NOVEL DIGITAL VLSI IMPLEMENTATION OF DATA ENCRYPTION ALGORITHM USING NANO-METRIC CMOS TECHNOLOGY

A THESIS PRESENTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE

OF

DOCTOR OF PHILOSOPHY

IN

ENGINEERING

AT

MASSEY UNIVERSITY

AUCKLAND

NEW ZEALAND

NABIHAH @ NORNABIHAH AHMAD

2013

ABSTRACT

Implementations of the Advanced Encryption Standard (AES) have rapidly grown in various applications including telecommunications, finance and networks that require a low power consumption and low cost design. Presented in this thesis is a new 8-bit stream cipher architecture core for an application specific integrated circuit AES crypto-processor. The chip area and power are optimised along with high throughput by employing circuit-level techniques, resource sharing and low supply voltage. The proposed design includes a novel S-box/ InvS-box, MixColumn/ InvMixColumn and ShiftRow/ InvShiftRow with a novel low power Exclusive OR (XOR) gate applied to all sub systems to minimise the power consumption. It is implemented in a 130nm CMOS process and supports both encryption and decryption in Electronic Codebook Mode (EBC) using 128-bit keys with a throughput of 0.05Gbit/s (at 100MHz clock). This design utilises 3152 gate equivalents, including an on-the-fly key scheduling unit along with 4.23µW/MHz power consumption. The area of the chip is 640µm×325µm (0.208 square mm), excluding the bonding pads. Compared to other 8-bit implementations, the proposed design achieves a smaller chip size along with higher throughput and lower power dissipation. This thesis also describes a new fault detection scheme for S-box/ InvS-box that is parity prediction based to protect the key from fault attacks.

ACKNOWLEDGEMENTS

I would firstly like to profoundly thank my supervisor Dr. Rezaul Hasan for his endless support, encouragement, guidance and advice throughout my doctoral studies. There were many difficulties during the theoretical and experimental aspects of the work, but Rezaul's direction and never ending patience with me has helped me to finally accomplish this thesis. Rezaul's deep knowledge of VLSI and integrated circuit design along with his research insights facilitated me to achieve the research goals. Without him this work could not have been brought to its final accomplishment. In addition, his guidance and effort through long days and long hours of authorship work has resulted in several high standard international journal publications during the course of this research.

Thanks are also due to the staff and senior students at the School of Engineering and Advanced Technology (SEAT) for their friendship and continuous cooperation in dealing with the issues arising during my studies. I would like to especially thank Ananiah Sundararajan, Robert Fisk, Jack Li, Sadia Alam, Stepan Lapshav, Dmitri Roukin, Loke Chun Eng and Muhammad Khurram for their time in helping me with the software and my research design, as well as, their advice and suggestions. The support of the MOSIS academic research program is also gratefully acknowledged for covering the cost of my prototype fabrication.

I would also like to thank my family for their unconditional support and encouragement during the course of my doctoral studies.

DECLARATION

The author declares that this is her own work except where due acknowledgement has been given. It is being submitted for the PhD in Engineering to Massey University, New Zealand.

This thesis describes the research carried out by the author at the School of Engineering, Massey University, New Zealand from March 2009 to July 2013, supervised by Dr. Rezaul Hasan.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

DEFINITIONS AND ABBREVIATIONS

AES	Advanced Encryption Standard (AES),
DES	Data Encryption Standard (DES)
NIST	National Institute of Standards and Technology
WLAN	Wireless Local Area Networks
FPGA	Field Programmable Gate Array
ASIC	Application-Specific Integrated Circuits
VHDL	VHSIC Hardware Description Language
VLSI	Very Large Scale Integration
RFID	Radio-Frequency Identification Devices
CMOS	Complementary MOS
XOR	Exclusive OR
EBC	Electronic Codebook
LUT	Lookup Tables
ROM	Read-Only-Memory
ANF	Algebraic Normal Form
CSE	Common Sub expression Elimination
BDD	Binary Decision Diagram
TBDD	Twisted Binary Decision Diagram
DSE	Decoder-Switch-Encoder structure
SOP	Sum of Product
SABL	Sense Amplifier Based Logic
CTR	Counter Mode
CBC	Cipher-block Chaining
CFB	Cipher feedback
OFB	Output feedback
IV	Initialisation vector
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
LP	Low Power
GP	General Purpose

Vt	Threshold voltage
CLB	Configurable Logic Blocks
GE	Gate Equivalent
ASIP	Application-Specific-Instruction-Processor
RAM	Random-access memory
SOC	System-on-chip
CC	Cross-coupled
P-XOR	Powerless XOR gate
GDI	Gate-Diffusion-Input
PDP	Power-Delay Product
DRC	Design Rule Checking
LVS	Layout vs. Schematic
EDP	Energy-Delay Product
PAP	Power-Area Product
PAT	Power-Area-Latency
ESD	Electrostatic Discharge

CHAPTER 1

INTRODUCTION

1.1 Introduction

A significant trend in the global network allows users to communicate and share information securely with other systems located around the world. The exponential growth of virtual communication has made cryptography algorithms one of the most important forces for security and to protect information. The concern with secure communication involving wireless networks has led to the development of strong and advanced cryptographic algorithms. The cryptographic algorithm plays a significant role in ensuring the integrity of data, by protecting the data from being modified or corrupted. It is used in applications such as electronic commerce, digital signature and electronic communication for user authentication to ensure the confidentiality, security and privacy of data from unauthorised users.

Encryption is the process of converting plaintext into cipher text by applying an algorithm that performs a substitution and transformation on the plaintext. Decryption is the reverse process of converting the cipher text to the original plaintext. Substitution maps each element in the plaintext into another element, while transformation rearranges the element in the plaintext. Plaintext is the original intelligible data as an input to the algorithm, and cipher text is the scrambled data produced as output after the encryption process.

A cryptographic algorithm also uses a secret key as input, which affects the output produced by the algorithm. There are many encryption algorithms available that are widely used for security of information. They can be categorised into two (2) categories, which are Symmetric (private) and Asymmetric (public) key encryption.

In Symmetric key encryption, or secret key encryption, only one key is used by the sender and receiver to encrypt and decrypt data for secure communication. The key is distributed between entities before the transmission. This encryption is widely used in government, private sector application, and military. The strength of Symmetric key encryption depends upon the size of the key used; a longer key makes the algorithm harder to break. Symmetric key can be divided into two (2) subcategories: block cipher and stream cipher. Block cipher operates on a fixed and large number of bits to provide a high degree of security. Stream cipher operates on a single bit of data, adding a bit from a plaintext bit with a continuous stream of key bits. The most common symmetric encryption algorithms are Advanced Encryption Standard (AES), Data Encryption Standard (DES), 3DES, RC2, Blowfish, and RC6.

In Asymmetric key encryption, two keys are used: private and public keys. The public key is used for encryption, and openly available, while the private key is secret, and used for decryption. It has solved the problem of key distribution in Symmetric key encryption, but it is based on mathematical functions that are computationally intensive and require computational processing power that makes it almost 1000 times slower than Symmetric encryption [1], and not very efficient for small mobile devices [2]. It is generally not used to encrypt the whole data message, but only to encrypt secret keys. Examples of Asymmetric keys are RSA, ElGamal, and LUC.

The National Institute of Standards and Technology (NIST) organised a contest to provide a standard for an encryption algorithm to replace the earlier insecure standard symmetric key algorithm, DES, as DES was found to be vulnerable to brute force attack. Rijndael algorithm, which was developed by Joan Daemen and Vincent Rijmen [3], was announced as the new AES algorithm in 1997, based on the primary criteria of security, performance, efficiency in software and hardware implementation and flexibility.

AES is one of the most common symmetric encryption algorithms and has been widely adopted for a variety of encryption needs, such as Wireless Local Area Networks (WLAN), smart cards and secure transactions via the Internet. AES can be implemented on a wide range of platforms under different constraints.

AES can be implemented either using software or a hardware platform. The major factors that influence the implementation choices are speed and area cost. Software implementations are designed and coded in programming languages such as Java, C and C++. The advantages of using software implementation is that it is easy to implement and cheap, but the drawbacks are slow performance, high-power consumption and decreased security. Software implementations are usually developed to run on digital signal processors and general-purpose processors where the speed of encryption and level of security are not a main requirement.

Hardware implementations have been developed to satisfy different constraints and requirements such as low power consumption, low area and high data rate. The efficiency of AES hardware implementation in terms of size, speed, security and power consumption depends largely on the AES architecture [4]. Moreover, different encryption applications, depending on purpose and priority, are suited to different implementations. At the cost of increased area and power consumption, higher throughputs may be accomplished by using a loop-unrolled hardware structure. Decreasing the hardware size with a small depth of the datapath offers low power consumption and improves the throughput of the system. Hardware implementation approaches can be either based on Field Programmable Gate Array (FPGA), or Application-Specific Integrated Circuits (ASIC) designed and coded in hardware description languages (VHDL or Verilog HDL), although using schematic design entry is more suitable in such a case. The FPGA can be reconfigured to perform different functions. ASIC are designed from the behavioral description until the physical layout. In portable application computing, resources are usually limited and dedicated hardware implementation of the security process is essential [5]. Implementation using FPGA offers flexibility and security, but medium speed as compared to ASIC, and is unsuitable for such applications, mainly due to size and power constraints. FPGAs are slower compared to ASICs due to extra logic overhead to support reconfiguration [6].

A compact, small foot-print, full-custom chip is more suitable in such a case. In addition, such a dedicated hardwired AES implementation can provide a higher data rate for fast handling of ciphered network data packets in applications such as routers, compared to software packages.

The hardwired implementation is also physically secure since tampering by an attacker is more difficult. An ASIC chip can be designed to be tamper resistant in order to avoid information leakage based either on fault attack or power consumption analysis. For FPGA implementation, the configuration bit-stream can be used to identify the algorithm, hence making it less secure against physical manipulation when there is physical access to the device. For high throughput, a loop-unrolled pipelined structure is used; alternatively to save power and area, an iterative single round with resource sharing can be implemented.

AES is the official standard and is easy to use and implement: all cryptography libraries support this standard [7]. AES is also believed to provide security strength for longer key sizes (128, 192 and 256 bits). A related-key attack on the AES has been reported that it could break up to only eight rounds of the cipher with less work than that of an exhaustive key-search [8].

1.2 Motivation

There are critical issues facing nanometer VLSI technology in order to design a low-power system, due to the continuous scaling of technology. Additionally, power has become a bottleneck for battery-powered portable and high speed devices. Higher power consumption raises the chip temperature, which affects the circuit's reliability and increases the cost of cooling [9]. Currently, the AES system implemented in VLSI technology emphasises low power consumption due to the popularity of portable device applications such as Radio-Frequency Identification Devices (RFID) tags, and wireless communication, which require low power, low area and high security.

The reason to reduce power consumption is to save energy within the device, and to reduce heat dissipation. Power consumption also correlates to the security level of data that is processed and stored in cryptographic devices: the instantaneous power consumption of a device depends upon the data manipulated during logical operations due to the physical properties of Complementary Metal-Oxide Semiconductors (CMOS) logic, and at the lowest level equates to transistors' switching.

It is, therefore, crucial to design an AES system that is resistant to a power analysis attack, because nowadays attackers can break the security secret key through side-channel information leaked by the switching behavior of the digital CMOS gate [5] and [10]. The attacks exploit the dependence of the dynamic power consumption on the inputs of a cryptographic algorithm. A system with decreased information leakage is needed to prevent the extraction of secret keys by attackers.

Leakage power has become more significant in the power dissipation of nanoscale CMOS circuits, due to scaling technology and supply voltage. The most efficient technique to save power is to reduce the supply voltage of integrated circuits. The smaller supply voltage means less power. As the transistor geometries are reduced, only a small voltage is required to avoid electrical breakdown and obtain the required performance. The threshold voltage has to be scaled to satisfy the high-performance requirement, however this exponentially increases in the leakage current. There is, therefore, a need to design a CMOS circuit that can satisfy a low-power consumption requirement without scarifying the performance.

Another contributing factor to the power consumption in an AES system is the Substitution box (S-box). This is the core of any AES implementation and is considered a fully complex design that consumes the major portion of the power and energy budget of the AES hardware. An efficient technique is to optimise the modules' architecture and reduce the number of S-boxes used in AES to achieve low power consumption and small area.

In the interests of low power design, 8-bit datapath architecture has been reported as one of the techniques to reduce the power consumption in AES. Reducing the depth of the datapath can also decrease the size (hardware area) and compromise the throughput. With an 8-bit datapath, sub-components in the AES will be reduced resulting in reduced system power consumption. Numerous designs on 8-bit datapath AES architecture have been implemented in FPGA technology. The motivation for this thesis is to develop and analyze the performance, power consumption and area of full custom 8-bit AES crypto-processor implementation.

1.3 Research Objectives

In this thesis, the design techniques for each front-end component are explored in relation to finding a design optimisation solution for an AES crypto-processor. The objectives for this thesis are:

1. To establish an understanding, through reviewing relevant knowledge and publications, of design techniques for implementing state-of-the-art, front-end blocks for an encryption/ decryption AES crypto-processor. To analyse these techniques to identify shortcomings in existing design methodologies and investigate opportunities for further improvement and development within the reviewed literature.

2. To design and implement an AES crypto-processor to improve performance (area, power and delay). To gain experience working with integrated circuit (IC) design and simulation software to meet targeted design specifications.

3. To evaluate, fabricate and verify functionality and timing, testing the microchip and performance of the novel AES crypto-processor against its own predicted performance, as well as against previously published designs, using the developed methodology identified in (1). Therefore, new recommendations and insights into the future direction of a compact AES crypto-processor can be introduced.

1.4 Contributions to Knowledge

The major contribution of this thesis is the design and development of the 8-bit width architecture of the AES crypto-processor. The research work described in subsequent chapters of this thesis makes original contributions to knowledge within the field of AES crypto-processor through:

1 Design and construction a low power, small area 2-input Exclusive OR (XOR) gate to optimise the layout area and power consumption of an AES system. Two XOR gates have six transistors, which provide full swing output voltage for all

input combinations. The area of the core circuit is only about 56 square µm; the critical path propagation delay is 1.5659ns with a power dissipation of 0.2312nW at 0.8V supply voltage. This resolves the silicon constraints architecture for each sub-module of AES and contributes to the overall low power consumption of AES crypto-processor. This proposed XOR gate also can be used in other system-on-chip (SOC) implementations require for a low-power and low-area design.

2 Design and development of a fully-integrated compact 8-bit stream cipher core of an application-specific integrated circuit AES crypto-processor that supports both encryption and decryption using 128-bit data block and 128-bit key in 130nm manufacturing process. This results in a smaller chip size with gate count 3152 gate equivalents, including on-the fly Key Scheduling architecture with throughput of 0.05 Gbit/s (at 100 MHz clock) along with 4.23µW/MHz power consumption.

3 Developing new, improved efficient S-box/ InvS-box architecture by merging the sub component of the typical multiplicative inverse to optimise and reduce the hardware complexity of the circuit. This new architecture only utilised 147 gates: 105 XOR gates, 38 AND gates, 3 NAND gates and 1 OR gate. This proposed S-box/ InvS-box contributes 7% of the total AES crypto-processor area. The new error detection method using parity code error detection for S-box of AES was proposed for better protection of S-box.

4 Developing a new optimized MixColums/ InvMixColumns architecture using the decomposition method requiring 664 XOR gates for full *State* matrix transformation that lead to the compact area with a small propagation delay of AES system.

5. Developing a joint ShiftRow and InvShiftRow architecture using twelve 8-bit registers, along with six 2-to-1, and one 4-to-1 multiplexer requiring 448 gates, which is suitable for 8-bit and 32-bit datapath contributes 14% of the overall area of AES system .

1.5 Thesis Organisation

This thesis is organized as follows: Chapter 2 reviews the literature relevant to the objectives, details an overview of the mathematical definitions in Galois field of AES algorithm and a description of the AES architecture system. In Chapter 3, a novel XOR gate, low-power, low-voltage full swing for CMOS Galois field arithmetic is reported and discussed. Chapter 4 describes a new compact combinational logic design of S-box/ InvS-box and simulation results. This is followed by a chapter on the fault detection scheme of AES S-box/ InvS-box that is parity prediction based. Chapter 6 presents the hardware architecture, layout implementation, discussion of the analysis result and comparison with other designs for the efficient integrated AES crypto-processor for 8-bit stream cipher. It also covers the new topology for MixColumns/ InvMixColumns, ShiftRows/ InvShiftRows and Key scheduling transformation. The next chapter describes a short introduction about the verification, evaluation and the results of the performances of the front-end components based on analysis, simulation and testing, including experimental output generated from the simulation results. Finally, Chapter 8 summarizes all research outcomes from the research, the challenges and issues arising from this research and provides possible directions for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents the brief essential concepts in order to understand AES algorithm. The first section introduces the fundamental mathematical concepts of finite fields and the polynomial multiplier over binary finite fields, followed by a discussion of its different implementation architectures for each AES transformation. Then the design requirements and specifications of AES are discussed. Finally, this chapter also reviews a literature on existing AES systems with various widths of datapath architecture.

2.2 Mathematical Background

AES algorithm is based on operations over the finite field concept. Finite fields are fields with a finite number of elements, and operate at byte level and are known as an element of Galois field (2^8).

2.2.1 Finite Fields

All byte values are presented as the concatenation of the individual bit values, (0 or 1) of elements $\{b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0\}$ and are represented in polynomial notation as $b_7x^7 + b_6x^6 + b_5x^5 + b_4\ x^4 + b_3\ x^3 + b_2\ x^2 + b_1\ x^1 + b_0$. For example, {0011 0101} is interpreted as the finite field element of $x^5 + x^4 + x^2 + 1$. There are two operations in Galois field (2^8): first is multiplication indicated by the symbol "·" and second is addition indicated by the symbol "+". These operations are different from conventional addition and multiplication, but they obey the basic algebraic properties.

Following are the relative finite field properties, for (F, +, ·) is a field if the following properties hold:

- The elements of F form a group under addition, multiplication, subtraction and division.
- Inverse elements always exist, which are additive inverses and multiplicative inverses.
- Associativity of addition and multiplication, for all x, y, and z in F, the following equalities hold: $x + (y + z) = (x + y) + z$ and $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.
- Commutative: The addition and multiplication operations are commutative, i.e. $x + y = y + x$ and $xy = yx$ for all x, y Є F.
- The multiplication operation can be distributed through the addition operation, which is $x \cdot (y + x) = x \cdot y + x \cdot z$ for all x, y, and z Є F.
- A field F with a finite number of elements is a finite field.
- A non-zero element of a finite field F is said to be a primitive element or generator of F if its powers cover all nonzero field elements.
- For every prime number, a unique finite field exists. These fields are denoted as GF (p^m) where p is prime and m is a positive integer.
- A basis for GF (2^m) over GF (2) is a set of m linearly independent elements of GF (2^m). Any element of GF (2^m) can be represented as an algebraic sum of the basis elements. The binary field GF (2^m) contains 2^m elements. Each element is represented by the selected basis. The most common representation is based on the polynomial basis. With the polynomial basis, $\alpha = \{1, \alpha, \alpha^{2,} \ldots, \alpha^{m-1}\}$, the elements of GF ($2^m$) can be represented as the polynomial of degree m-1 as GF (2^m) = $\{A|A= a_0 + a_1{}^{\alpha} + \cdot \cdot \cdot + a_{m-1}\alpha^{m-1}$, where a_j Є GF(2), $0 \leq j \leq m\text{-}1\}$where α is the root of any irreducible polynomial F(x) of degree m over GF(2).

Let

$$F(x) = 1 + f_1x + f_2x^2 + \cdot \cdot \cdot + f_{m-1}x^{m-1} + x^m \qquad (2.1)$$

where f_i ЄGF(2), $0 \leq i \leq m\text{-}1$.

The irreducible polynomial F(x) is often referred to as the field polynomial. The arithmetic in AES has a polynomial basis and uses the polynomial $F(x) = 1 + x + x^3 + x^4 + x^8$ as the field polynomial.

2.2.2 Operations over Binary Finite Fields GF(2^8)

2.2.2.1 Field Addition

An addition operation in Galois field (2^8) of two elements A and B is produced by adding the coefficients of the corresponding powers in the polynomials for both elements. The addition of finite field element operation is performed as a modulo-2 addition of the corresponding bits in the byte and translated as XOR operation, which is denoted by the operator symbol $\oplus$. Field's addition is computed by an m-bit XOR operation and does not require a carry chain. In modulo 2 additions:

$$\begin{aligned} 0 \oplus 0 &= 0, \\ 0 \oplus 1 &= 1, \\ 1 \oplus 0 &= 1, \\ 1 \oplus 1 &= 0. \end{aligned} \qquad (2.2)$$

Subtraction of polynomials is identical to addition of polynomials. For two elements A $\{a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0\}$ and B $\{b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0\}$ their sum is C $\{c_7\ c_6\ c_5\ c_4\ c_3\ c_2\ c_1\ c_0\}$ which is written as $C(x) = A(x) + B(x) = \sum_{i=0}^{7}(a_i + b_i)x_i$.

2.2.2.2 Field Multiplication

Field multiplication over a finite field GF (2^m) corresponds to the multiplication of polynomials modulo an irreducible polynomial of degree 8. A polynomial is irreducible if its only divisors are one and itself. The multiplication results may have a degree greater than 7, which requires the AES irreducible polynomial $m(x) = 1 + x + x^3 + x^4 + x^8$

For example, $\{83\} \cdot \{57\} = \{C1\}$

$(x^7 + x + 1)(x^6 + x^4 + x^2 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$

$\{83\} \cdot \{57\} = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ (modulo)}$$
$$(1 + x + x^3 + x^4 + x^8)$$
$$= x^7 + x^6 + 1 \{C1\}$$

The multiplicative inverse of any nonzero binary polynomial b(x) of degree less than 8, is denoted by $b^{-1}(x)$ and the inversion operation is computed using Extended Euclidean algorithm to find a(x) and c(x) as,

$$b(x)a(x) + m(x)c(x) = 1$$
$$a(x) \cdot b(x) \bmod m(x) = 1 \quad (2.3)$$
$$b^{-1}(x) = a(x) \bmod m(x)$$

Multiplication by x, such as $(b_7x^7 + b_6x^6 + b_5x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0) \cdot (x)$ will result in $b_7x^8 + b_6x^7 + b_5x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$. If b_7 is '0', the equals zero results are already in the reduced form. If b_7 is '1', then the byte is XOR'ed with polynomial m(x), {1B}.

2.3 AES System and Architecture

AES is a symmetric encryption algorithm process with a fixed 128-bit data block and variable length cipher keys of 128, 192 and 256 bits. The basic AES operation is performed by a two-dimensional array of bytes, called the *State* matrix. Each byte in the *State* is considered as an element in GF (2^8) and denoted by S_{ij} where $0 \leq i,j < 4$. NIST FIPS-197 AES Standard uses the key parameters of:

Nb - the length of the Cipher Text / Plain Text
Nk - the length of the Cipher Key
Nr - the rounds of text transformation

The AES is an iterative algorithm that performs iteratively for 10, 12 or 14 rounds depending upon the key length (128, 192 or 256 bits), and provides effective protection of transmitted and stored data against cryptanalytic attacks [3].

For example, for a data block length of 128 bits and key length of 128 bits, the number of standard rounds needed to be implemented is Nr=10, including the final round. The round-keys needed are 11, which are obtained from the key expansion unit. Table 2.1 shows the number of rounds depending upon the key size of AES.

Table 2.1 *AES Key Block Round Combinations in FIPS-197*

	Key size (*Nk* words)	**Block size (*Nb* words)**	**No of rounds**
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

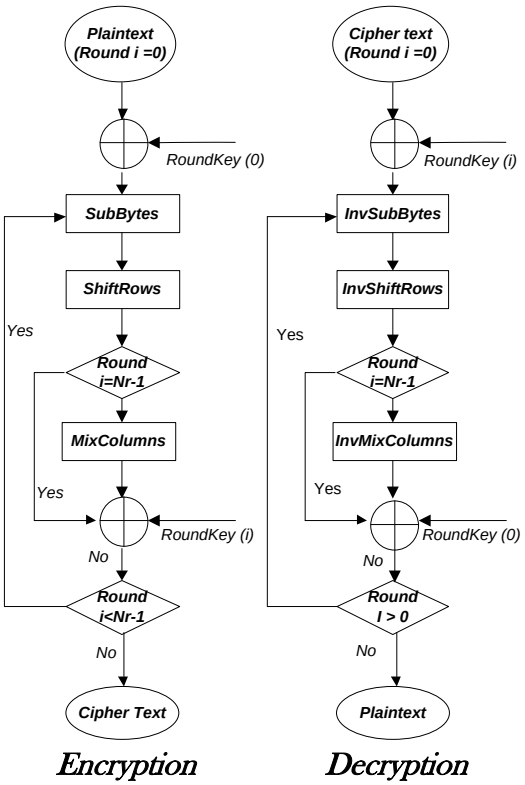


Figure 2.1 *AES flow for encryption and decryption*

2.3.1 Round Transformation

AES algorithm is round-based, using a round function that consists of four different data transformations: SubBytes, ShiftRows, MixColumns and AddRoundKey as shown in Figure 2.1. The initial round only performs AddRoundKey with the *State*, and in the final round, the MixColumns does not apply. The decryption consists of inverse transformations.

2.3.1.1 SubBytes/ InvSubBytes Transformation

The SubBytes transformation is a bijective mapping from eight bits to eight bits [11] applied independently on each byte of the *State* using 16-byte (128-bit) S-boxes. The S-box is a substitution function whose non-linearity is used to defend against linear cryptanalysis. Expressed mathematically, it is an affine transformation of the multiplicative inverse of each byte of the *State* in GF(2^8) . The irreducible polynomial used by the field is $m(x) = x^8+x^4+x^3+x+1$. The operation of the S-box can be expressed as in equation (2.4 - 2.9)[3]

$$S'_{i,j} = MS_{i,j}^{-1} + C \tag{2.4}$$

where, M is an 8x8 matrix and C is a 8x1 vector for affine transformation with elements in

$GF(2)$ with,

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

and,

$$C=[0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1] \tag{2.6}$$

The inverse operation (InvSubBytes) is represented by the operation,

$$S_{i.j=}(M^{-1}(S'_{i,j}+C))^{-1} \tag{2.7}$$

where, M^{-1} and $C' = (M^{-1}.C)$ for an inverse affine transformation are given by,

$$M^{-1} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.8}$$

and

$$[C]=[0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1]. \tag{2.9}$$

Power consumption of AES is related to technology feature size and the used architecture in the system. The S-box is the most expensive building block of any hardware implementation of the AES, and the multiplicative inversion is the most costly step of the S-box transformation. As reported in [12] when S-Box architecture is implemented using conventional architecture which is look up table, it shows that between the power consumption of each primitive component in AES circuits, S-box in the SubBytes component consumes much of the total power (for instance, 75% is consumed in one round/ cycle of loop architecture).

There are various reported techniques to implement the S-box to satisfy criteria such as power, speed and delay for different applications. Among them, there are two main streams: (a) an early technique of implementation using 8 *x* 8 lookup tables (LUTs) that store all predefined 256 8-bit values of an S-box in a Read-Only-Memory (ROM). The advantage of using LUT is that it offers a shorter critical path.

However, it has the drawback of an unbreakable delay path [3] in pipelined designs, and hence it is unsuitable for high speed applications. This delay prohibits each round unit from being divided into more than two sub-stages to achieve any further increase in processing speed. It also requires a large area to implement both AES encryption and decryption as a different table is used in each case. For software implementation, the LUT is used to realise an S-box inversion in GF(2^8), since general-purpose processors are not able to calculate efficiently in term of performance as it require more processing memory that would increase the energy. (b) The alternative way is to design the S-box circuit using combinatorial logic directly from its arithmetic operations. This approach has a breakable delay-path for S-box processing.

Rashmi et al. [13] implemented an S-box using combinational logic without involving computation in Galois field. They proposed two techniques for implementing the S-box: (1) based on logic synthesis using truth tables and (2) direct implementation of the Algebraic Normal Form (ANF) expression for each column. It claims to have a very low critical path delay compared to designs based on the composite field. Ning Chen et al. [14] proposed an efficient technique by applying the common subexpression elimination (CSE) algorithm to reduce the complexities of SubBytes and MixColumns. This technique reduces the area through smaller computational complexities.

Other S-box architectures, such as the positive polarity Reed-Muller structure [12], the binary decision diagram (BDD) [15], or its variance, the twisted binary decision diagram (TBDD) [16], can achieve a high speed design but suffer from extremely large-area cost. For instance, the fastest TBDD design can achieve a 10Gbps throughput rate, but the area is almost eight times larger than the area of the composite field. Another way is using an intermediate one-hot encoding of the input to realise the arbitrary logic functions, also denoted as Decoder-Switch-Encoder structure (DSE). This can minimise the power consumption and delay, but it utilises a large silicon area [17]. In fact, the power consumption of the Sum of Product (SOP) S-box design [18] [13], [19] is less than that of a composite field S-box, but suffers from large silicon area penalty.

A well-known approach to design an S-box from its arithmetic operations involves the multiplicative inversion in GF (2^8) using composite field arithmetic. Rijmen, one of the inventors of the Rijndael algorithm [3], suggested using subfield arithmetic in the computation of an inverse in a Galois field to decompose the field operations from GF(2^8) to GF((2^4)^2). In this technique, hardware area cost can be reduced substantially by sharing the multiplicative inverse step for the SubBytes and the InvSubBytes operations.

In addition, among existing techniques, composite field S-box architecture is the most area-efficient approach for the AES encryption/ decryption algorithm, as the computation cost of certain Galois field operations is lower when the operation is performed in an isomorphic composite field. There are different construction schemes using a composite field in [12], [20] and [21]. Rudra et al. discuss decomposing GF (2^8) into GF (2^4)^2, and all the transformations within the AES system are applied to the composite field. This was first proposed by Wolkerstofer et al. [26], who decomposed the element of GF (2^8) into GF (2^4)^2 to implement the multiplicative inverse in SubBytes. The transformation matrix from GF(2^8) to GF(2^2)^2)^2 was proposed by Morioka and Satoh [12]. Both are using the polynomial basis to represent the field. The most compact composite field design is in [22] using only 92 gates for S-Box design. Canbright [22] proposed a normal basis for the finite field element's representation over GF(2^2). Hardware implementations using normal basis arithmetic typically have less power consumption than other bases [23].

Liu and Parhi [24] reported a fast composite field S-box architecture that showed an increased throughput rate of 56.25%, along with reduced pipeline latency by 40% - 60% compared with other conventional designs. The approaches in [11] and [22] result in a very small sized S-box, but suffer from a longer critical path than the LUT technique. The LUT technique, on the other hand, has a shorter critical path as compared to the composite field approach, but its area-size is two to three times larger.

The S-box is optimised by performing the inversion operation in a composite Galois field of GF ((2^4)^2) or GF (((2^2)^2)^2), obtained from GF (2^8) via isomorphic mapping. The composite field inversion can be used to create compact AES implementations [21] [12].

The composite field is constructed by applying multiple extensions of smaller degrees. To achieve minimal area cost, the composite field is built by repeating degree-2 extensions under a polynomial basis using these irreducible polynomials [3]:

$GF\ (2^2)$ $\quad GF\ (2)$: $x^2 + x + 1$

$GF\ ((2^2)^2)$ $\quad GF\ (2^2)$: $x^2 + x + \varphi$

$GF\ (((2^2)^2)^2)$ $\quad GF\ ((2^2)^2)$: $x^2 + x + \lambda$ (2.10)

There is a different way to represent the field element depending upon the basis for $GF(2^m)$, either using a polynomial basis or the normal basis [12], [25], [21], [26], [22] and [27].

A polynomial basis is a basis of the form $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$ where α is the root of an irreducible polynomial p(t), of degree m, with coefficients from GF(2). For normal basis, the field element is $\{\beta, \beta^2, \beta^4, ..., \beta^{2m-1}\}$ linearly independent.

A research report focusing on power consumption of SubBytes was done by Liang et al. [5] using a Sense Amplifier Based Logic (SABL), which operates with constant power consumption. There is a way to utilise the SubBytes operation effectively, which can save a substantial amount of area by sharing the multiplicative inverse step for SubBytes and InvSubBytes operation. Next, considering the S-box design methodologies reported so far, only [20], [28] and [5] evaluated the performances of the S-box using a full custom-design technique.

2.3.1.2 ShiftRows/ InvShiftRows Transformation

ShiftRows is a row-wise linear cyclic byte-shift operation on 4-byte data blocks of the *State* matrix with different offsets (*0* to *3*-bytes respectively for the 1^{st} to the 4^{th} row) to the left. The inverse of this transformation is computed by performing the corresponding rotations to the right. ShiftRows and InvShiftRows operations involve the whole 128-bit operation.

Most of the ShiftRows operations are implemented by using thirty-two (32) 8-bit registers, or straightforward implementation, which requires a simple wiring between two 128-bit registers and consumes a large area in AES cipher. It is also unsuitable for small datapath architecture. In [29], they utilised two 128-bit shift registers for ShiftRows and InvShiftRows operations in dedicated FPGA. This can be simplified using one 128-bit, one 96-bit and one 32-bit 4 to 1 multiplexer [29].

Hua and Jianzhou [30] proposed a compact hardware implementation for a 32-bit datapath ShiftRows and InvShiftRows operation, which requires sixteen 8-bit registers and seven 8-bit multiplexers. It requires fewer hardware resources compared to Morioka and Satoh [12] and Mangard et al. [31]. Later, byte permutation for ShiftRows/ InvShiftRows in [32] was proposed, which is suitable for 8-bit datapath implementation using twelve 8-bits registers, but they require another four 8-bit registers to process and maintain the remaining four bytes of *State*.

2.3.1.3 MixColumns/ InvMixColumns Transformation

The MixColumns transformation performs a polynomial multiplication over GF (2^8) modulo x^4+1 of a 4 byte x 4 byte matrix, generated by 4 right cyclic rotations (*0* to *3* byte locations) of the coefficients of the constant polynomial $c(x)=\{02\}_{16}x^3+\{03\}_{16}x^2+\{01\}_{16}x+\{01\}_{16}$ and each column vector for the *State* matrix. The 4 bytes in each column of the *State* are considered as coefficients of polynomials over GF (2^8) modulo x^4+1. This leads to transform columns of the *State* matrix [3] given by:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} \{02\}_{16} & \{03\}_{16} & \{01\}_{16} & \{01\}_{16} \\ \{01\}_{16} & \{02\}_{16} & \{03\}_{16} & \{01\}_{16} \\ \{01\}_{16} & \{01\}_{16} & \{02\}_{16} & \{03\}_{16} \\ \{03\}_{16} & \{01\}_{16} & \{01\}_{16} & \{02\}_{16} \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \tag{2.11}$$

where $S'_{0,c}$, $S'_{1,c}$, $S'_{2,c}$ and $S'_{3,c}$ are the output *State* after MixColumns and $S_{0,c}$, $S_{1,c}$, $S_{2,c}$ and $S_{3,c}$ are the original *State*.

For the InvMixcolumns for decryption, the same procedure is followed using the constant polynomial [3]

$$c'(x)=\{0E\}_{16}x^3+\{0B\}_{16}x^2+\{0D\}_{16}x+\{09\}_{16}. \quad (2.12)$$

The Xtime based implementation is a byte-level method that uses the xtime(x) functional block to perform the constant multiplication by {02} in GF(2^8) and to realise other constant multiplications based on xtime(x) function. Xtime(x) function performs a multiplication by x in finite field in GF(2^8). The byte-level substructure sharing method finds common terms, with the aim to reduce the number of xtime blocks by pre-computing additions of some variables before using the xtime unit.

Numerous numbers of architectural methods have been proposed for efficient computation of MixColumns/ InvMixColumns, including substructure sharing methods for byte-level and bit-level optimisations. The bit-level substructure sharing method is to extract two-term common factors from the bit-level expressions in MixColumns and InvMixColumns by selecting the factors that occur most frequently. Xinmiao and Parhi [33] implement MixColumns and InvMixColumns using 193 XOR gates with 7 XOR gates in the critical path, while Fischer et al. [34] use the decompositions resulting in 192 XOR gates for each column, with 7 XOR gates in the critical path. In [35], Hua and Friggstad proposed an architecture of MixColumns and InvMixColumns with 324 XOR gates with 6 XOR gates in the critical path. Other implementations of AES MixColumns and InvMixColumns can be found in [9], [36], [37] and [38].

2.3.1.4 Add Round Key Transformation

The AddRoundKey process is a simple bitwise XOR operation on the 128-bit round keys and the data in each round of the AES process. Round keys are generated by a Key Scheduling process. The initial round key is derived from the original initial key, and the next round key generated by the XOR operation between two previous columns. For columns that are in multiples of four, the process involves an addition process with round constant, S-box and rotation. In the decryption mode, the operation

is reverted, where the initial round key is the last round key from the encryption process.

2.4 Mode of operation

A mode of operation is a technique for enhancing the effect of a cryptographic algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. NIST has defined five confidentiality modes of operation for AES in Special Publication 800-38A [39]. These five modes can be divided into two major categories [3]. There are:

a. Non-feedback modes: Electronic Codebook Mode (ECB) and Counter Mode (CTR)
b. Feedback modes: Cipher-block Chaining (CBC), Cipher feedback (CFB) and Output feedback (OFB)

An initialisation vector (IV) is a block of bits that is used by several modes to randomise the encryption and to produce cipher texts, even if the same plaintext is encrypted multiple times.

2.4.1 Electronic Codebook Mode (ECB)

ECB is the simplest mode in AES. The message is divided into blocks, and each block is encrypted separately. Identical plain texts will produce identical outputs when encrypted with the same key. In ECB encryption and ECB decryption, multiple forward cipher functions and inverse cipher functions can be computed in parallel or pipeline. Only a single block of data will be affected if any encryption error occurs. The ECB mode is shown in Figure 2.2.

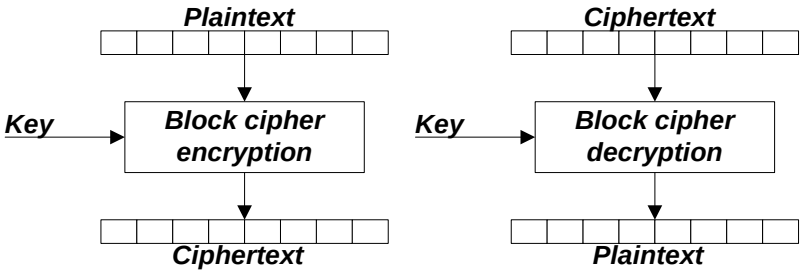


Figure 2.2 *Electronic Codebook (ECB) mode for Encryption and Decryption [39]*

2.4.2 Counter Mode (CTR)

The CTR mode turns a block cipher into a stream cipher. A set of input data groups, called counters, are used to produce a set of keystreams that are XORed with the plaintext to produce the cipher text, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. In CTR encryption and CTR decryption, only the forward cipher function is invoked on each counter group: there is no inverse cipher function. The resulting key streams are XORed with the corresponding plaintext or cipher text blocks to produce the ciphertext or plaintext blocks. Errors in encryption effect the current block and the next block, which the cipher will correct itself after that. The forward cipher functions can be performed in parallel and pipelined. The CTR mode is shown in Figure 2.3.

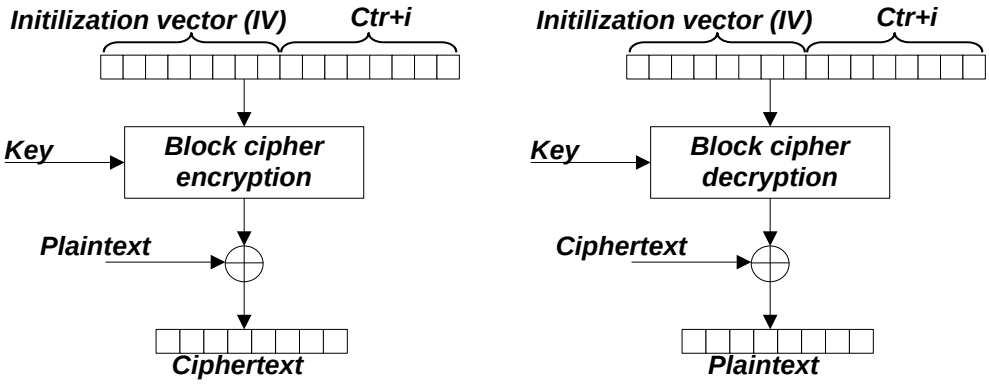


Figure 2.3 *Counter Mode (CTR) mode for Encryption and Decryption [39]*

2.4.3 Cipher-block Chaining mode (CBC)

In the cipher-block chaining (CBC) mode, each block of plaintext is XORed with the previous cipher text block before being encrypted. Each cipher text block is dependent on all plaintext blocks processed up to that point. An initialisation vector is used in the first block to make each message unique. The drawback of this mode is that the encryption/ decryption processes cannot be parallel, and that the message must be padded to a multiple of the cipher block size. Figure 2.4 shows the CBC mode for encryption and decryption.

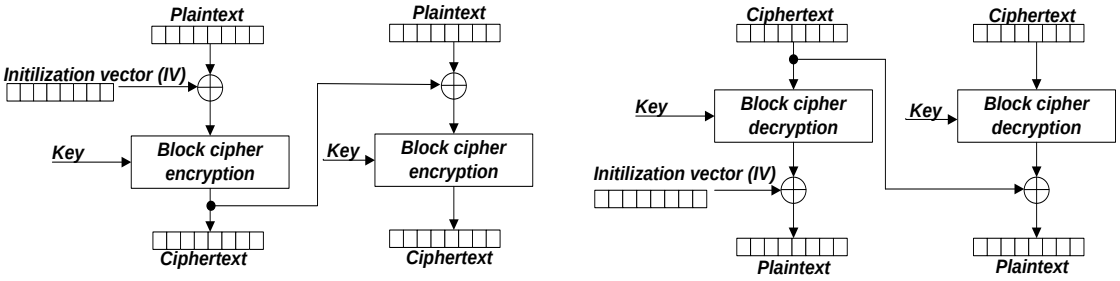


Figure 2.4 *Cipher-block Chaining (CBC) mode for Encryption and Decryption [39]*

2.4.4 Cipher Feedback mode (CFB)

CFB mode turns a block cipher into a self-synchronising stream cipher. This mode does not require the plain text to be padded to the block size of the cipher. The feedback of successive cipher text segments into the input blocks of the forward cipher generates output blocks that are exclusive-ORed with the plaintext to produce the cipher text, and vice versa. The CFB mode requires an initialisation vector (IV) as the initial input block as shown in Figure 2.5.

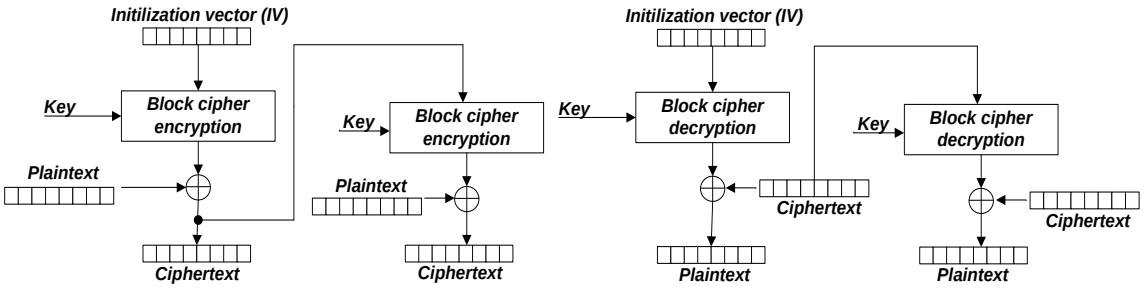


Figure 2.5 *Cipher Feedback (CFB) mode for Encryption and Decryption [39]*

2.4.5 Output Feedback mode (OFB)

The output feedback (OFB) mode makes a block cipher into a synchronous stream cipher. The iteration of the forward cipher on an IV generates key stream blocks, which are then XORed with the plaintext blocks to get the cipher text, and vice versa. Each output feedback block cipher operation depends on all previous ones. The OFB mode is shown in Figure 2.6.

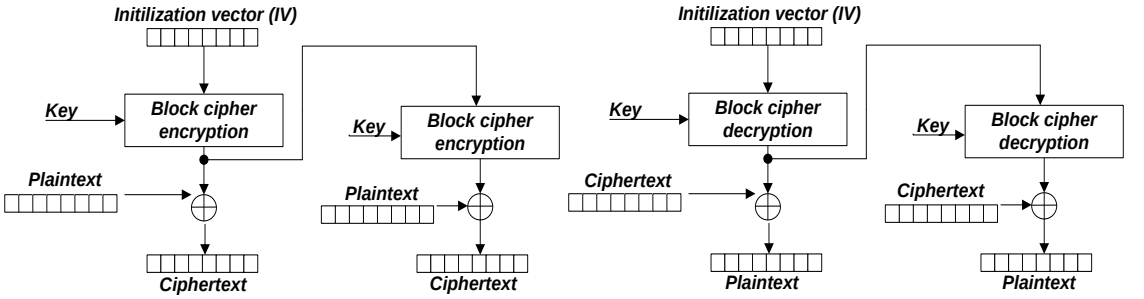


Figure 2.6 *Output Feedback (OFB) mode for Encryption and Decryption [39]*

2.5 Design Evaluation Criteria and Specifications

The key considerations for AES front-end design are power consumption, area, speed of operation and delay. These criteria characteristics are represented in the detailed evaluation and specifications of each parameter. An overview and description for these design criteria are detailed below to give a better understanding for further design methodology.

2.5.1 Speed of operation

Two parameters that characterise the speed of the AES system implementations are throughput and latency. Throughput denotes the speed of the encryption process. It can be defined as the number of encrypted or decrypted bits in a unit of time, or its average number of bits processed per second [40]. Throughput is expressed in bit per second (bps). For AES, the throughput can be calculated as:

Throughput = 128 bits/ (Cycles per Encrypted Block * Time period) (2.13)

where cyles is the clock cycle per byte to encrypt a block of data and time period is the reciprocal of the frequency clock.

Parallelism in the encryption algorithm and pipelining could be used to achieve higher speed in the circuit. Parallelism allows certain transformations to perform simultaneously. Pipelining divides the processing unit into several stages, which execute sequentially that increase throughput by overlapping (parallelising) the operation of the various pipeline stages.

Latency is defined as the time needed to complete the encryption or decryption process of a single data block and is expressed in the number of clock cycles. It is calculated from the time a block of data enters the encryption unit, until it leaves the unit. The latency can be calculated as:

Latency = clock cycle/ f_{clk} (2.14)

where f_{clk} is the frequency of clock.

The total latency of a system is a sum of latencies of all modules' processing data sequentially. Higher operating frequencies can improve the performance of the system.

2.5.2 Power Consumption

The total power dissipation budget for any CMOS implementation consists of static power and dynamic power.

$$P_{total} = P_{static} + P_{dynamic} \quad (2.15)$$

The total power consumption of an AES crypto chip is:

$$P_{total} = P_{AES}$$
$$= P_{SubBytes} + P_{ShiftRow} + P_{MixColumn} + P_{AddroundKey} + P_{Keyschedule} + P_{others,} \quad (2.16)$$

where P_{others} = power consumption of registers [41].

Static power dissipation is due to the sub-threshold currents in the weak inversion cut-off states of the metal-oxide semiconductor field-effect transistor (MOSFET), and the leakage currents in the reverse-biased source, and drain p-n junction diodes. The leakage current depends on the specific fabrication technology. Figure 2.7 shows the sources of static leakage dissipation. Dynamic power dissipation occurs during the charging and discharging of the capacitive load (C_L) during logic state transition. It is the sum of the short circuit dissipation due to simultaneous conduction of the nMOS and pMOS devices, and the load capacitance switching dissipation.

Dynamic switching dissipation is dominant in CMOS circuitry [42], and is given by:

$$P_{swiching} = \alpha f_{CLK} C_L V_{DD}^2 \quad (2.17)$$

where α is the activity factor, f_{CLK} is the frequency of operation, and V_{DD} is the supply voltage.

C_L is the capacitance that loads the output node and can be defined as: $C_L = C_{internal} + C_{interconnect} + C_{load}$, with $C_{internal}$ being the internal capacitance of the output device associated with its drain, $C_{interconnect}$ being the interconnect capacitance, and C_{load} being the input capacitance associated with the loading logic gates. Capacitances at logic outputs in the S-box thus have to be reduced by proper loading, and an interconnect-minimised logic design, in addition to considering supply voltage scaling and throughput constraints. Hence, one technique to reduce power consumption is to reduce

the switching activity within the AES circuit logic, if the supply voltage and the IC technology are fixed.

On the other hand, if the supply voltage is scalable, then an efficient technique to save power is to decrease the supply voltage. As the transistor geometries are reduced, a smaller supply voltage is needed to avoid electrical breakdowns and to obtain the needed performance. However, the threshold voltage has to be scaled to satisfy high-performance requirements in case of reduced supply voltage, results in an exponentially increasing leakage current, which can be detrimental to the integrity of the AES implementation.

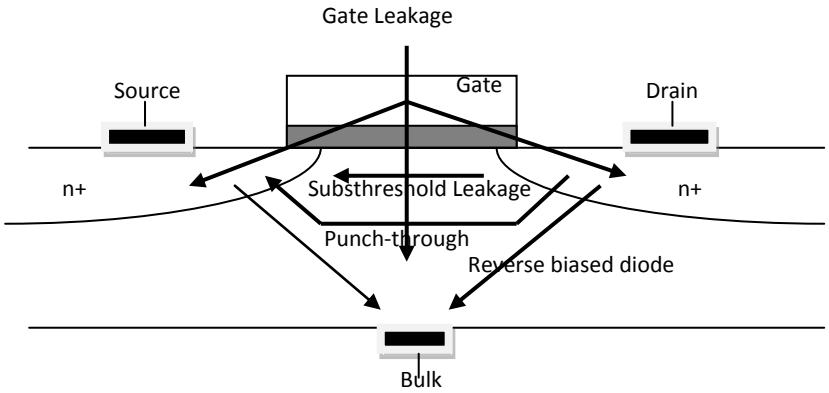


Figure 2.7 *Static CMOS leakage sources*[42]

2.5.2.1 Design techniques for low power system

There are a few techniques to reduce the power consumption of the AES crypto-processor. Firstly, by selecting nanometer CMOS technology, and secondly through the appropriate ultra-low voltage operation. Total power can be minimized through a careful selection of supply and threshold voltage and device sizes so that the leakage and switching components of the dissipation are equal.

2.5.2.1.1 CMOS technology selection

Selecting an advanced CMOS technology can contribute to the static and dynamic power, especially for high-performance applications. As the technology scales down, the static and dynamic power decreases due to the reduction of load capacitances. By using an advanced technology, there is a reduction in the total silicon area occupied by the circuit, which is proportional to increasing the overall functional density of the chip.

The trade-off of using nanometer CMOS technology is the increasing power leakage due to the inherent reduction of the threshold voltage, Vt and gate oxide thickness, Tox. Sub-threshold leakage is the main component of the leakage power and is caused by current flowing through a transistor, even when it is supposedly turned off. In advanced technology nodes such as 130nm, there are Low Power (LP) and general purpose (GP) flavours developed for different power requirements. This technology also provides different threshold voltage devices, high-Vt and low-Vt, to achieve differing performances while considering the leakage power. There are several techniques to reduce the leakage power, such as adopting a low leakage transistor with high-Vt on the non-critical path, which offers less leakage current but reduces the speed. A high-Vt transistor is used to gate the circuit power supply to significantly decrease sub-threshold leakage, and a low-Vt transistor is used to preserve performance by providing a fast speed, although it has high leakage due to the formulation of leakage power:

$$P_{leakage}=N*V*e^{-Vt} \quad (2.18)$$

where N = number of transistors, V= voltage and V_t = threshold voltage.

The method to select an appropriate transistor to reduce leakage power involves determining the threshold gate input values during the idle state, using the low-Vt transistor for transistors that turn on during idle state, and replacing the turn-off transistor with the minimal number of high-Vt transistors as close as possible to power/ ground rail.

2.5.2.1.2 Ultra-low voltage operation

Another approach to reduce power consumption is by using a voltage scaling strategy with aggressive voltage reduction. Scaling down the supply voltage will lower the switching power without loss in throughput, by reducing the threshold voltage of the device. Delay drastically increases as V_{DD} approaches the threshold voltage of the device, while dynamic power is proportional to the square of the supply voltage:

$$\text{Td} = \frac{C_L \times V_{DD}}{I} = \frac{C_L \times V_{DD}}{\mu C_{ox}(W/L)(V_{DD}-V_t)^2} \quad (2.19)$$

where C_L = load capacitance, V_{DD}= voltage supply, V_t= threshold voltage, (W/L) = aspect ratio, where W is the width of transistor channel and L is length of transistor channel, μ = surface mobility of electrons in channel and I= saturation current.

2.5.2.1.3 Low power design

Considering low-power implementation, the authors in [43] claim to be the first submicrowatt designers of an AES encryption with power dissipation of 692 nW, by using only 356 cycles. This design uses a 100 kHz clock, a core voltage of 0.75 V and has a die area of 21000 square μm. However, a full custom design of an S-box with a low-power consumption of less than 100 nW can be found in [44]. Other reported work focusing on power consumption of SubBytes was carried out by Liang et al. [5] using a Sense Amplifier Based Logic (SABL), which operated with a constant power consumption.

2.5.3 Delay

A maximum delay path is called a critical path, and is a delay between the output node of the critical path and the gate node, causing the state change of the output node. It represents a signal line that executes the longest calculation operation in a calculating circuit, within a cycle of a synchronous signal. It is proportional to the operating frequency; shortening the critical path will increase an operating frequency. The delay of a circuit is:

$$t_d = max_n\{d(c_n)\} \quad (2.20)$$

where $d(c_n)$ is the delay of the n-th critical path comprising the circuit.

The delay of the circuit is obtained by calculating the delay of each critical path, and calculating the maximum of these delays.

2.5.4 Area

Depending on implementation technology, either ASIC or FPGA technology, the size of the circuit can be expressed in different ways. For ASIC technology, area is defined as the size of a die in square μm, logic gates, or number of transistors. Size of silicon can be optimized by using a small design area. Die cost is proportional to the fourth or higher power of the die area [45]. Reduction of the circuit sizes can reduce costs of system.

In FPGA technology, area is expressed as the number of configurable logic blocks (CLB). Proper circuit minimisation can decrease the area and increase the speed. Another method is using resource sharing and resource reusing methods with integration of the Encrypter and Decrypter system to reduce the die size, and also to reduce the power consumption overhead of these blocks. Resource sharing is done by combining both transformation and inverse transformation. It also means utilising one unit and performing all computations in different clock cycles.

2.6 AES architecture

2.6.1 AES 128-bit datapath

An AES 128-bit datapath is typically designed to meet a high throughput performance. This architecture offers the greatest degree of parallelism to increase the concurrency of an AES computation. Qingfu and Shuguo [46] propose a high throughput 128-bit

datapath AES that supports both encryption and decryption algorithms, with 128-bit, 198-bit and 256-bit cipher keys in 0.18μm CMOS technology. The iterative architecture adopted consists of an on-the fly key expansion unit and achieves a throughput of 1.16 Gbps with 19476 gate equivalent (GE).

Chih-Pin et al. [47] reduced the area of an S-box using a basis transformation technique in order to reduce the total area of AES design. They implemented a pipeline to achieve a high throughput of 2.977 Gbps at 250MHz clock in 0.25um CMOS process with a gate count of 63.04k gates. Lan and Luke's [48] AES design was designed in 0.18μm CMOS, to support the encryption and decryption algorithm; a throughput of 1Gbps was achieved and the core area was 2.6 square mm. Mayhew and Muresan [49] report an AES encryption design using 0.18μm CMOS technology with a circuit area of 2.45 square mm and power consumption of 200 mW at 25MHz.

2.6.2 AES 32-bit datapath

Many researchers have considered the 32-bit datapath in ASIC implementation to satisfy the requirement of low power. One of the examples can be found in Satoh et al. [4] who proposes AES encryption on a 0.11μm technology, with a throughput of 311Mbps at 130MHz in 54 clock cycles, and hardware complexity of 5.4k gates consisting of a core path and a key generator. An AES fully loop-unrolled design without pipeline is presented in [50] using 0.36μm CMOS technology, utilising 612k gates and a throughput of 1.95 Gbps.

Mangard et al. [31] report a high throughput design in a 0.6μm CMOS process, offering a throughput of 128 Mbps and 241 Mbps in 34 cycles for encryption and decryption, respectively. The architecture consists of 16 data cells and 16 S-boxes with a pipelined implementation of the S-box. In [51], they implement an AES design in 0.18μm technology consisting of 173k gates offering a performance rate of 1.82Gbps. Pramstaller et al. [52] report three versions of a 32-bit datapath implementation, which cover the high, standard and minimum standard implemented in 0.6um technology. The design consumes 15850 GE for high performance, 11205 GE for standard and 8541 GE

for the minimum version. The throughput for high performance offers 183 Mbps, 98 Mbps in the standard version and 70 Mbps in the minimum version. The overall performance of this design is better than in [12], in terms of the critical path, which consists of only one pipeline S-box stage.

Shin-Yi and Chih-Tsun [53] propose a high throughput, low power 32-bit AES crypto-processor, implemented using 0.13μm CMOS process that comprised 86.2k gates and had power consumption of 40.9mW at 333MHz clock frequency. This design also implements various modes of operation: ECB, CBC and CCM modes with maximum throughput of 4.27 Gbps. Another high throughput design is discussed in [54] which gave a maximum throughput of 10.656 Gbps at 333MHz in 0.18μm CMOS technology. The proposed design implements fully pipelined architecture using an efficient finite field GF(2^2) in MixColumns/ InvMixColumns operations, shared with ShiftRows operation to reduce the area of the system.

2.6.3 AES 8-bit datapath

Recently, the 8-bit AES implementation has been employed for low-power and small-area systems in FPGA and ASIC implementation. The first fabricated ASIC design of 8-bit architecture was done by Feldhofer et al. [38]. It uses 0.35μm CMOS for 8-bit implementation, using one combinational logic S-box and shift-register based MixColumns architecture. The design takes up a 0.25 square mm area, equivalent to 3400 gates, with throughput of 9.9 Mbps at 80MHz. For low throughput, it operates at 100 kHz and 1.5V with a power of 4.5μW and current of 3.0μA. This design has been enhanced by adding decryption functionality in [9]. Hamalainen et al. [32] employ 8-bit datapath that support 128-bit keys, and use a byte permutation unit for ShiftRows transformation. The throughput is 121 Mbps at 153MHz, and is higher than in [9], but it does not support a decryption algorithm.

In [55], the 8-bit AES was proposed by Tim Good using an 8-bit Application-Specific-Instruction-Processor (ASIP) in FPGA. It was designed for 128-bit key encryption/ decryption. It has 15 instruction programs to control the AES, which are stored in two blocks of RAM.

The cycle count is 13546 in order to perform the encryption, including the key expansion. Another 8-bit AES encryption design proposed in FPGA implementation is found in [56]. It uses an efficient MixColumns architecture operated at 16 cycles in parallel with the AES core operation.

The recent design from Good and Benaissa [43] reports a low power and area 8-bit design where the average core power is 692nW using 0.75V, 100 kHz. It uses 356 cycles for encryption and 21000 square μm. Dalmisli and Ors, [57] report an 8-bit implementation to perform encryption with a 14.3 Mbps rate and 4.3k gate equivalent using one S-box and quarter MixColumns in the design. In [58], they propose an AES crypto-processor that supports an encryption and decryption algorithm in 0.18um CMOS technology, with low power consumption of 49μW/MHz at 128MHz. This AES core consumes 5.6k gates and has a throughput of 203 Mbps.

A compact and ultra low power AES coprocessor is presented by Hocquet et al. [59] using ultra-low power techniques. The design is implemented in a 65nm CMOS processor utilising 3.5k GE and offering a low power consumption of 0.21μW and 0.85μW at 31kbps/ 0.35V and 100kbps/ 0.4V, respectively.

Table 2.2 summarizes the different AES datapath architecture; 128-, 32- and 8-bit in term of area, throughput and power consumption.

2.1 Conclusions

This chapter reports an introduction to the basic concept of AES implementation architectures including the explanation on the mathematical concept used in AES, description of each AES transformation and mode of operation. The design requirements, criteria and techniques for design methodology consideration also have been discussed to achieve the objective of this research. Finally, previous existing works describe the different parameters and architecture has been reviewed.

Table 2.2 *AES architecture of different datapath*

Datapath(bit)	Year	Technology (µm)	Author	Area (kgate)	Throughput (Mbps)	Frequency (MHz)	Power (mW)	Mode
128	2009	0.18	[46]	19476 GE	1160	100	NA	Enc/ Dec
	2003	0.25	[47]	63.04	2977	250	NA	Enc/ Dec
	2003	0.18	[48]	2.6 mm^2	1000	100	110	Enc/ Dec
	2009	0.18	[49]	2.45 mm^2	NA	25	200	Enc
32	2001	0.11	[4]	5.4	311	130	NA	Enc
	2000	0.36	[50]	612	1950	NA	NA	Enc/ Dec
	2003	0.6	[31]	NA	128	NA	NA	Enc
				NA	241	NA	NA	Dec
	2002	0.18	[51]	173	1820	NA	NA	Enc/ Dec
	2005	0.6	[52]	15850 GE	183	NA	NA	Enc/ Dec
				11205 GE	98	NA	NA	Enc/ Dec
				8541 GE	70	NA	NA	Enc/ Dec
	2007	0.13	[53]	86.2	4270	333	40.9	Enc/ Dec
	2011	0.18	[54]	NA	10656	333	NA	Enc/ Dec
8	2004	0.35	[38]	3.595	9.9	80	NA	Enc
					NA	0.1	4.5	
	2005	0.35	[9]	3.4	9.9	80	3.6	Enc/ Dec
	2006	0.13	[32]	3.1	121	153	5.6	Enc
	2006	FPGA	[55]	242 slices	2.3	70	NA	Enc/ Dec
	2004	FPGA	[56]	337 slices	53	110	NA	Enc/ Dec
	2010	0.13	[43]	5.5	0.036	0.1	6.92	Enc
	2009	FPGA	[57]	4.3	14.3	142.8	43	Enc
	2010	0.18	[58]	5.6	203	128	49	Enc/ Dec
	2011	0.065	[59]	3.5	0.031	0.322	0.21	Enc/ Dec
				3.152	0.1	0.89	0.85	Enc/ Dec

*GE implies Gate Equivalent, NA implies not available

CHAPTER 3

NOVEL XOR GATE LOW-POWER LOW-VOLTAGE FULL SWING OUTPUT VOLTAGE FOR CMOS GALOIS FIELD ARITHMETIC

3.1 Introduction

Low-power design has become a major design strategy due to the overwhelming growth of portable applications. High power consumption raises the chip temperature and affects the circuit's reliability and has a high cooling cost. Various low power techniques have been explored to enhance the basic logic gates such as the XOR gate, which influences the overall power consumption in many SOC implementations.

AES architecture requires a large number of XOR operations whose efficient and low power implementation can result in a substantially improved CMOS AES hardware design. S-box architectures, especially the composite field approach, use the XOR gate as the fundamental logic function, along with AND gates. Consequently, enhancing the performance of the XOR gates can significantly improve the critical path performance and die area of the S-box design, and overall architecture of AES system.

Over the years, many designs of 2-input XOR gates have been reported, in order to enhance the performance of various applications [60-69]. Examples include full adder, parity generator, encryption processor and comparator, which all influence the overall power consumption in many SOC implementations. XOR gate optimisation in terms of power, speed and transistor count have significantly improved the performance of large and complex circuits. Hence, when designing an XOR gate, several related issues such as power consumption, area, noise immunity, and driving capability have to be considered.

Some research has been done to reduce the transistor count, but a threshold loss problem that causes non full swing output voltage with low noise immunity, especially when the cells are cascading for the larger circuit, has to be faced [70]. Traditional XOR gate design is based on eight static CMOS transistors which can operate with full swing output voltage, but with the drawback of acquiring large amounts of transistors [67]. On the other hand, XOR gate based on a transmission gate [71] can be used to overcome the signal degradation caused by the PMOS and NMOS devices in pass-transistor logic. It offers better quality but has the drawback of loss of driving capability, and requires a complementary signal value to control the gates of PMOS and NMOS, which adds more transistors and area.

A cross-coupled (CC) XOR gate based on the pass transistor logic is reported in [72], which claims to improve speed and power consumption, compared to the six devices pass-transistor XOR gate, and works well under a low-voltage regime. A six transistor XOR gate, realised from a modified four transistor XOR gate by cascading a standard inverter as output driver, can be found in [65], and improves the poor output signal level for certain inputs. A powerless XOR gate (P-XOR) is proposed in [66], using a four transistor circuit with no power supply connection, which consumes less power than other designs but at the expense of a large delay. Another four transistor XOR gate design is reported in [73] based on the Gate-Diffusion-Input (GDI) cell.

A three transistor XOR gate is found in [69] that uses a CMOS inverter and a PMOS pass transistor. It provides low power-delay product (PDP) but has a voltage degradation, with the input combination *A=1* and *B=0*. On the other hand, Elgamel et al. [74] also propose a three transistor XOR gate, but it consumes high power when *A=1* and *B=0*, and in addition produces a poor logic '*1*' for this input combination. However, it may reach an acceptable logic level with appropriate transistor sizing. Thus, both circuits [69] and [74] may not operate reliably at low supply voltage.

In this chapter, we proposed two novel low-power, low-voltage with full swing output voltage designs, for a 2-input XOR gate, using six transistors implemented using 65nm and 130nm CMOS technology. Also, a thorough analysis is conducted to compare the performance of the proposed XOR gate in terms of power consumption and delay to peer designs.

3.2 Novel XOR Topology

We have designed two topologies of a 2-input XOR gate, namely XOR1 and XOR2, using six transistors that provide full swing output voltage for all input combinations, and enable low-voltage operation with small propagation delay. The XOR1 is a further low-power constrained novel 2-input XOR gate using six transistors, including an inverter that provides full swing output voltage for all input combinations, and enables low-voltage operation with small propagation delay.

Both of the proposed XOR circuits are based on the concept of pass transistor logic and inverters for complementary input. The pass transistor design enables a small transistor count, along with smaller input loads (with signal input to source/ drain) offering very low-power operation with high-performance. Some of the signal also connected to transistor gate as to operate the XOR gate properly. This is because signal input at the source/ drain requires the charging and discharging of the source/ drain-to-body depletion capacitance, which is usually smaller than the gate-to-source parallel plate capacitance, in case of signal inputs at the gate. Since a NMOS device passes a strong '*0*', but a weak '*1*' , while a PMOS device passes a strong '*1*', but a weak '*0*', the complementary pass transistors are organised to pass a strong output logic level for all input combinations of '*1*' and '*0*'. The transistor sizes are carefully chosen for optimum power- delay performance under various operating conditions. With respect to the transistor sizing, all are designed with minimum gate length. Figure 3.1 and 3.2 show the schematics of the proposed XOR gate circuits, XOR1 and XOR2, respectively. They perform a perfect full swing output voltage operation for every input pattern.

For the XOR1 design, the output *Y* generates '*0*' corresponding to *A=B=0*. For this condition, transistors M1, M2 and M4 are ON, but the transistor M4 will pass a strong '*0*' to the output *Y*. When *A=0*, *B=1*, transistors M2, M3 and M4 are ON and a '*1*' is generated at the output *Y* with transistor M2 pass the strong '*1*'. For *A=1*, *B=0*, only the device M1 is ON, and a strong '1' is passed to the output *Y*. With *A=B=1*, only transistor M3 is ON and a strong '*0*' is passed to the output. This pass transistor XOR thus does not suffer from signal level deteriorations like other pass-transistor XOR gates.

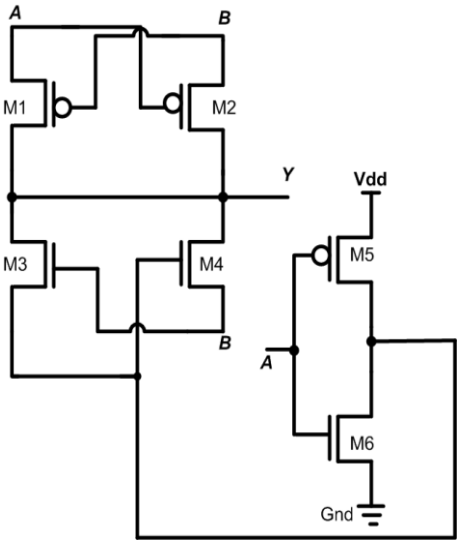


Figure 3.1 *Proposed XOR1 circuit for low-power Galois field arithmetic*

For the proposed XOR2 design, Vdd connections to the source terminals of M3 and M4 are used to drive a full swing output voltage of '*1*'. For the *A=B=0* condition, transistor M3 is ON, passing a strong '*1*' from Vdd to the inverter input, which generates a '*0*' at the output *Y*, to turn ON M4. When *A=0, B=1*, transistors M2 and M3 are ON, which passes the signals '*0*' and '*1*', respectively. To solve this problem, the aspect ratio (W/L) of M2 is increased making it larger than the aspect ratio of M3, so as to pass only a signal '*0*' from M2 to the inverter input and generate the strong output *Y=1*. Only aspect ratio of M2 is increase to pass a signal '0' to the inverter and the W/L has been chosen carefully and aspect ratios of other transistors are kept small to balance within the power and speed of the XOR gate.

For *A=1, B=0*, only the device M1 is ON, and a strong '*0*' is passed to the inverter generating a full swing output voltage '*1*' to output *Y*. With *A=B=1*, transistors M1 and M2 are ON and a weak signal '*1*' is passed to the inverter input and as a result the output Y will also be degraded. However, the feedback path causes transistor M4 to turn ON when *Y=0*, thus passing the perfect signal '*1*' from Vdd to the inverter input, resulting in the generation of the perfect signal '*0*' at *Y*.

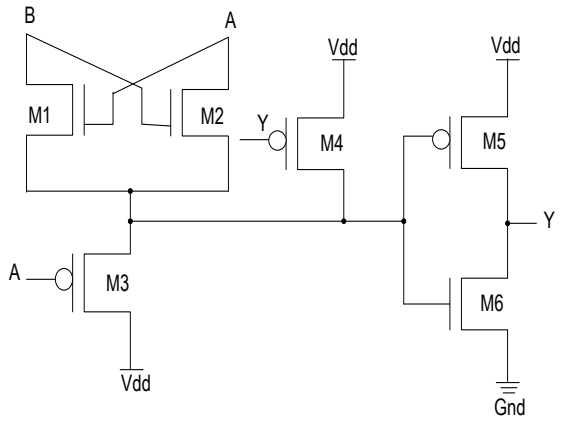
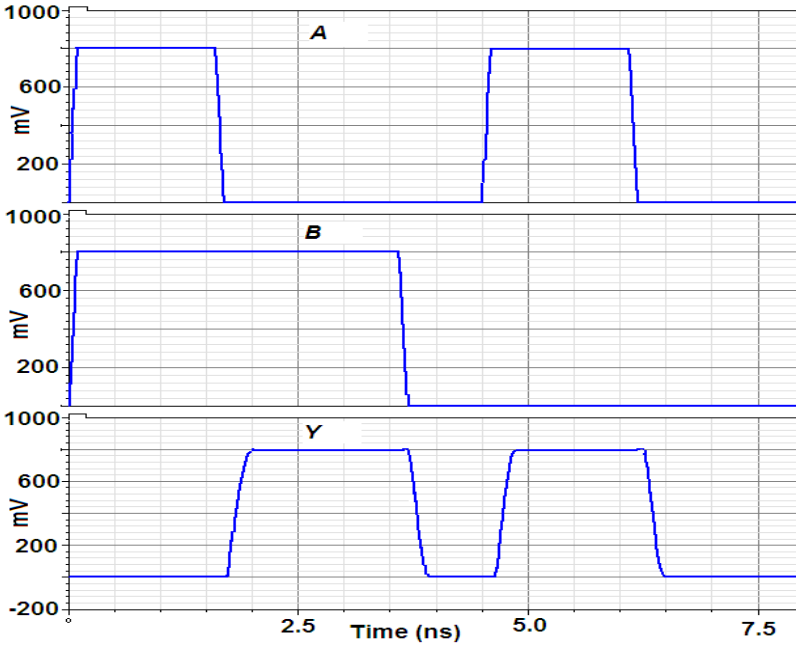


Figure 3.2 *Proposed XOR2 gate circuit*

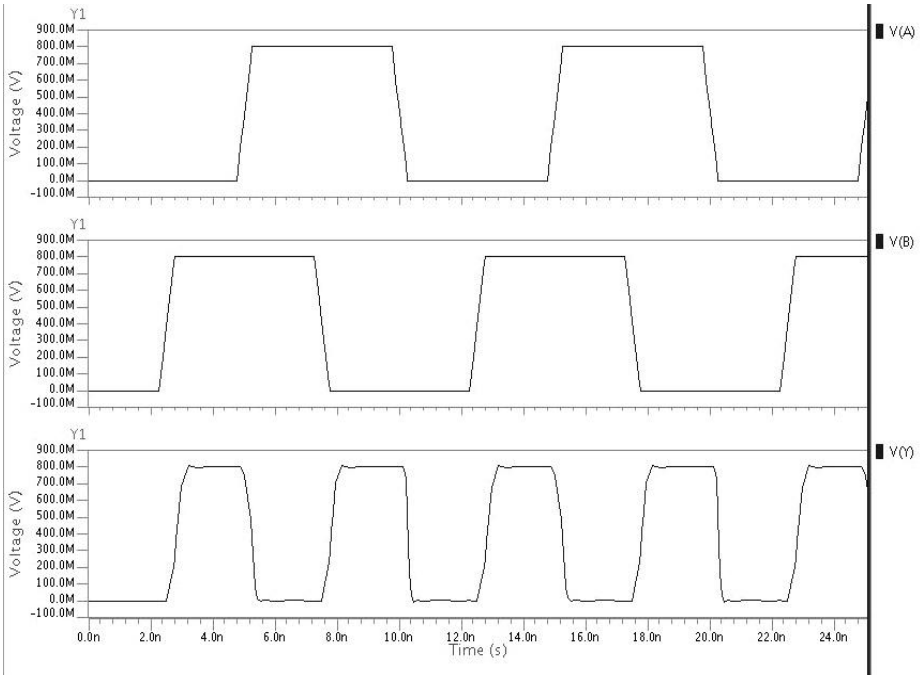
3.3 Performance and simulation results in 65 nm CMOS

Both of the proposed XOR circuits are implemented in 65nm IBM CMOS technology. Comparative studies and extensive simulations on both of the proposed XOR gates, and several previous XOR circuit topologies found in literature, have been realized and simulated in order to analyze the performance comparison with both the proposed XOR gates. The simulations were carried out using the same 65nm IBM CMOS technology on a Cadence Spectre platform, to measure the propagation delay and the power dissipation. All the simulations were carried out using the same testing conditions, with a 0.6V to 1.2V supply voltage range, a load capacitance of 50fF, and a throughput (clocking) rate of 500MHz. In order to have a closer analysis, several input patterns were applied to cover all the input cases, and simulation results verified the correct functionality for every input combination with a supply voltage of 0.8V.

Figure 3.3 shows the simulated full swing output voltage operation of the pass-transistor XOR gate verified by Cadence Spectre. Table 3.1 summarises the results of these simulations in 65nm CMOS technology, which show the propagation delay, the power dissipation, and the power delay product (PDP) for various topologies. Power consumption is obtained by averaging over 10 inputs of transition. Propagation delay is evaluated from 50% of the voltage level of input to 50% of the voltage level of output. PDP is calculated from production of worst case delay and average power consumption.



(a)



(b)

Figure 3.3 *Full swing output voltage of the proposed (a) XOR1 gate (b) XOR2 gate*

Table 3.1 *Comparison of simulation results for the XOR gate in 65nm CMOS technology*

	V (v)	Proposed XOR1 6T [75]	Proposed XOR2 6T [76]	4T [65]	3T [69]	6T [65]
Delay (ns)	0.6	1.69	2.81	2.656	3.08	5.006
	0.8	1.585	2.75	2.994	2.061	3.251
	1.0	1.562	2.642	2.094	2.045	2.111
	1.2	1.553	2.099	2.068	2.035	2.075
Average power (fW)	0.6	1.82	2.312	4.189	8.703	14.66
	0.8	3.256	6.069	7.613	16.25	89.57
	1.0	15.015	22.77	12.12	25.36	238.9
	1.2	17.144	25	18.81	34.96	463.3
PDP (yJ)	0.6	3.076	6.497	11.13	26.81	73.36
	0.8	5.161	16.679	22.79	33.49	291.2
	1.0	23.453	60.158	25.38	51.86	504.2
	1.2	26.625	52.475	38.91	71.15	961.5

The simulation results show that the proposed XOR1 gate has the lowest propagation delay, as well as the lowest power consumption against voltage scaling except for the case in four transistor in [65] when V=1.0V, along with high output driving capability in low power supply operations, as shown in Figure 3.4 and Figure 3.5. The delay of the proposed XOR2 gate is between the four transistor XOR gates in [65], and three transistors in [69], but is less than the six transistors in [65] when supply voltage less than 1.0V. The worst circuit in terms of speed is the six transistor XOR gate in [65] when supply voltage is less than 1.0V.

Power consumption increases as the supply voltage is increased. The proposed XOR2 circuit gives a lower power dissipation at low voltage, compared to the other designs in 4T [65] and 3T [69], which are slightly more than [65] in high voltage, but less than [69]. But when compared to 6T in [65], it consumes less power.

Based on Table 3.1 and Figure 3.6, it demonstrates that the overall PDP for both proposed circuits has improved more than 50% from other design in [65] and [69]. Compared to the design in [65] and [69], the power-delay product of both proposed XOR gates is less at a supply voltage between 0.6V and 0.8V. The PDP of six transistors in [65] is the highest of other designs, including the proposed design. The improvements attained by the proposed circuit are thus clearly evident when compared to these other XOR gate circuits when supply voltage is less than 1.0V.

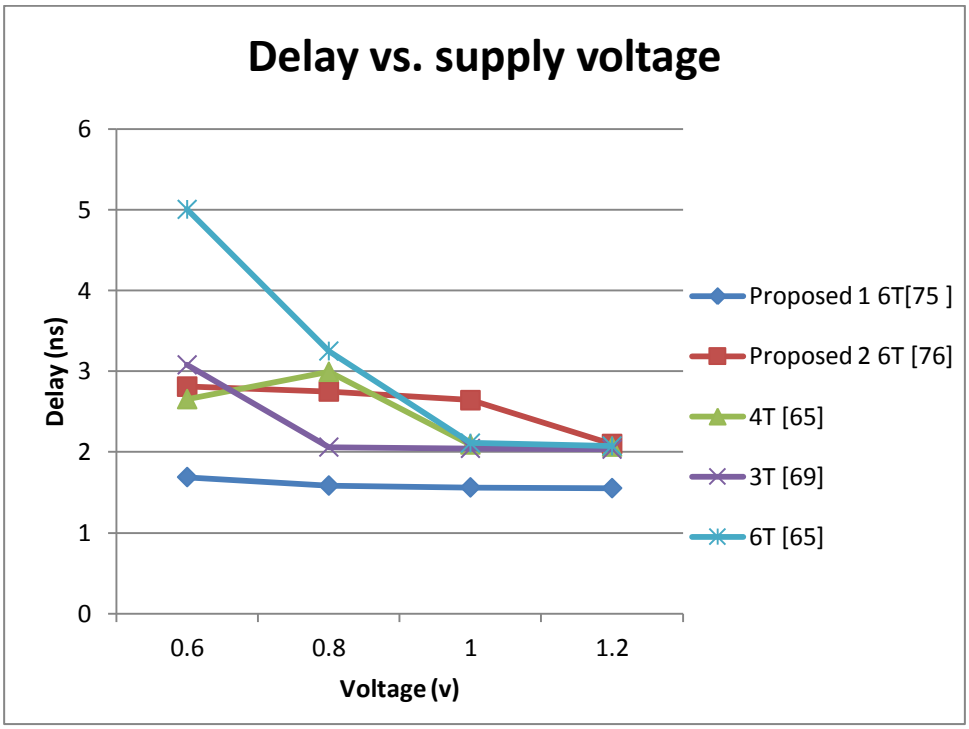


Figure 3.4 *Propagation delay with supply voltage scaling for different XOR gates in 65nm CMOS technology.*

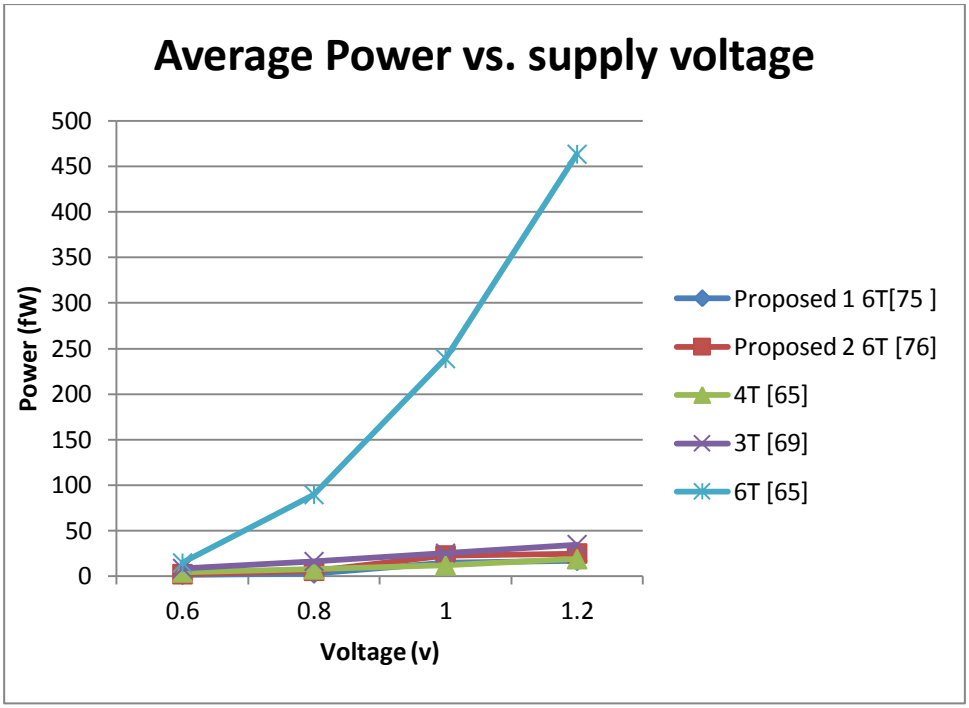


Figure 3.5 *Power dissipation with supply voltage scaling for different XOR gates in 65nm CMOS technology.*

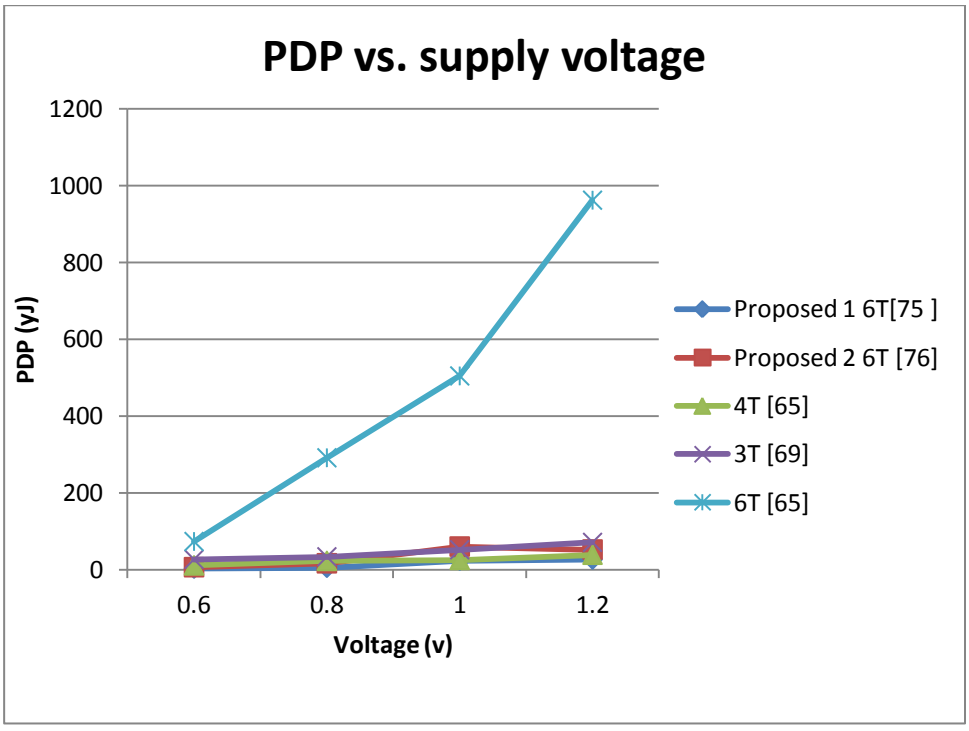


Figure 3.5 *Power-delay product (PDP) with supply voltage scaling for different XOR gates in 65nm CMOS technology.*

3.4 Performance analyses and simulation results in 130nm CMOS

Extensive simulation of both proposed XOR gates along with five other existing XOR gates found in literature was carried out using the 130nm IBM CMOS technology in order to analyse the performance comparison. Both of the proposed XOR circuits, and previously reported XOR circuit topologies, were simulated on a Cadence Spectre platform using the same testing conditions to measure the propagation delay and the power dissipation in each case. All the simulations were carried out with a 0.6V to 1.2V supply voltage range, a load capacitance of 10fF, and a throughput (clocking) rate of 200MHz, with rise and fall time of 500ps.

The design simulations consist of functional verification, power and timing analysis verified by Cadence Spectre, as well as design rule checking (DRC) and layout vs. schematic (LVS) of the layout, verified using Cadence Assura. The performance of all test circuits was evaluated in terms of worst-case propagation delay.

In order to achieve a closer analysis, several input patterns that covered all possible cases of input values were applied. Simulation results verified the correct functionality for every input combination up to the lower end of the supply voltage range. Table 3.2 summarises the results of these simulations in 130nm CMOS technology, providing a comparison of the propagation delay, the power dissipation and the PDP between both the proposed circuits and recently reported designs.

Table 3.2 *Simulation results of XOR gate in 130nm CMOS technology*

	V (v)	Proposed XOR1 (6T)[75]	Proposed XOR2 (6T)[76]	6T [65]	6T [72]	4T [66]	4T [73, 77]	3T [69, 74, 78]
Delay (ns)	0.6	1.8974	2.1107	2.3113	7.0638	2.5818	4.1910	15.48
	0.8	1.3120	1.5659	1.7672	3.6837	4.5128	3.9308	8.7314
	1	1.2413	1.4691	1.6475	1.6469	4.2216	8.9244	13.038
	1.2	1.2101	1.4200	1.6018	1.5500	4.1087	7.2075	12.901
Average power (nW)	0.6	0.1020	0.1351	0.1352	0.1385	0.2672	0.2672	0.2672
	0.8	0.1452	0.2312	0.2319	0.2401	0.4586	0.4586	0.4586
	1	0.1578	0.3674	0.3689	0.3842	0.7269	0.7269	0.7268
	1.2	0.2574	0.5523	0.5574	0.5835	1.0931	1.0931	1.0928
PDP (aJ)	0.6	0.1935	0.2852	0.3125	0.9783	0.6899	1.1198	4.1363
	0.8	0.1905	0.3620	0.4098	0.8845	2.0696	1.8027	4.0042
	1	0.1958	0.5397	0.6078	0.6327	3.0687	6.4871	9.3764
	1.2	0.3115	0.7843	0.8928	0.9044	4.4912	7.8785	14.0982

The proposed XOR1 gate offers less propagation delay followed by the proposed XOR2 ,and six transistor XOR gate in [65], as shown in Figure 3.7. The worst circuit in terms of speed is the three transistor XOR gate. It has the highest propagation delay against voltage scaling. The six transistor designs in [72] and [65] are close to the proposed XOR2 gate in terms of power dissipation and propagation delay, but the proposed XOR1 and XOR2 gates provide better overall improvement compared to these previous designs.

The XOR1 and XOR2 circuits demonstrate a reduction of power, and perform satisfactorily for low supply voltages, with a 0.1452nW and 0.2312nW power dissipation respectively, compared to the 0.2319nW dissipation by the design in [65] (@ VDD=0.8V). The XOR gates in [66], [73], [77]**,** [69], [74] and [78] have almost identical average power dissipation in all the supply voltages. Concentrating on the power consumption, Figure 3.8 demonstrates the effect of voltage scaling on the average power dissipation.

All other circuits have higher PDP than both proposed XOR circuits, as evidenced in Table 3.2 and illustrated in Figure 3.9. At 0.8V supply voltage, the proposed XOR1 and XOR2 circuits have at least 115.1% and 13.2% improvement in PDP respectively, over the design in [65] and 364.3% and 143.6% over the circuit in [72], respectively.

As output load is one of the parameters that effects the performance of the circuits, we have varied the output load from 10fF to 50fF at 0.8V supply voltage for all circuits, to study its effect on propagation delay. The proposed XOR1 gate is found to be the best circuit in terms of propagation delay for all values of output loads as shown in Figure 3.10, and as the fastest circuit in the high output load, followed by the circuit in [65]. Thus, from the simulation results, it is clear that the proposed XOR1 gate has the lowest propagation delay, as well as the lowest power consumption, along with a high output driving capability in low power supply operations. The improvements attained by the proposed circuit are thus clearly evident when compared to these other XOR gate circuits.

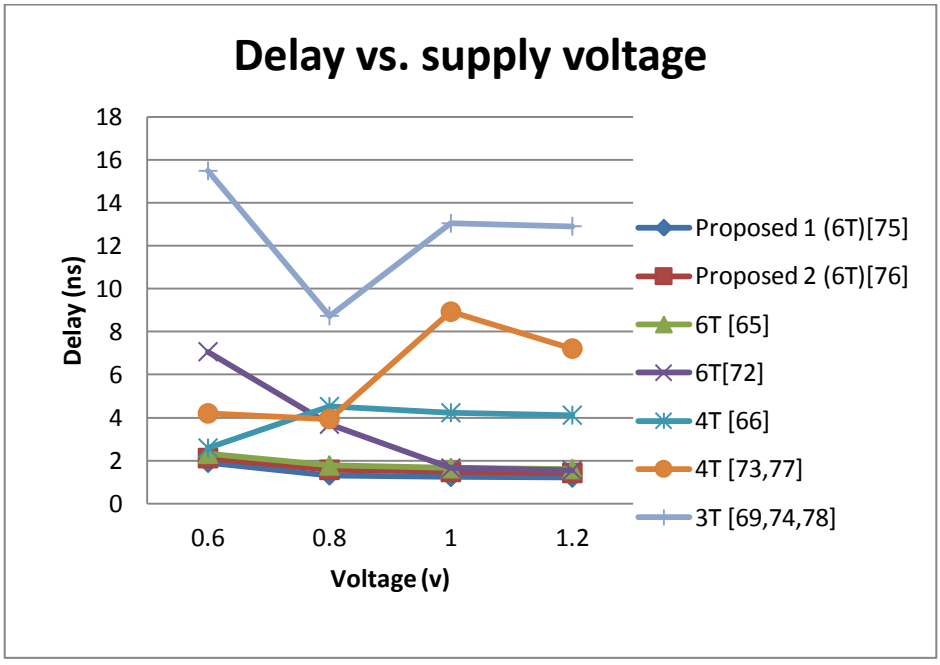


Figure 3.7 *Propagation delay with supply voltage scaling for different XOR gates in 130nm CMOS technology*

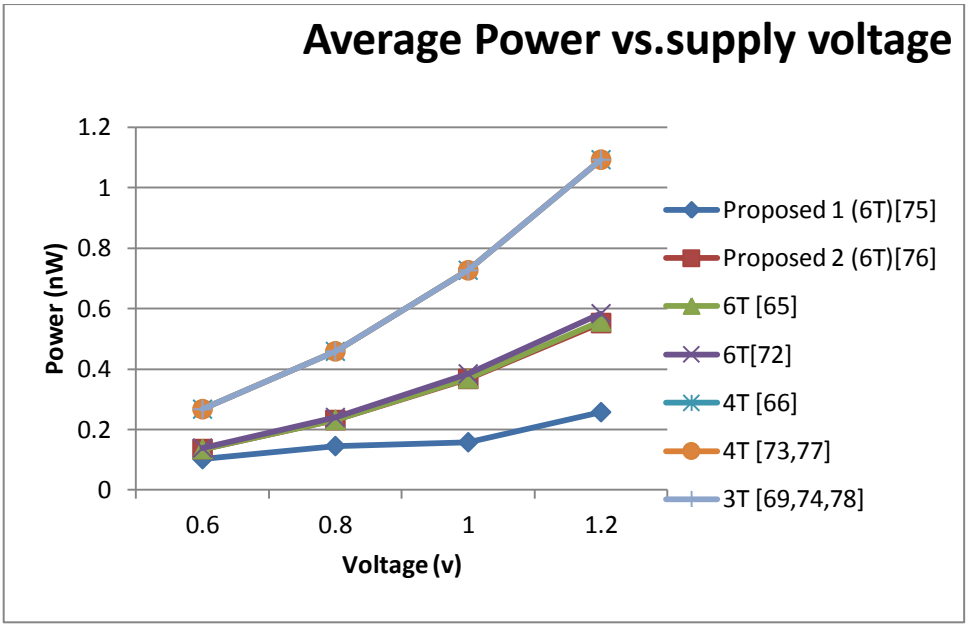


Figure 3.8 *Power dissipation with supply voltage scaling for different XOR gates in 130nm CMOS technology.*

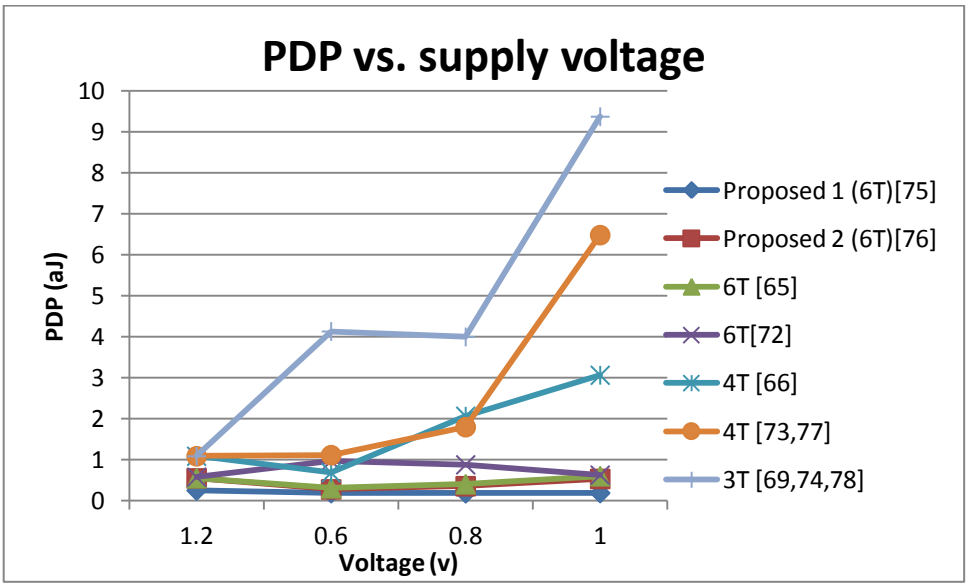
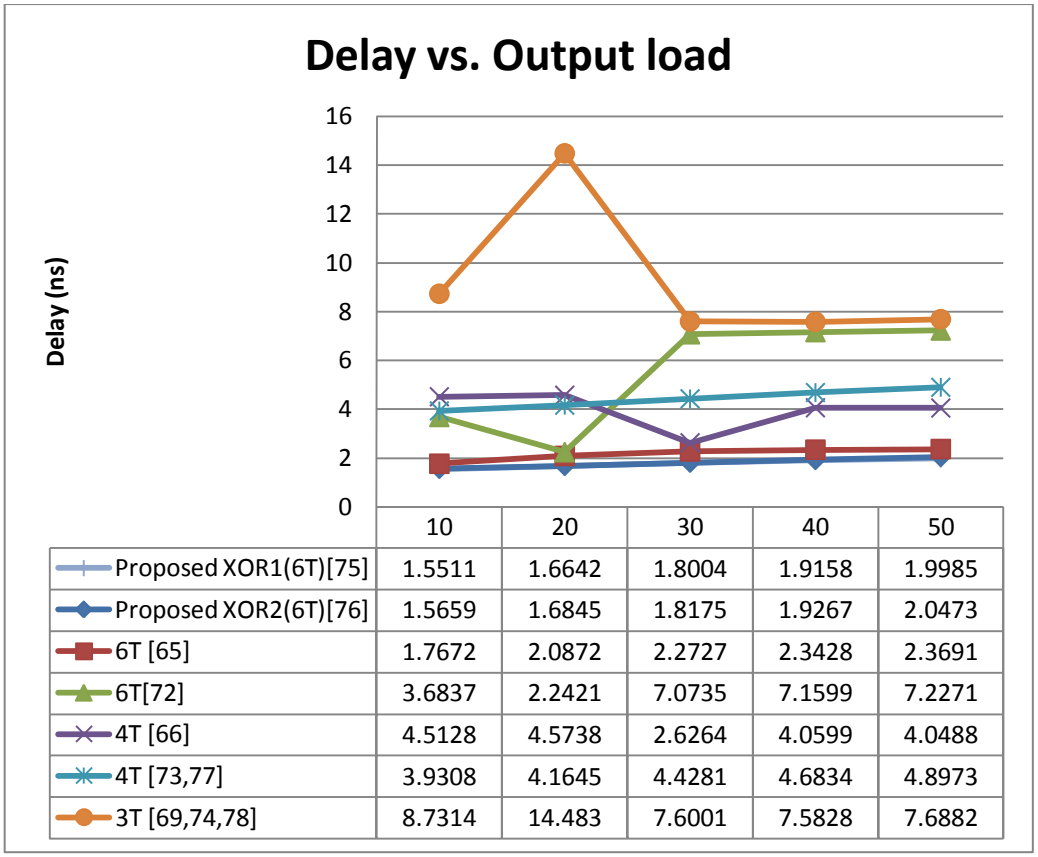


Figure 3.9 *Power-delay product (PDP) with supply voltage scaling for different XOR gates in 130nm CMOS technology.*



	10	20	30	40	50
Proposed XOR1(6T)[75]	1.5511	1.6642	1.8004	1.9158	1.9985
Proposed XOR2(6T)[76]	1.5659	1.6845	1.8175	1.9267	2.0473
6T [65]	1.7672	2.0872	2.2727	2.3428	2.3691
6T[72]	3.6837	2.2421	7.0735	7.1599	7.2271
4T [66]	4.5128	4.5738	2.6264	4.0599	4.0488
4T [73,77]	3.9308	4.1645	4.4281	4.6834	4.8973
3T [69,74,78]	8.7314	14.483	7.6001	7.5828	7.6882

Figure 3.10 *Propagation Delay vs. output load*

In order to verify the noise immunity, the noise margins (NM_H and NM_L) for both of the proposed XOR gates, and the other gates, was determined based on DC input-output voltage transfer analysis. The noise margins for a 0.8V supply voltage are shown in Table 3.3. Both proposed XOR gates indicate acceptable values of noise margin compared with the other XOR gates.

The noise margins are measured based on DC analyses to determine the allowable noise voltage on the input of a gate. The noise margin is the ability to tolerate noise without effecting the correct operation of the circuit [78]. The higher the noise margin value, the greater the differences between valid high or valid low. The low noise margin, N_{ML} and high noise margin, N_{MH} are expressed in the following equations (3.1) and (3.2) respectively.

$$N_{ML} = V_{IL} - V_{OL} \quad (3.1)$$

$$N_{MH} = V_{OH} - V_{IH} \quad (3.2)$$

where,

V_{IH} = minimum HIGH input voltage

V_{IL} = maximum LOW input voltage

V_{OH} = minimum HIGH output voltage

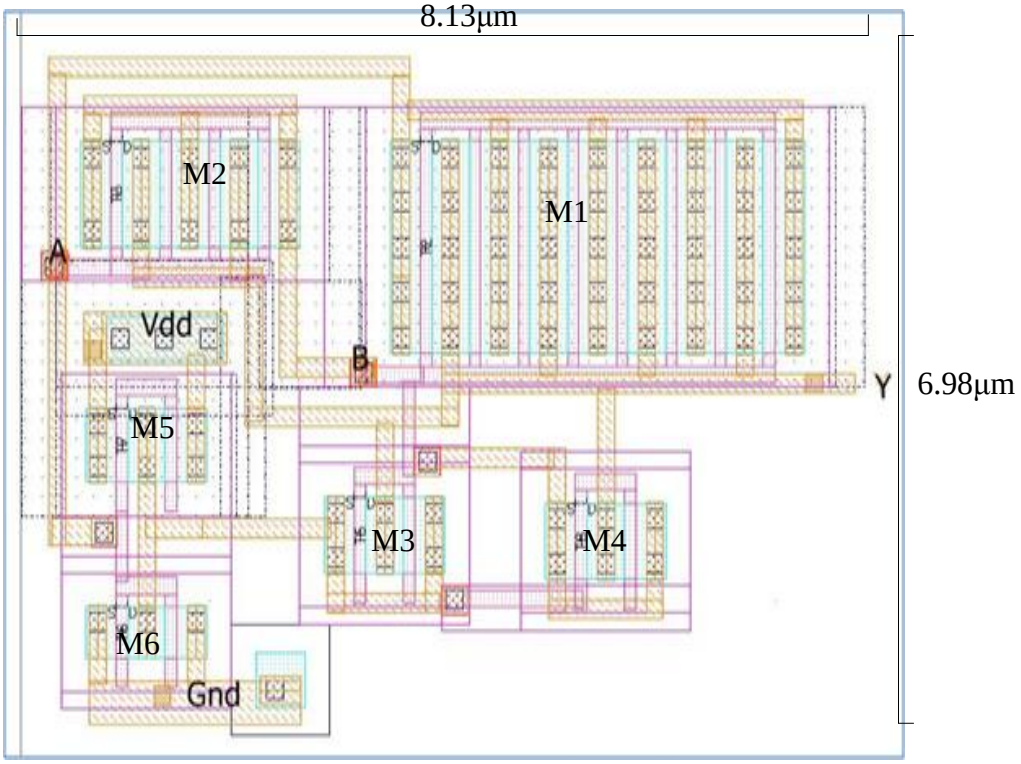
V_{OL} = maximum LOW output voltage

Table 3.3 *Noise Margin of different XOR gate*

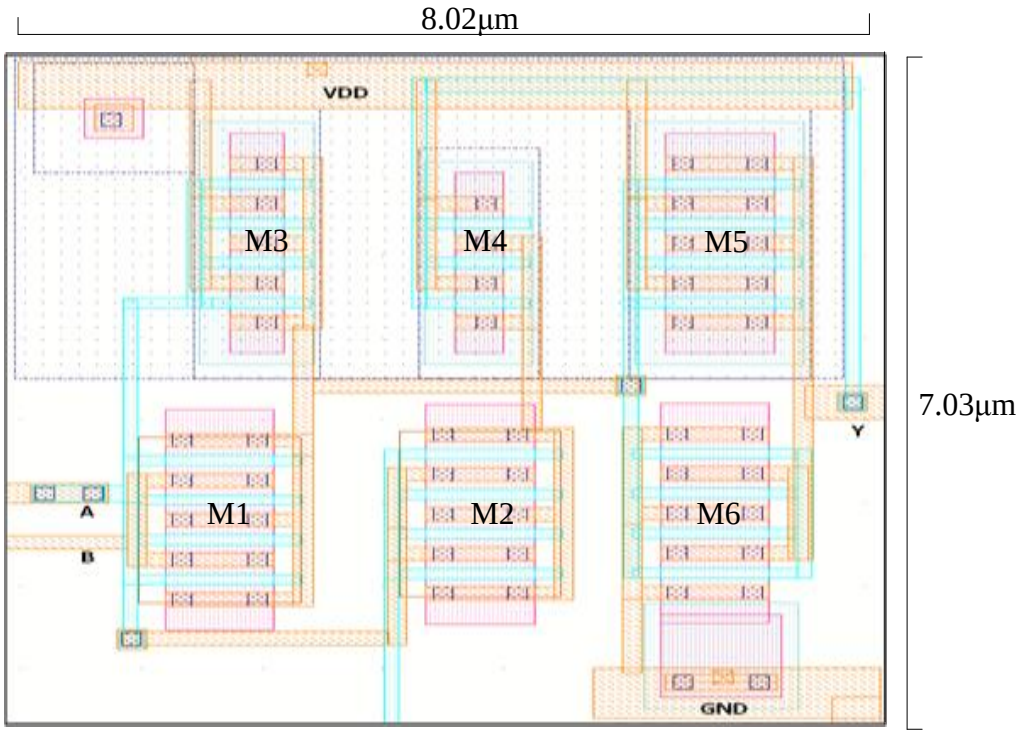
Type of XOR	V(v)	Voh(V)	Vih(V)	Vil(V)	Vol(V)	Nmh(V)	Nml(V)
Proposed 1 (6T)[75]	0.8	0.7101	0.3875	0.2844	0.0778	0.3226	0.2066
Proposed 2 (6T)[76]	0.8	0.7186	0.3889	0.2867	0.0796	0.3297	0.2071
6T [65]	0.8	0.7204	0.3967	0.3022	0.0687	0.3237	0.2335
6T[72]	0.8	0.7205	0.3967	0.3022	0.0687	0.3238	0.2335
4T [66]	0.8	0.9460	0.6000	0.4489	0.1172	0.346	0.3317
4T [73, 77]	0.8	1.0809	0.6444	0.4233	0.1423	0.4365	0.2810
3T [69, 74, 78]	0.8	1.0797	0.5978	0.4100	0.1332	0.4819	0.2768

3.5 XOR gate chip and experimental results

The complete circuit simulation, optimisation, layout and parasitic extraction were carried out using Cadence tools. The mask layout of the proposed XOR gates illustrated in Figure 3.11 were customised (with manual placement and routing) using Virtuoso layout, which is parasitic extracted, back annotated, and verified using Cadence Assura. Minimum channel length is used for all devices and optimum channel width is carefully chosen for each device to achieve verified functionality with low power dissipation and the smallest possible propagation delay.



(a)



(b)

Figure 3.11 *Layout of the 2-input XOR gate (a) XOR1 gate (b) XOR2 gate*

Due to the timeline, only the proposed XOR2 gate design was fabricated in order to verify the functionality. The proposed XOR2 gate was fabricated using a 130nm IBM CMOS process. The photomicrograph of the fabricated dies of the XOR gate chip, along with bonding pads, is shown in Figure 3.12 The silicon-area of the XOR gate is 8.02µm x 7.03µm (≈56 square µm), excluding the bonding pads. Transient measurement is carried out using the Tektronix TLA5202 Logic Analyzer, Tektronix CFG253 Function Generator and Hewlett Packard E3630A Triple Output DC Power Supply.

A level shifter is used to generate low voltage (0.8V) from the high voltage (5V) of a pattern generator, to the XOR gate input. Figure 3.13 displays the functional verification of the XOR gate with a snapshot of XOR input and output waveforms for several input combinations using the Tektronix TLA5202 Logic Analyzer. Figure 3.14 provides the Agilent DCA-J (86100C Infiniium) oscilloscope electrical waveforms for the fabricated XOR gate (Yellow = input A, Blue = input B and Green= output Y), indicating correct operation for a supply voltage of 0.8V. Figure 3.15 and Figure 3.16 indicate the rise time of 319.4ps and fall time of 290.18ps of the output signal.

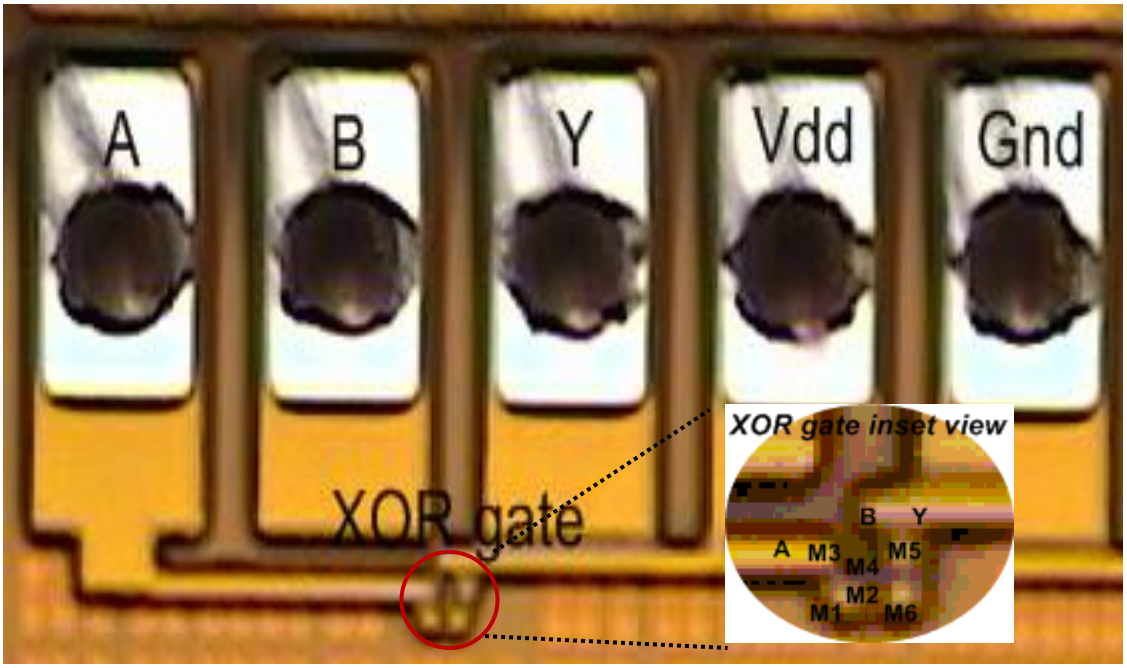


Figure 3.12 *Microphotograph of the fabricated novel CMOS pass-transistor based full swing output voltage XOR2 gate along with bonding pads and XOR gate inset view.*

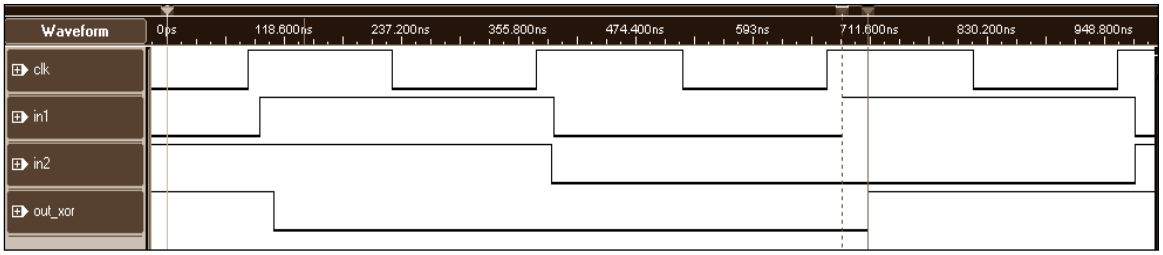


Figure 3.13 *Logic Analyzer waveform of the inputs and the output for the fabricated XOR2 gate*

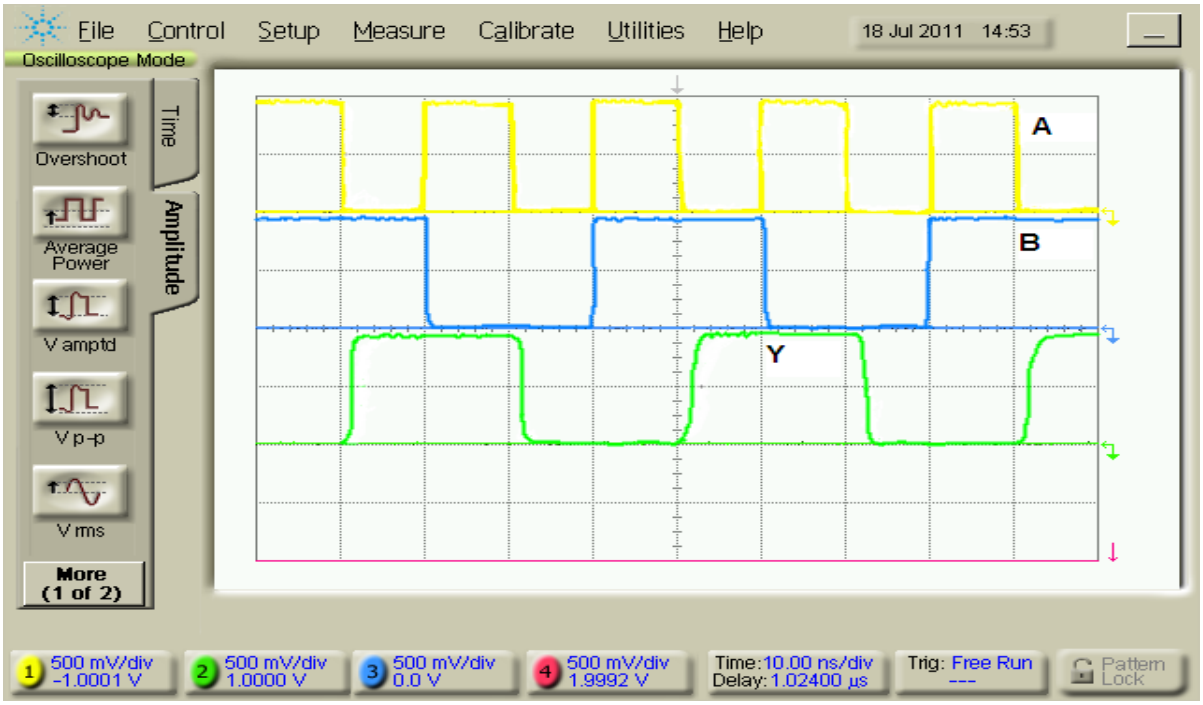


Figure 3.14 *Oscilloscope waveforms for the fabricated XOR2 gate (Yellow = input A, Blue= input B and Green=output Y)*

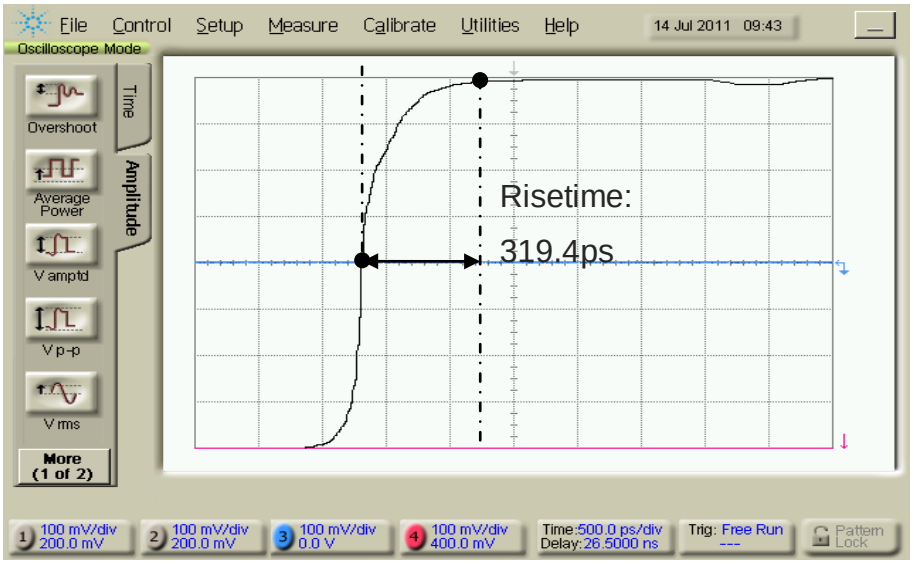


Figure 3.15 *Rise time of output signal*

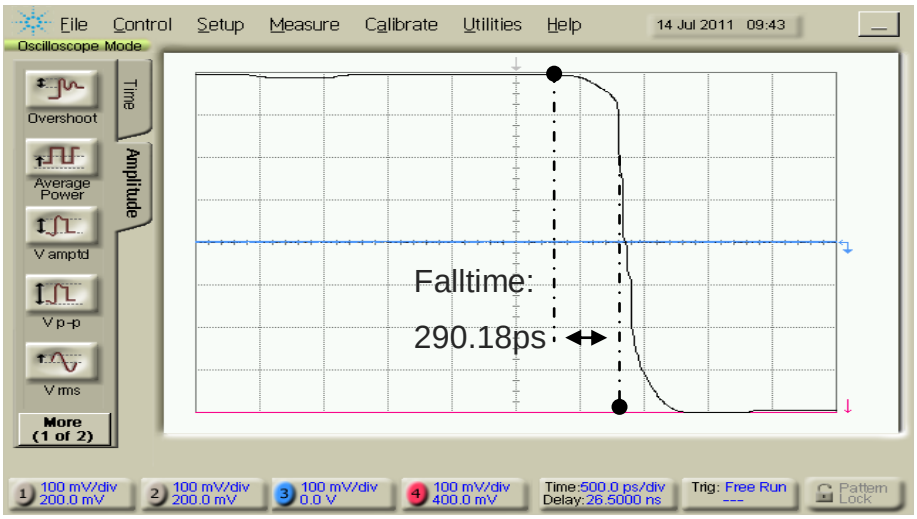


Figure 3.16 *Fall time of output signal*

3.6 Conclusions

This chapter reports two new topologies of 2-input XOR gate featuring low-power operation and small chip area, provide a full swing output voltage for all input combinations, implemented in a 65nm and 130nm CMOS process suitable to reduce the power and delay of an overall system-on-chip. Both of the designs adopt pass transistor

for low transistor count and inverter, to enhance the driving capability in a complex design. By enhancing the driving capability, it facilitates lower voltage operation with lower power consumption and small propagation delay. Both of the proposed XOR gates were compared to other peer XOR designs with extensive simulations and the results indicate satisfactory performance and improvements in terms of power consumption, propagation delay and PDP. Based on the results, the proposed XOR1 gate has the lowest gate propagation delay as well as the lowest power consumption except when V=1.0, and a lower PDP than its peer designs, followed by the proposed XOR2 gate design.

Taking into consideration power consumption and chip area, the proposed XOR gates presents an interesting option, as it offers low power, low delay and small gate count. Hence, the new proposed XOR1 gate is optimally suitable for implementing CMOS Galois field arithmetic for the S-Box/ InvS-Box and full custom AES crypto-processor to achieve an overall low-power and low-cost system.

CHAPTER 4

A COMPACT COMBINATIONAL LOGIC DESIGN OF S-BOX/ INVS-BOX FOR AES CRYPTO-PROCESSOR

4.1 Introduction

In this chapter, we present a low-power low-complexity design methodology for the S-box/ InvS-box which includes minimising the comprehensive circuit size and critical path delay, scaling down the supply voltage and the transistor size, along with choosing an advanced technology for an optimised CMOS full custom design. The advantage of full custom designs utilizing state of the art CMOS processes that it is possible to scale all the transistors down with process scaling by selecting the optimum transistor size to balance within the power and speed for the target specification, without deteriorating the overall performance, and with increased speed in most cases. This leads to a smaller chip area and low power consumption. Another design methodology is to reduce power consumption by using an advanced process technology that offers very low supply voltage. This approach also leads to a reduction in the die area.

The S-box is at the core of any AES implementation and is considered a full complexity design, consuming the major portion of the power and energy budget of the AES hardware. This chapter explores the implementation of an S-box in a SubBytes transformation that operates on individual bytes using a substitution table (S-box) containing a permutation of all 256 possible 8-bit values. Two transformations are required, firstly byte replacement, with its multiplicative inverse in the finite field GF (2^8), using the irreducible polynomial $p(x) = x^8+x^4+x^3+x+1$ and secondly, an affine transformation over GF (2^8). For the decryption operation, an inverse S-box is obtained by applying an inverse affine transformation, followed by a multiplicative inversion in GF (2^8).

As mentioned in the chapter introduction, this chapter focuses on the composite field technique for a full-custom CMOS implementation of the AES S-box using low-power XOR gates. Composite field is a sub field of the finite field GF (2^8) and a brief explanation on composite field have been explained in Chapter 2. Computations in the field GF (2^8) are simplified by the computations in the composite field GF $(2^4)^2$ or GF $(2^2)^2)^2$ by using equation (2.10) in chapter 2 to reduce the hardware complexities. Several existing different construction schemes using composite field are found in [4], [20] and [21]. Daemen and Rijmen [3] propose the first efficient hardware implementation of the multiplication inversion in GF(2^8), by decomposing the finite field GF(2^8) to its sub-field GF(2^4), which leads to low complexity in hardware. Furthermore, Rudra et al. [21] and Wolkerstofer, Oswald and Lamberger [26] decompose the elements of GF(2^8) into GF$(2^4)^2$ to implement the multiplicative inverse in SubBytes and [26] implement it in ASIC implementation. The transformation matrix from GF(2^8) to GF$(2^2)^2)^2$ is proposed by Satoh et al. [4] and is claimed to be the most minimised hardware implementation to date, with a gate complexity of 5400 gates. Mentens et al. [27] also use the same approach as Satoh et al., [4] but with different polynomial coefficients, and achieve slightly better optimised hardware than that by Marioko and Satoh [12]. Other efforts towards the efficient implementation of the S-box include those by Canright [22], Burns et al. [23] and Liu and Parhi [10], which further improve the performance in area, power and delay. This chapter is focused on an area-efficient low-voltage and low-power CMOS implementation of the S-Box/ Inv S-Box.

4.2 S-box/ InvS-box design methodology

This chapter firstly explains that S-box/ InvS-box architecture employs combinational logic using composite field arithmetic based on Satoh et al.'s work [4] and optimized by Mentens et al. [27] with a different choice of polynomial coefficients, and the implementation of constant multiplication with λ. Next, we discussed the improvement and reduction of the hardware complexity of the S-box/ InvS-box. This architecture is implemented using XOR circuits, multiplexers, and basic logic gates.

The optimization of the low-voltage and low-power composite field S-box implementation has been further enhanced by using a new six-transistor XOR gate. In comparison to [4] and [27], the inverse S-box for decryption is also implemented by the same chip. In addition, it employs a low-power design methodology, such as minimizing the circuit size by efficient logic implementations, as well as reducing supply voltage using an advanced CMOS technology. The composite field inversion by extending $GF(2^8)$ over $GF(((2^2)^2)^2)$ can be used to create compact AES implementations [4] and [21]. This approach was chosen to achieve small area design. First the column vectors of the *State* matrix go into the isomorphic transformation from $GF(2^8)$ into the composite field $GF(((2^2)^2)^2)$, followed by inversion in the composite field and inverse isomorphic transformation. Then an affine transformation is carried out to create the cipher data. This can be represented by the elementary transformations $\varsigma \xrightarrow{ISO} \Gamma \xrightarrow{MINV} \Lambda \xrightarrow{INVISO} \Gamma' \xrightarrow{AFFINE} H$ where, ς are byte elements from the *State* matrix, Γ are byte elements of the isomorphic mapping transformation, Λ are multiplicative inverse elements of the isomorphic state, Γ' are the elements after inverse isomorphic mapping, and finally H are elements after affine transformation. The InvSubBytes involves an isomorphic transformation followed by an inverse affine transformation.

The inversion in composite field is carried out followed by inverse isomorphic mapping. This can be represented by the elementary transformations $\varsigma'' \xrightarrow{ISO} \Gamma'' \xrightarrow{INVAFFINE} H' \xrightarrow{MINV} \Lambda \xrightarrow{INVISO} \Gamma'$. The affine transformation, AT operates on the $GF(2^8)$ multiplicative inverse of bytes, ς'' of the state matrix, while the inverse affine transformation AT^{-1} operates on the isomorphic affine transformed GF (2^8) multiplicative inverse of the same bytes, b represented by Γ''. The resultant matrix operations of AT and AT^{-1} using equation (2.4) - (2.9) can be translated to the logical implementations given by:

$$[AT(\Lambda)] = \begin{bmatrix} H_0 \\ H_1 \\ H_2 \\ H_3 \\ H_4 \\ H_5 \\ H_6 \\ H_7 \end{bmatrix} = \begin{bmatrix} \overline{\Lambda_0 \oplus \Lambda_4 \oplus \Lambda_5 \oplus \Lambda_6 \oplus \Lambda_7} \\ \overline{\Lambda_0 \oplus \Lambda_1 \oplus \Lambda_5 \oplus \Lambda_6 \oplus \Lambda_7} \\ \Lambda_0 \oplus \Lambda_1 \oplus \Lambda_2 \oplus \Lambda_6 \oplus \Lambda_7 \\ \Lambda_0 \oplus \Lambda_1 \oplus \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_7 \\ \Lambda_0 \oplus \Lambda_1 \oplus \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_4 \\ \overline{\Lambda_1 \oplus \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_4 \oplus \Lambda_5} \\ \overline{\Lambda_2 \oplus \Lambda_3 \oplus \Lambda_4 \oplus \Lambda_5 \oplus \Lambda_6} \\ \Lambda_3 \oplus \Lambda_4 \oplus \Lambda_5 \oplus \Lambda_6 \oplus \Lambda_7 \end{bmatrix} \tag{4.1}$$

$$[AT^{-1}(\Gamma)] = \begin{bmatrix} H'_0 \\ H'_1 \\ H'_2 \\ H'_3 \\ H'_4 \\ H'_5 \\ H'_6 \\ H'_7 \end{bmatrix} = \begin{bmatrix} \overline{\Gamma''_2 \oplus \Gamma''_5 \oplus \Gamma''_7} \\ \Gamma''_0 \oplus \Gamma''_3 \oplus \Gamma''_6 \\ \overline{\Gamma''_1 \oplus \Gamma''_4 \oplus \Gamma''_7} \\ \Gamma''_0 \oplus \Gamma''_2 \oplus \Gamma''_5 \\ \Gamma''_1 \oplus \Gamma''_3 \oplus \Gamma''_6 \\ \Gamma''_2 \oplus \Gamma''_4 \oplus \Gamma''_7 \\ \Gamma''_0 \oplus \Gamma''_3 \oplus \Gamma''_5 \\ \Gamma_1 \oplus \Gamma_4 \oplus \Gamma_6 \end{bmatrix} \tag{4.2}$$

The S-box is optimised by performing the inversion operation in a composite Galois field of GF($(2^4)^2$) or GF($((2^2)^2)^2$), obtained from GF(2^8) via isomorphic mapping. The isomorphic mapping, *ISO* and its inverse, ISO^{-1} [12] and [27] as well as their logical implementations are given by:

$$[ISO(\varsigma)] = \begin{bmatrix} \Gamma_0 \\ \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \\ \Gamma_4 \\ \Gamma_5 \\ \Gamma_6 \\ \Gamma_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \varsigma_0 \\ \varsigma_1 \\ \varsigma_2 \\ \varsigma_3 \\ \varsigma_4 \\ \varsigma_5 \\ \varsigma_6 \\ \varsigma_7 \end{bmatrix}$$

$$= \begin{bmatrix} \varsigma_0 \oplus \varsigma_2 \\ \varsigma_1 \oplus \varsigma_6 \oplus \varsigma_7 \\ \varsigma_2 \oplus \varsigma_5 \\ \varsigma_1 \oplus \varsigma_3 \oplus \varsigma_6 \oplus \varsigma_7 \\ \varsigma_1 \oplus \varsigma_5 \oplus \varsigma_7 \\ \varsigma_1 \oplus \varsigma_4 \oplus \varsigma_5 \oplus \varsigma_6 \\ \varsigma_1 \oplus \varsigma_2 \oplus \varsigma_3 \oplus \varsigma_4 \oplus \varsigma_5 \oplus \varsigma_6 \\ \varsigma_5 \oplus \varsigma_7 \end{bmatrix} \tag{4.3}$$

$$[ISO^{-1}(\Lambda)] = \begin{bmatrix} \Gamma'_0 \\ \Gamma'_1 \\ \Gamma'_2 \\ \Gamma'_3 \\ \Gamma'_4 \\ \Gamma'_5 \\ \Gamma'_6 \\ \Gamma'_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \Lambda_0 \\ \Lambda_1 \\ \Lambda_2 \\ \Lambda_3 \\ \Lambda_4 \\ \Lambda_5 \\ \Lambda_6 \\ \Lambda_7 \end{bmatrix}$$

$$= \begin{bmatrix} \Lambda_0 \oplus \Lambda_1 \oplus \Lambda_3 \oplus \Lambda_5 \oplus \Lambda_6 \\ \Lambda_4 \oplus \Lambda_7 \\ \Lambda_1 \oplus \Lambda_3 \oplus \Lambda_5 \oplus \Lambda_6 \\ \Lambda_1 \oplus \Lambda_3 \\ \Lambda_1 \oplus \Lambda_5 \oplus \Lambda_7 \\ \Lambda_1 \oplus \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_5 \oplus \Lambda_6 \\ \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_4 \oplus \Lambda_5 \oplus \Lambda_6 \\ \Lambda_1 \oplus \Lambda_2 \oplus \Lambda_3 \oplus \Lambda_5 \oplus \Lambda_6 \oplus \Lambda_7 \end{bmatrix} \tag{4.4}$$

Elements in GF (2^8) after isomorphic transformation can be represented as $gx + e$ in the composite field GF (((2^2)^2)^2), where g is the most significant nibble while e is the least significant nibble and g, $e \in$ GF((2^2)^2). This way GF (((2^2)^2)^2) is generated as a field extension of degree 2 over the field GF ((2^2)^2) using the irreducible polynomial *f(x)*

$=x^2+Ax+B$ where, $A, B \in$ GF $((2^2)^2)$. The multiplicative inverse can be computed using the equation below [25]:

$$(gx+e)^{-1} = g\,(g^2B + geA + e^2)^{-1}x + (e+gA)\,(g^2B + geA + e^2)^{-1} \quad (4.5)$$

Choosing $A = 1$ and $B = \lambda$, the equation can be simplified to

$$(gx+e)^{-1} = g\,(g^2\,\lambda + e\,(g + e))^{-1}x + (e+g)(g^2\,\lambda + e(g + e))^{-1} \quad (4.6)$$

In equation (4.6), the multiplication, addition, squaring and inversions are in GF (2^4). This is shown in Figure 4.1. It is to be noted here that addition of words in Galois field is a simple bitwise XOR without carry, unlike binary carry-propagation arithmetic. Using several layers of hierarchical composite field decompositions, the multiplicative inverse computation is simplified resulting in simple bitwise XOR and AND operations. The hierarchical composite field decomposition using extensions over lower order fields and the corresponding choice of irreducible polynomials and coefficients are as follows:

$GF\,(((2^2)^2)^2) \rightarrow \quad GF((2^2)^2)$: $x^2 + A\,x + B$,
where,
A, $B \in GF\,(2^2)^2)$ and A=1, $B=\lambda$ (4.7)

$GF\,((2^2)^2) \rightarrow \quad GF(2^2)$: $x^2 + Z_1x + Z_0$,
where
$Z_1, Z0 \in GF(2^2)$ and $Z_1=1$, $Z_0 = \varphi$ (4.8)

$GF(2^2) \rightarrow \quad GF(2)$: $x^2 + x + 1$ (4.9)

The constants are chosen based on Mentens et al. [27] for the optimal hardware solution, so that $\varphi = \{10\}_2$ and $\lambda = \{1000\}_2$, which leads to a total number of '*1*' entries of the transformation matrices equal to 54, reduced by five from Satoh's [4] constant multiplication, but this requires one extra XOR gate. It also has the lowest gate count and the shortest critical path.

The following describes how the various arithmetic operations are carried out by composite field decomposition into lower order fields.

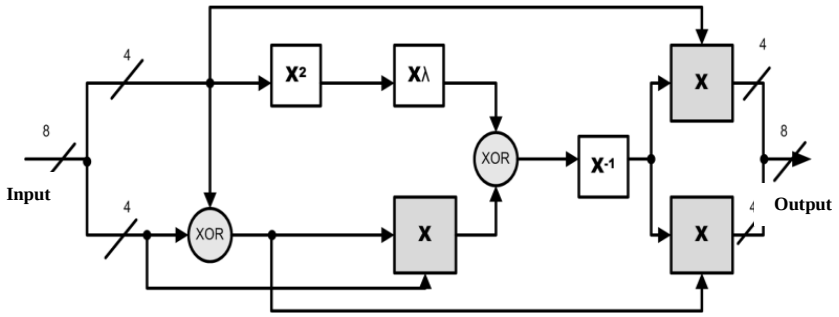


Figure 4.1 *Multiplicative Inverse in GF (2^8) as extension of degree 2 over GF ((2^2)^2)*

4.2.1 Multiplication of nibble, λ

For multiplication of nibble, q= $\{q_3\ q_2\ q_1\ q_0\}_2$ with constant, λ= $\{1000\}_2$ in GF(2^4), let k= $\{k_3\ k_2\ k_1\ k_0\}_2$ be the product nibble in GF(2^4). Then, q, λ and k can be expressed as extensions over the GF(2^2), so that, k=$\{k_3\ k_2\}_2x$+$\{k_1\ k_0\}_2$=k_Hx+k_L where, k_H=$\{k_3\ k_2\}_2$ and k_L=$\{k_1\ k_0\}_2$. Furthermore, q= $\{q_3\ q_2\}_2x$+ $\{q_1\ q_0\}_2$ =q_Hx+q_L where, q_H= $\{q_3\ q_2\}_2$ and q_L= $\{q_1\ q_0\}_2$. In addition, λ= $\{1\ 0\}_2x$+$\{0\ 0\}_2$ =λ_Hx+λ_L where, λ_H=$\{10\}_2$ and λ_L=$\{0\ 0\}_2$. Here, bit-pairs k_H, k_L, q_H, q_L, λ_H and λ_L are ϵ GF (2^2). Then the product, k= (q_Hx+q_L) (λ_Hx+λ_L) =$q_H\lambda_Hx^2$+$q_L\lambda_Hx$. Next, by substituting x^2=x+φ, the new equation for k is given by k = $q_H\lambda_H(x+\varphi)$ +$q_L\lambda_Hx$ = ($q_H\lambda_H$+$q_L\lambda_H$) x+$q_H\lambda_H\varphi$ = k_Hx+k_L where, k_H= ($q_H\lambda_H$ +q_L λ_H), k_L=$q_H\lambda_H\varphi$ are ϵ GF(2^2). Now, expressing the bit-pairs k_H, k_L, q_H, q_L, λ_H, λ_L and φ over GF(2), we have: k_H = (q_3x+q_2) ($1x$+0) + (q_1x+q_0) ($1x$+0) = q_3x^2+q_2x+q_1x^2+q_0x. Next, making the substitution x^2=x+1, k_H=$q_3(x+1)$ +q_2x+$q_1(x+1)$ +q_0x= (q_3+q_2+q_1+q_0) x+ (q_3+q_1) =k_3x+k_2. Similarly, we have: k_L= (q_3x+q_2) ($1x$+0) ($1x$+0) = (q_3x+q_2) x^2=q_3x^3+q_2x^2. Now, x^3=x^2.x=(x+1) x=x^2+x=x+1+x=1 in GF(2), so that, k_L=q_3+$q_2(x+1)$ =q_2x+ (q_3+q_2) =k_1x+k_0. Hence, finally, product nibble k is as follows:

$$\left.\begin{aligned} k_3 &= (q_3 + q_2 + q_1 + q_0) \\ k_2 &= (q_3 + q_1) \\ k_1 &= q_2 \\ k_0 &= (q_3 + q_2) \end{aligned}\right\} \qquad (4.10)$$

4.2.2 Squaring nibbles

For squaring the nibble $q = \{q_3\ q_2\ q_1\ q_0\}_2 \in GF(2^4)$, let k= $\{k_3\ k_2\ k_1\ k_0\}_2$ be the squared nibble in GF (2^4). Then, q and k can be expressed as extensions over the GF(2^2), so that, $k = \{k_3\ k_2\}_2 x + \{k_1\ k_0\}_2 = k_H x + k_L$ where $k_H = \{k_3\ k_2\}_2$, and $k_L = \{k_1\ k_0\}_2$. Also, $q = \{q_3\ q_2\}_2 x + \{q_1\ q_0\}_2 = q_H x + q_L$ where $q_H = \{q_3\ q_2\}_2$, and $q_L = \{q_1\ q_0\}_2$. Here, $k_H, k_L, q_{H,\ and}\ q_L$ are $\in$ GF(2^2). Then $k = (q_H x + q_L)^2 = q_H^2 x^2 + q_L^2$. Next, by substituting $x^2 = x + \varphi$, the new equation for k is given by $k = q_H^2(x + \varphi) + q_L^2 = q_H^2 x + q_H^2 \varphi + q_L^2 = k_H x + k_L$ where $k_H = q_H^2$, $k_L = (q_H^2 \varphi + q_L^2)$ are $\in$ GF (2^2).

Now, expressing k_H, k_L, q_H, q_L and φ over GF (2), we have $k_H = (q_3 x + q_2)^2 = q_3 x^2 + q_2$. Next, making the substitution $x^2 = x + 1$, $k_H = q_3(x+1) + q_2 = q_3 x + (q_2 + q_3) = k_3 x + k_2$. Similarly, we have, $k_L = (q_3 x + q_2)^2 (1x + 0) + (q_1 x + q_0)^2 = (q_3^2 x^2 + q_2^2)\ x + q_1^2 x^2 + q_0^2 = q_3 x^3 + q_2 x + q_1 x^2 + q_0$. By substituting the x^3 term into the equation with 1, and x2 by $x + 1$, $k_L = q_3 + q_2 x + q_1(x+1) + q_0 = q_3 + q_2 x + q_1 x + q_1 + q_0 = (q_2 + q_1)x + (q_3 + q_1 + q_0) = k_1 x + k_0$. Consequently, the squared nibble k is as follows:

$$\left.\begin{aligned} k_3 &= q_3 \\ k_2 &= (q_2 + q_3) \\ k_1 &= (q_1 + q_2) \\ k_0 &= (q_0 + q_1 + q_3) \end{aligned}\right\} \qquad (4.11)$$

4.2.3 Multiplication of nibbles in GF (2^4)

For multiplication of nibble $q = \{q_3\ q_2\ q_1\ q_0\}_2$, with the nibble $w = \{w_3\ w_2\ w_1\ w_0\}_2$ in GF(2^4), let k= $\{k_3 k_2\ k_1\ k_0\}_2$ be the product nibble in GF(2^4). Then, q, w and k can be expressed as extensions over the GF (2^2), so that, $k = \{k_3\ k_2\}_2 x + \{k_1\ k_0\}_2 = k_H x + k_L$, where, $k_H = \{k_3\ k_2\}_2$ and $k_L = \{k_1\ k_0\}_2$.

Additionally, $q = \{q_3\ q_2\}_2 x + \{q_1\ q_0\}_2 = q_H x + q_L$, where $q_H = \{q_3\ q_2\}_2$, and $q_L = \{q_1\ q_0\}_2$. In addition, $w = \{w_3\ w_2\}_2 x + \{w_1\ w_0\}_2 = w_H x + w_L$, where $w_H = \{w_3\ w_2\}_2$, and $w_L = \{w_1\ w_0\}_2$. Here, k_H, k_L, q_H, q_L, w_H and w_L are ϵ GF(2^2). Then $k = (q_H x + q_L)(w_H x + w_L) = (q_H w_H x^2 + q_H w_L x + q_L w_H x + q_L w_L) = q_H w_H x^2 + (q_H w_L + q_L w_H)x + q_L w_L$.

By substituting x^2 with $x+\varphi$, we have, $k = q_H w_H (x+\varphi) + (q_H w_L + q_L w_H)x + q_L w_L = (q_H w_H + q_H w_L + q_L w_H)x + q_H w_H \varphi + q_L w_L = (q_H w_H + q_H w_L + q_L w_H + q_L w_L + q_L w_L)x + q_H w_H \varphi + q_L w_L = \{(q_H + q_L)(w_H + w_L) + q_L w_L\}x + (q_H w_H \varphi + q_L w_L) = k_H x + k_L$, where, $k_H = \{(q_H + q_L)(w_H + w_L) + q_L w_L\}$ and $k_L = (q_H w_H \varphi + q_L w_L)$.

The product in GF(2^4) is then expressed as:

$$\left.\begin{aligned} k_H &= \{(q_H + q_L)(w_H + w_L) + q_L w_L\} \\ k_L &= (q_H w_H \varphi + q_L w_L) \end{aligned}\right\} \qquad (4.12)$$

4.2.4 Multiplication of bit-pairs in GF (2^2)

Let $k = qw$, where bit-pairs $k = \{k_1\ k_0\}_2$, $q = \{q_1\ q_0\}_2$ and $w = \{w_1\ w_0\}_2$ are elements in GF(2^2). Now expressing k, q and w over GF (2), we have $k = k_1 x + k_0 = (q_1 x + q_0)(w_1 x + w_0) = q_1 w_1 x^2 + q_0 w_1 x + q_1 w_0 x + q_0 w_0$. Using the irreducible polynomial substitution, $x^2 = x + 1$, $k = q_1 w_1 (x+1) + q_0 w_1 x + q_1 w_0 x + q_0 w_0 = (q_1 w_1 + q_0 w_1 + q_1 w_0)x + (q_1 w_1 + q_0 w_0) = k_1 x + k_0$. Consequently, the product bits are as follows:

$$\left.\begin{aligned} k_1 &= (q_1 w_1 + q_0 w_1 + q_1 w_0) \\ k_0 &= (q_1 w_1 + q_0 w_0) \end{aligned}\right\} \qquad (4.13)$$

4.2.5 Multiplication of bit-pairs with constant φ in GF(2^2)

Let $k = q\varphi$, where the bit-pairs $k = \{k_1\ k_0\}_2$, $q = \{q_1\ q_0\}_2$ and $\varphi = \{10\}_2$ are elements in GF(2^2). Now expressing k, q and φ over GF (2), we have, $k = k_1 x + k_0 = (q_1 x + q_0)(x + 0) = q_1 x^2 + q_0 x$. Next, making the substitution, $x^2 = x + 1$, $k = q_1(x + 1) + q_0 x = (q_1 + q_0)x + q_1 = k_1 x + k_0$. Consequently:

$$\left.\begin{aligned} k_1 &= (q_1 + q_0) \\ k_0 &= q_1 \end{aligned}\right\} \tag{4.14}$$

4.2.6 Multiplicative inverse of nibble in GF(2^4)

Based on Xinmiao and Parhi [33], the multiplicative inverse in GF(2^4) of nibble input $\gamma = \{\gamma_3\ \gamma_2\ \gamma_1\ \gamma_0\}$ is given by output $\theta = \{\theta_3\ \theta_2\ \theta_1\ \theta_0\}_2$ where,

$$\left.\begin{aligned} \theta_3 &= \gamma_3 + \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_0 + \gamma_2 \\ \theta_2 &= \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_2\gamma_0 + \gamma_3\gamma_0 + \gamma_2 + \gamma_2\gamma_1 \\ \theta_1 &= \gamma_3 + \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_1\gamma_0 + \gamma_2 + \gamma_2\gamma_0 + \gamma_1 \\ \theta_0 &= \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_2\gamma_0 + \gamma_3\gamma_1 + \gamma_3\gamma_1\gamma_0 + \\ &\gamma_3\gamma_0 + \gamma_2 + \gamma_2\gamma_1 + \gamma_2\gamma_1\gamma_0 + \gamma_1 + \gamma_0 \end{aligned}\right\} \tag{4.15}$$

4.3 Proposed S-box/ InvS-box architecture

A new proposed SubByte and InvSubByte that merges the sub-component of the typical multiplicative inverse, using a circuit minimisation technique to optimise and reduce the hardware complexity of the circuit as discussed in section 4.2, consists of Stage 1, the inversion and the combination of multiplication in GF(2^4).

Stage 1 includes a logic optimisation of multiplication in GF(2^4), multiplication with constant, squaring in GF(2^4), and addition included in one circuit. CombineXAXB is minimised for multiplication in GF(2^4) after multiplicative inversion in GF(2^4). This new architecture reduces the gate count compared to a typical circuit using typical composite field architecture as in reference [4, 12, 79]. The comparison between these references is made based on the same architecture employs a composite field with similar polynomial coefficients, and the implementation of constant multiplication with λ.

4.3.1 Stage 1

This architecture was developed by merging the transformation of multiplication in GF(2^4), multiplication with lambda, squaring in the GF(2^4) and modulo-2 addition process based on composite field arithmetic. The logic expression is present in the low-complexity formulation, and consists of 24 XOR gates, and 16 AND gates, with critical path delay of four gates.

Lemma 4.1: Let the input of Stage 1 be $w = \{w_3\ w_2\ w_1\ w_0\}_2$ and $q = \{q_3\ q_2\ q_1\ q_0\}_2$. The output is $\gamma = \{\gamma_3\ \gamma_2\ \gamma_1\ \gamma_0\}_2$.

$\gamma = \{\ squarer + multiplication\ with\ lambda + multiplication\ in\ \mathrm{GF}(2^4) +$ addition$\}$ (4.16)

The formulation equation for output, γ of Stage 1 is as follows:

$$\left.\begin{aligned}
\gamma_3 &= q_3 + q_0 + m_0 w_3 + m_1 \partial + m_2 \chi + m_3 \varrho \\
\gamma_2 &= \kappa + q_1 + m_0 w_2 + m_1 w_3 + m_2 \upsilon + m_3 \chi \\
\gamma_1 &= \kappa + m_0 w_1 + m_1 \varkappa + m_2 \partial + m_3 w_2 \\
\gamma_0 &= q_2 + m_0 w_0 + m_1 w_1 + m_2 w_3 + m_3 \partial
\end{aligned}\right\} \quad (4.17)$$

where $\chi = w_3 + w_1$, $\upsilon = w_2 + w_0$, $\partial = w_3 + w_2$, $\varkappa = w_0 + w_1$, $\varrho = \chi + \upsilon$ and $\kappa = q_3 + q_2$

Proof: The equation (4.10) and (4.11) are joined together. The output for the combination of squaring and multiplication by lambda obtains:

$l = \{q_3 + q_0, q_3 + q_2 + q_1, q_3 + q_2, q_2\}$ (4.18)

For modulo-2 addition and multiplication in GF(2^4), the equations (4.12) – (4.14) are used. By using subexpression sharing, the derived equation is:

$$\left.\begin{aligned}
\eta_3 &= m_0 w_3 + m_1 (w_3 + w_2) + m_2 (w_3 + w_1) + m_3 (w_3 + w_1 + w_3 + w_2) \\
\eta_2 &= m_0 w_2 + m_1 w_3 + m_2 (w_2 + w_0) + m_3 (w_3 + w_1) \\
\eta_1 &= m_0 w_1 + m_1 (w_0 + w_1) + m_2 (w_3 + w_2) + m_3 w_2 \\
\eta_0 &= m_0 w_0 + m_1 w_1 + m_2 w_3 + m_3 (w_3 + w_2)
\end{aligned}\right\} \quad (4.19)$$

where $m_3 = q_3 + w_3,\ m_2 = q_2 + w_2, m_1 = q_1 + w_1$ and $m_0 = q_0 + w_0$

The output for Stage 1 is found by adding modulo-2 between output l and η, to reach equation (4.17).

Table 4.1 *Gate count comparison between typical composite field architecture and proposed Stage 1*

Architecture	**Total num. of XOR gate**	**Total num. of AND gate**	**Critical path delay (gate)**
multiplication in *GF(2^4)* multiplication with lambda squaring in *GF(2^4)* modulo-2 addition [4, 12, 79]	37	9	6
Stage 1	24	14	4

4.3.2 Simplification of multiplicative inverse of nibble in GF(2^4)

Lemma 4.2: Let the input of the inversion in GF(2^4) be $\gamma = \{\gamma_3\ \gamma_2\ \gamma_1\ \gamma_0\}_2$. The output is $\theta = \{\theta_3\ \theta_2\ \theta_1\ \theta_0\}_2$. The formulation result obtained using polynomial basis with φ = $\{10\}_2$ is as follows:

$$\left.\begin{aligned}
\theta_3 &= \gamma_3\overline{\gamma_0} + \gamma_2(\overline{\gamma_3\gamma_1}) \\
\theta_2 &= \gamma_3(\gamma_0 \cup \gamma_2) + \gamma_2\overline{\gamma_1} \\
\theta_1 &= \gamma_1\overline{\gamma_3\gamma_2} + \gamma_3\overline{\gamma_1\gamma_0} + \gamma_2\overline{\gamma_0} \\
\theta_0 &= (\gamma_0 + \gamma_1)\overline{\gamma_3\gamma_2} + \gamma_2(\gamma_0 \cup \overline{\gamma_1})
\end{aligned}\right\} \quad (4.20)$$

where ∪ and + are OR gate and XOR gate implementation, respectively.

Proof: Equation (4.15) is simplified using Boolean minimisation with theorems in equations (4.21) - (4.22) to reach the result in equation (4.19).

$$\bar{x} = x + 1 \tag{4.21}$$

$$\bar{x} + x.y = \bar{x} \cup y \tag{4.22}$$

$$x + \bar{x}.y = x \cup y \tag{4.23}$$

From θ_3 in equation (4.15), θ_3 in equation (4.20) can be obtained using equation (4.21),

$$\begin{aligned}\theta_3 &= \gamma_3 + \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_0 + \gamma_2 \\ &= \gamma_3(1 + \gamma_0) + \gamma_2(1 + \gamma_3\gamma_1) \\ &= \gamma_3\overline{\gamma_0} + \gamma_2(\overline{\gamma_3\gamma_1})\end{aligned} \tag{4.24}$$

To obtain θ_2 in equation (4.20), the θ_2 in equation (4.15) is factored and substituted with equations (4.21) and (4.22).

$$\begin{aligned}\theta_2 &= \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_2\gamma_0 + \gamma_3\gamma_0 + \gamma_2 + \gamma_2\gamma_1 \\ &= \gamma_3\gamma_0(\gamma_2 + 1) + \gamma_3\gamma_2\gamma_1 + \gamma_2(1 + \gamma_1) \\ &= \gamma_3\gamma_0\overline{\gamma_2} + \gamma_3\gamma_2\gamma_1 + \gamma_2\overline{\gamma_1} \\ &= \gamma_3\gamma_0\overline{\gamma_2} + \gamma_2(\gamma_3\gamma_1 + \overline{\gamma_1}) \\ &= \gamma_3\gamma_0\overline{\gamma_2} + (\gamma_2\gamma_3 \cup \gamma_2\overline{\gamma_1}) \\ &= \gamma_3(\gamma_0\overline{\gamma_2} \cup \gamma_2) + \gamma_2\overline{\gamma_1} \\ &= \gamma_3(\gamma_0 \cup \gamma_2) + \gamma_2\overline{\gamma_1}\end{aligned} \tag{4.25}$$

The formulation of θ_1 in equation (4.20) can be proved by using equation (4.21).

$$\begin{aligned}\theta_1 &= \gamma_3 + \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_1\gamma_0 + \gamma_2 + \gamma_2\gamma_0 + \gamma_1 \\ &= \gamma_3(1 + \gamma_3\gamma_2) + \gamma_3(\gamma_1\gamma_0 + 1) + \gamma_2(1 + \gamma_0) \\ &= \gamma_1\overline{\gamma_3\gamma_2} + \gamma_3\overline{\gamma_1\gamma_0} + \gamma_2\overline{\gamma_0}\end{aligned} \tag{4.26}$$

Proving θ_0 in equation (4.20) can be obtained according to θ_0 in equation (4.15). Using the Boolean equation in equations (4.21), (4.22) and (4.23), the following is reached:

$$\theta_1 = \gamma_3\gamma_2\gamma_1 + \gamma_3\gamma_2\gamma_0 + \gamma_3\gamma_1 + \gamma_3\gamma_1\gamma_0 + \gamma_3\gamma_0 + \gamma_2 + \gamma_2\gamma_1 + \gamma_2\gamma_1\gamma_0 + \gamma_1 + \gamma_0$$
$$= \gamma_0(1 + \gamma_3\gamma_2) + \gamma_1(1 + \gamma_3\gamma_2) + \gamma_2(1 + \gamma_1\gamma_0) + \gamma_3\gamma_1(1 + \gamma_0) + \gamma_3\gamma_0 + \gamma_2\gamma_1$$
$$= (1 + \gamma_3\gamma_2)(\gamma_0 + \gamma_1) + \gamma_2((1 + \gamma_1\gamma_0) + \gamma_1) + \gamma_3\gamma_1\overline{\gamma_0} + \gamma_3\gamma_0$$
$$= (1 + \gamma_3\gamma_2)(\gamma_0 + \gamma_1) + \gamma_2((1 + \gamma_1\gamma_0) + \gamma_1) + \gamma_3\gamma_1\overline{\gamma_0} + \gamma_3\gamma_0$$
$$= (1 + \gamma_3\gamma_2)(\gamma_0 + \gamma_1) + \gamma_2(\gamma_1\gamma_0 + \overline{\gamma_1}) + \gamma_3(\gamma_1\overline{\gamma_0} + \gamma_0)$$
$$= (1 + \gamma_3\gamma_2)(\gamma_0 + \gamma_1) + \gamma_2(\gamma_0 \cup \overline{\gamma_1}) + \gamma_3(\gamma_1 \cup \gamma_0)$$
$$= \gamma_0(1 + \gamma_3\gamma_2) + \gamma_1(1 + \gamma_3\gamma_2) + \gamma_2(\gamma_0 \cup \overline{\gamma_1})$$
$$= (\gamma_0 + \gamma_1)\overline{\gamma_3\gamma_2} + \gamma_2(\gamma_0 \cup \overline{\gamma_1}) \quad (4.27)$$

Table 4.2 *Gate count comparison between typical inversion in GF(2^4) composite field architecture and proposed inversion in GF(2^4)*

Inversion in *GF(2^4)* Architecture	**Total num. of XOR gate**	**Total num. of AND gate**	**Total num. of NAND gate**	**Total num. of OR gate**	**Critical path delay (gate)**
[4, 12, 79]	14	8	-	-	5
Proposed	6	9	3	1	3

4.3.3 CombineXAXB

The architecture of CombineXAXB represents the merging of two multiplications in *GF (2^4)*, after the multiplicative inverse of nibbles in GF(2^4), using Boolean simplification to achieve a low gate count for this architecture.

Lemma 4.3: Let $\Lambda = \{\Lambda_7\ \Lambda_6\ \Lambda_5\ \Lambda_4\ \Lambda_3\ \Lambda_2\ \Lambda_1\ \Lambda_0\}_2$ be the output, with the input of $\theta = \{\theta_3\ \theta_2\ \theta_1\ \theta_0\}_2$, $m = \{m_3\ m_2\ m_1\ m_0\}_2$ and $q = \{q_3\ q_2\ q_1\ q_0\}_2$

The equations of the result are following:

$$\left.\begin{aligned}
\Lambda_7 &= q_0\theta_3 + q_1\varepsilon + q_2\alpha + q_3\zeta \\
\Lambda_6 &= q_0\theta_2 + q_1\theta_3 + q_2\beta + q_3\alpha \\
\Lambda_5 &= q_0\theta_1 + q_1\eta + q_2\varepsilon + q_3\,\theta_2 \\
\Lambda_4 &= q_0\theta_0 + q_1\theta_1 + q_2\theta_3 + q_3\varepsilon \\
\Lambda_3 &= m_0\theta_3 + m_1\varepsilon + m_2\alpha + m_3\zeta \\
\Lambda_2 &= m_0\theta_2 + m_1\theta_3 + m_2\beta + m_3\alpha \\
\Lambda_1 &= m_0\theta_1 + m_1\eta + m_2\varepsilon + m_3\,\theta_2 \\
\Lambda_0 &= m_0\theta_0 + m_1\theta_1 + m_2\theta_3 + m_3\varepsilon
\end{aligned}\right\} \quad (4.28)$$

Proof: By adding equation (4.12) for two of the multiplications in GF (2^4), the output equation of Λ are following:

$$\left.\begin{aligned}
\Lambda_7 &= q_0\theta_3 + q_1(\theta_3 + \theta_2) + q_2(\theta_3 + \theta_1) + q_3(\theta_3 + \theta_2 + \theta_1 + \theta_0) \\
\Lambda_6 &= q_0\theta_2 + q_1\theta_3 + q_2(\theta_2 + \theta_0) + q_3(\theta_3 + \theta_1) \\
\Lambda_5 &= q_0\theta_1 + q_1(\theta_1 + \theta_0) + q_2(\theta_3 + \theta_2) + q_3\,\theta_2 \\
\Lambda_4 &= q_0\theta_0 + q_1\theta_1 + q_2\theta_3 + q_3(\theta_3 + \theta_2) \\
\Lambda_3 &= m_0\theta_3 + m_1(\theta_3 + \theta_2) + m_2(\theta_3 + \theta_1) + m_3(\theta_3 + \theta_2 + \theta_1 + \theta_0) \\
\Lambda_2 &= m_0\theta_2 + m_1\theta_3 + m_2(\theta_2 + \theta_0) + m_3(\theta_3 + \theta_1) \\
\Lambda_1 &= m_0\theta_1 + m_1(\theta_1 + \theta_0) + m_2(\theta_3 + \theta_2) + m_3\,\theta_2 \\
\Lambda_0 &= m_0\theta_0 + m_1\theta_1 + m_2\theta_3 + m_3(\theta_3 + \theta_2)
\end{aligned}\right\} \quad (4.29)$$

Substitute $\alpha = \theta_3 + \theta_1$,$\beta = \theta_0 + \theta_2$, $\varepsilon = \theta_3 + \theta_2$, $\zeta = \alpha + \beta$ and $\eta = \theta_1 + \theta_0$ to prove equation (4.28).

Table 4.3 *Gate count comparison between typical multiplication in GF(2^4) and proposed CombineXAXB*

Architecture	**Total num. of XOR gate**	**Total num. of AND gate**	**Critical path delay (gate)**
two multiplication in *GF(2^4)* [4, 12, 79]	42	18	5
CombineXAXB	28	15	4

4.4 Hardware Implementation of S-box/ InvS-box

The new proposed S-box architecture illustrated in Figure 4.2 is implemented to perform both encryption and decryption, with the S-box and InvS-box sharing the same hardware simply by switching combinatorial logic blocks using multiplexers. It is an improved modification of the architecture using a composite field based on the polynomial basis. This modification enables the implementation of inverse SubBytes for decryption by reusing the same S-box resources. Technique of merging the sub-component of the typical multiplicative inverse has reduced the hardware complexity as discussed in section 4.3.

Another technique to reduce the area complexity and power consumption, an appropriate logic gate style should be used. Logic style can affect the size of a transistor, wiring load and power dissipation, especially in full-custom implementation. Full custom-design implementation offers an alternative to use a differential logic style. Rather than using the standard cell, the transistor gate can be scaled down without deteriorating performance. As most functionalities of the S-box are based on combinational logic dominated by an XOR gate, it is wise to use good performance and a low XOR gate count. Hence, by using the novel low-power and low-area XOR1 gate discussed in Chapter 3, circuit level optimised S-box/ InvS-box hardware, in terms of silicon area and power dissipation, has been achieved.

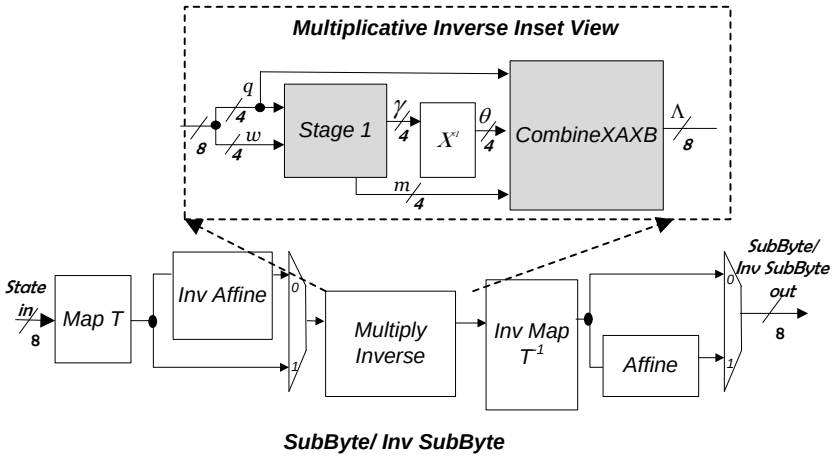


Figure 4.2 *Proposed Multiplicative Inverse in GF(2^8) architecture*

4.5 Evaluation and comparison with other designs

The complete circuit simulation, optimisation, layout and parasitic extraction were carried out using Mentor Graphics tools. The mask layout of the S-box/ InvS-box illustrated in Figure 4.3 was customised (with manual *placement* and *routing*) in 130nm IBM CMOS, with copious instances of the novel XOR gate for the CMOS Galois field/ composite field arithmetic. A 130nm CMOS technology has been selected for the implementation because it is one of the advanced CMOS technology offers different flavors of the process and also of the limited resources provided by the foundry.

Minimum channel length was used for all devices, and optimum channel width was carefully chosen for each device to achieve verified functionality with low power dissipation, and the smallest possible propagation delay. Figure 4.4 shows the complete S-box/ InvS-box chip with the bonding I/O pads.

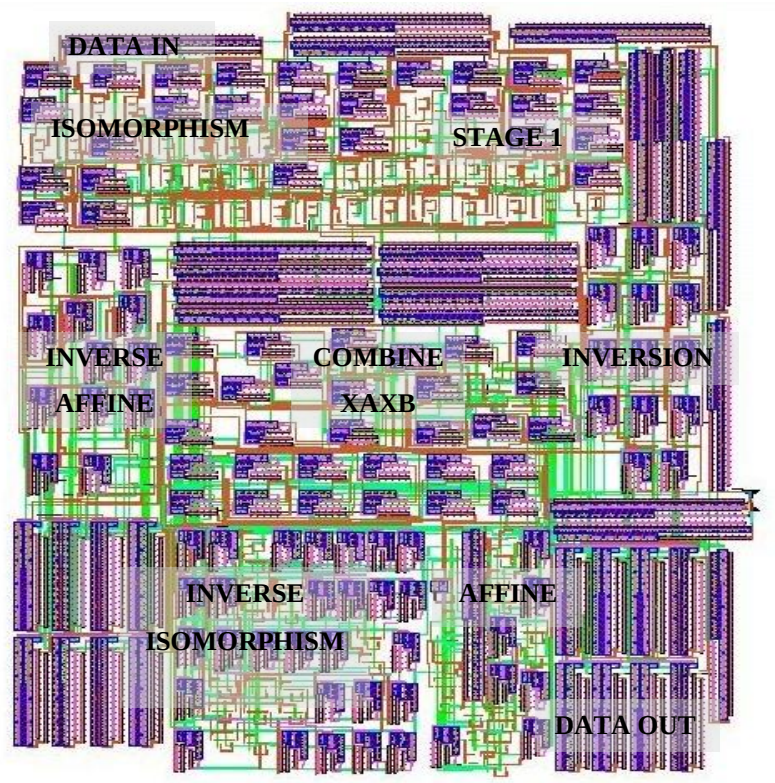


Figure 4.3 *Complete layout of the S-box/ InvS-box.*

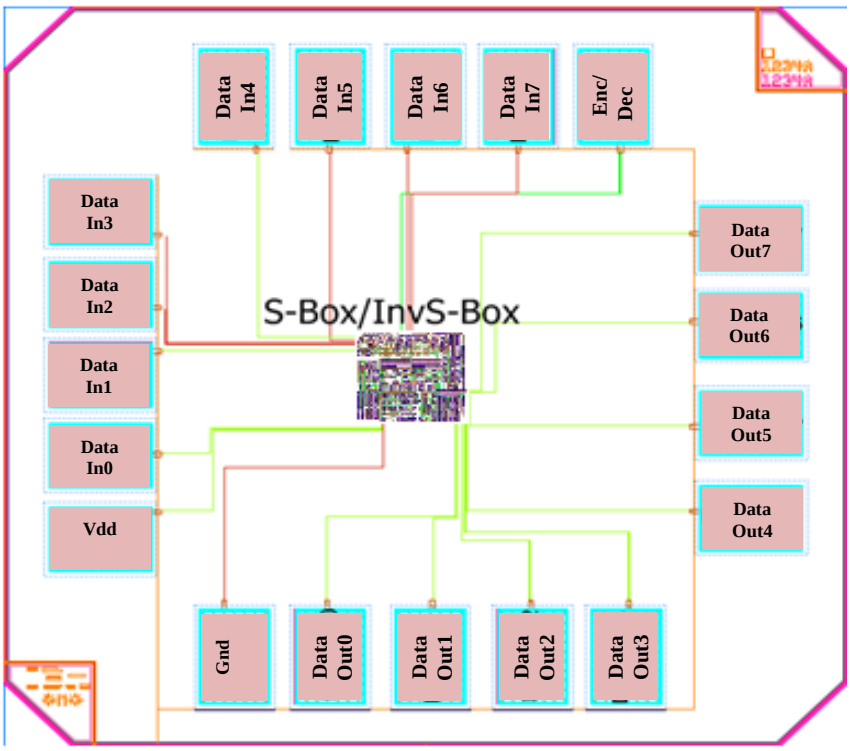


Figure 4.4 *Complete chip die of the S-box/ InvS-box with bonding pads.*

The design simulations consisted of functional verification, power and timing analysis, as well as design rule checking (DRC), and layout vs. schematic (LVS). Sets of NIST test vectors were selected randomly and used to verify the functionality. Simulation results verified the correct functionality for every input SubByte combination with a supply voltage of only 0.8V.

Figures 4.5 and 4.6 display the correct SubBytes and Inverse SubBytes functional operations for several computation cycles, respectively. The worst case S-box input-to-output delay was around 3.235ns, thus allowing a throughput of around 309 Mega-SubBytes per second. The silicon-area of the S-box/ InvS-box is only 39.44 square µm using 130nm CMOS process, and offers to-date, the smallest reported silicon-area of any implementation with shared S-box and inverse S-box.

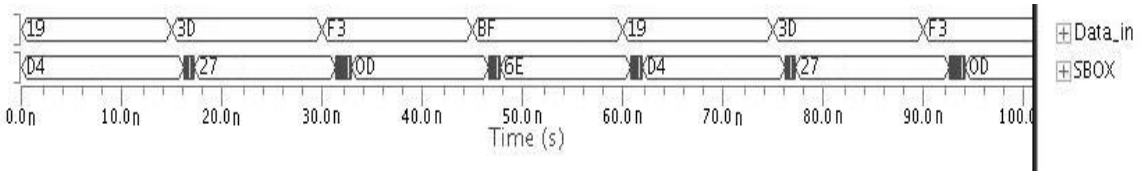


Figure 4.5 *S-box functional test verification of the SubBytes operation.*

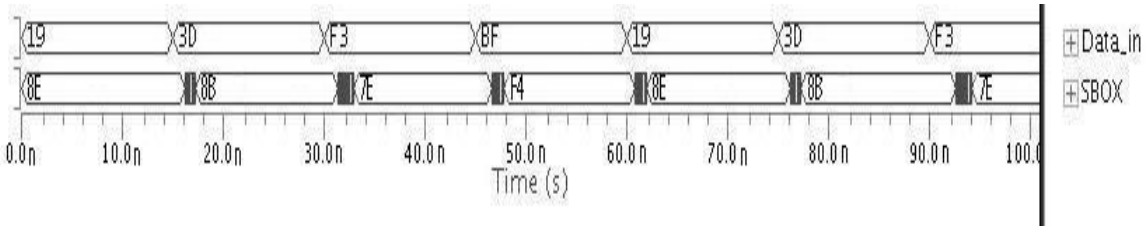


Figure 4.6 *InvS-box functional test verification of the Inverse SubBytes operation*

Design complexity was measured by using theoretical time and space complexity. Table 4.4 shows the list and count of various logic gates (and transistors) for the different blocks of the S-box/ InvS-box implementation, with the used notation of XOR and τxor, which are denoted for space and time complexity, indicating the large number of XORs dominating the transistor budget, with a total chip device count of 876.

This architecture achieves a small area using only 147 gates (105 XOR gates, 38 AND gates, 3 NAND gates and 1 OR gate), which is about 17% and 7% smaller than the typical composite field SubByte architectures recorded by Satoh et al. [4] and Nabihah Ahmad and Rezaul Hasan [79], with 172 gates (137 XOR gates and 35 AND gates) and

158 gates (123 XOR gates and 35 AND gates), respectively. For multiplicative inverse, the proposed architecture has a critical path of 11 gate delay compared to a 17 gate delay for a typical composite field design which is cut off by more than 5%.

Table 4.5 shows a comparison of the proposed implementation results with other recent S-box designs, including the normal basis design in [14], which is also discussed in [80]. Only the proposed design and S-box design in [4], [12] and [79] support both encryption and decryption transformation. According to this table, the proposed implementation achieves the lowest core area through the circuit level optimisation, saving about 80% of the area cost of design [79]. The second-lowest area S-box/ InvS-box design reported by the authors in [79] was 288 square μm using 65 nm technology.

Table 4.4 *Complexity of proposed S-box/ InvS-box implementation*

Architecture	**Total num. of XOR gate**	**Total num. of AND gate**	**Total num. of NAND gate**	**Total num. of OR gate**	**Critical path delay τxor**
Stage 1	24	14	-		4
Inversion in GF(2^4)	6	9	3	1	3
CombineXAXB	28	15	-		4
Total	**58**	**38**	**3**	**1**	**11**
Map T (ISO)	12	-	-		
Map T^{-1} (inv ISO)	12	-	-		
Affine	12	-	-		
Inv Affine	12	-	-		
Total all	**105**	**38**	**3**	**1**	**11**

In [4] and [12], the area of the S-box was presented in terms of gate equivalent, which are 381GE and 725GE in 130nm technology, respectively. Even though only implemented for encryption transformation, other designs consume more area. Extensive performance factors were considered, including additional criteria such as power-delay product (PDP), energy-delay product (EDP), and power-area product (PAP), when comparing the different S-Box circuits. EDP and PAP were calculated

using the average power dissipation and the worst-case delay. The PAP parameter is used to define a design which has small silicon area and low-power dissipation, while EDP is used to evaluate the energy efficiency of the circuit.

As shown in Table 4.5, the proposed design with 7.33 µW (@ 100 MHz) has the third lowest power dissipation, when compared to the 65nm CMOS S-Box in [79] and [44] with 0.09 µW (@ 125 MHz) and 0.04 µW (@ 10 MHz) respectively, but the lowest power dissipation when compared to the 130nm CMOS S-box in [44] and [12]. It has a lower delay time than the design in [44]. However, the S-box in [44] is only designed for SubBytes transformation, whereas the proposed design includes both the transformations, with S-box and inverse S-box sharing the same hardware. Designs in [79] and [44] use the 65nm node with low threshold voltage, allowing ultra-low Vdd (0.3–0.5V). By using deep nanometer CMOS along with voltage scaling as an effective constraint, the proposed architecture can achieve an even lower power budget than in [79]. In [44] the area of the design is not mentioned, so the PAP calculation for [44] was not available to compare with our lowest PAP result in the table, of 0.0259 $mW.\mu m^2$ (@ 125 MHz).

Furthermore, our design achieved the best PAP, EDP and PDP values when compared to all the other designs, including the normal basis design in [81] (@ 610 MHz), except for a design in [44] implemented in 65nm CMOS technology and [79]. However, the S-box design in [44] does not support decryption mode and have higher delay time and low throughput, while in [79], it requires more silicon area and delay time than the proposed design. Although the design in [12] has the shortest delay times for encryption and decryption implementation, it consumes 90% more power than the proposed design. In addition, it is to be noted that the design in [81] does not consider the implementation of the inverse S-box for decryption. When the same frequency (cycles/sec) of operation is considered for all the designs in Table 4.5, the throughput rate (Gbps) is the equivalent for all cases. This is because one 8-bit SubBytes output data is available per cycle. Therefore, for example, if the frequency is reduced to a low value of 5MHz, the throughput will be 0.04 Gbps for all the designs in Table 4.5. However, at the same reduced frequency, the proposed design will have significantly reduced power dissipation (with reduced logic switching) and will consequently, be the most energy-efficient design in Table 4.5.

Table 4.5 *AES S-box performance and comparison with previous work.*

	Proposed	**[79]**	**[27, 80]**	**[80],[81]**	**[44]**	**[44]**	**[28]**	**[20]**	**[4]**	**[12]**
Technology (nm)	130	65	65	65	130	65	250	250	130	130
Year	2012	2012	2012	2010	2009	2009	2007	2007	2001	2003
Vdd (V)	0.8	0.8	NA	NA	1.2	0.8	2.5	2.5	1.5	1.5
Decryption	yes	yes	no	no	no	no	no	no	yes	Yes
Frequency	100	125	763	610	10	10	10	10	10	10
Delay (ns)	3.235	7.322	1.31	1.64	3.3	7.5	4.39	9	3.69	2.00
Average power (µW)	7.33	0.09	54.99	44.39	12.1	0.037	207.3	140	NA	79
PDP (fJ)	23.71	0.659	72.04	72.8	39.93	0.278	910.05	1260	NA	158
Area (μm^2)	39.44	288	525.2	403.2	NA	NA	10791	8744	381GE	725GE
EDP (yJ.s)	76.70	4.825	94.37	119.39	131.77	20.85	3995.12	113400	NA	3160
PAP (mW. μm^2)	0.2891	0.0259	28.88	17.89	NA	NA	2236.97	1224	NA	NA
Throughput (Gbps)	0.8	1.0	6.1	4.9	0.08	0.08	NA	NA	NA	NA

*GE implies Gate Equivalent, NA implies not available

4.6 Conclusions

In conclusion, this chapter presents a new full custom hardware implementation of a low-power AES S-box/ InvS-box GF(2^8) Galois field inversion based on the polynomial basis, using composite field arithmetic architecture in the 130nm CMOS process, employing circuit level optimisation. The design demonstrated a new approach to minimise the silicon-area of an S-box by using a new 2-input XOR gate for low-power composite field arithmetic, in order to reduce the power dissipation and delays for the overall circuit.

Efficient design of S-box/ InvS-box was achieved by minimising the combinational logic and merging the sub-blocks of multiplicative inverse. Investigation of several S-box architectures was done in terms of power, area, throughput and delay, and compared to the proposed design. The area of the core circuit is approximately 39.44 square µm, while the hardware cost of the S-box/ InvS-box is about 147 logic gates, equivalent to 876 transistors with a critical path propagation delay of 3.235ns. Furthermore, based upon the performance of the implementation, the results indicate that this design is suitable for applications that require small-area and low-power consumption, such as RFID tags.

CHAPTER 5

FAULT DETECTION SCHEME OF AES S-BOX/ INVS-BOX USING PARITY PREDICTION BASED METHOD

5.1 Introduction

There are various possible faults in the hardware implementation of a cryptographic algorithm, for example physical faults, such as a wire bonding failure and gate oxide breakdown, which may cause a temporary or permanent fault. It can also be caused by a fault injection by attackers into the cryptography unit, in order to retrieve the internal key and obtain secret information. This physical injection attack can be done using an electrical noise injection on the power source or clock signal to create a malfunction in operating units.

Concurrent fault detection plays a vital role in hardware implementation in order to prevent losing the original message, which can cause an erroneous message output, and also to protect against malicious attacks that aim to extract the encryption secret key. There are two types of attacks against the cryptographic hardware: invasive and non-invasive. Invasive attacks are based on reverse engineering and require special laboratory equipment, while non-invasive attacks exploit hardware implementation weaknesses, such as the power consumption of the device. These are called side channel attacks or Differential Power Analysis (DPA), and differential fault analysis (DFA), and are based on the injection of a transient fault into the cryptosystem core.

Timing attacks exploit the timing characteristics of the implementation of operations, which used in a cipher to break the secret key. In DPA attacks, one power measurement is taken in each slot of the divided encryption time for different input plain texts, and these measurements correlate with each bit of the internal stage during encryption.

Different countermeasures against fault attacks in AES have been developed [82] and [100]. A fault detection scheme is chosen not only based on the reliability and capability of the scheme, but also on the optimal hardware complexity and critical path delay. There are various techniques for fault detection of the AES hardware implementation. The first technique is based on various forms of redundancy, either time or hardware, using the decryption module to decrypt the encrypted data and then comparing the result with the original plaintext, as proposed in [83] and [82]. This technique is used where algorithm-level, operation-level and round-level fault detection for the AES are applied, with the drawback of large area, power and delay overheads. Fault detection is presented using look-up table (LUT) implementation in [84], which requires more memory cells to generate the predicted parity bit.

Another fault detection technique is based on using an error detection code (EDC), which makes use of a comparison between the predicted parity outputs of a block from the input data, with the actual parity from computation of the output data of the block. This technique has proven to be very efficient, with low hardware cost and high fault detection. A well-known EDC is parity code error detection, with a number of parity bits capable of detecting all single bit errors and multiple bit errors, with an odd number of errors. The output parity bits of each transformation are predicted from the inputs using the prediction boxes and compared with the actual parities using the actual block.

The first parity-based fault detection method for AES was proposed by [85] using 16 parity bits for the *State*, adding one additional parity bit per byte for 128-bit data. Each S-box is modified into a 9-bit S-box. The parity bit does not change after the ShiftRows transformation, while 4 parity bits is predicted and added per byte in MixColumns transformation as it operates on 32-bits. This design has the drawback of high area complexity, especially for the predicted parity of MixColumns/ InvMixColumns. In [86], the parity bit is determined by XOR-ing the 16 parity bits of the 16 S-boxes. One bit parity is used for 128-bit data using the LUT for the S-box. The input parity of a round is compared with the predicted output parity of each the previous rounds to check the correctness of the other blocks of the round.

Most of the EDC methods focus on the S-box, as it is the only non-linear transformation in AES. A concurrent fault detection scheme proposed in [87], applies

to the joint S-box and inverse S-box. In [86], concurrent error detection uses a double parity bit for each S-box, one parity bit for the input byte, and one parity bit for the output byte, then both parities are compared to check the correctness of the S-box. The composite fields of the S-box/InvS-box are divided into sub-blocks and parity predictions in [88], [89] and [90]. The composite field S-box in [88] is divided into five partition blocks, and the predicted parity bit of each block is compared with the actual parity to obtain the error indication flag of the corresponding block. Enhancement of the double parity bit method is proposed in [91], by combining the designs in [86] and [86]. The predicted input parity bit is compared with the actual input parity of each S-box, and the indication error flag is obtained by OR-ing the 16 indication flags from each S-box. They also modified the double parity bit method in [84], by adding detection logic after ShiftRows transformation, in order to detect the error within the S-box and ShiftRows transformation.

This chapter explores the new low-cost fault detection scheme for the S-box/ InvS-box of AES, by enhancing the scheme in [88] for better protection. The S-box/ InvS-box architecture was developed using a composite field algorithm to reduce the area overhead. The proposed fault detection scheme is using a parity prediction based method.

5.2 Proposed Fault Detection Scheme for AES S-box/ InvS-box architecture

The fault detection scheme has been developed by comparing the actual parity output, and predicted parity output results in the error indication flag for the corresponding block. This scheme is implemented on the AES S-box and InvS-box architecture based on a compact composite field, using a polynomial basis. The transformation of the S-box uses an irreducible polynomial of $p(x) = x^8+x^4+x^3+x+1$ to construct the binary field, $GF(2^8)$. It consists of multiplicative inversion, followed by an affine transformation.

The proposed fault detection is presented using the low-power and low-area S-box/ InvS-box discussed in Chapter 4. The joint S-box and inverse S-box have been divided into seven blocks that cover each sub-structure inside it, with seven predicted parities.

Seven error indication flags are observed, and for zero error computation, the output of flags should be zero when compared with the actual parities. The predicted parity is obtained using the input of each block, while the actual parity is obtained from the output of each block. XOR gate implementation is utilized to compare the two parity outputs and to obtain the fault indication flag. We optimized the logic area complexity for each of the predicted parity units, to cover all faults, in every output of the S-box/InvS-box. Each block of the S-Box is modified in order to detect all single faults leading to an odd number of errors in the output.

The implementation of differential blocks and predicted parities are obtained by using the best choice of $\varphi = [92]_2$ and $\lambda=\{1000\}_2$. Figure 5.1 illustrates the block diagram of the proposed parity prediction fault detection blocks, for the composite field S-box and InvS-box. Blocks 1 and 6 cover the fault detection for isomorphic and inverse isomorphic, while blocks 2 and 7 consist of affine and inverse affine predicted parity. Blocks 3, 4 and 5 were developed to implement the fault detection for multiplicative inversion transformation, consisting of Stage 1, inversion in GF(2^4), and multiplication in GF(2^4) and CombineXAXB unit.

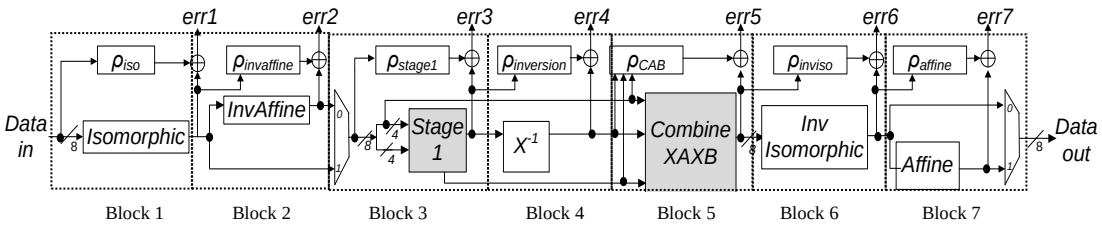


Figure 5.1 *Proposed parity prediction fault detection blocks for the composite field S-box and InvS-box*

5.2.1 Blocks 1 and 6: Predicted Parity of Isomorphic and Inverse Isomorphic Mapping

Blocks 1 and 6 represent the isomorphic and inverse isomorphic mapping based on φ = $\{10\}_2$ and λ= $\{1000\}_2$, for the best optimum logic implementation to obtain the low area and critical path delay.

Lemma 5.1: Let $\varsigma = \{\varsigma_7 \varsigma_6 \varsigma_5 \varsigma_4 \varsigma_3 \varsigma_2 \varsigma_1 \varsigma_0\}$ be the input of isomorphic mapping in GF(2^4) and $\Gamma = \{\Gamma_7 \Gamma_6 \Gamma_5 \Gamma_4 \Gamma_3 \Gamma_2 \Gamma_1 \Gamma_0\}$ be the input of predicted parities of isomorphic mapping. The derivation for the predicted parities of block 1, ρ_{iso} is as follows:

$$\rho_{iso} = \varsigma_0 + \varsigma_1 + \varsigma_2 + \varsigma_5 \tag{5.1}$$

Proof : The formulation for the predicted parities of block 1 is obtained as follows:

$$\rho_{iso} = \Gamma_0 + \Gamma_1 + \Gamma_2 + \Gamma_3 + \Gamma_4 + \Gamma_5 + \Gamma_6 + \Gamma_7 \tag{5.2}$$

By referring to the equation in (4.3), the equation obtained is:

$$\rho_{iso} = (\varsigma_0 + \varsigma_2) + (\varsigma_1 + \varsigma_6 + \varsigma_7) + (\varsigma_2 + \varsigma_5) + (\varsigma_1 + \varsigma_3 + \varsigma_6 + \varsigma_7) + (\varsigma_1 + \varsigma_5 + \varsigma_7) + (\varsigma_1 + \varsigma_4 + \varsigma_5 + \varsigma_6) + (\varsigma_1 + \varsigma_2 + \varsigma_3 + \varsigma_4 + \varsigma_5 + \varsigma_6) + (\varsigma_5 + \varsigma_7) \tag{5.3}$$

By using the XOR property for Boolean minimisation, $X + X = 0$, the equation in (5.1) can be obtained. The total number of XOR gates needed for implementation of block 1, ρ_{iso} in the S-Box/ InvS-box is three XOR gates.

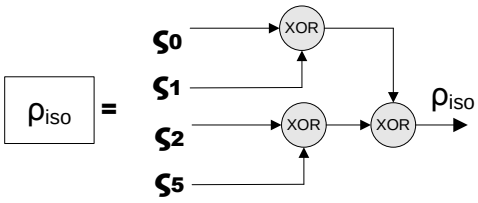


Figure 5.2 *Predicted Parity of Isomorphic Mapping*

Lemma 5.2: Let $\Lambda = \{\Lambda_7 \Lambda_6 \Lambda_5 \Lambda_4 \Lambda_3 \Lambda_2 \Lambda_1 \Lambda_0\}$ be the input of inverse isomorphic mapping in GF(2^4), and $\Gamma' = \{\Gamma'_7 \Gamma'_6 \Gamma'_5 \Gamma'_4 \Gamma'_3 \Gamma'_2 \Gamma'_1 \Gamma'_0\}$ be the input of the predicted parity of inverse isomorphic mapping. The predicted parity of block 6, ρ_{inviso} is obtained as follows:

$$\rho_{inviso} = \Lambda_0 + \Lambda_2 + \Lambda_6 + \Lambda_7 \tag{5.4}$$

Proof: The formulation for the predicted parities for the inverse isomorphic mapping of block 6 is obtained by using equation in (4.4):

$$\rho_{inviso} = \Gamma'_0 + \Gamma'_1 + \Gamma'_2 + \Gamma'_3 + \Gamma'_4 + \Gamma'_5 + \Gamma'_6 + \Gamma'_7 \tag{5.5}$$

$$\rho_{inviso} = (\Lambda_0 + \Lambda_1 + \Lambda_3 + \Lambda_5 + \Lambda_6) + (\Lambda_4 + \Lambda_7) + (\Lambda_1 + \Lambda_3 + \Lambda_5 + \Lambda_6) + (\Lambda_1 + \Lambda_3) + (\Lambda_1 + \Lambda_5 + \Lambda_7) + (\Lambda_1 + \Lambda_2 + \Lambda_3 + \Lambda_5 + \Lambda_6) + (\Lambda_2 + \Lambda_3 + \Lambda_4 + \Lambda_5 + \Lambda_6) + (\Lambda_1 + \Lambda_2 + \Lambda_3 + \Lambda_5 + \Lambda_6 + \Lambda_7) \tag{5.6}$$

Using similar methods to previous, the equation in (5.4) can be obtained.

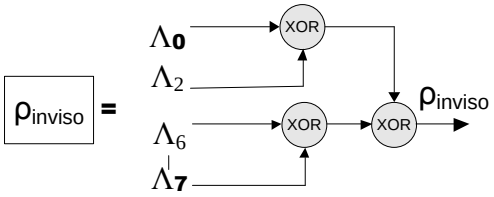


Figure 5.3 *Predicted Parity of Inverse Isomorphic Mapping*

The hardware implementation complexity for the predicted parities of block 6, ρ_{inviso} is three XOR gates.

5.2.2 Block 3: Parity Stage 1

Block 3 of the S-box/ InvS-box implements Stage 1 architecture, which consists of multiplication in GF(2^4), multiplication with lambda, squaring in GF(2^4) and a modulo-2 addition process based on composite field arithmetic.

Lemma 5.3: Let the input of Stage 1 be $w = \{w_3\ w_2\ w_1\ w_0\}_2$ and $q = \{q_3\ q_2\ q_1\ q_0\}_2$, while $\gamma = \{\gamma_3\ \gamma_2\ \gamma_1\ \gamma_0\}_2$ is the input of the predicted parity of Stage 1. The predicted parity of block 3, ρ_{stage1} as follows:

$$\rho_{stage1} = (w_0 \cup (q_0 + q_1 + q_2 + q_3)) + w_1(q_0 + q_2) + w_2(q_0 + q_1 + q_3) + w_3(q_0 + q_2 + \overline{q_3}) \quad (5.7)$$

where ∪ represents the OR operation.

Proof: The formula for the predicted parity of Stage 1 is obtained by the total output of Stage 1.

$$\rho_{stage1} = \gamma_0 + \gamma_1 + \gamma_2 + \gamma_3 + \gamma_4 + \gamma_5 + \gamma_6 + \gamma_7 \quad (5.8)$$

By using the equation stated in (4.17), the formula for the predicted parity of Stage 1 is:

$$\rho_{stage1} = (q_0 + q_1 + q_2 + q_3) + q_0w_0 + q_0w_1 + q_0w_2 + q_0w_3 + q_1w_0 + q_1w_2 + q_2w_0 + q_2w_1 + q_2w_3 + q_3w_0 + q_3w_2 + q_3w_3 + w_0 + w_3 \quad (5.9)$$

Using $X + 1 = \bar{X}$ and $X + \bar{X}Y = X \cup Y$, the predicted parity of Stage 1 as an equation in (5.7) is obtained. The hardware implementation of the predicted parity for block 3 requires seven XOR gates, three AND gates, one OR gate, and one inverter gate as shown in Figure 5.4.

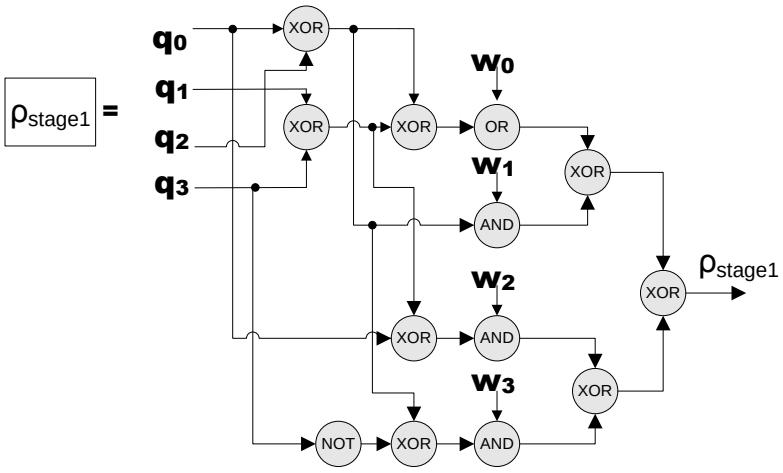


Figure 5.4 *Predicted Parity circuit for Stage 1 implementation*

5.2.3 Block 4: Parity Inversion

Lemma 5.4: Let the input of the inversion in GF(2^4) be $\gamma = \{\gamma_3\ \gamma_2\ \gamma_1\ \gamma_0\}_2$ and $\theta = \{\theta_3\ \theta_2\ \theta_1\ \theta_0\}_2$ is the input for the predicted parity of the inversion. The derivations of the predicted parity inversion, $\rho_{inversion}$ are obtained as follows:

$$\rho_{inversion} = \gamma_0(\overline{\gamma_2\gamma_1}) + \gamma_3(\gamma_0 + \gamma_1) \tag{5.10}$$

Proof: The formula of the predicted parity for inversion is as follows:

$$\rho_{inversion} = \theta_0 + \theta_1 + \theta_2 + \theta_3 \tag{5.11}$$

Substituting the above equation with the equation in (4.20), the following formulation is obtained:

$$\rho_{inversion} = \gamma_1\gamma_2\gamma_3 + \gamma_0\gamma_3 + \gamma_3 + \gamma_2 + \gamma_1\gamma_2\gamma_3 + \gamma_0\gamma_2\gamma_3 + \gamma_0\gamma_3 + \gamma_1\gamma_2 + \gamma_2 + \gamma_1\gamma_2\gamma_3 + \gamma_0\gamma_1\gamma_3 + \gamma_0\gamma_2 + \gamma_3 + \gamma_2 + \gamma_1 + \gamma_1\gamma_2\gamma_3 + \gamma_0\gamma_2\gamma_3 + \gamma_0\gamma_1\gamma_3 + \gamma_0\gamma_1\gamma_2 + \gamma_1\gamma_3 + \gamma_0\gamma_3 + \gamma_1\gamma_2 + \gamma_0 + \gamma_1 + \gamma_2 \tag{5.12}$$

The above equation is minimised using $X + 1 = \bar{X}$ to get equation (5.10).

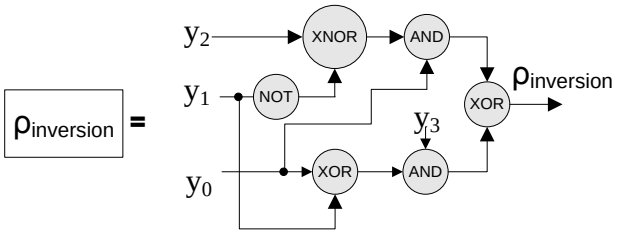


Figure 5.5 *Predicted Parity circuit for inversion in GF(2^4) implementation*

Figure 5.5 illustrates the hardware implementation for the predicted parity of inversion, which utilises two XOR gates, two AND gates, one NAND gate and one inverter.

5.2.4 Block 5: Parity CombineXAXB

Block 5 consists of two multiplications in GF(2^4), after the multiplicative inverse of nibble in GF(2^4). The architecture is optimized using a Boolean simplification technique in order to achieve a low gate count.

Lemma 5.5: Let $\theta = \{\theta_3\ \theta_2\ \theta_1\ \theta_0\}_2$, $m = \{m_3\ m_2\ m_1\ m_0\}_2$ and $q = \{q_3\ q_2\ q_1\ q_0\}_2$ be the input of CombineXAXB, while $\Lambda = \{\Lambda_7\ \Lambda_6\ \Lambda_5\ \Lambda_4\ \Lambda_3\ \Lambda_2\ \Lambda_1\ \Lambda_0\}_2$ is the input for the predicted parity of CombineXAXB. The derivation of the the predicted parity is:

$$\rho_{CAB} = (\Psi + \theta_3)(q_3 + m_3) + (£ + \theta_0)(q_2 + m_2) + \Psi(q_1 + m_1) + €(q_0 + m_0) \tag{5.13}$$

where $\Psi = \theta_2 + \theta_0$, $£ = \theta_1 + \theta_3$ and $€ = \Psi + £$

Proof: The formula for the predicted parity of CombineXAXB is derived by totalling the output of CombineXAXB.

$$\rho_{CAB} = \Lambda_0 + \Lambda_1 + \Lambda_2 + \Lambda_3 + \Lambda_4 + \Lambda_5 + \Lambda_6 + \Lambda_7 \tag{5.14}$$

By using the equation stated in (4.28), the equation for the predicted parity of CombineXAXB is:

$$\rho_{CAB} = m_0\theta_0 + m_1\theta_1 + m_2\theta_3 + m_3(\theta_3 + \theta_2) + m_0\theta_1 + m_1(\theta_1 + \theta_0) + m_2(\theta_3 + \theta_2) + m_3\theta_2 + m_0\theta_2 + m_1\theta_3 + m_2(\theta_2 + \theta_0) + m_3(\theta_3 + \theta_1) + m_0\theta_3 + m_1(\theta_3 + \theta_2) + m_2(\theta_3 + \theta_1) + m_3(\theta_3 + \theta_2 + \theta_1 + \theta_0) + q_0\theta_0 + q_1\theta_1 + q_2\theta_3 + q_3(\theta_3 + \theta_2) + q_0\theta_1 + q_1(\theta_1 + \theta_0) + q_2(\theta_3 + \theta_2) + q_3\theta_2 + q_0\theta_2 + q_1\theta_3 + q_2(\theta_2 + \theta_0) + q_3(\theta_3 + \theta_1) + q_0\theta_3 + q_1(\theta_3 + \theta_2) + q_2(\theta_3 + \theta_1) + q_3(\theta_3 + \theta_2 + \theta_1 + \theta_0) \quad (5.15)$$

The above equation is minimized to obtain the new equation below:

$$\rho_{CAB} = (\theta_2 + \theta_0 + \theta_3)(q_3 + m_3) + (\theta_1 + \theta_3 + \theta_0)(q_2 + m_2) + \theta_2 + \theta_0(q_1 + m_1) + (\theta_2 + \theta_0 + \theta_1 + \theta_3)(q_0 + m_0) \quad (5.16)$$

This equation is substituted with the variables to obtain the equation in (5.13).

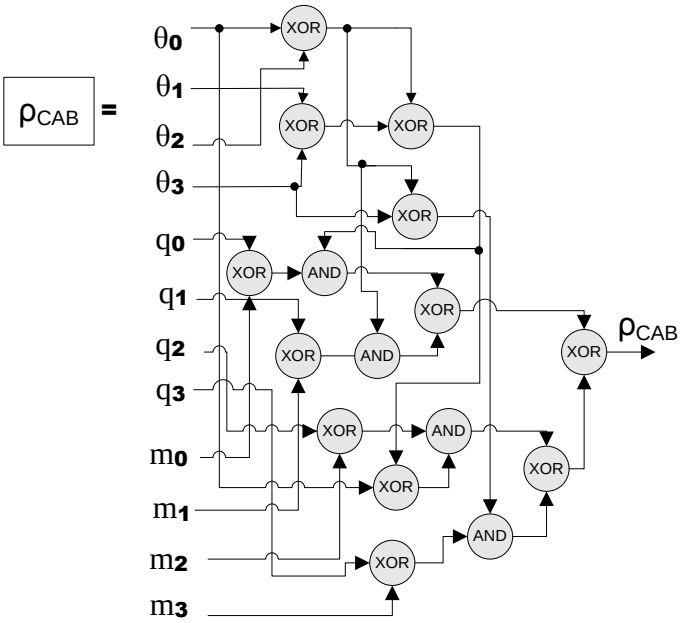


Figure 5.6 *Predicted parity circuit of CombineXAXB implementation*

The number of gates required for implementing the predicted parity of block 5, ρ_{CAB} is 15 XOR gates and four AND gates, as shown in Figure 5.6.

5.2.5 Blocks 2 and 7: Parity Affine and Inverse Affine

Lemma 5.6: Let $\Gamma' = \{\Gamma'_7 \Gamma'_6 \Gamma'_5 \Gamma'_4 \Gamma'_3 \Gamma'_2 \Gamma'_1 \Gamma'_0\}$ be the input of affine in GF(2^4). The derivation for the predicted parities of block 2 is as follows:

$$\rho_{affine} = \overline{c + b} + \overline{b + e + \Gamma'_6} + \overline{a + d + \Gamma'_1} + \overline{d + c} + b + e + d$$

(5.17)

Proof: By substituting $a = \Gamma'_4 + \Gamma'_5, c = a + \Gamma'_6, b = \Gamma'_0 + \Gamma'_7, d = \Gamma'_2 + \Gamma'_3$ and $e = \Gamma'_1 + \Gamma'_5$, into equation (5.18), the equation in (5.17) is obtained for the predicted parities of block 2.

$$\rho_{affine} = \overline{\Gamma'_4 + \Gamma'_5 + \Gamma'_6 + \Gamma'_0 + \Gamma'_7} + \overline{\Gamma'_0 + \Gamma'_7 + \Gamma'_1 + \Gamma'_5 + \Gamma'_6} + \overline{\Gamma'_4 + \Gamma'_5 + \Gamma'_2 + \Gamma'_3 + \Gamma'_1} + \overline{\Gamma'_2 + \Gamma'_3 + \Gamma'_4 + \Gamma'_5 + \Gamma'_6} + \Gamma'_0 + \Gamma'_7 + \Gamma'_1 + \Gamma'_5 + \Gamma'_2 + \Gamma'_3$$

(5.18)

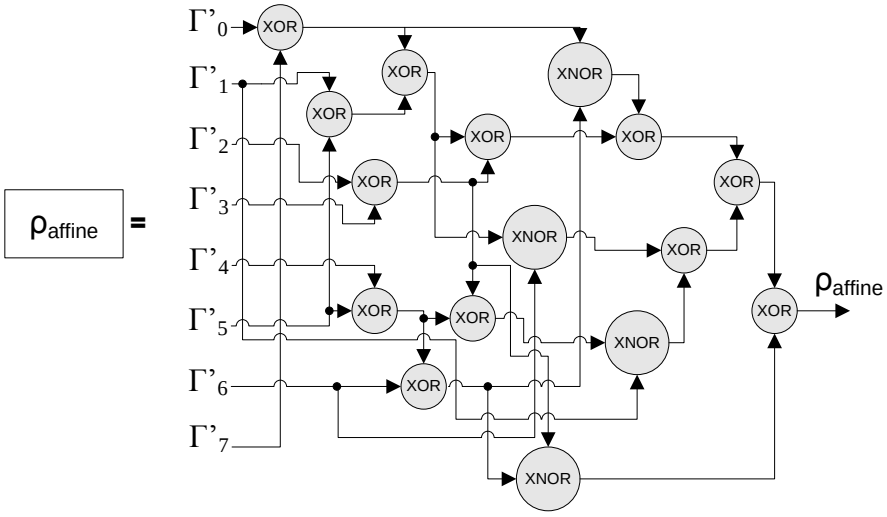


Figure 5.7 *Predicted parity circuit of affine implementation*

Hardware implementation for the predicted parity of block 2, ρ_{affine} requires 12 XOR gates and four XNOR gates.

Lemma 5.7: Let the input of the inverse affine be $\Gamma = \{\Gamma_7 \Gamma_6 \Gamma_5 \Gamma_4 \Gamma_3 \Gamma_2 \Gamma_1 \Gamma_0\}$. The predicted parity of Stage 1 is derived as follows:

$$\rho_{invaffine} = \overline{\Gamma_2 + \Gamma_5 + \Gamma_7} + \overline{\Gamma_1 + \Gamma_4 + \Gamma_7} + \Gamma_0 + \Gamma_3 + \Gamma_6 + \Gamma_7 \quad (5.19)$$

Proof: For $\Gamma = \{\Gamma_7 \Gamma_6 \Gamma_5 \Gamma_4 \Gamma_3 \Gamma_2 \Gamma_1 \Gamma_0\}$ as the input and $\rho_{invaffine}$ as the output, the equation for the predicted parity block of the inverse affine is as follows:

$$\rho_{invaffine} = \overline{\Gamma_2 + \Gamma_5 + \Gamma_7} + (\Gamma_0 + \Gamma_3 + \Gamma_6) + \overline{\Gamma_1 + \Gamma_4 + \Gamma_7} + (\Gamma_0 + \Gamma_2 + \Gamma_5) + (\Gamma_1 + \Gamma_3 + \Gamma_6) + (\Gamma_2 + \Gamma_4 + \Gamma_7) + (\Gamma_0 + \Gamma_3 + \Gamma_5) + (\Gamma_1 + \Gamma_4 + \Gamma_6) \quad (5.20)$$

By using Boolean algebra for minimisation, the equation in (5.19) is obtained. The number of gates needed for implementing the predicted parity of block 7, $\rho_{invaffine}$ shown in Figure 5.8 is seven XOR gates and two XNOR gates.

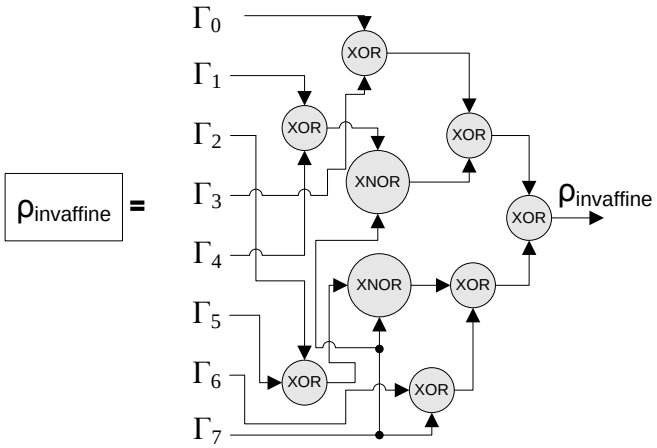


Figure 5.8 *Predicted parity circuit of inverse affine implementation*

5.3 Hardware Complexity Analysis

The total area implementation for the fault detection predicted parity block of the S-box/ InvS-box required 49 XORs, six XNORs, nine ANDs, one inverter, two ORs and one NAND gate. Table 5.1 summarises the hardware complexities for each of the predicted parities for blocks 1- 7.

Table 5.1 *Hardware complexities for proposed predicted parity of S-box/ InvS-box*

Block	XOR	XNOR	AND	INV	OR	NAND
1, ρ_{iso}	3	-	-	-	-	-
2, ρ_{affine}	12	4	-	-	-	-
3, ρ_{Stage1}	7	-	3	1	1	
4, $\rho_{inversion}$	2	-	2	-	1	1
5, ρ_{CAB}	15	-	4	-	-	-
6, ρ_{inviso}	3	-	-	-	-	-
7, $\rho_{invaffine}$	7	2	-	-	-	-
Total	**49**	**6**	**9**	**1**	**2**	**1**

The actual parities for each block of the S-box/ InvS-box required an XOR gate to obtain the output parity, to compare with the predicted parity. Furthermore, seven XOR gates are needed to obtain the indication flag, by comparing seven of the predicted blocks with the actual parities.

5.4 Fault coverage of the proposed fault detection scheme

The proposed fault detection scheme was simulated using 130nm CMOS technology, in the Mentor Graphic environment. The evaluation for single and multiple stuck-at errors model were carried out to evaluate the fault coverage of the proposed fault detection. Stuck-at error model is a functional fault on a Boolean (logic) function implementation that force nodes to be stuck at logic one or zero independent of the fault-free logic values.

All possible single stuck-at errors were inserted randomly on the input and output nodes of the logic gates of the S-box. Nodes in the circuit are selected randomly to be force with the fault. Fifty data inputs for the S-box/ InvS-box were selected and the correct input of each block was replaced by an erroneous value, corresponding to a

stuck-at fault at an input line of each block. The output error is detected by comparing the parity bit with the actual parity of the outputs. All the single faults will result in single errors in an odd number of erroneous bits at its output, and all the possible faults are detected by parity checking at each of the blocks and ends of the S-box/ InvS-box block. The proposed fault detection was also injected with multiple stuck-at errors, by using a Fibonacci implementation of the LFSRs where, the numbers, the locations and the types of the errors are randomly selected. 50 nodes were made faulty for a multiple fault.

This simulation proves that the predicted parity fault detection has almost 100% fault coverage at the byte level. For a single stuck-at error, it shows that the faults are covered 99.9 % for both entire SubBytes and inverse SubBytes. For multiple stuck-at errors, 96% fault coverage resulted, which covers 48 nodes that were identified from the 50 injected nodes in both the S-box and the inverse S-box. It is expected that the error coverage of about 100 percent if the number of errors injected is increased. Table 5.2 represents the fault coverage for single and multiple stuck-at errors for the S-box and inverse S-box.

Table 5.2 *Fault coverage for fault detection scheme*

Faults	**Fault coverage (%)**
Single stuck-at errors	99.9
Multiple stuck-at errors	96

5.5 Conclusions

In this chapter, the new fault detection scheme, based on parity bits, has been developed for the S-box/ Inv S-box architecture. It has been shown that the proposed fault detection scheme, using the new optimum composite field S-box/ InvS-box, has lower complexities and delay overheads than other previous designs. Based on the simulation results, high fault coverage was obtained for the proposed fault detection scheme. This scheme also offers low hardware complexities, which contributes to a low cost and low power dissipation design estimated about 20uW.

CHAPTER 6

EFFICIENT INTEGRATED AES CRYPTO - PROCESSOR ARCHITECTURE FOR 8-BIT STREAM CIPHER

6.1 Introduction

This chapter describes the new compact AES 8-bit stream cipher architecture. The first section explains the top level design of the AES core, and is followed by discussion on the architecture of each component part of AES core and the comparative performance of other existing AES designs.

6.2 AES Architecture

The new AES core supports 128-bit keys in an 8-bit wide stream datapath, in order to minimise the silicon area and power consumption at high speed, using circuit optimisation techniques. Figure 6.1 illustrates the block diagram of an AES crypto-processor, while Figure 6.2 represents the top-core AES architecture of the proposed design. The component parts are: the SubByte/ InvSubByte, MixColumns/ InvMixColumns, ShiftRows/ InvShiftRows, Key Scheduling unit, control unit and clock distribution unit. The control unit executes the sequence of the AES operation and provides control signals to each of the AES components. Meanwhile, the clock distribution unit distributes the global clock to the components in parallel synchronism, avoiding delay mismatch. The control unit consists of a 4-bit counter that gives a signal input to the multiplexer selector to control data input in each round of operation.

The data block and key initial are fed into an AES and XORed before being substituted in SubBytes/ InvSubBytes transformation. In each round, *State* data are then processed in ShiftRows/ InvShiftRows in column order, and passed to MixColumns/ InvMixColumns transformation before being XORed with the key round. Each round is completed in 28 clock cycles. In the last round of AES operation, the data output from ShiftRows/ InvShiftRows are XORed with the last key round.

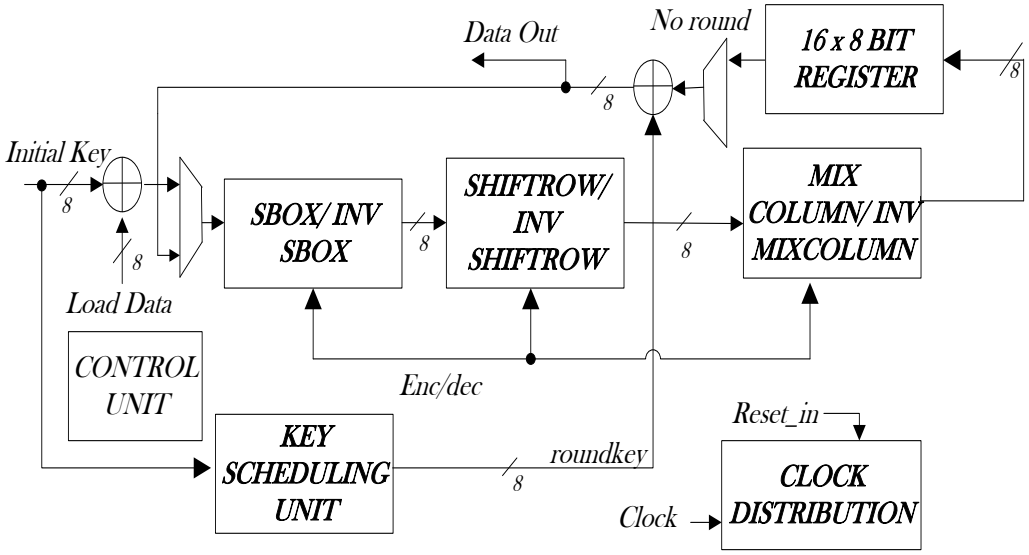


Figure 6.1 *Block diagram of AES crypto-processor*

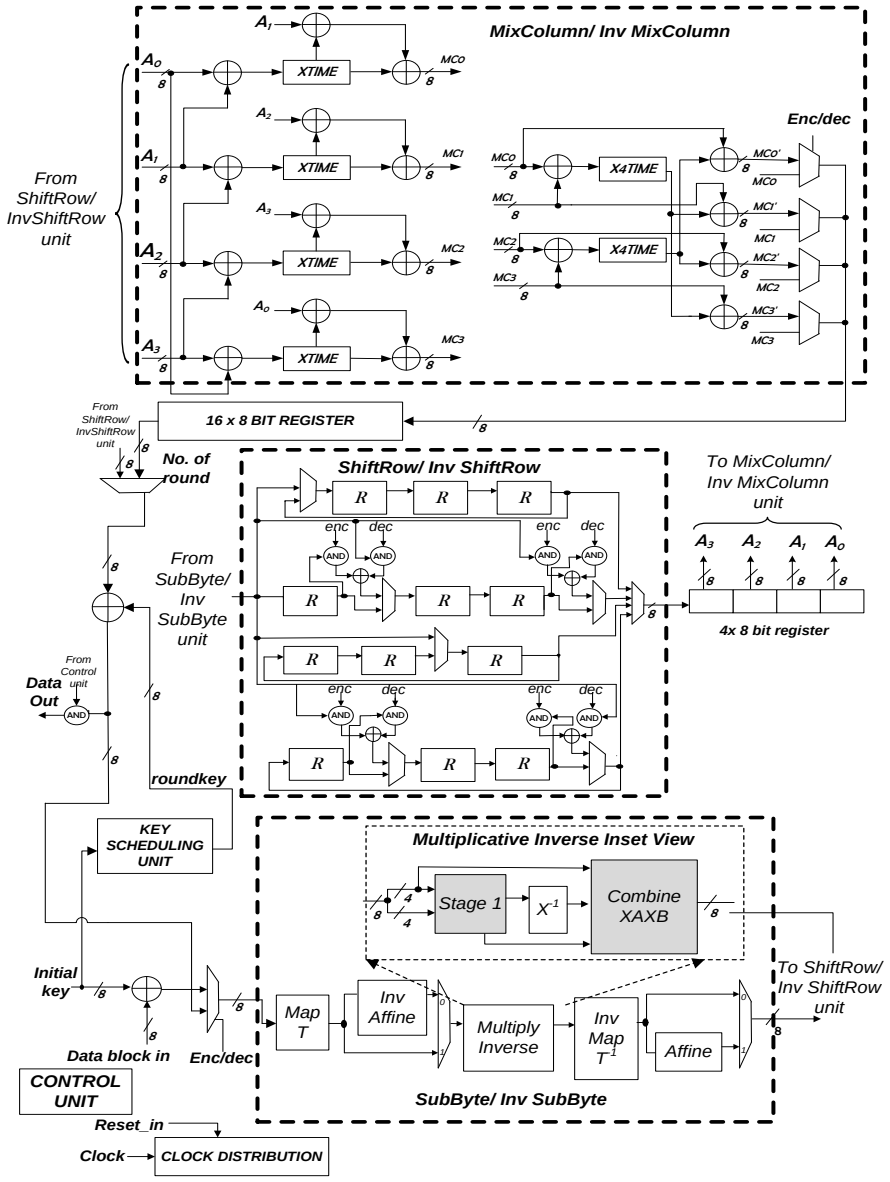


Figure 6.2 *High level architecture of the AES crypto-processor*

6.3 SubBytes and InvSubBytes Architecture

A new proposed SubBytes and InvSubBytes architecture merging the sub component of the typical multiplicative inverse to optimise the circuit is discussed in detail in Chapter 4. It consists of Stage 1, the inversion and the combination of multiplication in GF(2^4). Stage 1 includes a logic optimisation of multiplication in GF(2^4), multiplication with constant, squaring in GF(2^4), and addition transformation included in one circuit. CombineXAXB circuit is obtained by minimising the circuit of multiplication in GF(2^4) after multiplicative inversion in GF(2^4). This new architecture reduces gate

count compared to the typical circuit using typical composite field architecture. The proposed design has been compared to existing designs suitable for AES 8-bit datapath and 32-bit datapath implementation. All the results of performance analysis can be found in Chapter 4.

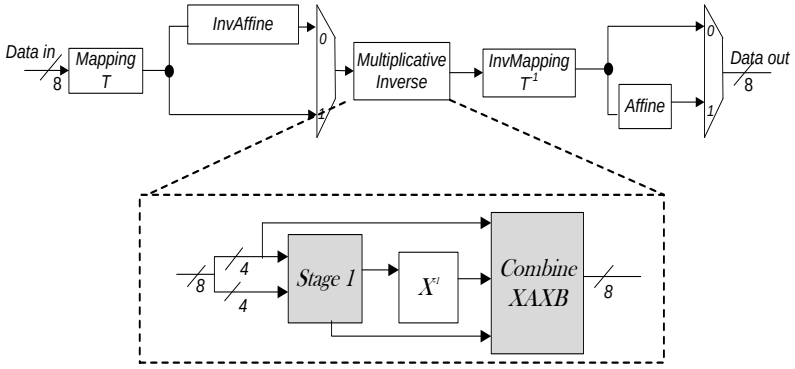


Figure 6.3 *Proposed Multiplicative Inverse in GF(2^8) transformation architecture*

6.4 New Topology of MixColumns/ InvMixColumns

MixColumns and InvMixColumns are modular multiplications by the constant vectors, and matrix expressions of the MixColumns and InvMixColumns for one output column of the *State* array are shown in equations (6.1) and (6.2) also equivalent to equation (2.11) and (2.12) respectively.

$$\begin{bmatrix} mc0 \\ mc1 \\ mc2 \\ mc3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \mathrm{X} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{6.1}$$

where $mc0$, $mc1$, $mc2$ and $mc3$ are the output *State* after MixColumns and a_0, a_1, a_2 and a_3 are the original *State*.

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \mathrm{X} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{6.2}$$

where $mc0'$, $mc1'$, $mc2'$ and $mc3'$ are the output *State* after InvMixColumns and a_0, a_1, a_2 and a_3 are the original *State*.

MixColumns/ InvMixColumns architecture is one of high area consumption in AES design, and demands efficient low-area approaches to implement it. Optimized MixColumns and InvMixColumns are implemented using the decomposition method, by applying byte-level resource sharing to the computation within a byte in a given column of the *State* matrix. Byte-level resource sharing is used to improve the logic minimisation, and reduce the hardware complexity efficiently.

The MixColumns matrix in equation (6.1) can be decomposed into a linear combination of products of the elements in GF(2^8), as multiplication over addition in GF(2^8) is distributive. The decomposed MixColumns is shown as

$$\left.\begin{aligned} mc0 &= 02(a_0 + a_1) + (a_2 + a_3) + a_1 \\ mc1 &= 02\,(a_1 + a_2) + (a_3 + a_0) + a_2 \\ mc2 &= 02\,(a_2 + a_3) + (a_0 + a_1) + a_3 \\ mc3 &= 02\,(a_0 + a_3) + (a_1 + a_2) + a_0 \end{aligned}\right\} \quad (6.3)$$

The MixColumns matrix is then expressed as:

$$\begin{bmatrix} mc0 \\ mc1 \\ mc2 \\ mc3 \end{bmatrix} = \begin{bmatrix} 02 * (a_0 + a_1) + (a_2 + a_3) + a_1 \\ 02 * (a_1 + a_2) + (a_3 + a_0) + a_2 \\ 02 * (a_2 + a_3) + (a_0 + a_1) + a_3 \\ 02 * (a_3 + a_0) + (a_1 + a_2) + a_0 \end{bmatrix} \quad (6.4)$$

Constant multiplication by {02} and {04} in GF(2^8) as mentioned in Chapter 2 are performed by Xtime(x) and X4time(x) units based on the byte-level method. The byte-level substructure sharing method finds common terms with the aim to reduce the number of Xtime(x) blocks by pre-computing additions of some variables before using the Xtime(x) unit. The Xtime(x) process is the multiplication by x, which is 02[x]. The Xtime(x) circuit is implemented using combinations of XOR gates of subsequent conditional XOR with {1B}, and hard-wired logic shift operations of left shift. The Xtime(x) unit requires 3 bit level XORs as shown in Figure 6.4.

Xtime[7:0] = {in[6], in[5], in[4], (in[3] $\oplus$ in[7]), (in[2] $\oplus$ in[7]), in[1], (in[0] $\oplus$in [7]), in[7]} (6.5)

Multiplication for the higher power of x can be done by repeat of Xtime(x). X4time(x) is the same as X4time = $(\text{Xtime})^2$ = Xtime (Xtime[x]) = Xtime (2[x]) = 04[x]. The module for X4time(x) requires 5 XORs as shown in Figure 6.4.

The matrix multiplication of MixColumns could be represented as follows:

$$\left.\begin{aligned} mc0 &= xtime(a_0 \oplus a_1) \oplus (a_2 \oplus a_3) \oplus a_1 \\ mc1 &= xtime(a_1 \oplus a_2\ ) \oplus (a_3 \oplus a_0) \oplus a_2 \\ mc2 &= xtime(a_2 \oplus a_3) \oplus (a_0 \oplus a_1) \oplus a_3 \\ mc3 &= xtime(a_3 \oplus a_0) \oplus (a_1 \oplus a_2) \oplus a_0 \end{aligned}\right\} \quad (6.6)$$

Assume d =$(a_0 \oplus a_1)$, e = $(a_1 \oplus a_2\ )$, f = $(a_2 \oplus a_3)$, and g =$(a_3 \oplus a_0)$. These are the common terms in MixColumns and can be shared to reduce hardware complexities.

$$\begin{aligned} &\text{w} = xtime(a_0 \oplus a_1), \\ &\text{x} = xtime(a_1 \oplus a_2\ ), \\ &\text{y} = xtime(a_2 \oplus a_3), \\ &\text{z} = xtime(a_3 \oplus a_0) \end{aligned} \quad (6.7)$$

Equation (6.8) is replaced with (6.7) to create a new equation:

$$\left.\begin{aligned} mc0 &= w \oplus f \oplus a_1 \\ mc1 &= x \oplus g \oplus a_2 \\ mc2 &= y \oplus d \oplus a_3 \\ mc3 &= \text{z} \oplus e \oplus a_0 \end{aligned}\right\} \quad (6.8)$$

The InvMixColumns equation (6.2) can also be expressed as equation (6.9). InvMixColumns decomposition using distributivity of GF(2^8) can be expressed as equation (6.9).

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 04 & 00 & 02 & 00 \\ 00 & 04 & 00 & 04 \\ 02 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \end{bmatrix} \mathrm{X} \begin{bmatrix} 02 & 02 & 02 & 02 \\ 00 & 00 & 02 & 02 \\ 02 & 00 & 02 & 02 \\ 00 & 00 & 02 & 02 \end{bmatrix} X \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (6.9)$$

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} =$$

$$\begin{bmatrix} 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_0+a_1)+(a_2+a_3)+a_1+4*(a_0+a_2') \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_1+a_2')+(a_3+a_0)+a_2+4*(a_1+a_3) \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_2+a_3)+(a_0+a_1)+a_3+4*(a_2+a_0) \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_3+a_0)+(a_1+a_2)+a_0+4*(a_3+a_1) \end{bmatrix} \quad (6.10)$$

In order to reduce the cost of the hardware, InvMixColumns can be decomposed to share logic resources with MixColumns. The equation in (6.10) is substituted with equation 6.4 to share the architecture, and obtain the new equation (6.11) as follows:

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 4*2*[(a_0+a_1)+(a_2+a_3)]+mc0+4*(a_0+a_2) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc1+4*(a_1+a_3) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc2+4*(a_2+a_0) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc3+4*(a_3+a_1) \end{bmatrix} \quad (6.11)$$

The matrix multiplication of InvMixColumns could be represented as follows:

$$mc0' = x4time(xtime)[(a_0 \oplus a_1)\ (a_2 \oplus a_3)] \oplus mc0 \oplus x4time(a_0 \oplus a_2)$$
$$mc1' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc1 \oplus x4time(a_1 \oplus a_3)$$
$$mc2' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc2 \oplus x4time(a_2 \oplus a_0)$$
$$mc3' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc3 \oplus x4time(a_3 \oplus a_1) \quad (6.12)$$

Assuming $s = (a_0 \oplus a_2)$, $t = (a_3 \oplus a_1)$, these are the common terms in InvMixColumns:

$$u = xtime[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] = xtime\ (d \oplus f)$$
$$p = x4time\ (u \oplus s)$$
$$v = x4time\ (u \oplus t) \quad (6.13)$$

By replacing equation (6.13) in equation (6.12), we obtained a new equation as follows, further reducing the area of the InvMixColumns:

$$mc0' = p \oplus mc0$$
$$mc1' = v \oplus mc1$$
$$mc2' = p \oplus mc2$$
$$mc3' = v \oplus mc3 \quad (6.14)$$

From the equations (6.8) and (6.14), MixColumns and InvMixColumns are optimized by sharing the same resources. The hardware implemented for MixColumns and InvMixColumns is illustrated in Figure 6.4. For 8-bit datapath implementation, we used an additional shift register and multiplexers. Shift registers are used to shift the input data to MixColumns/ InvMixColumns, and multiplexers are used to select an encryption or decryption output. MixColums/ InvMixColumns input, A_0-A_3, is an individual byte coming from ShiftRows/ InvShiftRows unit.

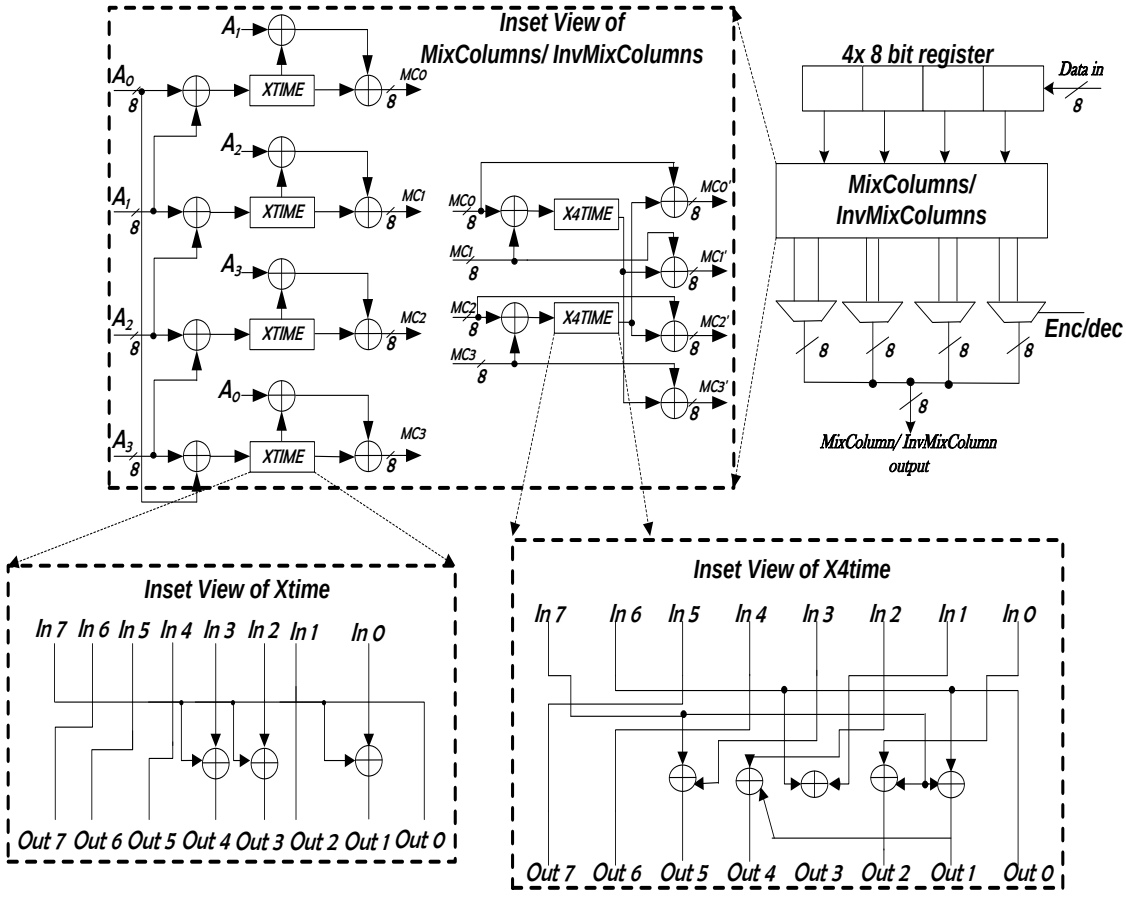


Figure 6.4 *MixColumns and InvMixColumns circuit*

For MixColumns and InvMixColumns, the total number of gates in the proposed architecture is 166 XOR gates for computation of one column of a *State*. The full *State* matrix transformation requires 664 XOR gates and results in a critical path of 7 XOR gate delay. Table 6.1 shows a comparison with other implementations, in terms of area (AXOR) and critical path delay (TXOR). The proposed integrated MixColumns and InvMixColumns design achieves a 64.8% area reduction compared to the direct implementation [36]. On the other hand, the design in [36] with a shorter critical path delay by 2 XOR gates, requires 16% more area compared to the proposed architecture. With the same area as the design in [93], the proposed architecture can lead to better performance with a shorter critical path delay.

Table 6.1 *Comparison of different MixColumn/ InvMixColumn designs*

Method	**Year**	**Mode (Enc/ Dec)**	**Area (A_{XOR}) One column**	**Area (A_{XOR}) Full state**	**Delay (τxor)**
Direct implementation	2005	Both	472	1888	5
[37]	2002	Both	304	1216	8
[33]	2004	Both	193	772	7
[36] Method 1 Method 2	2005	Both	 221 213	 884 852	 5 6
[35]	2005	Both	324	1296	6
[34] Serial Parallel	2005	Both	 192 219	 768 876	 8 7
[94]	2009	Both	314	1256	8
[93]	2010	Both	166	664	8
Proposed	2012	Both	166	664	7

6.5 ShiftRows/ InvShiftRows Architecture Implementation

ShiftRows and InvShiftRows operation is a cyclic shifting of each row of a *State* matrix over different offsets. The first row is not shifted. The shift offsets depend on the block length, *Nb*. Table 6.2 shows the different values of the offsets for each row as a function of the block length.

ShiftRows is a linear cyclic shift operation in each row of four 4-byte data blocks, with different offsets (0~3-byte offsets) to the left. The inverse of this transformation is performed by corresponding rotations to the right.

Table 6.2 *ShiftRow offset values*

Data Block length, *Nb*	**Row 1/bytes**	**Row 2/bytes**	**Row 3/bytes**
4	1	2	3
6	1	2	3
8	1	3	4

The new proposed AES core employs a joint ShiftRows and InvShiftRows architecture using twelve 8-bit registers along with six 2-to-1 and one 4-to-1 multiplexer as shown in Figure 6.5. This design is also suitable for AES 32-bit datapath implementation. The operation of the registers is controlled by four different clocks generated by clock distribution, CLK1, CLK2, CLK3 and CLK4 for each row. These different clocks are distributed synchronously with the clock system to generate each output for row in different time period. Sel_load signal is used to enable the register to load data or to shift data through the registers. The shift registers are used to shift and hold the current *State* data before being replaced by the next *State*, until the ShiftRows/ InvShiftRows operation is completed. Multiplexers are used to select the datapath for the input of the register, either from the external data input or the input data feedback from the previous register.

The first row of registers sequentially shifts the data through the registers. When comparing the ShiftRows and InvShiftRows operation, the same hardware architecture is used for all rows, using a different control signal according to the encryption and decryption processes. All data generated after 12 cycles requires one 4-to-1 multiplexer, with a select signal that controls the row generating the data out, and another additional AND gate to control the data shifted out only after 12 cycles.

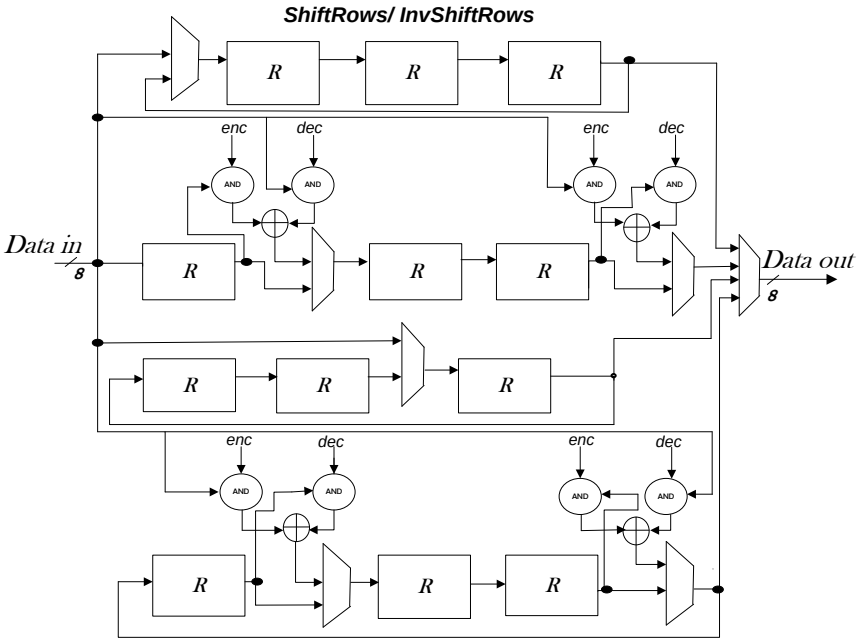


Figure 6.5 *ShiftRows and InvShiftRows circuit for 8-bit datapath*

Table 6.3 indicates the cost for different implementations based on the number of registers and multiplexers in GE measure. The new proposed ShiftRows/ InvShiftRows offers less hardware complexity when compared to the design in [32], which also implements an 8-bit AES datapath. The design in [32] requires an additional four 8-bit registers and one multiplexer. The proposed architecture requiring 448 gates (equivalent to 1792 transistors) reduces the hardware area by 22%. In addition, the proposed ShiftRows/ InvShiftRows has the lowest gate equivalent when compared to other designs.

Table 6.3 *Comparison of total hardware cost for different ShiftRows/ InvShiftRows designs*

Method	Year	Mode (Enc/ Dec)	Data path	Register	Multiplexer	Gate equivalent (GE)
Direct implementation [36] Shift Row Inv Shift Row	2005	 Enc Dec	 128 128	 2 x 128 bit 2 x 128 bit	 - -	 896 896
Byte permutation [32] Shift Row	2006	Enc	8	16 x 8 bit	7 (2 to 1), 1 (4 to 1)	572
[30]	2007	Both	32	16 x 8 bit	7 (2 to 1)	532
[95]	2008	Both	32	7 x 32 bit	2 (2 to 1)	808
[101]	2005	Both	32	20 x 8 bit	2 (2 to 1)	584
Proposed	2012	Both	8	12 x 8 bit	6 (2 to 1) 1 (4 to 1)	448

6.6 Add Round Key Implementation

In the initial round, the initial key is XORed with the initial data block. During the standard round, the round key generated from the key scheduling unit is XORed with the output obtained from the MixColumns transformation. While in the final round, the output obtained from ShiftRow transformation is XORed with the round key.

6.7 Key Scheduling Unit

The two main components of the key scheduling unit are the key expansion and the round key selection. An AES Key Expansion algorithm is used to derive the round key from the original encryption key shown in List 6.1.

The initial key is the cipher key, and is used in the initial round of the algorithm. The key expansion expanded from the cipher key into an expanded key array, consists of 4 rows and *Nb* (*Nr*+1) columns and has a total number equal to the block length

multiplied by the number of rounds plus 1. *Nb* is the number of words in an AES block, and is equal to 4. The key length is variable between 128, 192 and 256-bits. The initial key consists of *Nk* words, or 4**Nk* bytes. The expanded key, w consists of *Nb**(*Nr*+1) words, or 4**Nb**(*Nr*+1) bytes as shown in Table 6.4.

```
int Nb = 4;
int Nk = 4, 6, or 8;
void KeyExpansion(byte[] key, word[] w, int Nw) {
int Nr = Nk + 6;
w = new byte[4*Nb*(Nr+1)];
int temp;
int i = 0;
while ( i < Nk) {
w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]);
i++;
}
i = Nk;
while(i < Nb*(Nr+1)) {
temp = w[i-1];
if (i % Nk == 0)
temp = SubWord(RotWord(temp)) ^ Rcon[i/Nk];
else if (Nk > 6 && (i%Nk) == 4)
temp = SubWord(temp);
w[i] = w[i-Nk] ^ temp;
i++;
}
}
```

List 6.1 *Pseudo-code algorithm for the key expansion*

Table 6.4 *Expanded Key Sizes in Words*

Key Length (*Nk* words)	**No. of rounds (*Nr*)**	**Key Expansion Size (*Nb(Nr+1)* words)**
4	10	44
6	12	52
8	14	60

SubWord() applies the S-box value used in SubBytes for each of the 4 bytes in the argument. RotWord() is a simple cyclic permutation of a word one position to the left, which changes [a0,a1,a2,a3] to [a1,a2,a3,a0]. RON is the round constant word array, applied only on the first byte of the key input array.

RCON unit is shown in Figure 6.6, developed using an 8-bit shift register with multiplexer to control the output and XOR gates. RCON[i] generator is a 4-byte value, and the lower 3 bytes are 0 for all *i*, and the highest byte is the bit representation of the polynomial x^i mod $m(x)$ and contains values of powers of x. RCON for key generation can be found in Table 6.5.

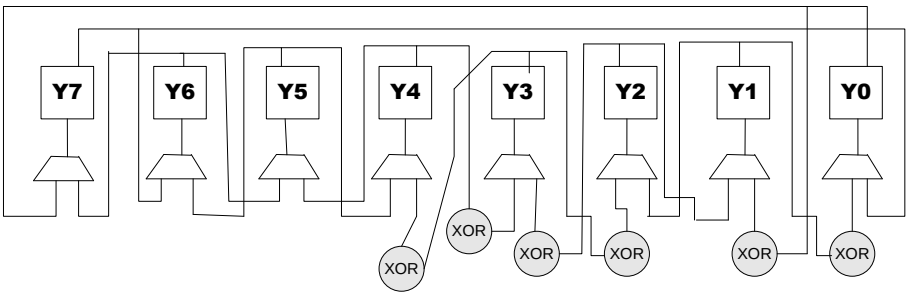


Figure 6.6 *RCON[i] Generator*

Table 6.5 *RCON[i] for key generation*

i	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
RCON[i]	00	01	02	04	08	10	20	40	80	1B	36

There are two methods to design a Key Scheduling unit. The first method is by precalculation and store all round keys, where the key is expanded once and stored in a buffer, then used for all coming data blocks until the reset operation is selected [29, 56]. This method offers simpler and faster operation when the same key is used for all the data blocks, but consumes bigger data storage hardware. Another method is on-the fly key expanded, which generates the keys for each round without storing all the keys in the memory register [9, 32, 38, 52, 58]. For a 128-bit input key, the original key will be expanded for 10 rounds, from step to step for every new coming data block, producing a 10x 128-bit key round. This method allows a new key inserted with every coming data block. Key generation for decryption is more complex, and requires more area for generation.

Key Scheduling unit proposed in [32] is used to generate the on-the fly key expansion only for encryption consists of 17 8-bit registers with 4 multiplexers, 3 XOR gates, 1 AND gate, Round Constant (RCON) and S-box unit. While in [58], the Key Scheduling unit is improved over the architecture in [32] including on-the fly key expansion for decryption. This architecture consists of 22 8-bit registers, 8 multiplexers, 4 XOR gates, 1 AND gate, InvMixColumns unit, RCON and S-box unit. They also used buffer to maintain the input of the InvMixColumns unchanged during the encryption. Both of this architecture stored the key data in the register before it had been expanded which is not mentioned in the architecture.

The new on-fly key scheduling is a developed component in an 8-bit datapath, and is able to generate key expansion sizes of 128-bit, and supports the encryption and decryption process. Figure 6.7 shows the 8-bit Key Scheduling unit which consists of 28 byte shift register and S-box unit, RCON unit, InvMixColumns unit, 13 multiplexer and 5 exclusive OR gate. This proposed architecture includes the stored data for the key which make it appear larger than architecture in [32] and [58] because in both of the design did not include the registers for key storage in the diagram. Multiplexers are used as a control signal to select an appropriate datapath, either the initial key, or the round key generated from the initial key. It uses 28 8-bit registers, either to buffer the round key, or process the round key for encryption and decryption. The RCON unit consists of combinational logic circuits.

The initial key is fed into the 16-byte shift register for sixteen consecutive clocks to form a 128-bit input. In the encryption process, every last word of the previous round key is cyclical shifted before passing through the S-Box, and XORed with the RCON value. Each transformation generates round keys as 4-byte words.

In the decryption process, round keys used in the encryption process are applied in reverse order to all rounds. The second to last word is shifted before being passed through the InvMixColumns. Key expansions that are produced are shifted into a register until all round keys are generated. The first key expansion round is generated after 28 clocks, and the last key is generated after 264 clock cycles simultaneously with AES core operation.

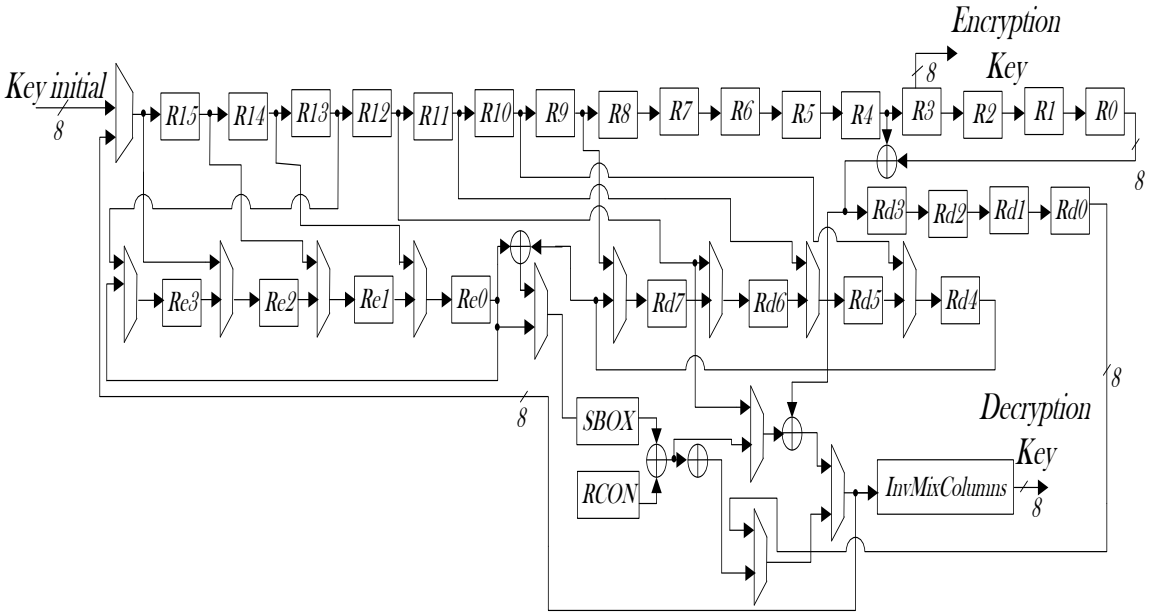


Figure 6.7 *Key Scheduling unit*

6.8 Conclusions

In this chapter, the development of a compact and high-speed AES architecture with an 8-bit stream datapath to achieve a small-area circuit with a marginal power budget, without sacrificing the throughput performance, has been discussed. The proposed architecture performs both encryption and decryption processes with a 128-bit data block and 128-bit cipher key. In order to minimise the hardware size, logic optimisation techniques were applied to each of the arithmetic components. We introduced a new architecture for S-box/ InvS-box, MixColumns/ InvMixColumns, and ShiftRows/ InvShiftRows, including the new on-the fly Key Scheduling unit for encryption and decryption. This architecture is designed as suitable for an 8-bit datapath implementation to achieve a low-power, low-area and better performance AES system.

CHAPTER 7

IMPLEMENTATION, VERIFICATION AND RESULTS OF AES CRYPTO-PROCESSOR

7.1 Introduction

This chapter discusses the physical design implementation of the AES crypto-processor system, verification of the design, and the results of the experimental methodology for design reliability. Physical design implementation covers the layout implementation and specifications. The goals of design verification were to screen the AES architecture and verify that the device met the functional specifications.

First, the methodology employed verifying the front-end block's operation is discussed; second, system performance and parameters of merit are simulated by software, including functional output, power consumption and time delay; third, the testing method is explained and discussed in order to evaluate the system performance of the fabricated microchip during the test experiments. In the end, the practical measurement result is achieved and reported.

7.2 Physical implementation

7.2.1 Layout consideration for design

There is Low Power (LP) and general purpose (GP) flavours developed in advanced CMOS technology for different power requirements using low-Vt transistors and high-Vt CMOS transistors. In this design, a low-Vt transistor is used in the critical path to provide a fast speed, but it has high leakage current. A high-Vt transistor is used for the

non-critical paths, which provide less leakage current with a drawback of low in speed. Proper transistor width and minimum channel length were chosen carefully to achieve a low-power and high-performance system. Input and output was designed to interface directly to the 8-bit data block input, key initial, and cipher out for AES. Its operation is entirely controlled by a logic-level clock, enable and reset inputs, and provides a serial digital output. Another external interface was also connected to check the correctness of the sub-block output from the AES system. Table 7.1 illustrates the list of digital I/O implemented on the AES chip.

Table 7.1 *Complete list of input and output ports on the implemented design.*

Name	**Description**	**Direction (I/O)**	**Width (bit)**
Clk	System clock input, 100MHz	I	-
rst	Reset control. Device must be reset after power-up.	I	-
en	Enable control the operation	I	1
Enc_dec	Selection for encryption or decryption process	I	1
Data_in	Data block input	I	8
Key_in	Key initial input	I	8
Cont_q0, q1, q2, q3	Output Counter Mod 10 for AES round indication	O	4
Sel_data	Selection signal for AES round	O	1
Mix_Column	Output for Mix Column unit	O	8
Key_round	Output for Key Scheduling unit	O	8
Data_out	Data out for AES system	O	8
Vdd	DC Supply voltage of 0.8V	I	-
Gnd	Ground	I	-

Design work and simulation used a supply voltage of 0.8V less than the 1.20V specified by IBM. The power pin is connected to a well-regulated and bypassed source to ensure stable operation. The AES crypto-processor chip was fabricated using the IBM 130nm CMOS process. The complete layout was submitted through the MOSIS academic program and a batch of 20 devices was received, packaged in 108-pin

ceramic PGA108M. Figure 7.1 shows the external connections to the 108-pin PGA package and Figure 7.2 illustrates the packaged AES microchip.

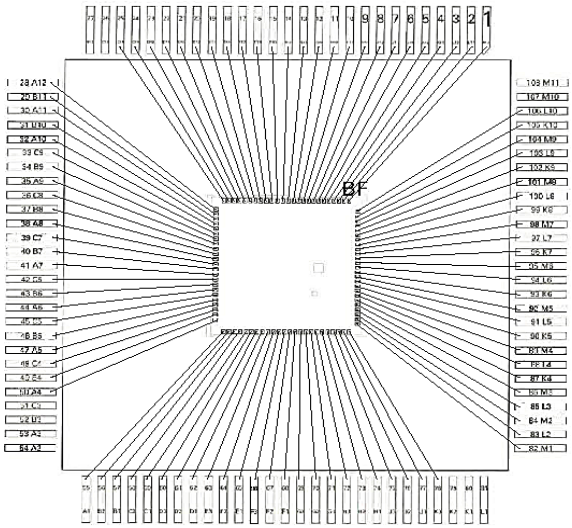


Figure 7.1 *External connections of AES crypto-processor on the 108-pin PGA package.*

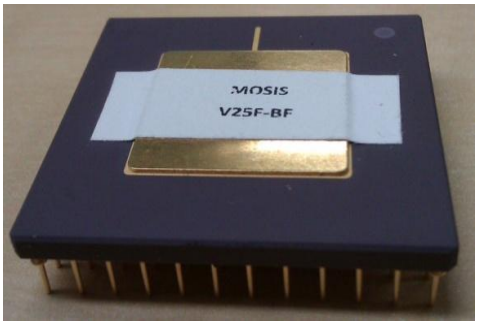


Figure 7.2 *Packaged AES microchip in PGA108M*

The final completed AES crypto-processor front-end layout diagram includes the controller and AES system, as shown in Figure 7.3. The photomicrograph of the fabricated die is shown in Figure 7.4 and has an active circuitry chip area of 640µm×325µm (0.208 square mm), excluding the bonding pads.

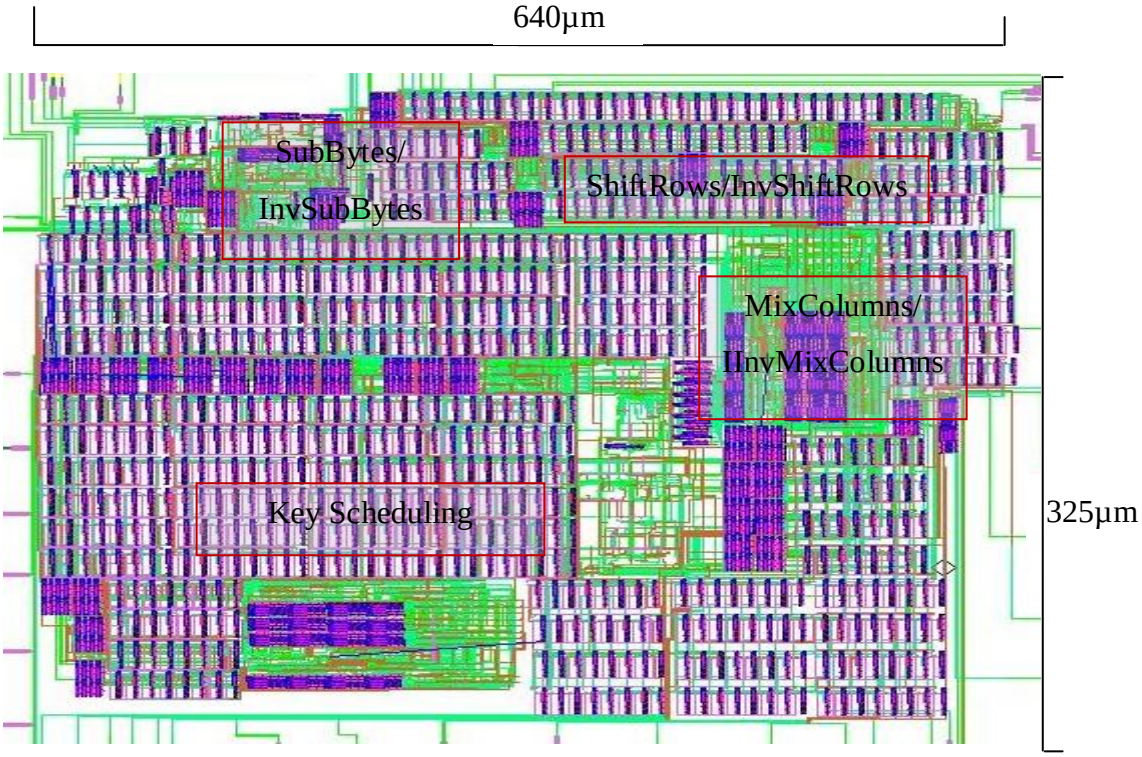


Figure 7.3 *Complete layout of active circuitry for the AES crypto-processor*

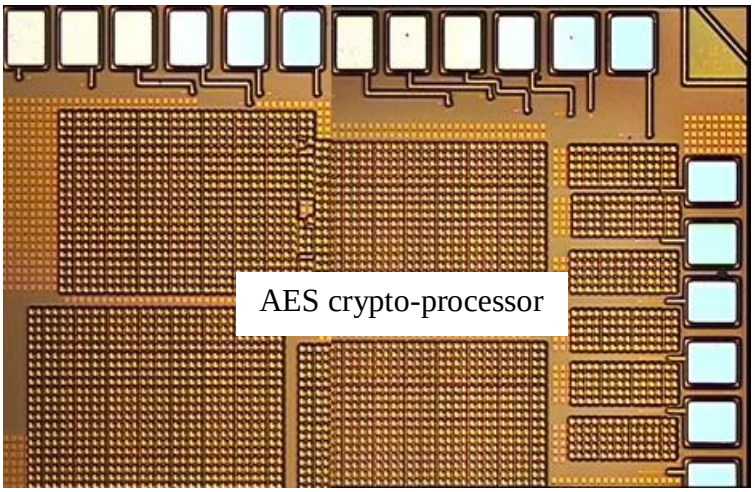


Figure 7.4 *Photomicrograph of the AES fabricated die*

7.3 Verification Methodologies

As the system design size and complexity increases, high abstraction level design methods are needed to verify system functionality. There are two general types of design verification testing: functional and physical verification.

7.3.1 Functional Verification

Functional verification is used to determine the functionality of the device against the actual system specification at high-level design and subsequently at the chip-level of the actual fabricated chip. The main objective of functional verification is to ensure the functionality of device operation, find any errors and resolve design discrepancies against the expected architectural specifications.

A set of test patterns are fed in as input on the input pins, and the correct output is verified on the output pins. The following are the properties of a functional test:

1) Used to verify the functionalities of the device
2) Test patterns must be customised for optimised verification
3) Can be used to operate the device at functional speed
4) Does not require extra on-die logic nor consume any area on-chip.

7.3.2 Physical verification

Physical verification is used to verify the physical design implementation using EDA software tools, including Design Rule Check (DRC), Layout versus Schematic (LVS), antenna check, electrical rule check (ERC), signal integrity, and power analysis. It is used to validate that the design meets the design criteria.

DRC is used to check whether the layout complies with the layout design rule parameters. This is important to make sure the layout is able to be manufactured. LVS is used to compare the layout and schematic, whether they match with each other, as well as recognising the wiring connection and electrical components of the layout.

7.4 Simulation and Measurement results

7.4.1 Composite result and discussions

During the validation phase, design errors are identified using simulation methods. Simulation verification is the process of validating a design for each of the functional components in the AES system, by simulating all the possible input combinations exhaustively using computer based tools. This type of verification methodology requires an input stimulus of a test vector, together with an actual design implementation. The simulated behaviour from the captured output value is interpreted and compared with the expected computed state from the design specification. In order to increase test coverage, this verification methodology requires the computation of all possible input test vectors.

Simulations for AES design have been completed using the circuit simulator Eldo platform, provided by Mentor Graphics. The simulation was carried out with a 0.8V supply voltage and a throughput (clocking) rate of 100MHz, with rise time and fall time of 500ps. The same testing conditions have been used to measure the propagation delay and the power dissipation in each case. Simulation results verified the correct functionality for every input data block and key initial AES combination with a supply voltage of only 0.8V. A set of 20 test vectors from NIST were randomly selected and used to verify the encryption and decryption process which covers most of the possible input test vectors.

The new AES crypto-processor architecture has been implemented in 130nm IBM CMOS technology, employing the above described optimised gate level hardware. Sets of NIST test vectors were used to verify the functionality. Operating with a 100MHz global clock (@ VDD= 0.8V) the design achieved a throughput of 0.05 Gbps.

$$Throughput = \frac{128*10^6}{280} = 45.7Mbps \approx 0.05Gbps \quad (7.1)$$

Table 7.2 compares the proposed AES chip with previous ASIC implementations. The proposed AES core requires 3152 GEs, slightly more than the smallest design in [32] which only supports the encryption mode and utilises 3100 GEs. Furthermore, the proposed design utilises a lower gate count compared to the recent design reported in [59], using the latest technology, which requires 3500 GEs. This design [59] supports both encryption and decryption algorithms in the system. The key scheduling unit is the largest part that supports an on-the-fly key generating a round key per clock, without additional memory to store the keys. Even the proposed AES chip does not have the lowest silicon area when compared to [59], as it is implemented in 65nm technology, but it has still been accepted as a small-area design. With 65nm technology, the proposed architecture can achieve a smaller silicon area than [59] because of the small scale of devices in a 65nm library and low gate count in the proposed AES architecture.

Table 7.2 also shows a summary of performance comparisons of the proposed AES with some other designs, indicating the proposed performance is achieved at low power dissipation. The proposed design offers the second lowest power consumption compared to the design in [59], which uses the 65nm node with low threshold voltage, allowing ultra-low Vdd (≈0.3-0.5V). By using deep nanometer CMOS along with voltage scaling as an effective constraint, the proposed architecture can achieve an even lower power budget design. With the same technology and supply voltage of 0.8V, the proposed design has the lower power consumption than the design in [43], which only applied the encryption algorithm.

Considering the throughput factor, the proposed AES offers higher performance, except for the designs in [32] and [58]. Moreover, higher throughput is achieved compared to the design in [59], with an improvement of 99% of the throughput. The proposed design required 280 clock cycles to complete the algorithm, higher than the designs in [32] and [58]. For a high speed system, a throughput-to-area ratio can be used as a performance metric. However, for resource critical applications, the power-area-latency (P-A-T) metric is used. The proposed AES design demonstrates the best P-A-T efficiency compared to other existing designs except for design in [59]. The design in [38] has the worst P-A-T efficiency, followed by [58].

Table 7.2 *Comparison with previous 8-bit AES design*

Design	**Year**	**Mode**	**Tech (µm)**	**V (v)**	**Cycle count**	**Core Area (mm^2)**	**Area (GEs*)**	**Through-put (Mbps)**	**Max freq (MHz)**	**Power (µW/ Mhz)**	**P-A-T (µJ-gate)**
[38]	2004	Enc	0.35	1.5	1016	NA	3595	0.013	0.1	122.25	446.52
[9]	2005	Enc/ Dec	0.35	1.5	1032	0.25	3400	9.9	80	45	29.56
[32]	2006	Enc	0.13	1.2	160	NA	3100	121	153	36.76	18.23
[43]	2010	Enc	0.13	0.75	356	210	5500	0.036	0.1	6.92	13.53
		Enc	0.13	0.8	356	210	5500	4.31	12	8.25	16.15
[58]	2010	Enc/ Dec	0.18	1.8	160	NA	5600	104	130	57	51.07
[59]	2011	Enc/ Dec	0.065	0.36	1142	0.018	3500	0.036	0.322	0.78	3.12
		Enc/ Dec	0.065	0.4	1142	0.018	3500	0.1	0.89	0.96	3.84
Ours	2012	Enc/ Dec	0.13	0.8	280	1.28	3152	45.7	100	4.23	3.2

*GE implies Gate Equivalent, NA implies not available

7.4.2 Circuit Simulation Results

All sub-blocks in the AES system were simulated using the Eldo simulator and were proven to operate correctly. The Data Block and cipher key used in this simulation (128-bit) are as follows:

Data block: 3243F6A8885A308D313198A2E0370734

Cipher key: 2B7E151628AED2A6ABF7158809CF4F3C

For graphical purposes, only the encryption process will be displayed in this chapter. Figure 7.5 illustrates the waveform simulation result for encryption SubBytes transformation. This design is independent of the clock input signal and generated after the first data in and key initial is received.

The output of SubBytes is:

After SubBytes: D42711AEE0BF98F1B8B45DE51E415230

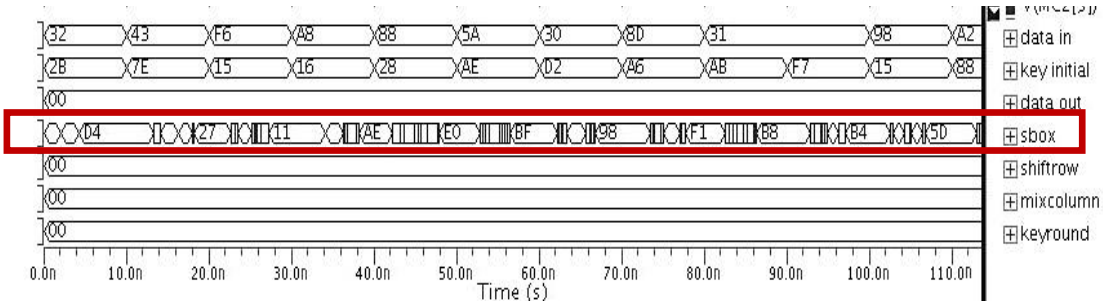


Figure 7.5 *Waveform simulation for SubByte transformation*

Waveform simulation for ShiftRows transformation is shown in Figure 7.6, where the 8-bit output is shifted out from this sub-block after 12 clock cycles. Data output shifted is:

After ShiftRows: D4BF5D30E0B452AEB84111F11E2798E5

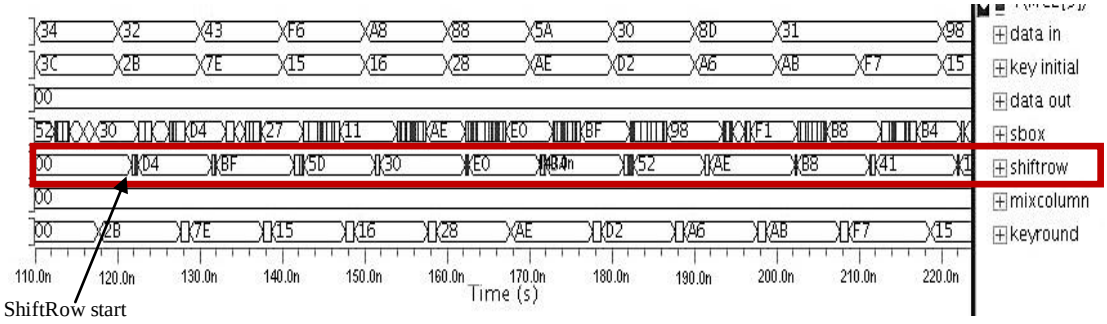


Figure 7.6 *Waveform simulation for ShiftRow transformation*

MixColumn transformation output is produced after 28 clock cycles, simultaneously with key round output of the Key scheduling unit. Figure 7.7 represents the waveform simulation for MixColumn transformation with outputs of:
After MixColumns: 0466811E5E0CB199A48F8D37A2806264C

Also, the first key round that was generated from the key scheduling unit:
A0FAFE1788542CB123A339392A6C7605

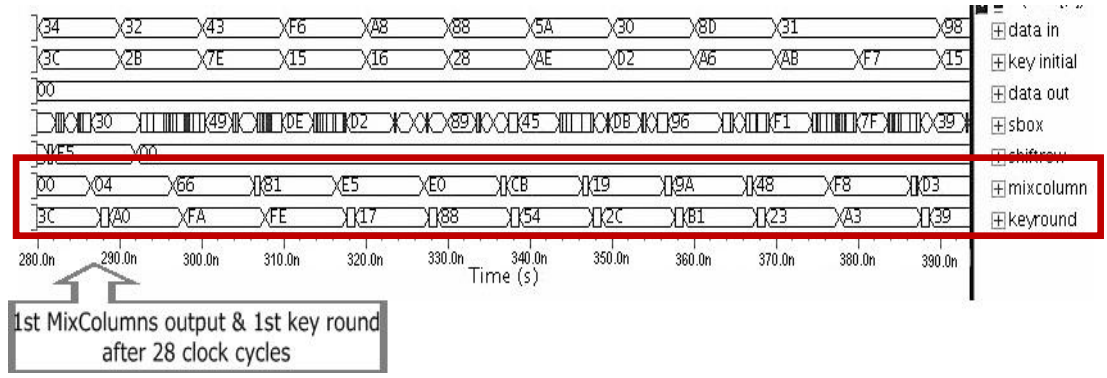


Figure 7.7 *Waveform simulation for MixColumns and key round transformation*

The key scheduling unit is used to generate the key round for 10 rounds of the AES system. The cipher key used (128-bit) in this simulation:
2B7E151628AED2A6ABF7158809CF4F3C

The following is the last expanded key round for encryption (128-bit):
D014F9A8C9EE2589E13F0CC8B6630CA6

The cipher out is generated at 264 clock cycles, and produced the data output of:
3925841D02DC09FBDC118597196A0B32

Figure 7.8 *Waveform simulation for last key round and data output of AES encryption*

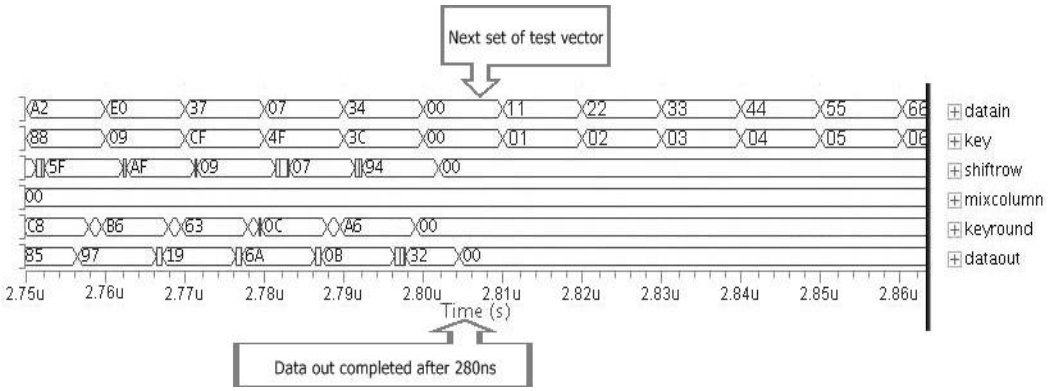


Figure 7.9 *Next set of test vectors for AES simulation after encryption process was completed*

7.5 Measurement testing setup for AES chip

An electrical test is the process of testing each contact on a component to ensure it matches the component electrical properties such as the functionality of the device. It can detect defects caused by Electrostatic Discharge (ESD) damage, internal physical damage caused by heat damage, and electrical overstress. For this AES chip, an unpowered electrical test was completed by switching off the component, without adding any stimulus to the electrical component. When the result is satisfactory, the functional test will be set up.

The physical testing of the fabricated chip was performed to verify the functionality of the AES crypto-processor chip, and make sure the results were identical to the simulation results in term of functionality. Power analysis and propagation delay were not performed to the fabricated chip because it required additional hardware with advanced equipment which is not available in the university. The functional verification results during power-on are presented. The additional circuitry and testing system are interfacing with the AES chip in order to perform a test. After a connection is

established, the values are updated every second. The test circuit setup used for field testing is shown in the figure below (Figure 7.11). Functional verification is performed with 8-bit input of the data block and initial cipher key to produce 8-bit output of data. The same set of test vectors from NIST as the section 7.4.1 were used to verify the encryption and decryption process.

The operation of this chip is controlled by four logic-level inputs; clock, reset, enable and enc_dec. The encryption and decryption process depends upon the signal enc_dec, when enc is selected, it signals the datapath that the encryption needs to be performed; when dec is selected, the decryption will be performed. Signal reset is a global signal used to reset the chip. En signal is an enable signal that enables the operation of the chip when it is selected. The chip was also tested for verification 8-bit output of sub-block MixColumn and key round from the Key Scheduling unit.

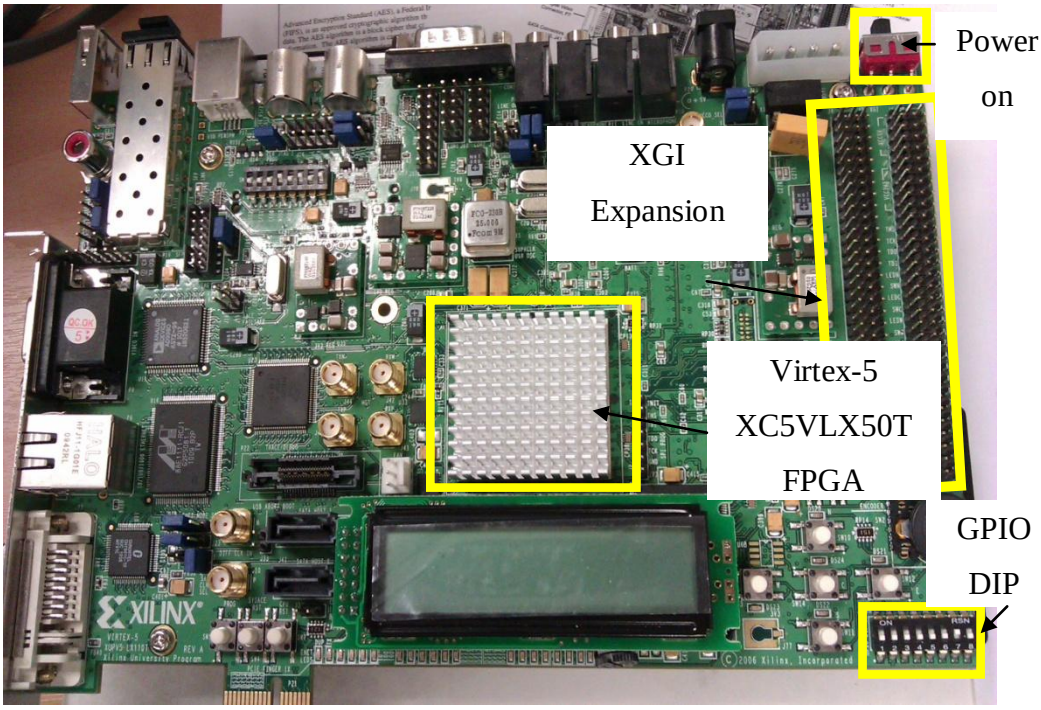


Figure 7.10 *ML505 FPGA board for test vectors generator*

A testing circuit has been developed by interfacing the AES chip to the Field Programmable Gate Array (FPGA). Figure 7.10 shows the ML505 evaluation board that has been used in the testing. The actual test results obtained with test vectors generated and output displayed using the ML505 Xilinx FPGA Development board using the verification methodologies is described as follows:

a. The FPGA Test System was developed and implemented using a FPGA ML505 board that consists of a Test Pattern Generator and Output Buffer Comparator. The Test Pattern Generator is used to generate a selected 20 set of test vectors from FIPS 197 for data block and key initial described in VHDL programming and carried out using Xilinx software ISE 9.1. This test vector was used to debug and verify the AES chip by interfacing the chip with the FPGA board. An Output Buffer Comparator was used to display the results generated from an AES chip in the PC, using ChipScope software and compare the results with the reference output. If the results are not the same, it displays '1' at the comparative output.

b. A control input bit stream, including a set of test vectors, clock signal 100MHz and reset signal from FPGA board are interfaced to the AES chip by an XGI Expansion Interface, to provide the input for the AES chip. The hardware testing setup is illustrated in Figure 7.11. Data inputs are sent from the FPGA board to the chip to process the encryption or decryption algorithm. Enc_dec and enable signals are connected directly from external switches to the AES chip. The AES chip then transmits the data results to the FPGA board to display the output on the computer.

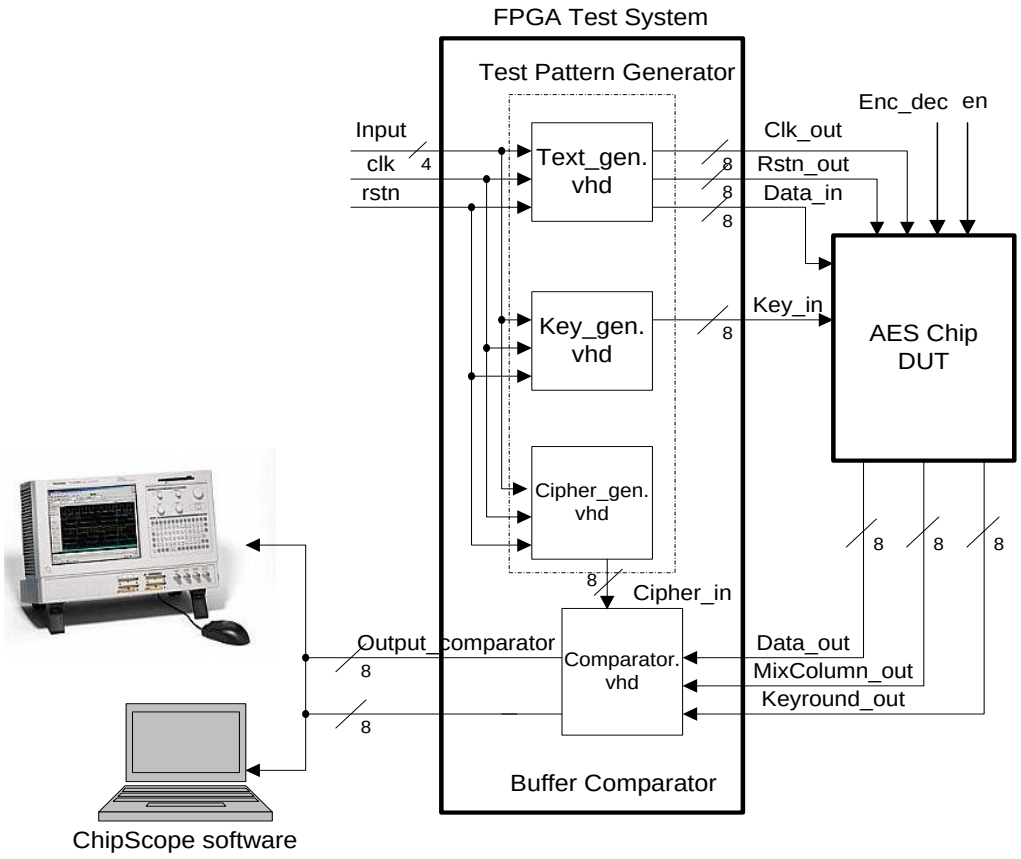


Figure 7.11 *Hardware connection setup for AES testing*

c. A printed circuit board for the AES chip is made and connected together to the FPGA board. As the output voltage from FPGA is 3.3V, and the AES chip only operates at in 0.8V, FPGA voltage output is translated to 0.8V using a 4-bit dual supply translating transceiver, 74AVC4T245. It is also used to translate from 0.8V to 3.3V to display the AES output in FPGA. AES output is connected back to FPGA logic test system to display signals during measurement and compare with expected output using Xilinx ChipScope software.

d. The functional test included the following steps:

i) Test input signal of Reset and Enable

   a. Reset the chip, this clears the registers and counters that affect the output of the system.

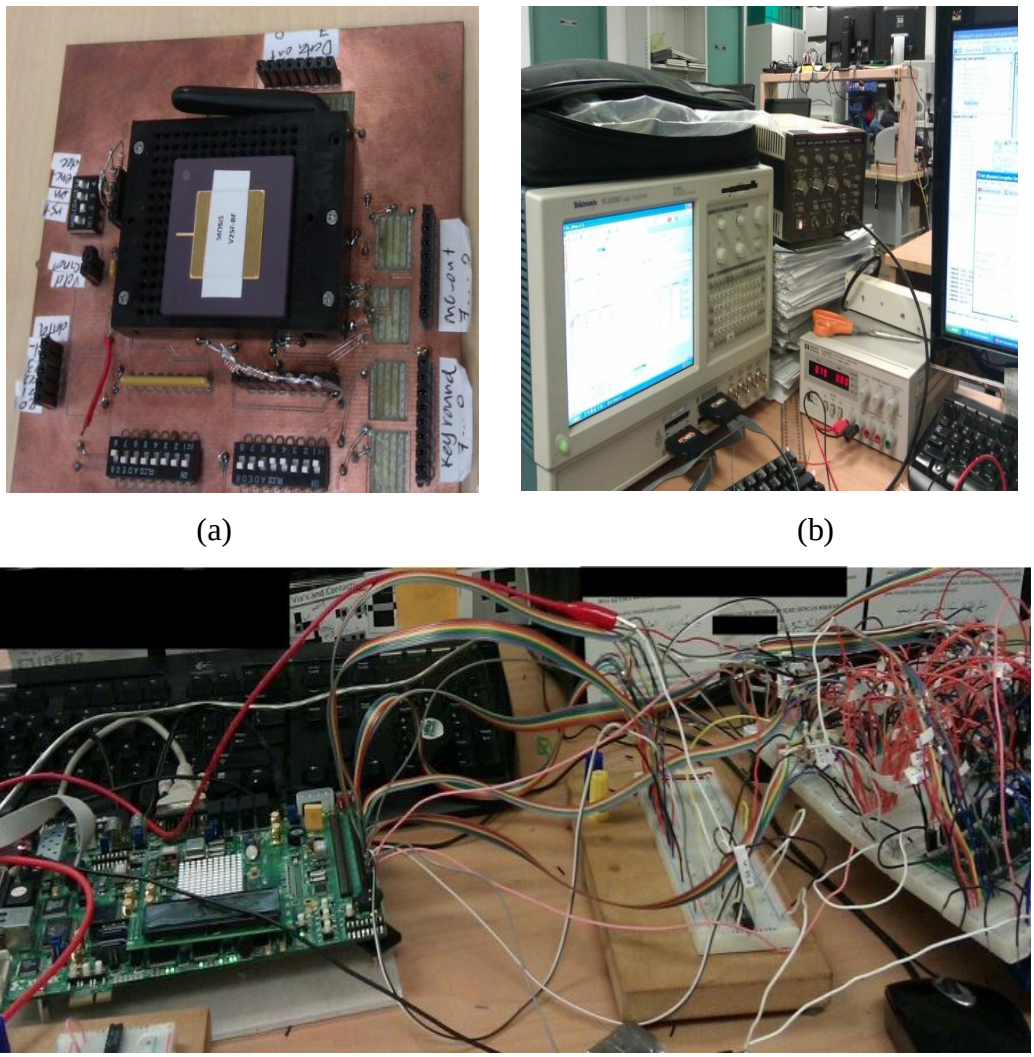
   b. Disable the chip: causes no operation of the system.

ii) Test Counter Mod 10 in the system to produce the output of the counter, (q_3, q_2, q_1, q_0) and Sel_data requiring 100MHz clock; set the enable and reset=0.

iii) Test the Mix Column unit, Key Scheduling unit and AES system.

a. Set the enable, reset=0 and set the Enc_Dec =1 for encryption
b. Give 100 MHz clock and a sequence 8-bit data in 16 cycles clock for Data_in and Key_in. The sequence is repeated for 20 sets of test vectors for the encryption and decryption process in the AES system.

e. Logic analyzer Textronix TLA5202 will also used to display the clock, the inputs and outputs of the AES chip from the FPGA to get the different view of the AES output. The oscilloscope is used for power measurement of the AES chip voltage.



(a) (b)

(c)

Figure 7.12 *Circuit testing for AES (a) PCB with the test socket for AES and 4-bit dual supply translating transceiver, (b) Logic analyzer to display the output, and (c) Circuit setup with FPGA interface and AES chip*

7.6 Experimental Results

The first fabricated AES chip was tested, and results showed that there was a functional problem and it was not working properly, or generating the output as expected. Testing was done to debug the problem, including all the outputs for each of the sub-blocks of the AES system. From the testing observation, counter mod-10 that generates the cont_(q0-q3) and Sel_data signal give a zero output at all times, and this influences the overall output of the AES system. This signal affects multiplexer selection for the internal or external data being processed and also some controller input for sub-blocks of the AES system. Figure 7.12 shows the connection of Sel_data in the AES circuit. Further tests were done, and it was detected that counter mod-10 was not working, because the clock of the counter, produced from the MSB output (q44) of counter mod-28, was not generated properly. This output, q44 also controls another clock of sub-block in the Key Scheduling unit and control output of ShiftRows/ InvShiftRows. As a result, all the sub-blocks of AES had error regarding this fault, leading to an unsuccessful chip.

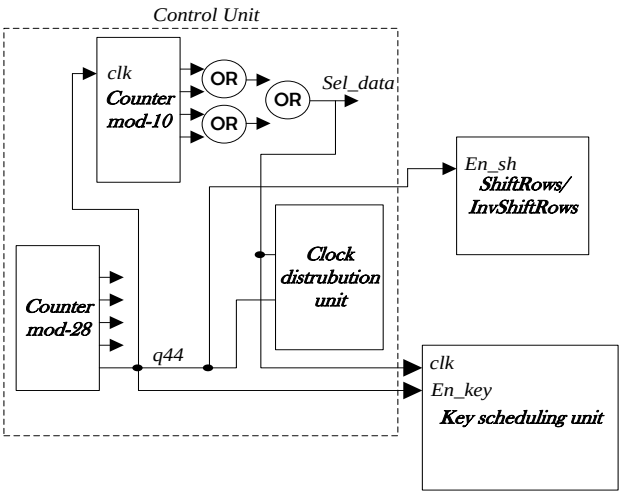


Figure 7.12 *Sel_data and q44 connection in the AES circuit*

The mask layout was re-checked to detect the problem, and it showed that the substrate of NFET transistor was not connected properly and this lead to a grounding problem. Based on IBM PDK documentation, the SXCUT layer in the layout is used to break the substrate region for multiple substrates with different names. The AES layout has used a SXCUT layer without a label on it, which generates the isolated substrate region and

prevents detection of soft-connect violation in the layout. Due to the imperfect nature of the manufacturing process, defects may be introduced during fabrication, resulting in chips that could potentially malfunction. The new layout has been re-drawn without errors and sent back to the foundry for IC fabrication.

The re-fabricated chip was tested and evaluated with the same testing setup using the ML505 Xilinx FPGA Development board. The results obtained show that the AES chip was working properly as expected and was fully operational at the clock rate of 100MHz. Figure 7.13 and Figure 7.14 show the output displayed using ChipScope software on the computer. Based on the results, it demonstrates that the fabricated AES chip meets the prediction of the simulation in term of the functionality with power dissipation of 5.2 µW/ MHz.

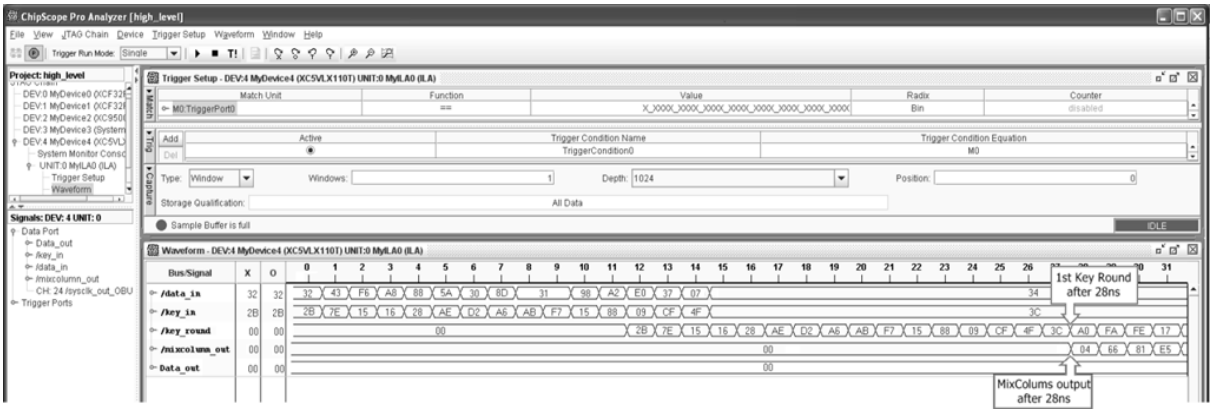


Figure 7.13 *Outputs for MixColumns and first key round generated*

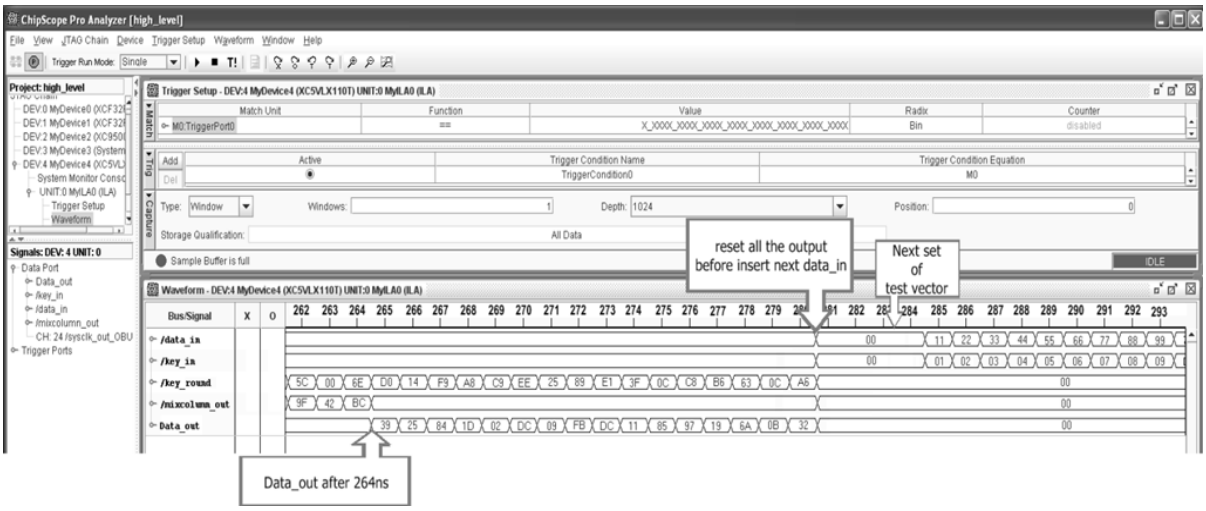


Figure 7.14 *Data output generated after 264 clock cycles and the next test vector input*

7.7 Conclusions

This chapter described a full-custom compact AES crypto-processor implemented in a 130nm CMOS process. The AES chip was evaluated and the performance compared to different 8-bit AES architectures. Through the simulation, the AES chip offers a throughput of 0.05 Gbps with a low gate count of 3152 gate equivalent. Moreover, according to the results, the AES chip offers a small silicon area of 1.28square mm and estimated power dissipation 5.2 µW/ MHz. The actual experimental testing verified the functionality of the working fabricated chip.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

8.1 Introduction

This chapter summarises the work and research outcomes of this thesis. Conclusions are drawn from the findings, and the key contributions to knowledge are identified from previous chapters. New recommendations and significant opportunities for future research work are identified and highlighted, to improve the current system.

8.2 Thesis Summary

In this thesis, the complete AES architecture is designed and developed in ASIC implementation. The AES crypto-processor is implemented to support the encryption and decryption algorithms, with a data block size of 128-bit, and a cipher key of 128-bit. This architecture includes the new architecture of SubBytes/ InvSubBytes, ShiftRows/ InvShiftRows, MixColumns/ InvMixColumns and key scheduling, for making it compact and efficient for an 8-bit datapath width in VLSI implementation. Furthermore, the proposed architecture was optimized for high-performance, low-area and low-power hardware implementation. The following paragraph summarises the achievements of the research, described in each of the chapters in this thesis.

In Chapter 3, two new low-area and low-power 2-input XOR gates were discussed. Moreover, the work presented in [76] and [75] was evaluated and compared with peer designs implemented in a 65nm and 130nm CMOS process. Based on the simulation results, power consumption, propagation delay, and area were analyzed, and the proposed XOR gate in [75] resulted in low-power and high-performance, making it

suitable to use in the AES crypto-processor architectures, to fulfil the constraints of the requirement.

In Chapter 4, we proposed the new compact combinational logic low-power, low-complexity design of the S-box/ InvS-box for AES architecture. The circuit of the S-box/InvS-box was minimised, by using the most optimised composite field formulations in Galois field and logic minimisation approaches, for the least hardware complexities. The new proposed XOR gate in Chapter 3 was used in the circuit to achieve a low-power dissipation and low-area design of the S-box/ InvS-box. Exhaustive simulation for the input transitions was conducted between the proposed S-box/InvS-box and other existing designs. Through the simulations, it was shown that our proposed design achieved a compact core circuit: it only has an area approximately 39.44 square µm, while the hardware cost of the S-box/ InvS-box is about 147 logic gates, equivalent to 876 transistors. It has a critical path propagation delay of 3.235ns and power consumption of 7.33µW at a frequency of 100MHz.

In Chapter 5, the new fault detection scheme for the AES S-box/ InvS-box, using a parity prediction technique, was presented. The predicted block was divided into seven blocks, to compare between the actual parity output and the predicted parity output results. The predicted blocks were developed with formulations compatible with the new S-box/ InvS-box, as described in Chapter 4. The proposed fault detection has the small area, critical path delay, and power consumption.

The following chapter described the new architecture of the comprehensive AES crypto-processor, for an 8-bit datapath. All the new sub-component architectures in the AES that were proposed were suitable for an 8-bit width datapath. Each of the new sub-component architectures was simulated, and the results were compared to other previous designs. Combinations of the new sub-components in the AES crypto-processor resulted in a low-power, low-area and high-performance overall AES design.

Finally, in Chapter 7 the AES implementation was described, as well as verification of the design, and experimental testing results for the fabricated AES chip. An exhaustive simulation on the AES performance was done, and the results showed that the AES chip reached a throughput of 0.05 Gbps, with a low gate count, small silicon area, and

low power consumption when compared to other existing AES designs. The experimental testing showed that the AES was working functionally, as expected.

In conclusion, the research outcomes and contributions of knowledge to this thesis are as follows:

1 The literature review, incorporating various methods of designing and constructing a low-power, low-area AES system, were explored. As a result, the new low-power, small-area 2-input Exclusive OR (XOR) gate was proposed as one of the techniques to optimise the layout area and power consumption of the AES system. By using the proposed XOR gate, silicon architecture constraints for each sub-module of AES was resolved.

2 Designed and developed a fully-integrated, compact, 8-bit stream cipher core of an application-specific integrated circuit AES crypto-processor. This design supports both encryption and decryption process with a 128-bit data block and a 128-bit key in a 130nm CMOS technology by using iterative resource sharing for both configurations.

3 By merging the sub component of the typical multiplicative inverse in composite field, a new improved and efficient S-box/ InvS-box architecture was developed in order to optimize and reduce the hardware complexity of the circuit. A surface level of research focussed on the new error detection method for the S-box of AES was proposed using parity code error detection to detect any fault attack.

4 To reduce the total area of AES crypto-processor, a new optimized MixColums/ InvMixColumns architecture that suitable for 8-bit and 32-bit datapath is presented using the decomposition byte level resource sharing method, which involves a low gate count and short time delay.

5 Requiring only 448 gates and suitable for an 8-bit and 32-bit AES datapath, a new joint ShiftRow and InvShiftRow architecture using twelve 8-bit registers,

along with six 2-to-1, and one 4-to-1 multiplexer was proposed and contribute to design a compact AES crypto-processor.

6 As one of the area contributor in AES system, an improved on-the fly Key Scheduling architecture was developed, which is compatible with the AES 8-bit datapath and requires no memory structure to optimize the area of AES chip which generate a key expansion for encryption and decryption.

8.3 Recommendations and future work

Further research and design could be performed to improve the work of this thesis. Several improvements could be done, such as:

1 Enhance the security of the AES against modern attacks, such as side-channel attacks, using masking or other modern techniques to protect the AES system.

2 The fault detection schemes for the S-box/ InvS-box could be fabricated and actual testing performed by inserting the actual error injections, in order to verify the functionality of the proposed scheme. Another further work on fault detection for the whole AES system could be designed and evaluated to reach AES chip architecture free from fault errors.

3 As a future work, one could design a compact, AES 8-bit datapath that supports all key sizes: 128-bit, 192-bit and 256-bit for the encryption and decryption algorithm, by modifying the key scheduling unit architecture.

4 Finally, the proposed AES crypto-processor could be designed using the latest nanometer CMOS technology, for example 22nm CMOS technology, and make use of the low power flavours device for different power requirements. This work could further reduce the power consumption, silicon area and enhance the overall performance of the AES crypto-processor.

BIBLIOGRAPHY

[1] T. Hardjono and L. R. Dondeti, "Security In Wireless LANS And MANS", *(Artech House Computer Security)*: Artech House, Inc., 2005.

[2] P. K. N. RuangChaijatupon, "Encryption and Power Consumption in Wireless LANs-N," in *The Third IEEE Workshop on Wireless LANs*, Massachusetts, 2001.

[3] J. Daemen and V. Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard", *Springer-Verlag New York*, Inc., 2002.

[4] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," in *Advances in Cryptology — ASIACRYPT 2001*. vol. 2248, C. Boyd, Ed., ed: Springer Berlin Heidelberg, pp. 239-254, 2001.

[5] L. Liang, H. Jun, Z. Xiaoyang, and Z. Jia, "A full-custom design of AES SubByte module with signal independent power consumption," in *IEEE International Symposium on Circuits and Systems, 2008 (ISCAS 2008)*, pp. 3302-3305, 2008

[6] K. Gaj and P. Chodowiec, "FPGA and ASIC Implementations of AES," in *Cryptographic Engineering*, Ç. Koç, Ed., ed: Springer US, pp. 235-294, 2009.

[7] N. Ferguson and B. Schneier, "Practical Cryptography", *John Wiley& Sons*, 2003.

[8] L. Elbaz and H. Bar-El, "Strength Assessment of Encryption Algorithms", *EETimes* [Technical paper], 2003.

[9] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand," in *IEE Proceedings Information Security*, vol. 152, pp. 13-20, 2005.

[10] N. J. E. Hess, B. Meyer, T. Schütze, "Information leakage attacks against smart card implementations of cryptographic algorithms and countermeasures," in *Proceedings of EUROSMART-Security-Conference 2000*, pp. 53-64, 2000.

[11] S. Tillich, M. Feldhofer, T. Popp, and J. Groszschaedl, "Area, Delay, and Power Characteristics of Standard-Cell Implementations of the AES S-Box," *Journal of Signal Processing Systems*, vol. 50, pp. 251-261, 2008.

[12] S. Morioka and A. Satoh, "An Optimized S-Box Circuit Architecture for Low Power AES Design," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 2523, pp. 172-186, 2003.

[13] R. R. Rachh and P. V. Ananda Mohan, "Implementation of AES S-Boxes using combinational logic," in *IEEE International Symposium on Circuits and Systems, 2008 (ISCAS 2008)*, pp. 3294-3297, 2008.

[14] C. Ning and Y. Zhiyuan, "High-performance designs of AES transformations," in *IEEE International Symposium on Circuits and Systems, 2009 (ISCAS 2009)*, pp. 2906-2909, 2009.

[15] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation," *IEEE Transactions on Computers*, vol. C-35, pp. 677-691, 1986.

[16] S. Morioka and A. Satoh, "A 10 Gbps full-AES crypto design with a twisted-BDD S-Box architecture," in *IEEE International Conference on Computer Design: VLSI in Computers and Processors 2002*, pp. 98-103, 2002.

[17] G. Bertoni, M. Macchetti, L. Negri, and P. Fragneto, "Power-efficient ASIC synthesis of cryptographic Sboxes," in *Proceedings of the 14th ACM Great Lakes symposium on VLSI*, pp. 277-281, 2004.

[18] N. Ahmad, R. Hasan, and W. M. Jubadi, "Design of AES S-box using combinational logic optimization," in *IEEE Symposium on Industrial Electronics & Applications (ISIEA) 2010*, pp. 696-699, 2010.

[19] N. Chen and Z. Yan, "High-performance designs of AES transformations," in *IEEE International Symposium on Circuits and Systems 2009 (ISCAS 2009)*, pp. 2906-2909, 2009.

[20] Z. Liu, Y. Zeng, X. Zou, Y. Han, and Y. Chen, "A High-Security and Low-Power AES S-Box Full-Custom Design for Wireless Sensor Network," in *International Conference on Wireless Communications, Networking and Mobile Computing, 2007 (WiCom 2007)*, pp. 2499-2502, 2007.

[21] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic," in *Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 2162, pp. 171-184 2001.

[22] D. Canright, "A very compact s-box for AES," in *Proceedings of the 7th international conference on Cryptographic hardware and embedded systems*, pp. 441-455, 2005.

[23] F. Burns, J. Murphy, A. Koelmans, and A. Yakovlev, "Efficient advanced encryption standard implementation using lookup and normal basis," *Computers & Digital Techniques, IET*, vol. 3, pp. 270-280, 2009.

[24] R. Liu and K. K. Parhi, "Fast composite field S-box architectures for advanced encryption standard," in *Proceedings of the 18th ACM Great Lakes symposium on VLSI*, pp. 65-70, 2008.

[25] R. Vincent, "Efficient Implementation of the Rijndael S-box," Katholieke Universiteit Leuven, Dept. ESAT. Belgium.

[26] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC Implementation of the AES SBoxes," in *Proceedings of the The Cryptographer's Track at the RSA Conference on Topics in Cryptology*, pp.67-78, 2002.

[27] N. Mentens, L. Batina, B. Preneel, and I. Verbauwhede, "A systematic evaluation of compact hardware implementations for the rijndael s-box," in *Proceedings of the 2005 international conference on Topics in Cryptology*, pp. 323-333, 2005.

[28] Y.-h. Zeng, X.-c. Zou, Z.-l. Liu, and J.-m. Lei, "Low-power clock-less hardware implementation of the rijndael S-box for wireless sensor networks," *The Journal of China Universities of Posts and Telecommunications*, vol. 14, pp. 104-109, 2007.

[29] P. Chodowiec and K. Gaj, "Very Compact FPGA Implementation of the AES Algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2003*. vol. 2779, pp. 319-333, 2003.

[30] L. Hua and L. Jianzhou, "A New Compact Architecture for AES with Optimized ShiftRows Operation," in *IEEE International Symposium on Circuits and Systems (ISCAS 2007)*, pp. 1851-1854, 2007.

[31] S. Mangard, M. Aigner, and S. Dominikus, "A highly regular and scalable AES hardware architecture," *IEEE Transactions on Computers*, vol. 52, pp. 483-491, 2003.

[32] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core," in *9th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools (DSD 2006)*, pp. 577-583, 2006.

[33] Z. Xinmiao and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 12, pp. 957-967, 2004.

[34] V. Fischer, M. Drutarovsky, P. Chodowiec, and F. Gramain, "InvMixColumn decomposition and multilevel resource sharing in AES implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* , vol. 13, pp. 989-992, 2005.

[35] L. Hua and Z. Friggstad, "An efficient architecture for the AES mix columns operation," in *IEEE International Symposium on Circuits and Systems 2005, (ISCAS 2005),* Vol. 5, pp. 4637-4640 2005.

[36] S. F. Hsiao and M. C. Chen, "Efficient substructure sharing methods for optimising the inner-product operations in Rijndael advanced encryption standard," in *IEE Proceedings Computers and Digital Techniques,* vol. 152, pp. 653-665, 2005.

[37] L. Chih-Chung and T. Shau-Yin, "Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter," in *Proceedings of The IEEE International Conference on Application-Specific Systems, Architectures and Processors,* pp. 277-285, 2002.

[38] M. Feldhofer, S. Dominikus, and J. Wolkerstorfer, "Strong Authentication for RFID Systems Using the AES Algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2004.* vol. 3156, pp. 357-370, 2004.

[39] M. Dworkin and and P. D. Gallagher, "NIST Recommendation for Block Cipher Modes of Operation, Methods and Techniques," 2001.

[40] C. Nalini, Nagaraj, P. V. Anandmohan, D. V. Poornaiah, and V. D. Kulkarni, "An FPGA Based Performance Analysis of Pipelining and Unrolling of AES Algorithm," in *International Conference on, Advanced Computing and Communications 2006 (ADCOM 2006),* pp. 477-482, 2006.

[41] Z. L. X. Guo, J. Xing, W. Fan and X. Zou, "Optimized AES Crypto Design for Wireless Sensor Networks with a Balanced S-box Architecture," in *Proceedings of International Conference on Informatics and Control Technologies (ICT2006),* pp. 203-208, 2006.

[42] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," in *Proceedings of the 28th European Solid-State Circuits Conference 2002 (ESSCIRC 2002),* pp. 403-406, 2002.

[43] T. Good and M. Benaissa, "692-nW advanced encryption standard (AES) on a 0.13-µm CMOS," *IEEE Transaction on Very Large Scale Integration System,* vol. 18, pp. 1753-1757, 2010.

[44] D. Kamel, F.-X. Standaert, and D. Flandre, "Scaling trends of the AES S-box low power consumption in 130 and 65 nm CMOS technology nodes," in *IEEE International Symposium on Circuits and Systems 2009, (ISCAS 2009),* pp. 1385-1388, 2009.

[45] P. Gargini, "The International Technology Roadmap for Semiconductors (ITRS): "Past, present and future"," in *GaAs IC Symposium, 2000. 22nd Annual,* pp. 3-5, 2000.

[46] C. Qingfu and L. Shuguo, "A high-throughput cost-effective ASIC implementation of the AES Algorithm," in *IEEE 8th International Conference on ASIC 2009 (ASICON '09,* pp. 805-808, 2009.

[47] S. Chih-Pin, L. Tsung-Fu, H. Chih-Tsiun, and W. Cheng-Wen, "A high-throughput low-cost AES processor," *Communications Magazine, IEEE,* vol. 41, pp. 86-91, 2003.

[48] L. Lan and D. Luke, "Implementation of AES as a CMOS core," in *IEEE Canadian Conference on Electrical and Computer Engineering 2003 (CCECE 2003)*, 2003, vol.1, pp. 53-56, 2003.

[49] M. Mayhew and R. Muresan, "Low-power AES coprocessor in 0.18um CMOS technology for secure microsystems," in *Microsystems and Nanoelectronics Research Conference 2009 (MNRC 2009)*, pp. 140-143, 2009.

[50] T. Ichikawa, T. Kasuya and M. Matsui, "Hardware evaluation of the AES finalists," presented at the In The Third Advanced Encryption Standard Candidate Conference, pp. 279--285, 2000.

[51] H. Kuo, I. Verbauwhede, and P. Schaumont, "A 2.29 Gbits/sec, 56 mW non-pipelined Rijndael AES encryption IC in a 1.8 V, 0.18 μm CMOS technology," in *Proceedings of the IEEE Custom Integrated Circuits Conference 2002*. pp. 147-150, 2002.

[52] N. Pramstaller, S. Mangard, S. Dominikus, and J. Wolkerstorfer, "Efficient AES Implementations on ASICs and FPGAs," in *Advanced Encryption Standard – AES*. vol. 3373, pp. 98-112, 2005.

[53] L. Shin-Yi and H. Chih-Tsun, "A High-Throughput Low-Power AES Cipher for Network Applications," in *Design Automation Conference, 2007. ASP-DAC '07. Asia and South Pacific*, pp. 595-600, 2007.

[54] P. Saravanan, N. Renuka Devi, G. Swathi and P. Kalpana, "A High-Throughput ASIC implementation of Configurable Advanced Encryption Standard (AES) Processor," *IJCA Special Issue on Network Security and Cryptography*, vol. NSC, pp. 1-6, 2011.

[55] T. Good and M. Benaissa, "AES as stream cipher on a small FPGA," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, 2006, p. 4 pp.

[56] S. M. Farhan, S. A. Khan, and H. Jamal, "Mapping of high-bit algorithm to low-bit for optimized hardware implementation," in *Proceedings of The 16th International Conference on Microelectronics 2004 (ICM 2004)*, pp. 148-151, 2004.

[57] K. V. Dalmisli and B. Ors, "Design of new tiny circuits for AES encryption algorithm," in *3rd International Conference on Signals, Circuits and Systems (SCS)*, pp. 1-5, 2009.

[58] F. Haghighizadeh, H. Attarzadeh, and M. Sharifkhani, "A Compact 8-Bit AES Crypto-processor," in *Second International Conference on Computer and Network Technology (ICCNT)*, pp. 71-75, 2010.

[59] C. Hocquet, D. Kamel, F. Regazzoni, J.-D. Legat, D. Flandre and D. Bol, "Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-low-voltage 65 nm AES coprocessor for passive RFID tags," *Journal of Cryptographic Engineering*, vol. 1, pp. 79-86, 2011.

[60] B. Hung Tien, W. Yuke, and J. Yingtao, "Design and analysis of low-power 10-transistor full adders using novel XOR-XNOR gates," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, pp. 25-30, 2002.

[61] A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, pp. 20-29, 2002.

[62] D. Radhakrishnan, "Low-voltage low-power CMOS full adder," in *IEE Proceedings Circuits, Devices and Systems*, vol. 148, pp. 19-24, 2001.

[63] L. Hanho and G. E. Sobelman, "New XOR/XNOR and full adder circuits for low voltage, low power applications," *Microelectronics Journal*, vol. 29, pp. 509-517, 1998.
[64] L. Hanho and G. E. Sobelman, "New low-voltage circuits for XOR and XNOR," in *IEEE Proceedings Southeastcon '97. Engineering new New Century*, 1997, pp. 225-229.
[65] W. Jyh-Ming, F. Sung-Chuan, and F. Wu-Shiung, "New efficient designs for XOR and XNOR functions on the transistor level," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 780-786, 1994.
[66] B. Hung Tien, A. K. Al-Sheraidah, and W. Yuke, "New 4-transistor XOR and XNOR designs," in *Proceedings of the Second IEEE Asia Pacific Conference on ASICs, 2000 (AP-ASIC 2000)*, pp. 25-28, 2000.
[67] *Principles CMOS VLSI Design*: Pearson Education India.
[68] W. Dan, Y. Maofeng, C. Wu, G. Xuguang, Z. Zhangming, and Y. Yintang, "Novel low power full adder cells in 180nm CMOS technology," in *4th IEEE Conference on Industrial Electronics and Applications 2009 (ICIEA 2009)* pp. 430-433, 2009.
[69] A. R. A. B. Chowdhury, H. Saha, "A high Speed 8 Transistor Full Adder Design using Novel 3 Transistor XOR Gates," *International Journal of Electronics, Circuits and Systems*, vol. 2, pp. 217-223, 2008.
[70] M. N. a. M. M. M.H. Ghadiry, "A New Full Swing Full Adder Based on a New Logic Approach," *World Applied Sciences Journal* vol. 11, pp. 808-812, 2010.
[71] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*: Tata McGraw-Hill, 2003.
[72] L. Kun-Jin and W. Cheng-Wen, "A low-cost realization of multiple-input exclusive-OR gates," in *Proceedings of the Eighth Annual IEEE International ASIC Conference and Exhibit 1995*, pp. 307-310, 1995.
[73] A. Morgenshtein, A. Fish, and A. Wagner, "Gate-diffusion input (GDI)-a novel power efficient method for digital circuits: a design methodology," in *Proceedings 14th Annual IEEE International ASIC/SOC Conference, 2001*, pp. 39-43, 2001..
[74] M. Elgamel, S. Goel, and M. Bayoumi, "Noise tolerant low voltage XOR-XNOR for fast arithmetic," in *Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pp. 285-288, 2003.
[75] N. Ahmad and R. Hasan, "Topology of 2 input subnanowatt XOR gate in 65nm CMOS technology," in *IEEE International Conference on Semiconductor Electronics 2012 (ICSE2012)*, 2012.
[76] N. Ahmad and R. Hasan, "Design of XOR gates in VLSI implementation," in *Electronic New Zealand Conference (ENZCON 2010)*, pp. 51-55, 2010.
[77] A. K. Nishad and R. Chandel, "Analysis of Low Power High Performance XOR Gate Using GDI Technique," in *International Conference on Computational Intelligence and Communication Networks*, pp. 187-191, 2011.
[78] M. Hosseinghadiry, "Two New Low Power High Performance Full Adders with Minimum Gates," *International Journal of Electrical and Computer Engineering*, vol. 4, pp. 671-678.
[79] N. Ahmad and S. M. Rezaul Hasan, "Low-power compact composite field AES S-Box/Inv S-Box design in 65nm CMOS using Novel XOR Gate," *Integration, the VLSI Journal*, Available online 10 July 2012, ISSN 0167-9260, http://dx.doi.org/10.1016/j.vlsi.2012.06.002, 2012

[80] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient and High-Performance Parallel Hardware Architectures for the AES-GCM," *IEEE Transactions on Computers,* vol. 61, pp. 1165-1178, 2012.

[81] J. Boyar and R. Peralta, "A New Combinational Logic Minimization Technique with Applications to Cryptology," in *Experimental Algorithms*. vol. 6049, pp. 178-189, 2010.

[82] Y. Chih-Hsu, "Simple Error Detection Methods for Hardware Implementation of Advanced Encryption Standard," *IEEE Transactions on Computers,* vol. 55, pp. 720-731, 2006.

[83] R. Karri, W. Kaijie, P. Mishra, and K. Yongkook, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 21, pp. 1509-1517, 2002.

[84] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *IEEE Transactions on Computers,* vol. 52, pp. 492-505, 2003.

[85] W. Kaijie, K. Ramesh, G. Kuznetsov, and M. Goessel, "Low cost concurrent error detection for the advanced encryption standard," in *Proceedings of International Test Conference 2004 (ITC 2004),* pp. 1242-1248, 2004.

[86] M. M. Kermani and A. Reyhani-Masoleh, "Parity-Based Fault Detection Architecture of S-box for Advanced Encryption Standard," in *21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 2006 (DFT '06),* pp. 572-580, 2006.

[87] K. Bousselam, G. Di Natale, M. Flottes, and B. Rouzeyre, "Evaluation of concurrent error detection techniques on the advanced encryption standard," in *2010 IEEE 16th International On-Line Testing Symposium (IOLTS),*pp. 223-228, 2010.

[88] G. Di Natale, M. L. Flottes, and B. Rouzeyre, "A Novel Parity Bit Scheme for SBox in AES Circuits," in *IEEE Design and Diagnostics of Electronic Circuits and Systems 2007 (DDECS '07)* pp. 1-5, 2007.

[89] A. Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-Performance Concurrent Error Detection Scheme for AES Hardware," in *Cryptographic Hardware and Embedded Systems – CHES 2008*. vol. 5154, pp. 100-112, 2008.

[90] 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, pp. 85-91, 2011.

[91] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A Structure-independent Approach for Fault Detection Hardware Implementations of the Advanced Encryption Standard," in *Workshop on Fault Diagnosis and Tolerance in Cryptography 2007 (FDTC 2007),* pp. 47-53, 2007.

[92] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A parity code based fault detection for an implementation of the Advanced Encryption Standard," in *Proceedings 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 2002 (DFT 2002)* pp. 51-59, 2002.

[93] N. Yu and H. M. Heys, "A compact ASIC implementation of the advanced encryption standard with concurrent error detection," in *Proceedings of the Fifth IASTED International Conference on Circuits, Signals and Systems*, pp. 152-159, 2007.

[94] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Fault Detection Structures of the S-boxes and the Inverse S-boxes for the Advanced Encryption Standard," *Journal of Electronic Testing*, vol. 25, pp. 225-245, 2009.

[95] S. A. Reddy and M. A. Kumar, "Efficient fault detection scheme for reliable AES architecture," in *2011 International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT)*, pp. 1004-1009, 2011.

[96] M. M. Kermani and A. Reyhani-Masoleh, "Parity Prediction of S-Box for AES," in *Canadian Conference on Electrical and Computer Engineering 2006 (CCECE '06)*, pp. 2357-2360, 2006.

[97] C. Nalini, P. V. Anandmohan, D. V. Poomaiah, and V. D. Kulkarni, "Compact Designs of SubBytes and MixColumn for AES," in *IEEE International Advance Computing Conference 2009 (IACC 2009)*, pp. 1241-1247, 2009.

[98] E. G. Ahmed, E. Shaaban, and M. Hashem, "Lightweight mix columns implementation for AES," in P*roceedings of the 9th WSEAS international conference on Applied informatics and communications*, pp. 253-258, 2009.

[99] C. Chi-Jeng, H. Chi-Wu, C. Kuo-Huang, C. Yi-Cheng, and H. Chung-Cheng, "High throughput 32-bit AES implementation in FPGA," in *IEEE Asia Pacific Conference on Circuits and Systems 2008 (APCCAS 2008)*, pp. 1806-1809, 2008.

[100] P.Jemima Anlet, M.J., "*Parity Based Fault Detection Approach for the Low Power S-Box and Inverse S-Box", International Journal of Computer Technology and Electronics Engineering*, vol. 2(2): p. 76-81, 2012.

[101] S. Choomchuay, S.Pongyupinpanich and S.Pathumvanh, "A Compact 32-bit Architecture for an AES System," *ECTI Transactions on Computer and Information Theory*, vol. 1, pp. 24-29, 2005.

APPENDICES

This appendix presents the VHDL programming for the FPGA Test System consists of a Test Pattern Generator and Output Buffer Comparator.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top_level_generator1 is
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          E : in  STD_LOGIC;
                        I: in STD_LOGIC_VECTOR (7 downto 0);
          text_out : out  STD_LOGIC_VECTOR (7 downto 0);
          key_out : out  STD_LOGIC_VECTOR (7 downto 0);

          clk_out : out  STD_LOGIC;
          rstn_out : out  STD_LOGIC;
                        Q: out STD_LOGIC_VECTOR (7 downto 0);
                         less : out  STD_LOGIC;
          equal  : out  STD_LOGIC;
          greater : out  STD_LOGIC);
end top_level_generator1;

architecture Behavioral of top_level_generator1 is

component top_clkdiv is
generic(DIVRATIO:integer:=4);
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          clkout_div64 : out  STD_LOGIC;
          clkout_div128 : out  STD_LOGIC
                        );
end component;

component count_I is
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          I_out : out  STD_LOGIC_VECTOR (3 downto 0));
end component;

component count_Din is
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          E : in  STD_LOGIC;
          D : out  STD_LOGIC_VECTOR (3 downto 0));
```

```
end component;

component top_gen is
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
          I : in  STD_LOGIC_VECTOR (3 downto 0);
          clk_tk : in  STD_LOGIC;
                        clk_c: in std_logic;
          rstn : in  STD_LOGIC;
          load : in  STD_LOGIC;
          text_out : out  STD_LOGIC_VECTOR (7 downto 0);
          key_out : out  STD_LOGIC_VECTOR (7 downto 0);
          cipher_out : out  STD_LOGIC_VECTOR (7 downto 0);

          rstn_out : out  STD_LOGIC);
end component;

component reg is
   Port ( I : in  STD_LOGIC_VECTOR (7 downto 0);
          clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          Q : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component comparator is
   Port ( cip_fpga : in  STD_LOGIC_VECTOR (7 downto 0);
          cip_aes : in  STD_LOGIC_VECTOR (7 downto 0);
          less : out  STD_LOGIC;
          equal  : out  STD_LOGIC;
          greater : out  STD_LOGIC);
end component;

signal clk_64: std_logic;
signal clk_128: std_logic;
signal clk_pk: std_logic;
signal clk_cout: std_logic;
signal notclk: std_logic;
signal ORI: std_logic;
signal ANDC: std_logic;
signal Iload: std_logic_vector(3 downto 0);
signal Dload:std_logic_vector(3 downto 0);
signal loadin: std_logic;
signal Lin: std_logic_vector(3 downto 0);
signal NORL:std_logic;
signal a3: std_logic;
signal a2: std_logic;
signal a1: std_logic;
signal a0: std_logic;
signal b3: std_logic;
signal b2: std_logic;
signal b1: std_logic;
```

```
signal b0: std_logic;
signal cip_out: std_logic_vector(7 downto 0);

begin
U1: top_clkdiv generic map (DIVRATIO=>64) port map (clk=>clk, rstn=>rstn, clkout_div64=>clk_64, clkout_div128=>clk_128);
U2: count_I port map (clk=>clk_64, rstn=>rstn, I_out=>Iload);
U3: count_Din port map (clk=>clk_64, rstn=>rstn, E=>E, D=>Dload);
U4: top_gen port map (Din_k=>Dload, I=>Iload, clk_tk=>clk_pk, clk_c=>clk_cout, rstn=>rstn, load=>loadin, text_out=>text_out, key_out=>key_out, cipher_out=>cip_out, rstn_out=>rstn_out);
U5: count_I port map (clk=>clk, rstn=>rstn, I_out=>Lin);
U6: reg port map (I=>I, clk=>clk, rstn=>rstn, Q=>Q);
U7: comparator port map (cip_fpga=>cip_out, cip_aes=>I, less=>less, equal=>equal, greater=>greater);

notclk<=not clk_64;
clk_pk<=clk and (not clk_64);
a3<=Iload (3);
a2<= Iload(2);
a1<= Iload(1);
a0<=Iload(0);
ORI<= a3 or a2 or a1 or a0;
ANDC<= ORI and clk;
clk_cout<= clk_128 and clk;
b3<=Lin(3);
b2<=Lin(2);
b1<=Lin(1);
b0<=Lin(0);
NORL<=(b3 or b2) nor (b1 or b0);
loadin<=clk and NORL;
clk_out<=clk;
end Behavioral;

----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity top_clkdiv is
generic(DIVRATIO:integer:=4);
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;

          clkout_div64 : out  STD_LOGIC;
          clkout_div128 : out  STD_LOGIC
                        );
end top_clkdiv;

architecture Behavioral of top_clkdiv is
```

```
signal clkout: std_logic;


COMPONENT clkdiv
                generic (DIVRATIO: integer:=4);
   PORT(
       clk : IN  std_logic;
       rstn : IN  std_logic;
       clkout : OUT  std_logic
      );
   END COMPONENT;
begin

clk1: clkdiv generic map ( DIVRATIO => 32)
port map ( clk => clk,
rstn => rstn,
clkout => clkout_div64 );

clk2: clkdiv generic map ( DIVRATIO => 64)
port map ( clk => clk,
rstn => rstn,
clkout => clkout_div128 );
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity count_I is
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          I_out : out  STD_LOGIC_VECTOR (3 downto 0));
end count_I;

architecture Behavioral of count_I is

begin
process (clk, rstn)
variable count: std_logic_vector (3 downto 0);

begin
if rstn='0' then
count:="0000";
elsif clk'event and clk='0' then
count:=count+1;
end if;
I_out<=count;
```

```
end process;
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity count_Din is
   Port ( clk : in  STD_LOGIC;
          rstn : in  STD_LOGIC;
          E : in  STD_LOGIC;
          D : out  STD_LOGIC_VECTOR (3 downto 0));
end count_Din;

architecture Behavioral of count_Din is

begin
process (clk, rstn)
variable count: std_logic_vector (3 downto 0);

begin
if rstn='0' then
count:="0000";
elsif clk'event and clk='0' then
if E='1' then
count:=count+4;
else
count:=count;
end if;
end if;
D<=count;
end process;
end Behavioral;
----------------------------------------------------------------------------------------------------

--------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;

entity top_gen is
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
          I : in  STD_LOGIC_VECTOR (3 downto 0);
          clk_tk : in  STD_LOGIC;
                        clk_c: in std_logic;
          rstn : in  STD_LOGIC;
```

```
        load : in  STD_LOGIC;
        text_out : out  STD_LOGIC_VECTOR (7 downto 0);
        key_out : out  STD_LOGIC_VECTOR (7 downto 0);
        cipher_out : out  STD_LOGIC_VECTOR (7 downto 0);

        rstn_out : out  STD_LOGIC);
end top_gen;

architecture Behavioral of top_gen is

component key_gen_top
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
        I : in  STD_LOGIC_VECTOR (3 downto 0);
        clk : in  STD_LOGIC;
                        rstn: in std_logic;
        load : in  STD_LOGIC;
        key_out : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component text_gen2
   Port (
        I : in  STD_LOGIC_VECTOR (3 downto 0);
        clk : in  STD_LOGIC;
        rstn : in  STD_LOGIC;
        load : in  STD_LOGIC;
        text_out : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

component cipher_generator
   Port (I : in  STD_LOGIC_VECTOR (3 downto 0);
              clk : in  STD_LOGIC;
        rstn : in  STD_LOGIC;
        c_out : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

begin
U1: key_gen_top port map (Din_k, I, clk_tk, rstn, load, key_out);
U2: text_gen2 port map (I, clk_tk, rstn, load, text_out);
U3: cipher_generator port map (I, clk_c, rstn, cipher_out);
rstn_out<=rstn;
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;

entity key_gen_top is
```

```
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
          I : in  STD_LOGIC_VECTOR (3 downto 0);
          clk : in  STD_LOGIC;
                        rstn: in std_logic;
          load : in  STD_LOGIC;
          key_out : out  STD_LOGIC_VECTOR (7 downto 0));
end key_gen_top;

architecture Behavioral of key_gen_top is

component mod5
   Port ( clk : in  STD_LOGIC;
                                rstn: in std_logic;
          q : out  STD_LOGIC);
end component;

component key_gen
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
                                I: in STD_LOGIC_VECTOR (3 downto 0);
          clk : in  STD_LOGIC;
                        rstn: in std_logic;
                        w:in std_logic;
                        load: in std_logic;
          key_out : out  STD_LOGIC_VECTOR (7 downto 0));
end component;

signal qw: std_logic;
begin
G1: mod5 port map (clk,rstn,qw);
G2: key_gen port map (Din_k, I, clk, rstn, qw, load, key_out);
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity mod5 is
   Port ( clk : in  STD_LOGIC;
          rstn: in std_logic;
          q : out  std_logic);
end mod5;

architecture Behavioral of mod5 is
signal dir: std_logic_vector(3 downto 0);
begin
process(clk, rstn)
begin
```

```
if rstn ='0' then
dir<="0000";
elsif (clk='0' and clk'event) then
dir<=dir + "0001";
end if;
end process;

process(dir)

begin
case dir is
when "0000"=> q <='1';
when "0001"=> q <='1';
when "0010"=> q <='1';
when "0011"=> q <='0';
when "0100"=> q <='1';
when "0101"=> q <='1';
when "0110"=> q <='1';
when "0111"=> q <='0';
when "1000"=> q <='1';
when "1001"=> q <='1';
when "1010"=> q <='1';
when "1011"=> q <='0';
when "1100"=> q <='1';
when "1101"=> q <='1';
when "1110"=> q <='1';
when "1111"=> q <='0';
when others => q <='0';
end case;
end process;
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;

entity key_gen is
   Port ( Din_k : in  STD_LOGIC_VECTOR (3 downto 0);
          I: in STD_LOGIC_VECTOR (3 downto 0);
          clk : in  STD_LOGIC;
                              rstn: in std_logic;
                              w:in std_logic;
                              load: in std_logic;
          key_out : out  STD_LOGIC_VECTOR (7 downto 0));
end key_gen;

architecture Behavioral of key_gen is
```

```
signal dirn: std_logic_vector (7 downto 0);

begin

process(clk, rstn,dirn)
begin
        if rstn='0' then
        dirn <="00000000";

elsif (clk='0' and clk'event)then
        if load='1' then
        dirn<= I & Din_k;
        else
if      w='1' then
                dirn<= dirn+1;
                else
                dirn<=dirn+2;
                end if;
        end if;
end if;
key_out<=dirn;
end process;
end Behavioral;
--------------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.numeric_std.all;

entity text_gen2 is
   Port (
        I : in  STD_LOGIC_VECTOR (3 downto 0);
        clk : in  STD_LOGIC;
        rstn : in  STD_LOGIC;
        load : in  STD_LOGIC;
        text_out : out  STD_LOGIC_VECTOR (7 downto 0));
end text_gen2;

architecture Behavioral of text_gen2 is
signal dirn: std_logic_vector (7 downto 0);

begin
process(clk, rstn,dirn)
begin
        if rstn='0' then
        dirn <="00000000";
elsif (clk='0' and clk'event)then
        if load='1' then
```

```
        dirn<= I & "0000";
        else
        dirn<= dirn + "00010001";
        end if;
end if;
text_out<=dirn;
end process;
end Behavioral;
----------------------------------------------------------------------------------------------------
--------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.numeric_std.all;
use work.cipher_lut.all;

entity cipher_generator is
   Port ( I : in  STD_LOGIC_VECTOR (3 downto 0);
        clk : in  STD_LOGIC;
        rstn : in  STD_LOGIC;
        c_out : out  STD_LOGIC_VECTOR (7 downto 0));
end cipher_generator;

architecture lut of cipher_generator is
type testvec_t is array (0 to 15) of integer range 0 to 255;
type ciph_arr_t is array (0 to 15) of testvec_t;

constant ctext: ciph_arr_t:=
((16#e9#, 16#8e#, 16#0c#, 16#d2#, 16#f6#, 16#d3#, 16#cc#, 16#90#, 16#a5#, 16#6d#,
16#16#, 16#6e#, 16#7c#, 16#32#, 16#d2#, 16#9f#),

(16#a4#, 16#59#, 16#91#, 16#31#, 16#ec#, 16#bd#, 16#88#, 16#1a#, 16#f7#, 16#7f#,
16#6d#, 16#48#, 16#05#, 16#e4#, 16#38#, 16#3b#),

(16#e3#, 16#3c#, 16#55#, 16#bd#, 16#8b#, 16#67#, 16#ea#, 16#a8#, 16#33#, 16#99#,
16#fa#, 16#82#, 16#06#, 16#41#, 16#f4#, 16#df#),

(16#e3#, 16#81#, 16#58#, 16#a5#, 16#cf#, 16#2c#, 16#e7#, 16#d8#, 16#92#, 16#08#,
16#fd#, 16#38#, 16#b3#, 16#f9#, 16#d0#, 16#c9#),

(16#85#, 16#69#, 16#4f#, 16#93#, 16#32#, 16#38#, 16#ba#, 16#83#, 16#35#, 16#df#,
16#69#, 16#26#, 16#c1#, 16#0d#, 16#20#, 16#4a#),

(16#fd#, 16#11#, 16#58#, 16#17#, 16#b2#, 16#08#, 16#ba#, 16#0c#, 16#36#, 16#1d#,
16#c4#, 16#34#, 16#18#, 16#fa#, 16#b2#, 16#34#),

(16#d5#, 16#dc#, 16#24#, 16#a5#, 16#c2#, 16#75#, 16#8a#, 16#f9#, 16#ab#, 16#85#,
16#a7#, 16#bf#, 16#0c#, 16#63#, 16#c7#, 16#7e#),
```

```
(16#5b#, 16#b6#, 16#7c#, 16#ef#, 16#07#, 16#2a#, 16#5c#, 16#3d#, 16#8e#, 16#44#,
16#ce#, 16#c0#, 16#f1#, 16#4a#, 16#dd#, 16#93#),

(16#08#, 16#4d#, 16#04#, 16#be#, 16#66#, 16#c3#, 16#fc#, 16#61#, 16#47#, 16#3d#,
16#69#, 16#d0#, 16#50#, 16#3b#, 16#98#, 16#f3#),

(16#85#, 16#a7#, 16#f3#, 16#d0#, 16#df#, 16#aa#, 16#59#, 16#41#, 16#57#, 16#4d#,
16#21#, 16#af#, 16#6f#, 16#e0#, 16#2b#, 16#e4#),

(16#e3#, 16#02#, 16#31#, 16#bf#, 16#a4#, 16#29#, 16#86#, 16#aa#, 16#32#, 16#9c#,
16#f9#, 16#4d#, 16#51#, 16#98#, 16#08#, 16#e5#),

(16#08#, 16#13#, 16#d7#, 16#89#, 16#20#, 16#8d#, 16#ac#, 16#cb#, 16#87#, 16#98#,
16#c9#, 16#fc#, 16#f5#, 16#60#, 16#52#, 16#a1#),

(16#63#, 16#e5#, 16#11#, 16#aa#, 16#44#, 16#70#, 16#28#, 16#f3#, 16#06#, 16#e7#,
16#fa#, 16#25#, 16#97#, 16#02#, 16#cd#, 16#b0#),

(16#02#, 16#1f#, 16#d0#, 16#ec#, 16#d6#, 16#e5#, 16#e5#, 16#bb#, 16#c3#, 16#13#,
16#2b#, 16#8e#, 16#d5#, 16#73#, 16#40#, 16#1c#),

(16#41#, 16#4c#, 16#62#, 16#24#, 16#75#, 16#af#, 16#fb#, 16#2a#, 16#fc#, 16#50#,
16#4d#, 16#12#, 16#5b#, 16#62#, 16#28#, 16#a0#),

(16#5a#, 16#64#, 16#34#, 16#09#, 16#74#, 16#7f#, 16#f6#, 16#b2#, 16#0d#, 16#ab#,
16#ec#, 16#dc#, 16#67#, 16#b7#, 16#22#, 16#8e#));

signal I_in:integer range 0 to 15:=0;
signal count: natural;
begin
counter:process(clk)
begin
if clk'event and clk='0' then
if rstn='0' then
count<=0;
else count<=count+1;
end if;
end if;
end process;

process (clk, rstn)
begin
if rstn ='0' then
        c_out<="00000000";
        elsif clk'event and clk='1' then
        for i in  15 downto 0 loop
        c_out<=std_logic_vector(to_unsigned(ctext (i)(count),8));
        end loop;
        end if;
```

```
        end process;
end lut;
--------------------------------------------------------------------------------------------------------
-------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comparator is
   Port ( cip_fpga : in  STD_LOGIC_VECTOR (7 downto 0);
          cip_aes : in  STD_LOGIC_VECTOR (7 downto 0);
          less : out  STD_LOGIC;
          equal  : out  STD_LOGIC;
          greater : out  STD_LOGIC);
end comparator;

architecture Behavioral of comparator is

begin
process(cip_fpga,cip_aes)
begin
if (cip_fpga> cip_aes ) then  --checking whether num1 is greater than num2
less <= '0';
equal <= '0';
greater <= '1';
elsif (cip_fpga < cip_aes) then    --checking whether num1 is less than num2
less <= '1';
equal <= '0';
greater <= '0';
else     --checking whether num1 is equal to num2
less <= '0';
equal <= '1';
greater <= '0';
end if;
end process;
end Behavioral;
```

LIST OF PUBLICATIONS

Journal Papers

I) Journal paper titled "A 0.8 V 0.23 nW 1.5 ns full-swing pass-transistor XOR gate in 130 nm CMOS, " in *Active and passive electronic components*, vol. 2013, Article ID 148518, 6 pages, Hindawi Publishing Corporation, 2013.

II) Journal paper titled "Efficient integrated AES crypto-processor architecture for 8-bit stream cipher," in *Electronic Letters*, IET, vol. 48, no. 23, pp. 1456-1457, 2012.

III) Journal paper titled "Low-power compact composite field AES S-Box/Inv S-Box design in 65nm CMOS using Novel XOR Gate," in *Integration, the VLSI Journal*, Elsevier publishers, Available online 10 July 2012.

Conference Proceedings

I) Conference paper titled “Fault Detection Scheme for AES S-box/ InvS-box,” *Electronic New Zealand Conference (ENZCON 2013)*, Auckland, New Zealand, pp. 75-79, 2013.

II) Conference paper titled "Topology of 2 input subnanowatt XOR gate in 65nm CMOS technology," in *Proceedings of IEEE International Conference on Semiconductor Electronics 2012 (ICSE2012)*, Sept. 19-21, Kuala Lumpur, Malaysia, 2012

III) Conference paper titled "Decomposition method for AES Mix Column and Inv Mix Column VLSI architecture optimization," in *Proceedings 18th Electronics New Zealand Conference*, Palmerston North, New Zealand, pp. 91-94, 2011.

IV) Conference paper titled "A new design of XOR-XNOR gates for low power application," in *Proceedings 2011 IEEE International Conference on Electronic Devices, Systems and Applications (ICEDSA)*, 25-27 Apr, Kuala Lumpur, Malaysia, pp.45-49, 2011.

V) Conference paper titled "Design of XOR gates in VLSI implementation," in *Proceedings 17th Electronics New Zealand Conference*, Nov. 22-23, Hamilton, New Zealand, pp. 51-55, 2010.

VI) Conference paper titled "Design of AES S-box using combinational logic optimization," in *Proceedings of IEEE Symposium on Industrial Electronics & Applications (ISIEA)*, Oct. 3-5, Penang, Malaysia, pp. 679-682, 2010.

JOURNAL PAPER I

Hindawi Publishing Corporation
Active and Passive Electronic Components
Volume 2013, Article ID 148518, 6 pages
http://dx.doi.org/10.1155/2013/148518

Research Article

A 0.8 V 0.23 nW 1.5 ns Full-Swing Pass-Transistor XOR Gate in 130 nm CMOS

Nabihah Ahmad and Rezaul Hasan

Center for Research in Analog and VLSI microsystem Design (CRAVE), School of Engineering and Advanced Technology (SEAT), Massey University, Auckland 0632, New Zealand

Correspondence should be addressed to Rezaul Hasan; hasanmic@massey.ac.nz

Received 18 December 2012; Accepted 5 March 2013

Academic Editor: Ching Liang Dai

A power efficient circuit topology is proposed to implement a low-voltage CMOS 2-input pass-transistor XOR gate. This design aims to minimize power dissipation and reduce transistor count while at the same time reducing the propagation delay. The XOR gate utilizes six transistors to achieve a compact circuit design and was fabricated using the 130 nm IBM CMOS process. The performance of the XOR circuit was validated against other XOR gate designs through simulations using the same 130 nm CMOS process. The area of the core circuit is only about 56 sq · μm with 1.5659 ns propagation delay and 0.2312 nW power dissipation at 0.8 V supply voltage. The proposed six-transistor implementation thus compares favorably with other existing XOR gate designs.

1. Introduction

Low-power circuits have become a major design forethought with the overwhelming growth of portable applications, particularly, for battery operated hand-held devices. In addition, higher power consumption raises the temperature of the chip which in turn affects the reliability of the devices and circuits [1]. Various low-power techniques have been explored to enhance the basic logic gates such as the XOR gate which influence the overall power consumption in many system-on-chip (SOC) implementations. One of the effective ways to reduce the overall power consumption is by reducing the supply voltage. XOR gate optimization in terms of power, speed, and transistor count has significantly improved the performance of larger and complex circuits. Over the years, various 2-input XOR gate designs have been widely reported to enhance the performance of various applications such as full adder, parity generator, encryption processor, and comparator. Traditional eight-transistor static CMOS XOR gate can operate with full output swing but with the drawback of power dissipation and transistor count [2]. On the other hand, XOR circuit based on the transmission gate [3] is used to overcome the signal degradation caused by the PMOS and NMOS devices in pass-transistor logic. However, it has the drawback of the loss of driving capability and requires complementary signals to switch the PMOS and NMOS devices and hence needs more transistors and area. A cross-coupled (CC) XOR gate based on the pass-transistor logic has been reported in [4], which claims to have improved speed and power consumption than the six device pass-transistor XOR gate and works well under a low-voltage regime. A six-transistor XOR gate realized from a modified four-transistor XOR gate by cascading a standard inverter as an output driver can be found in [5], which improves the poor output signal level for certain inputs. Powerless XOR gate (P-XOR) is proposed in [6] using a four-transistor circuit with no power supply connection which consumes less power than other designs but at the expense of a large delay. Another four-transistor XOR gate design was reported in [7], based on the Gate-Diffusion-Input (GDI) cell [8]. A three-transistor XOR gate can be found in [9] using a CMOS inverter and a PMOS pass transistor. It provides low-power-delay product (PDP) but has a voltage degradation with the input combination $A = 1$ and $B = 0$. Elgamel et al. [10] also proposed a similar three-transistor XOR gate, but it consumes high power when $A = 1$ and $B = 0$ and, in addition, produces a poor logic "1" for this input combination. However, it may reach an acceptable logic high voltage level with appropriate transistor sizing. Thus,

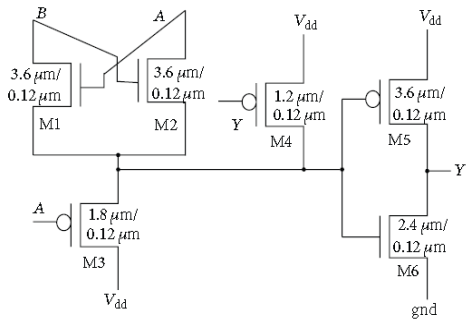


FIGURE 1: Circuit diagram of the proposed six-transistor pass-transistor-based full-swing CMOS XOR gate.

both the circuits [9, 10] may not operate reliably at low-supply voltage. In this brief paper, we present a novel low-power, low-voltage, and full-swing 2-input XOR circuit using 6 devices, implemented in 130 nm IBM CMOS technology.

2. Novel XOR Gate Topology for Low-Power CMOS Design

A low-power constrained 2-input XOR gate using six transistors is proposed which provides full output voltage swing for all input combinations and enables low-voltage operation with a small propagation delay. The proposed XOR circuit is based on pass-transistor logic with an inverter as the output driver to achieve perfect output swing. Pass-transistor design enables small transistor count along with smaller input loads (with signal input to source/drain instead of gate) offering very low-power operation with high performance. Since an NMOS device passes a strong "0" but a weak "1," while a PMOS device passes a strong "1" but a weak "0;" the complementary pass transistors are organized to pass a strong output logic level for all input combinations of "1" and "0".

Figure 1 shows the schematic of the proposed XOR circuit along with the device sizes using the 130 nm CMOS process. It performs a perfect full-swing operation for every input pattern. V_{dd} connections to the source terminals of M3 and M4 are used to drive a full-rail output of "1". For $A = B = 0$ condition, transistor M3 is ON, passing a strong "1" from V_{dd} to the inverter input which generates a "0" at the output Y to turn ON M4, when $A = 0$, $B = 1$, and transistors M2 and M3 are ON, which passes signals "0" and "1," respectively. To solve this problem, W/L ratio of M2 is increased making it larger than the W/L ratio of M3 so as to pass only a signal "0" to the inverter input and generate the strong output $Y = 1$. For $A = 1$ and $B = 0$, only the device M1 is ON, and a strong "0" is passed to the inverter generating a full-rail "1" to output Y. With $A = B = 1$, transistors M1 and M2 are ON, and a weak signal "1" is passed to the inverter input and, as a result, the output Y will also be degraded. However, the feedback path causes transistor M4 to turn ON when $Y = 0$ thus passing the perfect signal "1" from V_{dd} to the inverter input resulting in the generation of the perfect signal "0" at Y. This pass-transistor XOR thus does not suffer from signal level deteriorations like other pass-transistor XOR gates. The transistor sizes are carefully chosen for optimal power-delay performance under various operating conditions.

3. XOR Gate Performance Analysis and Simulation Results

Extensive simulations of the proposed XOR gate along with five other existing XOR gates found in the literature have been carried out using the 130 nm IBM CMOS technology in order to analyze the performance comparison. The simulations were carried out on the Cadence Spectre platform and the Synopsys HSPICE platform using the same test environment to measure the propagation delay and the power dissipation in each case. All the simulations were carried out with a 0.6 V to 1.2 V supply voltage range, a load capacitance of 10 fF and a throughput (clocking) rate of 200 MHz. The simulations consist of functional verification, power, timing analysis, design rule checking (DRC), and layout versus schematic (LVS) of the layout verified using Cadence Assura. The performance of all the XOR test circuits has been evaluated in terms of the worst-case propagation delay. Propagation delay is evaluated from the time interval between the 50% input and the 50% output voltage transition points. The figure-of-merit power-delay product (PDP) is calculated from the product of the worst case propagation delay and the average power consumption. Several input patterns that covered all possible cases of input values were applied, and the simulation results verified the correct functionality for every input combination up to the lower end of the supply voltage range. Figure 2 shows the HSPICE transient circuit simulation of the full-swing operation of this pass-transistor XOR gate indicating rise and fall times in the range of few hundred pico seconds (@ supply voltage, $V_{dd} = 0.8$ V). Table 1 summarizes the results of these simulations providing a comparison of the propagation delay, the power dissipation, and the PDP between the proposed circuit and the other recently reported designs. The proposed XOR gate offers lower-propagation delay than the other six-transistor XOR gates as shown in Table 1 and depicted in the line graph in Figure 3. The worst circuit in term of speed is the three-transistor XOR gate. It has the highest propagation delay against voltage scaling. The six-transistor designs by the authors in [4, 5] are close to the proposed XOR gate in term of power dissipation and propagation delay, but the proposed XOR gate provides better overall improvement compared to these other previous designs. The proposed XOR circuit performed satisfactorily for low-supply voltages, with the 0.2312 nW power dissipation compared to the 0.2319 nW dissipation by the design in [5] (@ supply voltage, $V_{dd} = 0.8$ V). The 4T XOR gates in [6, 7] and the 3T XOR gates in [1, 9, 10] have almost identical average power dissipation for all the supply voltages. The line graph in Figure 4 demonstrates the effect of voltage scaling on the average power dissipation. All other circuits have higher PDP than the proposed XOR circuit as evidenced in

Table 1: Comparison of the simulation results for different CMOS XOR gates with the proposed XOR gate.

	V_{dd} (v)	Proposed (6T)	6T [5]	6T [4]	4T [6]	4T[7]	3T [1, 9, 10]
Delay (ns)	0.6	2.1107	2.3113	7.0638	2.5818	4.1910	15.48
	0.8	1.5659	1.7672	3.6837	4.5128	3.9308	8.7314
	1	1.4691	1.6475	1.6469	4.2216	8.9244	13.038
	1.2	1.4200	1.6018	1.5500	4.1087	7.2075	12.901
Average power (nW)	0.6	0.1351	0.1352	0.1385	0.2672	0.2672	0.2672
	0.8	0.2312	0.2319	0.2401	0.4586	0.4586	0.4586
	1	0.3674	0.3689	0.3842	0.7269	0.7269	0.7268
	1.2	0.5523	0.5574	0.5835	1.0931	1.0931	1.0928
PDP (aJ)	0.6	0.2852	0.3125	0.7526	0.6899	1.1198	4.1363
	0.8	0.3620	0.4098	0.8845	2.0696	1.8027	4.0042
	1	0.5397	0.6078	0.6327	3.0687	6.4871	9.4760
	1.2	0.7843	0.8928	0.9044	4.4912	7.8785	14.098

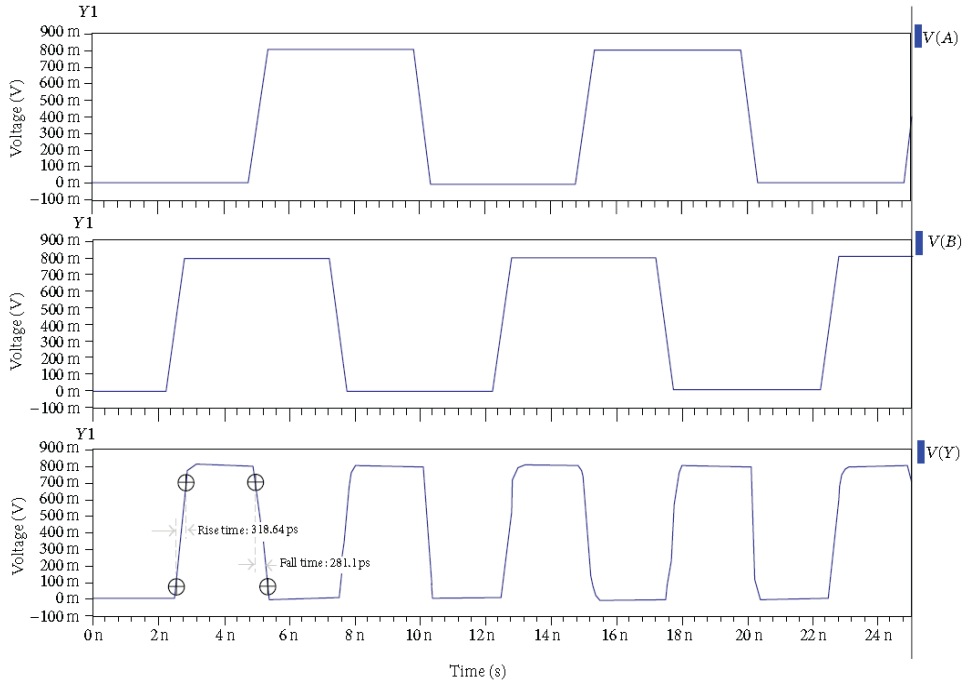


Figure 2: HSPICE transient simulation of full-swing output voltage of the novel pass-transistor XOR gate.

Table 1 and illustrated in Figure 5. At 0.8 V supply voltage, the proposed XOR circuit has at least 13.2% improvement in PDP over the design by the authors in [5] and 143.6% over the circuit by the authors in [4]. As output load is one of the parameters that affect the performance of the circuits, we have varied the output load from 10 fF to 50 fF at 0.8 V supply voltage for all circuits to study its effect on the propagation delay. The proposed XOR gate is found to be the best circuit in terms of the propagation delay for all values of output loads as shown in the line graph in Figure 6. Thus, from the simulation results, it is clear that the proposed new XOR gate has the lowest propagation delay as well as the lowest power consumption along with high output driving capability. The improvements attained by the proposed circuit are thus clearly evident when compared to these other XOR gate circuits.

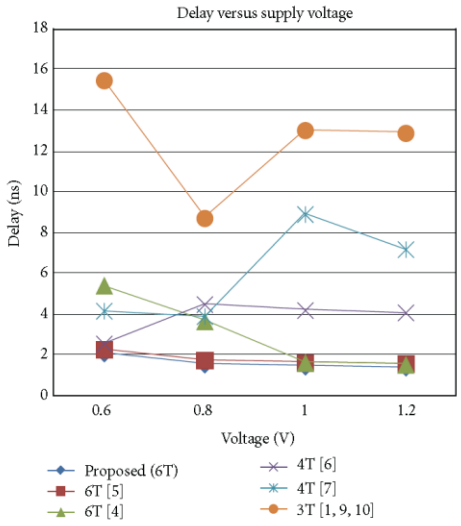


FIGURE 3: Line graph showing the comparison of the propagation delay versus supply voltage scaling for different CMOS XOR gates with the proposed novel CMOS pass-transistor-based full-swing XOR gate.

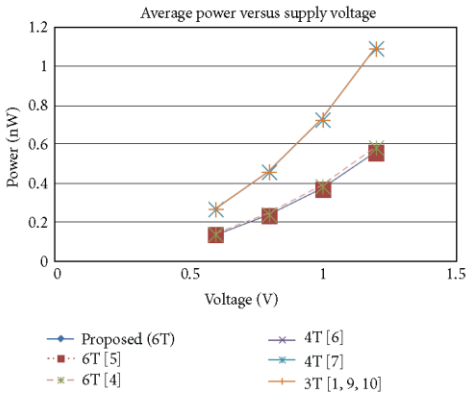


FIGURE 4: Line graph showing the comparison of the power dissipation versus supply voltage scaling for different CMOS XOR gates with the proposed novel CMOS pass-transistor-based full-swing XOR gate.

In order to verify the noise immunity of the proposed XOR gate, the noise margins (NM_H and NM_L) of the proposed XOR gate and the other gates were determined based on DC input-output voltage transfer analysis. The noise margins for a 0.8 V supply voltage are shown in Table 2. The proposed XOR gate indicates acceptable values of noise margin when compared with the other XOR gates.

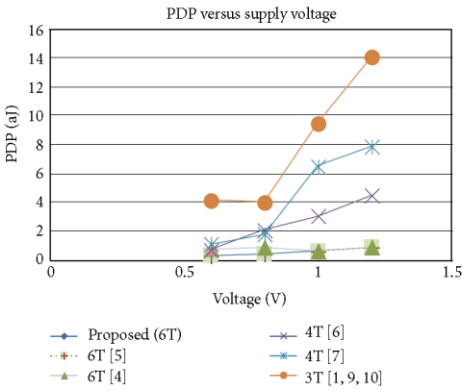


FIGURE 5: Line graph showing the comparison of the power-delay product (PDP) versus the supply voltage scaling for various XOR gates with the proposed novel CMOS pass-transistor-based full-swing XOR gate.

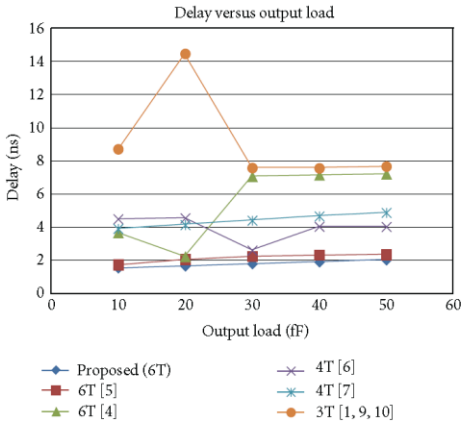


FIGURE 6: Line graph showing the comparison of the propagation delay versus output load for various CMOS XOR gates with the proposed novel CMOS pass-transistor-based full-swing XOR gate.

4. XOR Gate Fabrication and Experimental Results

The mask layout of the XOR gate illustrated in Figure 7 was fabricated in the 130 nm IBM CMOS process. Minimum channel length is used for all the devices, and optimum channel width is carefully chosen for each device in order to achieve verified functionality with low-power dissipation and smallest possible propagation delay. The photomicrograph of the fabricated XOR gate along with bonding pads is shown in Figure 8. The silicon area of the XOR gate was 8.02 μm × 7.03 μm (≈56 sq·μm) excluding the bonding pads. Figure 9

Table 2: Comparison of noise margins of different CMOS XOR gates with the proposed XOR gate.

Type of XOR	V_{dd} (V)	V_{oh} (V)	V_{ih} (V)	V_{il} (V)	V_{ol} (V)	NM_H (V)	NM_L (V)
Proposed (6T)	0.800	0.7186	0.3889	0.2867	0.0796	0.3297	0.2071
6T [5]	0.800	0.7204	0.3967	0.3022	0.0687	0.3237	0.2335
6T[4]	0.800	0.7205	0.3967	0.3022	0.0687	0.3238	0.2335
4T [6]	0.800	0.9460	0.6000	0.4489	0.1172	0.346	0.3317
4T [7]	0.800	1.0809	0.6444	0.4233	0.1423	0.4365	0.2810
3T [1, 9, 10]	0.800	1.0797	0.5978	0.4100	0.1332	0.4819	0.2768

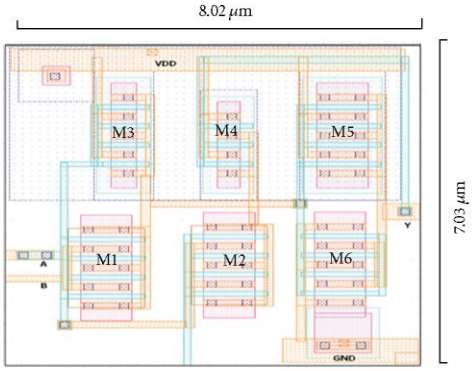


Figure 7: Layout of the proposed novel CMOS pass-transistor-based full-swing 2-input XOR gate.

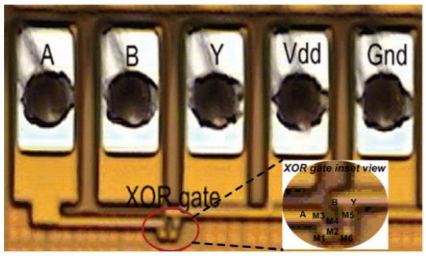


Figure 8: Microphotograph of the fabricated novel CMOS pass-transistor-based full-swing XOR gate along with bonding pads.

displays the functional verification of the XOR gate with a snapshot of the XOR input and output waveforms for several input combinations using the Tektronix TLA5202 Logic Analyzer. Level shifter was used to generate low voltage (0.8 V) from high voltage (5 V) of the pattern generator to the XOR gate input. Finally, Figure 10 provides the Agilent DCA-J (86100C Infiniium) oscilloscope electrical waveforms for the fabricated XOR gate (yellow = input A, blue = input B, and green = output Y) indicating a correct operation for a supply voltage of 0.8 V.

Figure 9: Logic analyzer waveform of the input and the output for the fabricated novel CMOS pass-transistor-based full-swing XOR gate.

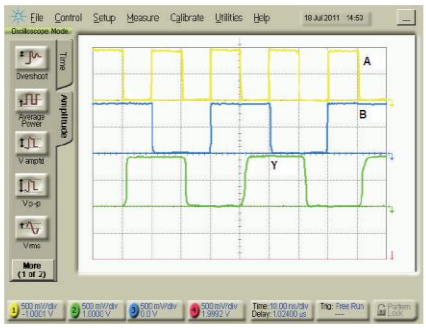


Figure 10: Oscilloscope waveforms for the fabricated novel CMOS pass-transistor-based full-swing XOR gate (yellow = input A, blue = input B, and green = output Y).

5. Conclusion

This paper demonstrates a new pass-transistor-based full-swing 2-input XOR gate topology implemented in 130 nm CMOS process suitable for reducing the power and propagation delay of an overall system on chip. The proposed XOR gate was compared to other peer XOR designs, and the results indicate satisfactory performance and improvements in term of power consumption, propagation delay, and power-delay product compared to the other designs. The proposed XOR gate has a lower-gate delay and a lower-power-delay product in comparison to its peer designs. It is thus suitable for small area and low-power applications such as RFID tags.

References

[1] M. Hosseinghadiry, H. Mohammadi, and M. Nadisenejani, "Two new low power high performance full adders with minimum gates," *International Journal of Electrical and Computer Engineering*, vol. 4, no. 10, pp. 671–678, 2009.

[2] N. Weste and K. Eshraghian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley Longman, Reading, Mass, USA, 2nd edition, 1993.

[3] S. M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits: Analysis and Design*, McGraw Hill, New York, NY, USA, 3rd edition, 2002.

[4] K. J. Lin and C. W. Wu, "Low-cost realization of multiple-input exclusive-OR gates," in *Proceedings of the 8th Annual IEEE International ASIC Conference and Exhibit*, pp. 307–310, September 1995.

[5] J. M. Wang, S. C. Fang, and W. S. Feng, "New efficient designs for XOR and XNOR functions on the transistor level," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 7, pp. 780–786, 1994.

[6] H. T. Bui, A. K. Al-Sheraidah, and Y. Wang, "New 4-transistor XOR and XNOR designs," in *Proceedings of the 2nd IEEE Asia Pacific Conference on ASICs (AP-ASIC '00)*, pp. 25–28, Cheju, Korea, 2000.

[7] A. K. Nishad and R. Chandel, "Analysis of low power high performance XOR gate using GDI technique," in *Proceedings of the International Conference on Computational Intelligence and Communication Networks*, pp. 187–191, 2011.

[8] A. Morgenshtein, A. Fish, and I. A. Wagner, "Gate-diffusion input (GDI)—a novel power efficient method for digital circuits: a design methodology," in *Proceedings of the 14th Annual IEEE International ASIC/SOC Conference*, pp. 39–43, September 2001.

[9] S. R. Chowdhury, A. Banerjee, A. Roy, and H. Saha, "A high speed 8 transistor full adder design using novel 3 transistor XOR gates," *International Journal of Electrical and Computer Engineering*, vol. 3, no. 12, pp. 784–790, 2008.

[10] M. Elgamel, S. Goel, and M. Bayoumi, "Noise tolerant low voltage XOR-XNOR for fast arithmetic," in *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI '03)*, pp. 285–288, April 2003.

JOURNAL PAPER II

Efficient integrated AES crypto-processor architecture for 8-bit stream cipher

N. Ahmad and S.M. Rezaul Hasan

Adoption of the Advanced Encryption Standard (AES) as a symmetric encryption algorithm for numerous applications requires a low cost and low power design. Presented is a new 8-bit stream cipher architecture core for an application specific integrated circuit AES crypto-processor. The chip area and power are optimised along with high throughput. It is implemented in a 130nm CMOS process and supports both encryption and decryption using 128-bit keys with a throughput of 0.05 Gbit/s (at 100 MHz clock). This design utilises 3152 gate equivalents including an on-the-fly key scheduling unit along with 4.23 μW/MHz power consumption. Compared to other 8-bit implementations, the proposed design achieves a smaller chip size along with higher throughput and lower power dissipation.

Introduction: The National Institute of Standards and Technology (NIST) selected the Rijndael algorithm for the Advanced Encryption Standard (AES) in 1997. The AES is widely adopted for a variety of encryption needs, such as Wireless Local Area Networks (WLAN) and smart cards. Hardware implementations have been developed to satisfy different constraints and requirements such as low power consumption, low area and high data rate. At the cost of increased area and power consumption, higher throughputs may be accomplished by using a loop-unrolled hardware structure. On the other hand, reducing the depth of the data path can decrease the size (hardware area) while compromising the throughput. The 8-bit stream data path architecture reported here results in a minimised power, low resource area and high throughput design.

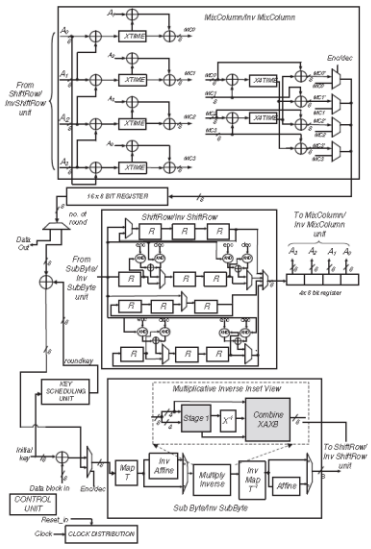


Fig. 1 *High level architecture of AES crypto-processor*

AES algorithm and architecture implementation: The AES algorithm processes fixed 128-bit data blocks and variable length cipher keys of 128, 192 and 256 bits. The basic AES operation is performed by a two-dimensional array of bytes, called the State matrix. The AES is an iterative algorithm consisting of four different data transformations: SubByte, ShiftRow, MixColumn and AddRoundKey. It is a block cipher but can be adopted to operate in the stream cipher mode.

The proposed AES core supports 128-bit keys in an 8-bit wide stream data path in order to minimise the silicon area and power consumption. Fig. 1 represents the top-core AES architecture of the proposed design. The component parts are, the SubByte/Inv SubByte, MixColumn/Inv MixColumn, ShiftRow/Inv ShiftRow, Key Scheduling unit, control unit and clock distribution unit. Each round is completed in 28 clock cycles. The control unit executes the sequence of the AES operation while the clock distribution unit provides the global clock to the components in parallel synchronism avoiding delay mismatch.

SubByte is a nonlinear operation on each byte of the State matrix employing 16-byte S-Boxes defined as multiplicative inverse in the finite field $GF(2^8)$ using the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ followed by an affine transformation. The new architecture uses a technique based on composite field arithmetic with a different choice of the polynomial coefficients and the implementation of the constant multiplication, in order to reduce gate count and gate delay. It consists of stage 1 in Fig. 1, along with inversion and multiplication in $GF(2^4)$. A design using only 147 gates (107 XOR gates and 40 AND gates) was achieved, compared to typical architectures in [1] and [2] with 172 gates (137 XOR gates and 35 AND gates) and 158 gates (123 XOR gates and 35 AND gates), respectively. For multiplicative inverse, the proposed architecture has a critical path of 13 gate delay compared to 17 gate delay for typical composite field design.

ShiftRows and the Inv ShiftRow is a linear cyclic shift operation in each row of four 4-byte data blocks with different offset (0–3-byte offsets) to the left or the right, respectively. The proposed AES core employs a joint ShiftRow and Inv ShiftRow architecture using twelve 8-bit registers along with six 2-to-1 and one 4-to-1 multiplexers. The operation of the registers is controlled by four different clocks for each row. Table 1 indicates the cost for different implementations based on the number of registers and multiplexers in the gate equivalent (GE) measure. Compared to [3], which also implements an 8-bit AES data path, the proposed architecture requiring 448 gates (equivalent to 1792 transistors) reduces the hardware area by 22%.

Table 1: Comparison of ShiftRow/Inv ShiftRow architectures in GEs

ShiftRow/Inv ShiftRow implementation	[4] enc	[4] dec	[3] enc	[5]	This work
GE	896	896	572	532	448

MixColumn performs a polynomial multiplication over $GF(2^8)$ modulo $x^4 + 1$ of a 4 byte × 4 byte matrix generated by four right cyclic rotations of the coefficients of the polynomial $c(x) = \{02\}_{16}x^3 + \{03\}_{16}x^2 + \{01\}_{16}x + \{01\}_{16}$. Optimised MixColumn and Inv MixColumn is implemented using the decomposition method, by applying resource sharing to the computation within a byte in a given column of the State matrix. The full State matrix transformation requires 664 XOR gates and results in a critical path of seven XOR gate delay. Fig. 2 shows a comparison with other implementations in terms of area (AXOR) and critical path delay (TXOR). The proposed integrated MixColumn and Inv MixColumn design achieved 64.8% area reduction compared to the direct implementation [4]. On the other hand, the design in [6] with a shorter critical path delay by two XOR gates, requires 16% more area compared to the proposed architecture. With the same area as the design in [7], the proposed architecture can lead to better performance with a shorter critical path delay.

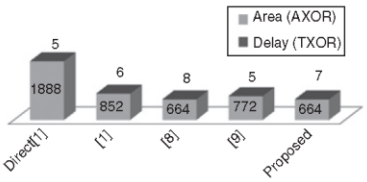


Fig. 2 *Comparison of MixColumn/Inv MixColumn architectures*

Composite results and discussion: The new AES crypto-processor architecture has been implemented in 130nm IBM CMOS technology employing the above described optimised gate level hardware. Sets of NIST test vectors were used to verify the functionality. Operating with a 100MHz global clock (at VDD = 0.8V) the design achieved a throughput of 0.05 Gbit/s. The comparison in Table 2 shows that the proposed AES core requires 3152 GEs, slightly more than the smallest design in [3] which only supports the encryption mode. It offers the second lowest power consumption compared to the design in [8] which uses the 65 nm node with low threshold voltage allowing

ultra-low Vdd (≃0.3–0.5V). By using deep nanometer CMOS along with voltage scaling as an effective constraint, the proposed architecture can achieve even lower power budget. Considering the throughput factor, the proposed AES offers higher performance except for the designs in [3] and [9]. For the high-speed system, a throughput-to-area ratio can be used as a performance metric. However, for resource-critical applications, the power-area-latency (P-A-T) metric is used. The proposed AES design demonstrates the best P-A-T efficiency. Although the low power dissipation provides some immunity against side-channel attack, this implementation may not be suitable in a hostile environment.

Table 2: Comparison with previous 8-bit AES implementations

Design	Tech. (nm)	Cycle count	Area GEs*	Throughput (Mbit/s)	Max. freq. (MHz)	Power (μw/MHz)	P-A-T (μJ-gate)
[3] Enc only	130	160	3100	121	153	36.76	18.23
[10]	130	356	5500	0.036	0.1	6.92	13.53
	130	356	5500	4.31	12	8.25	16.15
[9]	180	160	5600	104	130	57	51.07
[8]	65	1142	3500	0.036	0.322	0.78	3.12
	65	1142	3500	0.1	0.89	0.96	3.84
This work	130	240	3152	53.3	100	4.23	3.2

*Gate equivalent (GE) in terms of 2-input NAND gate

Conclusion: A compact AES architecture with an 8-bit stream data path has been developed to achieve a small area circuit with marginal power budget without sacrificing the throughput performance.

2 September 2012
doi: 10.1049/el.2012.2932

N. Ahmad and S.M. Rezaul Hasan (*Center for Research in Analog and VLSI MicroSystem Design, School of Engineering and Advanced Technology, Massey University, New Zealand*)

E-mail: hasanmic@massey.ac.nz

References

1 Satoh, A., Morioka, S., Takano, K., and Munetoh, S.: 'A compact Rijndael hardware architecture with S-Box optimization'. ASIACRYPT'01, Springer-Verlag, 2001, pp. 239–254
2 Ahmad, N., and Rezaul Hasan, S.M.: 'Low-power compact composite field AES S-Box/Inv S-Box design in 65nm CMOS using novel XOR gate', *Integration*, the VLSI journal, 2012. http://www.sciencedirect.com/science/article/pii/S0167926012000375
3 Hamalainen, P., Alho, T., Hannikainen, M., and Hamalainen, T.D.: 'Design and implementation of low-area and low-power AES encryption hardware core'. Conf. on Digital System Design: Architectures, Methods and Tools, Dubrovnik, Croatia, 2006, pp. 577–583
4 Hsiao, S.F., and Chen, M.C.: 'Efficient substructure sharing methods for optimizing the inner-product operations in Rijndael advanced encryption standard', *IEE Proc., Comput. Digit. Tech.*, 2005, **152**, (5), pp. 653–665
5 Li, H., and Li, J.: 'A new compact architecture for AES with optimized ShiftRows operation'. IEEE Int. Symp. on Circuits and Systems, New Orleans, LA, USA, 2007, pp. 1851–1854
6 Hsiao, S.F., Chen, M.C., and Tu, C.S.: 'Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations', *IEEE Trans. Circuits Syst. I*, 2006, **53**, (3), pp. 615–626
7 Nalini, C., Anandmohan, P.V., Poomaiah, D.V., and Kulkarni, V.D.: 'Compact designs of Subbytes and MixColumn for AES'. IEEE Int. Advance Computing Conf., Patiala, India, 2009, pp. 1241–1247
8 Hocquet, C., Kamel, D., Regazzoni, F., Legat, J.D., Flandre, D., Bol, D., and Standaert, F.X.: 'Harvesting the potential of nano-CMOS for lightweight cryptography: an ultra-low-voltage 65 nm AES coprocessor for passive RFID tags', *Springer J. Cryptograph. Eng.*, 2011, **1**, (1), pp. 79–89
9 Haghighizadeh, F., Attarzadeh, H., and Sharifkhani, M.: 'A compact 8-bit AES crypto-processor'. 2nd. Int. Conf. on Computer and Network Technology, Bangkok, Thailand, 2010, pp. 71–75
10 Good, T., and Benaissa, M.: '692-nW Advanced Encryption Standard (AES) on a 0.13-μm CMOS', *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, **18**, (12), pp. 1753–1757

JOURNAL PAPER III

ARTICLE IN PRESS

INTEGRATION, the VLSI journal ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at SciVerse ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Low-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate

Nabihah Ahmad, S.M. Rezaul Hasan *

Center for Research in Analog & VLSI microsystem Design (CRAVE), School of Engineering and Advanced Technology (SEAT), Massey University, Albany, Auckland 0632, New Zealand

ARTICLE INFO

Article history:
Received 30 September 2011
Received in revised form
23 April 2012
Accepted 22 June 2012

Keywords:
S-Box
AES
Composite field
Galois Field
Low power
Delay
Rijndael algorithm

ABSTRACT

The Substitution box (S-Box) forms the core building block of any hardware implementation of the Advanced Encryption Standard (AES) algorithm as it is a non-linear structure requiring multiplicative inversion. This paper presents a full custom CMOS design of S-Box/Inversion S-Box (Inv S-Box) with low power *GF* (2^8) Galois Field inversions based on polynomial basis, using composite field arithmetic. The S-Box/Inv S-Box utilizes a novel low power 2-input XOR gate with only six devices to achieve a compact module implemented in 65 nm IBM CMOS technology. The area of the core circuit is only about 288 μm^2 as a result of this transistor level optimization. The hardware cost of the S-Box/Inv S-Box is about 158 logic gates equivalent to 948 transistors with a critical path propagation delay of 7.322 ns enabling a throughput of 130 Mega-SubBytes per second. This design indicates a power dissipation of only around 0.09 μW using a 0.8 V supply voltage, and, is suitable for applications such as RFID tags and smart cards which require low power consumption with a small silicon die. The proposed implementation compares favorably with other existing S-Box designs.

1. Introduction

As cryptography plays a crucial role in the security of data transmission, AES based on Rijndael algorithm [1] was selected as a data encryption standard by the National Institute of Standards and Technology (NIST) in 1997 based on the primary criteria of security, performance, efficiency in software and hardware implementation, and flexibility. AES is one of the most common symmetric encryption algorithms and is widely adopted for a variety of encryption needs, such as wireless networks and secure transactions via the Internet. AES can be implemented on a wide range of platforms under different constraints [2]. In portable applications computing resources are usually limited and dedicated hardware implementation of the security process is essential [39]. Implementation using Field Programmable Gate Array (FPGA) is not suitable for such applications mainly due to size and power constraints. FPGA being a general purpose logic array usually there is some residual (unused) logic and I/O blocks, and consequently, highly compact implementation is difficult to achieve. In addition, FPGA implementation is prone to switching noise induced power analysis attack [42]. A compact small footprint full-custom chip is more suitable in such a case. In addition, such a dedicated hardwired AES implementation can provide higher data rate for fast handling of ciphered network data packets in applications such as routers compared to software packages. The hardwired implementation is also physically secure since tempering by an attacker is more difficult. The overall efficiency of AES hardware implementation in terms of size, speed, security and power dissipation depends largely on the AES architecture [40]. For high throughput, loop-unrolled pipelined structure [4] is used, but on the other hand, to save power and area, iterative single round with resource sharing is implemented.

The S-Box is at the core of any AES implementation and is considered a full complexity design consuming the major portion of the power and energy budget of the AES hardware. This paper is focused on area-efficient low-voltage and low-power CMOS implementation of the S-Box/Inv S-Box. There are various reported techniques to implement the S-Box to satisfy the varying criteria such as power, speed and delay for different applications. Among them there are two main streams: (a) Implementation using look up tables (LUTs) which stores all predefined 256 8-bit values of S-Box in a Read-Only-Memory (ROM). The advantage of using LUT is that it offers a shorter critical path. However, it has a drawback of the unbreakable delay path [3] in pipelined designs, and hence it is not suitable for high speed applications. This delay prohibits each round unit from being divided into more than two sub-stages to achieve any further increase in processing speed [41]. It also requires a large area to implement both AES encryption and decryption as a different table is used in each case.

* Corresponding author. Tel.: +64 9 414 0800x41283; fax: +64 9 443 9774.
E-mail addresses: N.Ahmad@massey.ac.nz (N. Ahmad), hasanmic@massey.ac.nz (S.M. Rezaul Hasan).

0167-9260/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.vlsi.2012.06.002

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

(b) The alternative way is to design the S-Box circuit using combinatorial logic directly from its arithmetic operations. This approach has breakable delay-path for S-Box processing. Other S-Box architectures, such as positive polarity Reed–Muller structure [6], binary decision diagram (BDD) [7], or its variance, the twisted binary decision diagram (TBDD) [8] can achieve a high speed design but suffer from extremely large area cost. The S-Box design based on sum of product (SOP) expressions in Refs. [9–11] also suffer from large silicon area penalty.

A well-known approach to design S-Box from its arithmetic operations involves multiplicative inversion in $GF(2^8)$ using composite field arithmetic [12,13], decomposing the field operations from $GF(2^8)$ to $GF((2^4)^2)$. Subfield arithmetic is thus used in the computation of an inverse in the Galois Field. In this technique, hardware area cost can be reduced substantially by sharing the multiplicative inverse step for the SubBytes and the InvSubBytes operations. Also, among existing techniques, composite field S-Box architecture is the most area-efficient approach for AES encryption/decryption algorithm as the computation cost of certain Galois Field operations is lower when the operation is performed in an isomorphic composite field. The authors in Ref. [14] reported a fast composite field S-Box architecture that showed an increased throughput rate of 56.25% along with reduced pipeline latency by 40%–60% compared with other conventional designs. The approaches in Refs. [2, 15] results in a very small size of the S-Box, but suffers from a longer critical path than LUT technique. The LUT technique on the other hand has a shorter critical path compared to the composite field approach, but its area-size is 2–3 times larger.

Next, considering the S-box design methodologies reported so far, only Refs. [16,17,36] evaluated the performances of the S-Box using the full custom design technique. The advantage of full custom design using state of the art CMOS processes is that it is possible to scale all the transistors down with process scaling without deteriorating the overall performance along with increased speed in most cases. This leads to smaller chip area and low power consumption. Another design methodology is to reduce power consumption by using advanced process technology that offers very low supply voltage. This approach also leads to a reduction in the die area.

S-Box architectures, especially the composite field approach uses the XOR gate as the fundamental logic function along with AND gates. Consequently, enhancing the performance of the XOR gates can significantly improve the critical path performance and die area of the S-Box design. In this paper, we present a low-power design methodology for the S-Box/Inv S-Box which includes minimizing the overall circuit size and critical path delay by implementing a new XOR gate, scaling down the supply voltage and the transistor size, along with choosing an advanced technology for optimized CMOS full custom design. Our approach of optimized full-custom S-Box/Inv S-Box implementation in low cost isomorphic composite field arithmetic using low power minimal transistor count XOR gates have not been considered before in the context of AES implementations. To the best of the authors' knowledge, most reported works use standard static CMOS XOR logic gates requiring 12 transistors resulting in a larger overall silicon-area in spite of any architectural optimization. In addition, minimized implementation of InvSubBytes for Inv S-Box by sharing S-Box resources on the same chip was not considered in many previously reported works.

2. AES algorithm and s-box implementation preliminaries

AES is a symmetric encryption algorithm which processes a fixed 128-bit data block and variable length keys of 128, 192 and 256 bits. The data block is mapped into a 4×4 array of byte elements called the State matrix. Each byte in the State is considered an element in $GF(2^8)$ and denoted by S_{ij} $(0\le i,j<4)$. The AES is also an iterative algorithm which performs iteratively for 10, 12 or 14 rounds depending on the key length. The AES contains four different data transformations: SubBytes, ShiftRows, MixColumns and the AddRoundKey using a round function. The initial round performs only the AddRoundKey with the State, and in the final round the MixColumns does not apply. The decryption consists of inverse transformations. The irreducible polynomial used to build the $GF(2^8)$ for the AES algorithm is

$$p(x)=x^8+x^4+x^3+x+1.$$

The SubBytes transformation is a bijective mapping from eight bits to eight bits [2] applied independently on each byte of the State using 16-byte (128-bit) S-Boxes. It is a substitution function whose non-linearity is used to defend against linear cryptanalysis. Expressed mathematically it is an affine transformation of the Multiplicative inverse of each byte of the state in $GF(2^8)$ [1],

$$S'_{i,j}=MS_{i,j}^{-1}+C \tag{1}$$

Where, M is an 8×8 matrix and C is a 8×1 vector for affine transformation with elements in $GF(2)$ with,

$$M=\begin{bmatrix}1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\\1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\end{bmatrix} \tag{2}$$

and,

$$C=\begin{bmatrix}0&1&1&0&0&0&1&1\end{bmatrix} \tag{3}$$

The inverse operation (InvSubBytes) is represented by the operation,

$$S'_{i,j}=[M^{-1}(S_{i,j}+C)]^{-1} \tag{4}$$

where, M^{-1} and C' $(=M^{-1}\cdot C)$ for the inverse affine transformation are given by,

$$M^{-1}=\begin{bmatrix}0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\\0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\end{bmatrix} \tag{5}$$

And,

$$[C']=\begin{bmatrix}0&0&0&0&0&1&0&1\end{bmatrix} \tag{6}$$

ShiftRows is a row-wise linear cyclic byte-shift operation on 4-byte data blocks of the State matrix with different offsets (0–3-bytes respectively for the 1st–4th row) to the left. The inverse of this transformation is computed by performing the corresponding rotations to the right. The MixColumns transformation performs a polynomial multiplication over $GF(2^8)$ modulo x^4+1 of a 4 byte $\times$ 4 byte matrix generated by 4 right cyclic rotations (0–3 byte locations) of the coefficients of the constant polynomial $c(x)=\{02\}_{16}x^3+\{03\}_{16}x^2+\{01\}_{16}x+\{01\}_{16}$ and each column vector of the State matrix. The 4 bytes in each column of the State are considered as coefficients of polynomials over $GF(2^8)$

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

modulo x^4+1. This leads to transformed columns of the State matrix given by

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02_{16} & 03_{16} & 01_{16} & 01_{16} \\ 01_{16} & 02_{16} & 03_{16} & 01_{16} \\ 01_{16} & 01_{16} & 02_{16} & 03_{16} \\ 03_{16} & 01_{16} & 01_{16} & 02_{16} \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \quad (7)$$

For the InvMixColumns for decryption, the same procedure is followed using the constant polynomial $c'(x)=\{0E\}_{16}x3+\{0B\}_{16}x^2+\{0D\}_{16}x+\{09\}_{16}$.

Finally, the AddRoundKey is a simple bit-wise XOR operation on the 128-bit round keys and the data in each round of the AES process. Round keys consists of four 4-byte words generated by a key scheduling process.

This paper explores the implementation of S-Box in SubBytes transformation that operates on individual bytes using a substitution table (S-Box) containing a permutation of all 256 possible 8-bit values. Two transformations are required, first, byte replacement with its multiplicative inverse in the finite field $GF(2^8)$ using the irreducible polynomial $p(x)=x^8+x^4+x^3+x+1$ and, second, an affine transformation over $GF(2^8)$. For decryption operation, inverse S-Box is obtained by applying inverse affine transformation followed by multiplicative inversion in $GF(2^8)$. As mentioned in the introduction, this paper focuses on the composite field technique for full-custom CMOS implementation of the AES S-Box using low-power XOR gates. There are different construction schemes using composite field found in Refs. [2,5,30]. Rijmen [13] proposed the first efficient hardware implementation of the multiplication inversion in $GF(2^8)$. Rudra et al. [30] and Wolkerstofer et al. [31] decomposed the elements of $GF(2^8)$ into $GF(2^4)^2$ to implement the multiplicative inverse in SubBytes. The transformation matrix from $GF(2^8)$ to $GF(((2^2)^2)^2)$ was proposed by Satoh et al. [3] and has been claimed to be the most minimized hardware implementation to date with a gate complexity of 5400 gates. Mentens et al. [32] also used the same approach as in Ref. [3] but with different polynomial coefficients and achieved slightly more optimized hardware than that in Ref. [3]. Other efforts towards the efficient implementation of the S-Box include those by authors in Refs. [33–35, 45].

Low Power Strategies: The total power dissipation budget for any CMOS S-Box implementation consists of static power and dynamic power. The static power dissipation is due to the gate leakage current, the subthreshold currents in the weak inversion cutoff states of the MOSFETs, and the leakage currents in the reverse-biased source and drain p–n junction diodes. The leakage current depends on the specific fabrication technology. Technologies with smaller feature sizes will usually have higher static power consumption due to the leakage currents [43]. Fig. 1 shows the sources of static leakage dissipation. The dynamic power dissipation occurs during the charging and discharging of capacitive load (C_L) during logic state transition. It is the sum of the short circuit dissipation due to the simultaneous conduction of the n-MOS and p-MOS devices, and the load capacitance switching dissipation. Dynamic switching dissipation is usually dominant in CMOS circuitry [37], which is given by, $P_{switching}=\alpha f_{CLK}C_L V_{DD}^2$, where α is the activity factor, f_{CLK} is the frequency of operation and V_{DD} is the supply voltage. Also, C_L is the capacitance that loads the output node which can be defined as $C_L=C_{internal}+C_{interconnect}+C_{load}$, with, $C_{internal}$ being the internal capacitance of the output device associated with its drain, $C_{interconnect}$ being the interconnect capacitance, and, C_{load} being the input capacitance associated with the loading logic gates. Capacitances at logic outputs in the S-Box thus has to be reduced by proper loading and interconnect minimized logic design, in addition to considering supply voltage scaling and throughput

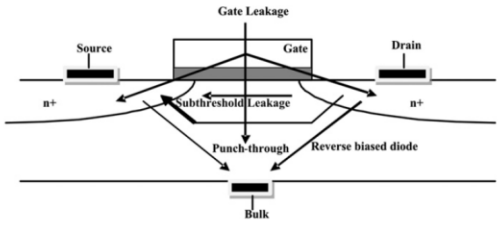


Fig. 1. Static CMOS leakage sources.

constraints. Hence one technique to reduce power consumption is by reducing the switching activity within the S-Box logic if the supply voltage and the IC technology are fixed. On the other hand, if the supply voltage is scalable, then the efficient technique to save power is to reduce the supply voltage. As the transistor geometries are reduced, progressively smaller supply voltage is required in order to avoid electrical break down as well as achieve the required performance. The threshold voltage must be decreased in case of reduced supply voltage, but this results in higher leakage currents in the transistors [44] which can be detrimental for the integrity of the S-Box implementation. Considering low-power implementation, the authors in Ref. [38] claim to be the first submicrowatt design of AES encryption with power dissipation of 692 nW. This design used a 100 kHz clock, a core voltage of 0.75 V and had a die area of 21,000 μm^2. However, a full custom design of S-Box with a low power consumption of less than 100 nW can be found in Ref. [18]. Other reported work focused on power consumption of SubBytes was carried out by Liang Li et al. [36] using a Sense Amplifier Based Logic (SABL) which operates with a constant power consumption.

3. Design methodology and proposed S-box/INV S-box architecture

The proposed S-Box/Inv S-Box architecture employs combinational logic using composite field arithmetic based on Ref. [3] and optimized in Ref. [32] with a different choice of the polynomial coefficients and the implementation of the constant multiplication with λ. The S-Box is implemented using XOR circuits, multiplexers and AND gates. The Optimization of the low voltage and low power composite field S-Box implementation has been further enhanced in this paper by using a new six transistors XOR gate and compared to [32] the Inverse S-Box for decryption is also implemented by the same chip. In addition, it employs a low-power design methodology such as minimizing the circuit size by efficient logic implementation as well as reduced supply voltage using an advanced CMOS technology. The proposed S-Box architecture is illustrated in Fig. 2 which can perform both SubBytes and InvSubBytes simply by switching combinatorial logic blocks using multiplexers. The composite field inversion by extending $GF(2^8)$ over $GF(((2^2)^2)^2)$ can be used to create compact AES implementations [3,30]. This approach has been chosen to achieve small area design. The column vectors of the State matrix first goes into isomorphic transformation from $GF(2^8)$ into the composite field $GF(((2^2)^2)^2)$ followed by inversion in composite field and inverse isomorphic transformation. Then an affine transformation is carried out to create the cipher data. This can be represented by the elementary transformations, $b \xrightarrow{ISO} q \xrightarrow{MINV} q' \xrightarrow{INVISO} b' \xrightarrow{AFFINE} b''$ where, b are byte elements of the state matrix, q are byte elements of the

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

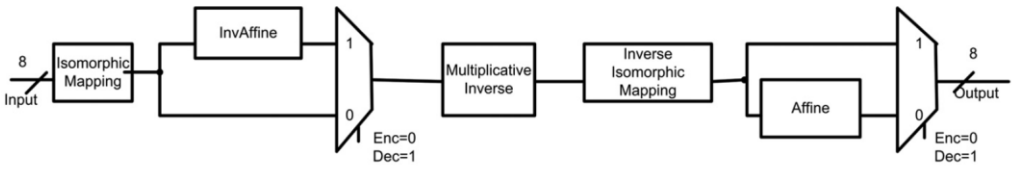


Fig. 2. SubBytes and InvSubBytes shared architecture block diagram.

isomorphic mapping transformation, q' are multiplicative inverse elements of the isomorphic state, b' are the elements after inverse isomorphic mapping and finally b'' are elements after affine transformation. The Inverse SubBytes involves first an isomorphic transformation followed by an inverse affine transformation. Then inversion in composite field is carried out followed by inverse isomorphic mapping. This can be represented by the elementary transformations, $b'' \xrightarrow{ISO} q'' \xrightarrow{INVAFFINE} q' \xrightarrow{MINV} q \xrightarrow{INVISO} b$. The affine transformation, AT operates on the $GF(2^8)$ multiplicative inverse of bytes, b of the state matrix while the inverse affine transformation AT^{-1} operates on the isomorphic affine transformed $GF(2^8)$ multiplicative inverse of the same bytes, b represented by q''. The resultant matrix operations of AT and AT^{-1} using Eqs. (1). (2), (3), (4), (5) and (6) can be translated to the logical implementations given by

$$[AT(b')] = \begin{bmatrix} b_0'' \\ b_1'' \\ b_2'' \\ b_3'' \\ b_4'' \\ b_5'' \\ b_6'' \\ b_7'' \end{bmatrix} = \begin{bmatrix} \overline{b_0' \oplus b_4' \oplus b_5' \oplus b_6' \oplus b_7'} \\ \overline{b_0' \oplus b_1' \oplus b_5' \oplus b_6' \oplus b_7'} \\ b_0' \oplus b_1' \oplus b_2' \oplus b_6' \oplus b_7' \\ b_0' \oplus b_1' \oplus b_2' \oplus b_3' \oplus b_7' \\ b_0' \oplus b_1' \oplus b_2' \oplus b_3' \oplus b_4' \\ \overline{b_1' \oplus b_2' \oplus b_3' \oplus b_4' \oplus b_5'} \\ \overline{b_2' \oplus b_3' \oplus b_4' \oplus b_5' \oplus b_6'} \\ b_3' \oplus b_4' \oplus b_5' \oplus b_6' \oplus b_7' \end{bmatrix} \tag{8}$$

$$[AT^{-1}(q'')] = \begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \\ q_4' \\ q_5' \\ q_6' \\ q_7' \end{bmatrix} = \begin{bmatrix} \overline{q_2'' \oplus q_5'' \oplus q_7''} \\ q_0'' \oplus q_3'' \oplus q_6'' \\ \overline{q_1'' \oplus q_4'' \oplus q_7''} \\ q_0'' \oplus q_2'' \oplus q_5'' \\ q_1'' \oplus q_3'' \oplus q_6'' \\ q_2'' \oplus q_4'' \oplus q_7'' \\ q_0'' \oplus q_3'' \oplus q_5'' \\ q_1'' \oplus q_4'' \oplus q_6'' \end{bmatrix} \tag{9}$$

The S-Box is optimized by performing the inversion operation in a composite Galois Field of $GF((2^4)^2)$ or $GF(((2^2)^2)^2)$, obtained from $GF(2^8)$ via isomorphic mapping. The isomorphic mapping, ISO and its inverse, ISO^{-1} [3,32] as well as their logical implementations are given by

$$[ISO(b)] = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

$$= \begin{bmatrix} b_0 \oplus b_2 \\ b_1 \oplus b_6 \oplus b_7 \\ b_2 \oplus b_3 \oplus b_5 \oplus b_7 \\ b_2 \oplus b_5 \\ b_1 \oplus b_3 \oplus b_6 \oplus b_7 \\ b_1 \oplus b_4 \oplus b_5 \oplus b_6 \\ b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \\ b_5 \oplus b_7 \end{bmatrix} \tag{10}$$

$$\left[ISO^{-1}(q)\right] = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{bmatrix}$$

$$= \begin{bmatrix} q_0 \oplus q_1 \oplus q_3 \oplus q_5 \oplus q_6 \\ q_4 \oplus q_7 \\ q_1 \oplus q_3 \oplus q_5 \oplus q_7 \\ q_1 \oplus q_3 \\ q_1 \oplus q_5 \oplus q_7 \\ q_1 \oplus q_2 \oplus q_3 \oplus q_5 \oplus q_6 \\ q_2 \oplus q_3 \oplus q_4 \oplus q_5 \oplus q_6 \\ q_1 \oplus q_2 \oplus q_3 \oplus q_5 \oplus q_6 \oplus q_7 \end{bmatrix} \tag{11}$$

Elements in $GF(2^8)$ after isomorphic transformation can be represented as $gx+e$ in the composite field $GF(((2^2)^2)^2)$, where g is the most significant nibble while e is the least significant nibble and g, e $GF((2^2)^2)$. This way $GF(((2^2)^2)^2)$ is generated as a field extension of degree 2 over the field $GF((2^2)^2)$ using the irreducible polynomial $f(x)=x^2+Ax+B$ where, A,B $GF((2^2)^2)$. The multiplicative inverse can be computed using the equation below [13]:

$$(gx+e)^{-1} = g\left(g^2B+geA+e^2\right)^{-1}x+(e+gA)\left(g^2B+geA+e^2\right)^{-1} \tag{12}$$

Choosing $A=1$ and $B=\lambda$, the equation can be simplified to

$$(gx+e)^{-1} = g\left(g^2\lambda+e(g+e)\right)^{-1}x+(e+g)\left(g^2\lambda+e(g+e)\right)^{-1} \tag{13}$$

In Eq. (13), the multiplication, addition, squaring and inversion are in $GF(2^4)$. This is shown in Fig. 3. It is to be noted here that addition of words in Galois Field is a simple bitwise XOR without carry unlike binary carry-propagation arithmetic. Using several layers of hierarchical composite field decompositions the multiplicative inverse computation is simplified resulting in simple bitwise XOR and AND operations. The hierarchical composite field decomposition using extensions over lower order fields and the corresponding choice of irreducible polynomials and coefficients

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

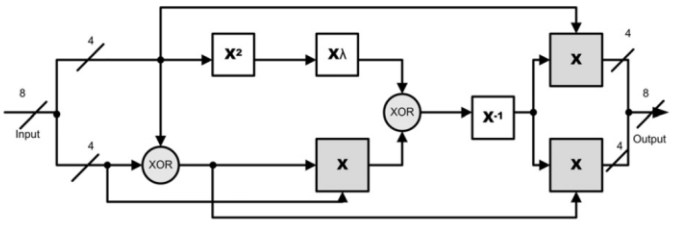


Fig. 3. Multiplicative Inverse in $GF(2^8)$ as extension of degree 2 over $GF((2^2)^2)$.

are as follows:

$$GF\left(\left(\left(2^2\right)^2\right)^2\right)\rightarrow \mathrm{GF}\left(\left(2^2\right)^2\right) : x^2+Ax+B,$$

where,

$$\mathrm{A},BGF\left(2^2\right)^2) \quad \text{and} \quad \mathrm{A}=1,B=\lambda \tag{14}$$

$$GF\left(\left(2^2\right)^2\right)\rightarrow GF\left(2^2\right) : x^2+Z_1x+Z_0,$$

where

$$Z_1,Z_0GF\left(2^2\right) \quad \text{and} \quad Z_1=1,Z_0=\varphi \tag{15}$$

$$GF\left(2^2\right)\rightarrow GF(2) : x^2+x+1 \tag{16}$$

The constants are chosen based on Ref. [32] for the optimal hardware solution, so that $\varphi=\{10\}_2$ and $\lambda=\{1000\}_2$ which leads to a total number of '1' entries for the transformation matrices equal to 54 reduced by 5 from Satoh's constant multiplication but requires 1 extra XOR gate. The following describes how the various arithmetic operations are carried out by composite field decomposition into lower order fields.

3.1. Multiplication of nibble with constant, λ

For multiplication of nibble, $q=\{q_3q_2q_1q_0\}_2$ with constant, $\lambda=\{1000\}_2$ in $GF(2^4)$, let $k=\{k_3k_2k_1k_0\}_2$ be the product nibble in $GF(2^4)$. Then, q, λ and k can be expressed as extensions over the $GF(2^2)$, so that, $k=\{k_3k_2\}_2x+\{k_1k_0\}_2=k_Hx+k_L$ where, $k_H=\{k_3k_2\}_2$ and $k_L=\{k_1k_0\}_2$. Also, $q=\{q_3q_2\}_2x+\{q_1q_0\}_2=q_Hx+q_L$ where, $q_H=\{q_3q_2\}_2$ and $q_L=\{q_1q_0\}_2$. In addition, $\lambda=\{10\}_2x+\{0\ 0\}_2=\lambda_Hx+\lambda_L$ where, $\lambda_H=\{10\}_2$ and $\lambda_L=\{0\ 0\}_2$. Here, bit-pairs k_H, k_L, q_H, q_L, λ_H and λ_L are $\in GF(2^2)$. Then the product, $k=(q_Hx+q_L)(\lambda_Hx+\lambda_L)=q_H\lambda_Hx^2+q_L\lambda_Hx$. Next, by substituting $x^2=x+\varphi$, the new equation for k is given by, $k=q_H\lambda_H(x+\varphi)+q_L\lambda_Hx=(q_H\lambda_H+q_L\lambda_H)x+q_H\lambda_H\varphi=k_Hx+k_L$ *where,* $k_H=(q_H\lambda_H+q_L\lambda_H)$, $k_L=q_H\lambda_H\varphi$ are $\in GF(2^2)$. Now, expressing the bit-pairs k_H, k_L, q_H, q_L, λ_H, λ_L and φ over $GF(2)$, we have, $k_H=(q_3x+q_2)(1x+0)+(q_1x+q_0)(1x+0)=q_3x^2+q_2x+q_1x^2+q_0x$. Next, making the substitution $x^2=x+1$, $k_H=q_3(x+1)+q_2x+q_1(x+1)+q_0x=(q_3+q_2+q_1+q_0)x+(q_3+q_1)=k_3x+k_2$. Similarly, we have, $k_L=(q_3x+q_2)(1x+0)(1x+0)=(q_3x+q_2)x^2=q_3x^3+q_2x^2$. Now, $x^3=x^2\cdot x=(x+1)\ \ x=x^2+x=x+1+x=1$ in $GF(2)$, so that, $k_L=q_3+q_2(x+1)=q_2x+(q_3+q_2)=k_1x+k_0$. Hence, finally the product nibble k is as follows:

$$\left.\begin{aligned} k_3&=(q_3+q_2+q_1+q_0)\\ k_2&=(q_3+q_1)\\ k_1&=q_2\\ k_0&=(q_3+q_2)\end{aligned}\right\} \tag{17}$$

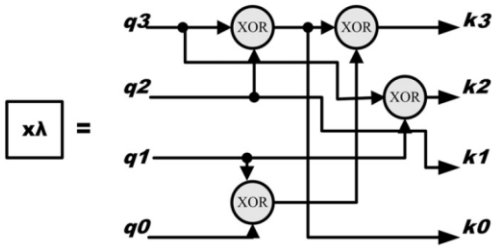


Fig. 4. Multiplication with constant, λ.

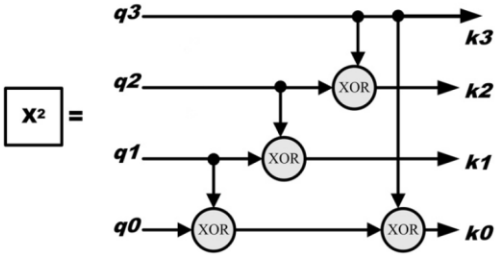


Fig. 5. Squaring in $GF(2^4)$.

Fig. 4 illustrates the hardware implementation for Eq. (17).

3.2. Squaring nibbles

For squaring the nibble $q=\{q_3q_2q_1q_0\}_2$ $GF(2^4)$, let $k=\{k_3k_2k_1k_0\}_2$ be the squared nibble in $GF(2^4)$. Then, q and k can be expressed as extensions over the $GF(2^2)$, so that, $k=\{k_3k_2\}_2x+\{k_1k_0\}_2=k_Hx+k_L$ where, $k_H=\{k_3k_2\}_2$ and $k_L=\{k_1k_0\}_2$. Also, $q=\{q_3q_2\}_2x+\{q_1q_0\}_2=q_Hx+q_L$ where, $q_H=\{q_3q_2\}_2$ and $q_L=\{q_1q_0\}_2$. Here, k_H, k_L, q_H, *and* q_L are $GF(2^2)$. Then, $k=(q_Hx+q_L)^2=q_H^2x^2+q_L^2$. Next, by substituting $x^2=x+\varphi$, the new equation for k is given by, $k=q_H^2(x+\varphi)+q_L^2=q_H^2x+q_H^2\varphi+q_L^2=k_Hx+k_L$ *where,* $k_H=q_H^2$, $k_L=(q_H^2\varphi+q_L^2)$ are $\in GF(2^2)$. Now, expressing k_H, k_L, q_H, q_L and φ over $GF(2)$, we have, $k_H=(q_3x+q_2)^2=q_3x^2+q_2$. Next, making the substitution $x^2=x+1$, $k_H=q_3(x+1)+q_2=q_3x+(q_2+q_3)=k_3x+k_2$. Similarly, we have, $k_L=(q_3x+q_2)^2(1x+0)+(q_1x+q_0)^2=(q_3^2x^2+q_2^2)x+q_1^2x^2+q_0^2=q_3x^3+q_2x+q_1x^2+q_0$. By substituting the x^3 term in the equation with 1, and, x^2 by $x+1$, $k_L=q_3+q_2x+q_1(x+1)+q_0=q_3+q_2x+q_1x+q_1+q_0=(q_2+q_1)\ \ x+(q_3+q_1+q_0)=$

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

k_1x+k_0. Hence finally the squared nibble k is as follows:

$$\left.\begin{aligned} k_3 &= q_3 \\ k_2 &= (q_2+q_3) \\ k_1 &= (q_1+q_2) \\ k_0 &= (q_0+q_1+q_3) \end{aligned}\right\} \quad (18)$$

The logic implementation of the squaring function is shown in Fig. 5.

3.3. Multiplication of nibbles in GF (2^4)

For multiplication of nibble, $q=\{q_3\ q_2\ q_1\ q_0\}_2$ with the nibble, $w=\{w_3\ w_2\ w_1\ w_0\}_2$ in $GF\ (2^4)$, let $k=\{k_3k_2\ k_1\ k_0\}_2$ be the product nibble in $GF\ (2^4)$. Then, q, w and k can be expressed as extensions over the $GF\ (2^2)$, so that, $k=\{k_3k_2\}_2x+\ k_1\ k_0\}_2=k_Hx+k_L$ where, $k_H=\{k_3\ k_2\}_2$ and $k_L=\{k_1\ k_0\}_2$. Also, $q=\{q_3\ q_2\}_2x+\ \{q_1\ q_0\}_2=q_Hx+q_L$ where, $q_H=\{q_3\ q_2\}_2$ and $q_L=\{q_1\ q_0\}_2$. In addition, $w=\{w_3w_2\}_2x+\{w_1w_0\}_2=w_Hx+w_L$ where, $w_H=\{w_3w_2\}_2$ and $w_L=\{w_1w_0\}_2$. Here, k_H, k_L, q_H, q_L, w_H and w_L are $GF\ (2^2)$. Then $k=(q_Hx+q_L)(w_Hx+w_L)=(q_Hw_Hx^2+q_Hw_Lx+q_Lw_Hx+q_Lw_L)=q_H\ w_Hx^2+(q_Hw_L+q_Lw_H)x+q_Lw_L$. *Now* by substituting x^2 with $x+\varphi$, we have, $k=q_Hw_H(x+\varphi)+(q_Hw_L+q_Lw_H)\ x+q_Lw_L=(q_Hw_H+q_Hw_L+q_Lw_H)x+q_Hw_H\varphi+q_Lw_L=(q_Hw_H+q_Hw_L+q_Lw_H+q_Lw_L)\ x+q_Hw_H\varphi+q_Lw_L=\{(q_H+q_L)\ (w_H+w_L)+q_Lw_L\}x+(q_Hw_H\varphi+q_Lw_L)=k_Hx+k_L$, where, $k_H=\{(q_H+q_L)(w_H+w_L)+q_Lw_L\}$ and $k_L=(q_Hw_H\ \varphi+q_Lw_L)$. The product in $GF\ (2^4)$ is then expressed as

$$\left.\begin{aligned} k_H &= (q_H+q_L)(w_H+w_L)+q_Lw_L \\ k_L &= (q_Hw_H\phi+q_Lw_L) \end{aligned}\right\} \quad (19)$$

Fig. 6 shows the hardware implementation of Eq. (19) for the multiplication of variable nibbles in $GF\ (2^4)$.

3.4. Multiplication of bit-pairs in GF (2^2)

Let $k=qw$, where bit-pairs $k=\{k_1\ k_0\}_2$, $q=\{q_1\ q_0\}_2$ and $w=\{w_1\ w_0\}_2$ are elements in $GF\ (2^2)$. Now expressing k, q and w over $GF\ (2)$, we have, $k=k_1x+k_0=(q_1x+q_0)\ (w_1x+w_0)=q_1\ w_1x^2+q_0\ w_1x+q_1w_0x+q_0w_0$. Using the irreducible polynomial substitution, $x^2=x+1$, $k=q_1w_1(x+1)+q_0w_1x+q_1w_0x+q_0w_0=(q_1w_1+q_0w_1+q_1w_0)x+(q_1w_1+q_0w_0)=k_1x+k_0$. Hence, the product bits are as follows:

$$\left.\begin{aligned} k_1 &= (q_1w_1+q_0w_1+q_1w_0) \\ k_0 &= (q_1w_1+q_0w_0) \end{aligned}\right\} \quad (20)$$

The hardware implementation for Eq. (20) is shown in Fig. 7.

3.5. Multiplication of bit-pairs with constant φ in GF (2^2)

Let $k=q\varphi$, where the bit-pairs $k=\{k_1k_0\}_2$, $q=\{q_1q_0\}_2$ and $\varphi=\{10\}_2$ are elements in $GF\ (2^2)$. Now expressing k, q and φ over $GF\ (2)$, we have, $k=k_1x+k_0=(q_1x+q_0)(x+0)=q_1x^2+q_0x$. Next, making the substitution, $x^2=x+1$, $k=q_1(x+1)+q_0x=(q_1+q_0)x+q_1=k_1x+k_0$. Hence,

$$\left.\begin{aligned} k_1 &= (q_1+q_0) \\ k_0 &= q_1 \end{aligned}\right\} \quad (21)$$

The circuit diagram for Eq. (21) is shown in Fig. 8.

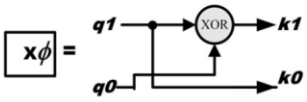


Fig. 8. Multiplication with constant, φ.

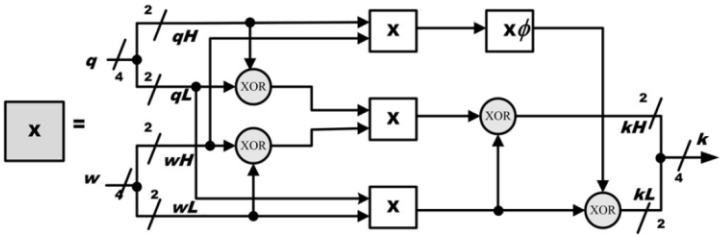


Fig. 6. Multiplication in $GF\ (2^4)$.

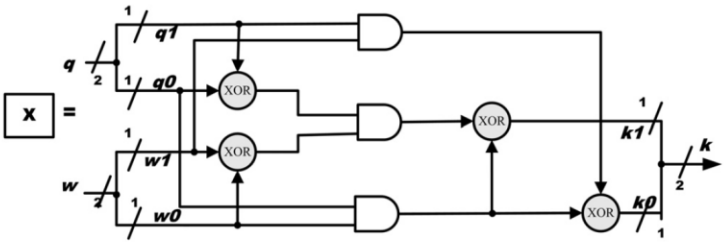


Fig. 7. Multiplication in $GF\ (2^2)$.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

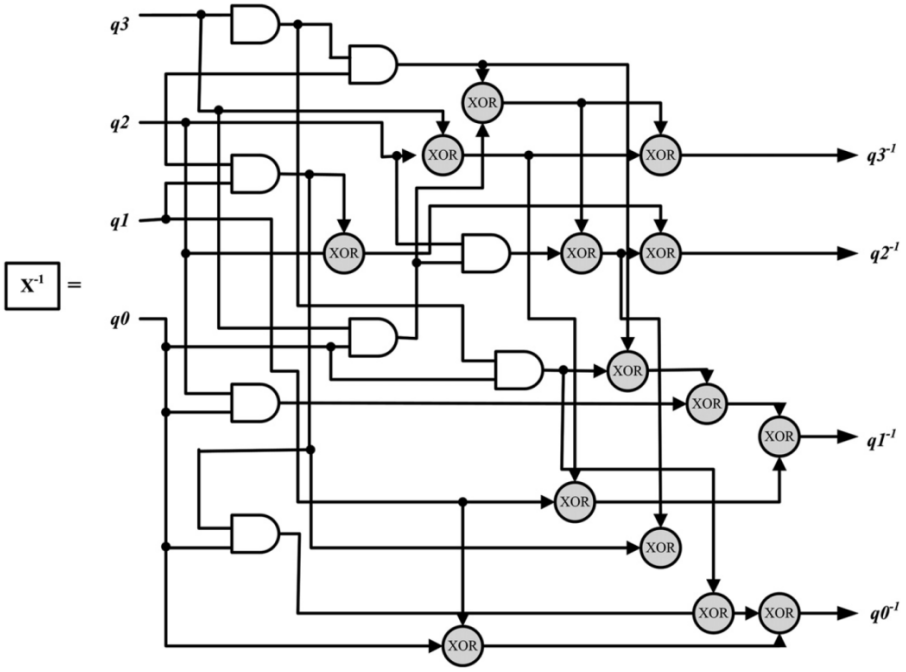


Fig. 9. Multiplicative Inversion in $GF(2^4)$.

3.6. Multiplicative inverse of nibble in GF (2^4)

Based on [5], the multiplicative inverse in $GF(2^4)$ of nibble $q=\{q_3\ q_2\ q_1\ q_0\}_2$ is given by $q^{-1}=\{q_3{}^{-1}q_2{}^{-1}q_1{}^{-1}q_0{}^{-1}\}_2$ where,

$$\left.\begin{aligned} q_3^{-1} &= q_3+q_3q_2q_1+q_3q_0+q_2\\ q_2^{-1} &= q_3q_2q_1+q_3q_2q_0+q_3q_0+q_2+q_2q_1\\ q_1^{-1} &= q_3+q_3q_2q_1+q_3q_1q_0+q_2+q_2q_0+q_1\\ q_0^{-1} &= q_3q_2q_1+q_3q_2q_0+q_3q_1+q_3q_1q_0\\ &\quad +q_3q_0+q_2+q_2q_1+q_2q_1q_0+q_1+q_0 \end{aligned}\right\} \quad (22)$$

Fig. 9 illustrates the hardware implementation of Eq. (22).

4. Novel XOR gate for low power CMOS Galois field arithmetic

From the above Galois Field arithmetic for S-Box and the corresponding Figs. 3–9 it is clearly evident that the implementation of S-Box/Inv S-Box requires a large number of XOR operations whose efficient and low power implementation can result in a substantially improved CMOS S-Box hardware design.

4.1. Novel XOR topology

Over the years, various 2-input XOR gate designs have been reported to enhance the performance for various applications [20–29]. We previously reported in Ref. [19] an XOR gate using six transistors which provides full output swing for all input combinations and enables low voltage operation with small propagation delay. Here we propose a further low-power constrained novel 2-input XOR gate using six transistors including an inverter.

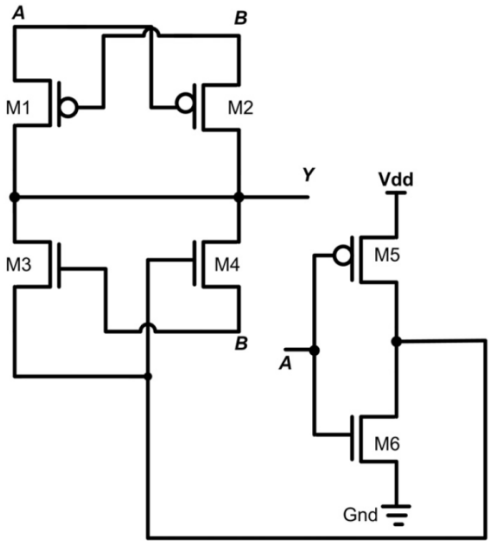


Fig. 10. The proposed XOR circuit for low-power Galois Field arithmetic.

The proposed XOR circuit is based on the concept of pass transistor logic and inverter for complementary input. Pass transistor design enables small transistor count along with smaller input loads

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

(with signal input to source/drain instead of gate) offering very low-power operation with high-performance. This is because signal input at the source/drain requires the charging and discharging of the source/drain-to-body depletion capacitance which is usually smaller than the gate-to-source parallel plate capacitance in case of signal inputs at the gate. Since a nMOS device passes a strong '0', but a weak '1' while a pMOS device passes a strong '1', but a weak '0', the complementary pass transistors are organized to pass a strong output logic level for all input combinations of '1' and '0'. Fig. 10 shows the schematic of the proposed XOR circuit. It performs a perfect full swing operation for every input pattern. The output Y generates '0' corresponding to $A=B=0$. For this condition, transistors M1, M2 and M4 are ON, but the transistor M4 will pass a strong '0' to the output Y. When $A=0$, $B=1$, transistors M2, M3 and M4 are ON and a '1' is generated at the output Y with transistor M2 passing the strong '1'. For $A=1$, $B=0$, only the device M1 is ON and a strong '1' is passed to the output Y. With $A=B=1$, only transistor M3 is ON and a strong '0' is passed to the output. This pass transistor XOR thus does not suffer from signal level deteriorations like other pass-transistor XOR gates. The transistor sizes are carefully chosen for optimal power-delay performance under various operating conditions.

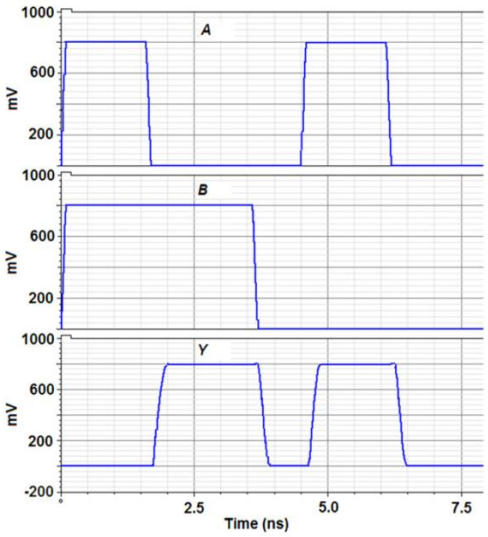


Fig. 11. Full-swing logic of the novel pass-transistor XOR gate for Galois Field arithmetic.

4.2. Performance in 65 nm CMOS

The XOR circuit was implemented in 65 nm IBM CMOS technology. Fig. 11 shows the simulated full-swing operation of this pass-transistor XOR gate verified by Cadence Spectre while its Virtuoso layout is shown in Fig. 12 which was parasitic extracted, back annotated and verified using Cadence Assura. Several previous XOR circuit topologies were also simulated and compared with the proposed new design. The simulations were carried out using the same testing condition on Cadence Spectre platform to measure the propagation delay and the power dissipation in each case. All the simulations were carried out using the same 65 nm IBM CMOS technology with a 0.6–1.2 V supply voltage range, a load capacitance of 50 fF, and, a throughput (clocking) rate of 500 MHz. Table 1 summarizes the results of these simulations which shows the propagation delay, the power dissipation and the power delay product (PDP) for various topologies. From the simulation results, it is clear that the proposed new XOR gate has the lowest propagation delay as well as the lowest power consumption. Hence, the new XOR gate is optimally suitable for implementing CMOS Galois Field arithmetic for the S-Box/Inv S-Box.

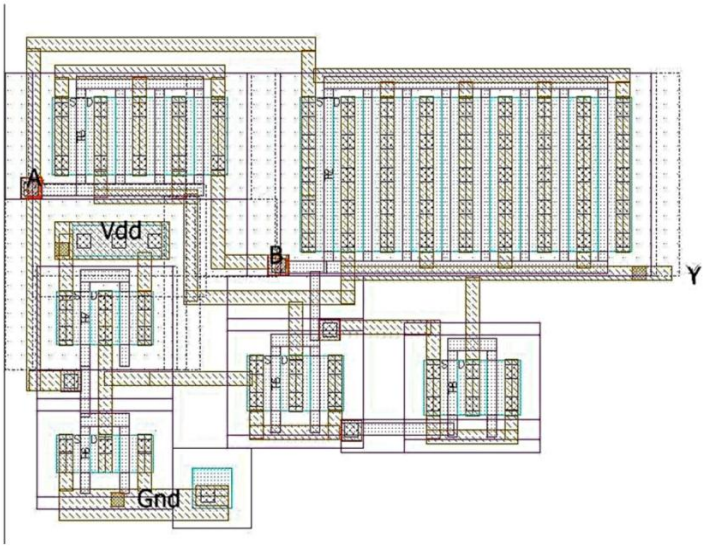


Fig. 12. Layout of the 2-input XOR gate.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

5. Compact S-Box/Inv S-Box chip and comparison with other designs

The Hardware architecture in Fig. 2 is implemented to perform both encryption and decryption with S-Box and Inv S-Box sharing the same hardware. It is an improved modification of the architecture in Ref. [32] along with this inclusion of the inverse S-Box which was not implemented in Ref. [32]. This modification enables the implementation of Inverse SubBytes for decryption by reusing the same S-Box resources. Using the novel low-power and

Table 1
Comparison of simulation results for the XOR gate.

V(v)	Proposed 6T	6T [19]	4T [25]	3T [29]	6T [25]
Delay (ns)					
0.6	1.69	2.81	2.656	3.08	5.006
0.8	1.585	2.75	2.994	2.061	3.251
1.0	1.562	2.642	2.094	2.045	2.111
1.2	1.553	2.099	2.068	2.035	2.075
Average power (fW)					
0.6	1.82	2.312	4.189	8.703	14.66
0.8	3.256	6.069	7.613	16.25	89.57
1.0	15.015	22.77	12.12	25.36	238.9
1.2	17.144	25	18.81	34.96	463.3
PDP (yJ)					
0.6	3.076	6.497	11.13	26.81	73.36
0.8	5.161	16.679	22.79	33.49	291.2
1.0	23.453	60.158	25.38	51.86	504.2
1.2	26.625	52.475	38.91	71.15	961.5

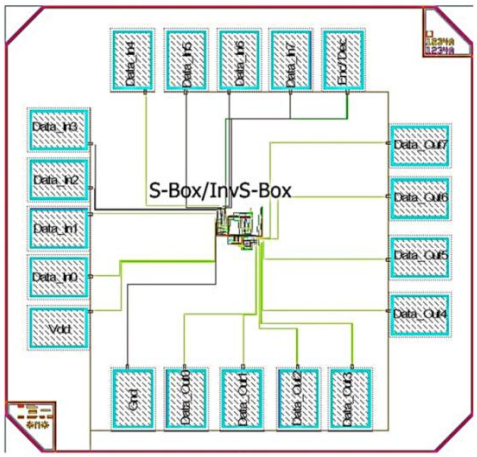


Fig. 14. Complete chip die of the S-Box/Inv S-Box with bonding pads.

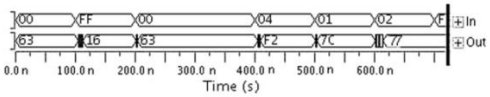


Fig. 15. S-Box functional test verification of the SubBytes operation.

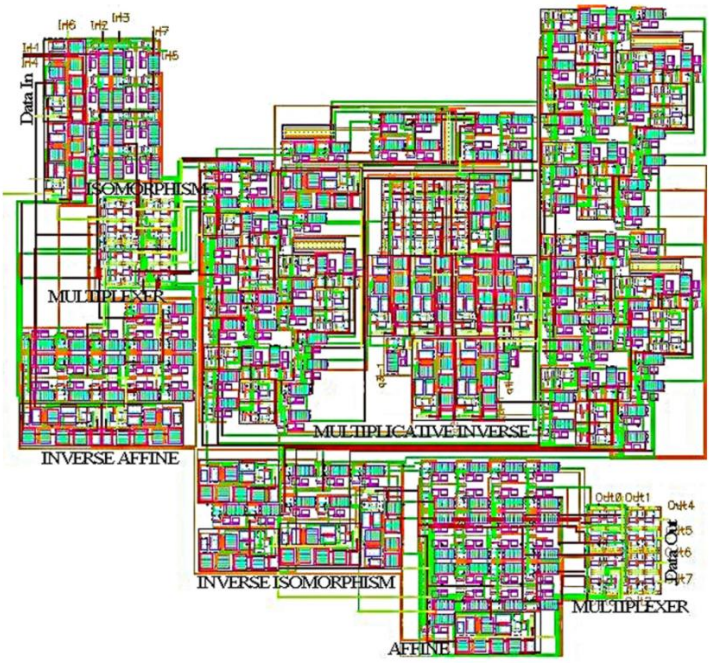


Fig. 13. Complete layout of the S-Box/Inv S-Box.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

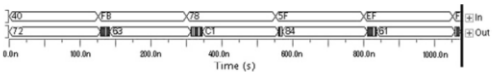


Fig. 16. Inv S-Box functional test verification of the Inverse SubBytes operation.

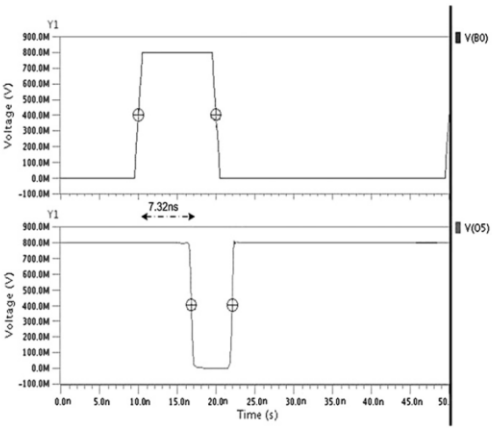


Fig. 17. Worst case propagation delay of the full-custom CMOS S-Box.

Table 2
S-box/INV S-box logic gate and device count.

Operation	Num. of XOR (6T) per operation	Num. of AND (6T) per operation	Num. of operations in the system	Total num. of XOR gates	TOTAL num. of AND gates
Affine	12	–	1	12	–
Inv Affine	12	–	1	12	–
Iso	12	–	1	12	–
Inv Iso	12	–	1	12	–
Mul_con, λ	4	–	1	4	–
Squaring	4	–	1	4	–
Mul_GF4	4	–	3	12	–
Mul_GF2	4	3	9	36	27
Mul_con, Ø	1	–	3	3	–
Inverse	14	8	1	14	8
Addition	1	–	2	2	–
				123	35

low-area XOR gate of the previous section, a circuit level optimized S-Box/Inv S-Box hardware in terms of silicon area and power dissipation has been achieved. The complete circuit simulation, optimization, layout and parasitic extraction were carried out using Cadence tools. The mask layout of the S-Box/Inv S-Box illustrated in Fig. 13 was customized (with manual *placement* and *routing*) in 65 nm IBM CMOS with copious instances of the novel XOR gate of Fig. 12 for the CMOS Galois Field/Composite Field arithmetic. Minimum channel length is used for all the devices and optimum channel width is carefully chosen for each device to achieve verified functionality with low power dissipation and smallest possible propagation delay. Fig. 14 shows the complete S-Box/Inv S-Box chip with the bonding I/O pads. The design simulations consists of functional verification, power and timing analysis, as well as, design rule checking (DRC) and layout vs. schematic (LVS). Simulation results verified the correct functionality for every input SubByte combination with a supply voltage of only 0.8 V. Figs. 15 and 16 displays respectively the correct SubBytes and Inverse SubBytes functional operations of the chip for several computation cycles. Fig. 17 indicates the worst case S-Box input-to-output delay of around 7.32 ns thus allowing a throughput of around 130 Mega-SubBytes per second. The silicon-area of the S-Box/Inv S-Box was only 288 μm^2 using the 65 nm CMOS process, and, offers to-date the smallest reported silicon-area of any implementation with shared S-Box and Inverse S-Box. Table 2 shows the list and count of various logic gates (and transistors) for the different blocks of the S-Box/Inv S-Box implementation, indicating the large number of XORs dominating the transistor budget with a total chip device count of 948 (excluding bonding pad area). Table 3 shows a comparison of the proposed implementation results with other recent S-Box designs, including the normal basis design in Ref. [45] which is also discussed in Ref. [46]. It can be seen that the proposed implementation achieves the lowest core area through the circuit level optimization. Extensive performance factors are considered including additional criterion such as power-delay product (PDP), energy-delay product (EDP) and power-area product (PAP) in comparing the different S-Box circuits. As shown in Table 3, the proposed design with 0.09 μW (@ 125 MHz) has the second lowest power dissipation when compared to the 65 nm CMOS S-Box in Ref. [18] with 0.04 μW (@ 10 MHz), but it has lower delay time than this design in Ref. [18]. However, the S-Box in Ref. [18] is only designed for SubBytes transformation, whereas the proposed design includes both the transformations with S-Box and Inverse S-Box sharing the same hardware. In Ref. [18] the area of the design was not mentioned, so the PAP calculation for Ref. [18] was not available to compare with our lowest PAP result in the table of 0.0259 mW μm^2 (@ 125 MHz). Also, our design

Table 3
AES S-Box performance and comparison with previous work.

	Proposed Sbox/Inv Sbox	Polynomial basis [32,46]	Normal basis [45,46]	Sbox [18]	Sbox [18]	Sbox [17]	Sbox [16]	Composite field [3]	PPRM shared Sbox [6]
Technology (nm)	65	65	65	130	65	250	250	130	130
Year	2012	2012	2010	2009	2009	2007	2007	2001	2003
Vdd (V)	0.8	–	–	1.2	0.8	2.5	2.5	1.5	1.5
Decryption	yes	no	no	no	no	no	no	yes	yes
Frequency (MHz)	125	763	610	10	10	10	10	10	10
Delay (ns)	7.322	1.31	1.64	3.3	7.5	4.39	9	2.53	2.0
Average power (μW)	0.09	54.99	44.39	12.1	0.037	207.3	140	179	79
PDP (fJ)	0.659	72.04	72.8	39.93	0.278	910.05	1260	452.87	158
Area (μm^2)	288	525.2	403.2	–	–	10,791	8744	381 GE*	725 GE*
EDP (yJ s)	4.825	94.37	119.39	131.77	2.085	3995.12	11,340	1145.8	316
PAP (mW.μm^2)	0.0259	28.88	17.89	–	–	2236.97	1224	–	–
Throughput (Gbps)	1.0	6.1	4.9	0.08	0.08	–	–	–	–

* GE implies Gate Equivalent.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

achieved the best PAP, EDP and PDP values when compared to all the other designs including the normal basis design in Ref. [45] (@ 610 MHz). In addition, it is to be noted that the design in Ref. [45] does not consider the implementation of the inverse S-Box for decryption. When the same frequency (cycles/sec) of operation is considered for all the designs in the Table 3, the throughput rate (Gbps) is the same for all the cases. This is because one 8-bit SubBytes output data is available per cycle. Hence, for example, if the frequency is reduced to a low value of 5 MHz, the throughput will be 0.04 Gbps for all the designs in Table 3. However, at the same reduced frequency, the proposed design will have a significantly reduced power dissipation (with reduced logic switching) and will consequently be the most energy efficient design in the table.

6. Conclusion

This paper presents a full custom hardware implementation of low power AES S-Box/Inv S-Box architecture in the 65 nm CMOS process employing circuit level optimization. The design demonstrated a new approach to minimize silicon-area of S-Box design by using a new 2-input XOR gate for low-power composite field arithmetic in order to reduce the power dissipation and delay for the overall circuit. The results indicate that our design is suitable for applications which require small area and low power consumption such as RFID tags.

Acknowledgment

The authors wishes to acknowledge the anonymous reviewers for their comments which helped in enhancing the quality of the paper. Acknowledgment is also due to Dr. Shaun Cooper of the Institute of Information and Mathematical Sciences for discussions on Galois Field arithmetic.

References

[1] J. Daemen, V. Rijmen, The Design of Rijndael, Springer-Verlag, New York, 2002, Inc.
[2] S. Tillich, M. Feldhofer, T. Popp, J. Groszchaedl, Area, delay, and power characteristics of standard-cell implementations of the AES S-Box, Journal of Signal Processing Systems 50 (2) (2008) 251–261.
[3] A. Satoh, S. Morioka, K. Takano, S. Munetoh, A compact Rijndael hardware architecture with S-Box optimization, ASIACRYPT 2001, Lecture Notes in Computer Science 2248 (2001) 239–254.
[4] N. Sklavos, O. Koufopavlou, Architectures and VLSI implementations of the AES-proposal Rijndael, IEEE Transactions on Computers 51 (2002) 1454–1459.
[5] X. Zhang, K.K. Parhi, High-speed VLSI architectures for the AES algorithm, IEEE Transactions on VLSI Systems 12 (2004) 957–967.
[6] S. Morioka, A. Satoh, An optimized S-box circuit architecture for low power AES design, in: Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002), 2002, 2523 (2003) 172–186.
[7] R.E. Bryant, Graph-Based Algorithms for Boolean Function Manipulation, IEEE Transactions on Computers C-35 (8) (1986) 677–691.
[8] S. Morioka, A. Satoh, A 10-Gbps Full-AES crypto design with a twisted BDD S-Box architecture, IEEE Transactions on VLSI Systems 12 (7) (2004) 98–103.
[9] N. Ahmad, R. Hasan, W.M. Jubadi, Design of AES S-box using combinational logic optimization, in: Proceedings of the IEEE International Symposium on Industrial Electronics and Applications (ISIEA 2010), 2010, 696–699.
[10] R.R. Rach, P.V. Ananda Mohan, Implementation of AES S-Boxes using combinational logic, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2008), 2008 3294–3297.
[11] N. Chen, Z. Yan, High-performance designs of AES transformations, in: Proceedings of the International Symposium on Circuits and Systems 2009 (ISCAS 2009), 2009 2906–2909.
[12] C. Nalini, P.V. Anandmohan, D.V. Poomaiah, V.D. Kulkarni, Compact designs of SubBytes and MixColumn for AES, in: Proceedings of the IEEE International Advance Computing Conference (IACC 2009), 2009 1241–1247.
[13] V. Rijmen, Efficient implementation of the Rijndael S-Box, 2000. Available from: 〈http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf〉.
[14] R. Liu, K.K. Parhi, Fast composite field S-box architectures for advanced encryption standard, in: Proceedings of the ACM Great Lakes Symposium on VLSI 2008 (GLSVLSI '08), 2008 65–70.
[15] D. Canright, A. Very, Compact S-Box for AES, workshop on cryptographic hardware and embedded systems 2005 (CHES 2005), Lecture Notes in Computer Science 3659 (2005) 441–455.
[16] L. Zhenglin, Z. Yonghong, Z. Xuecheng, H. Yu, C. Yicheng, A high security and low-power AES S-Box full-custom design for wireless sensor network, Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing (2007) 2499–2502.
[17] Y.H. Zeng, X.C. Zou, Z.L. Liu, J.M. Lei, Low-power clock-less hardware implementation of the Rijndael S-box for wireless sensor networks, The Journal of China Universities of Posts and Telecommunications 14 (4) (2007) 104–109.
[18] D. Kamel, F.X. Standaert, D. Flandre, Scaling trends of the AES S-box lower power consumption in 130 and 65 nm CMOS Technology Nodes, in: Proceedings of the International Symposium on Circuits and Systems 2009 (ISCAS 2009), 2009 1385–1388.
[19] N. Ahmad, R. Hasan, Design of XOR gates in VLSI implementation, in: Proceedings of the Electronic New Zealand Conference (ENZCON 2010), 2010, 51–55.
[20] H.T. Bui, Y. Wang, Y. Jiang, Design and analysis of low-power 10-transistor full adders using novel XOR-XNOR gates, IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 49 (1) (2002) 25–30.
[21] A.M. Shams, T.K. Darwish, M.A. Bayoumi, Performance analysis of low-power 1-bit CMOS full adder cells, IEEE Transactions on VLSI Systems 10 (1) (2002) 20–29.
[22] D. Radhakrishnan, Low-voltage low-power CMOS full adder, IET Proceedings on Circuits, Devices and Systems 148 (1) (2001) 19–24.
[23] H. Lee, G.E. Sobelman, New XOR/XNOR and full adder circuits for low voltage, low power applications, Microelectronics Journal 29 (8) (1998) 509–517.
[24] H. Lee, G.E. Sobelman, New low-voltage circuits for XOR and XNOR, Proceedings of the IEEE (1997) 225–229, Southeastcon 97.
[25] J.M. Wang, S.C. Fang, W.S. Feng, New efficient designs for XOR and XNOR functions on the transistor level, IEEE Journal of Solid-State Circuits 29 (7) (1994) 780–786.
[26] H.T. Bui, A.K. Al-Sheraidah, Y. Wang, New 4-transistor XOR and XNOR designs, in: Proceedings of the IEEE Asia Pacific Conference on ASICs (AP-ASIC 2000), 2000 25–28.
[27] N. Weste, K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, Addison-Wesley Longman Publishing Co. Inc, 1993.
[28] D. Wang, M. Yang, W. Cheng, X. Guan, Z. Zhu, Y. Yang, Novel low power full adder cells in 180 nm CMOS technology, in: Proceedings of the 4th Industrial Electronics and Applications (ICIEA2009), 2009 430–433.
[29] A.B. Chowdhury, A. Roy, H. Saha, A high speed 8 transistor full adder design using novel 3 transistor XOR gates, International Journal of Electronics, Circuits and Systems 2 (4) (2008) 217–223.
[30] A. Rudra, P.K. Dubey, C.S. Jutla, V. Kumar, J.R. Rao, P. Rohatgi, Efficient Rijndael encryption implementation with composite field arithmetic, in: Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems (CHES2001), 2001 175–188.
[31] J. Wolkerstorfer, E. Oswald, M. Lamberger, An ASIC implementation of the AES S-Boxes, in: Proceedings of The Cryptographer's Track at the RSA Conference on Topics in Cryptology, 2271, 2002 67–78.
[32] N. Mentens, L. Batina, B. Preneel, I. Verbauwhede, A systematic evaluation of compact hardware implementations for the Rijndael S-Box, CTRSA, Lecture Notes in Computer Science 3376 (2005) 323–333.
[33] G. Bertoni, M. Macchetti, L. Negri, Power-efficient ASIC synthesis of cryptographic S-boxes, in: Proceedings of the ACM Great Lakes Symposium on VLSI 2004 (GLSVLSI'04), 2004 277–281.
[34] F. Burns, J. Murphy, A. Koelmans, A. Yakovlev, Efficient advanced encryption standard implementation using lookup and normal basis, IET Journals of Computers and Digital Techniques 3 (2009) 270–280.
[35] S. Nikova, V. Rijmen, M. Schläffer, Using normal bases for compact hardware implementations of the AES S-Box, in: Proceedings of the 6th International Conference on Security and Cryptography for Networks (SCN '08), 2008 236–245.
[36] L. Li, J. Han, X. Zeng, J. Zhao, A. Full-custom Design of AES SubByte module with signal independent power consumption, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS 2008), 2008 3302–3305.
[37] K. Martin, Digital Integrated Circuit Design, Oxford University Press, USA, 2000.
[38] T. Good, M. Benaissa, 692-nW Advanced Encryption Standard (AES) on a 0.13-μm CMOS, IEEE Transactions on VLSI Systems 18 (12) (2009) 1753–1757.
[39] C.H. Wang, C.L. Chuang, C.W. Wu, An efficient multi-mode multiplier supporting AES and fundamental operations of public-key cryptosystems, IEEE Transactions on VLSI Systems 18 (4) (2010) 553–563.
[40] K. Gaj, P. Chodowiec, FPGA and ASIC Implementation of AES, in Cryptographic Engineering, Springer, 2008, pp. 235–294.
[41] S. Mangard, M. Aigner, S. Dominikus, A highly regular and scalable AES hardware architecture, IEEE Transactions on Computers 52 (4) (2003) 483–491.
[42] K.H. Boey, P. Hodgers, Y. Lu, M. O'Neill, R. Woods, Security of AES Sbox designs to power analysis, in: Proceedings of the 17th IEEE International Conference on Electronics, Circuits and Systems (ICECS), 2010 1232–1235.
[43] IEEE ITRS Technology Roadmap, 〈http://public.itrs.net〉.
[44] A. Agarwal, S. Mukhopadhyay, A. Raychowdhury, K. Roy, C.H. Kim, Leakage power analysis and reduction for nanoscale circuits, IEEE Micro 26 (2) (2006) 68–80.
[45] J. Boyar, R. Peralta, A new combinational logic minimization technique with applications to cryptology, Proceedings of the SEA (2010) 178–189.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

[46] M.M. Kermani, A.R. Masoleh, Efficient and high-performance parallel hardware architectures for the AES-GCM, IEEE Transactions on Computers. http://dx.doi.org/10.1109/TC.2011.125.

Nabihah Ahmad received the B.S. in electrical, electronic and system engineering from Universiti Kebangsaan Malaysia (UKM) and M.S. degrees in electronic engineering from Universiti Tun Hussein Onn Malaysia (UTHM) in 2002 and 2006, respectively. She is currently a Ph.D. candidate with the Center for Research in Analog and VLSI Microsystem Design at School of Engineering and Advanced Technology, Massey University, New Zealand. Her research interests include low power VLSI circuit design, cryptography algorithms, and architectures for low power digital system.

S. M. Rezaul Hasan received his Ph.D. in Electronics Engineering from the University of California Los Angeles (UCLA) in 1985. From 1983 to 1986 he was a VLSI design engineer at Xerox Microelectronics Center in El Segundo, CA., where he worked in the design of CMOS VLSI microprocessors. In 1986 he moved to the Asia-Pacific region and served several institutions including Nanyang Technological University, Singapore (1986–1988), Curtin University of Technology, Perth, Western Australia (1990–1991) and University Sains Malaysia, Perak, Malaysia (1992–2000). At University Sains Malaysia he held the position of Associate Professor and was the coordinator of the Analog and VLSI research laboratory. He spent the next four years (2000–2004) in the West Asia-Gulf region where he served as an Associate Professor of Microelectronics, Integrated Circuit Design and VLSI Design in the Department of Electrical and Computer Engineering at the University of Sharjah, Sharjah, United Arab Emirates. While in Sharjah he received the National Bank of Sharjah Award for outstanding research publication in Integrated Circuit Design. Presently he is the Director of the Center for Research in Analog and VLSI microsystems dEsign (CRAVE) at Massey University, Auckland, New Zealand. He is also a senior faculty member within the School of Engineering and Advanced Technology (SEAT) in Electronics and Computer Engineering, teaching courses in Advanced Microelectronics and Integrated Circuit Design. He has published over 138 papers in international journals and conferences in the areas of Analog, Digital, RF and Mixed-Signal Integrated Circuit Design and VLSI Design. Dr. Hasan has also served as a consultant for many electronics companies. His present areas of interest include Analog and RF Integrated Circuit and Microsystem Design, VLSI signal processing, CMOS sensors, CMOS Bioelectronics and Biological (*gene-protein*) Circuit Design. He is a senior member of the IEEE and an editor of the Hindawi journal of active and passive electronic components.

Please cite this article as: N. Ahmad, S.M. Rezaul HasanLow-power compact composite field AES S-Box/Inv S-Box design in 65 nm CMOS using Novel XOR Gate, INTEGRATION, the VLSI journal (2012), http://dx.doi.org/10.1016/j.vlsi.2012.06.002

IEEE-ICSE2012 Proc., 2012, Kuala Lumpur, Malaysia

Topology of 2 input subnanowatt XOR gate in 65nm CMOS technology

Nabihah Ahmad and Rezaul Hasan
Faculty of Electrical and Electronic, Universiti Tun Hussein Onn Malaysia
Batu Pahat, Johor, Malaysia
School of Engineering and Advanced Technology, Massey University
Auckland, New Zealand
nabihah@uthm.edu.my, hasanmic@massey.ac.nz

***Abstract*- Exclusive OR (XOR) gate is highly utilized in various digital system applications such as full adder, comparator, parity generator and encryption processor, which leads to increased in the interests to enhance the performance of XOR gate. A novel design of low power and high performance XOR gate using six transistors application are proposed in this paper. The new XOR gate has been compared with previous design in term of power, delay and power-delay product (PDP). The XOR gate is simulated using Cadence Spectre with 65nm Complementary Metal Oxide Semiconductor (CMOS) technology at different supply voltages with a range of 0.6V to 1.2V. The area of the core circuit is approximately 48 µm2. The critical path propagation delay is 1.585 ns with power dissipation of only around 3.256 fW at 0.8V supply voltage .The results demonstrate that the proposed design achieve a lowest power consumption and high speed with respect to other designs.**

Keywords-XOR gate, low power, VLSI

I. INTRODUCTION

With the growth of integrated circuit towards very high integration density and high operating frequencies, it is crucial to design a logic circuit that can contribute a low power and high speed to overall system. Low power design has become major design forethought with the enormous growth of portable applications. Various low power techniques have been explored and one of it is improving the circuit level including enhancing the basic gate such as XOR gate which influence the overall power system. Optimization of XOR circuit design to achieve low power, small size and delay is needed due to the important role played by XOR gate in various circuits especially in arithmetic circuits such as full adder, parity generator and cryptography circuits. As XOR gate is use in a critical path in most of the system, it significantly affects the overall performance of larger and complex circuits.

Over the years, various 2-input XOR gate designs have been reported to enhance the performance for various applications [3-15]. By designing XOR gate, consideration must be made in power consumption and delay in the critical path and full output voltage swing with low number of transistors to implement it. One of the useful methods to reduce power consumption is by voltage scaling. In the other hand, reducing the voltage will reduce the speed. Another effective method is by avoiding non-full swing nodes connected to the gate of the other transistors especially to the input of inverters.

In this paper, we present a novel low-power design for 2-input XOR gate using six transistors implemented using 65nm CMOS technology. The paper is organised as follows: in Section II, previous work is reviewed. Subsequently, in section III, the proposed design of XOR gate is presented. In section IV, the simulation results are given and discussed. The comparison and evaluation for proposed and existing designs are carried out. Finally a conclusion will be made in the last section.

II. PREVOIUS WORK

Traditional XOR gate design based on eight transistors static CMOS can operate with full output voltage swing with the drawback of acquires large amount of transistors [1]. XOR gate based on transmission gate [2] is used to overcome the signal degradation cause by the PMOS and NMOS. It offer a better quality but with the drawback of loss of driving capability and need complementary signal value to control gates of PMOS and NMOS which require more transistors and area. Cross-coupled (CC) XOR gate based on the pass transistor logic have been reported in [3] claims to have better speed and power consumption and works well under a lower supply voltage than six transistors XOR gate.

The XOR gate realization of the circuit using six transistors can be found in [4] after modifying their four by transistors XOR gate design cascading a standard inverter as a driving output. This design has improved the poor signal output for a certain input signal by using this topology. Powerless XOR gate (P-XOR) in [5] is proposed using the set of four transistors circuit with no power supply connection consumes less power than other design but with a large delay. Other four transistors XOR design was reported in [6] based on Gate-Diffusion-Input (GDI) cell. XOR gate with three transistors can be found in [7] using CMOS inverter and PMOS pass transistor. It provide less power-delay product but have a voltage degradation when the input A=1 and B=0. Elgamel et al. [8] also proposed three transistors XOR gate but consumes high power when A=1 and B=0 which produce poor logic '1'.

III. Proposed new XOR gate topology

We previously reported in [15] an XOR gate using six transistors which provides full output swing for all input combinations and enables low voltage operation with small propagation delay. Here we propose a further low-power constrained novel 2-input XOR gate using six transistors including inverter. The proposed XOR circuit is based on the concept of pass transistor logic and inverter for complementary input. Pass transistor design enables small transistor count along with smaller input loads (with signal input to source/drain instead of gate) offering very low-power operation with high-performance. Since a nMOS device passes a strong *'0'*, but a weak *'1'* while a pMOS device passes a strong *'1'*, but a weak *'0'*, the complementary pass transistors are organized to pass a strong output logic level for all input combinations of *'1'* and *'0'*.

Fig. 1 shows the circuit schematic of the proposed XOR circuit. It performs a perfect full swing operation for every input pattern. The output *Y* generates '*0*' corresponding to *A=B=0*. For this condition, transistor M1, M2 and M4 are ON, but the transistor M4 will pass a strong '*0*' to the output *Y*. When *A=0*, *B=1*, transistor M2, M3 and M4 are ON and a '*1*' is generated at the output *Y* with transistor M2 passing the strong '*1*'. For *A=1*, *B=0*, only the device M1 is ON and a strong '1' is passed to the output *Y*. With *A=B=1*, only transistor M3 is ON and a strong '*0*' is passed to the output. The transistor sizes are carefully chosen for optimal power-delay performance under various operating conditions. This pass transistor XOR thus does not suffer from signal level deteriorations like other pass-transistor XOR gates.

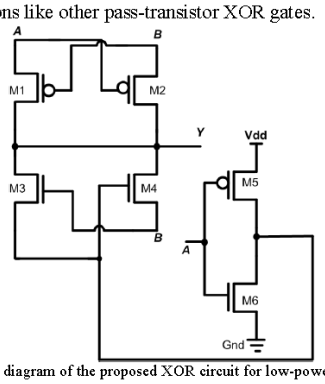


Fig. 1. Circuit diagram of the proposed XOR circuit for low-power application.

IV. Result and analyses

The complete circuit simulation, optimization, layout and parasitic extraction were carried out using Cadence tools. The mask layout of the XOR gate illustrated in Fig. 2 was customized (with manual *placement* and *routing*) in 65nm IBM CMOS. Minimum channel length is used for all the devices and optimum channel width is carefully chosen for each device to achieve verified functionality with low power dissipation and smallest possible propagation delay. The silicon-area of the XOR gate is approximately 48 μm^2 using the 65nm CMOS process.

Comparative studies and extensive simulation on the proposed XOR gate and five existed XOR gates found in literature have been realized using 65nm IBM CMOS technology in order to analyze the performance comparison with the proposed XOR gate. Fig. 3 displays the functional full-swing operations of the chip for several computation cycles verified by Cadence Spectre. The simulations were carried out using the same testing condition on Cadence Spectre platform to measure the propagation delay and the power dissipation in each case. All the simulation were carried out using the same 65nm IBM CMOS technology with a 0.6V to 1.2V supply voltage range, a load capacitance of 50fF, and, a throughput (clocking) rate of 500MHz. Propagation delay is evaluated from 50% of voltage level of input to 50% of voltage level of output. Power delay product (PDP) is calculated from production of worst case delay and average power consumption. In order to have closer analysis, several input pattern have been applied to cover all the input cases and simulation results verified the correct functionality for every input combination with a supply voltage of 0.8V. Table 1 summarizes the measured performance of the proposed XOR gate and provides a comparison of the circuit with recently reported designs.

From the simulation results, it is clear that the proposed new XOR gate has the lowest propagation delay as well as the lowest power consumption. The improvement attained by the proposed circuit is clearly evident when compared to these other XOR gate circuits. The worst circuit in term of speed is six transistor XOR gate in [4]. It has the highest propagation delay against voltage scaling. Power consumption increased as the supply voltage is increased. The proposed XOR circuit design worked successfully in low voltage supply of 0.8V with 3.256 fW which is the lowest power dissipation when compared to the six transistors in [4] and [15].

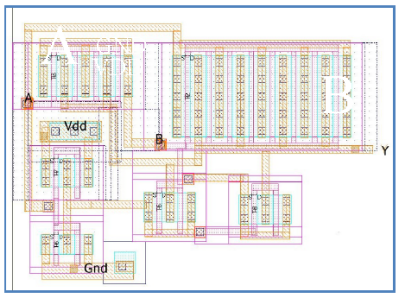


Fig. 2. Layout of the 2-input XOR gate.

TABLE I
SIMULATION RESULTS OF XOR GATE

	V(v)	Proposed 6T	6T[15]	4T [4]	3T [7]	6T [4]
Delay (ns)	0.6	1.69	2.81	2.66	3.08	5.01
	0.8	1.59	2.75	2.99	2.06	3.25
	1.0	1.56	2.64	2.09	2.05	2.11
	1.2	1.55	2.09	2.07	2.04	2.08
Average power	0.6	1.82	2.31	4.19	8.70	14.66
	0.8	3.26	6.07	7.61	16.25	89.57
(fW)	1.0	15.02	22.77	12.12	25.36	238.90
	1.2	17.14	25.00	18.81	34.96	463.30
PDP (yJ)	0.6	3.08	6.49	11.13	26.81	73.36
	0.8	5.16	16.68	22.79	33.49	291.20
	1	23.45	60.16	25.38	51.86	504.20
	1.2	26.63	52.48	38.91	71.15	961.50

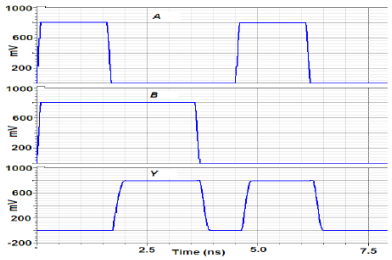


Fig. 3. Full-swing logic of the novel pass-transistor XOR gate

I. CONCLUSIONS

This paper presents a new 2-input XOR gate topology with a full-swing voltage output in 65nm CMOS process for low-power application in order to reduce the power dissipation and delay for the overall circuit. The performances of the proposed XOR gate was compared to other peer designs in various conditions and the results show that there are improvements in term of power consumption, propagation delay and PDP compared to others design. It has a good driving capability with good output signal in all input combinations and better performance especially in low supply voltage The proposed XOR gate has a much less delay and hence much less PDP than its peer designs. It give better performance which can operate at low voltages which is the best choice for low-power and low-voltage application with requires small area and low power consumption such as RFID tags.

ACKNOWLEDGMENT

This work was financial supported from Universiti Tun Hussein Onn Malaysia, Malaysia

REFERENCES

[1] K. E. Neil H. E. Weste, "Principles of CMOS VLSI Design: A Systems Perspective," 1993.
[2] S.M. Kang and Y. Leblebici, *"CMOS Digital Integrated Circuits"*, McGraw Hill, 2002.
[3] Lin K J and Wu C W 1995 A low-cost realization of multiple-input exclusive-OR gates *Proc. 8th Annual IEEE International ASIC Conference and Exhibit* (Austin: IEEE) pp 307 – 310.
[4] W. Jyh-Ming, F. Sung-Chuan, and F. Wu-Shiung, "New efficient designs for XOR and XNOR functions on the transistor level," *Solid- State Circuits, IEEE Journal of*, vol. 29, 1994, pp. 780-786.
[5] B. Hung Tien, A. K. Al-Sheraidah, and W. Yuke, "New 4-transistor XOR and XNOR designs," in *ASICs, 2000. AP-ASIC 2000. Proceedings of the Second IEEE Asia Pacific Conference on*, 2000, pp. 25-28.
[6] A. Morgenshtein, A. Fish and A. Wagner, "Gate-Diffusion Input (GDI)-A Novel Power Efficient Method for Digital Circuits: A Design Methodology", in *Proc. 14th Annual IEEE International ASIC/SOC Conference*, 2001, pp 39-43
[7] A. B. Shubhajit Roy Chowdhury, Aniruddha Roy, Hiranmay Saha, "A high Speed 8 Transistor Full Adder Design using Novel 3 Transistor XOR Gates," *Int. Journal of Electronics, Circuits and Systems 2;4* 2008.
[8] M.A. Elgamel, S. Goel and M.A. Bayoumi, "Noise Tolerant Low Voltage XOR-XNOR for Fast Arithmetic" in *Proc. ACM Great Lakes Symposium on VLSI*, 2003, pp 285-288
[9] B. Hung Tien, W. Yuke, and J. Yingtao, "Design and analysis of low-power 10-transistor full adders using novel XOR-XNOR gates," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 49, pp. 25-30, 2002.
[10] A.M. Shams, T.K. Darwish and M.A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *IEEE Trans. on VLSI Systems*, 2002, vol. 10 20-9, pp. 20-29.
[11] D. Radhakrishnan, "Low-voltage low-power CMOS full adder," *Circuits, Devices and Systems*, in *IEE Proceedings*, vol. 148, 2001, pp. 19-24.
[12] L. Hanho and G. E. Sobelman, "New XOR/XNOR and full adder circuits for low voltage, low power applications," *Microelectronics Journal*, 1998, vol. 29, pp. 509-517.
[13] L. Hanho and G. E. Sobelman, "New low-voltage circuits for XOR and XNOR," in *Southeastcon '97. 'Engineering new New Century', Proceedings. IEEE*, 1997, pp. 225-229.
[14] W. Dan, Y. Maofeng, C. Wu, G. Xuguang, Z. Zhangming, and Y. Yintang, "Novel low power full adder cells in 180nm CMOS technology," in *4th IEEE Conference on Industrial Electronics and Applications*, 2009, pp. 430-433.
[15] N. Ahmad and R. Hasan, "Design of XOR gates in VLSI implementation," in *Proc. ENZCON 2010*, 2010, pp 51-55.

Proceedings of the 18th Electronics New Zealand Conference, 21-22 November, 2011

Decomposition method for AES Mix Column and Inv Mix Column VLSI architecture optimization

Nabihah Ahmad
Centre for Research in Analog VLSI Microsystems Design CRAV
School of ngineering and Advanced Technology
Massey University Auc land
Auc land New ealand
N.Ahmad massey.ac.nz

S.M. Rezaul Hasan
Centre for Research in Analog VLSI Microsystems Design CRAV
School of ngineering and Advanced Technology
Massey University Auc land
Auc land New ealand
hasanmic massey.ac.nz

Abstract—**This paper presents an efficient integrated Mix Column and Inverse (Inv) Mix Column architecture for Advanced Encryption System (AES) using decomposition method to optimize the chip area and path delay. This development is suitable for compact 8-bit AES cipher. The evaluation of the proposed design is based in terms of total area of number of Exclusive OR (XOR) gates and critical path delay. The hardware cost of the proposed design is about 190 logic gates equivalent to1140 transistors with shorter critical path of 6 XOR gate results in reduction in area compared favourably with other existing design including direct implementation.**

Keywords- AES; Mix Column; Inv Mix Column; Decryption;, Encryption; VLSI

I. INTRODUCTION

A S based on Ri ndael algorithm 6 was selected as a data encryption standard by the National Institute of Standards and Technology NIST in 1997 based on the primary criteria of security performance efficiency in software and hardware implementation and fle ibility. It has been published as FIPS 197 in November 2001. A S is one of the most common symmetric encryption algorithms and is widely adopted for a variety of encryption needs such as wireless networ s and secure transactions via the Internet. A S can be implemented on a wide range of platforms under different constraints 7 .

It can be implemented in both software and hardware The overall efficiency of A S hardware implementation in terms of size speed security and power dissipation depends largely on the A S architecture 8 . In the pure hardware implementation the higher data rate could be obtained by parallel processing and pipelining 9 and it is physically secure since tempering by an attac er is more difficult compared to software implementation.

For compact implementation the same logic resource bloc is shared by both encryption and decryption process. Mi Colum transformation is one of the ma or critical resources consuming bloc besides SubByte Transformation. The ma or concern of this paper is the Mi Column and the Inv Mi Column architecture which with efficient implementation can result in dramatic increase in system performance.

The paper is organized as follows: in Section II detailed on the operation involved in Mi Column transformation is reviewed. Subse uently in section III the proposed design of efficient Mi Column/Inv Mi Column is detailed. In section IV discussion on VLSI architecture is presented. The comparison and evaluation for proposed and e isting designs are carried out. Finally a conclusion will be made in the last section.

II. MIXCOLUMN/ INV MIX COLUMN TRANSFORMATION

Mi Column transformation is a linear diffusion process that operates the 4-byte data bloc s on each column of a *State* matri individually. ach column of the *State* matri is considered as polynomials over GF 2^8 and is multiplied by a fi ed matri M where bytes are treated as a polynomial of degree less than 4 and it is defined as:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \quad 1$$

For the Inv Mi Column each column is multiplied with

$$b(x)= \{0B\}x^3+\{0D\}x^2+\{09\}x+\{0E\} \text{ modulo } x^4+1 \quad 2$$

The Mi Column and Inv Mi Column transformations operate column-by-column on the 4 4 state arrays. Mi Column multiplies the input polynomial by a constant polynomial c modulo 4+1 given by

$$C(x) = \{03\}\, x^3 + \{01\}\, x^2 + \{01\}x + \{02\} \quad 3$$

The multiplication by 02 is denoted time . Inverse Mi Column multiplies the polynomial with

$$c^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad 4$$

Various architectural methods have been proposed for efficient computation of Mi Column/ Inv Mi Column including substructure sharing methods for byte-level and bit-level optimizations.

The Xtime based implementation is a byte-level method that uses the time functional bloc to perform the constant multiplication by 02 in GF 2^8 and to realise other constant multiplications based on time function. The byte-level substructure sharing method finds the common terms with the aim to reduce the number of time bloc by precomputing additions of some variables before using time unit.

The bit-level substructure sharing method is to e tract two term common factors form the bit-level e pressions in Mi Column and Inv Mi Column by selecting the factors that occur most fre uently. hang et al. 5 implement Mi Column and Inv Mi Column using 193 XOR gates and 7 XOR gates in the critical path while Fischer et al. 4 use the decompositions resulted in 192 XOR gates for each column with 7 XOR gates in critical path. In 3 proposed architecture of Mi Column and Inv Mi Column with 292 XOR gate with 6 XOR gates of critical path. Other implementations of A S Mi Column and Inv Mi Column can be found in 1 and 2 .

III. HARD AR IMPL M NTATION

Mi Columns and Inv Mi Columns are modular multiplications by the constant vectors and matri e pression of the Mi Column and Inv Mi Column for one output column of the State array are shown in uations 5 and 6 respectively.

$$\begin{bmatrix} mc0 \\ mc1 \\ mc2 \\ mc3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \mathrm{X} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad 5$$

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \mathrm{X} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad (6)$$

Mi Column matri can be decomposed to a linear combination of products of elements in GF 2^8 as multiplication over addition in GF 2^8 is distributive.

$$mc0 = 02(a_0 + a_1) + (a_2 + a_3) + a_1$$
$$mc1 = 02\,(a_1 + a_2) + (a_3 + a_0) + a_2$$
$$mc2 = 02\,(a_2 + a_3) + (a_0 + a_1) + a_3$$
$$mc3 = 02\,(a_0 + a_3) + (a_1 + a_2) + a_0 \quad 7$$

Mi Column matri is then e pressed as

$$\begin{bmatrix} mc0 \\ mc1 \\ mc2 \\ mc3 \end{bmatrix} = \begin{bmatrix} 02 * (a_0 + a_1) + (a_2 + a_3) + a_1 \\ 02 * (a_1 + a_2) + (a_3 + a_0) + a_2 \\ 02 * (a_2 + a_3) + (a_0 + a_1) + a_3 \\ 02 * (a_3 + a_0) + (a_1 + a_2) + a_0 \end{bmatrix} \quad 8$$

Xtime is the multiplication by which is 2 . Xtime circuit is implemented using combinations of XOR gates of subse uent conditional XOR with 1B and hard-wired logic shift operations of left shift.

xtimeout [7:0] = { in[6], in[5], in[4], (in[3] $\oplus$ *in[7]), (in[2]* $\oplus$ *in[7]), in[1], (in[0]* $\oplus$ *in[7]), in[7] }* 9

Multiplication for higher power of can be done by repeated of time . For 4time is same as 4time time ^{2} time time time 2 4

Xtime and X4time circuit is shown in Figure 1 and Figure 2 respectively

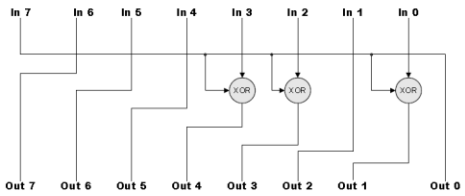


Figure 1. XTime circuit

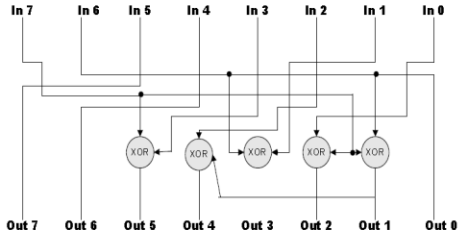


Figure 2. X4Time circuit

The matri multiplication of Mi Column could be represented as follows

$$mc0 = xtime(a_0 \oplus a_1) \oplus (a_2 \oplus a_3) \oplus a_1$$
$$mc1 = xtime(a_1 \oplus a_2) + (a_3 \oplus a_0) \oplus a_2$$
$$mc2 = xtime(a_2 \oplus a_3) + (a_0 \oplus a_1) \oplus a_3$$
$$mc3 = xtime(a_3 \oplus a_0) + (a_1 \oplus a_2) \oplus a_0 \quad 10$$

Inv Mi Column e uation in 6 also can be e pressed as e uation 11 . For Inv Mi Column decomposition using distributivity of GF 2^8 can be e pressed as in e uation 12 . In order to reduce the cost of the hardware Inv Mi Column can be decomposed to share the logic resources with Mi Column. The e uation is substitute with e uation 8 to share the architecture obtained the new e uation 13 as following.

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 04 & 00 & 02 & 00 \\ 00 & 04 & 00 & 04 \\ 02 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \end{bmatrix} \mathrm{X} \begin{bmatrix} 02 & 02 & 02 & 02 \\ 00 & 00 & 02 & 02 \\ 02 & 00 & 02 & 02 \\ 00 & 00 & 02 & 02 \end{bmatrix} X \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \quad 11$$

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_0+a_1)+(a_2+a_3)+a_1+4*(a_0+a_2') \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_1+a_2')+(a_3+a_0)+a_2+4*(a_1+a_3) \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_2+a_3)+(a_0+a_1)+a_3+4*(a_2+a_0) \\ 4*2*[(a_0+a_2)+(a_1+a_3)]+2*(a_3+a_0)+(a_1+a_2)+a_0+4*(a_3+a_1) \end{bmatrix} \quad 12$$

$$\begin{bmatrix} mc0' \\ mc1' \\ mc2' \\ mc3' \end{bmatrix} = \begin{bmatrix} 4*2*[(a_0+a_1)+(a_2+a_3)]+mc0+4*(a_0+a_2) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc1+4*(a_1+a_3) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc2+4*(a_2+a_0) \\ 4*2*[(a_0+a_1)+(a_2+a_3)]+mc3+4*(a_3+a_1) \end{bmatrix} \quad 13$$

The matri multiplication of Inv Mi Column could be represented as follows

$$mc0' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc0 \oplus x4time(a_0 \oplus a_2)$$
$$mc1' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc1 \oplus x4time(a_2 \oplus a_3)$$
$$mc2' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc2 \oplus x4time(a_2 \oplus a_0)$$
$$mc3' = x4time(xtime)[(a_0 \oplus a_1) \oplus (a_2 \oplus a_3)] \oplus mc3 \oplus x4time(a_3 \oplus a_1) \quad 14$$

The hardware implemented for Mi Column and Inv Mi Column from e uation 10 and 14 is illustrated in Figure 3.

IV. DISCUSSIONS

The total number of gates in the proposed architecture is 190 XOR gates and it results in a critical path of 6 T_{XOR}. Table 1 shows the comparison results in terms of number of gate and critical path delay with other method implementation. The proposed integrate Mi Column and Inv Mi Column design achieved 60 reductions in area as compared to direct implementation 1 .

Method	Area gate	Delay T_{XOR}
Direct implementation 1	1888	5
Shared XTime 2	1216	8
Hua Li 3	1168	6
Fischer 4	768	-
Proposed	760	6

TABL I. COMPARISON OF DIFF R NT MC/ IMC D SIGNS

V. CONCLUSIONS

Here we have detailed the new alternative design of compact hardware architecture for both Mi Column and Inv Mi Column transform in the A S. This cost of 190 XOR gates for computation of one column of a state 760 XOR gate for computation of full state and critical path of 6 T_{XOR} hardware comple ities. This design proves to be low in gate count which is suitable for compact system of 8-bit A S system.

R F R NC S

1 S.-F. Hsiao M.-C.Chen *Efficient substructure sharing methods for optimising the inner-product operations in Rijndael advanced encryption standard* I Proc Computers and Digital Techni ues vol. 152 no. 5 2005.

2 Chih-Chung Lu and Shau- in Tseng, *Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter* in Proc The I International Conference on Application-Specific Systems Architectures and Processors 2002

3 H.Li and . Friggstad *An efficient architecture for the AES* mi columns operation. In Proceedings of ISCAS 2005

4 V. Fischer P. Chodowiec and F. Gramain *InvMixColumn Decomposition and Multilevel Resource Sharing in AES Implementations* I Trans. on VLSI Systems vol.13 no. 8 pp. 989-992 2005.

5 X. hang and K.K.Parhi *High Speed VLSI Architectures for AES algorithm* I Trans. VLS vol. 2 pp. 957-967 2004

6 . Daemen and V. Ri men *The Design of Rijndael Springer-Verlag* New or Inc. 2002.

7 S. Tillich M. Feldhofer T. Popp . Gro *Area, delay, and power characteristics of standard-cell implementations of the AES S-Box* . Signal Process. Syst., vol. 50 pp. 251-261 2008.

8 A. Satoh S. Morio a K. Ta ano and S. Munetoh *A Compact Rijndael Hardware Architecture with S-Box Optimization* in Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology: Springer-Verlag 2001.

9 N. S lavos and O. Koufopavlou *Architectures and VLSI implementations of the AES-Proposal Rijndael* Computers I Transactions on vol. 51 pp. 1454-1459 2002.

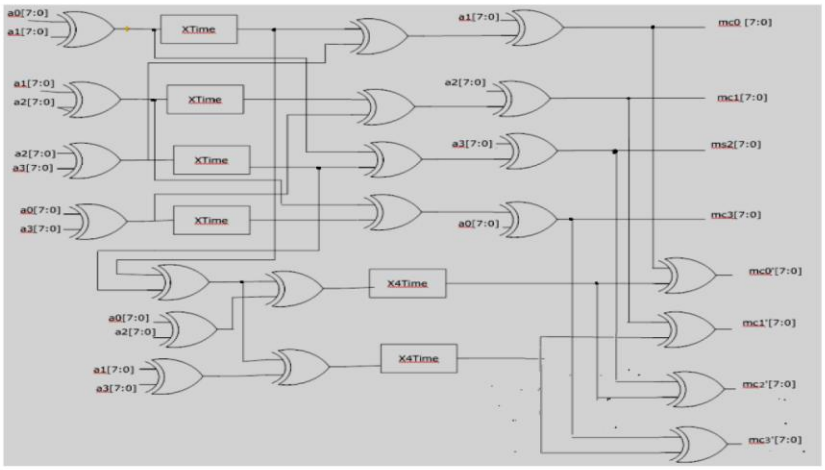


Figure 3. Mi Column/ Inv Mi Column architecture

2011 International Conference on Electronic Devices, Systems & Applications (ICEDSA)

A New Design of XOR-XNOR gates for low power application

Nabihah Ahmad
Faculty of Electrical and Electronic,
Universiti Tun Husseion Onn Malaysia
Batu Pahat, Johor, Malaysia
nabihah@uthm.edu.my

Rezaul Hasan
School of Engineering and Advanced Technology
Massey University
Auckland, New Zealand
hasanmic@massey.ac.nz

***Abstract*—XOR and XNOR gate plays an important role in digital systems including arithmetic and encryption circuits. This paper proposes a combination of XOR-XNOR gate using 6-transistors for low power applications. Comparison between a best existing XOR-XNOR have been done by simulating the proposed and other design using 65nm CMOS technology in Cadence environment. The simulation results demonstrate the delay, power consumption and power-delay product (PDP) at different supply voltages ranging from 0.6V to 1.2V. The results show that the proposed design has lower power dissipation and has a full voltage swing.**

Keywords-XOR-XNOR gate, low power, delay

I. INTRODUCTION

Circuit realization for low power and low area has become an important issue with the growth of integrated circuit towards very high integration density and high operating frequencies. Due to the important role played by XOR and XNOR gate in various circuits especially in arithmetic circuits, optimized design of XOR and XNOR circuit to achieve low power, small size and delay is needed. The primary concern to design XOR-XNOR gate is to obtain low power consumption and delay in the critical path and full output voltage swing with low number of transistors to implement it.

In this paper, we propose a new design of XOR-XNOR gate using 6 transistors. The paper is organized as follows: in Section II, previous work is reviewed. Subsequently, in section III, the proposed design of XOR-XNOR gate is presented. In section IV, the simulation results are given and discussed. The comparison and evaluation for proposed and existing designs are carried out. Finally a conclusion will be made in the last section.

II. PREVIOUS WORK

Numerous designs were reported to realize the XOR-XNOR functions using different number of circuit techniques and approaches [1], [2], [3], [4], [5], [6], [7]. They vary in methodologies and transistor count to improve the circuit performance in term of speed and density. Among these, the conventional design of XOR-XNOR circuit using static CMOS network can be found in [4]. Each input is connected to both an NMOS transistor and a PMOS transistor. It provide a full output voltage swing but with a large number of transistors.

Complementary pass transistor logic (CPL) is used in [1]. Wang et al. [2] report the XOR-XNOR circuits based on transmission gates. It uses eight transistors and complementary inputs and has a drawback of loss of driving capability. Wang et al. also designed XOR-XNOR circuits based on inverter gates. It does not require a complementary inputs but it has no driving capability because there is no direct connection to Vdd and Gnd. The improved version of this circuit has been designed by adding a standard inverter to the output. This modified circuit provides a good driving capability but uses twelve transistors for XOR-XNOR circuits.

Shiv et al. [3] proposed two of XOR-XNOR circuits (Figure 1 and 2) and claimed to have lower PDP, less power dissipation and faster compared to design in [5] with a low supply voltage. However both of the circuits give a poor signal output voltage in certain input combination. In [6], the XOR and XNOR circuit based on Pass Transistor Logic (PTL) using 6 transistors is reported as shown in Figure 3. It has a full output voltage swing and better driving capability by using Vdd and Gnd connection.

Elgamel et al. [7] proposed an improved version of [6] in Figure 4 and has better power-delay product and higher noise immunity. In [5], the XOR-XNOR circuit adds a forward and backward feedback between the XOR and XNOR gate and additional transistors to rectify the degraded logic level problem. This configuration shown in Figure 5 provides a better performance of the circuit.

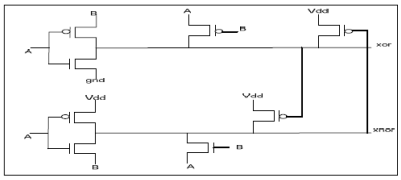


Figure 1. XOR-XNOR gate using 8 transistors in [3]

Universiti Tun Hussein Onn Malaysia

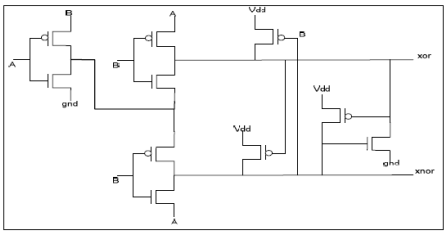


Figure 2. XOR-XNOR gate using 10 transistors in [3]

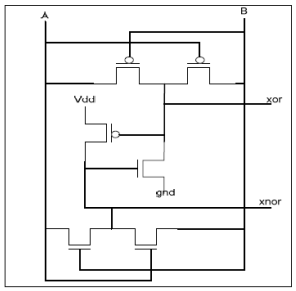


Figure 3. XOR-XNOR gate using 6 transistors in [6]

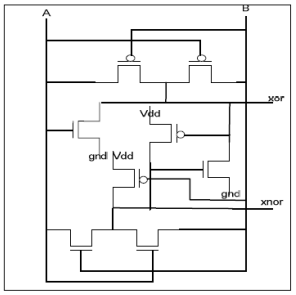


Figure 4. XOR-XNOR gate using 8 transistors in [7]

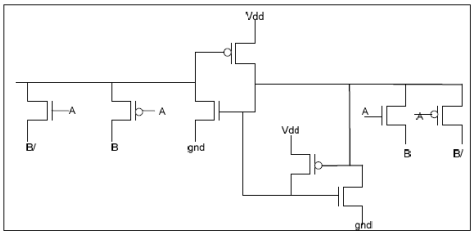


Figure 5. XOR-XNOR gate using 8 transistors in [5]

III. PROPOSED XOR-XNOR GATE CIRCUIT

The XOR and XNOR gate functions are shown in Table 1 and denoted by $\oplus$ and $\odot$ respectively. The logic expression for XOR and XNOR are

$$A \oplus B = A'B + AB' \quad (1)$$

$$A \odot B = A'B' + AB \quad (2)$$

TABLE I. XOR AND XNOR GATE FUNCTION

A	B	XOR	XNOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

The proposed design of XOR-XNOR gate using six transistors is shown in Figure 6. It uses a concept of pass transistor and CMOS inverter. The inverter is used as a driving output to achieve a perfect output swing. Vdd connection to transistor M3 and M4 are use to drive a good output of '1'. Transistor M4 is used to drive the output signal when XOR output '0' when input signal A=B=1. In this condition, when transistor M1 or M2 is ON, it will pass a poor signal '1' with respect to the input to the inverter. The output XOR will also be degraded and to achieve a good output signal, transistor M4 is ON when output Xor is '0', then it will pass the perfect signal '1' from Vdd to the output XOR.

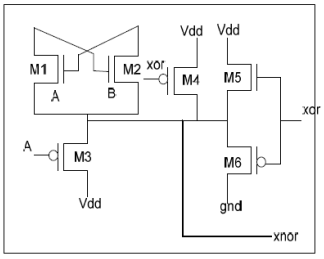


Figure 6. Proposed XOR-XNOR gate using 6 transistors

IV. SIMULATION RESULTS OF 6-TRANSISTOR XOR-XNOR GATE

The XOR-XNOR gate is simulated using Spectre Cadence in the voltage range of 0.6V to 1.2V using 65nm CMOS technology. Simulation is performed at varying supply voltages to show the effect of different voltages to the power dissipation of XOR-XNOR circuit. Comparative analysis has been carried out on the different types of combination XOR-XNOR based on the best previous design. The transient analysis of the circuits were performed with a load capacitance of 50fF at 500MHz and simulated using the same conditions to measure propagation delay and power dissipation. The delay has been computed between the time when the changing input reaches 50% of voltage level to the time it output reaches 50% of

voltage level for both rising and fall transition. The power-delay product (PDP) is measured as the product of the average delay and the average power.

The comparisons of output value for XOR-XNOR circuits for each input combination are shown in Table 2. The results of simulation which included a delay, power dissipation and power delay product are listed in Table 3 and also are represented in Figure 7, 8, 9, 10 and 11. The results indicate that the delay of the proposed XOR-XNOR circuit is smaller than previous circuit in Figure 2[3], [6] and [5], and nearly similar to circuit in Fig 1[3] and [7]. But in terms of power consumption, the proposed circuit consumes less power compared to other design except for the circuit in [6] which consumes less power than other circuits but [6] offer slowest speed at a low voltage. The overall PDP for the proposed circuit have been improved more than 50% from other circuit except from the circuit in [6], but nearly similar at the low supply voltage. Regarding to the simulation results, the proposed XOR-XNOR circuit are the most energy efficient and very well suited to low voltage applications.

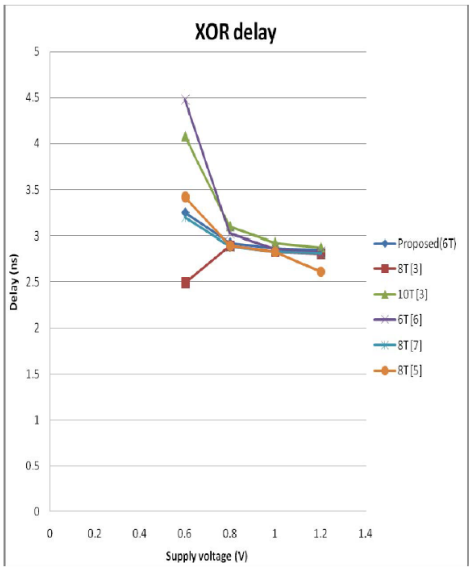


Figure 7. Worst case delay of different XOR circuit

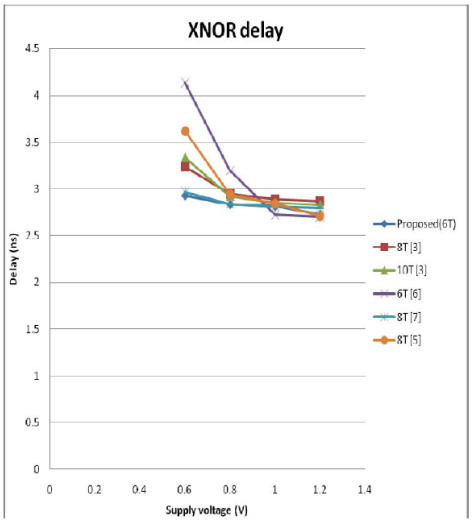


Figure 8. Worst case delay of different XNOR circuit

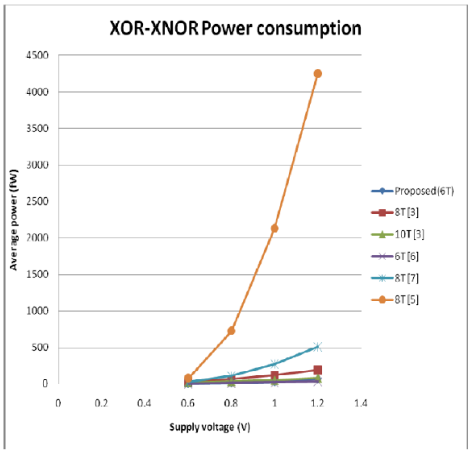


Figure 9. Power consumption of different XOR-XNOR circuit

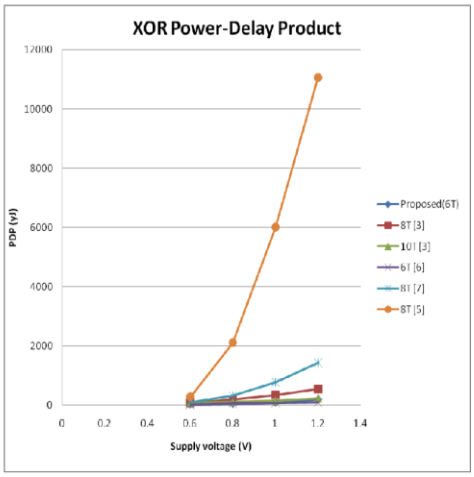


Figure 10. Power-delay products (PDP) of different XOR circuit

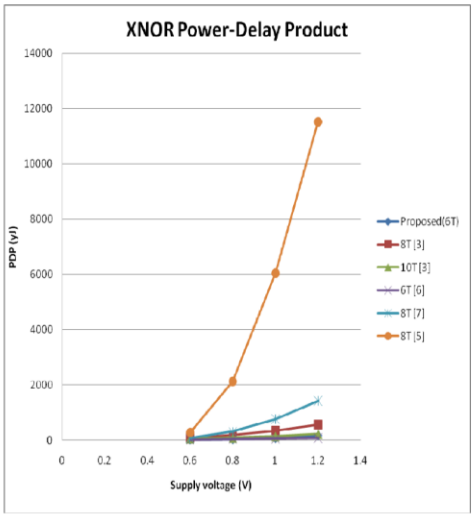


Figure 11. Power-delay products (PDP) of different XNOR circuit

CONCLUSIONS

In this paper, we proposed the new design of combination XOR-XNOR circuit configuration. The performances of this circuit have been compared to previous reported XOR design based on delay, power dissipation and PDP. According to the simulation results, the proposed circuit offers a better and more competitive than other design. It offers the lowest power dissipation at a low supply voltage. It has a good driving capability with good output signal in all input combinations and better performance especially in low supply voltage compared to the previous designs. Thus, the proposed circuit is suitable for low-voltage and low-power application.

ACKNOWLEDGMENT

The authors would like to acknowledge the financial support from Universiti Tun Hussein Onn Malaysia (UTHM).

REFERENCES

[1] U. Ko, P. T. Balsara, and W. Lee, "Low-power design techniques for high-performance CMOS adders," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 3, pp. 327-333, 1995.

[2] W. Jyh-Ming, F. Sung-Chuan, and F. Wu-Shiung, "New efficient designs for XOR and XNOR functions on the transistor level," *Solid-State Circuits, IEEE Journal of*, vol. 29, pp. 780-786, 1994.

[3] S. W. Shiv Shankar Mishra, R. K. Nagaria, and S. Tiwari, "New Design Methodologies for High Speed Low Power XOR-XNOR Circuits," *World Academy of Science, Engineering and Technology*, 2009.

[4] K. E. Neil H. E. Weste, "Principles of CMOS VLSI Design: A Systems Perspective," 1993.

[5] S. Goel, M. A. Elgamel, M. A. Bayoumi, and Y. Hanafy, "Design methodologies for high-performance noise-tolerant XOR-XNOR circuits," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 53, pp. 867-878, 2006.

[6] D. Radhakrishnan, "Low-voltage low-power CMOS full adder," *Circuits, Devices and Systems, IEE Proceedings -*, vol. 148, pp. 19-24, 2001.

[7] M. Elgamel, S. Goel, and M. Bayoumi, "Noise tolerant low voltage XOR-XNOR for fast arithmetic," in *Proceedings of the 13th ACM Great Lakes symposium on VLSI* Washington, D. C., USA: ACM, 2003.

[8] A. B. Shubhajit Roy Chowdhury, Aniruddha Roy, Hiranmay Saha, "A high Speed 8 Transistor Full Adder Design using Novel 3 Transistor XOR Gates," *International Journal of Electronics, Circuits and Systems 2;4* 2008.

TABLE II. COMPARISON OF OUTPUT VALUE FOR XOR-XNOR GATE

Inputs		Proposed (6T)		8T [3]		10T [3]		6T [6]		8T [7]		8T [5]	
A	B	XOR	XNOR	XOR	XNOR	XOR	XNOR	XOR	XNOR	XOR	XNOR	XOR	XNOR
0	0	Good 0	Good 1	Bad 0	Good 1	Good 0	Good 1	Bad 0	Good 1	Good 0	Good 1	Good 0	Good 1
0	1	Good 1	Good 0	Good 1	Good 0	Good 1	Good 0	Good 1	Bad 0	Good 1	Good 0	Good 1	Bad 0
1	0	Good 1	Good 0	Bad 1	Good 0	Good 1	Bad 0	Good 1	Good 0	Bad 1	Bad 0	Bad 1	Good 0
1	1	Good 0	Good 1	Good 0	Good 1	Good 0	Good 1	Good 0	Good 1	Good 0	Good 1	Good 0	Bad 1

TABLE III. SIMULATION RESULT OF XOR-XNOR GATE

	V(v)	Proposed(6T)	8T [3]	10T [3]	6T [6]	8T [7]	8T [5]
Delay for XOR (ns)	0.6	3.252	2.484	4.083	4.483	3.205	3.422
	0.8	2.925	2.892	3.101	3.025	2.886	2.891
	1	2.865	2.825	2.928	2.86	2.828	2.824
	1.2	2.843	2.802	2.874	2.823	2.81	2.603
Delay for XNOR (ns)	0.6	2.926	3.239	3.339	4.136	2.966	3.62
	0.8	2.832	2.947	2.922	3.195	2.835	2.942
	1	2.813	2.89	2.852	2.72	2.81	2.844
	1.2	2.731	2.867	2.831	2.701	2.798	2.711
Average power for XOR-XNOR (fW)	0.6	6.379	27.7	19.93	2.251	25.69	77.46
	0.8	12.62	64.36	33.65	13.75	109.4	726
	1	28.78	121.4	50.19	21.36	270.5	2129
	1.2	65.04	196.2	79.98	30.55	510.3	4254
PDP for XOR (yJ)	0.6	20.75	68.8	81.38	10.09	82.34	265.07
	0.8	36.91	186.1	104.35	41.53	315.73	2099
	1	82.46	343	146.96	61.09	764.97	6012.3
	1.2	184.9	549.8	229.86	86.24	1433.94	11073
PDP for XNOR (yJ)	0.6	18.66	89.721	66.55	9.31	76.2	280.41
	0.8	35.74	189.67	98.33	43.87	310.15	2135.89
	1	80.96	350.85	143.14	58.1	760.11	6054.88
	1.2	177.62	562.51	226.42	82.52	1427.82	11532.6

CONFERENCE PROCEEDING IV

N. Ahmad, R. Hasan, 'Design of XOR gates in VLSI implementation', *Proceedings of Electronics New Zealand Conference 2010*, pp. 51–55, Hamilton, New Zealand, November 2010.

Design of XOR gates in VLSI implementation

Nabihah Ahmad[1] and Rezaul Hasan[2]

School of Engineering and Advanced Technology
Massey University, Auckland
[1]N.Ahmad@massey.ac.nz, [2]hasanmic@massey.ac.nz

Abstract: Exclusive OR (XOR) gate is a fundamental building block in various digital applications such as full adder, comparator, parity generator and encryption processor, which leads to increased interest in enhanced performance of XOR gate. This paper proposes a new design of XOR gate using six transistors for low power application. The new XOR gate has been compared with previous designs in term of power, delay and power-delay product (PDP). The XOR gate is simulated using Cadence Virtuoso Analog Environment in 65nm Complementary Metal Oxide Semiconductor (CMOS) technology at different supply voltages with a range of 0.6V to 1.2V.

Keywords: XOR gate, power, delay, PDP

1 INTRODUCTION

As the essential unit in digital logic design, XOR gate contribute to the overall performance and power of the system. Therefore it is required to design the XOR gate which satisfies the low power dissipation and delay with small size.

In this paper, we propose a novel design of 2 input XOR gate using six transistors. The paper is organized as follows: in Section II, previous work is reviewed. Subsequently, in section III, the proposed design of XOR gate is presented. In section IV, the simulation results are given and discussed. The comparison and evaluation for proposed and existing designs are carried out. Finally a conclusion will be made in the last section.

1.1 Previous Work

There are varieties of XOR gate design have been reported in literature [1-7]. The conventional design of XOR gate is based on eight transistors in static CMOS [8]. It can operate with full output voltage swing but it requires more numbers of transistors. Emphasis has been done on the design of four transistor XOR gate [1, 3, 6, 7, 9]. Radhakrisnan [3] proposed a combination of XOR and Exclusive NOR (XNOR) circuit using 6 transistors.

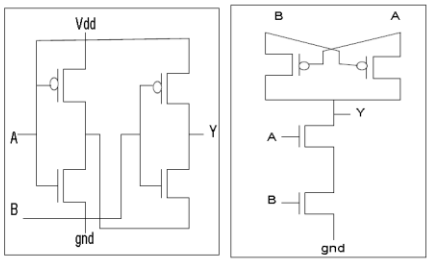
Wang et al. [6] proposed four transistor XOR gate architecture shown in Figure 1(a) and Figure 1(b). These architecture gives a poor signal output for a certain input. The average delay for Figure 1(a) and 1(b) was 3.84ns and 1.42ns respectively with 400uW and 310uW. They improved the level output by cascading a standard transistor inverter as a driving output to achieve a perfect output. This XOR gate is consist of six transistors in total as shown in Figure 2.

In [7], the set of four transistors P-XOR circuit called powerless is proposed which consumes less power than other design because it has no power supply connection. The drawback is it causes a large delay but better than conventional CMOS. The delay was 350ps with maximum input frequency 200MHz.

XOR gate based on Gate-Diffusion-Input(GDI) cell was reported in [9] which requires 4 transistors..

Figure 3 shows the XOR gate circuit in [10] by using three transistors which modify a CMOS inverter with a PMOS pass transistor. It have a voltage degradation when the input A=1 and B=0, but can be minimized by

increasing the W/L ratio of PMOS transistor. It offer less power-delay product compared to four transistors design in [6] using 0.35um technology.



(a) (b)

Fig. 1(a) and (b) Design of 4 transistors XOR gate by [6]

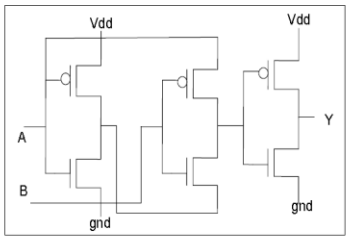


Fig. 2 Improved 6 transistors XOR gate by [6]

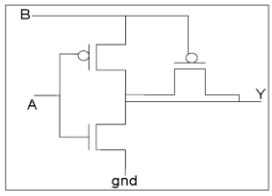


Fig. 3 Design of 3 transistors XOR gate by [10]

2 DISCUSSION

The XOR gate functions is shown in Table 1 and denoted by ⊕. The logic expression for XOR is

$$A \oplus B = A'B + AB' \quad (1)$$

TABLE I
XOR GATE FUNCTION

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

The proposed design of XOR gate using six transistors is shown in Figure 4. It uses a concept of pass transistor and CMOS inverter. The inverter is used as a driving output to achieve a perfect output swing. Vdd connection to transistor M3 and M4 are used to drive a good output of '1'. Transistor M4 is used to drive the output signal when Y=0 when input signal A=B=1. In this condition, when transistor M1 or M2 are ON, it will pass a poor signal '1' with respect to the input to the inverter. The output Y will also be degraded and to achieve a good output signal, transistor M4 is ON when Y=0, then pass the perfect signal '1' from Vdd. The W/L ratio of transistor M2 is bigger than the W/L ratio of transistor M3 to get the output Y=1 as both of the transistors will be ON when A=0 and B=1.

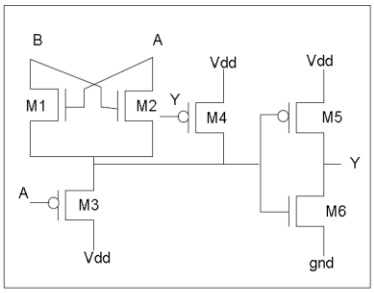


Fig. 4 Proposed 6 transistor XOR gate

The input and output waveform of XOR gate are shown in Figure 5. From the figure it is found that the output waveform for each input combination is full output voltage swing compared to previous design of XOR gate. It eliminates the voltage degradation in certain input.

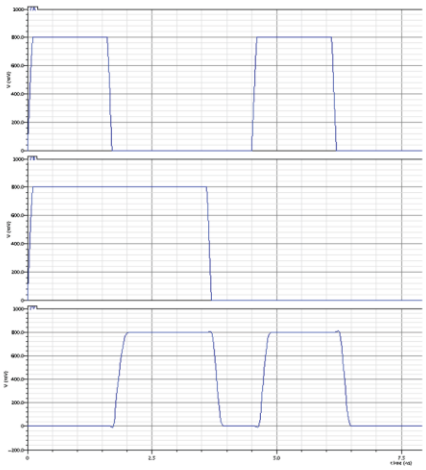


Figure 5 Waveform of the proposed 6 transistor XOR gate

Comparative analysis has been carried out on the different types of XOR gates based on previous XOR design. By using Spectre Cadence, each of the design is simulated using the same conditions to measure propagation delay and power dissipation. The DC and transient analysis of the circuits were performed at a supply voltage at range of 0.6 to 1.2V using 65nm technology with load capacitance of 50fF at 500MHz waveform. The results of simulation which included a delay, power dissipation and power delay product are represented in Table 1 and Figures 6, 7 and 8. The delay has been computed between the time the changing input reaches 50% of voltage level to the time it output reaches 50% of voltage level for both rising and falling transition. The power-delay product (PDP) is measured as the product of the average delay and the average power.

The results indicate that the delay of the proposed XOR gate is between the four transistors XOR gates in [6]and three transistors in [10], and less than six transistors in [6]. The proposed circuit give a lower power dissipation in a low voltage compared to the other design in 4T [6] and 3T [10] which is slightly more than [6] in high voltage but less than [10]. But when compared to 6T in [6], it consumes less power than it. Compared to the design in [6] and [10], the power-delay product of proposed XOR gate is less at the supply voltage between 0.6V and 0.8V. The PDP of six transistors in [6] is the highest from other design including the proposed design.

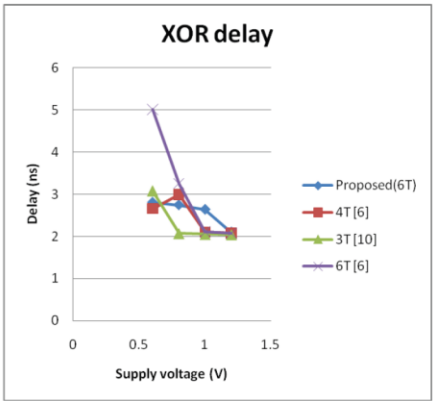


Figure 6 Delay of different XOR gates

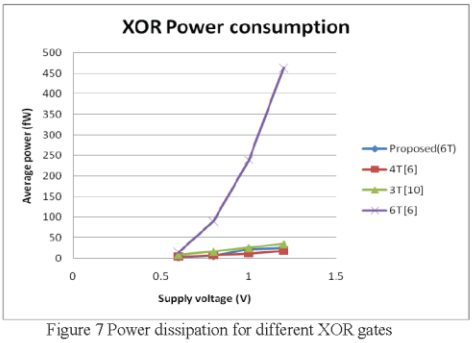


Figure 7 Power dissipation for different XOR gates

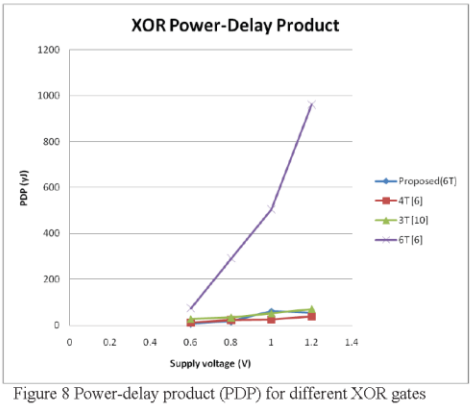


Figure 8 Power-delay product (PDP) for different XOR gates

Table 1 Simulation Results of XOR gate

	V(v)	Proposed(6T)	4T [6]	3T [10]	6T [6]
Delay (ns)	0.6	2.81	2.656	3.08	5.006
	0.8	2.75	2.994	2.061	3.251
	1	2.642	2.094	2.045	2.111
	1.2	2.099	2.068	2.035	2.075
Average power (fW)	0.6	2.312	4.189	8.703	14.66
	0.8	6.069	7.613	16.25	89.57
	1	22.77	12.12	25.36	238.9
	1.2	25	18.81	34.96	463.3
PDP (yJ)	0.6	6.497	11.13	26.81	73.36
	0.8	16.679	22.79	33.49	291.2
	1	60.158	25.38	51.86	504.2
	1.2	52.475	38.91	71.15	961.5

Table 2 Noise Margin of different XOR gate

Type of XOR	V(v)	Voh(V)	Vih(V)	Vil(V)	Vol(V)	Nmh(V)	Nml(V)
Proposed (6T)	0.600	0.600	0.300	0.295	0.000	0.300	0.295
	0.800	0.800	0.400	0.390	0.000	0.400	0.390
	1.000	1.000	0.523	0.477	0.000	0.477	0.477
4T [6]	0.600	0.600	0.340	0.300	0.000	0.260	0.300
	0.800	0.800	0.400	0.400	0.000	0.400	0.400
	1.000	0.908	0.500	0.500	0.000	0.408	0.500
3T [10]	0.600	0.600	0.270	0.330	0.260	0.330	0.070
	0.800	0.800	0.370	0.430	0.300	0.430	0.130
	1.000	1.000	0.470	0.530	0.000	0.530	0.530

Based on DC analyses, the noise margins are measured. The noise margin is the ability to tolerate noise without affecting the correct operation of the circuit [11] . The low noise margin, Nml and high noise margin, Nmh are in the following equation (1) and (2) respectively.

$$Nml = Vil - Vol \quad (1)$$

$$Nmh = Voh - Vih \quad (2)$$

The noise margin of XOR gate has been studied at the different supply voltage as shown in Table 2. The proposed XOR gate indicates acceptable values of noise margin compared with other two XOR gates.

3 FURTHER WORK

The proposed XOR gate will be use in the design of Substitution Box (S-BOX) in Advanced Encryption System (AES) to achieve a good performance because it offer low power and reliable output.

4 CONCLUSIONS

In this paper, a new design of XOR gate has been proposed using six transistors. The performances of this circuit have been compared to previous reported XOR design based on delay, power dissipation and PDP. The proposed circuit give a perfect output signal in all input combinations and better performance especially in low supply voltage compared to the previous designs.

5 REFERENCES

[1] B. Hung Tien, W. Yuke, and J. Yingtao, "Design and analysis of low-power 10-transistor full adders using novel XOR-XNOR gates," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on,* vol. 49, pp. 25-30, 2002.

[2] A. M. Shams, T. K. Darwish, and M. A. Bayoumi, "Performance analysis of low-power 1-bit CMOS full adder cells," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 10, pp. 20-29, 2002.

[3] D. Radhakrishnan, "Low-voltage low-power CMOS full adder," *Circuits, Devices and Systems, IEE Proceedings -,* vol. 148, pp. 19-24, 2001.

[4] L. Hanho and G. E. Sobelman, "New XOR/XNOR and full adder circuits for low voltage, low power applications," *Microelectronics Journal*, vol. 29, pp. 509-517, 1998.

[5] L. Hanho and G. E. Sobelman, "New low-voltage circuits for XOR and XNOR," in *Southeastcon '97. 'Engineering new New Century'., Proceedings. IEEE*, 1997, pp. 225-229.

[6] W. Jyh-Ming, F. Sung-Chuan, and F. Wu-Shiung, "New efficient designs for XOR and XNOR functions on the transistor level," *Solid-State Circuits, IEEE Journal of*, vol. 29, pp. 780-786, 1994.

[7] B. Hung Tien, A. K. Al-Sheraidah, and W. Yuke, "New 4-transistor XOR and XNOR designs," in *ASICs, 2000. AP-ASIC 2000. Proceedings of the Second IEEE Asia Pacific Conference on*, 2000, pp. 25-28.

[8] N. Weste and K. Eshraghian, "Principles of CMOS VLSI Design: A Systems Perspective," Addison-Wesley Longman Publishing Co., Inc. 1993.

[9] W. Dan, Y. Maofeng, C. Wu, G. Xuguang, Z. Zhangming, and Y. Yintang, "Novel low power full adder cells in 180nm CMOS technology," in *Industrial Electronics and Applications, 2009. ICIEA 2009. 4th IEEE Conference on*, 2009, pp. 430-433.

[10] A. B. Shubhajit Roy Chowdhury, Aniruddha Roy, Hiranmay Saha, "A high Speed 8 Transistor Full Adder Design using Novel 3 Transistor XOR Gates," *International Journal of Electronics, Circuits and Systems 2;4* 2008.

[11] S. Brown and Z. Vranesic, "Fundamentals of Digital Logic with VHDL Design," McGraw-Hill Higher Education 2005.

2010 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2010), October 3-5, 2010, Penang, Malaysia

Design of AES S-Box using combinational logic optimization

Nabihah Ahmad
Faculty of Electrical and Electronic Engineering
Universiti Tun Hussein Onn Malaysia (UTHM)
Johor, Malaysia
Email:nabihah@uthm.edu.my

Rezaul Hasan
School of Engineering and Advanced Technology
Massey University
New Zealand
Email: hasanmic@massey.ac.nz

Warsuzarina Mat Jubadi
Faculty of Electrical and Electronic Engineering
Universiti Tun Hussein Onn Malaysia (UTHM)
Johor, Malaysia
Email:suzarina@uthm.edu.my

Abstract— **Advanced Encryption Standard (AES) is one of the most common symmetric encryption algorithms. The hardware complexity in AES is dominated by AES substitution box (S-box) which is considered as one of the most complicated and costly part of the system because it is the only non-linear structure. The proposed work employs a combinational logic design of S-Box implemented in Virtex II FPGA chip. The architecture employs a Boolean simplification of the truth table of the logic function with the aim of reducing the delay. The S-Box is designed using basic gates such as AND gate, NOT gate, OR gate and multiplexer. Theoretically, the design reduces the overall delay and efficiently for applications with high-speed performance. This approach is suitable for FPGA implementation in term of gate area. The hardware, total area and delay are presented.**

Keywords—**S-Box, combinational logic, AES, FPGA**

I. INTRODUCTION

This AES algorithm is based on Rijndael algorithm, which was developed by Joan Daemen and Vincent Rijnmen. It was announced by National Institute of Standards and Technology (NIST) in 1997 as AES algorithm [1] according to the primary criteria of security, performance, efficiency in software and hardware, flexibility, and implementability.

One of the strength of AES is it simplicity which facilitates the implementation on a wide range of different platforms under different constraints [2]. AES can be implemented using software and hardware. Hardware implementation can be either based on Field Programmable Gate Array (FPGA) or Application Specific Integrated Circuits (ASIC). AES implemented on FPGA offered flexibility and security, but medium speed compared to ASIC. The major factors that influence the implementation choices are speed and area cost.

By using hardware, a higher data rate for fast applications such as routers can be achieved compared to software implementation. The hardware implementation is also physically secure since tempering by an attacker is difficult. The efficiency of AES hardware implementation in terms of size, speed, security and power consumption depends largely on the AES architecture [3]. For high speed throughput, loop-unrolled pipelined [4] is used and to save power and area, iterative single round with resource sharing is implemented.

As S-Box is considered as a full complexity design and causes high power dissipation in AES, this paper is focused on the way to implement it efficiently. Traditionally it was implemented by look up tables (LUT) which store all 256 bit predefined values of S-Box in a ROM. The advantage of using LUT is it offers a shorter critical path. However, it has a drawback of the unbreakable delay in high speed pipelined designs, and hence it cannot be used in high speed applications. This delay prohibits each round unit from being divided into more than two sub-stages to achieve any further increase in processing speed [5]. It also requires a large area to implement AES encryption and decryption system due to different table used for both systems.

Another way is to design the S-Box circuit using combinational logic only which is directly from its arithmetic properties. This approach has breakable delay of S-Box processing. One of the S-Box design is from its truth table using direct relationship between input and output values of the S-box and two-level logic, which make use of sum of products (SOP) expression, product of sums (POS) expression, positive polarity Reed-Muller (PPRM) structure, binary decision diagram (BDD) or its variance twisted binary decision diagram (TBDD). It can achieve a high speed design but suffer from extremely large area cost.

Rashmi et al. [6] implemented S-Box using a combinational logic without involving the computation in Galois field. He proposed two techniques for implementing the S-Box, (1) based on logic synthesis using truth table, and (2) direct implementation of the Algebraic Normal Form (ANF) expression for each column. It claims to have very low critical path delay compared to designs based on composite field.

Other approach involves multiplicative inversion in Galois Field GF 2^8 using composite field[7]. It leads to low area but the critical delay is increased and it has high hardware complexities.

In this paper, we focus on the Boolean simplification of the truth table to design the S-box circuit using basic gates and multiplexer. Comparison with previous work is

978-1-4244-7646-6/10/$26.00 ©2010 IEEE

carried out. In the next section, the AES algorithm is described briefly. In section III, it includes the discussion on the SubByte transformation. Subsequently, in section IV, the design optimization of S-Box will be proposed. Finally, the comparison and evaluation with previous work will be discussed. A conclusion is made in the last section.

I. AES ALGORITHM

AES is a symmetric encryption algorithm which processes a fixed 128-bit of data block and variable length of keys of 128, 192 and 256 bits. The basic operation of AES performed by two dimensional arrays of bytes, called the State, which consists of four columns and four rows of byte. AES uses a round function consisting of four different data transformation of SubByte, ShitfRow, MixColumn and AddRoundKey. The AES algorithm is an iterative algorithm which performs iteratively for 10, 12 and 14 times depending on the key length. The initial round only performs AddRoundKey with the State, and the final round the MixColumn does not apply. The decryption consists of inverse transformations.

The irreducible polynomial used in AES algorithm is

$$p(x) = x^8 + x^4 + x^3 + x + 1$$

Subbyte transformation operated independently on each byte of the State that uses 16-byte (128-bit) S-Boxes. S-Box is a substitution function whose non-linearity is used to defend against linear cryptanalysis. AES used S-Box which is bijective mapping from eight bits to eight bits [2].

ShiftRows is a linear cyclic shift operation in each row of four 4-byte data blocks with different offset(0~3-byte offsets) to the left. The inverse of this transformation is computed by performing the corresponding rotations to the right. The *MixColumn* transformation operates the 4-byte data blocks on each column of *State* individually. Each column of the *State* matrix is considered as polynomials over GF (2^8) and is multiplied by a fixed matrix M where bytes are treated as a polynomial of degree less than 4 and it is defined as:

$$M = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}$$

For the InvMixcolumn, each column is multiplied with b(x)= {0B}x^3 +{0D}x^2+{09}x+{0E} modulo x^4+1.

AddRoundKey is a simple bit-wise XOR operation on the 128-bit round keys and the data in each round of the AES process. Round Keys are generated by a Key Scheduling process. The initial roundkey derived from the original initial key, and the next roundkey generated by XOR operation between two previous columns. For columns that are in multiples of four, the process involves addition process with round constant, S-Box and rotation.

II. SUBBYTE TRANSFORMATION

SubByte transformation is a nonlinear substitution that operates on individual bytes using a substitution table (S-Box), which contains a permutation of all 256 possible 8-bit values. S-Box is defined as the multiplicative inverse in the finite field GF(2^8) with the irreducible polynomial m(x)=x^8+x^4+x^3+x+1 followed by an affine transformation.

The operation of the S-box can be expresses as

y = M * multiplicative_inverse(x) + c,

Where M is an 8x8 binary matrix and C is a 8-bit binary vector.

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and c=[0 1 1 0 0 0 1 1]

The inverse equation is x = multiplicative_inverse^{-1}(M^{-1}*(y+c)).

x = multiplicative_inverse(M^{-1}* (y+c))

Where M^{-1} = $\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

There are different implementations of AES S-box using composite field arithmetic, direct mapping from lookup table or using combinational logic only. Composite field arithmetic can reduce the area of design, however it increase the critical path delay.

III. S-BOX IMPLEMENTATION USING LOGIC OPTIMIZATION BASED ON TRUTH TABLE

The proposed S-Box design employs combinational logic to solve the unbreakable delay incurred by look-up table and reduces the critical path delay by using composite field arithmetic. The S-box has 8 bit input and 8 bit output. The first 4 bit data input of most significant bit (MSB) will be the input of the sixteen module logic function (M1, M2, M3… M16) derived using Boolean simplification based on Karnaugh map. Another 4 bit data of least significant bit (LSB) will be the selection input of 16 to 1 multiplexer that will derive the output for S-box. This architecture can be used for SubByte Transformation. The proposed S-Box architecture is

illustrated in Figure 1 below.

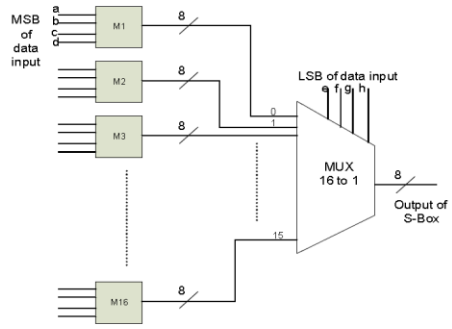


Figure 1. S-Box architecture using combinational logic

There are sixteen module logic functions, each of the module logic function consists of AND gate, NOT gate and OR gate. For example, the Boolean expression for Module 1(M1) as follows:

$$y7 = b\bar{c}\bar{d} + ab\bar{c} + ab\bar{d} + \bar{a}bcd,$$
$$y6 = \bar{a} + \bar{b}c\bar{d} + b(\bar{c}d),$$
$$y5 = ac + \bar{d} + \bar{a}\,\bar{c} + \bar{b}c,$$
$$y4 = \bar{a}\bar{b}(c + d) + \bar{c}\bar{d}(a + b) + abd,$$
$$y3 = \bar{a}\bar{c}d + bc\bar{d} + \bar{b}cd + abd,$$
$$y2 = ab\bar{c} + \bar{b}c\bar{d} + \bar{a}\,\bar{b}\bar{c}d + \bar{a}bc + abd,$$
$$y1 = b\bar{c} + \bar{b}c + \bar{a}\bar{d} + ab,$$
$$y0 = c\bar{d} + \bar{b}c + \bar{a}\bar{b}\,\bar{d} + \bar{a}bd + a\bar{c}d,$$

Figure 2 illustrates the internal architecture for output y7 of Module 1 (M1).The architecture of Module 1(M1) is shown in Figure 3.

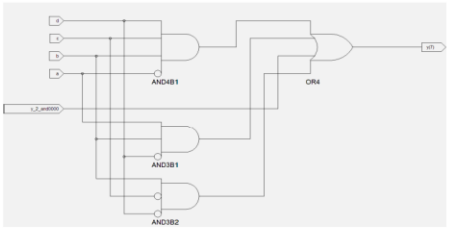


Figure 2. Architecture for output y7 of M1

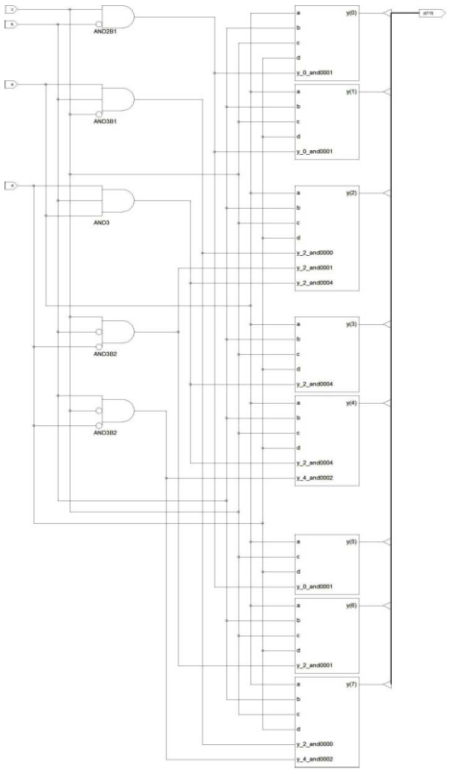


Figure 3. Architecture of Module 1(M1)

IV. RESULT

The architecture in Figure 1 is implemented on a Xilinx Virtex 2 XC2V1000 FPGA using VHDL programming. Xilinx ISE 9.1i software is used to synthesize the design. The results are expressed in term of area and delay. The total equivalent gate count for the proposed S-Box is 1650 gates and 153 slices out of 5120 slices. The maximum combinational path delay for the design is 10.802 ns. Table 1 shows the implementation results.

Table 1. Implementation Results

FPGA Device	Xilinx Virtex 2 XC2V1000
Maximum combinational path delay	10.802 ns
Total equivalent gate count for design	1650
Number of occupied slices	153/5120 2%

Table 2 compares the S-Box design with LUT and composite field implementation using the same FPGA technology. It is observed that our proposed S-Box has efficient area consumption compared to both of the other approaches. It is show that the delay of proposed S-Box is 73% less than the composite field approach.

Table 2. Architecture Comparison

Architecture	Delay(ns)	No of slices	Total no. of gates (gates)
Proposed	10.802	153	1650
Composite field [6]	14.653	380	-
LUT[6]	7.745	1024	-

Figure 4 represent the waveform generated by S-Box design in SubByte Transformation.

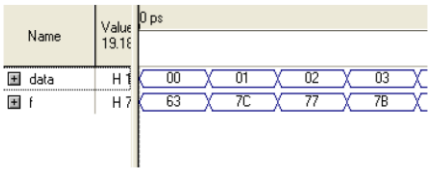


Figure 4. Waveform of SubByte Transformation

V. CONCLUSION

The proposed S-Box gives another option for hardware implementation other than composite field to represent Subbyte transformation. It reduces the complexities of hardware by avoiding the use of multiplicative inverse in Galois field. As compared to the LUT and composite field, the S-Box resulted in smaller area with medium delay. Further work will be carried out to apply the proposed design using ASIC implementation.

ACKNOWLEDGMENT

The authors would like to thank Dr Shaun Cooper from IIMS for his valuable discussions on Galois Field. This work is supported in part by the UTHM (Universiti Tun Hussein Onn Malaysia).

REFERENCES

[1] J. Daemen and V. Rijmen, *The Design of Rijndael*: Springer-Verlag New York, Inc., 2002.

[2] S. Tillich, M. Feldhofer, T. Popp, J. Gro, "Area, delay, and power characteristics of standard-cell implementations of the AES S-Box," *J. Signal Process. Syst.*, vol. 50, pp. 251-261, 2008.

[3] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*: Springer-Verlag, 2001.

[4] N. Sklavos and O. Koufopavlou, "Architectures and VLSI implementations of the AES-Proposal Rijndael," *Computers, IEEE Transactions on*, vol. 51, pp. 1454-1459, 2002.

[5] Z. Xinmiao and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 12, pp. 957-967, 2004.

[6] R. R. Rachh and P. V. Ananda Mohan, "Implementation of AES S-Boxes using combinational logic," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, 2008, pp. 3294-3297.

[7] C. Nalini, P. V. Anandmohan, D. V. Poornaiah, and V. D. Kulkarni, "Compact Designs of SubBytes and MixColumn for AES," in *Advance Computing Conference, 2009. IACC 2009. IEEE International*, 2009, pp. 1241-1247.