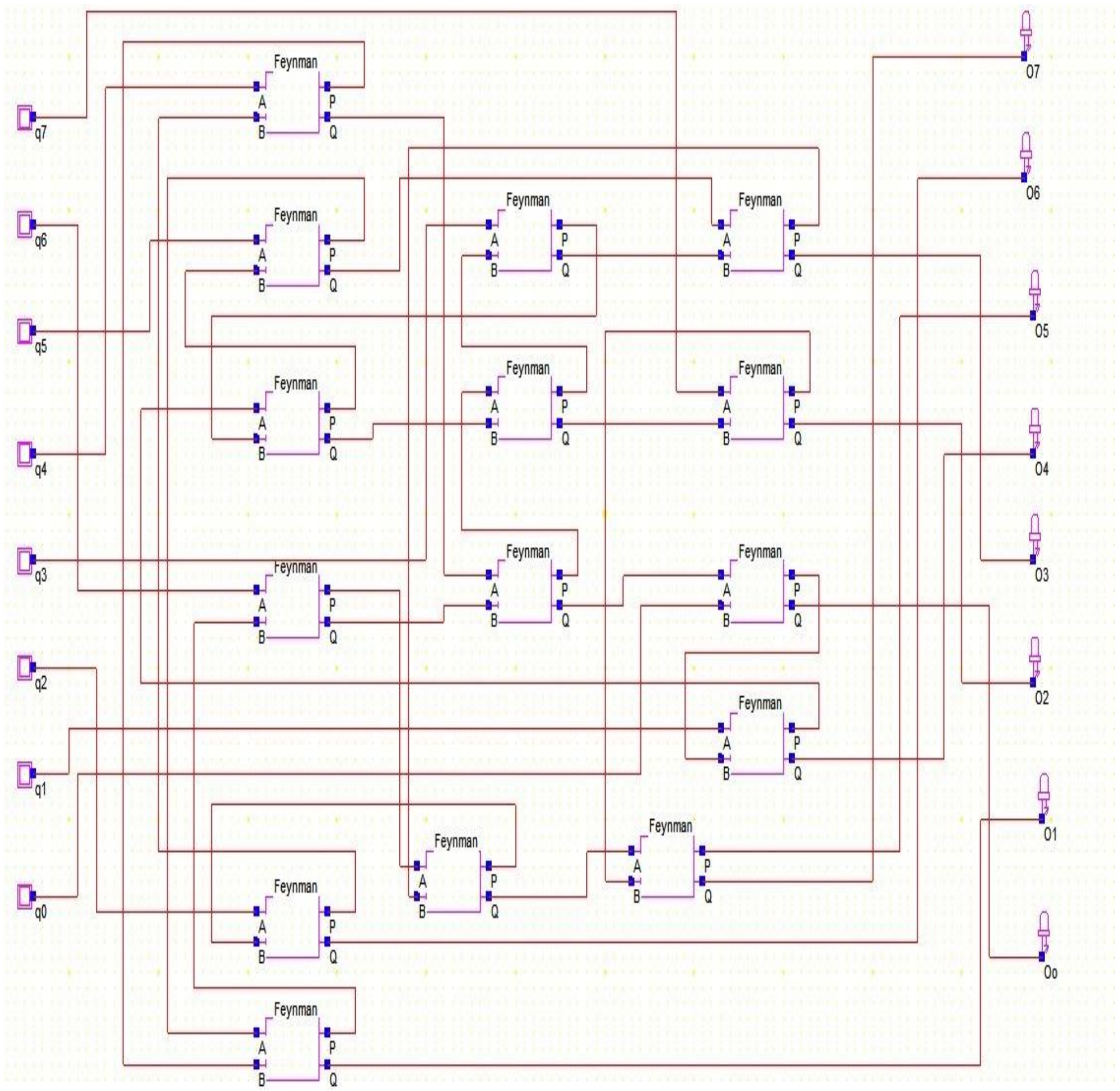


Figure 4.3 Inverse Isomorphic Mapping

4.1.3. Squarer

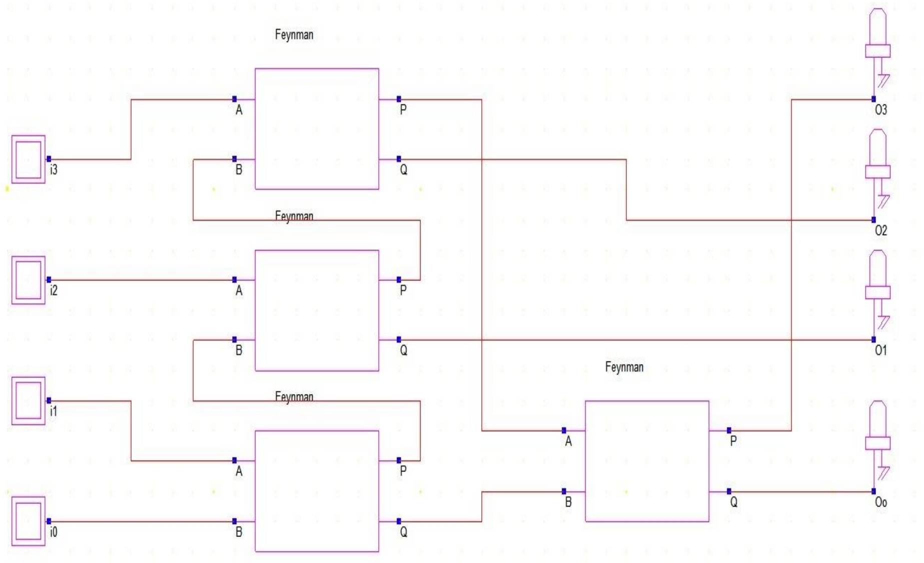


Figure 4.4 Squarer

An 4-bit Squarer logic diagram using feynman reversible logic gates is shown above.

4.1.4. Multiplication Operation in GF(2^4)

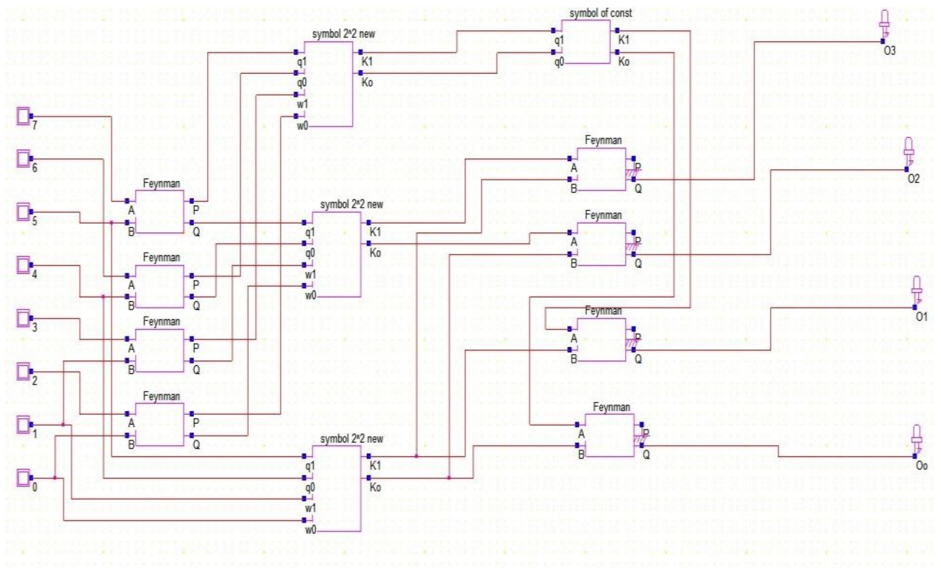


Figure 4.5 Multiplication Operation in GF(2^4)

An 4-bit Multiplication Operation in GF(2^4) logic diagram using feynman and toffoli reversible logic gates is shown above.

4.1.5. Multiplication Operation in GF(2^2)

An 2-bit Multiplication Operation in GF(2^2) logic diagram using feynman and toffoli reversible logic gates is shown below.

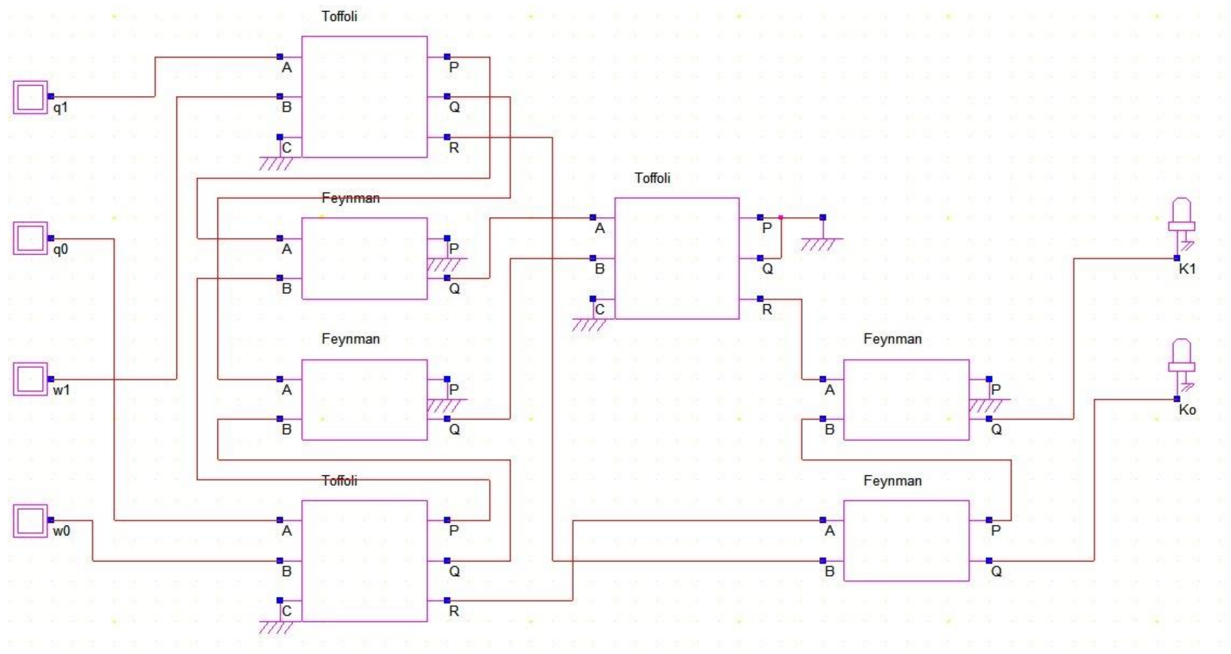


Figure 4.6 Multiplication Operation in GF(2^2)

4.1.6. Multiplication with constant (Xφ)

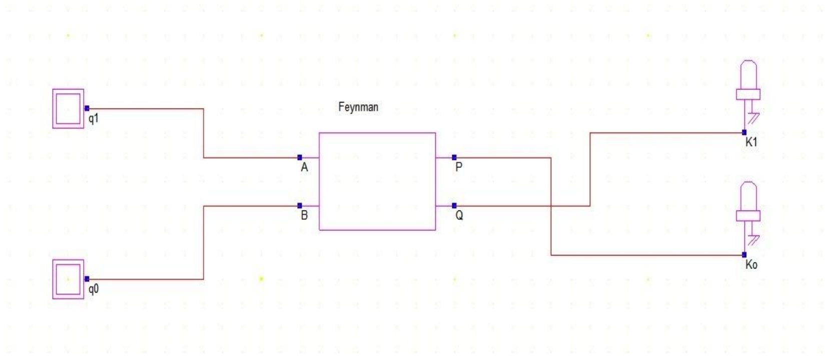


Figure 4.7 Multiplication with constant (Xφ)

A Multiplication with constant logic X(**φ**) diagram using feynman reversible logic gates is shown above.

4.1.7. Multiplication with constant (Xλ)

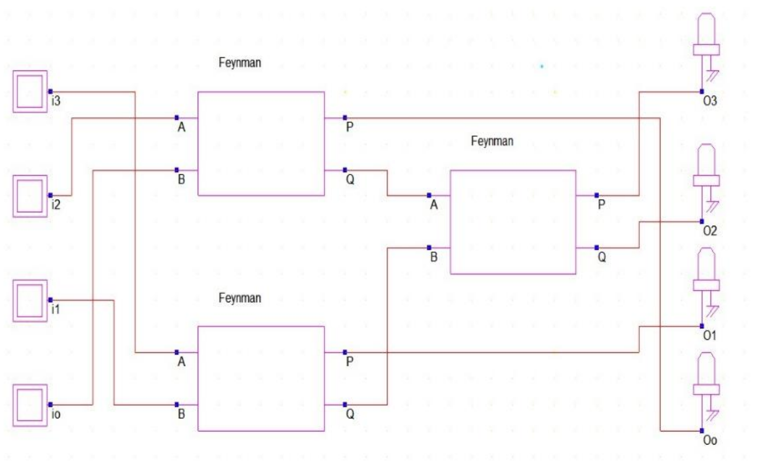


Figure 4.8 Multiplication with constant (Xλ)

A Multiplication with constant X(**λ**) logic diagram using feynman reversible logic gates is shown above.

4.1.8. Multiplication Inversion in GF(2^4)

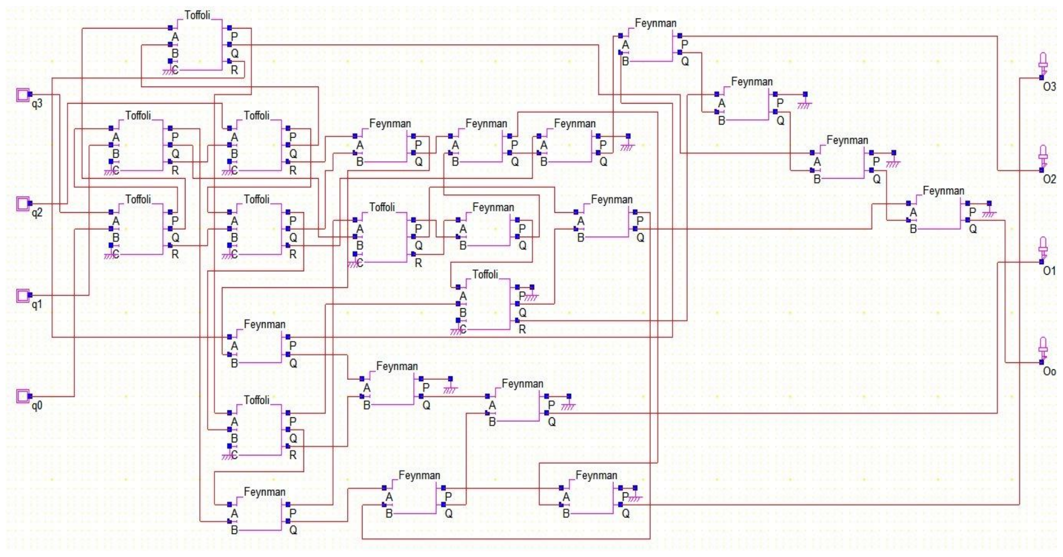


Figure 4.9 Multiplication Inverse in GF(2^4)

An 4-bit Multiplication Inversion in GF(2^4) logic diagram using feynman and toffoli reversible logic gates is shown above.

4.1.9. Affine Transformation

An 8-bit Affine transformation logic diagram using feynman and not reversible logic gates is shown below.

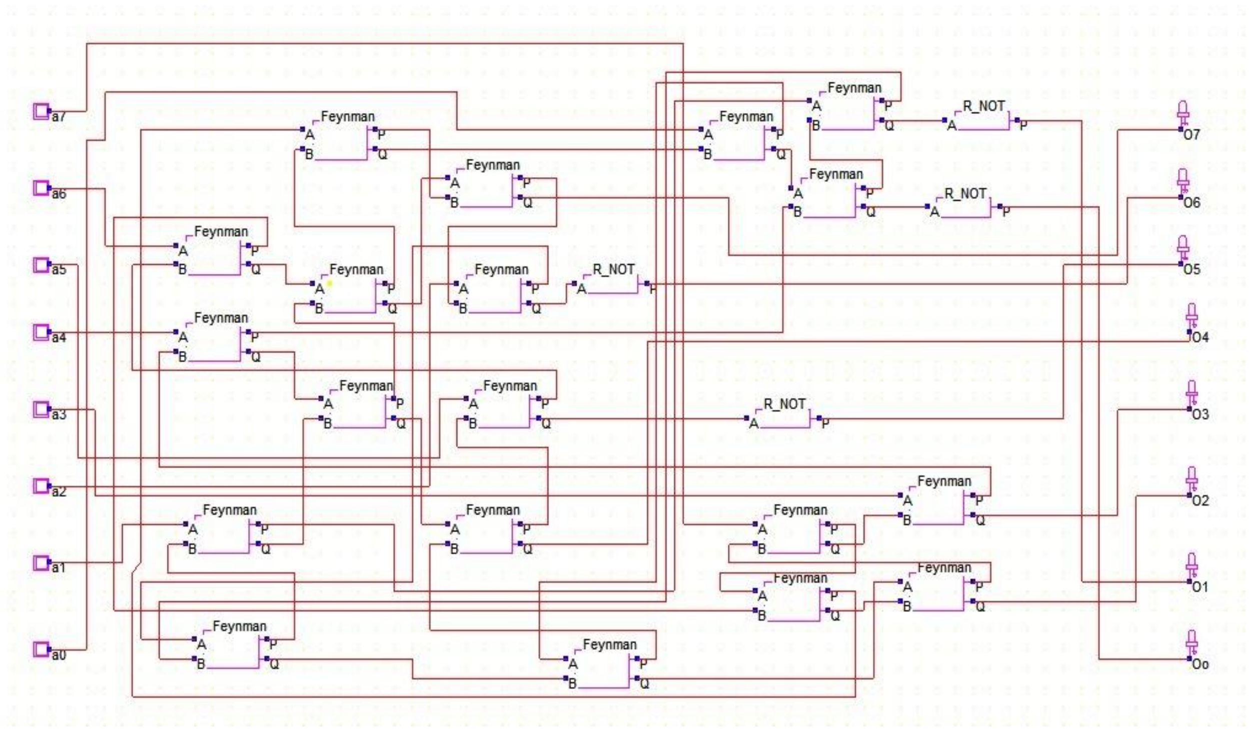


Figure 4.10 Affine Transformation

4.2. LINEAR FEEDBACK SHIFT REGISTER

A linear feedback shift register (LFSR) is a type of shift register that uses linear feedback to generate a sequence of bits. It is commonly used in various applications, including cryptography, error detection and correction, pseudorandom number generation, and digital signal processing.

The structure of an LFSR consists of a shift register, which is a chain of flip-flops (or cells) that store the bits, and a feedback function that determines the input for the first flip-flop based on the contents of other flip-flops. The feedback function is a combination of XOR (exclusive OR) gates that feed the output of selected flip-flops back into the input of the first flip-flop.

Each flip-flop (FF) stores a bit value, and the output of one flip-flop is connected to the input of the next flip-flop. The output of the last flip-flop is often used as the output sequence of the LFSR. The XOR gates (represented by (+)) are used for the feedback function, where selected flip-flop outputs are XORed together and fed back to the input of the first flip-flop.

The feedback taps determine which flip-flop outputs are used for the XOR operation. The positions of the taps determine the period (length) of the generated sequence. A maximum-length LFSR has taps at specific positions that result in a maximal period, producing a pseudorandom sequence that cycles through all possible bit patterns except an all-zero state.

By shifting the bits through the register and applying the feedback function, an LFSR can generate a pseudorandom sequence that appears random but is deterministic and repeatable. The sequence generated by the LFSR depends on the initial state of the flip-flops and the feedback taps chosen.

LFSRs are widely used in various applications due to their simplicity, efficiency, and ability to generate long and predictable sequences.

In order to understand the difference between the minimal and maximal binary sequence length, there are two examples used and explained about it.

4.2.1 Minimal Length LFSR

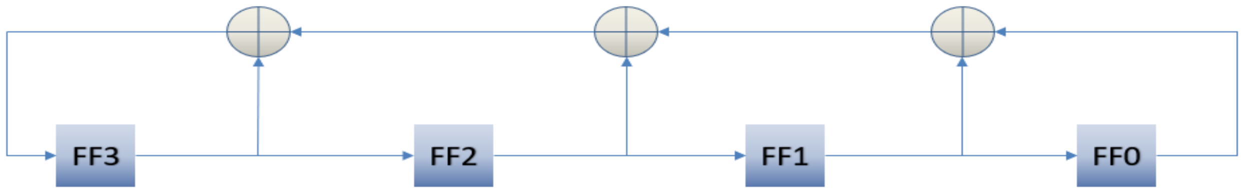


Figure 4.11 Minimal Length LFSR

FF3= FF3⊕FF2⊕FF1⊕FF0	FF2=FF3	FF1=FF2	FF0=FF1
1	0	0	1
0	1	0	0
1	0	1	0
0	1	0	1
0	0	1	0
1	0	0	1

INITIAL VALUE →

5 Clock Cycle

Table 4.1 Truth Table of Minimal Length LFSR

The above diagram is the minimal length LFSR in which all the outputs of the flip-flops are tapped using XOR gate. This is called as minimal length because it has a minimum period of clock cycle of 5 cycles, i.e. it can only able to produce five unique binary sequences.

4.2.2 Maximal Length LFSR

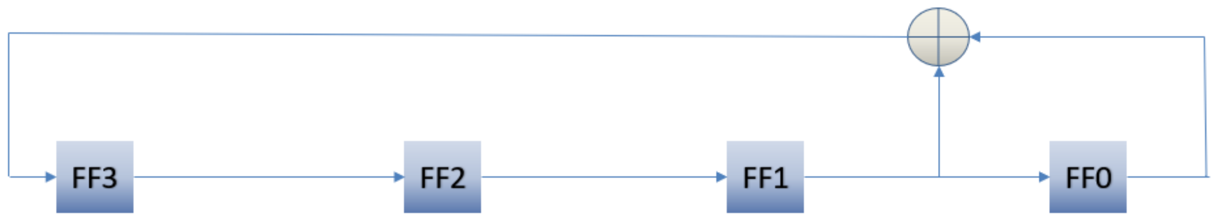


Figure 4.12 Maximal Length LFSR

INITIAL VALUE

FF3=FF1⊕FF0	FF2=FF3	FF1=FF2	FF0=FF1
1	0	0	1
1	1	0	0
0	1	1	0
1	0	1	1
0	1	0	1
1	0	1	0
1	1	0	1
1	1	1	0
1	1	1	1
0	1	1	1
0	0	1	1
0	0	0	1
1	0	0	0
0	1	0	0
0	0	1	0
1	0	0	1

15 Clock Cycle

Table 4.2 Truth Table of Maximal Length LFSR

The above diagram is the maximal length LFSR in which only one output of the flip-flop are tapped using XOR gate. This is called as maximal length because it has a maximum period of clock cycle of 15 cycles, i.e. it can able to produce fifteen unique binary sequences.

4.3 PROPOSED S-BOX ARCHITECTURE

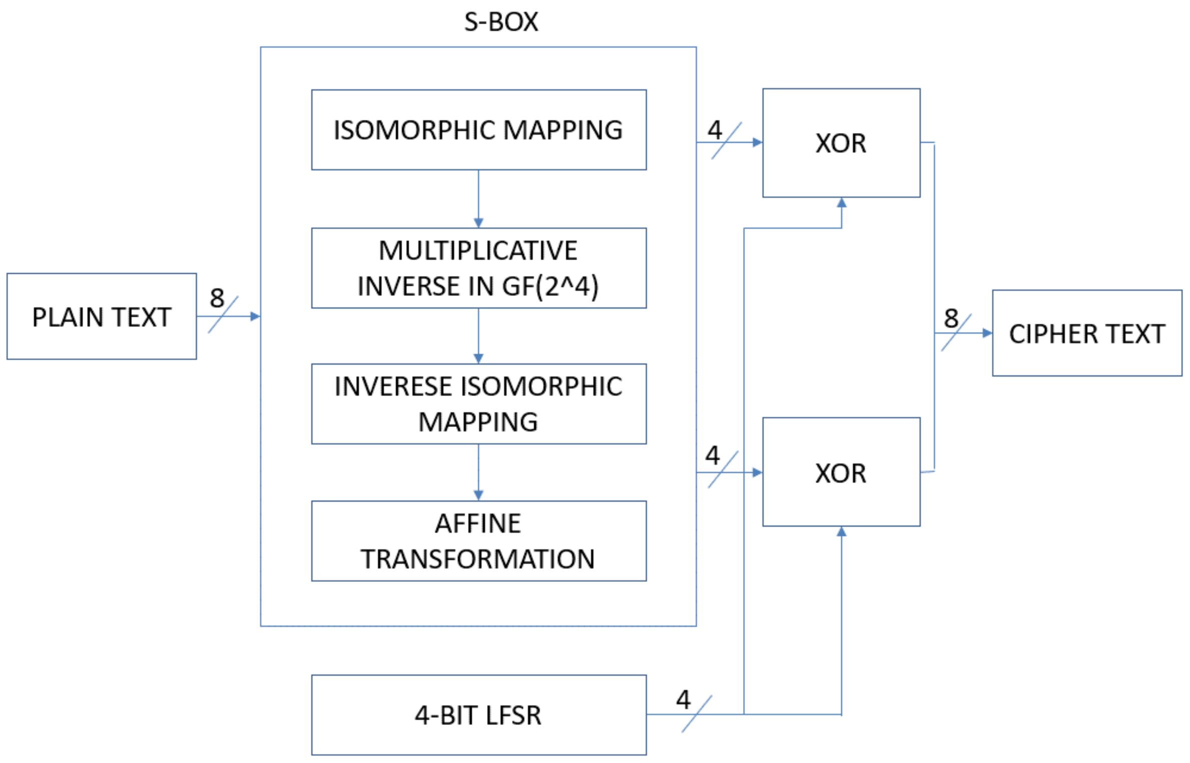


Figure 4.13 Proposed S-box Architecture

The 8-bit plain text is given to static S-box which involves the operation of isomorphic mapping, multiplicative inverse in GF(2^4), inverse isomorphic mapping and affine transformation. After all this process it will produce a 8-bit output data that is separated into two nibbles. The upper nibble is XORed with output of the 4-bit LFSR and the lower nibble is XORed with the output of the same 4-bit LFSR, together it will produce a 8-bit output data which is called as cipher data.

CHAPTER 5

RTL SCHEMATICS

5.1 RTL SCHEMATIC OF IRREVERSIBLE LOGIC GATES

5.1.1 RTL Schematic of Irreversible S-box without LFSR

RTL Schematic of 8-bit S-box without LFSR is obtained from Xilinx ISE 14.7, RTL Schematic (HDL).

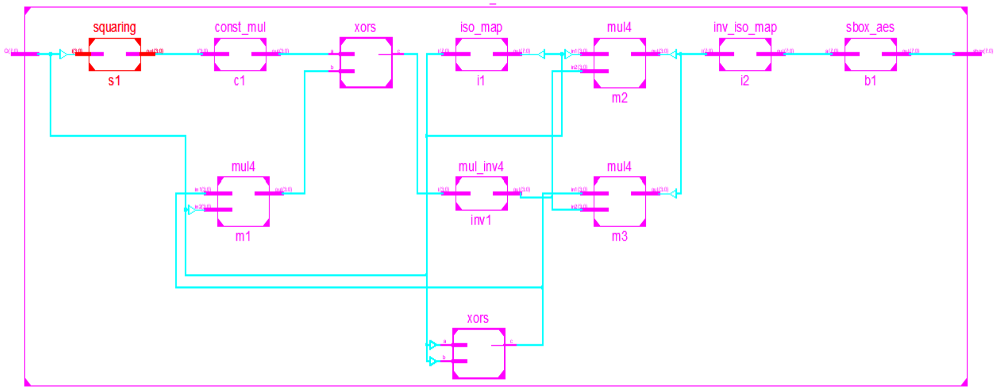


Figure 5.1 RTL Schematic of Irreversible S-box without LFSR

5.1.2 RTL Schematic of Irreversible S-box with LFSR

RTL Schematic of 8-bit S-box with LFSR is obtained from Xilinx ISE 14.7, RTL Schematic (HDL).

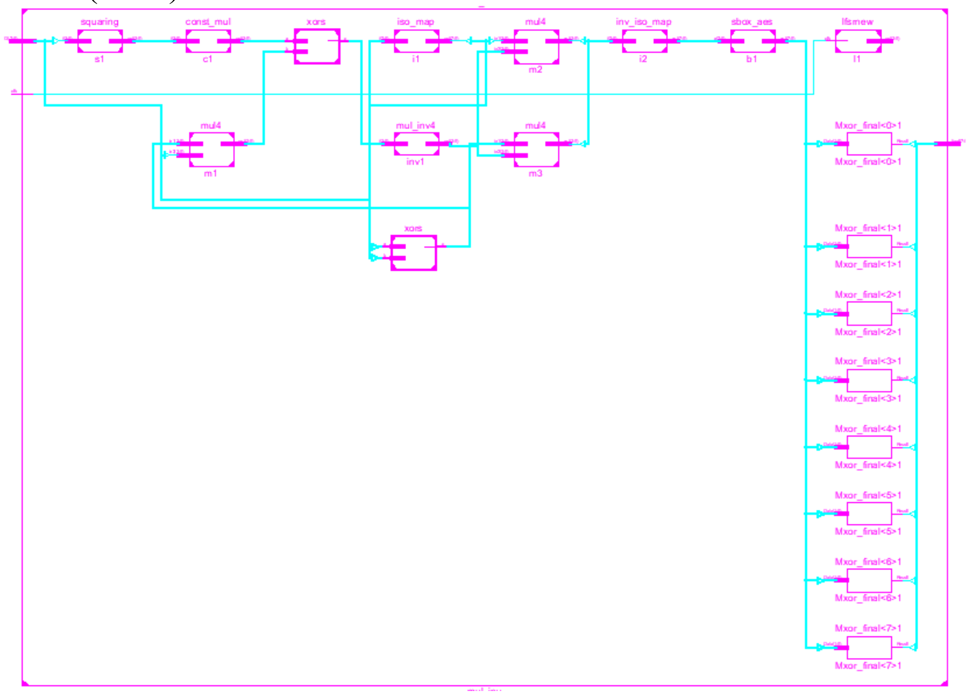
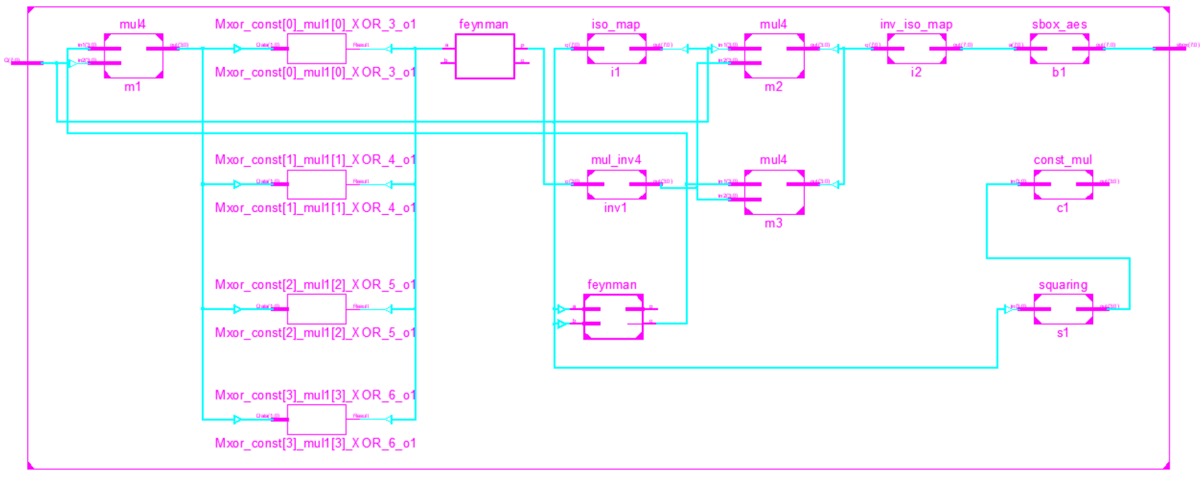


Figure 5.2 RTL Schematic of Irreversible S-box with LFSR

5.2 RTL SCHEMATIC OF REVERSIBLE LOGIC GATES

5.2.1 RTL Schematic of Reversible S-box without LFSR

RTL Schematic of 8-bit reversible S-box without LFSR is obtained from Xilinx ISE 14.7, RTL Schematic (HDL).



mul4
m1
Mxor_const[0]_mul1[0]_XOR_3_o1
Mxor_const[0]_mul1[0]_XOR_3_o1
feynman
iso_map
i1
mul4
m2
inv_iso_map
i2
sbox_aes
b1
Mxor_const[1]_mul1[1]_XOR_4_o1
Mxor_const[1]_mul1[1]_XOR_4_o1
mul_inv4
inv1
mul4
m3
const_mul
c1
Mxor_const[2]_mul1[2]_XOR_5_o1
Mxor_const[2]_mul1[2]_XOR_5_o1
feynman
squaring
s1
Mxor_const[3]_mul1[3]_XOR_6_o1
Mxor_const[3]_mul1[3]_XOR_6_o1

Figure 5.3 RTL Schematic of Reversible S-box without LFSR

5.2.2 RTL Schematic of Reversible S-box with LFSR

RTL Schematic of 8-bit squarer is obtained from Xilinx ISE 14.7, RTL Schematic (HDL).

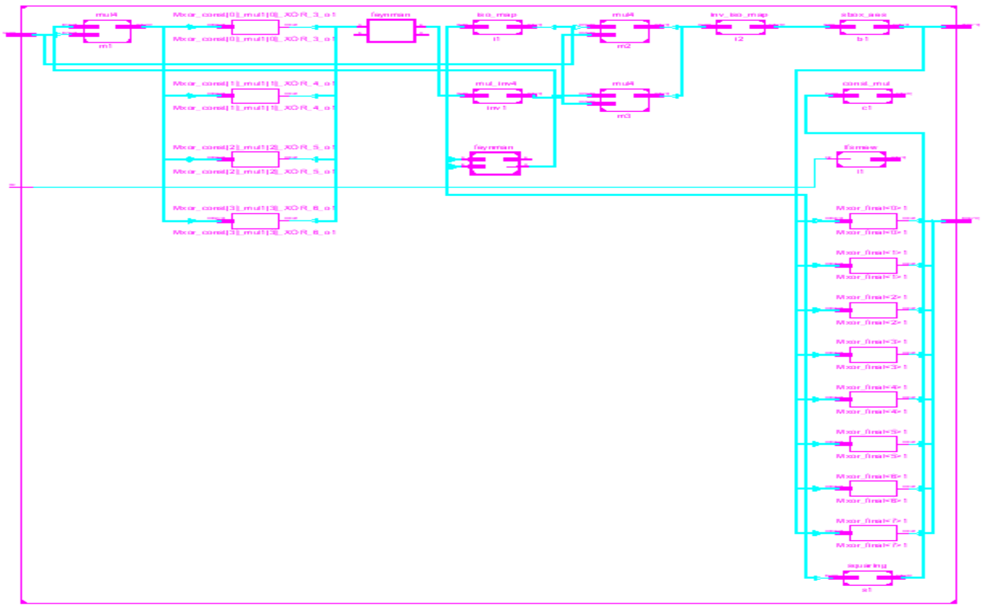


Figure 5.4 RTL Schematic of Reversible S-box with LFSR

CHAPTER 6

SIMULATION RESULTS

6.1 SIMULATION RESULT OF STATIC S-BOX

8-bit Static S-box is designed. This is coded using the programming language Verilog module. The written program is simulated using Xilinx ISE 14.7 for six set of values and the result is verified.

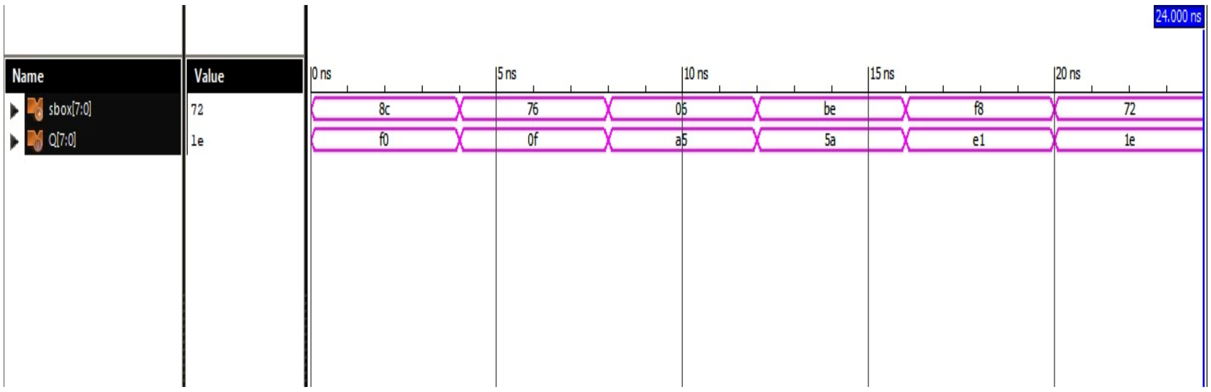


Figure 6.1 Simulation Result of Static S-box

6.2 SIMULATION RESULT OF DYNAMIC S-BOX

8-bit Dynamic S-box is designed. This is coded using the programming language Verilog module. The written program is simulated using Xilinx ISE 14.7 for four set of values and the result is verified.

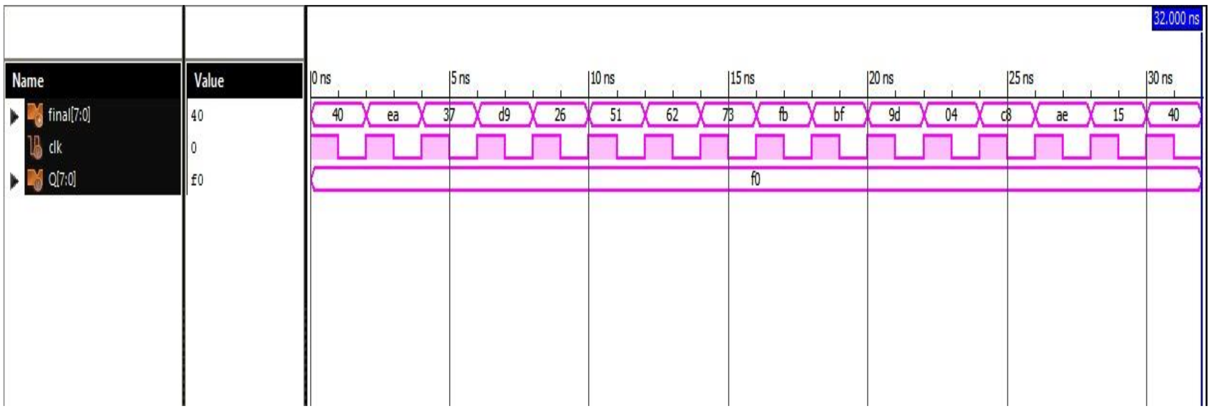


Figure 6.2 Simulation Result of Dynamic S-box

CHAPTER 7

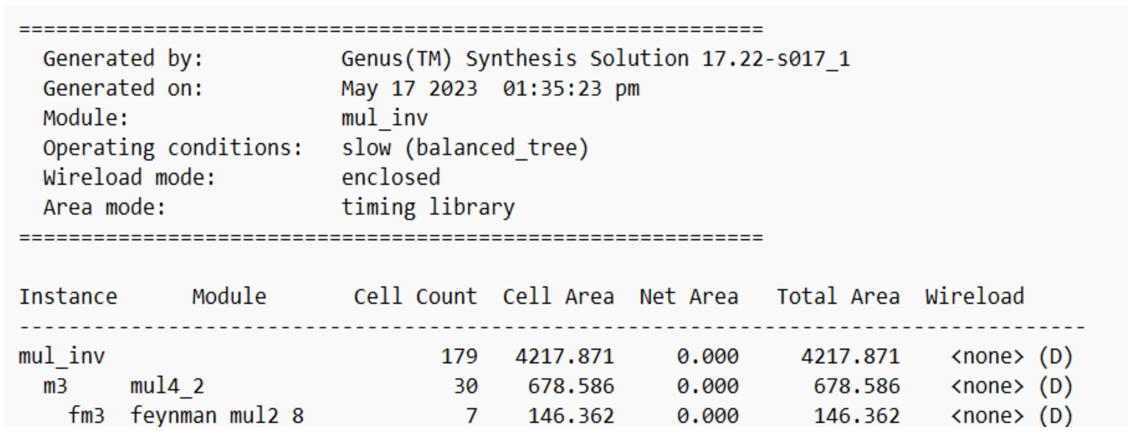
RESULTS AND COMPARISON

7.1 AREA REPORT

7.1.1 Area Report of Irreversible Gates

7.1.1.1 Area Report of Irreversible logic gate without LFSR

Area report of an irreversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

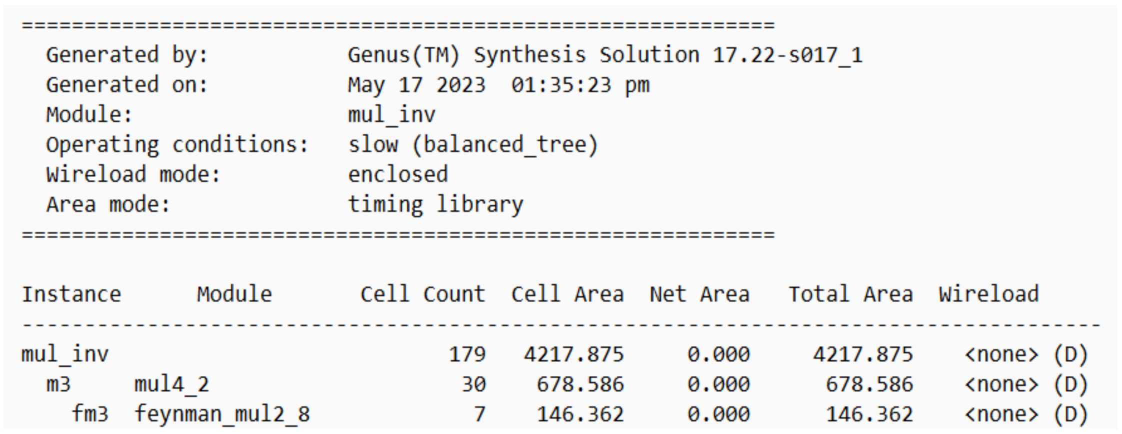


```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:35:23 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance      Module        Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------------------
mul_inv                            179   4217.871     0.000     4217.871    <none> (D)
  m3     mul4_2                     30    678.586     0.000      678.586    <none> (D)
    fm3  feynman mul2 8              7    146.362     0.000      146.362    <none> (D)
```

Figure 7.1 Area Report of Irreversible logic gate without LFSR

7.1.1.2 Area Report of Irreversible logic gate with LFSR

Area report of an irreversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



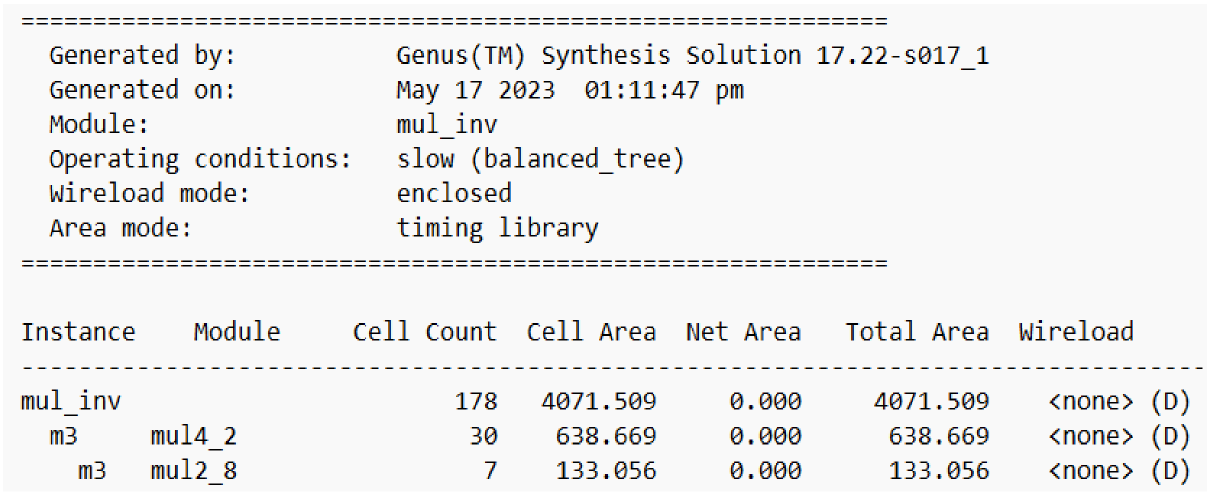
```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:35:23 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance      Module        Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------------------
mul_inv                            179   4217.875     0.000     4217.875    <none> (D)
  m3     mul4_2                     30    678.586     0.000      678.586    <none> (D)
    fm3  feynman_mul2_8              7    146.362     0.000      146.362    <none> (D)
```

Figure 7.2 Area Report of Irreversible logic gate with LFSR

7.1.2 Area report of Reversible Gates

7.1.2.1 Area Report of reversible logic gate without LFSR

Area report of an reversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

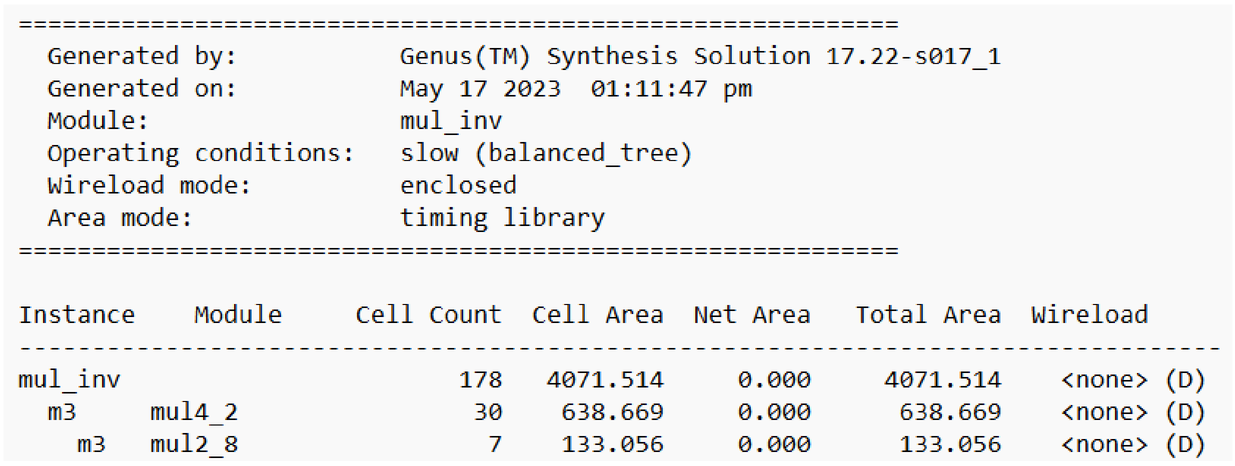


```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:          May 17 2023  01:11:47 pm
  Module:                mul_inv
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

Instance    Module     Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------------
mul_inv                       178   4071.509     0.000     4071.509    <none> (D)
  m3      mul4_2               30    638.669     0.000      638.669    <none> (D)
    m3    mul2_8                7    133.056     0.000      133.056    <none> (D)
```

Figure 7.3 Area Report of reversible logic gate without LFSR

7.1.2.2 Area Report of reversible logic gate with LFSR

Area report of an reversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:          May 17 2023  01:11:47 pm
  Module:                mul_inv
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================

Instance    Module     Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------------
mul_inv                       178   4071.514     0.000     4071.514    <none> (D)
  m3      mul4_2               30    638.669     0.000      638.669    <none> (D)
    m3    mul2_8                7    133.056     0.000      133.056    <none> (D)
```

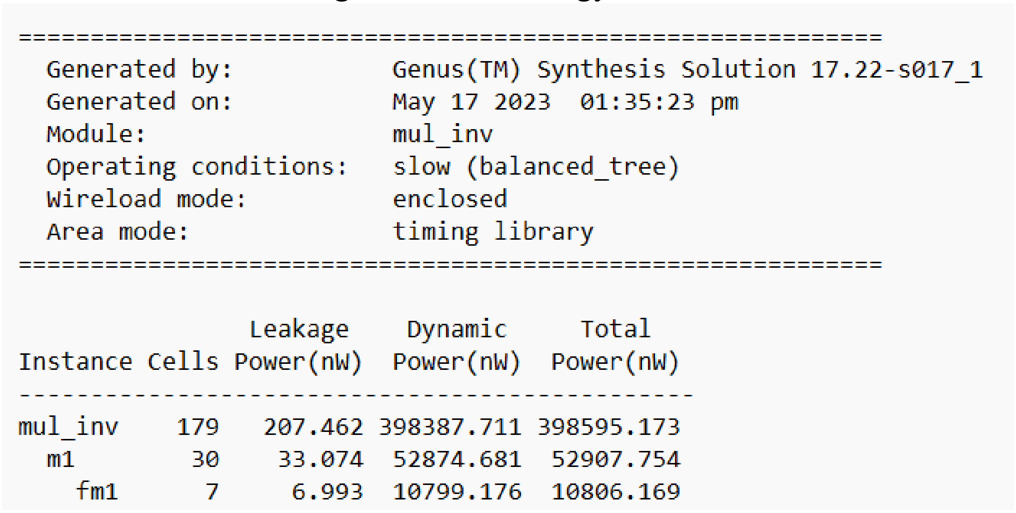
Figure 7.4 Area Report of reversible logic gate with LFSR

7.2 POWER REPORT

7.2.1 Power Report of Irreversible Gates

7.2.1.1 Power Report of Irreversible logic gate without LFSR

Power report of an irreversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

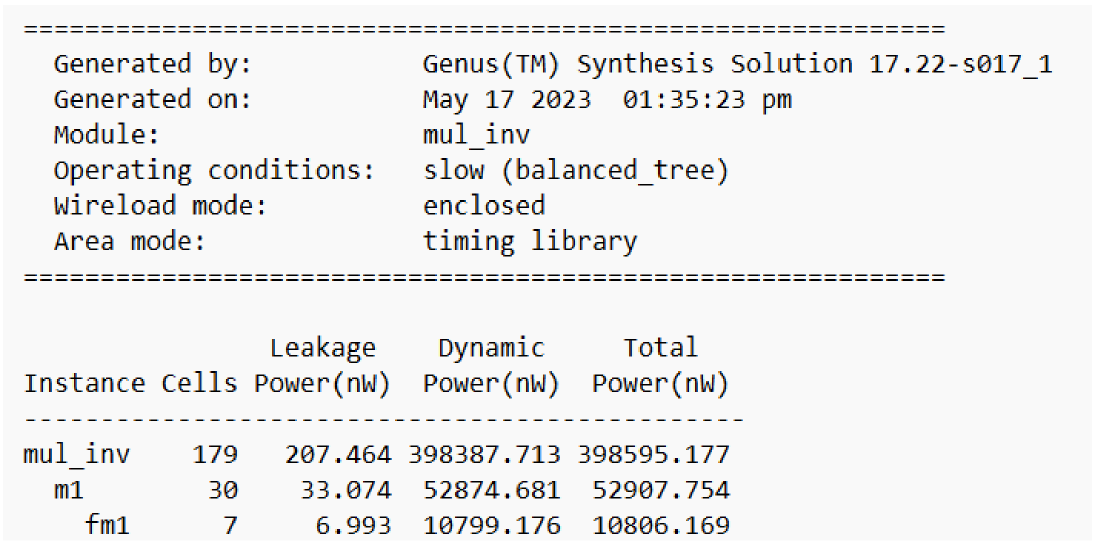


```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:          May 17 2023  01:35:23 pm
  Module:                mul_inv
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================


                 Leakage    Dynamic     Total
Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
mul_inv    179   207.462 398387.711 398595.173
  m1        30    33.074  52874.681  52907.754
    fm1      7     6.993  10799.176  10806.169
```

Figure 7.5 Power Report of irreversible logic gate without LFSR

7.2.1.2 Power Report of Irreversible logic gate with LFSR

Power report of an irreversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



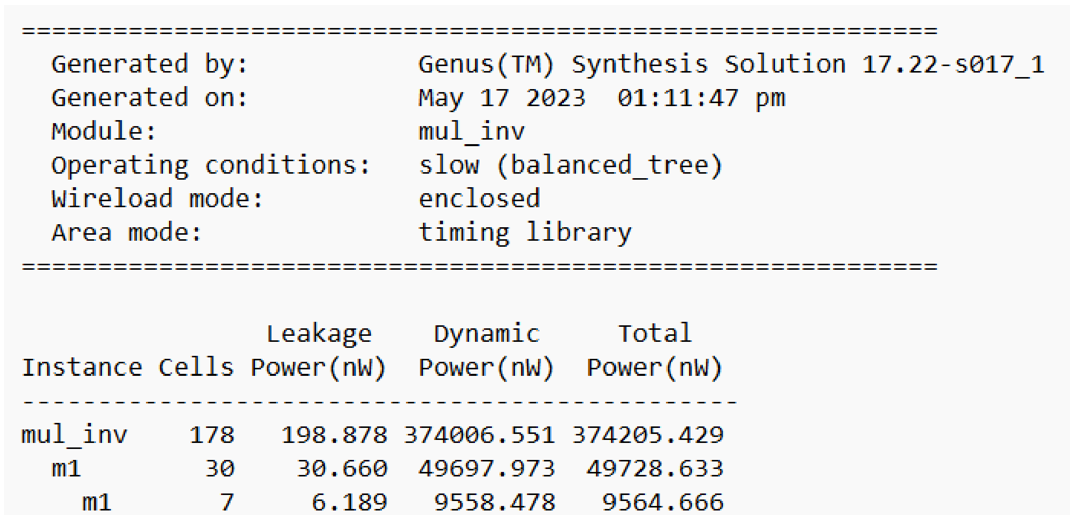
```
============================================================
  Generated by:          Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:          May 17 2023  01:35:23 pm
  Module:                mul_inv
  Operating conditions:  slow (balanced_tree)
  Wireload mode:         enclosed
  Area mode:             timing library
============================================================


                 Leakage    Dynamic     Total
Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
mul_inv    179   207.464 398387.713 398595.177
  m1        30    33.074  52874.681  52907.754
    fm1      7     6.993  10799.176  10806.169
```

Figure 7.6 Power Report of irreversible logic gate with LFSR

7.2.2 Power Report of Reversible Gates

7.2.2.1 Power Report of reversible logic gate without LFSR

Power report of a reversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

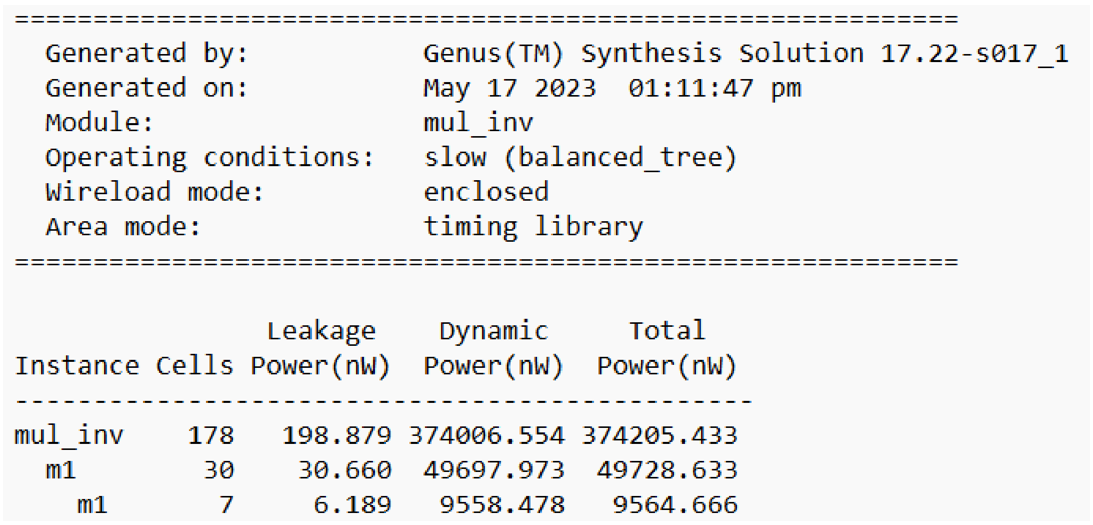


```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:11:47 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                Leakage    Dynamic     Total
Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
mul_inv    178   198.878 374006.551 374205.429
  m1        30    30.660  49697.973  49728.633
    m1       7     6.189   9558.478   9564.666
```

Figure 7.7 Power Report of reversible logic gate without LFSR

7.2.2.2 Power Report of reversible logic gate with LFSR

Power report of a reversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:11:47 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                Leakage    Dynamic     Total
Instance Cells Power(nW)  Power(nW)  Power(nW)
-----------------------------------------------
mul_inv    178   198.879 374006.554 374205.433
  m1        30    30.660  49697.973  49728.633
    m1       7     6.189   9558.478   9564.666
```

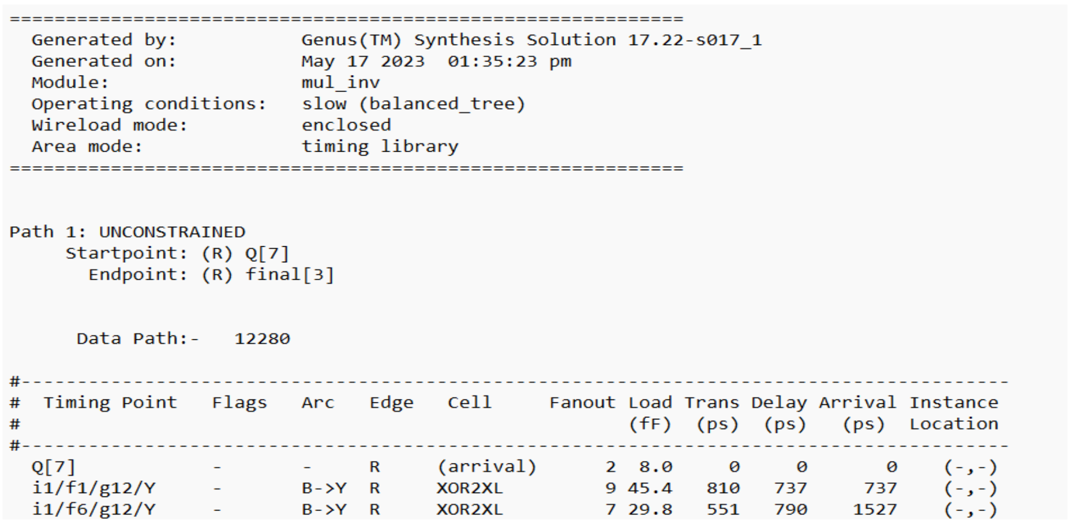
Figure 7.8 Power Report of reversible logic gate with LFSR

7.3 TIMING REPORT

7.3.1 Timing Report of Irreversible Gates

7.3.1.1 Timing Report of Irreversible logic gate without LFSR

Timing report of an irreversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

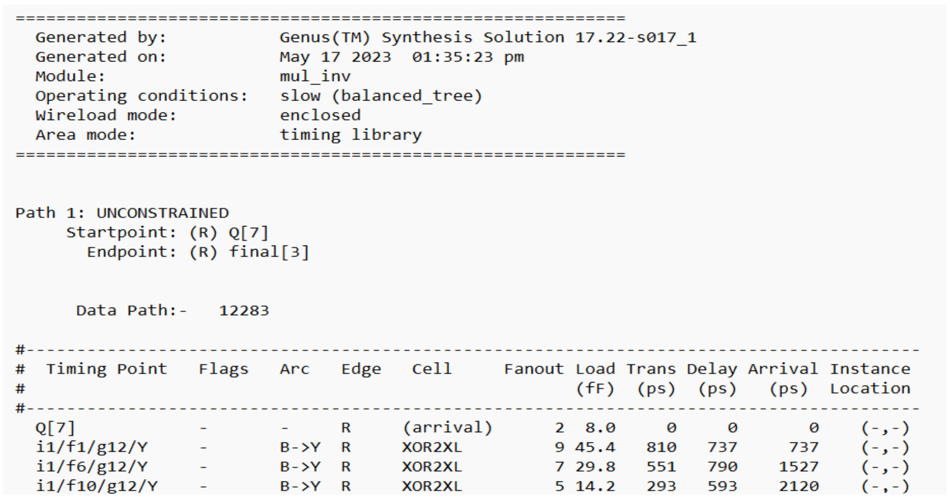


```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:35:23 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: UNCONSTRAINED
     Startpoint: (R) Q[7]
       Endpoint: (R) final[3]


      Data Path:-    12280

#-------------------------------------------------------------------------------------
#  Timing Point    Flags   Arc    Edge    Cell      Fanout  Load Trans Delay Arrival Instance
#                                                           (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------
  Q[7]              -       -      R     (arrival)       2  8.0     0     0       0    (-,-)
  i1/f1/g12/Y       -       B->Y   R     XOR2XL          9 45.4   810   737     737    (-,-)
  i1/f6/g12/Y       -       B->Y   R     XOR2XL          7 29.8   551   790    1527    (-,-)
```

Figure 7.9 Timing Report of irreversible logic gate without LFSR

7.3.1.2 Timing Report of Irreversible logic gate with LFSR

Timing report of an irreversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



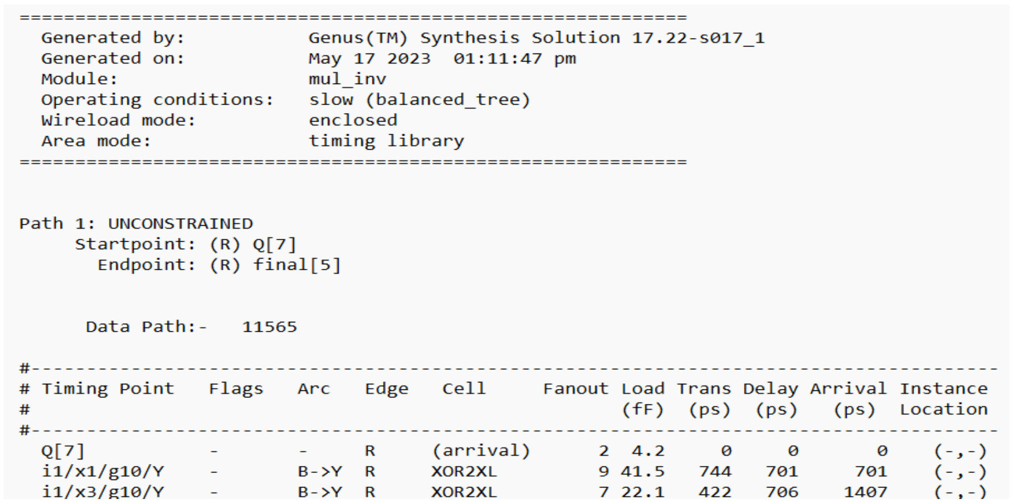
```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:35:23 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: UNCONSTRAINED
     Startpoint: (R) Q[7]
       Endpoint: (R) final[3]


      Data Path:-    12283

#-------------------------------------------------------------------------------------
#  Timing Point    Flags   Arc    Edge    Cell      Fanout  Load Trans Delay Arrival Instance
#                                                           (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------
  Q[7]              -       -      R     (arrival)       2  8.0     0     0       0    (-,-)
  i1/f1/g12/Y       -       B->Y   R     XOR2XL          9 45.4   810   737     737    (-,-)
  i1/f6/g12/Y       -       B->Y   R     XOR2XL          7 29.8   551   790    1527    (-,-)
  i1/f10/g12/Y      -       B->Y   R     XOR2XL          5 14.2   293   593    2120    (-,-)
```

Figure 7.10 Timing Report of irreversible logic gate with LFSR

7.3.2 Timing Report of Reversible Gates

7.3.2.1 Timing Report of reversible logic gate without LFSR

Timing report of a reversible logic gate without LFSR is obtained from Cadence Encounter RTL using 90nm technology.

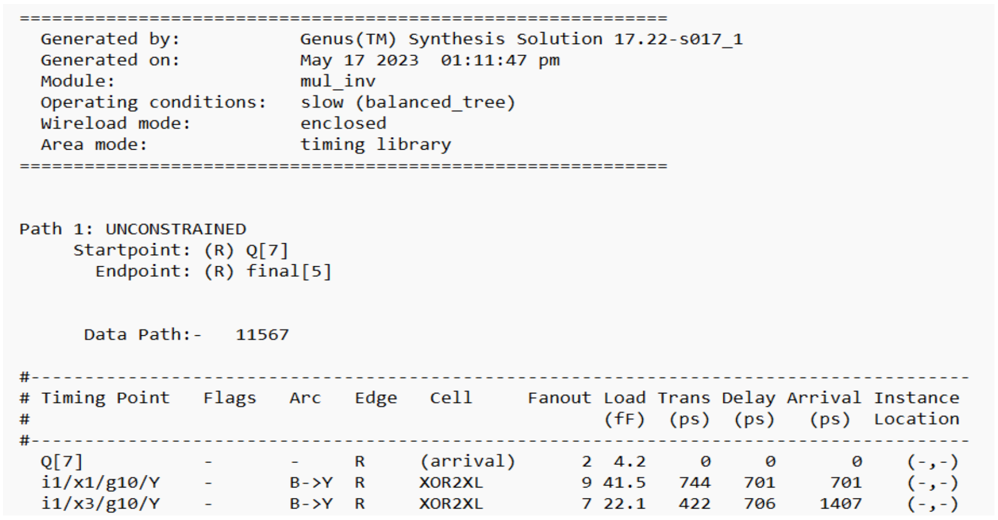


```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:11:47 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: UNCONSTRAINED
     Startpoint: (R) Q[7]
       Endpoint: (R) final[5]


      Data Path:-    11565

#---------------------------------------------------------------------------------------
# Timing Point    Flags    Arc    Edge    Cell        Fanout  Load Trans Delay Arrival Instance
#                                                             (fF)  (ps)  (ps)   (ps)  Location
#---------------------------------------------------------------------------------------
  Q[7]            -        -      R       (arrival)        2   4.2     0     0       0    (-,-)
  i1/x1/g10/Y     -        B->Y   R       XOR2XL           9  41.5   744   701     701    (-,-)
  i1/x3/g10/Y     -        B->Y   R       XOR2XL           7  22.1   422   706    1407    (-,-)
```

Figure 7.11 Timing Report of reversible logic gate without LFSR

7.3.2.2 Timing Report of reversible logic gate with LFSR

Timing report of a reversible logic gate with LFSR is obtained from Cadence Encounter RTL using 90nm technology.



```
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           May 17 2023  01:11:47 pm
  Module:                 mul_inv
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: UNCONSTRAINED
     Startpoint: (R) Q[7]
       Endpoint: (R) final[5]


      Data Path:-    11567

#---------------------------------------------------------------------------------------
# Timing Point    Flags    Arc    Edge    Cell        Fanout  Load Trans Delay Arrival Instance
#                                                             (fF)  (ps)  (ps)   (ps)  Location
#---------------------------------------------------------------------------------------
  Q[7]            -        -      R       (arrival)        2   4.2     0     0       0    (-,-)
  i1/x1/g10/Y     -        B->Y   R       XOR2XL           9  41.5   744   701     701    (-,-)
  i1/x3/g10/Y     -        B->Y   R       XOR2XL           7  22.1   422   706    1407    (-,-)
```

Figure 7.12 Timing Report of reversible logic gate with LFSR

7.4 COMPARISON

The comparison of Reversible and Irreversible gates taken for analysis is shown in Table 7.1. The Reversible and Irreversible gates are compared in terms of Area, Power, Time parameters using report obtained from Cadence Encounter RTL using 90nm technology.

Table 7.1. Comparison of Reversible and Irreversible gates

MODULE NAME	**AREA (μm^2)**	**POWER (nW)**	**TIME (ps)**	**ADP (10^{-14} m^2s)**	**PDP (10^{-9} Ws)**
Irreversible Static S-box	4217.871	398595.173	12280	5179.545	4894.748
Irreversible Dynamic S-box	4217.875	398595.177	12283	5180.815	4895.944
Reversible Static S-box	4071.509	374205.429	11565	4708.001	4327.685
Reversible Dynamic S-box	4071.514	374205.433	11567	4709.520	4328.434

The Comparison table shows that when the irreversible logic gate is used in the hardware there will be information loss due to power leakage because, in an irreversible logic gate, a one-to-one mapping is not available in the irreversible logic gate. while reversible logic gates are implemented in the hardware the information loss can be prevented due to one-to-one mapping, so the power leakage can be minimized, so we can view the power result from the cadence and comparison table that irreversible logic gate is consumed 1(mW) more than reversible logic gate.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The design of Dynamic S-box using Rijndael algorithm which include the use of reversible logic gates and Linear Feedback Shift Register is simulated using Xilinx ISE 14.7 Design suite and synthesised in Cadence Encounter RTL.

In VLSI design process, Area, Delay and Power are the important factors that determine the performance of any circuit. Hence the reports of the same were obtained from the Cadence Encounter RTL and analysed. The 8-bit S-box including isomorphic mapping, multiplicative inverse in GF(2^4), inverse isomorphic mapping and affine transformation were designed to produce a 8-bit cipher data.

Result analysis shows that the modified S-box is efficient in terms of power, because of the use of reversible logic gates which reduces the chance of losing information since it can able to extract input from the output. The Area and Timing report are also obtained from the Cadence Encounter RTL.

In future, the dynamic S-box can be used to encrypt different data such as images, videos, voice, etc by converting those data into binary values. The same dynamic S-box can be implemented in FPGA with a complete ability to encrypt 128 bits of plain text. The seed value used in the LFSR can be changed for different user so that each user will have a unique key which will improve the security of the more. The proposed S-box can be employed in many Internet of Things (IoT) applications which involves the use of wired or wireless computer network. The security of the S-box can be still improved by altering the LFSR’s seed value for each 16 S-box.

REFERENCES

1. A. Barrera, C.W. Cheng and S. Kumar, "Fast Implementation of the Rijndael Substitution Box for Cryptographic AES", Data Intelligence and Security, South Padre Island, TX, USA, pp. 20-25, 2020.

2. A. Nakashima, R. Ueno and N. Homma, "AES S-Box Hardware With Efficiency Improvement Based on Linear Mapping Optimization", in IEEE Transactions on Circuits and Systems II: Express Briefs, Oct 2022, pp. 3978-3982, Vol. 69, No. 10.

3. A. Reyhani-Masoleh, M. Taha, and D. Ashmawy, "New area record for the AES combined s-box/inverse s-box," in Proc. IEEE Computer Arithmetic, 2018, pp. 145-152.

4. Bahram Rashidi and Bahman Rashidi, "Implementation of An Optimized and Pipelined Combinational Logic Rijndael S-Box on FPGA", Computer Network And Information Security, pp. 41- 48, 2018.

5. C.V. Kavya Suvarchala, "Effective Implementation of AES Algorithm using Reversible Logic", International Journal of VLSI System Design and Communication Systems, 2016, pp. 2322-0929, Vol 04.

6. D.K. Sushma and Dr. Manju Devi, "Design of S- box and INV S-box using Composite Field Arithmetic for AES Algorithm", International Journal of Engineering Research & Technology, Vol 6, pp. 1-2, 2018.

7. D. Lee and Y. Kim, "Design of a Light-Weight Key Scheduler for AES using LFSR for IoT Applications," IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Gangwon, Korea, Republic of, pp. 1-2, 2021.

8. D Srinivas, Boda Aruna, Ravi Boda "Implementation of Advanced Encryption Standard using Reversible Logic Gate", International Journal of latest Engineering Research and Application, 2017, pp. 240-242, Vol 2.

9. J. J. Tay, M. L. Dennis, M. M. Wong, C. Zhang, and I. Hijazin, “Constructions of low multiplicative complexity inversion circuit for compact AES S-box”, IEEE Region 10 Conf, Oct. 2018, pp. 0540–0544.

10. J. Song, K. Lee and J. Park, "Low Area and Low Power Threshold Implementation Design Technique for AES S-Box", in IEEE Transactions on Circuits and Systems II: Express Briefs, March 2023, pp. 1169-1173 Vol. 70, No. 3.

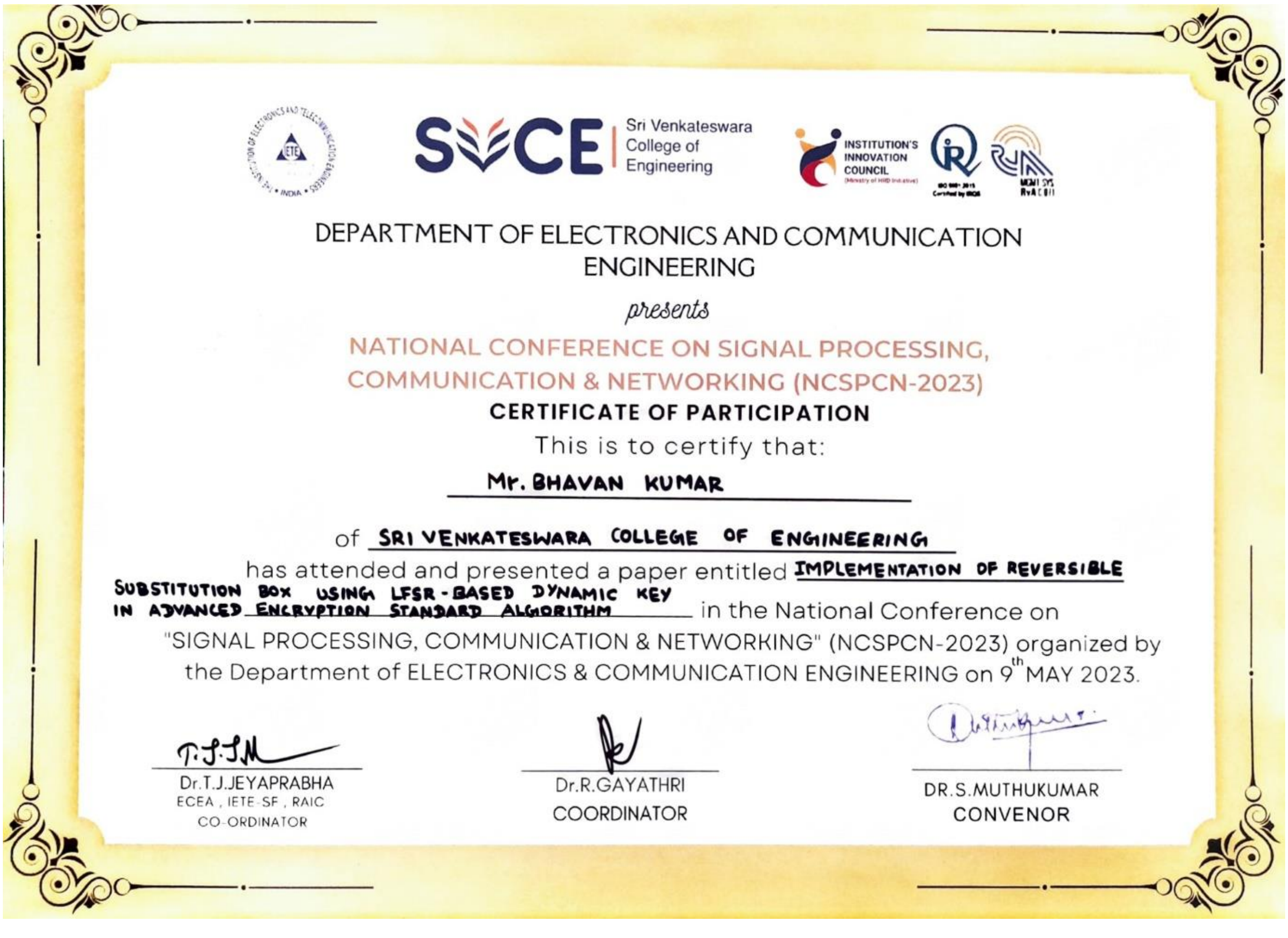
11. Mummadi Swathi, Dr. Bhawana Rudra “Implementation of Reversible Logic Gates with Quantum Gates”, IEEE 11th Annual Computing and Communication Workshop and Conference, 2021, pp. 438-440, Vol 55.

12. Saravanan P, Kalpana P “Novel Reversible Design of 128-bit Advanced Encryption Satandard (AES) cryptographic algorithm for Wireless Sensor”, Wireless Pers Communication, 2018, pp. 342-344, Vol 07.

13. Sushma D K , Manju Devi, “Design of S-box and INV S-box using composite field for AES algorithm”, International Journal of Engineering Research and Technology (IJERT) ,Vol. 06, 2018.

14. W. I. El Sobky, A. A. Isamail, A. S. Mohra and A.M. Hassan, "Implementation Mini by Substitution Box in Galois Field (24),“ International Telecommunications Conference (ITC-Egypt), Alexandria, Egypt, pp. 1-4, 2021.

15. Y.T. Teng, W.L. Chin, D.K. Chang, P.Y. Chen and P.W. Chen, "VLSI Architecture of S-Box With High Area Efficiency Based on Composite Field Arithmetic", in IEEE Access, 2022, pp. 2721-2728, Vol. 10.



SVCE Sri Venkateswara College of Engineering

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

presents

NATIONAL CONFERENCE ON SIGNAL PROCESSING, COMMUNICATION & NETWORKING (NCSPCN-2023)

CERTIFICATE OF PARTICIPATION

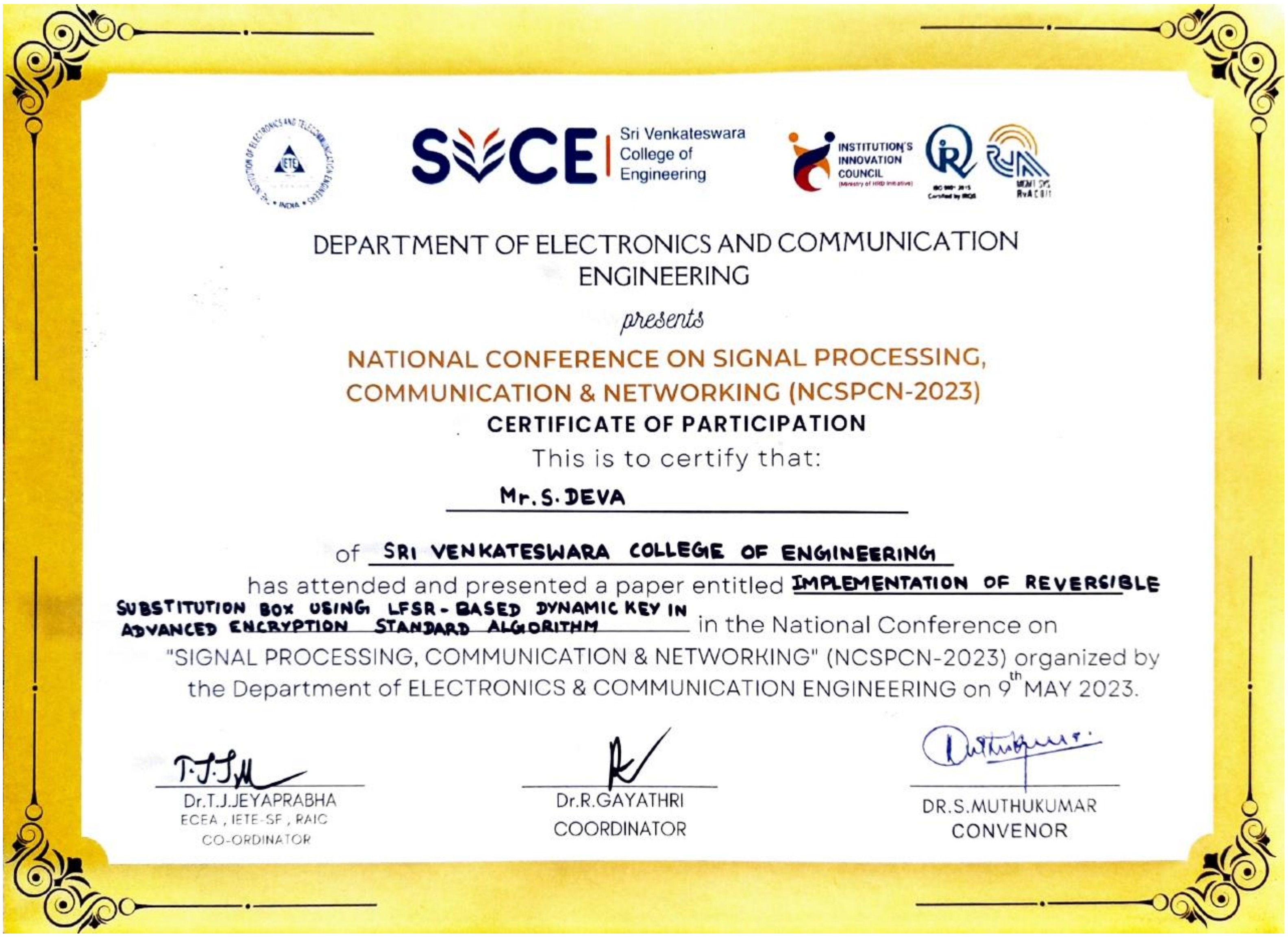
This is to certify that:

Mr. BHAVAN KUMAR

of SRI VENKATESWARA COLLEGE OF ENGINEERING

has attended and presented a paper entitled IMPLEMENTATION OF REVERSIBLE SUBSTITUTION BOX USING LFSR-BASED DYNAMIC KEY IN ADVANCED ENCRYPTION STANDARD ALGORITHM in the National Conference on "SIGNAL PROCESSING, COMMUNICATION & NETWORKING" (NCSPCN-2023) organized by the Department of ELECTRONICS & COMMUNICATION ENGINEERING on 9th MAY 2023.

[illegible]	[illegible]	[illegible]
Dr.T.J.JEYAPRABHA ECEA, IETE-SF, RAIC CO-ORDINATOR	Dr.R.GAYATHRI COORDINATOR	DR.S.MUTHUKUMAR CONVENOR



SVCE Sri Venkateswara College of Engineering

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

presents

NATIONAL CONFERENCE ON SIGNAL PROCESSING, COMMUNICATION & NETWORKING (NCSPCN-2023)

CERTIFICATE OF PARTICIPATION

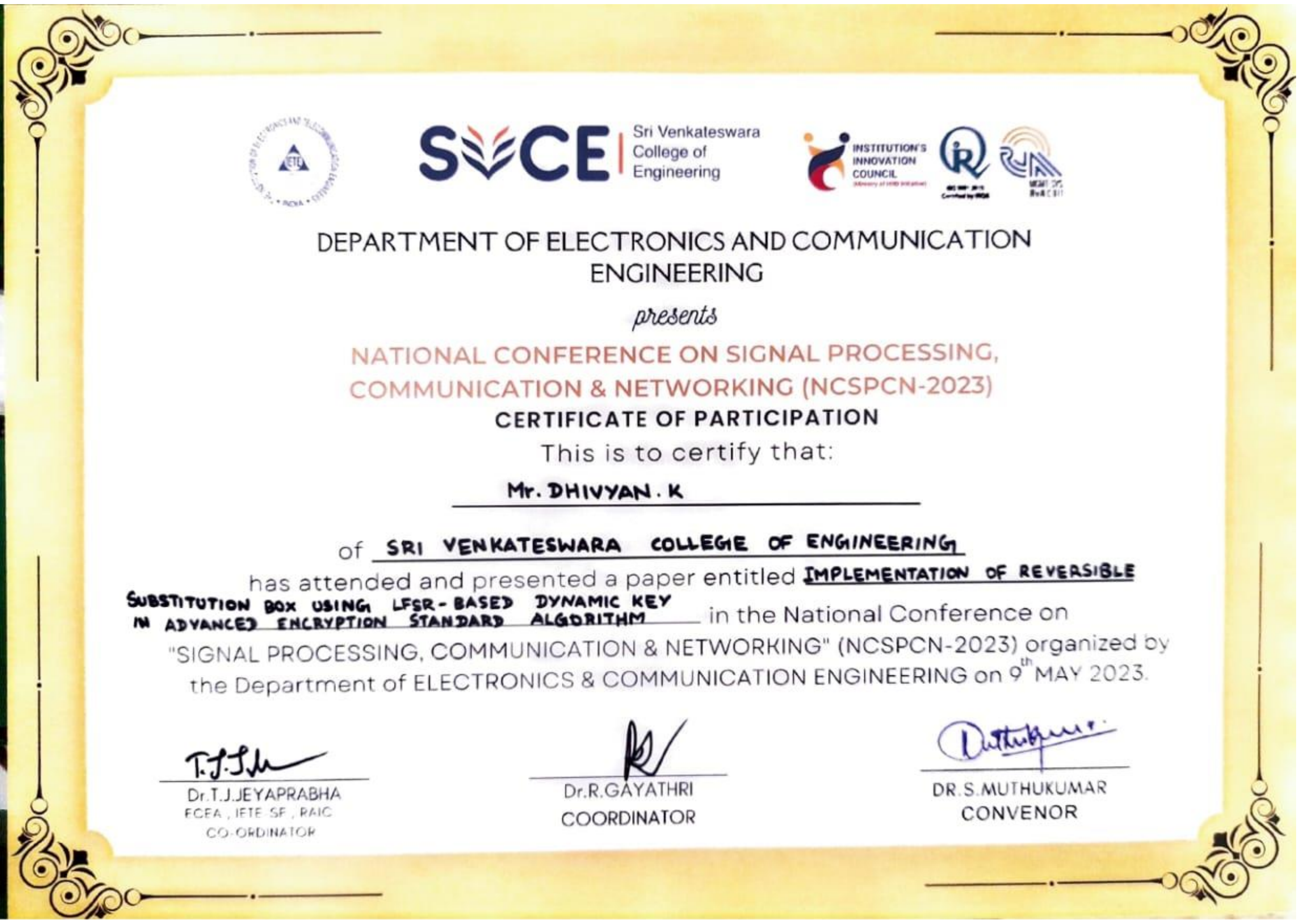
This is to certify that:

Mr. S. DEVA

of SRI VENKATESWARA COLLEGE OF ENGINEERING

has attended and presented a paper entitled IMPLEMENTATION OF REVERSIBLE SUBSTITUTION BOX USING LFSR-BASED DYNAMIC KEY IN ADVANCED ENCRYPTION STANDARD ALGORITHM in the National Conference on "SIGNAL PROCESSING, COMMUNICATION & NETWORKING" (NCSPCN-2023) organized by the Department of ELECTRONICS & COMMUNICATION ENGINEERING on 9th MAY 2023.

[illegible]	[illegible]	[illegible]
Dr.T.J.JEYAPRABHA ECEA, IETE-SF, RAIC CO-ORDINATOR	Dr.R.GAYATHRI COORDINATOR	DR.S.MUTHUKUMAR CONVENOR



Sri Venkateswara College of Engineering

INSTITUTION'S INNOVATION COUNCIL

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

presents

NATIONAL CONFERENCE ON SIGNAL PROCESSING, COMMUNICATION & NETWORKING (NCSPCN-2023)

CERTIFICATE OF PARTICIPATION

This is to certify that:

Mr. DHIVYAN. K

of SRI VENKATESWARA COLLEGE OF ENGINEERING

has attended and presented a paper entitled IMPLEMENTATION OF REVERSIBLE SUBSTITUTION BOX USING LFSR-BASED DYNAMIC KEY IN ADVANCED ENCRYPTION STANDARD ALGORITHM in the National Conference on "SIGNAL PROCESSING, COMMUNICATION & NETWORKING" (NCSPCN-2023) organized by the Department of ELECTRONICS & COMMUNICATION ENGINEERING on 9th MAY 2023.

[illegible]
Dr.T.J.JEYAPRABHA
FCEA, IETE-SF, RAIC
CO-ORDINATOR

[illegible]
Dr.R.GAYATHRI
COORDINATOR

[illegible]
DR.S.MUTHUKUMAR
CONVENOR

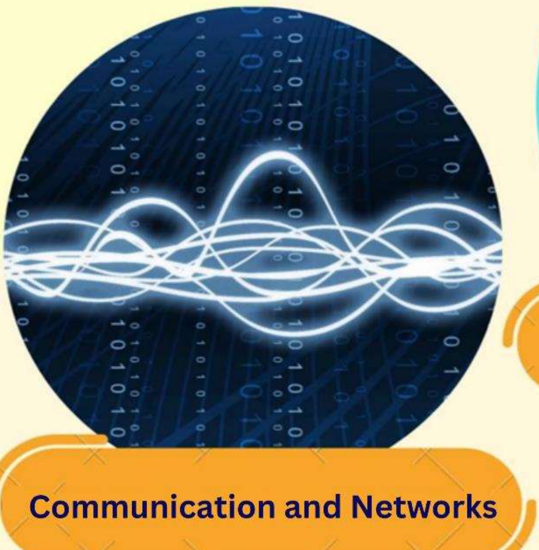
SVCE | Sri Venkateswara College of Engineering

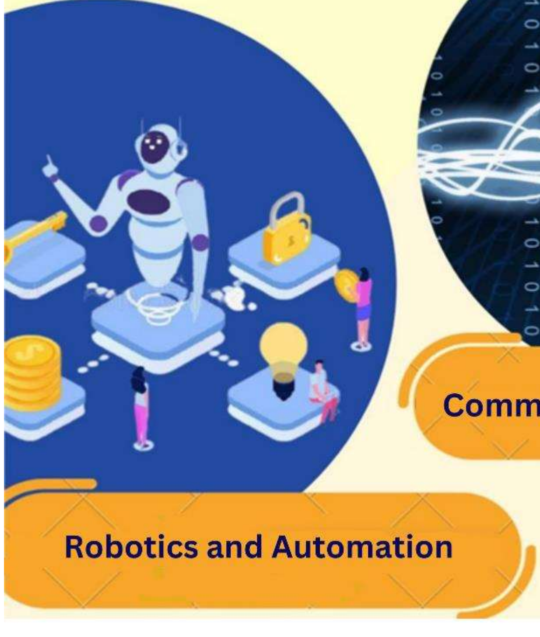
DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

National Conference on Signal Processing,Communication and Networking

NCSPCN-2023

May 9th,2023





svce.ac.in

Implementation of Reversible Substitution box usingLFSR-based dynamic key in Advanced Encryption Standard Algorithm

Bhavan Kumar C
Department of Electronics and Communication Engineering
Sri Venkateswara College of Engineering
Sriperumbudur, India
bhar19118@gmail.com

Dhivyan K
Department of Electronics and Communication Engineering
Sri Venkateswara College of Engineering
Sriperumbudur, India
dhivyankumar253@gmail.com

Deva S
Department of Electronics and Communication Engineering
Sri Venkateswara College of Engineering
Sriperumbudur, India
deva432002@gmail.com

Mrs.B.Sarala
Assistant Professor
Department of Electronics and Communication Engineering
Sri Venkateswara College of Engineering
Sriperumbudur, India
sarala@svce.ac.in

Abstract

The security of the data has been a primary concern as that of the data processing has been faster. The evolution of the data processing capability is being increased, so that the data extraction attacks on a device has also been increasing on day to day life. It is well known that the AES algorithm has been registered as the standard encryption model since 2001. The purpose of this project is to enhance the security of current Substitution box used in AES with thehelp of a Linear Feedback Shift Register (LFSR). This project work offers a simple and innovative method for building dynamic and key dependant S-boxes utilising Mul inv and Aff trans. Here AES with plain text (128-bit) given to the S-box, the 128 bits or 16 bytes is converted into four subcomponents each with 4 bytes. Each byte or 8-bit is givento the mul inv and Aff trans, The output of this above processwill be 128-bit cipher text. The 128-bit in the LFSR is xor with the output to become a dynamic s-box. The same will becontinued for N rounds. Followed by this, reversible logic is applied. Reversible logic has numerous methods. Toffoli, one of the methods in reversible logic, is used to maximize speedand reduce energy consumption. The proposed model of generating dynamic and key depended on s-box is the alternative to the existing s-box. As a conclusion, the efficiency ofthe LFSR based substitution-box increases, and encryption becomes stronger.

I. INTRODUCTION

The NIST Advance Encryptions Standard (AES) is one of the most familiar encryption algorithm ones. The USA government has approved it as a standard for encryption. It is offered in a variety of encryption packages. The National Security Agencies (NSA) initially approved an open, widely usable cipher, AES, for use with top-secret data. The AES was created by Joan and Vincet Rijmaen, two cryptographers from Belgium. Rijndael is a block cipher used by AES. The extremely durable AES algorithm has proven unbreakable to all kind of illegal attacks to date. LFSR-based AES is non-linear symmetric block encryption that can handle 128-bit data bit with cryptographic keys that are 128 bits long. AES-128 process the data block in 10 iterations,