

VLSI IMPLEMENTATION OF AES ALGORITHM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI Design and Embedded System

Submitted by

SAURABH KUMAR

Roll # 211EC2117



*Department of Electronics and communication
Engineering*
National Institute of Technology, Rourkela
2011-2013

VLSI IMPLEMENTATION OF AES ALGORITHM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

Master of Technology

In

VLSI Design and Embedded Systems

Submitted by

SAURABH KUMAR

Roll # 211EC2117

Under the guidance of

PROF. K. K. MAHAPATRA



*Department of Electronics and communication
Engineering*

National Institute of Technology, Rourkela
2011-2013



**NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA**

CERTIFICATE

This is to certify that the thesis titled, “**VLSI IMPLEMENTATION OF AES ALGORITHM**” submitted by **Saurabh Kumar, Roll No-211EC2117** in partial fulfilment of the requirements for the award of Master of Technology Degree in **Electronics & Communication Engineering** with specialization in **VLSI Design and Embedded System** during **2011-2013** at the National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

ROURKELA

PLACE-NIT Rourkela

DATE-03/06/2013

Prof. (Dr.) K. K. Mahapatra
Department of Electronics &
Communication Engineering
National Institute of Technology, Rourkela

Dedicated To My Family
And
My Teachers

ACKNOWLEDGEMENT

I am grateful to my research advisor **Prof. K. K. Mahapatra** for providing me the opportunity to realize this work. He inspired, motivated, encouraged and gave me full freedom to do my work with proper suggestions throughout my research work. I am grateful to **Prof. K. K. Mahapatra** for his kind and moral support throughout my academics at National Institute of Technology, Rourkela.

Next, I want to express my respects to **Prof. S. K. Patra, Prof. S. Meher, Prof. A. K. Swain, Prof. P. K. Tiwari, Prof. S. K. Behera, Prof. D. P. Acharya** and **Prof. N. Islam** for teaching me and also helping me how to learn. They have been great sources of inspiration to me and I thank them from the bottom of my heart. I would like to thank **Vijay Kumar Sharma** sir, my senior, for helping me in VLSI.

I would like to thank all faculty members and staff of the Department of Electronics and Communication Engineering, N.I.T. Rourkela for their generous help in various ways for the completion of this thesis.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I must thank Preeti madam, Jagannath sir and Tom sir for their support and help. I also thank Srinivas sir and Venkatratnam sir for their good attitude and behaviour.

I am especially indebted to my parents for their love, sacrifice, and support. They are my first teachers after I came to this world and have set great examples for me about how to live, study, and work.

SAURABH KUMAR

Saurabhsit098@gmail.com

CONTENTS

ACKNOWLEDGEMENTS.....	iv
LIST OF TABLES.....	viii
LIST OF FIGURES.....	ix
ABSTRACT.....	1
CHAPTER 1 INTRODUCTION.....	3
1.1 MOTIVATION.....	4
1.2 LITERATURE REVIEW.....	6
1.2.1 High Speed AES Design.....	6
1.2.2 Architecture and Implementation of S-BOX.....	7
1.3 RESEARCH OBJECTIVE.....	8
1.4 ORGANIZATION.....	8
CHAPTER 2 ADVANCED ENCRYPTION STANDARD (AES) ALGORITHM.....	9
2.1 DEFINITION AND HISTORY OF CRYPTOGRAPHY.....	10
2.2 GALOIS FIELD.....	11
2.3 THE DATA ENCRYPTION STANDARD (DES).....	11
2.4 THE ADVANCED ENCRYPTION STANDARD (AES).....	12
2.4.1 Subbytes/Inverse Subbytes Transformation.....	14
2.4.2 Shiftrows/Inverse Shiftrows Transformation.....	16
2.4.3 Mixcolumns/Inverse Mixcolumns Transformation.....	16
2.4.4 Addroundkey Transformation and Key Expansion.....	17
2.4.4.1 Addroundkey Transformation.....	17
2.4.4.2 Key Expansion.....	17
2.5 COMPOSITE FIELD ARITHMETIC S-BOX.....	18
2.5.1 Addition operation in GF (2^4).....	20
2.5.2 Squaring operation in GF (2^4).....	20
2.5.3 Multiplication with constant, λ	21
2.5.4 Galois field GF (2^4) multiplication.....	22
2.5.4.1 Multiplication with constant, ϕ	22

2.5.4.2 Galois field GF (2^2) multiplication.....	23
2.5.5 Multiplicative Inversion in GF (2^4).....	24
CHAPTER 3 IMPLEMENTATION OF PROPOSED ARCHITECTURE FOR S-BOX.....	25
3.1 INTRODUCTION.....	26
3.2 PROPOSED ARCHITECTURE OF S-BOX.....	26
3.2.1 Introduced an operator (op) after merging of some blocks.....	27
3.2.2 Implementation of multiplicative inverse in GF (2^4) using multiplexor.....	28
3.2.3 Reduced the critical path of multiplication in GF (2^2).....	30
3.3 IMPLIMENTATION OF PROPOSED ARCHITECTURE OF S-BOX.....	31
3.3.1 ASIC Implementation of MI in GF (2^4).....	31
3.3.1 ASIC Implementation of multiplication in GF (2^2).....	32
3.3.3 ASIC and FPGA Implementation of proposed S-BOX.....	32
3.4 CONCLUSION.....	34
CHAPTER 4 HIGH SPEED AES ENCRYPTION.....	35
4.1 INTRODUCTION.....	36
4.2 PROPOSED ARCHITECTURE FOR AES ENCRYPTION ALGORITHM.....	36
4.3 FPGA IMPLEMENTATION OF PROPOSED ARCHITECTURE OF AES.....	37
4.4 CONCLUSION.....	39
CHAPTER 5 FULL CUSTOM DESIGN FOR PROPOSED S-BOX.....	40
5.1 INTRODUCTION.....	41
5.2 NOVAL XOR GATE FOR LOW POWER FULL CUSTOM DESIGN OF S-BOX...41	
5.2.1 Conventional schematic for XOR gate and its simulated output.....	42
5.2.2 Proposed novel XOR schematic, layout and its simulated output.....	43
5.2.3 Comparison between Conventional and Proposed XOR Gate.....	44
5.3 SCHMATIC AND LAYOUT DESIGN FOR FIRST MODIFICATION OF PROPOSED ARCHITECTURE OF S-BOX.....	44
5.4 SCHMATIC AND LAYOUT OF PROPOSED MULTIPLICATIVE INVERSE (MI) IN GF (2^4).....	46

5.5 SCHEMATIC AND LAYOUT OF PROPOSED ARCHITECTURE OF MULTIPLICATION IN GF (2 ²).....	48
5.6 SCHEMATIC AND LAYOUT OF FOUR BIT XOR USING THE PROPOSED NOVEL XOR GATE.....	49
5.7 SHEMATIC OF PROPOSED S-BOX.....	50
5.8 LAYOUT AND POST LAYOUT SIMULATION RESULTS OF PROPOSED S-BOX.....	51
5.9 CONCLUSION.....	54
CHAPTER 6 CONCLUSION AND FUTURE WORK.....	55
6.1 CONCLUSION.....	56
6.2 FUTURE WORK.....	56
REFERENCES.....	57
PUBLICATIONS.....	60

LIST OF TABLES

Table I: Round key size and number of rounds in three versions of AES.....	12
Table II Subbytes Transformation Table.....	15
Table III: Inverse SubBytes Transformation Table.....	15
Table IV: Pre-computed results of the multiplicative inverse operation in GF (2^4).....	24
Table V: MI in GF (2^4).....	29
Table VI: Comparison of MI in GF (2^4) in ASIC.....	31
Table VII: Comparisons of multiplication in GF (2^2) in ASIC.....	32
Table VIII: FPGA Implementation Results and Comparisons In Xilinx Vertex-II Pro.....	33
Table IX: FPGA Implementation Results and Comparisons in Spartan6 (xc6slx16-3csg324).....	33
Table X: ASIC Implementation Results and Comparisons.....	34
Table XI: Design Summary of FPGA Implementation of proposed AES algorithm.....	38
Table XII: Comparison Results between Conventional and Proposed XOR.....	44
Table XIII: Comparison Results between Post layout and before Post layout simulation.....	54

LIST OF FIGURES

Figure 2-1 Basic step of Encryption in cryptography.....	10
Figure 2-2 Symmetric Key Cryptography.....	11
Figure 2-3 Rounds of AES Encryption algorithm.....	13
Figure 2-4 SubBytes Transformation.....	14
Figure 2-5 ShiftRows Transformation for AES Encryption.....	16
Figure 2-6 MixColumns transformation of AES Encryption.....	17
Figure 2-7 AddRoundKey transformation of AES Encryption.....	17
Figure 2-8 SubBytes and Inverse SubBytes transformation in composite field.....	18
Figure 2-9 The conventional S-box architecture in composite field.....	19
Figure 2-10 Meaning of symbols used in Figure 2-9.....	19
Figure 2-11 Logical hardware diagram of squarer for $GF(2^4)$	21
Figure 2-12 Logical hardware diagrams for multiplication with constant, λ	21
Figure 2-13 Logical hardware implementation of $GF(2^4)$ multiplier.....	22
Figure 2-14 Hardware implementation of multiplication with ϕ	23
Figure 2-15 Hardware implementation of $GF(2^2)$ multiplication.....	23
Figure 3-1 4:1 multiplexer for (a) LSB output and (b) MSB output for 2 bits output of multiplication in $GF(2^2)$	31
Figure 3-2 Proposed Multiplicative Inverse architecture.....	31
Figure 3-3 Hardware implementation Result of S-box obtained from XC2VP30 device using ChipScope pro logic analyzer.....	32
Figure 4-1 the proposed architecture of AES encryption algorithm.....	37

Figure 4-2 Simulation Output of proposed AES encryption for 128 bits in Xilinx ISE 13.4.....	38
Figure 5-1 Conventional Schematic for XOR gate.....	42
Figure 5-2 Simulated output of Conventional XOR.....	42
Figure 5-3 Proposed Schematic for novel XOR.....	43
Figure 5-4 Simulated output for proposed Schematic of novel XOR.....	43
Figure 5-5 Layout of proposed novel XOR gate.....	44
Figure 5-6 Schematic of the operator (op).....	45
Figure 5-7 Layout of the Schematic drawn in Fig.5-6.....	45
Figure 5-8 Schematic for Proposed Multiplicative Inverse (a) MSB as '0' (b) MSB as '1'....	46
Figure 5-9 Layouts for Proposed Multiplicative Inverse (a) MSB as '0' (b) MSB as '1'.....	47
Figure 5-10 Complete Schematic for Proposed Multiplicative Inverse in GF (2^4).....	47
Figure 5-11 Complete Layout of Proposed Multiplicative Inverse in GF (2^4).....	47
Figure 5-12 Schematic of proposed architecture of multiplication in GF (2^2).....	48
Figure 5-13 Layout of proposed architecture of multiplication in GF (2^2).....	48
Figure 5-14 Schematic of multiplication in GF (2^4).....	49
Figure 5-15 Layout of multiplication in GF (2^4).....	49
Figure 5-16 Schematic and Layout of 4 bits XOR (a) Schematic and (b) Layout of (a).....	50
Figure 5-17 Schematic of proposed S-BOX.....	50
Figure 5-18 Simulated output of Schematic of proposed architecture of S-BOX.....	51
Figure 5-19 the optimized layout of proposed S-BOX.....	52
Figure 5-20 DRC check of layout.....	52

Figure 5-21 LVS match of layout.....	52
Figure 5-22 Layout after parasitic extraction.....	53
Figure 5-23 Design summary and device count after parasitic extraction.....	53
Figure 5-24 Post layout simulated output after parasitic extraction.....	54

ABSTRACT

In the present era of information processing through computers and access of private information over the internet like bank account information even the transaction of money, business deal through video conferencing, encryption of the messages in various forms has become inevitable. There are mainly two types of encryption algorithms, a private key (also called symmetric key having a single key for encryption and decryption) and public key (separate key for encryption and decryption). In terms of computational complexity, private key algorithm is less complex than a public key algorithm. The simple architecture of private key algorithm attracts the VLSI implementation through the basic digital components like basic gates and flip-flops. Moreover, the high throughput architecture can be realized for encryption of very large amounts of data, e.g., images and videos, in real time. National Institute of Standards and Technology (NIST) adopted Advanced Encryption Standard (AES) as the standard for encryption and decryption of blocks of data. The draft is published under the name as FIPS-197 (Federal Information Processing Standard number 197). AES is a symmetric key block cipher. It encrypts data of block size 128 bits. The AES algorithm is used in diverse application fields like WWW servers, automated teller machines (ATMs), cellular phones and digital video recorders.

The AES is an iterative algorithm. It encrypts the data using four different transformations namely SubBytes, ShiftRows, MixColumns and AddRoundkeys. SubBytes transformation (also called substitution) is a non-linear operation in AES wherein each byte of a state is mapped to a different value. The SubBytes transformation is done through S-box and it is the most complex steps in terms of cost and implementation. Use of ROM table and composite field arithmetic are two techniques to perform substitutions. The ROM based approach requires high amount of memory and also it causes low latency and low throughput because of ROM access time. Composite field arithmetic is more suitable for S-box implementation of high speed AES encryption.

In the present work, hardware optimization for AES architecture has been done in different stages. The critical path delay in architectural path has been reduced using different logic components. For instance, there are a number of XOR gates and its combination in logic path of substitution block which uses Galois field (GF) arithmetic. The equation in GF has been restructured in order to have low delay components in the critical path. Moreover, the basic

components in GF arithmetic are also replaced with the digital components that can be realized using multiplexers. For the AES implementation, merging technique has been used wherein ShiftRows, MixColumns and AddRoundkeys transformations are performed in a single VLSI module. Two different architectures of AES, namely iterative and concurrent, have been implemented in Xilinx FPGA.

The hardware comparison results show that as AES architecture has critical path delay of 9.78 ns when conventional s-box is used, whereas it has a critical path delay of 8.17 ns using proposed s-box architecture. The total clock cycles required to encrypt 128 bits of data using a proposed AES architecture are 86 and therefore, throughput of the AES design in Spartan-6 of Xilinx FPGA is approximately 182.2 Mbits/s.

To achieve the very high speed, full custom design of the s - box in composite field has been done for the proposed s-box architecture in Cadence Virtuoso. The novel XOR gate is proposed for use in s-box design which is efficient in terms of delay and power along with high noise margin. The implementation has been done in 180 nm UMC technology. Total dynamic power in the proposed XOR gate is 0.63 μW as compared to 5.27 μW with the existing design of XOR. The designed s-box using proposed XOR occupies a total area of 27348 μm^2 . The s-box chip consumes 22.6 μW dynamic power and has 8.2 ns delay after post layout simulation has been performed.

CHAPTER 1

INTRODUCTION

In this chapter the motivation, research objectives and the organization of the thesis is presented.

1.1 MOTIVATION

With worldwide communication of private and confidential data over the computer networks or the Internet, there is always a possibility of threat to data confidentiality, data integrity and, also data availability. Data encryption maintains data confidentiality, integrity and authentication. Information has become of the most important assets in growing demand of need to store every single importance of events in everyday life. Messages need to be secured from unauthorized party. Encipherment is one of the security mechanisms to protect information from public access. Encryption hides the original content of a message so as to make it unreadable to anyone, except the person who has the special knowledge to read it.

In the past cryptography means only encryption and decryption using secret keys, nowadays it is defined in different mechanisms like asymmetric-key encipherment (public-key cryptography) and symmetric-key encipherment (called as private-key cryptography). The public key algorithm is complex and has very high computation time. Private Key algorithms involve only one key, both for encryption as well as decryption whereas, public key algorithms involve two keys, one for encryption and another for decryption. There were many cryptographic algorithms proposed such as Data Encryption Standard (DES), 2-DES, 3-DES, Elliptic Curve Cryptography (ECC), the Advanced Encryption Standard (AES) and other algorithms. Many investigators and hackers are always trying to break these algorithms using brute force and side channel attacks. Some attacks were successful as it was the case for the Data Encryption Standard (DES) in 1993[21].

The Advanced Encryption Standard (AES) is considered as one of the strongest published cryptographic algorithms. National Institute of Standards and Technology (NIST) adopted Advanced Encryption Standard (AES) as the standard for encryption and decryption of blocks of data after the failing of the Data Encryption Standard (DES). The draft is published under the name as FIPS-197 (Federal Information Processing Standard number 197)[5]. Moreover, it is used in many applications such as in ATM Machines, RFID cards, cellphones and large servers. AES is widely used for encryption of audio/video data contents in real time.

Due to the significance of the AES algorithm and the numerous real-time applications, the main concern of this thesis will be presenting new efficient hardware implementations for this algorithm.

AES algorithm is an iterative algorithm, which requires many computation cycles. A software platform cannot provide the high speed encryption of data, specially used for real-time applications. Audio/video content encryption is required in real-time in business deals via video conferencing. Therefore, dedicated hardware implementation is inevitable in such applications. Hardware implementation can be done through different architectures trading throughput with area and power consumption. At any time, designing best architecture for a particular design with low area and low latency is a challenge. Hardware implementations of the AES algorithm vary according to the application. While some applications require very high throughputs as in e-commerce servers, others require a medium throughput range as in designs for cell phones. Some others require very low area and low power implementations to be used application as RFID cards.

The AES is an iterative algorithm and uses four operations in different rounds, namely SubBytes, ShiftRows, MixColumns and Key Additions transformations. SubBytes transformation is done through S-box. S-box is the vital component in the AES architecture that decides the speed/throughput of the AES[1]. The ROM based approach requires high amount of memory and also it causes low latency because of ROM access time. Therefore, composite field arithmetic is more suitable for S-box (substitution) implementation its hardware optimization for VLSI implementation is very important to reduce the area and power of the AES architecture.

We have designed a custom S-box for AES encryption. Firstly the logic verification has been done by FPGA implementing using VHDL code of the S-box in Xilinx ISE and ASIC using 0.18 μm standard cell technology library. The optimization of the design has been done by proposing novel circuit for smaller components like XOR gate and other circuit components like Galois Field (GF) multiplier. The XOR has been designed using minimum number of transistors and it has high noise margin and low power consumption as compared to existing XOR designs.

The design optimization has been done by replacing conventional modules in AES architecture with a module which best suits for the area and latency reduction. Further, we have synthesized two different design styles of AES, namely, iterative and concurrent (pipeline) for implementation in Xilinx FPGA. Iterative architecture can be realized with low

area, but throughput is low as compared to concurrent architecture which has higher throughput.

1.2 LITERATURE REVIEW

The Advanced Encryption Standard (AES) algorithm has published by NIST as a draft FIPS-197 in 2001. There are numerous hardware implementations were suggested for it, among all the implementation mostly they have targeted the AES with 128-bits key size [1-3]. This key size is considered to be appropriate for most of the commercial applications, where using higher key sizes is considered as excess of resources. It involves higher area implementations with longer processing time and not easy to implement for small scale devices. Key sizes of 192-bit and 256 bits are used mostly in top secret military applications to confirm the maximum level of security.

1.2.1 High Speed AES Design

AES algorithm is an iterative algorithm, which requires many computation cycles. A software platform cannot provide the high speed encryption of data, specially used for real-time applications. Audio/video content encryption is required in real-time in business deals via video conferencing. Therefore, dedicated hardware implementation is inevitable in such applications.

Hardware implementation can be done through different architectures trading throughput with area and power consumption. The design optimization can be done by replacing conventional modules in AES architecture with a module which best suits for the area and latency reduction details in [13, 14].

Further, there are mainly two different design styles found and its implementation in different devices namely, iterative and concurrent (pipeline) for implementation in Xilinx FPGA [9, 10]. It has observed that concurrent implementation requires less time but the area is large with high power consumption. The transformations used in different rounds are same so, algorithm can be used repeatedly and area and power can save with the improvement in speed [13, 17]. The iterative implementation could be efficient as per the requirement published in [10].

1.2.2 Architecture and Implementation of S-Box

There are four transformations in the AES algorithm among all the transformation, SubBytes is complex and non-linear. There are two techniques found to implement S-BOX, one using RAM and another using composite field arithmetic architecture. The implementation of the composite field S-BOX is accomplished using combinational logic circuits rather than using pre-stored S-BOX values.

S-BOX substitution starts by finding the multiplicative inverse of the number in GF (2^8), and then applying the affine transformation. Implementing a circuit to find the multiplicative inverse in the finite field GF (2^8) is very complex and costly, therefore, [19] has suggested using the finite field GF (2^4) to find the multiplicative inverse of elements in the finite field GF (2^8). First detailed implementation of the composite field S-BOX was published in [16].

The S-Box is at the major of any AES implementation and is measured a full complexity design consuming the main portion of the power and energy inexpensive of the AES hardware. The substitute way is to design the S-Box circuit using combinational logic directly from its arithmetic operations. This method has a fine delay - path from S-Box processing. The AES algorithm can be implemented on a varied range of platforms under different constraints [2]. In transportable applications figuring resources are usually restricted and dedicated hardware implementation of the safety purpose is essential.

AES Implementation using FPGA (Field Programmable Gate Array) is not appropriate for such applications generally due to size and power limitations. A full-custom chip is more suitable for compact small foot-print design in such a case. The Galois Field arithmetic for S-Box, it is very clearly evident that the implementation of S-Box/InvS-Box needs a large number of XOR operations [15].

The novel XOR has been designed using minimum number of transistors and it has high noise margin and low power consumption as compared to existing XOR designs. The new approach to minimize the silicon - area of S-Box design demonstrated by using a new 2-input XOR gate for low-power composite field arithmetic to reduce the power dissipation and delays for the complete circuit [15].

1.3 RESEARCH OBJECTIVE

Based on the above discussion, the main objectives for efficient AES algorithm designs are:

1. To propose the high speed S-BOX and its implementation in FPGA and ASIC.
2. In Composite Field Arithmetic, XOR is used in addition so, XOR has been designed using minimum number of transistors and it has high noise margin and low power.
3. Full custom design of S-BOX for AES Encryption algorithm.
4. Implementation of high speed architecture of AES algorithm.

1.4 ORGANIZATION

This thesis is organized as follows:

Chapter 2 describes the AES algorithm in details. The four encryption steps are presented: Byte Substitution, Shift Rows, Mix Column and finally Add Round Key. It also describes the details of S-BOX implementation in Composite Field Arithmetic.

In Chapter 3, a proposed architecture of S-box with some modification in Conventional architecture is presented. The implementation results of proposed architecture in FPGA and ASIC with comparisons of previous work is also provided.

In Chapter 4, a high speed 128-bits pipelined and iterative AES encryption using new efficient merging technique is presented. The simulated output and comparisons with conventional works is also provided.

In Chapter 5, a proposed two inputs novel XOR gate has been presented with comparison of conventional XOR. The schematic and layout of all the require blocks and proposed S-BOX have been presented and comparison of post layout and before post layout simulation results have been done.

Finally, the conclusion and future work are presented in Chapter 6.

CHAPTER 2

ADVANCED

ENCRYPTION

STANDERD (AES)

ALGORITHM

The brief introduction of Advanced Encryption Standard (AES) and its steps has discussed in this chapter. Literature review of AES, S-BOX architecture and its implemented conventional hardware design is also present.

2.1 DEFINITION AND HISTORY OF CRYPTOGRAPHY

Information need to be secured from unauthorized party. Cryptography is one of the security mechanisms to protect information from public access. Cryptography is a Greek origin word which means “secret writing” to make the information secure and immune to attacks. Classic cryptography was used for top-secret communications between people. This kind of cryptography is commonly applied by replacing the message letters with other letters using definite formula. Nowadays it’s changed into algorithm based cryptography according to demand by the users. It has two processes encryption and decryption, in the first process encryption; the Plain text (Original message) will be converted into secured text or Ciphertext (Encrypted message) using a specific algorithm which is shown in Figure 2-1. The second process is decryption which is the reverse process of encryption; here Ciphertext will be converted into Plain text using all the inverse steps applied for encryption.

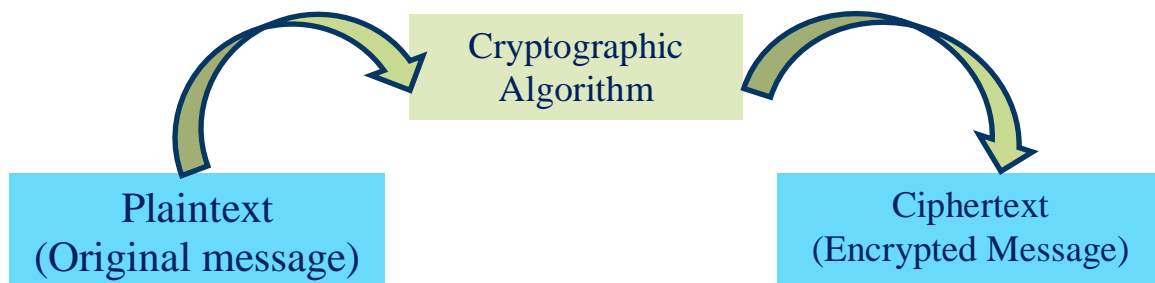


Figure 2-1 Basic step of Encryption in cryptography

There are mainly two types of encryption algorithms, a private key (also called symmetric key) and public key. Private Key algorithms involve only one key, both for encryption as well as decryption whereas; public key algorithms involve two keys, one for encryption and another for decryption.

The private Key algorithm is less complex and easy to implement for high speed application. Figure 2-2 illustrates the concept private key cryptography, an entity can send an encrypted message through insecure channels to another entity but the key should be sent from a secure channel to decrypt the message in Symmetric key cryptography. Advanced Encryption Standard (AES) is a private key algorithm [1, 3].

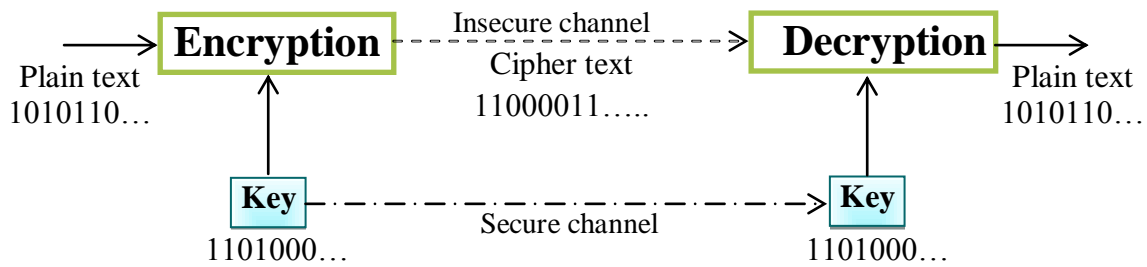


Figure 2-2 Symmetric Key Cryptography.

2.2 GALOIS FIELD

The Galois field (GF) or Finite Field with a finite number of elements are extensively used in cryptography. The total number of element present in GF is called the order of fields. A GF is of the form p^n , where n is a positive integer and p is a prime number also called the characteristic of the Galois field.

There are many cryptographic algorithms using GF among them, the AES algorithm uses the GF (2^8). The data byte can be represented using a polynomial representation of GF (2^8). Equation 2.1 shows the polynomial representation of data bytes in Finite Fields.

$$a(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0 \quad (2.1)$$

Arithmetic operation is totally different from normal arithmetic algebra; an addition can be found using bit-wise XOR operation. In Galois field, the multiplication product of polynomials will be modulo an irreducible polynomial so final answer can be within the used finite field. The polynomial which cannot be factorized of two or more than two is called as irreducible polynomial.

2.3 THE DATA ENCRYPTION STANDERED (DES)

National Institute of Standard and Technology (NIST) published a proposal from IBM in 1973 for symmetric key cryptosystem. DES was accepted and published in March 1975 as a draft of Federal Information Processing Standard (FIPS). It was finally published in January 1977 as FIPS 46 in the Federal Register.

DES is 64-bit cryptosystem, here 64-bit plain text takes and creates a 64-bit cipher text for the encryption process. Similarly, in decryption of a 64-bit cipher text taken to convert into

64-bit plain text. There is 56-bit same key has been used for both encryption and decryption. Round-key generator generates the different round key for each round.

The linear cryptanalysis attack could break the DES algorithm and made it unconfident algorithm. Several published brute force attacks started to fail DES algorithm. The NIST started looking for replacement of DES algorithm because of its failure.

The NIST specifications required 128 bits block size and three different key sizes of 128, 192 and 256 bits, should be an open algorithm. The NIST declared that **Rajndael** cipher was selected as *Advanced Encryption Standard* (AES).

2.4 THE ADVANCED ENCRYPTION STANDARD (AES)

The National Institute of Standards and Technology (NIST) announced that **Rajndael** pronounced as “**Rain Doll**” planned by two Belgium researchers Joan Daemen and Vincent Rijment was adopted as Advanced Encryption Standard (AES) for encryption and decryption of blocks of data. The draft is published in December 2001, under the name as FIPS-197 (Federal Information Processing Standard number 197).

The criteria defined by selecting AES fall into three areas Security, Implementation and cost of the algorithm. The main emphasis was the security of the algorithm to focus on resistance of cryptanalysis attacks, implementation cost should be less so it can be used for small devices like smart cards.

The AES algorithm is a private key block cipher. It encrypts data of block size 128 bits. It uses three key sizes, 128 bits, 192 bits and 256 bits in three versions. AES uses three different types of round operations. Table I shows the number of rounds in three versions of AES. But, in each version final round key is 128 bits.

Table I: Round key size and number of rounds in three versions of AES

Cipher Key size	No. Of Rounds (Nr)	Round Key size
128 bits	10	128 bits
192 bits	12	128 bits
256 bits	14	128 bits

The initialization is done by adding first round key (128 bits) with 128 bits plain text. In subsequent steps, the following transformations are done: SubBytes, ShiftRows, MixColumns and AddRoundKey. The last round is different from the previous rounds as there is no MixColumns transformation. Figure 1 shows the round in AES. The internal 128 bits data in AES are represented in the form of 4x4 square matrix containing elements of size 8 bits and named as state elements. The decryption process involves of the inverse steps, decryption round contains of: Inverse S-BOX used for Byte Substitution, Inverse Shift Rows, Add Round Key and Inverse Mix Columns. The round keys will be generated using a unit called the key generation unit. This unit will be generating 176, 208 or 240 bytes of round keys depending on the size of the used key.

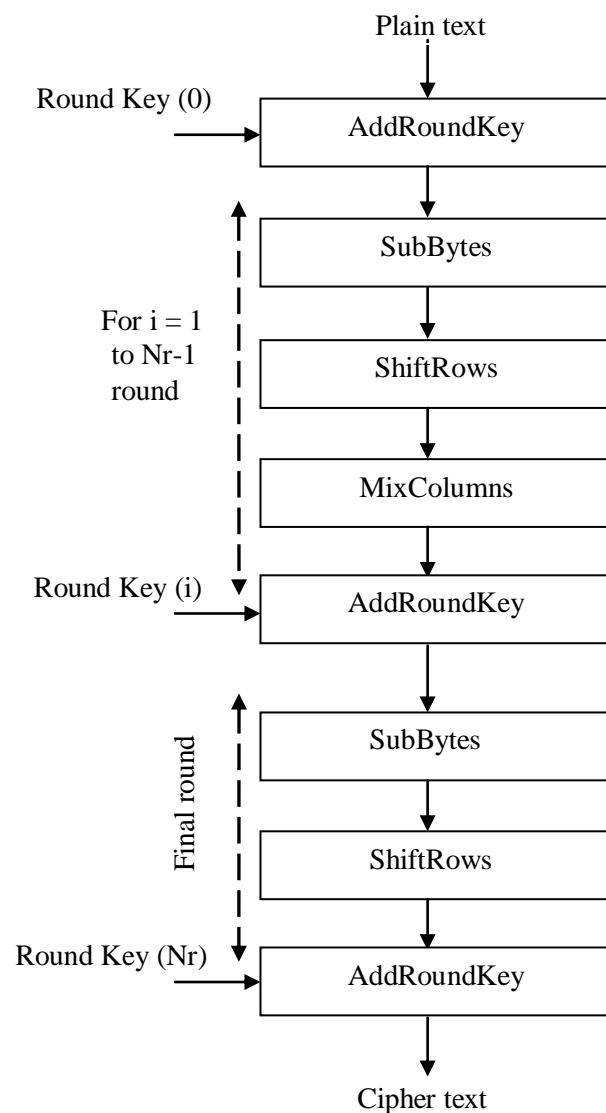


Figure 2-3 Rounds of AES Encryption algorithm.

2.4.1 SUBBYTES/INVERSE SUBBYTES TRANSFORMATION

The first transformation, SubBytes, is used for encryption and inverse SubBytes used for decryption. The SubBytes substitution is a nonlinear byte substitution that operates independently on each byte of the State using a substitution table (S-box). Take the multiplicative inverse in the finite field $GF(2^8)$ and affine transform to do the SubBytes transformation. Inverse affine transform have to find for inverse SubBytes transformation then multiplicative inverse of that byte.

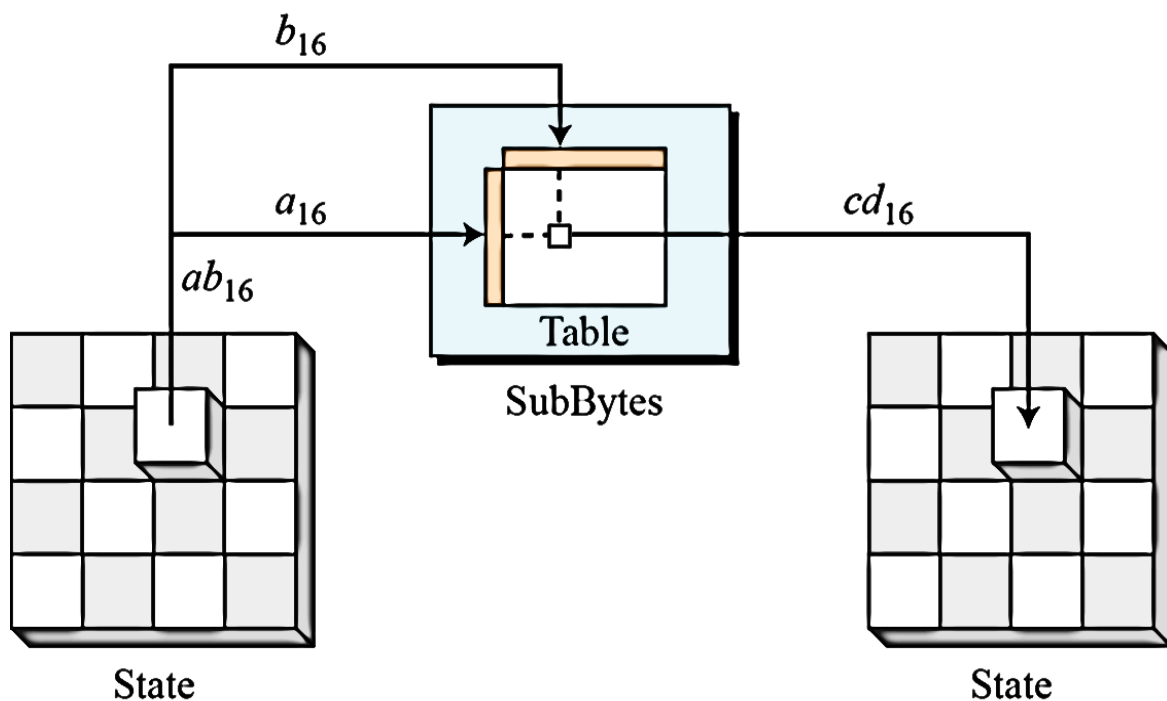


Figure 2-4 SubBytes Transformation [5].

Figure 2-4 indicates that how the transformation can be done. There are two hexadecimal digits a and b in one state element, the left digit (a) defines the row and the right digit (b) defines the column of the substitution table. The junction of these two digits is the new bytes.

Inverse SubBytes transformation is inverse of SubBytes transformation. It can find in the similar way only table which is used for mapping the byte is different. The SubBytes transformation is done through S-box. There are two techniques to perform substitutions, (i) using S-BOX table, and (ii) using composite field arithmetic. There are separate tables for SubBytes and its inverse; Table II is used for SubBytes transformation and Table III used for its inverse. It can be found using S-box architecture in composite field arithmetic which is discussed in the next chapter.

Table II: Subbytes Transformation Table [16]

		b															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
a	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table III: Inverse SubBytes Transformation Table [16]

		b															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
a	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

SubBytes table is also called as S-box and inverse SubBytes table is an Inverse S - box. There are two parts of affine transformation and its inverse; a constant matrix will be multiplied with the data in multiplication part, then the addition part, where a constant vector is added to multiplication result.

2.4.2 SHIFTRROWS/INVERSE SHIFTRROWS TRANSFORMATION

The transformation is called ShiftRows performs in encryption, in which rows are cyclic shifting to the left. The number of shifting depends upon the row number of the state matrix. First row no shifting, second row one byte, third row two bytes and fourth row three byte shifting left. In the decryption, InvShiftRows transformation performs the right cyclic shifting operation inverse of ShiftRows; number of shifting depends on number of row number. Figure 2-5 shows the Cyclic ShiftRows transformation for AES algorithm.

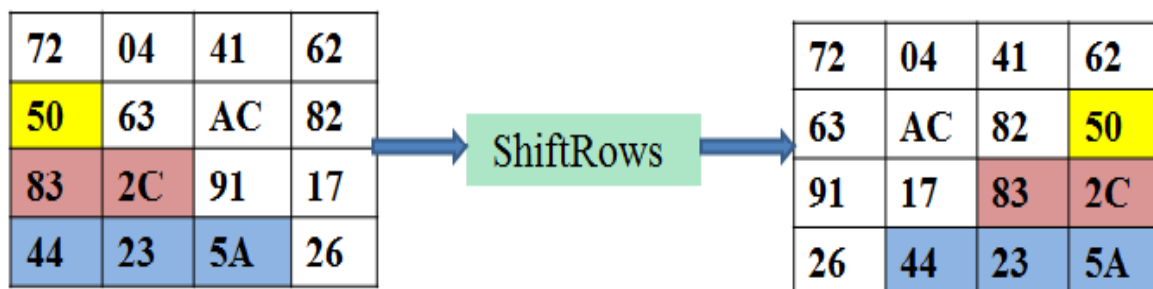


Figure 2-5 ShiftRows Transformation for AES Encryption.

2.4.3 MIXCOLUMNS/INVERSE MIXCOLUMNS TRANSFORMATION

The MixColumns transformation functions after the ShiftRows on the State column-by-column, considering each column as a four-term polynomial. Inverse MixColumns are the inverse process of MixColumns which is used in the decryption of cipher text. The columns are considered as polynomials over $GF(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $A(x)$, given in equation 2.2.

$$A(x) = \{03\} x^3 + \{01\} x^2 + \{01\} x + \{02\}. \quad (2.2)$$

The algorithm for MixColumns and Inverse MixColumns involves multiplication and addition in $GF(2^8)$. The MixColumns multiplies the rows of the constant matrix by a column in the state. Figure 2-6 describes the operation of this transformation; key addition is the next transformation of the encryption.

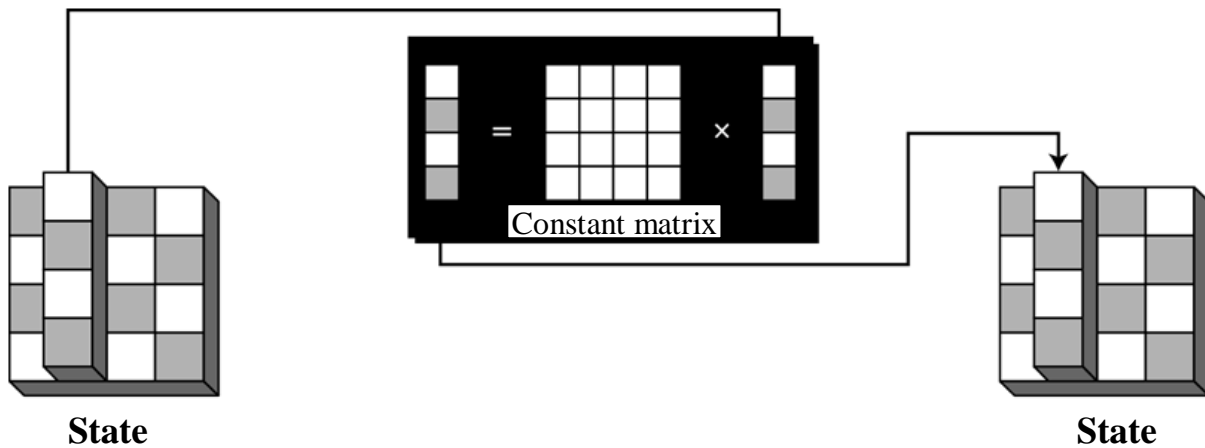


Figure 2-6 MixColumns transformation of AES Encryption.

2.4.4 ADDROUNDKEY TRANSFORMATION KEY EXPANSION

2.4.4.1 AddRoundKey Transformation

The AddRoundKey adds the round key word with each column of state matrix. It is similar to MixColumns; the AddRoundKey proceeds one column at a time. The most important in this transformation, that it includes the cipher key. The state column will get XOR with key which is generated by key generator and create another state as shown in Figure 2-7.

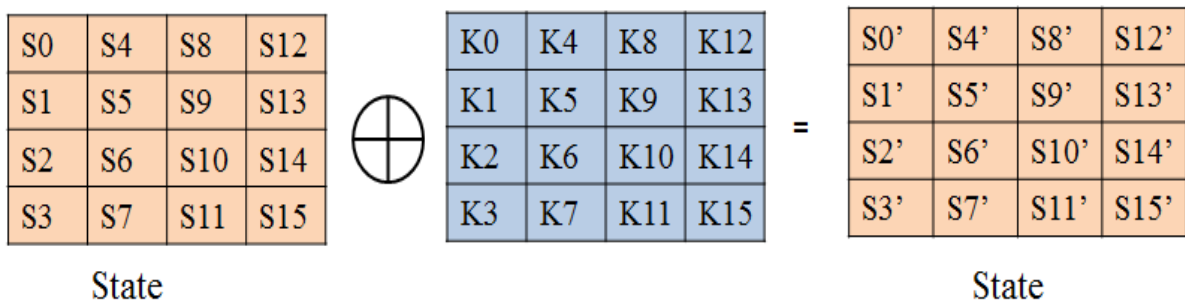


Figure 2-7 AddRoundKey transformation of AES Encryption.

2.4.4.2 Key Expansion

The key expansion term describes the operation of generating all Round Keys from the original input key. The initial round key is original key in case of encryption and in case of decryption the last group of the generated by key expansion keys will be original keys. As mentioned earlier this initial round key will be added to the input firstly before starting the encryption or decryption iterations. The 128 bits key size, 10 groups of round keys will be generated with 16 bytes size. The round keys are generated word by word. There are some similar encryption transformations used to generate the round key.

2.5 COMPOSITE FIELD ARITHMETIC S-BOX

The SubBytes transformation, done through S-BOX mapping is computationally inefficient when implemented using a ROM. But, it is not efficient for applications requiring very high throughput as ROM accessing involves one complete clock cycle for mapping one 8-bits state element and consequently 16 clock cycles are required to transform the 128 bits data (16 bytes). To increase the throughput, parallel ROMs are required resulting in large size of chip area.

Therefore, a more feasible solution is to implement an S - box is by using composite field arithmetic which uses only logic elements in the implementation. The S-BOX substitution starts by finding the multiplicative inverse of the data in $GF(2^8)$, and then applying the affine transformation. Figure 2-8 shows steps for the one byte forward and inverse SubBytes transformation using composite field arithmetic.

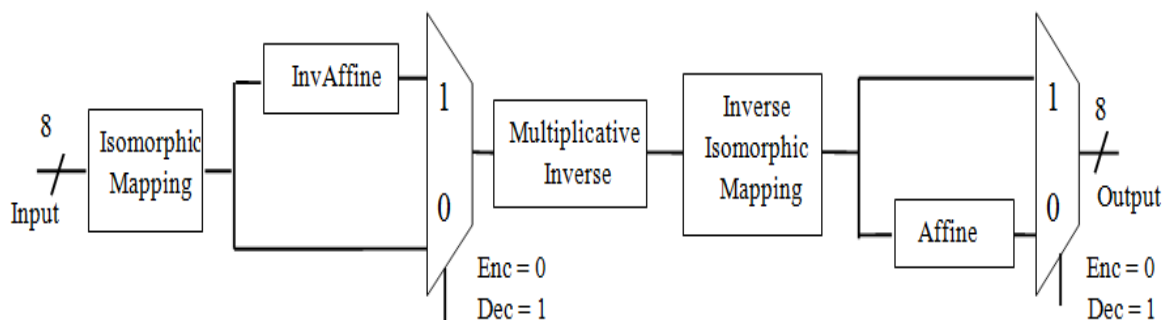


Figure 2-8 SubBytes and Inverse SubBytes transformation in composite field [6].

To find the S-BOX transformation first multiplicative inverse of $GF(2^8)$ then affine transformation calculated. Similarly, for InvSubBytes first InvAffine transformation then multiplicative inverse has to be calculated. There are one major operation involve here, which is to find the multiplicative inverse in $GF(2^8)$. This can be done by breaking the $GF(2^8)$ elements in $GF(2^4)$ and some more logical blocks. I.e., Any arbitrary polynomial in $GF(2^8)$ can be represented as $bx+c$ using an irreducible polynomial x^2+Ax+B . Here, b is the most significant nibble and c is the least significant nibble. The multiplicative inverse can be found by using the following equation (2.3).

$$\begin{aligned}
 & (bx + c)^{-1} \\
 & = b(b^2B + bcA + c^2)^{-1}x + (c + bA)(b^2B + bcA + c^2)^{-1} \\
 & = b(b^2\lambda + c(b + c))^{-1}x + (c + b)(b^2\lambda + c(b + c))^{-1}
 \end{aligned} \tag{2.3}$$

Where, $A=1$, $B=\lambda$, as the irreducible polynomial used is $x^2+x+\lambda$. Figure 2.9 shows the block diagram to find the multiplicative inverse in $GF(2^8)$ using $GF(2^4)$. Figure 2.10 shows the meanings the symbols used in Figure 2.9. The mapping structure in different fields along with the irreducible polynomials is as follows.

$$\begin{aligned}
 GF(2^2) &\rightarrow GF(2) && : x^2 + x + 1 \\
 GF((2^2)^2) &\rightarrow GF(2^2) && : x^2 + x + \phi \\
 GF(((2^2)^2)^2) &\rightarrow GF((2^2)^2) && : x^2 + x + \lambda
 \end{aligned}
 \tag{2.4}$$

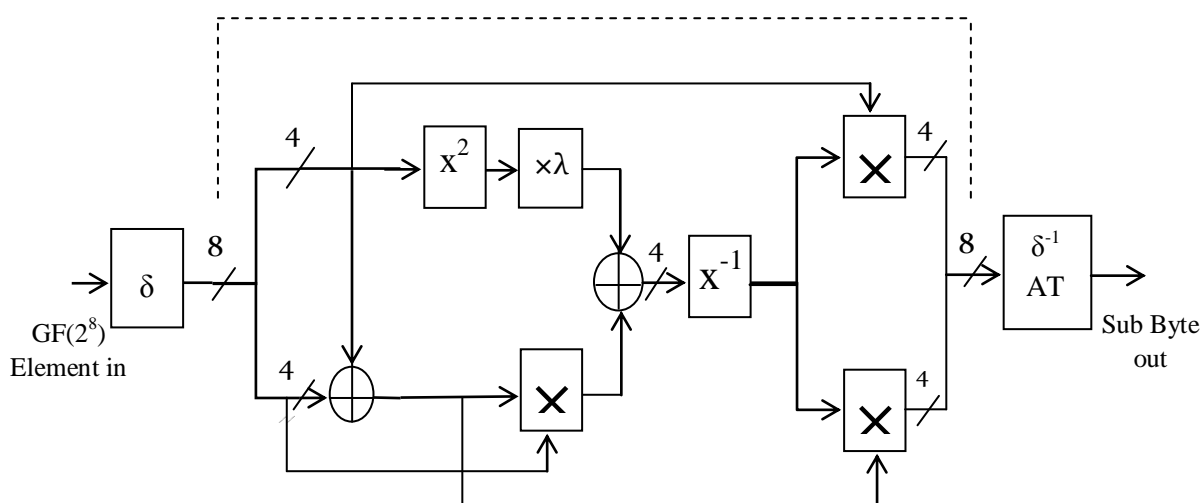


Figure 2-9 The conventional S-box architecture in composite field [6].

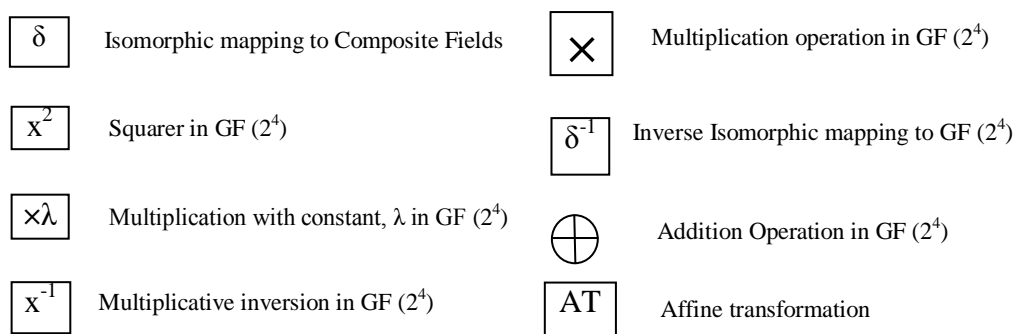


Figure 2-10 Meaning of symbols used in Figure 2-9.

Isomorphic mapping is the first step performed on the 8 bits sub byte input. The output of the isomorphic mapping is given to the input of multiplicative inverse (MI) module. Subsequently, inverse isomorphic mapping and affine transformations are the steps that follow. The detailed discussion of each block has given below.

2.5.1 Addition operation in GF (2⁴)

The addition operation in Galois Field can be interpreted to simple bitwise XOR operation between the two elements.

2.5.2 Squaring operation in GF (2⁴)

The squaring operation of 4 bits, i.e. x^2 term can be modulo reduced using the irreducible polynomial from (2.4), $x^2 + x + \varphi$. It can reduce into lower order of Galois field, by setting $x^2 = x + \varphi$ and replacing it into x^2 . Doing above operation GF (2⁴) is converted into GF (2²), here nibble is converted into 2 bit stream. It can be represented using equation 2.5.

$$k = q_h^2 x + q_l^2$$

$$k = q_h^2(x + \varphi) + q_l^2 \quad (2.5)$$

Where $k \{k_3 k_2 k_1 k_0\}$ is the four bits output of squarer and $q \{q_3 q_2 q_1 q_0\}$ is the input bit steam, here q_h , k_h , q_l , and k_l are higher 2 bits of q and k and lower 2 bits of q and k respectively. Now GF (2²) can be converted into GF (2), the x^2 term can be replaced $x^2 = x + 1$. For the case of x^3 , it can be obtained by multiplying x^2 by x , i.e. $x^3 = x(x) + x = x^2 + x$. after Substituting for x^2 , $x^3 = x + 1 + x$. The two x terms are presented which cancel out each other, leaving only $x^3 = 1$. Performing all the substitution output bit stream can be calculated by input bit streams in GF (2) the final expression yields the following equation 2.6.

$$k_h = q_3^2 x^2 + q_3 q_2 x + q_3 q_2 x + q_2^2$$

$$k_l = q_3 x^2 + q_2 x + q_1 x^2 + q_0 \quad (2.6)$$

Here we know that similar term in XORing operation will get cancelled, after that polynomial substitution has to do which is discussed earlier. Equation 2.7 gives the logical expression for all the output bits.

$$k_3 = q_3$$

$$k_2 = q_3 \oplus q_2$$

$$k_1 = q_2 \oplus q_1$$

$$k_0 = q_3 \oplus q_1 \oplus q_0 \quad (2.7)$$

The equation (2.7) can be realized hardware logic diagram using an XOR operation and diagram drawn in figure 2.11.

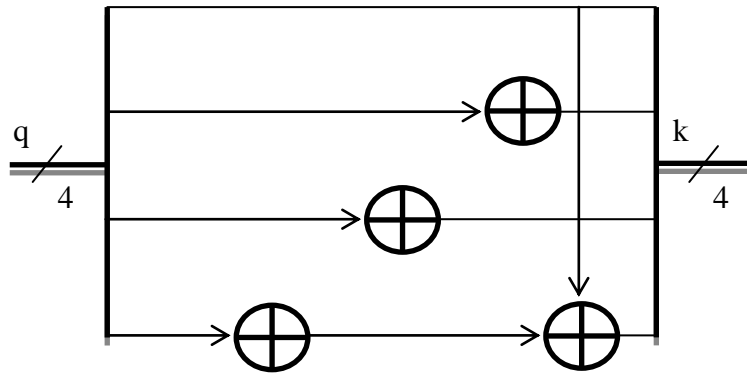


Figure 2-11 Logical hardware diagram of squarer for GF (2⁴)

2.5.3 Multiplication with constant, λ

The multiplication with constant λ , which value is {1100} in GF (2⁴) will give the polynomials. Modulo reduction can be performed by substituting $x^2 = x + \phi$ using the irreducible polynomial in (2.4) to yield the logical expression. The final output bits k in the form of input bits q can be calculated using irreducible polynomial, which represented in equation 2.8. There are total three XOR gate is required to implement the multiplication with λ . There are two XOR gates in critical path which will give the maximum delay. Figure 2-12 shows the logical hardware of equation 2.8.

$$\begin{aligned}
 k_3 &= q_2 \oplus q_0 \\
 k_2 &= q_3 \oplus q_2 \oplus q_1 \oplus q_0 \\
 k_1 &= q_3 \\
 k_0 &= q_2
 \end{aligned}
 \tag{2.8}$$

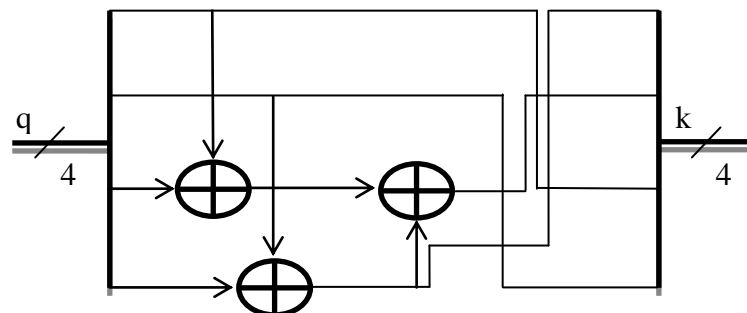


Figure 2-12 Logical hardware diagrams for multiplication with constant, λ

2.5.4 Galois field GF (2⁴) multiplication

The GF (2⁴) multiplier is a major component to find the multiplicative inverse using composite field arithmetic operation. It requires more hardware to implement in combinational logic. It is multiples of 4 bits with 4 bits and results also in 4 bits. Let k = qw, where k in the 4 bits binary output and q and w are 4 bits inputs. It can be observed that multiplication and addition operation in GF (2²), multiplication in GF (2²) is a major component in it.

It can be converted into a lower form of Galois field using irreducible polynomial present in equation 2.4. The final expression of logic implementation can be represented in equation 2.9. Figure 2-13 shows the logical hardware implementation of GF (2⁴) multiplication. Here ‘+’ represents the XOR operation.

$$k = k_h x + k_l$$

$$k = (q_h w_h + q_h w_l + q_l w_h)x + q_h w_h \phi + q_l w_l \quad (2.9)$$

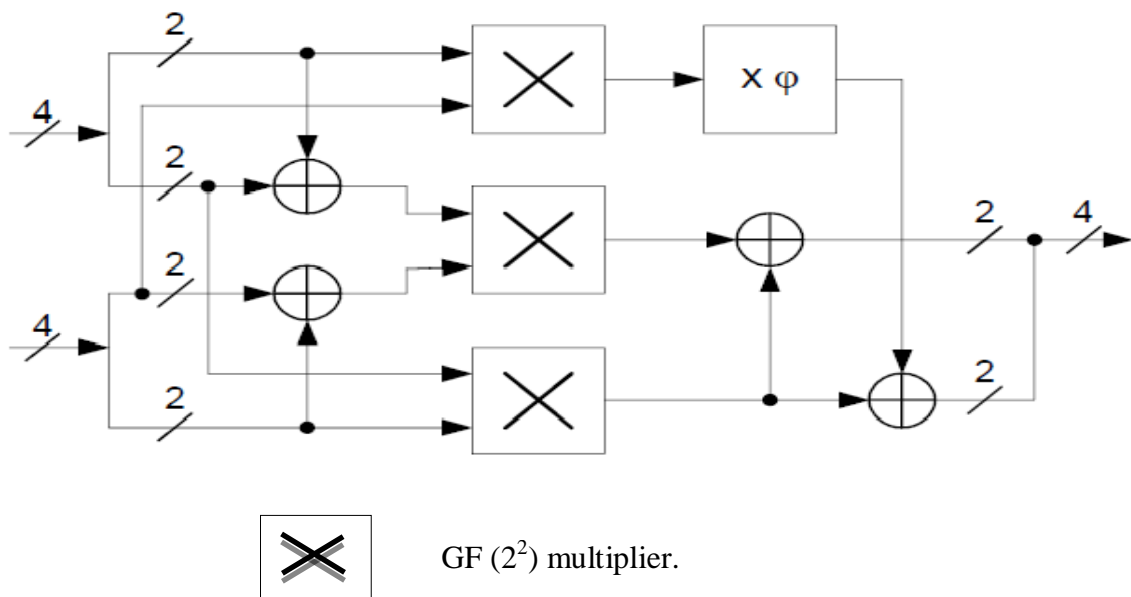


Figure 2-13 Logical hardware implementation of GF (2⁴) multiplier

2.5.4.1 Multiplication with constant, φ

The multiplication with constant φ, which has a constant value φ = {10}₂ is an element of GF (2²). It has two bits value that can be also represented in the form of combinational logic in equation 2.10. Figure 2-14 shows the hardware implementation of combinational logic. Here k is a two bits output and q two bits input of the component.

$$k_1 = q_1 \oplus q_0$$

$$k_0 = q_1 \tag{2.10}$$

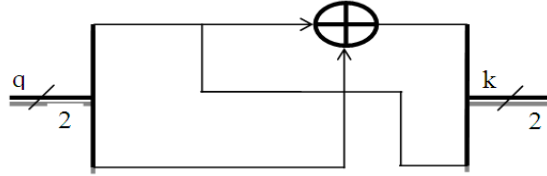


Figure 2-14 Hardware implementation of multiplication with ϕ

2.5.4.2 Galois field $GF(2^2)$ multiplication

The Galois field (2^2) multiplier is the major component in $GF(2^4)$ multiplication, which exist in the critical path. It can be represented in the input bit streams by using irreducible polynomial presents in equation 2.4.

It also implemented using combinational logic which presents in equation 2.11. Figure 2-15 shows its hardware implementation in composite field arithmetic. Here k is two bits output and q and w are the two bits input of the component.

$$k_1 = q_1 w_1 \oplus q_0 w_1 \oplus q_1 w_0$$

$$k_0 = q_1 w_1 \oplus q_0 w_1 \tag{2.11}$$

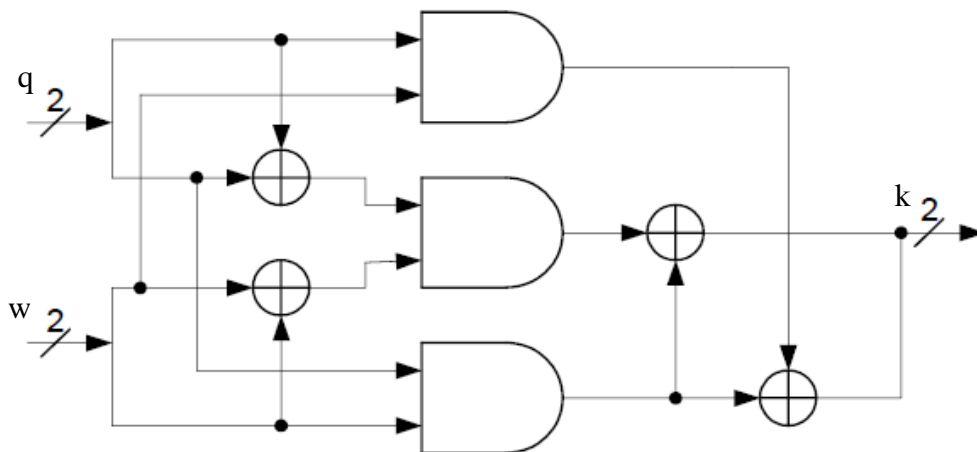


Figure 2-15 Hardware implementation of $GF(2^2)$ multiplication

2.5.5 Multiplicative Inversion in GF (2⁴)

The multiplicative inverse of q (where q is an element of GF (2⁴)) is an intermediate component of multiplicative inverse. It has derived a formula to compute the multiplicative inverse of q , such that $q^{-1} = \{q_3^{-1}, q_2^{-1}, q_1^{-1}, q_0^{-1}\}$. The inverses of the individual bits can be computed from the logical equation and pre-computed value can be stored in RAM. The pre-computed value can be seen in table IV which will be used to find the multiplicative inverse. The method using logic equation is given in equation 2.12.

Table IV: Pre-computed results of the multiplicative inverse operation in GF (2⁴) [16].

q	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
q^{-1}	0	1	3	2	f	c	9	b	a	6	8	7	5	e	d	4

The table containing the results of the multiplicative inverse in hexadecimal is shown above. The equation given below helped to hardware implementation in combinational logic where ‘+’ indicates the XOR operation.

$$\begin{aligned}
 q_3^{-1} &= q_3 + q_3q_2q_1 + q_3q_0 + q_2 \\
 q_2^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_0 + q_2 + q_2q_1 \\
 q_1^{-1} &= q_3 + q_3q_2q_1 + q_3q_1q_0 + q_2q_0 + q_2 + q_1 \\
 q_0^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_1 + q_3q_1q_0 + q_3q_0 \\
 &\quad + q_2 + q_2q_1 + q_2q_1q_0 + q_1 + q_0
 \end{aligned} \tag{2.12}$$

CHAPTER 3

IMPLEMENTATION OF PROPOSED ARCHITECTURE FOR S-BOX

3.1 INTRODUCTION

The SubBytes transformation is a non-linear operation in AES wherein each byte of a state is mapped to a different value. The SubBytes transformation is done through S-box. There are two techniques to perform substitutions, (i) using ROM table [10–12], and (ii) using composite field arithmetic [13–15]. The SubBytes transformation, done through S-box mapping is computationally inefficient when implemented using a ROM. But, it is not efficient for applications requiring very high throughput as ROM accessing involves one complete clock cycle for mapping one 8-bits state element and consequently 16 clock cycles are required to transform the 128 bits data (16 bytes).

To increase the throughput, parallel ROMs are required resulting in large size of chip area. Therefore, a more feasible solution is to implement an S - box is by using composite field arithmetic which uses only logic elements in the implementation. Substitution is the most complex steps in terms of cost and implementation [13]. Therefore, its hardware optimization for VLSI implementation is very important to reduce the area and power of the AES architecture. The ROM based approach requires high amount of memory and also it causes low latency because of ROM access time. Therefore, composite field arithmetic is more suitable for S-box (substitution) implementation.

The Speed improvement along with an area reduction has been the most challenging research in VLSI implementation. We propose a high speed VLSI architecture for S-box. The FPGA implementation of the architecture is done along with comparison with some existing transformation techniques. The proposed architecture has delayed improvement and low power consumption. The silicon validation of the architecture is done by programming XC2VP30 device on Virtex-II Pro FPGA board. The proposed architecture is also implemented in ASIC using 0.18 μm standard cell technology library.

3.2 PROPOSED ARCHITECTURE OF S-BOX.

The new architecture of S-BOX has proposed after 3 modifications in conventional architecture of S-BOX.

- I. Introduced an operator (op) after merging of some blocks.
- II. Implementation of multiplicative inverse in $\text{GF}(2^4)$ using multiplexor.
- III. Reduced the critical path of multiplication in $\text{GF}(2^2)$

3.2.1 Introduced an operator (op) after merging of some blocks.

An operator (op) has introduced after merging of blocks like squarer, multiplication with constant λ , a GF (2^4) multiplier and a four bits XOR. The equation of op has introduced using Galois field irreducible conversion technique and in the form of an input bit stream. One major operation involve here is finding the multiplicative inverse in GF (2^8).

This can be done by breaking the GF (2^8) elements in GF (2^4). I.e., Any arbitrary polynomial in GF (2^8) can be represented as $bx+c$ using an irreducible polynomial x^2+Ax+B . Here, b is the most significant nibble and c is the least significant nibble. The multiplicative inverse can be found by using the following expression [16].

$$\begin{aligned} & (bx+c)^{-1} \\ & = b(b^2B+bcA+c^2)^{-1}x+(c+bA)(b^2B+bcA+c^2)^{-1} \\ & = b(b^2\lambda+c(b+c))^{-1}x+(c+b)(b^2\lambda+c(b+c))^{-1} \end{aligned} \quad (1)$$

Where, $A=1$, $B=\lambda$, as the irreducible polynomial used is $x^2+x+\lambda$. Figure 3 shows the block diagram to find the multiplicative inverse in GF (2^8) using GF (2^4) [16]. The mapping structure in different fields along with the irreducible polynomials is as follows.

$$\begin{aligned} \text{GF}(2^2) & \rightarrow \text{GF}(2) & : x^2+x+1 \\ \text{GF}((2^2)^2) & \rightarrow \text{GF}(2^2) & : x^2+x+\varphi \\ \text{GF}(((2^2)^2)^2) & \rightarrow \text{GF}((2^2)^2) & : x^2+x+\lambda \end{aligned} \quad (2)$$

The expression $b^2\lambda+c(b+c)$ in Eq. (1) Can be written as,

$$b^2\lambda+bc+c^2 = b(b\lambda+c)+c^2 \quad (3)$$

Representing b , c and λ as,

$$b = b_Hx + b_L, \quad c = c_Hx + c_L, \quad \lambda = \lambda_Hx + \lambda_L$$

Where b_H and b_L are the upper and lower 2-bits of b , similarly, c_H and c_L are the upper and lower 2-bits of c and λ_H and λ_L are the upper and lower 2-bits of λ . The $b\lambda$, i.e., Multiplication with λ can be written as,

$$b\lambda = b_H\lambda_Hx^2 + b_L\lambda_Hx \quad (4)$$

And therefore,

$$b\lambda + c = b_H\lambda_Hx^2 + b_L\lambda_Hx + c_Hx + c_L \quad (5)$$

Now from Eq. (5),

$$\begin{aligned} & b(b\lambda + c) \\ &= (b_H\lambda_Hx^2 + b_H\lambda_H + c_Hx + c_L) \times (b_Hx + b_L) \\ &= (b_H^2\lambda_H\varphi + b_H^2\lambda_H + c_Hb_H + c_Lb_H + b_L^2\lambda_H + b_Lc_H)x \\ &+ (b_Lc_L + b_H^2\lambda_H\varphi + b_Hc_H\varphi) \end{aligned} \quad (6)$$

Here, $\lambda = (1100)_2$ and $\varphi = (10)_2$. Performing operations in GF $((2^2)^2)$, the following value can be obtained in terms of upper (b) lower nibble (c) bits of inputs. We can reduce the blocks in the proposed architecture from its conventional architecture of S-BOX.

$$\begin{aligned} & op(0) \\ &= b(0)c(0) + b(1)c(1) + b(2)c(3) + b(3)c(3) \\ &+ b(3)c(2) + in(0) + in(3) + in(5) + in(7) \\ & op(1) \\ &= b(0)c(1) + b(1)c(0) + b(1)c(1) + b(2)c(2) \\ &+ b(2)c(3) + b(3)c(2) + in(2) + in(3) + in(5) + in(6) \\ & op(2) \\ &= b(0)c(2) + b(1)c(3) + b(2)c(0) + b(2)c(2) \\ &+ b(3)c(1) + b(3)c(3) + in(1) + in(2) + in(4) + in(6) \\ & op(3) \\ &= b(3)(in(0) + in(3) + in(6)) + c(3)(b(0) + b(1) + b(2)) \\ &+ b(2)c(1) + b(1)c(2) + in(1) + in(3) + in(4) \end{aligned} \quad (7)$$

Where, in(0), in(1), in(2), in(3), in(4), in(5), in(6) are the input bits in the isomorphic mapping module (δ).

3.2.2 Implementation of multiplicative inverse in GF (2^4) using multiplexor.

MI in GF (2^4) represented by the symbol x^{-1} and multiplication in GF (2^4) are the two main components falls in the critical path of the design. MI in GF (2^4) consists of complex logic given by [12].

$$\begin{aligned}
q_3^{-1} &= q_3 + q_3q_2q_1 + q_3q_0 + q_2 \\
q_2^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_0 + q_2 + q_2q_1 \\
q_1^{-1} &= q_3 + q_3q_2q_1 + q_3q_1q_0 + q_2q_0 + q_2 + q_1 \\
q_0^{-1} &= q_3q_2q_1 + q_3q_2q_0 + q_3q_1 + q_3q_1q_0 + q_3q_0 \\
&\quad + q_2 + q_2q_1 + q_2q_1q_0 + q_1 + q_0
\end{aligned} \tag{8}$$

Where, $q_3^{-1}q_2^{-1}q_1^{-1}q_0^{-1}$ is 4-bits MI of 4-bit value $q_3q_2q_1q_0$ and + sign indicates XOR operation. It is evident that the realization of MI in GF (2^4) requires a number of exclusive-or (XOR) gates. By eliminating the XOR gates, delay and area can be reduced. Table V shows the input and output combination of MI in GF (2^4). The input combinations can be divided into two equal halves.

In the first half, MSB will have value '0' and in the second half, MSB will be '1'. This can be realized by a multiplexer, wherein, for '0' MSB in input, the 4-bit output will be given by a combination of three input bits (except MSB). Similarly, for MSB= '1', the 4-bit output will have 3-bit input combination (except MSB). The combinational logic of the first half and second half can be represented in terms of three input bits.

Table V: MI in GF (2^4) [16]

Input to MI in GF (2^4)	Output from MI in GF (2^4)	
$q_3q_2q_1q_0$	$q_3^{-1}q_2^{-1}q_1^{-1}q_0^{-1}$	
0000	0000	First half
0001	0001	
0010	0011	
0011	0010	
0100	1111	
0101	1100	
0110	1001	
0111	1011	
1000	1010	Second half
1001	0110	
1010	1000	
1011	0111	
1100	0101	
1101	1110	
1110	1101	
1111	0100	

By using a multiplexer, one of the outputs, can be selected depending on the MSB of the input. It is obvious that the combinational logic contains in equation (9a) and (9b) only OR and AND gates instead of XOR gates.

$$\begin{aligned}
q_3^{-1} &= (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \text{ or } (q_2 q_1 \bar{q}_0) \text{ or } (q_2 q_1 q_0) \\
q_2^{-1} &= (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \\
q_1^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \\
q_0^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \\
&\quad \text{or } (q_2 q_1 q_0)
\end{aligned} \tag{9a}$$

$$\begin{aligned}
q_3^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \\
q_2^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (q_2 \bar{q}_1 \bar{q}_0) \text{ or } (q_2 \bar{q}_1 q_0) \\
&\quad \text{or } (q_2 q_1 \bar{q}_0) \text{ or } (q_2 q_1 q_0) \\
q_1^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (\bar{q}_2 q_1 \bar{q}_0) \text{ or } (\bar{q}_2 q_1 q_0) \\
q_0^{-1} &= (\bar{q}_2 \bar{q}_1 \bar{q}_0) \text{ or } (\bar{q}_2 \bar{q}_1 q_0) \text{ or } (q_2 q_1 \bar{q}_0)
\end{aligned} \tag{9b}$$

3.2.3 Reduced the critical path of multiplication in GF (2²)

The Figure 2-15 shows the conventional architecture of multiplication in GF (2²). It is evident that there are two XOR gates and one AND gate in the critical path of GF (2²) multiplication. The output equation can be written as in equation (10).

$$\begin{aligned}
z(0) &= x(1)y(1) \oplus x(0)y(0) \\
z(1) &= x(1)y(1) \oplus x(0)y(1) \oplus x(1)y(0)
\end{aligned} \tag{10}$$

The above equation can be implemented using two 4:1 parallel multiplexers as follows. Suppose y is the select line. Then, for different values of y, multiplication result z, from Eq. (10), will have values as given in TABLE III. Figure 3-1 shows the proposed architecture of multiplier using two multiplexers.

One XOR gate has been eliminated from the critical path by using 4:1 multiplexer, i.e., there is one XOR gate and one multiplexer only in critical path, as compared to two XOR gates in conventional. Here, x and y are the 2 two bits inputs and z is the two bits output of the multiplication in GF (2²).

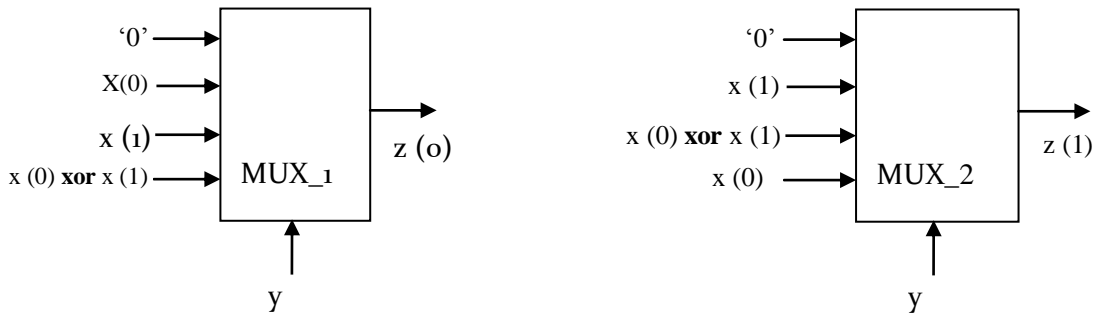


Figure 3-1 4:1 multiplexer for (a) LSB output and (b) MSB output for 2 bits output of multiplication in GF (2²)

3.3 IMPLIMENTATION OF PROPOSED ARCHITECTURE OF S-BOX

Figure 3-2 shows the proposed architecture of S-box for AES has been implemented in Xilinx FPGA and 180nm ASIC. The device taken for the implementation is XC2VP30 on Virtex-II pro board. For the comparison, S-box architecture in [16] has also been implemented using multiplicative inverse structure in the XC2VP30 device.

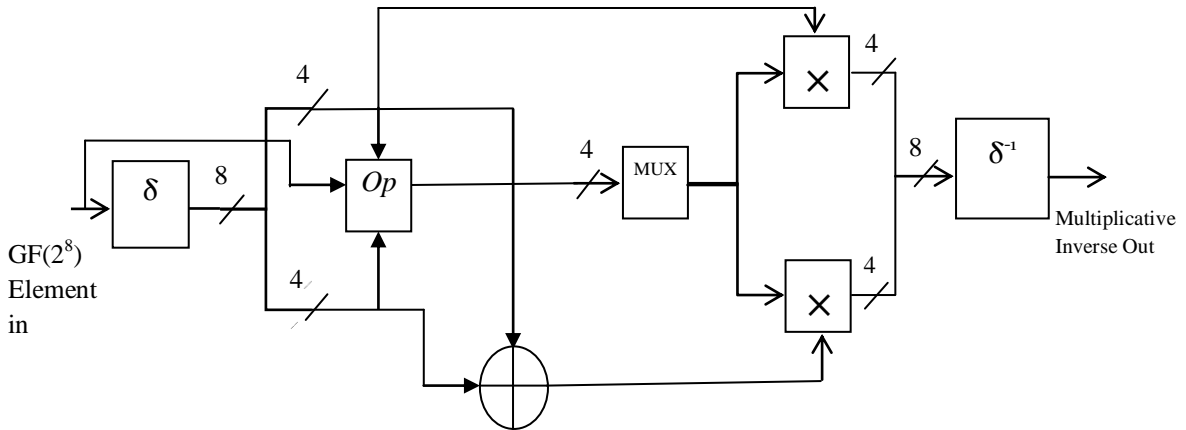


Figure 3-2 Proposed Multiplicative Inverse architecture

3.3.1 ASIC Implementation of MI in GF (2⁴)

The proposed multiplicative inverse implementation has been done in ASIC design. Table VI shows the ASIC implemented results and comparisons. It is evident that the proposed methods are delay and area efficient. The total dynamic power is also less.

Table VI: Comparison of MI in GF (2⁴) in ASIC

Technology 0.18 μm	Conventional structure	Proposed structure
Area (μm^2)	352	279.41
Total Dynamic Power (μW)	97.58	62.93
Delay (ns)	0.79	0.52

3.3.2 ASIC Implementation of multiplication in GF (2²)

The proposed architecture of multiplication in GF (2²) has implemented in ASIC. Table VII shows the speed and power improvement as compare with conventional architecture.

Table VII: Comparisons of multiplication in GF (2²) in ASIC

Technology 0.18 μm	Conventional structure	Proposed structure
Area (μm ²)	123.00	126.40
Total Dynamic Power (μW)	28.64	24.45
Delay (ns)	0.41	0.23

3.3.3 ASIC and FPGA Implementation of proposed S-BOX

The conventional S-box as well as proposed S-box architectures has been implemented using VHDL code. The device taken for the implementation is XC2VP30 on Virtex-II pro board. The proposed architecture has also implemented in Spartan 6 for the comparison, S-box architecture in [16] has also been implemented using multiplicative inverse structure in the XC2VP30 device.

The design synthesis has been done in different Xilinx devices. Table VIII shows the hardware utilization summary in terms of Slices and LUTs. Power consumption has been measured by Xpower analyser tool in ISE 10.1. From Table VIII, it is evident that there is an improvement in terms of delay and power consumption in the proposed structure as compared to the structure in [16].

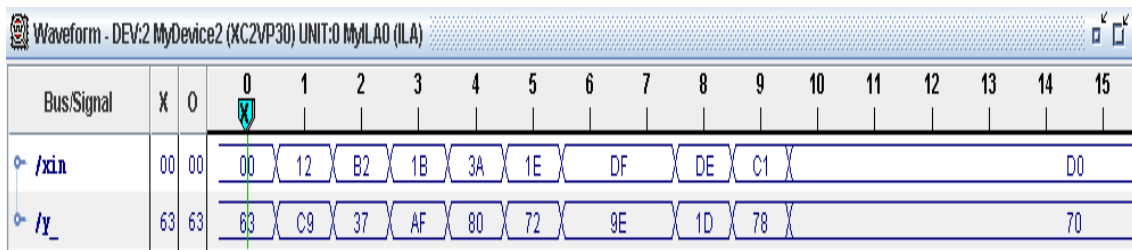


Figure 3-3 Hardware implementation Result of S-box obtained from XC2VP30 device using ChipScope pro logic analyzer

It can be seen that there is considerable area improvement in terms of FPGA slices, and speed improvement (the critical path delay) in our proposed method as compared to conventional architectures of S-box. Table IX shows the comparison results using a Spartan 6. The large improvement can be seen after implementing in SPARTAN6 (XC6SLX16-3CSG324). The improvement in terms of slices, LUTs and delay is also optimized. As compared to Xilinx vertex-II pro cost is less of this device so, it is useful for low utilization hardware implementation.

Figure 3.3 shows the sample outputs obtained from FPGA through the ChipScope pro logic analyzer after programming the XC2VP30 device. Table X shows the delay, power and area comparison in ASIC implementation using 180nm standard cell technology library. The proposed architecture has delay improvement of about 0.9 ns (=16 %) as compared to the conventional s-box architecture in [16] with little area cost.

TABLE VIII: FPGA IMPLEMENTATION RESULTS AND COMPARISONS IN XILINX VERTEX-II PRO

	Proposed structure	Structure in [16]	Structure in [14]
Device	XC2VP30	XC2VP30	XC2V1000
# of Slices	36	48	153
# of 4-input LUTs	63	85	NA
Max. Delay (ns)	15.0	15.6	10.82
Total Power (W)	7.27	9.74	NA

TABLE IX: FPGA IMPLEMENTATION RESULTS AND COMPARISONS IN SPARTAN6 (XC6SLX16-3CSG324)

	Proposed Structure	Conventional Structure
# of Slices	24	30
# of 4-input LUTs	54	81
Max. Delay (ns)	15.23	15.63
Total Power (mW)	20	20

TABLE X: ASIC IMPLEMENTATION RESULTS AND COMPARISONS

Technology 0.18 μm	Proposed Structure	Conventional Structure in [16]
Area (μm^2)	3968	3715
Gate Counts	404.89	379.08
Delay	4.6 (Improvement =16 % from conventional)	5.51
Total Dynamic Power (mW)	2.4985	2.4380
Efficiency (kbps/μm^2) (Throughput/Area)	438.25	390.57

3.4 CONCLUSION

An optimized architecture of S-box for AES encryption is proposed in this thesis. This novel architecture is implemented both in ASIC as well as FPGA. The ASIC implementation indicates speed improvement compared to conventional structure while maintaining area constant. FPGA implementation shows improvement in delay and area while a significant enhancement in terms of power compared to conventional architecture.

CHAPTER 4

**HIGH SPEED AES
ENCRYPTION**

4.1 INTRODUCTION

The data encryption maintains data confidentiality, integrity and authentication. AES (Advanced Encryption Standard) is a standard for encryption and decryption of blocks of data. It is adopted by the National Institute of Standards and Technology (NIST) and published under the name as FIPS-197 (Federal Information Processing Standard number 197). AES is widely used for encryption of audio/video data contents, data on smart cards, automated teller machines (ATMs), WWW servers, Network traffic, cellular phones, etc.

AES algorithm is an iterative algorithm, which requires many computation cycles. A software platform cannot provide the high speed encryption of data, specially used for real-time applications. Audio/video content encryption is required in real-time in business deals via video conferencing. Therefore, dedicated hardware implementation is inevitable in such applications. Hardware implementation can be done through different architectures trading throughput with area and power consumption. At any time, designing best architecture for a particular design with low area and low latency is a challenge.

We have designed the architecture of AES encryption which has low latency and low power consumption. The design optimization has been done by replacing conventional modules in AES architecture with a module which best suits for the area and latency reduction. Further, we have synthesized two different design styles of AES, namely, iterative and concurrent (pipeline) for implementation in Xilinx FPGA. Iterative architecture can be realized with low area, but throughput is low as compared to concurrent architecture which has higher throughput. Different implementation and hardware synthesis report have been presented.

4.2 PROPOSED ARCHITECTURE FOR AES ENCRYPTION ALGORITHM

The AES algorithm encrypts data in blocks of 128 bits. It uses three key sizes, 128 bits, 192 bits and 256 bits in three versions with three different N_r rounds 10, 12 and 14. But, in each version final round key is 128 bits. 128 bits cipher key size is used in first version with 10 rounds of transformations. With initial Roundkey addition transformation, there are 9 rounds of SubBytes, ShiftRows, MixColumns followed by AddRoundKeys transformations. The last round (10th round) is different from the previous rounds as there is no MixColumns transformation.

Figure 2-3 shows the rounds in AES of conventional architecture. The design optimization has been done by replacing conventional modules of AES architecture with a module which best suits for the area and high speed. Figure 4-1 shows our proposed architecture, in which ShiftRows and AddRoundKeys are merged in MixColumns transformation module. It means that these three transformations can be done using single clock cycle.

The proposed architecture of S-BOX with all three modifications (which discussed in the previous chapter) have used for SubBytes transformation in the proposed architecture of AES encryption algorithm. Iterative architecture can be realized with low area and proposed architecture helps to raise the speed.

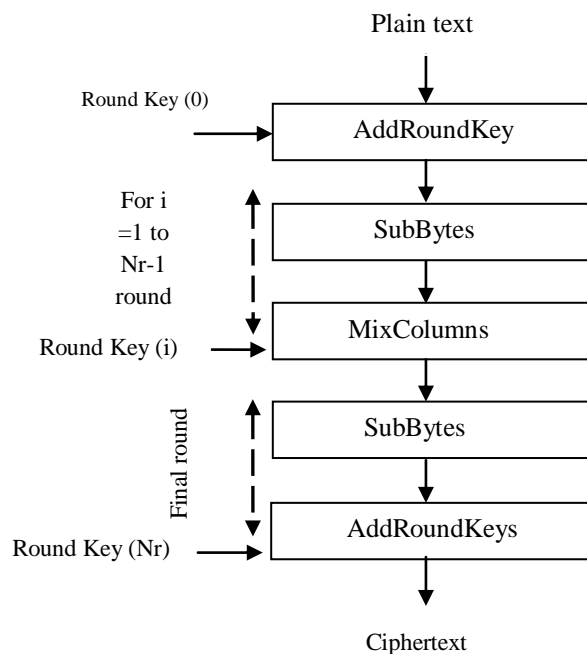


Figure 4-1 the proposed architecture of AES encryption algorithm

4.3 FPGA IMPLEMENTATION OF PROPOSED ARCHITECTURE OF AES

The conventional S-box as well as proposed S-box architectures has been implemented using VHDL code. The design synthesis has been done in different Xilinx devices. Table XI shows the device utilization summary of the proposed designs along with the conventional ones. It can be seen that there is considerable area improvement in terms of FPGA slices, and speed improvement (the critical path delay) in our proposed method as compared to conventional architectures of S-box and AES encryption algorithm.

The implementation of pipeline method takes more hardware. Therefore, it is not possible to implement the pipeline architecture in Spartan devices. The AES iterative design in Spartan6 FPGA takes 1 clock cycle in SubBytes transformation, 6 clock cycles in combine in AddRoundKeys, MixColumns and ShiftRows transformations for single round.

Therefore for 10 rounds, it takes $10 \times (6+1) + 16 = 86$ clock cycles. The maximum delay in proposed AES iterative design is 8.17 ns. So, it will take $(86 \times 8.17 = 702.62)$ ns to complete the AES transformation of 128 bits data. Figure 4-2 shows the simulation result of AES iterative architecture for 128 bits plaintext data.

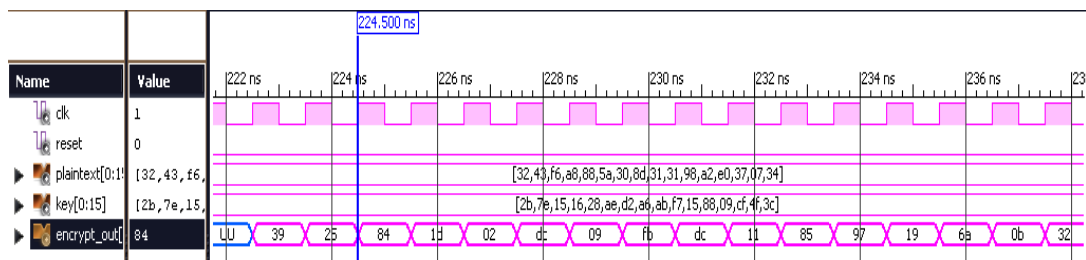


Figure 4-2 Simulation Output of proposed AES encryption for 128 bits in Xilinx ISE
13.4

Table XI: Design Summary of FPGA Implementation of proposed AES algorithm

Xilinx FPGA Device	Spartan6 (xc6slx16-3csg324)				Virtex-II pro (xc2vp30)
	S-box		AES Iterative		AES Pipeline
Design	With conventional logic	With proposed logic	With conventional logic (% utilization)	With proposed logic (% utilization)	With proposed logic (% utilization)
No. Of Slices	30	24	1017 (44 %)	730 (32 %)	9942 (72 %)
No. Of LUTs	81	54	2741 (30 %)	1838 (20 %)	18597 (67 %)
No. Of Slice Registers	0	0	952 (5 %)	788 (4 %)	5388 (19 %)
Delay	15.63 ns	15.23 ns	9.78 ns	8.17 ns	8.58 ns
Power	20 mW	20 mW	21 mW	21 mW	13.86 W

4.4 CONCLUSION

In this chapter, new hardware architectures for the Advanced Encryption Standard (AES) algorithm were presented. FPGA Xilinx technology was used to synthesise the designs and provide post placement results using Xilinx ISE 10.1 for AES pipeline architecture and Xilinx ISE 13.4 has used for the AES iterative algorithm. The maximum throughput of the design is 185.815 Mbits/s. Medium resolution video (640x480) of true colour depth (24 bits per pixel) has a bit rate of 184.3 Mbits/s.

Therefore, the proposed architecture implementation in spartan6 FPGA has enough throughputs to encrypt the video resolution mentioned above in real time. Because, the proposed iterative design has low area, it is suitable for the implementation in small devices like, smart cards, cellular phones, etc.

CHAPTER 5

**FULL CUSTOM
DESIGN FOR
PROPOSED S-BOX**

5.1 INTRODUCTION

The S-Box (Substitution box) forms the core building block of any hardware implementation of the Advanced Encryption Standard (AES) algorithm. This chapter presents a full custom CMOS design of S-Box with low power and high speed GF (2^8) Galois Field inversions based on polynomial basis, using composite field arithmetic. Field Programmable Gate Array (FPGA) implementation is not suitable for such applications mainly due to size and power constraints. It's difficult to achieve highly compact implementation using FPGA implementation.

The proposed architecture shows that XOR is the major component which is used to do the addition operation in composite field arithmetic. The optimization of the design has been done by proposing novel circuit for smaller components like XOR gate and other circuit components like Galois Field (GF) multiplier. The XOR has been designed using minimum number of transistors and it has high noise margin and low power consumption as compared to existing XOR designs. The full custom design is required for small devices like smart cards and high rate of data transmission.

5.2 NOVAL XOR GATE FOR LOW POWER FULL CUSTOM DESIGN OF S-BOX

It is clearly evident that the implementation of S-Box requires a large number of XOR operations whose efficient and low power implementation can result in a significantly improved CMOS S-Box hardware design. The numerous 2-input XOR gate designs have been described to enhance the performance for several applications. A XOR gate using six transistors including an inverter [15], is simulated in Cadence Spectre using UMC 180nm technology. The simulated output can be seen in Figure 5-2. The conventional XOR gate (Figure 5-1) has problem in two sets of input for that it's not giving proper output.

The novel XOR has been designed using minimum number of transistors. The pass transistor concept is used to design proposed XOR gate can be seen in Figure 5-3. A novel XOR has simulated in same technology and improvement can be seen in Figure 5-4. It has a high noise margin and low power consumption as compared to conventional XOR gate designs. The new approach to minimize the silicon - area of S-Box design demonstrated by using a new 2-input XOR gate for low-power composite field arithmetic to reduce the power dissipation and delays for the complete circuit.

5.2.1 Conventional schematic for XOR gate and its simulated output.

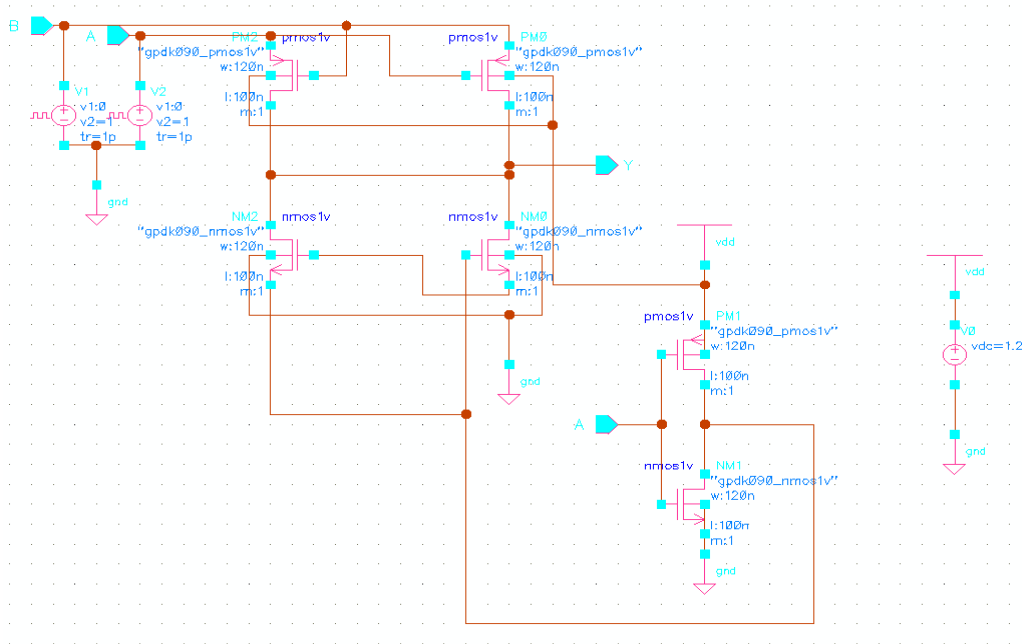


Figure 5-1 Conventional Schematic for XOR gate [15]

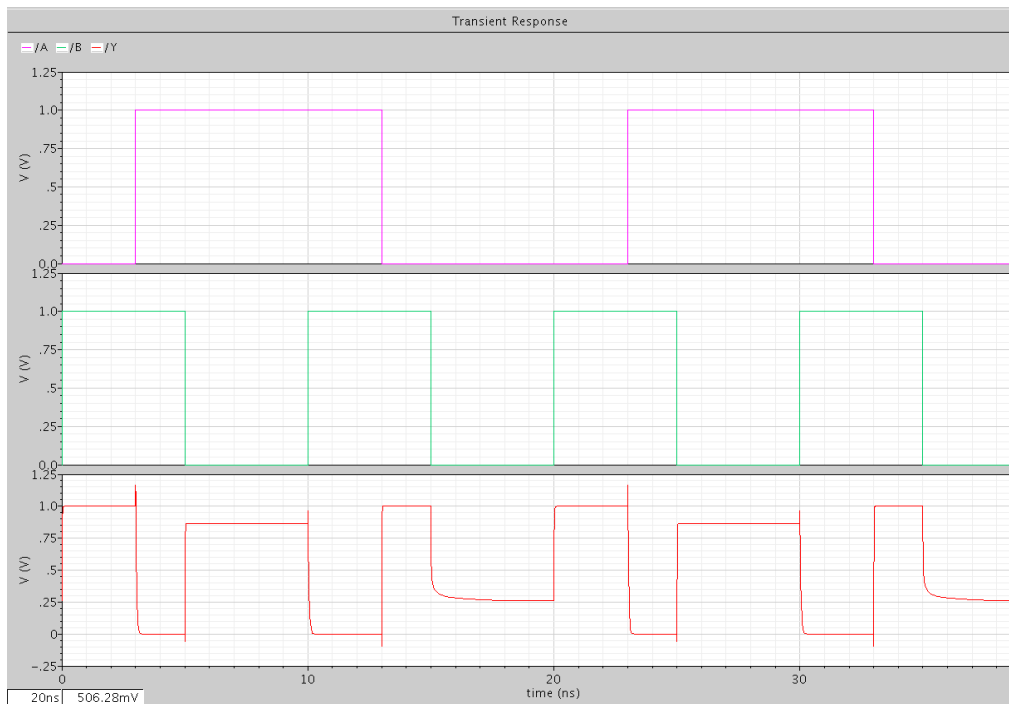


Figure 5-2 Simulated output of Conventional XOR.

The conventional XOR design has been simulated in spectre simulator of Cadence and can see the error present in Figure 5-2. Whenever input A is zero level PMOS transistor should pass the input B, and output Y should on proper level. But, it's giving the error of 0.3V.

5.2.2 Proposed novel XOR schematic, layout and its simulated output

A new novel XOR has proposed in Figure 5-3 and the simulated output can be seen in Figure 5-4. Figure 5-5 shows the layout of the proposed novel XOR which has drawn using Virtuoso Layout Editor of Cadence. This gives the proper level for all the sequences of inputs.

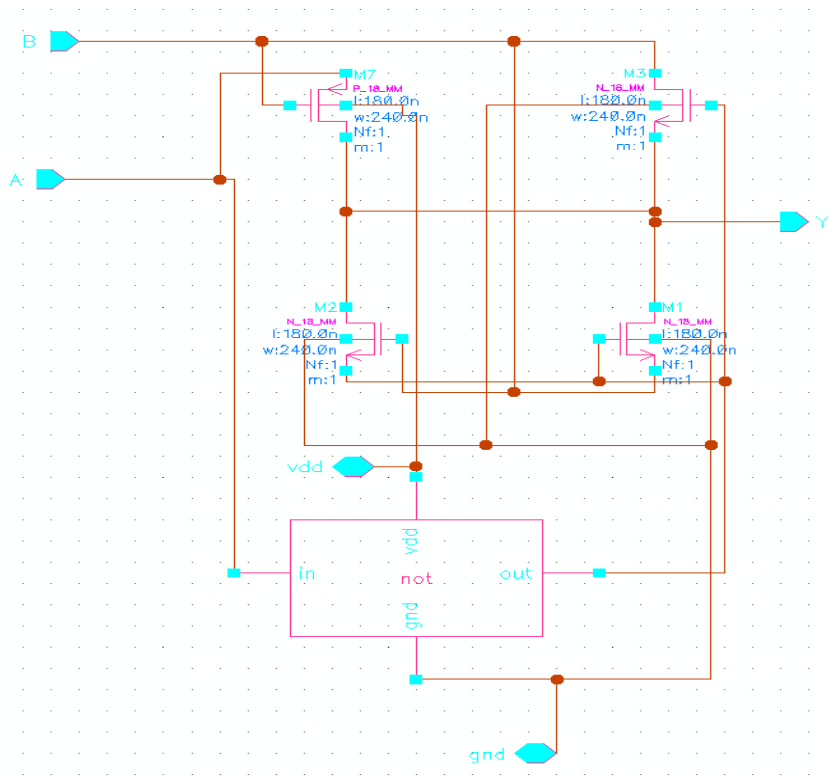


Figure 5-3 Proposed Schematic for novel XOR

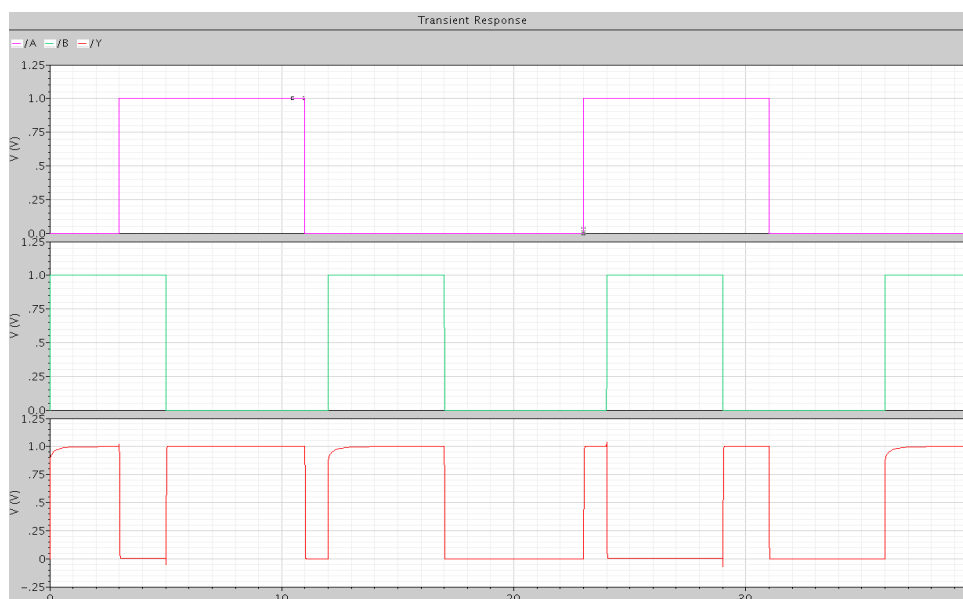


Figure 5-4 Simulated output for proposed Schematic of novel XOR

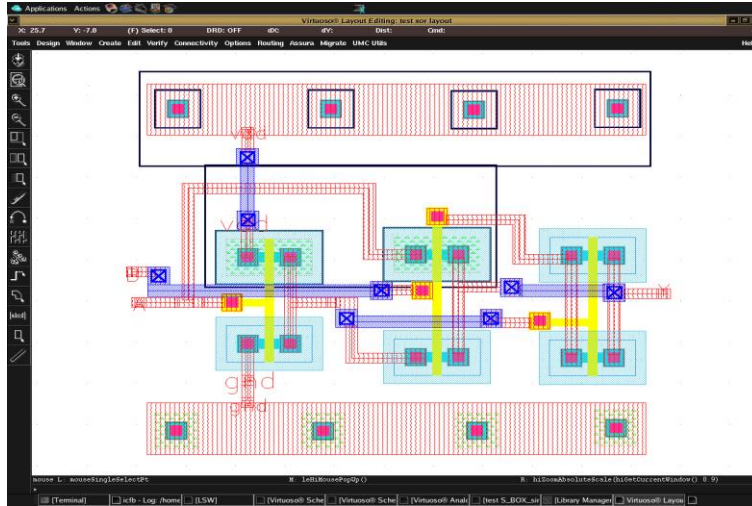


Figure 5-5 Layout of proposed novel XOR gate.

5.2.3 Comparison between Conventional and Proposed XOR Gate.

Table XII: Comparison Results between Conventional and Proposed XOR

	Proposed XOR	Conventional XOR
Maximum Delay (ns)	0.061	73
Total Average Dynamic Power (μw)	0.63	5.27

5.3 SCHEMATIC AND LAYOUT DESIGN FOR FIRST MODIFICATION OF PROPOSED ARCHITECTURE OF S-BOX

Equation 7 can be implemented in form of Schematic in Cadence spectre and layout of operator (op) has done using Virtuoso Layout Editor. All the bits of operator (op) have implemented in FPGA and verified the output of proposed S - Boxes, schematic of the architecture in Figure 5-6 and an optimized layout can be seen in Figure 5-7. The proposed novel XOR has used to implement the schematic and layout of operator (op) presents in equation 7.

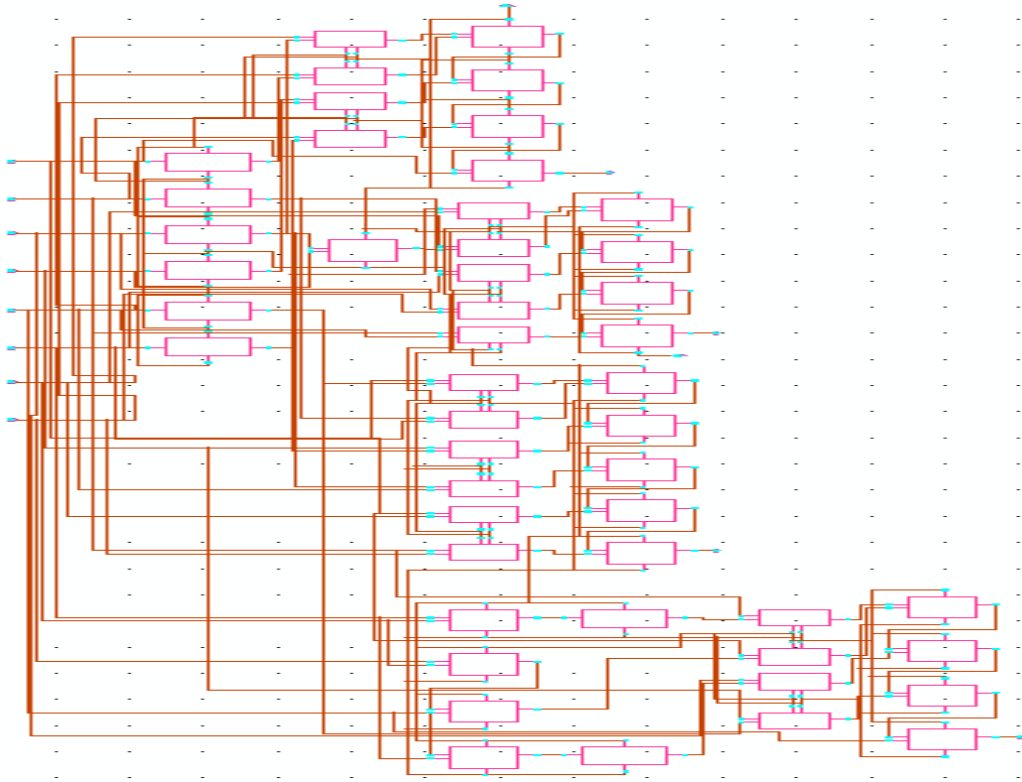


Figure 5-6 Schematic of the operator (op)

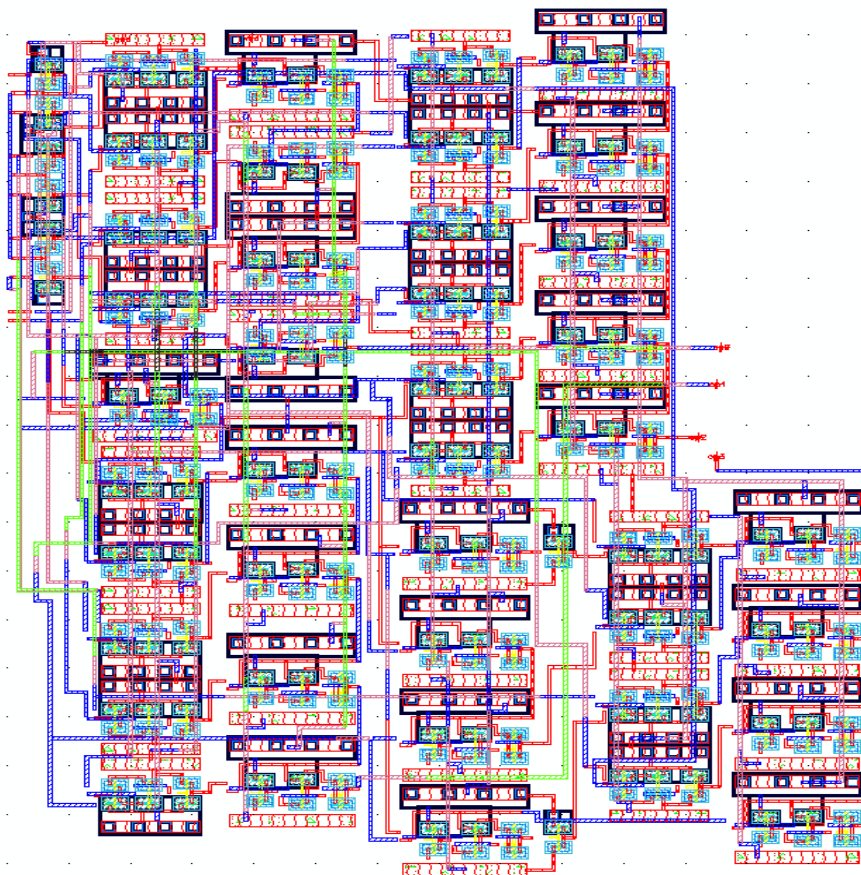


Figure 5-7 Layout of the Schematic drawn in Fig.5-6

5.4 SCHEMATIC AND LAYOUT OF PROPOSED MULTIPLICATIVE INVERSE (MI) IN GF (2^4)

The Schematic and layout has done, Cadence spectra used for Schematic and layout has drawn using the Virtuoso layout editor. There are two parts of MI one when taken MSB as ‘0’ and another for MSB as ‘1’ shown in Figure 5-8. Figure 5-8 (a) for MSB ‘0’ and (b) for MSB ‘1’ and complete Schematic is in Figure 5-10.

The layout of the multiplicative inverse (MI) is drawn partially wise first for the MSB ‘0’ and then of MSB ‘1’; the implementation equation is given in equation 9. Where the equation (9a) is for MSB ‘0’; and (9b) is for MSB ‘1’. The layout of the proposed architecture of MI has drawn, which is presented in Figure 5-9 (a) Layout for MSB ‘0’ , (b) Layout for MSB ‘1’. The complete optimized layout of MI has drawn in Figure 5-11.

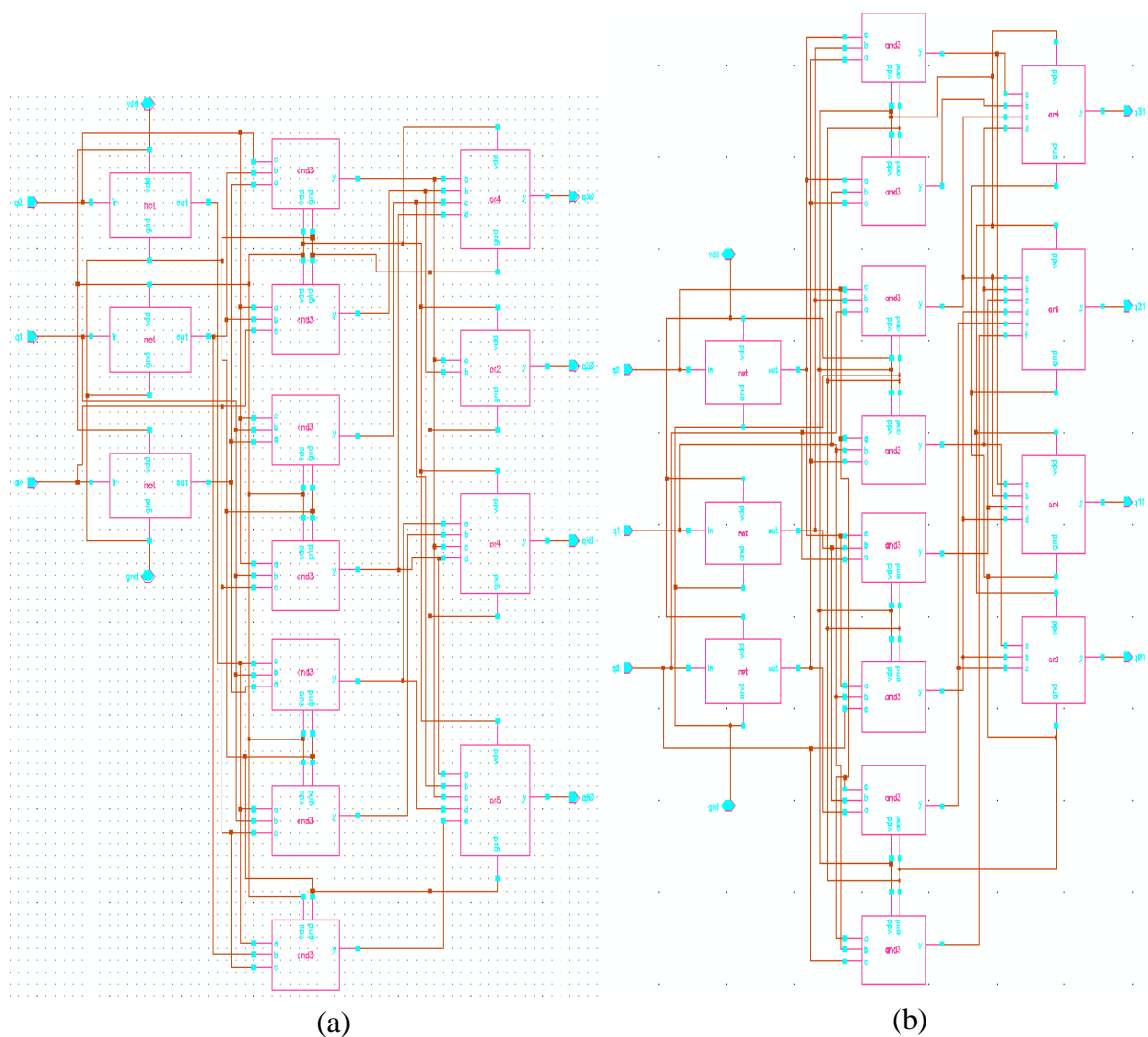
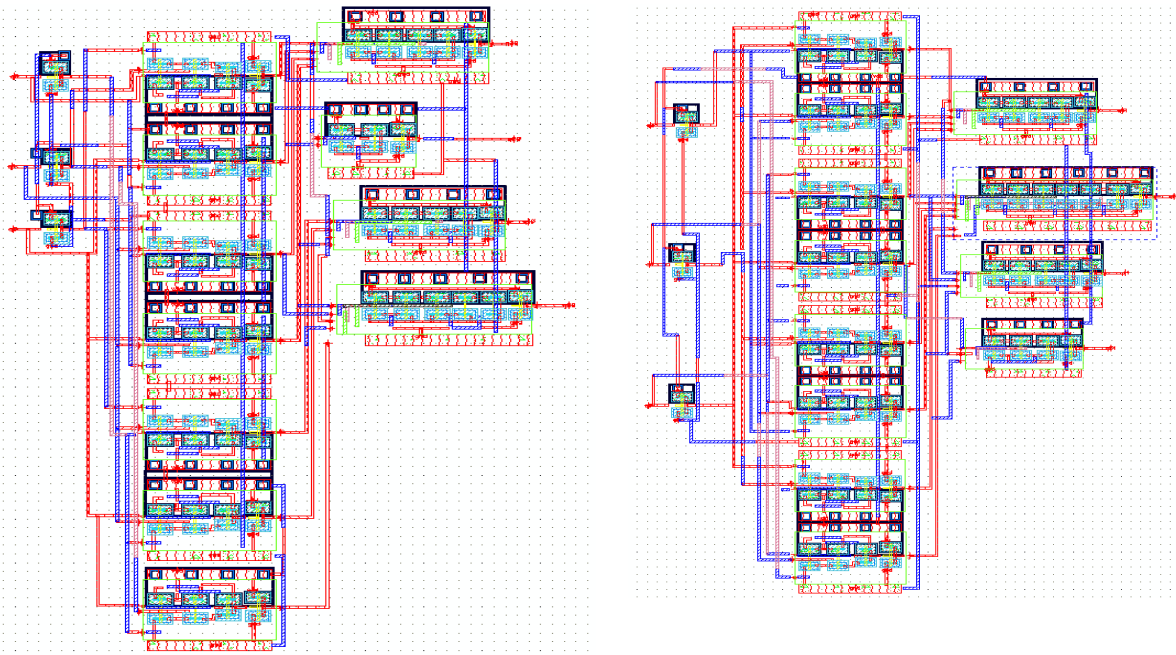


Figure 5-8 Schematic for Proposed Multiplicative Inverse (a) MSB as ‘0’ (b) MSB as ‘1’



(a)

(b)

Figure 5-9 Layouts for Proposed Multiplicative Inverse (a) MSB as '0' (b) MSB as '1'

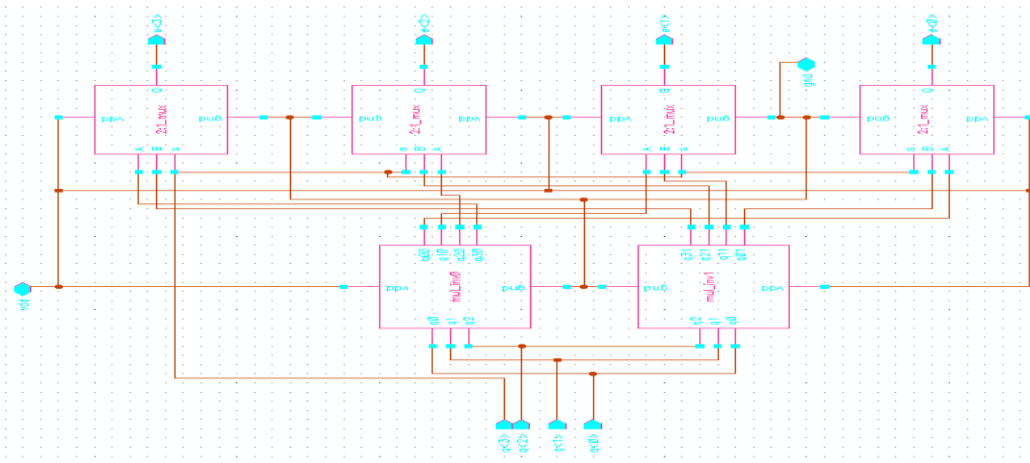


Figure 5-10 Complete Schematic for Proposed Multiplicative Inverse in GF (2⁴)

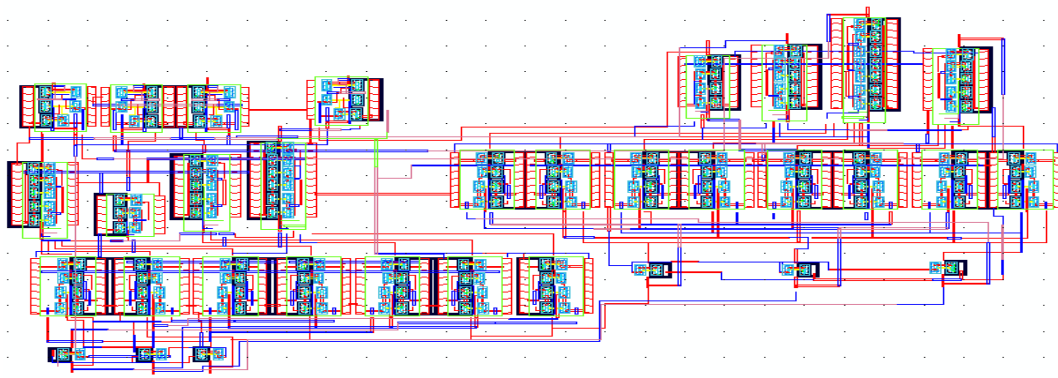


Figure 5-11 Complete Layout of Proposed Multiplicative Inverse in GF (2⁴)

5.5 SCHEMATIC AND LAYOUT OF PROPOSED ARCHITECTURE OF MULTIPLICATION IN GF (2^2)

The proposed architecture of multiplication in GF (2^2) shown in Figure 3-1. Here, the Schematic and Layout has drawn, Schematic has in Figure 5-12 and Layout in Figure 5-13. We know that multiplication in GF (2^2) is a major component to find the multiplication in GF (2^4) in composite field arithmetic which is essential for S-BOX. The schematic of multiplication in GF (2^4) and layout of it is shown in Figure 5-14 and 5-15 respectively.

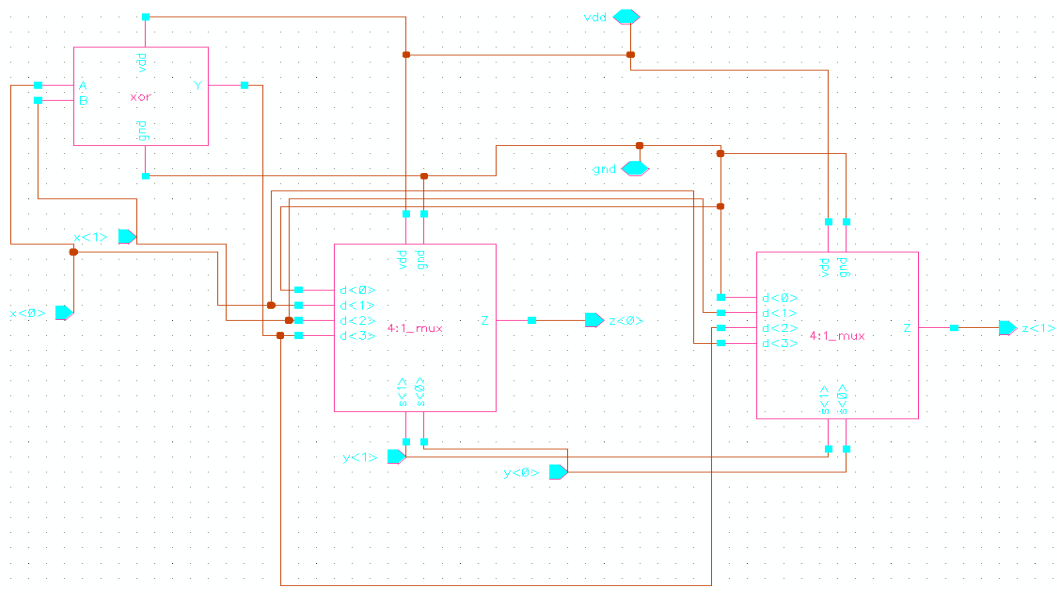


Figure 5-12 Schematic of proposed architecture of multiplication in GF (2^2)

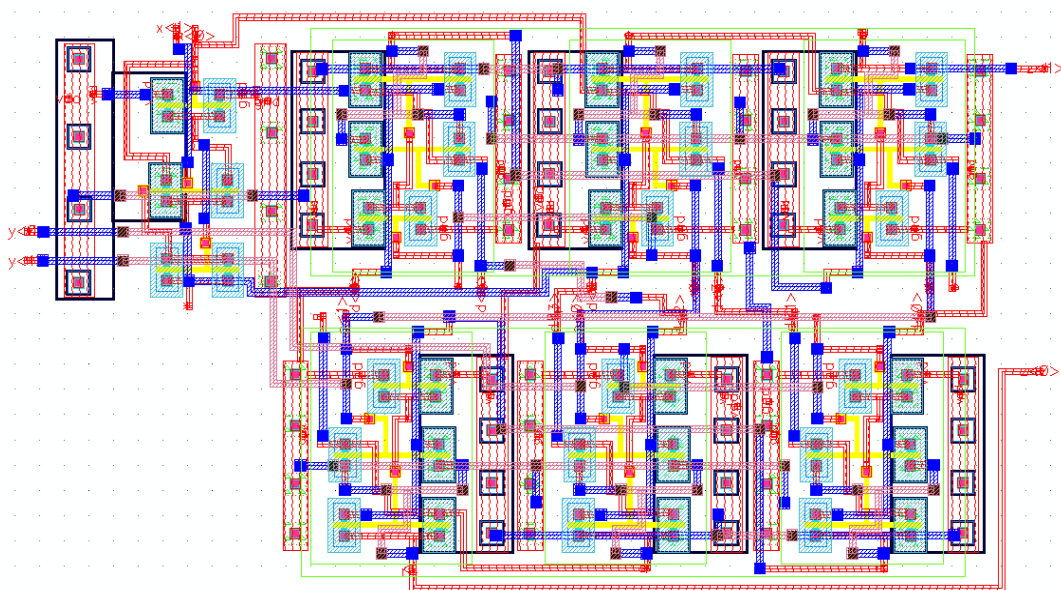


Figure 5-13 Layout of proposed architecture of multiplication in GF (2^2)

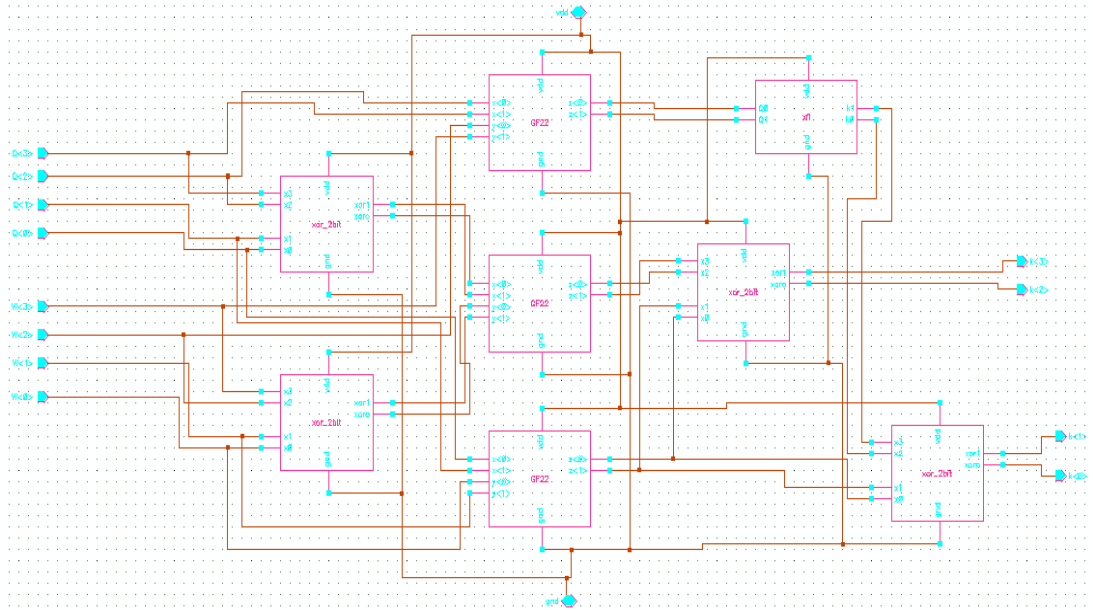


Figure 5-14 Schematic of multiplication in GF (2^4)

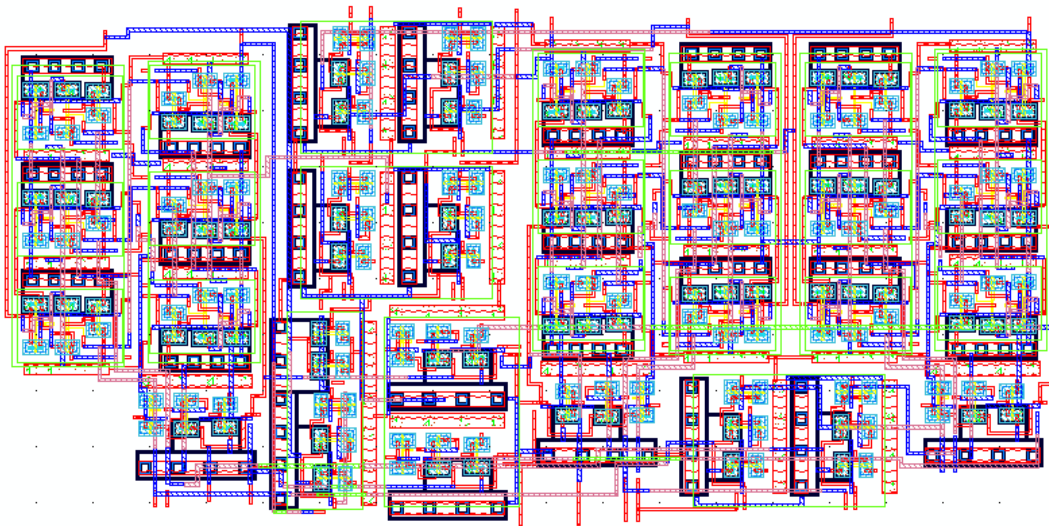


Figure 5-15 Layout of multiplication in GF (2^4)

The optimized layout of multiplication in GF (2^4) has drawn in the Cadence Virtuoso Layout Editor using UMC 180 nm technology. The area of the full custom design of multiplication in GF (2^4) is approximately $4000 \mu\text{m}^2$.

5.6 SCHEMATIC AND LAYOUT OF FOUR BIT XOR USING THE PROPOSED NOVEL XOR GATE.

The four bit XOR is used to do the addition operation of four bits to implement the S-BOX. The Schematic and Layout of 4 bit XOR has presented in Figure 5-16. Figure 5-16 (a) shows the Schematic; and Fig. 5-16 (b) shown layout of it.

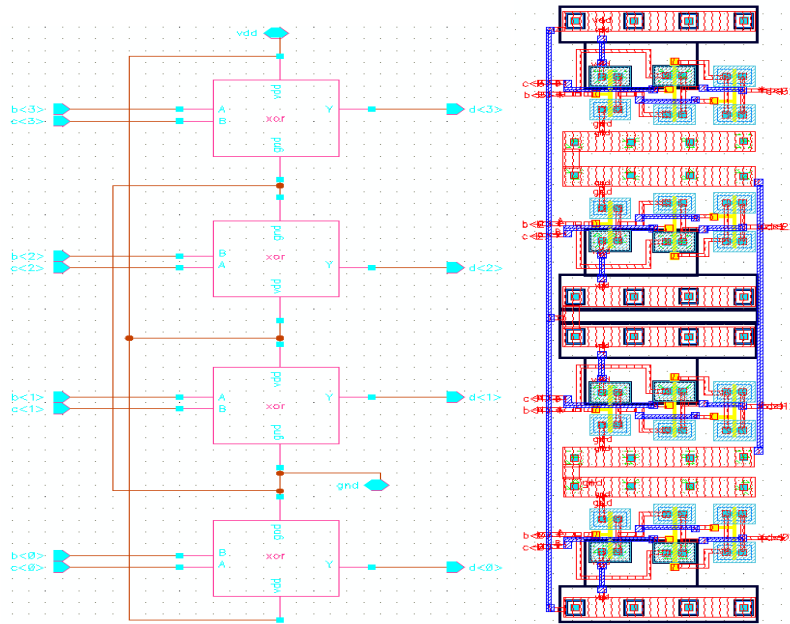


Figure 5-16 Schematic and Layout of 4 bits XOR (a) Schematic and (b) Layout of (a)

5.7 SHEMATIC OF PROPOSED S-BOX

The Schematic of proposed S-BOX has drawn using all the modification done in architecture level, Schematic diagram can be seen in Figure 5-17. The simulated output of the proposed architecture; shown in Figure 5-18 which is verified with the simulated output in FPGA.

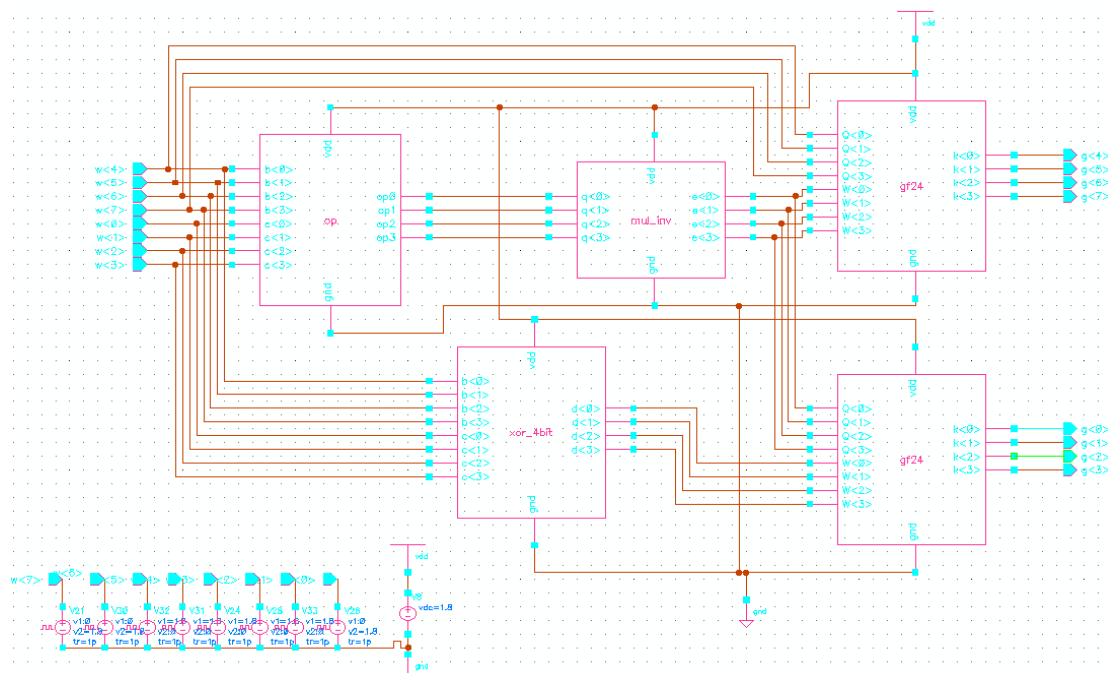


Figure 5-17 Schematic of proposed S-BOX

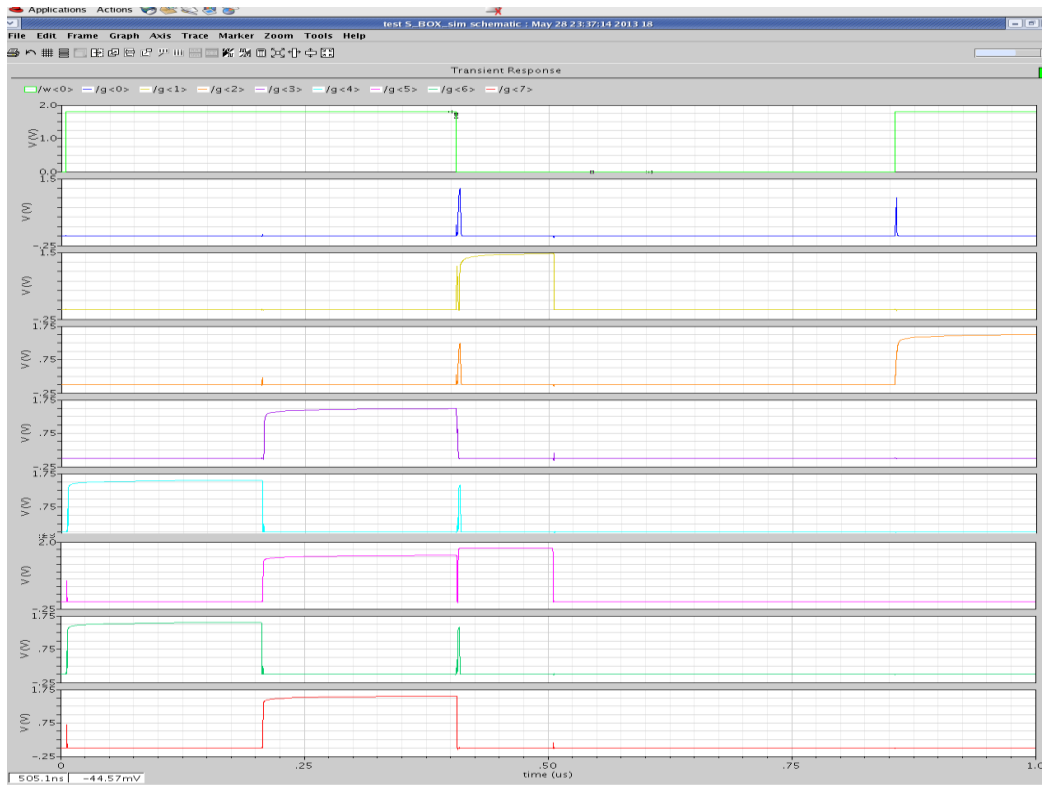


Figure 5-18 Simulated output of Schematic of proposed architecture of S-BOX

5.8 LAYOUT AND POST LAYOUT SIMULATION RESULTS OF PROPOSED S-BOX

To achieve the very high speed, full custom design of the s - box in composite field has been done for the proposed s-box architecture in Cadence Virtuoso. The optimized layout of proposed schematic of S-BOX has drawn in Virtuoso Layout Editor using the UMC 180nm technology library. Figure 5-19 shows the complete layout; DRC (Design Rule Check) and LVS (Layout and Schematic match) can be seen in Figure 5-20 and 5-21 respectively.

The parasitic extraction has done which is required for post layout simulation, layout after extraction shown in Figure 5-22 and extraction report shown in Figure 5-23. The post Layout simulation result can be seen in the Figure 5-24. Table XIII shows the comparative results of the proposed S-BOX which is compared between before post layout and after post layout simulations. The design summary gives the total number no transistors (nmos and pmos) presents in the layout and it also gives the total count of parasitic resistance and parasitic capacitance.

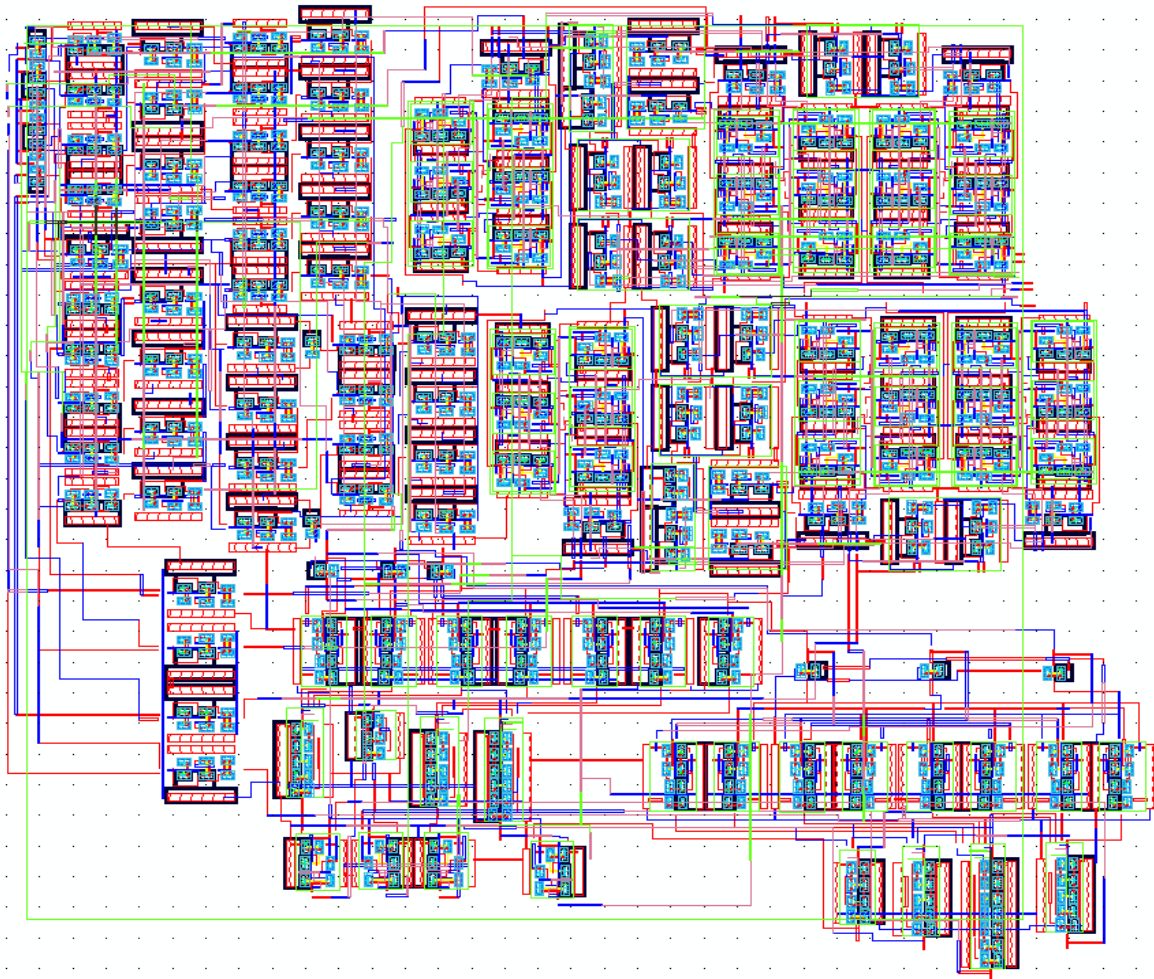


Figure 5-19 the optimized layout of proposed S-BOX

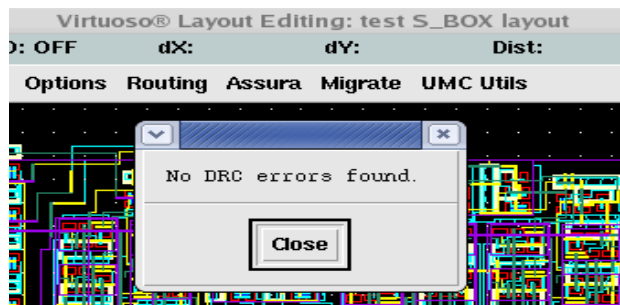


Figure 5-20 DRC check of layout

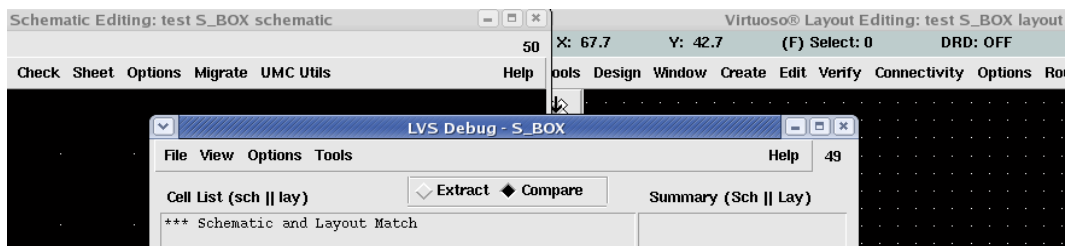


Figure 5-21 LVS match of layout

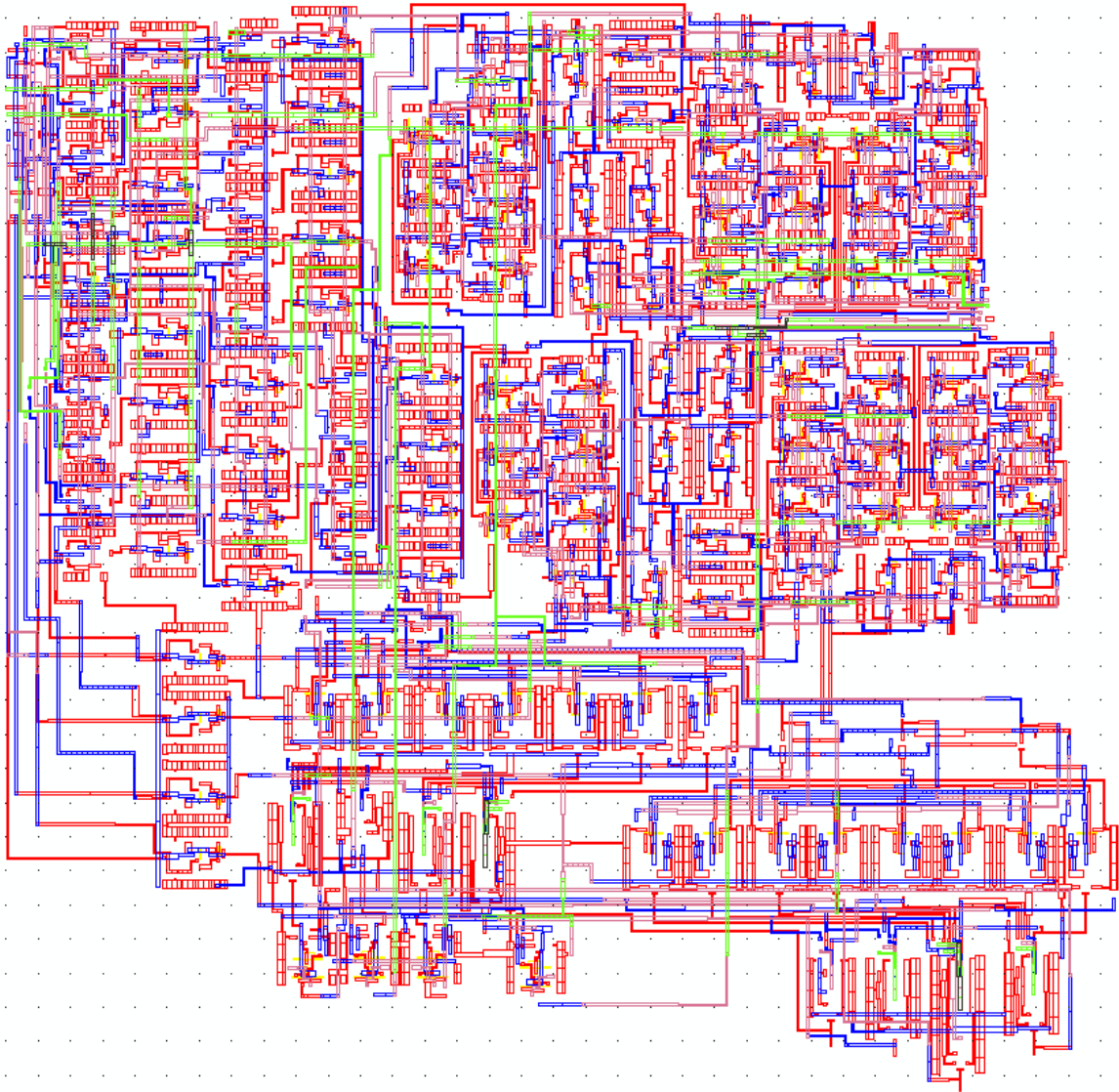
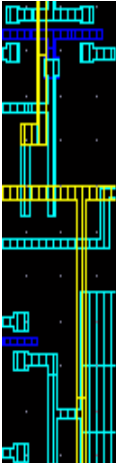


Figure 5-22 Layout after parasitic extraction



Summary for test/S_BOX/av_extracted

instance count totals:

lib	cell	view	total
UMC_18_CMOS	N_18_MM	symbol	495
UMC_18_CMOS	P_18_MM	symbol	395
UMC_18_CMOS	pcapacitor	symbol	6737
UMC_18_CMOS	presistor	symbol	10007

extracted view creation completed

Figure 5-23 Design summary and device count after parasitic extraction

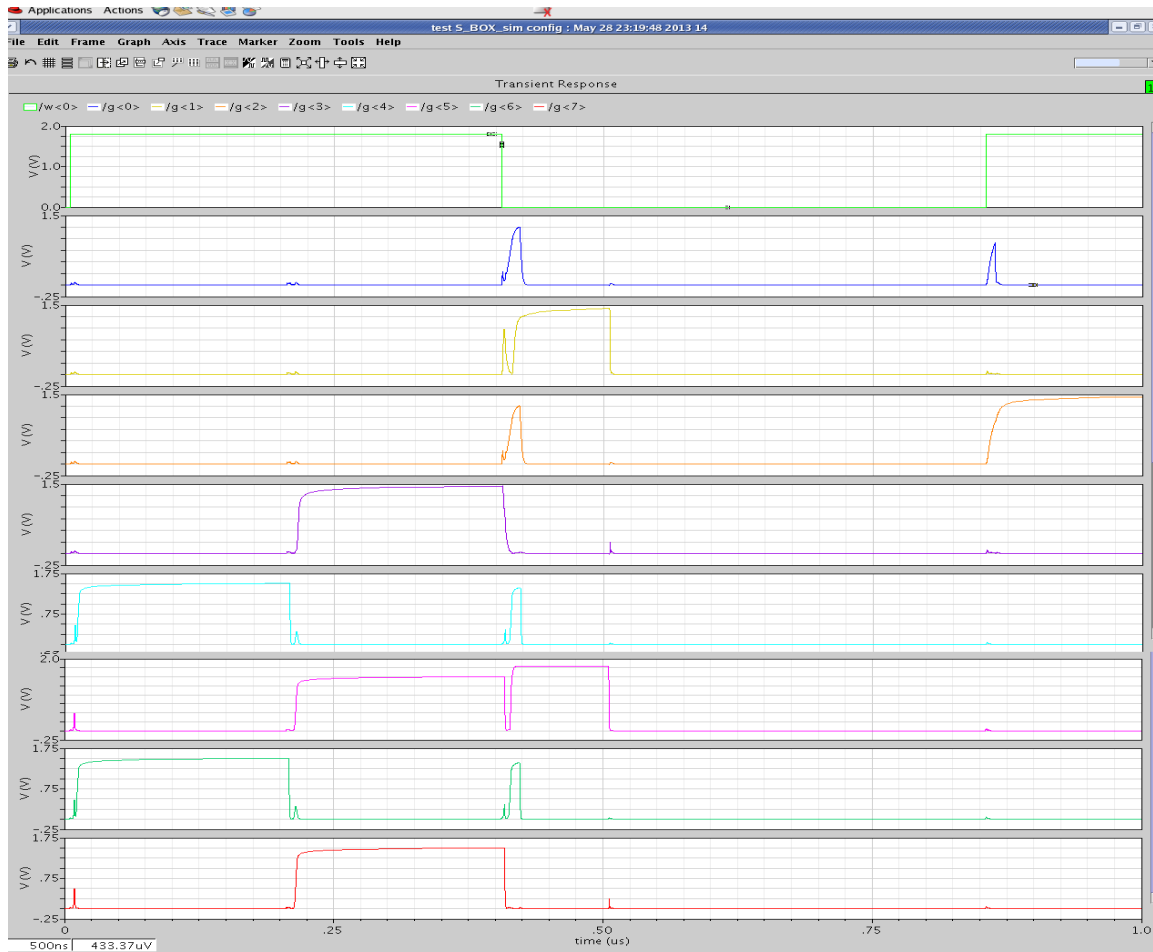


Figure 5-24 Post layout simulated output after parasitic extraction

Table XIII: Comparison Results between Post layout and before Post layout simulation

	Post Layout	Before Post Layout
Maximum Delay (ns)	8.2	1.81
Total Average Dynamic Power (μW)	22.6	7.6

5.9 CONCLUSION

The novel XOR gate is proposed for use in s-box design which is efficient in terms of delay and power along with high noise margin. The implementation has been done in 180 nm UMC technology. The designed s-box using proposed XOR occupies a total area of $27348 \mu\text{m}^2$.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

We have proposed optimized VLSI architecture of S-box for AES algorithm. The architecture of s-box in composite field has been modified in order to have high speed and low areas. Using the proposed s-box, AES architecture has been implemented using the merging technique in FPGA. The proposed AES architecture has delayed improvement of approx. 1.6 ns along with area improvement of 287 FPGA slices when implemented in the Spartan-6 FPGA of Xilinx. The full custom design of the s-box has been done in 180 nm technology in Cadence using novel XOR gate which has high speed and low power consumption as compared to existing one. The designed s-box chip consumes 22.6 μ W and has 8.2 ns delay after post layout simulation.

6.2 FUTURE WORK

- ❖ Full custom design of AES.
- ❖ Tape out of full custom AES.
- ❖ Video encryption in real-time using proposed design implemented in FPGA.

REFERENCES

- [1] B.A. Forouzan and D. Mukhopadhyay, *Cryptography and Network Security*, 2nd Ed., Tata McGraw Hill, New Delhi, 2012.
- [2] M. I. Soliman, G. Y. Abozaid, "FPGA implementation and performance evaluation of a high throughput crypto coprocessor," *Journal of Parallel and Distributed Computing*, Vol. 71 (8), pp.1075-1084, Aug. 2011.
- [3] V. K. Pachghare, *Cryptography and information security*, E. E. Ed., PHI Learning, New Delhi, 2009.
- [4] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A Lightweight High-Performance Fault Detection Scheme for the Advanced Encryption Standard Using Composite Fields," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 19 (1), pp. 85-91, Jan. 2011.
- [5] Federal Information Processing Standards Publication 197 (FIPS 197), available online, <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>.
- [6] X. Zhang, K. K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 12 (9), pp. 957-967, Sep. 2004.
- [7] M. Jridi and A. AlFalou, "A VLSI implementation of a new simultaneous images compression and encryption method," 2010 IEEE International Conference on Imaging Systems and Techniques (IST), pp.75-79, July 2010.
- [8] Chih-Pin Su, Tsung-Fu Lin, Chih-Tsun Huang, and Cheng-Wen Wu, "A High-Throughput Low-Cost AES Processor," *IEEE Communications Magazine*, Vol.41 (12), pp.86-91, Dec. 2003.
- [9] L. Ali, I. Aris, F. S. Hossain and N. Roy, "Design of an ultra-high speed AES processor for next generation IT security," *Computers and Electrical Engineering*, Vol.37 (6), pp.1160-1170, Nov. 2011.
- [10] K.H. Chang, Y.C. Chen, C. C. Hsieh, C. W. Huang and C. J. Chang, "Embedded a Low Area 32-bit AES for Image Encryption/Decryption Application," *IEEE International Symposium on Circuits and Systems*, pp. 1922-1925, May 2009.

- [11] J. M. G. Criado, M. A. V. Rodriguez, J. M. S. Perez, J. A. G. Pulido, "A new methodology to implement the AES algorithm using partial and dynamic reconfiguration," *Integration, the VLSI Journal*, Vol.43(1), pp. 72-80, Jan. 2010.
- [12] J. V. Dyken, J. G. Delgado-Frias, "FPGA schemes for minimizing the power-throughput trade-off in executing the Advanced Encryption Standard algorithm," *Journal of Systems Architecture*, Vol.56(2-3), pp. 116-123, Mar. 2010.
- [13] I. Hammad, K. E. Sankary and E. E. Masry, "High-Speed AES Encryptor With Efficient Merging Techniques," *IEEE Embedded Systems Letters*, Vol.2 (3), pp.67-71, Sept. 2010.
- [14] N. Ahmad, R. Hasan, W. M. Jubadi, "Design of AES S-Box using combinational logic optimization," *IEEE Symposium on Industrial Electronics & Applications*, pp.696-699, Oct. 2010.
- [15] N. Ahmad, S. M. R. Hasan, "Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate," *Integration, the VLSI Journal*, Article in Press.
- [16] Edwin NC Mui, "Practical Implementation of Rijndael S-Box Using Combinational Logic," *Custom R&D Engineer Taxco Enterprise Pvt. Ltd.*
- [17] Jarvinen, K., Tommiska, M., and Skytta, "A Fully Pipelined Memoryless 17.8 Gbps AES-128 Encryptor". *Proc. ACM/SIGDA 11th ACM Int. Symposium on Field-Programmable Gate Arrays, FPGA 2003, Monterey, CA, USA, February 2003*, pp. 207-215.
- [18] Kimmo Järvinen, Matti Tommiska and Jorma Skyttä, "Comparative Survey of High Performance Cryptographic Algorithm Implementations on FPGAs", *IEE Proceedings - Information Security*, vol. 152, no. 1, Oct. 2005, pp. 3-12.
- [19] A. Rudra, P.K. Dubey, C.S. Jutla, V. Kumar, J.R. Rao, and P. Rohatgi. "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic", *Workshop on Cryptographic Hardware and Embedded Systems (CHES2001)*, pages 175-188, May 2001.
- [20] Tim Good and Mohammed Benaissa, "Very Small FPGA Application-Specific Instruction Processor for AES", *IEEE Transactions on Circuit and Systems-I*, Vol. 53, No. 7, July 2006.

[21] Data Encryption Standard (DES), FIPS PUB (46-3), Oct. 25, 1999, Federal Information Processing Standard 46-3

PUBLICATIONS

1. **Saurabh Kumar**, V.K. Sharma and K.K. Mahapatra, “An Improved VLSI Architecture of S-box for AES Encryption/Decryption,” *2013 IEEE International Conference on Communication Systems and Network Technologies (CSNT)* (**Paper published**).
2. **Saurabh Kumar**, V.K. Sharma and K.K. Mahapatra, “Low latency VLSI Architecture of S-box for AES Encryption/Decryption,” *2013 IEEE International Conference on Circuit, power and computing Technologies (ICCPCT)* (**Paper published**).