國立交通大學

電子工程學系 電子研究所

博 士 論 文

具差動功率攻擊防禦之先進加密標準核心設計與安全性分析

Design and security analysis of DPA resistant AES cryptographic engine

研 究 生：劉柏均

指導教授：李鎮宜 教授

張錫嘉 教授

中 華 民 國一〇〇 年 十二 月

具差動功率攻擊防禦之先進加密標準核心設計與安全性分析

Design and security analysis of DPA resistant AES cryptographic engine

研 究 生：劉柏均　　Student：Po-Chun Liu

指導教授：李鎮宜　　Advisor：Chen-Yi Lee

張錫嘉　　Hsie-Chia Chang

國 立 交 通 大 學

電子工程學系 電子研究所

博 士 論 文

A Dissertation

Submitted to Department of Electronics Engineering and
Institute of Electronics
College of Electrical and Computer Engineering
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Electronics Engineering

December 2011
Hsinchu, Taiwan, Republic of China

中華民國一〇〇年十二月

具差動功率攻擊防禦之先進加密標準核心設計與安全性分析

學生：劉柏均　　指導教授：李鎮宜 教授
張錫嘉 教授

國立交通大學
電子工程學系
電子研究所

摘要

先進加密標準(AES)目前已是最為廣泛使用的對稱性加密演算法，其最主要的原因在於高安全性、高效能及以實現低複雜度。舉凡無線通訊系統、資料儲存系統、智慧型晶片卡以及銀行系統等等都大量地採用 AES 為加密標準。現有文獻已有不少 AES 硬體實作的探討、但是這些文獻往往未將低成本同時具備高效能的設計技巧考量進去。因此在本論文我們先探討在不同需求的應用上(包含超高速應用以及超低成本應用)的架構設計。綜合以上設計概念，我們提出了一個高效益的 AES 硬體架構。而此硬體架構可以支援標準所訂的三種金鑰長度以及加/解密能力。整體的架構主要是由一高度化簡之即時金鑰產開元件以及一高度整合之加/解密資料處理元件所組成。即時金鑰產開元件共用了不同金鑰長度所需要的硬體資源來大幅降低成本；而整合資料處理元件則是共用了加密與解密所需要的資料。在透過矽晶片實作以及量測後，同時透過矽晶片之實作與量測，驗證所提之方法之分析與效益。

然而，AES 演算法的硬體實作存在著一個相當大的安全性風險：差動功率攻擊。此種攻擊法可以很有效率地破解出 AES 晶片運算時所使用之金鑰。在本論文中，我們也進一步探討防禦此種攻擊法的方法。透過一基於數位振盪器以及亂數產生品的防禦電路，我們提出了一低成本且高效能的 AES 晶片來抵抗差動功率攻擊。相較於現有之文獻，

我們所提出來的方法無論是在額外成本支出或是效能降低比例都能大幅的改善。而透過矽晶片的實作與量測，我們所提出的抗差動功率攻擊的 AES 晶片可以達到最快 255MHz 的操作頻率，而在此操作頻率下的效能為 2.97Gb/s。同時防禦電路所需要的額外成本支出僅為原本 AES 電路的 6.2%。在安全度分析下，measurement to disclosure (MTD)可以從數千大幅增加至 10^7，至少增加了三個數量級以上的安全度。

Design and security analysis of DPA resistant AES cryptographic engine

Student: Po-Chun Liu

Advisors: Dr. Chen-Yi Lee
Dr. Hsie-Chia Chang

Department of Electronics Engineering
Institute of Electronics
National Chiao Tung University

ABSTRACT

The AES algorithm approved in 2001 has become the most popular symmetric-key encryption algorithm because of its high security, high performance, and low complexity. The AES algorithm is widely adopted in numerous applications such as wireless communications, storage devices, smart cards, or banking systems. Several implementations have been published but few of them considered the hardware cost and the throughput as a whole. In this dissertation, we first investigate architectures for high throughput and low cost applications. At last a cost efficient AES architecture, which is capable of both encryption and decryption with three different key lengths, is presented for high speed mobile applications. The overall hardware cost is optimized by a very compact on-the-fly key expansion unit and a highly integrated encryption/decryption data-path. The compact on-the-fly key expansion unit is achieved by sharing key scheduling processes of different key lengths. The integrated data-path shares hardware resources used in encryption and decryption. After manufactured in 90nm CMOS technology, the area of the chip is 15,577 equivalent gates with throughput up to 1.69 Gb/s operating at 131.8 MHz.

However, the hardware implementation of the AES algorithm is still vulnerable to side-channel attacks. The differential power analysis (DPA) attack is an efficient and low cost

method to disclose the secret key of the AES chip. In this dissertation, a low cost AES crypto core with resistance to the DPA attack is presented by exploiting a DPA countermeasure circuit based on digital ring oscillators and an on-chip random number generator (RNG). Two architectures with pseudo random and digital random number generator are presented. Compared with previous works that counteract the DPA attack by using data masking circuits or equalizing the power consumption, our proposed DPA countermeasure circuit can significantly reduce the area overhead without throughput degradation. The DPA resistant AES engines are fabricated in UMC 90 nm CMOS technology. For the pseudo random based architecture, the AES chip can achieve 2.76 Gb/s throughput at operating frequency of 237 MHz. The area overhead is minimized to 10.2%. For the digital random based architecture, the AES chip can achieve 2.97 Gb/s throughput at operating frequency of 255 MHz. The area overhead is slightly improved from 10.2% to 6.2% by resource sharing between the DPA countermeasure circuit and the random number generator. The digital random based architecture further resolves the “reset” problem, which may induce a security issue for the PRNG based architecture. The measurement to disclosure (MTD) of both AES engines is increased from several thousands to more than 10^7 measurements, indicating the security level is enhanced by at least three orders of magnitude.

誌　　謝

從開始寫下誌謝的這一刻起，正式宣告了我即將脫離了學生的生活。四年多前不知從那裡冒出來的念頭，沒有選擇跟大部分的同學一樣去就業，而是報考了博士班入學的考試。這四年來的學習路程是跟以往大大不同的，在博士班的生活裡，我必須自己發掘有研究價值的題目，而不是像從前一樣只要等著老師或學長丟題目來完成就好了，也因此在博士班的求學過程中也曾經遇到了瓶頸而一度想放棄。在這裡我必須要好好感謝我的指導老師李鎮宜教授以及張錫嘉教授，在我一度迷失方向的時候指引我並提供了豐富的資源。另外我也要感謝我的口試委員吳誠文教授、吳安宇教授、謝明得教授、魏慶隆教授、周世傑教授、陳榮傑教授、郭峻因教授以及賴伯承教授在百忙之中抽空參與我的口試，並且提供給學生諸多寶貴的意見，讓學生獲益良多。

再來我也要感謝交通大學 SI2 實驗室跟我一起打拼的伙伴們，讓我在這一段的求學生涯裡能過得相當精采。尤其我要特別感謝建青學長，當初沒有他的帶領我也不會進到 SI2 進行密碼學相關的研究，而在研究過程中多次跟學長討論也學習到很多知識以及研究方法。再來我要感謝 STAR group 的成員：大嘴、Q 毛、廷聿、耀琳、勇志、小約、靜瑜、星萍以及期赫，有你們的付出才有我們共享的研究成果。其他我也要感謝這一路來陪伴我的阿龍、義閔、小肥、佳龍、欣儒、許智翔、小朱哥、長宏還有其他同學們，因為有你們所以我的生活更豐富。最後還要謝謝美麗的子菁以及伶霞助理，幫我處理了很多行政上的問題。

當然，最後不免俗的要謝謝我的父母，他們提供了我最佳的避風港，在我有任何困難的時候都能尋求他們的幫助；而我兩位姐姐也在生活上提供了我許多的幫忙。另外，我還要感謝我女朋友這十年來的陪伴與鼓勵，讓我最終能完成這本博士論文。

Contents

1 Introduction **1**
- 1.1 Overview of Cryptographic Systems . . . 1
- 1.2 Motivation . . . 3
- 1.3 Dissertation Organization . . . 4

2 Fundamentals of the AES algorithm **6**
- 2.1 The AES algorithm . . . 6
  - 2.1.1 Mathematical Preliminaries . . . 7
  - 2.1.2 SubBytes and Inv-SubBytes . . . 11
  - 2.1.3 ShiftRows and Inv-ShiftRows . . . 14
  - 2.1.4 MixColumns and Inv-MixColumns . . . 14
  - 2.1.5 AddRoundKeys . . . 16
  - 2.1.6 Key Expansion . . . 16
- 2.2 Modes of operations for Block Ciphers . . . 18
  - 2.2.1 Cipher Block Chaining (CBC) Mode . . . 19
  - 2.2.2 Cipher Feedback (CFB) Mode . . . 20
  - 2.2.3 Output Feedback (OFB) Mode . . . 21
  - 2.2.4 Counter (CTR) Mode . . . 23
  - 2.2.5 Counter with CBC-MAC (CCM) Mode . . . 24
- 2.3 Applications of the AES algorithm . . . 26
  - 2.3.1 Wireless Communications . . . 26
  - 2.3.2 Network Protocols . . . 27
  - 2.3.3 Tamper Resistant Applications . . . 27

3 Design of AES Crypto Engines **29**
- 3.1 Previous Works on the AES algorithm . . . 29
  - 3.1.1 SubBytes Transformation . . . 30
  - 3.1.2 MixColumns Transformation . . . 37
- 3.2 High Throughput AES Engine for 100 Gigabit Ethernet . . . 39
  - 3.2.1 Data-path Unit . . . 39
  - 3.2.2 Key Expansion Unit . . . 42
  - 3.2.3 Implementation Results . . . 44
- 3.3 Low Cost AES Engine for Smart Cards or RFIDs . . . 46
  - 3.3.1 Data-path Unit . . . 47
  - 3.3.2 Key Expansion Unit . . . 51
  - 3.3.3 Implementation Results . . . 52
- 3.4 Median Throughput AES Engine for WLAN . . . 53
  - 3.4.1 Data-path Unit . . . 54

3.4.2 Key Expansion Unit . . . 56
3.4.3 Implementation Results . . . 59
3.5 Summary . . . 61

4 Power Analysis Attacks **63**
4.1 Simple Power Analysis . . . 64
4.1.1 General Description . . . 64
4.1.2 SPA on Asymmetric Ciphers . . . 65
4.1.3 SPA on Symmetric Ciphers . . . 66
4.2 Differential Power Analysis . . . 66
4.2.1 DPA Attack Flow . . . 67
4.2.2 Power Models . . . 68
4.2.3 Statistical Analysis . . . 69
4.3 Security Evaluation . . . 71
4.3.1 Measurement Environment . . . 71
4.3.2 Power Models . . . 73
4.3.3 DPA Results . . . 74
4.3.4 Accessing the Number of Needed Power Traces . . . 78

5 Design of DPA Resistant AES Engines **81**
5.1 Previous Works on DPA Countermeasure . . . 81
5.1.1 Power Masking Methods . . . 82
5.1.2 Power Hiding Methods . . . 85
5.2 Pseudo Random Based DPA Countermeasure Circuit . . . 88
5.2.1 Ring Oscillator Based DPA Countermeasure Circuit . . . 88
5.2.2 Dynamic Pseudo Random Number Generator . . . 92
5.2.3 Security Evaluation on SubBytes . . . 93
5.3 True Random Like Based DPA Countermeasure Circuit . . . 95
5.3.1 Security Issue on PRNG . . . 96
5.3.2 DPA Countermeasure with True Random Like Sequence . . . 98

6 Implementation Results and Comparison **103**
6.1 Chip Implementation Results . . . 103
6.1.1 Pseudo Random Based Architecture . . . 103
6.1.2 True Random Like Based Architecture . . . 106
6.2 Security Analysis Results . . . 107
6.2.1 Pseudo Random Based Architecture . . . 107
6.2.2 True Random Like Based Architecture . . . 109
6.3 Comparison . . . 111

7 Conclusion and Future Works **114**
7.1 Conclusion . . . 114
7.2 Future Works . . . 116

Bibliography **118**

List of Tables

2.1 The number of rounds required for different key sizes 7

3.1 Comparison between AES engines over 10 Gb/s 46
3.2 Comparison between low cost AES engines 53
3.3 Comparison between median throughput AES engines 60
3.4 Design summary on different AES architectures 62

4.1 Quantiles $z_{1-\alpha}$ of the normal distribution for different $1-\alpha$. 79
4.2 The estimated number of traces for different $\rho_{ck,ct}$. 79

6.1 Chip summary of pseudo random based architecture 104
6.2 Chip summary of true random like based architecture 106
6.3 Comparison for hardware cost 111
6.4 Comparison for throughput 112
6.5 Comparison for power 112
6.6 Comparison for figure of merit 113

List of Figures

1.1 The illustration of the general cryptographic system. 2

2.1 The data block arrangement. 7
2.2 The data flow of the AES algorithm. (a) Encryption. (b) Decryption. 8
2.3 The table for the SubBytes transformation. 12
2.4 The table for the Inv-SubBytes transformation. 13
2.5 The Inv-/SubBytes transformation on the AES data block. 14
2.6 The ShiftRows transformation on the AES data block. 14
2.7 The Inv-ShiftRows transformation on the AES data block. 15
2.8 The Inv-/MixColumns transformation on the AES data block. 16
2.9 The AddRoundKeys transformation on the AES data block. 16
2.10 The encryption of the ECB mode. 18
2.11 The data flow of the CBC mode. (a) Encryption. (b) Decryption. 19
2.12 The data flow of the CFB mode. (a) Encryption. (b) Decryption. 21
2.13 The data flow of the OFB mode. (a) Encryption. (b) Decryption. 22
2.14 The data flow of the CTR mode. (a) Encryption. (b) Decryption. 23
2.15 The data flow of the CCM mode. (a) Encryption. (b) Decryption. 25

3.1 Polynomial basis inverter (a) Over $GF(2^8)$. (b) Over $GF(2^4)$. 32
3.2 Normal basis inverter in $GF(2^8)$. 34
3.3 Normal basis multiplier in $GF(2^4)$. 34
3.4 Inv-/SubBytes sharing structure. 36
3.5 Implementation results of different SubBytes architecture. 41
3.6 Data path for 10 Gb/s throughput. 41
3.7 Data path for 50 Gb/s throughput. 42
3.8 The off-line key expansion unit. 43
3.9 Implementation results for 10 Gb/s throughput. 44
3.10 Implementation results for 50 Gb/s throughput. 45
3.11 8-bit data-path for low cost AES engine. 47
3.12 The low cost SubBytes transformation. 48
3.13 The processing order of data bytes after ShiftRows transformation. 48
3.14 The operation of the ShiftRows transformation for the 8-bit data-path. . . . 49
3.15 The operation of the MixColumns transformation for the 8-bit data-path. . . 50
3.16 The 8-bit key expansion unit. 52
3.17 Implementation results of low cost AES engine. 52
3.18 Block diagram of the median throughput AES architecture 54
3.19 Architecture of integrated data process unit 55
3.20 The data flow of encryption and decryption process. 55
3.21 On-the-fly key expansion unit for median throughput 57

3.22 Implementation results of median throughput AES engine. 59
3.23 Die micrograph for median throughput AES engine 59
3.24 Shmoo plot for median throughput AES engine 60
3.25 Summary of area throughput trade-offs of AES engines. 61

4.1 The power trace of a straightforward RSA implementation. 65
4.2 The power trace of an AES implementation. 66
4.3 The flow of the DPA attack. 67
4.4 The block diagram of the measurement environment. 72
4.5 The test chip measurement and analysis environment setup. 73
4.6 The last round model for power simulation. 74
4.7 The design and estimation flow for the AES engine. 75
4.8 The DPA results based on difference-of-means. 76
4.9 The DPA results based on correlation coefficient. 77
4.10 The DPA attack results on real AES chip. 78

5.1 The concept of the masking methods in algorithm level. 83
5.2 The modification of non-linear parts in AES algorithm. 83
5.3 The modified SubBytes by Akkar and Giraud. 83
5.4 The masked-AND logic. 85
5.5 The wave dynamic differential logic. 86
5.6 The differential routing technique. 86
5.7 The switching capacitors. 87
5.8 Block diagram of the pseudo random based DPA-resistant AES chip. 88
5.9 Detailed structure of the DPA countermeasure sub-circuit. 89
5.10 The power traces of ring oscillators with different inversion stages. 90
5.11 The power distribution of AES engines. 91
5.12 The block diagram of the on-chip dynamic PRNG. 93
5.13 DPA results for LUT based SubBytes. 94
5.14 DPA results for composite field based SubBytes. 94
5.15 Block diagram of true random like based DPA-resistant AES engine. 95
5.16 Power distributions of the unprotected AES engine. 97
5.17 Power distributions of the pseudo random based architecture. 97
5.18 Power distributions of DPA-resistant AES engine with reset. 98
5.19 (a) Fibonacci ring oscillator (b) Galois ring oscillator. 99
5.20 DPA countermeasure circuit with true random like sequence. 99
5.21 The randomness analysis of random sequence. 100
5.22 Power distributions of true random like based architecture. 101

6.1 Power breakdown of the test chip. 105
6.2 The die micrograph of pseudo random based test chip. 105
6.3 The die micrograph of true random like based test chip. 107
6.4 Power traces of the test chip for one encryption operation. 108
6.5 DPA results for pseudo random based test chip. 108
6.6 Power traces of the test chip for one encryption operation. 109
6.7 DPA results for true random like based test chip. 110

Chapter 1

Introduction

Cryptographic algorithms are used to provide authentication, confidentiality, or data integrity in several applications such as communication systems, storage systems, or banking systems. Authentication provides the assurance that data is originated from a particular party; confidentiality ensures that data is readable by authorized parties; and data integrity makes sure that received data is not tampered by unauthorized parties.

1.1 Overview of Cryptographic Systems

A general cryptographic system is shown in Fig. 1.1. The user data (plain-text) is encrypted by a series of transformations or mathematical operations with an (unique) encryption key E_k. The encrypted data (cipher-text) is then unreadable by unauthorized third parties. Only authorized parties, who possess the corresponding decryption key D_k, can recover the user data by a series of inverse transformations or mathematical operations. Note that the security of a cryptographic system is based on the secret key but not encryption/decryption algorithms. That is, the cipher-text is still unreadable if the encryption algorithm but not the secret key is recognized by third parties. Depending on the property of encryption/decryption secret keys, cryptographic systems can be categorized into two types: symmetric-key and asymmetric-key cryptographic algorithms.

Symmetric-key cryptographic algorithms use the same secret key to encrypt and decrypt

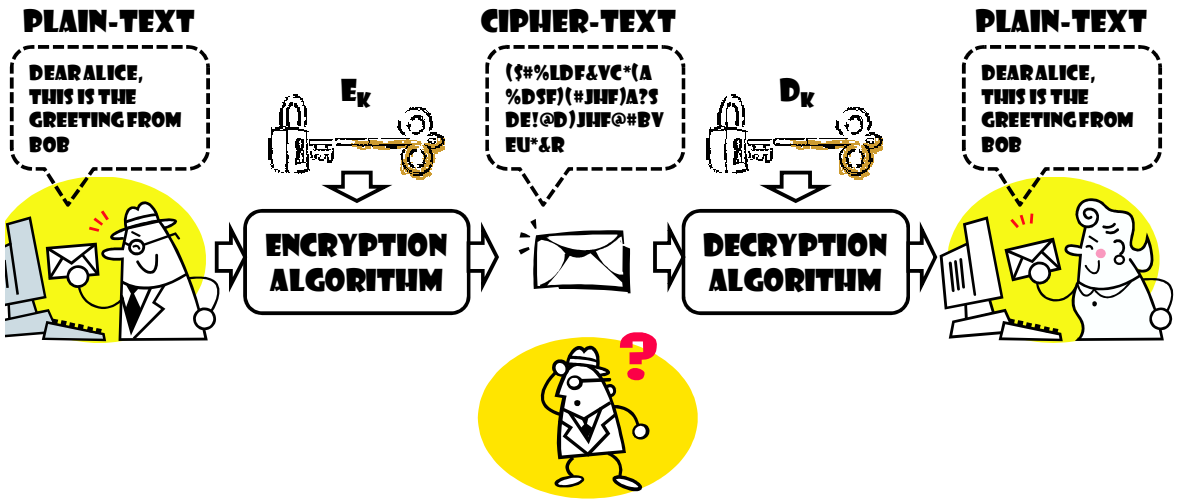


Figure 1.1: The illustration of the general cryptographic system.

user data. Authorized parties must possess the same secret key to exchange private information. Symmetric-key algorithms can be further categorized into stream ciphers [1–7] and block ciphers [8–13] by the encryption/decryption flow. Stream ciphers use the secret key as a ”seed” to generate one unique key stream and cipher-texts are generated by bit-wise XOR the key stream and plain-texts. To decrypt cipher-texts, the same secret key is used to generate the same key stream and the user data can be recovered by bit-wise XOR operation. Unlike stream ciphers, block ciphers incorporate the secret key into plain-texts through a series of transformations and permutations for several iterations. To decrypt cipher-texts, these transformations and permutations are processed in the reverse order to recover plain-texts. The most attractive feature of symmetric-key cryptographic algorithms is low complexity and high performance. However, the most critical problem is that both parties should possess the same secret key. If the secret key is intercepted or stolen by third parties, adversaries can use the secret key to recover the encrypted data.

The Diffie-Hellman key exchange is the earliest method to exchange secret key between two parties [14]. A random shared secret key after the key exchange protocol thus can be used for the subsequent symmetric-key cipher. However, the shared secret key is randomly generated and can not be determined by any one of the two parties. Asymmetric-key cryptographic algorithms such as the RSA [15] and elliptic curve cryptography (ECC) [16, 17] can encrypt the data without key exchange protocol. Instead of using the same key for en-

cryption and decryption, asymmetric-key algorithms use a public-key for encryption (E_k) and a private-key for decryption (D_k). The public-key and private-key are related by complicated mathematical arithmetic such that the private-key can not be easily derived from the public-key. The public-key is known to any parties and the private-key is only possessed by a specific user. Therefore, others who do not have the private-key can not decrypt cipher-texts.

In most cryptographic systems, symmetric-key algorithms are usually used for data encryption and asymmetric-key algorithms are used for key exchange or digital signature. For most communication or storage systems, secret keys for symmetric-key algorithms are infrequently changed and asymmetric-key algorithms are less frequently used than symmetric-key algorithms. Therefore, more discussions and analysis are addressed on symmetric-key algorithms in this dissertation.

The Advanced Encryption Standard (AES) [8], one of block ciphers approved by the National Institute of Standards and Technology (NIST) in 2001, has become the most widely used symmetric-key algorithm due to its low complexity and high performance. However, security weakness of silicon based crypto engine implementation is first investigated by Kocher *et al.* [18, 19]. Instead of using traditional cryptanalysis methods, Kocher uses the leakage information such as operation timing or power consumption to disclose secret keys of cryptographic devices. With such kind of timing analysis or power analysis, secret keys can be easily disclosed in a few days with little cost. Both AES architecture design and power analysis resistant techniques will be covered in this dissertation.

1.2 Motivation

To date, several AES hardware implementations have been proposed to provide high throughput [20–33] or low cost [34–39] solutions for different applications. The most common methods for high throughput AES are loop unrolling and pipelining. The pipelining technique can increase the maximum operating frequency and the loop unrolling technique can increase the level of parallelism. For AES designs that can achieve throughput over Gb/s

usually require more than 50,000 equivalent gates (2-input NANDs). On the other hand, low cost AES designs can be achieved by reducing the width of data-path from 128-bit to 32-bit or even 8-bit. However, throughput for such AES designs can only achieve several hundred Mb/s. Since the AES algorithm is more commonly adopted than other block ciphers, we would like to investigate the implementation of the AES algorithm for different applications. Architectures for ultra high throughput and ultra low cost AES cores are the very first step in this research. The most important goal is to design an area efficient AES cryptographic core that can be used not only for high throughput applications but also for area limited applications.

Furthermore, since cryptographic algorithms are used in security related applications, secret key attack methods must be considered carefully. Different power analysis attack methods are investigated because the power information can be easily obtained by existing equipments. For the simple power analysis (SPA) [19], attackers observe a single power trace of cryptographic devices to disclose secret keys. However, since the SPA utilizes key dependent characteristics of the power trace to disclose secret keys, this kind of attack is more suitable to attack asymmetric cryptographic algorithms. For symmetric cryptographic algorithms, the operation flow is independent of the key value; therefore, a stronger analysis method must be used to disclose secret keys. The differential power analysis (DPA) [19] attack uses the statistical analysis between the measured power consumption and the predicted power consumption to disclose secret keys. It has been shown in several literatures that the secret key of an AES cryptographic chip can be disclosed within 10,000 measurements [40–42]. Therefore, how to protect an AES engine from power analysis attacks will also be discussed in this dissertation.

1.3 Dissertation Organization

To have an overall inspection on the research topic, an overview of cryptographic systems and motivations are introduced in the first chapter. In chapter 2, fundamentals of the AES al-

gorithm are introduced, including specifications of the AES algorithms and operation modes for different applications. Since the AES algorithm is widely adopted, applications for the AES algorithm are also introduced in this chapter. The design and implementation of the AES algorithm is given in chapter 3. Previous works on the AES algorithm are first introduced and three different architectures for different design considerations are proposed. The motivation of this chapter is to give a thorough analysis on the architecture design of AES for different applications. Currently most publications focus only on a specific architecture for a specific application. We want to establish an implementation methodology for the AES for different applications with different considerations. As a result, implementation considerations for high throughput, median throughput and low cost are discussed with a corresponding architecture design. At last, a brief summary on architectures of the AES algorithm is given at the end of this chapter.

Chapter 4 illustrates common power analysis attacks for cryptographic systems, especially for symmetric-key cryptographic algorithms. Details of the SPA and DPA and attack results on an AES engine are presented in chapter 3 to demonstrate the efficiency of such power analysis attacks. Chapter 5 shows the design and DPA resistant methods for the AES cryptographic engine. Previous works on DPA countermeasure methods are first introduced and disadvantages of these works are discussed. The existing countermeasure methods usually result in huge area overhead and throughput degradation and we present architectures that optimize in terms of area overhead and throughput degradation while still increasing the security ability. Details of proposed architectures for DPA countermeasure are illustrated. A brief summary on these methods is give at the end of this chapter.

Chip implementation results of the AES cryptographic engine and the DPA countermeasure methods are given in chapter 6. In addition to physical implementation results, the DPA attack environment and real chip attack results are also given in this chapter to show the DPA resistant capability of proposed architectures. At last, chapter 7 briefly concludes this work and gives some potential research directions on the security analysis of AES cryptographic engines.

Chapter 2

Fundamentals of the AES algorithm

To be capable of protecting the top secret information for the twentieth century, the National Institute of Standard and Technology (NIST) worked on developing a new encryption standard. In 1997 they asked for candidates for the new algorithm. Five of them are adopted after two selection meetings, including the Rijndael, MARS, RC6, Serpent, and Twofish. The NIST officially announced that the Rijndael algorithm [43] is selected as the AES algorithm in 2000 and published the standard as FIPS PUB 197 [8] on November 26, 2001.

2.1 The AES algorithm

The AES algorithm is a symmetric-key block cipher that can encrypt data into an unintelligible form called cipher-text, and decrypt the cipher-text back into the original form called plain-text. The AES algorithm processes data blocks of 128 bits with three different key lengths, 128, 192, and 256 bits to provide different levels of security. A 128-bit data block in the AES algorithm is arranged as a 4-by-4 array called state illustrated in Fig. 2.1. In addition, the AES algorithm is also a kind of iterative cipher, which means that data blocks are repeatedly processed by the same function for several times. The number of rounds to be performed is dependent on the key size as listed in Table 2.1.

Round function of the AES algorithm is composed of four different transformations: 1) SubBytes, 2) ShiftRows, 3) MixColumns, and 4) AddRoundKeys. The data flow for the

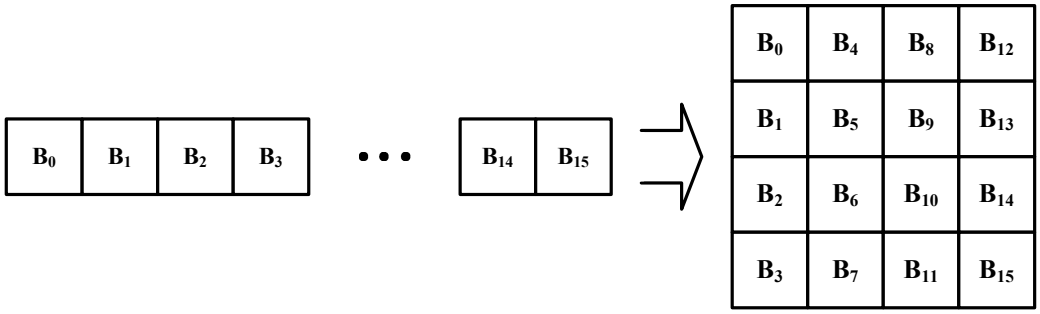


Figure 2.1: The data block arrangement.

Table 2.1: The number of rounds required for different key sizes

	Key size (bits)	Block size (bits)	Number of rounds
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

encryption process is shown in Fig. 2.2(a). The secret key is added into the data block in the initial round, and then the data block is applied to the round function for Nr rounds depending on the key size. Note that the MixColumns is skipped in the final round. The decryption flow is exactly in the reverse order of the encryption flow as shown in Fig. 2.2(b). All transformations except AddRoundKeys are in the inverse form with prefix Inv-. Details of mathematical preliminaries and these transformations are given in following subsections. Furthermore, round keys for every different round are derived from the secret key with the key expansion algorithm. The key expansion algorithm is also illustrated at the end of this section.

2.1.1 Mathematical Preliminaries

Fundamental operation of the AES algorithm is arithmetics in finite field $GF(2^8)$ with irreducible polynomial of degree 8. For the AES algorithm, the irreducible polynomial is defined as

$$m(x) = x^8 + x^4 + x^3 + x + 1, \tag{2.1}$$

or 011b in hexadecimal notation. The data block can be divided into 16 bytes and each byte can be represented as an element in this field. Therefore, all arithmetic operations such

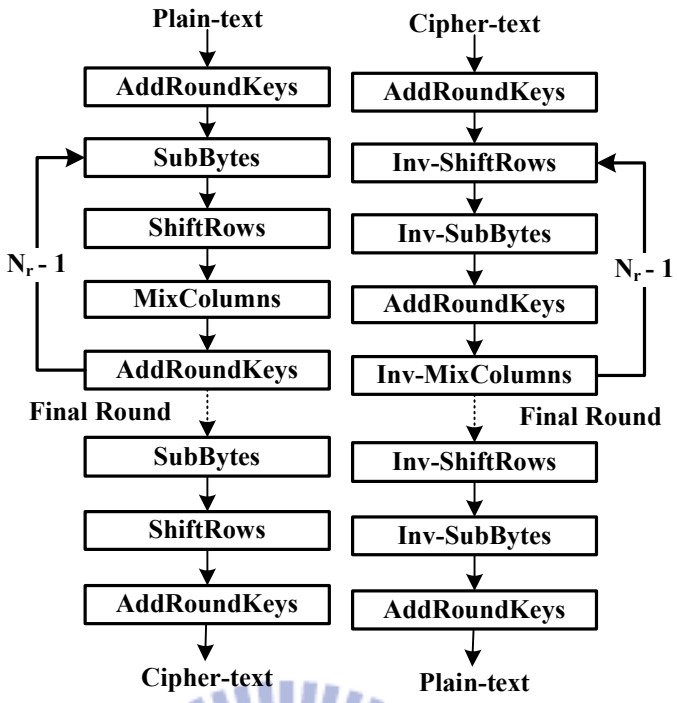


Figure 2.2: The data flow of the AES algorithm. (a) Encryption. (b) Decryption.

as addition and multiplication are performed under field $GF(2^8)$. The basic mathematical arithmetics for the AES algorithm are introduced as follows.

Additions

The addition of two elements in finite field can be achieved by adding coefficients of terms with the same degree. The addition is performed under $GF(2)$, that is, the addition is a simple XOR operation so that $1 + 1 = 0 + 0 = 0$, $1 + 0 = 0 + 1 = 1$. Since the addition is modulo 2 operation, the subtraction is identical to the addition operation because $(-1) = 1\ (mod\ 2)$.

Alternatively, elements in $GF(2^8)$ can be represented in the binary form $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ and $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$. The addition of these two bytes can be done by bit-wise XOR operation, that is $c_i = a_i \oplus b_i, i = 0 - 7$.

Multiplications

The multiplication of two elements can be done by multiplication of polynomials modulo the irreducible polynomial $m(x)$. Since the degree of $m(x)$ is 8, the modulo operation makes sure that the resulting product of two polynomials has degree less than 8 and can be represented by one byte. Instead of using long division algorithm, the modulo reduction can be facilitated by the finite field theorem. All the elements must satisfy the equation $m(x) = 0$, in other words, $x^8 = x^4 + x^3 + x + 1$. In this way, the resulting terms with degree lager than 8 can be decomposed into $x^8 x^d$ and represented as $(x^4 + x^3 + x + 1)x^d$. The modulo reduction then can be simplified to a series of decomposition and addition.

The multiplicative inverse of a non-zero element $b(x)$ in $GF(2^8)$ can be computed by the extended Euclidean algorithm [44]. The polynomial $a(x)$ and $c(x)$ are computed such that

$$b(x)a(x) + m(x)c(x) = 1. \tag{2.2}$$

As a result, $a(x)b(x) \equiv 1 \; mod \; m(x)$, that is,

$$b^{-1}(x) = a(x) \; mod \; m(x). \tag{2.3}$$

The commutative property, the associative property, and the distributive property are also held in the specified field. That is, following equations must be held in this field:

$$\begin{aligned} a(x) \times b(x) &= b(x) \times a(x) \\ (a(x) \times b(x)) \times c(x) &= a(x) \times (b(x) \times c(x)) \\ a(x) \times (b(x) + c(x)) &= a(x) \times b(x) + a(x) \times c(x) \end{aligned} \tag{2.4}$$

In $GF(2^8)$, the multiplication by x can be simplified into a conditional addition operation. The multiplication of polynomial $b(x)$ by x results in

$$b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x. \tag{2.5}$$

As mentioned earlier, the term x^8 is equivalent to $x^4 + x^3 + x + 1$. If the coefficient $b_7 = 0$, then the result is already in the reduced form. If $b_7 = 1$, the reduced form can be obtained by adding $x^4 + x^3 + x + 1$. Therefore, the multiplication by x can be implemented by a left shifter and a conditional addition of $\{00011011\}$. For the AES standard, the multiplication by x is denoted as *xtime()*. Multiplication by higher power of x then can be implemented by recursive property using the *xtime()* operation.

Polynomials with Coefficients in $GF(2^8)$

This subsection illustrates arithmetics in composite field $GF((2^8)^4)$. In this field, the four-term polynomial can be defined as:

$$A(x) = A_3x^3 + A_2x^2 + A_1x + A_0, \tag{2.6}$$

where coefficients $A_i, i = 0 - 3$ are elements in $GF(2^8)$. The addition in composite field $GF((2^8)^4)$ are done by adding coefficients of terms with the same degree. The addition of two four-term polynomials $A(x)$ and $B(x)$ can be expressed as the following equation:

$$A(x) + B(x) = (A_3 + B_3)x^3 + (A_2 + B_2)x^2 + (A_1 + B_1)x + (A_0 + B_0). \tag{2.7}$$

Additions of A_i and B_i are defined in field $GF(2^8)$ and can be performed by bit-wise XOR operation.

The multiplication in the field $GF((2^8)^4)$ can be achieved by two steps: the polynomial multiplication and polynomial reduction. In the first steps, the production of two polynomial $A(x)$ and $B(x)$ is expanded first as follows:

$$C(x) = C_6x^6 + C_5x^5 + C_4x^4 + C_3x^3 + C_2x^2 + C_1x + C_0. \tag{2.8}$$

Note that the maximum degree of resulting $C(x)$ is 6 because the maximum degree of both $A(x)$ and $B(x)$ is 3. To reduce $C(x)$ back to field $GF((2^8)^4)$, polynomial $x^4 + 1$ is used in

the AES algorithm. Therefore, the $C(x)$ can be reduced as follows:

$$C(x) = C'_3x^3 + C'_2x^2 + C'_1x + C'_0 \tag{2.9}$$

with

$$\begin{aligned} C'_0 &= A_0B_0 + A_3B_1 + A_2B_2 + A_4B_3 \\ C'_1 &= A_1B_0 + A_0B_1 + A_3B_2 + A_2B_3 \\ C'_2 &= A_2B_0 + A_1B_1 + A_0B_2 + A_3B_3 \\ C'_3 &= A_3B_0 + A_2B_1 + A_1B_2 + A_0B_3 \end{aligned} \tag{2.10}$$

The multiplication in field $GF((2^8)^4)$ can be written in the matrix form as:

$$\begin{bmatrix} C'_0 \\ C'_1 \\ C'_2 \\ C'_3 \end{bmatrix} = \begin{bmatrix} A_0 & A_3 & A_2 & A_1 \\ A_1 & A_0 & A_3 & A_2 \\ A_2 & A_1 & A_0 & A_3 \\ A_3 & A_2 & A_1 & A_0 \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \tag{2.11}$$

2.1.2 SubBytes and Inv-SubBytes

The SubBytes transformation (also called the S-box) is the only non-linear transformation in the AES algorithm and each byte of the data block is substituted independently. The transformation can be divided into two steps:

1. Find the multiplicative inverse of the data byte over field $GF(2^8)$. The multiplicative inverse of element $\{00\}$ is mapped to itself.

2. Apply the following affine transformation to each bit over $GF(2)$:

$$b'_i = b_i \oplus b_{i+4mod8} \oplus b_{i+5mod8} \oplus b_{i+6mod8} \oplus b_{i+7mod8} \oplus c_i, \tag{2.12}$$

where b_i and c_i is the ith bit of the data byte and a constant vector $\{01100011\}$. The

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

Figure 2.3: The table for the SubBytes transformation.

affine transformation can be expressed in the matrix form as:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{2.13}$$

Since the finite field and the affine transformation for the AES algorithm is pre-defined, the SubBytes transformation can be represented as a look up table shown in Fig. 2.3 [8]. The input data byte is represented as $\{xy\}$ in hexadecimal format, values of x and y are used as indexes for the row and column, respectively, to look up the output data byte. For example, if the data byte is $\{24\}$, then the cross value of the third row and the fifth column $\{36\}$ is then the substituted value.

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

Figure 2.4: The table for the Inv-SubBytes transformation.

The Inv-SubBytes transformation used in the decryption is inverse of the SubBytes transformation and can also be divided into two steps:

1. Apply the inverse affine transformation in the matrix form as:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} b_0 + 1 \\ b_1 + 1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 + 1 \\ b_6 + 1 \\ b_7 \end{bmatrix} \tag{2.14}$$

2. Find the multiplicative inverse of $\{b'_0 b'_1 b'_2 b'_3 b'_4 b'_5 b'_6 b'_7\}$ over $GF(2^8)$.

The Inv-SubBytes transformation can also be represented as a look up table shown in Fig. 2.4 [8].

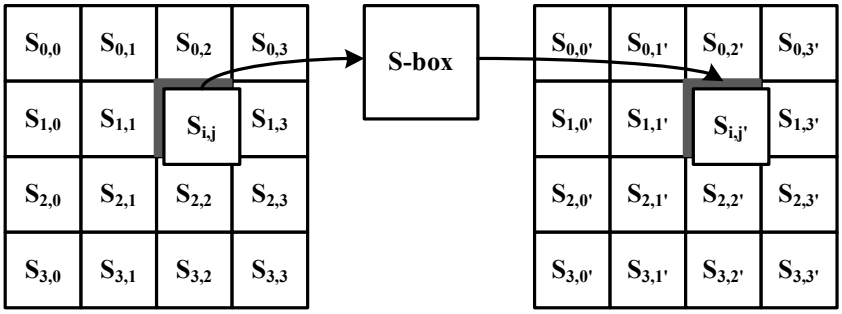


Figure 2.5: The Inv-/SubBytes transformation on the AES data block.

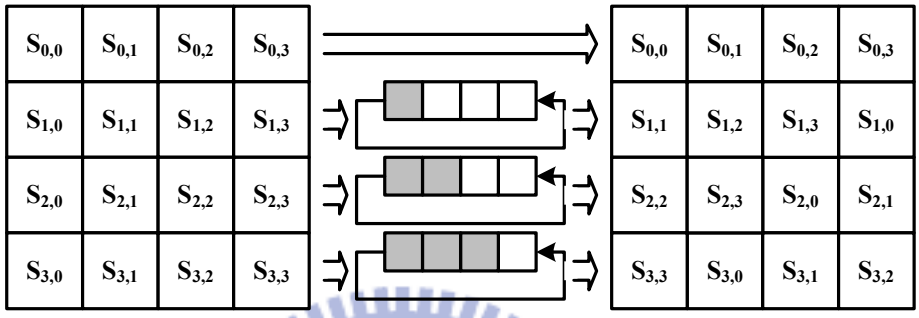


Figure 2.6: The ShiftRows transformation on the AES data block.

The data block of the AES algorithm can be divided into 16 bytes and substituted by 16 tables as shown in Fig. 2.5.

2.1.3 ShiftRows and Inv-ShiftRows

The ShiftRows transformation is a kind of permutation operation that cyclically shifts rows of the state with different number of bytes. The first row is not shifted while the other rows are cyclically shifted left by 1, 2, and 3 bytes, respectively. Fig. 2.6 shows how the state is shifted for the transformation.

The Inv-ShiftRows transformation used in decryption is the inverse operation of the ShiftRows. Instead of cyclically shifted left, the rows are now cyclically shifted right as shown in Fig. 2.7.

2.1.4 MixColumns and Inv-MixColumns

The MixColumns transformation treats columns of the data array as four-term polynomials $s(x)$ in field $GF((2^8)^4)$. These four-term polynomials are multiplied modulo $x^4 + 1$ by

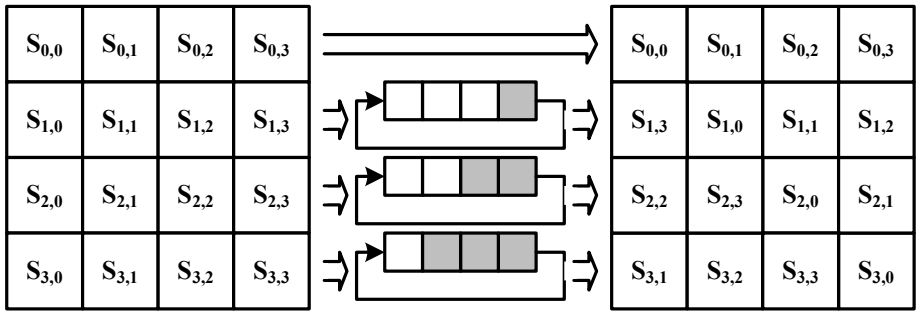


Figure 2.7: The Inv-ShiftRows transformation on the AES data block.

a fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ to obtain a new four-term polynomial $s'(x)$. Coefficients of $s'(x)$ can be obtained by matrix operation:

$$\begin{bmatrix} s'_{0,i} \\ s'_{1,i} \\ s'_{2,i} \\ s'_{3,i} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,i} \\ s_{1,i} \\ s_{2,i} \\ s_{3,i} \end{bmatrix} \tag{2.15}$$

where i is the column index.

The Inv-MixColumns is the inverse transformation of the MixColumns. Each column of the data array is expressed as a four-term polynomial $s(x)$. Then $s(x)$ is multiplied modulo $x^4 + 1$ by the inverse of $a(x)$, which is given by $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$. The resulting coefficients of $s'(x) = a^{-1}(x)s(x)$ can be obtained by matrix operation similar to MixColumns:

$$\begin{bmatrix} s'_{0,i} \\ s'_{1,i} \\ s'_{2,i} \\ s'_{3,i} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,i} \\ s_{1,i} \\ s_{2,i} \\ s_{3,i} \end{bmatrix} \tag{2.16}$$

The MixColumns and Inv-MixColumns transformation are applied to the data block as shown in Fig. 2.8. Note that 4 independent MixColumns and Inv-MixColumns are used on one 128-bit data block.

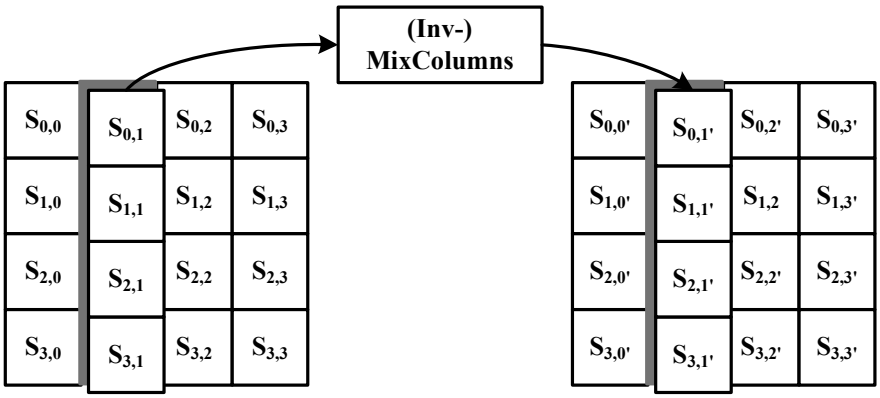


Figure 2.8: The Inv-/MixColumns transformation on the AES data block.

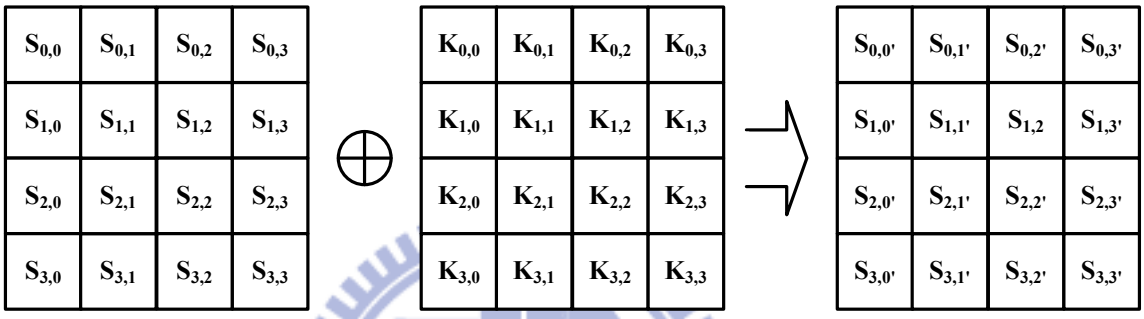


Figure 2.9: The AddRoundKeys transformation on the AES data block.

2.1.5 AddRoundKeys

The AddRoundKeys transformation adds round keys into the data block by simple XOR operation. The 128-bit round keys are derived from the secret key by the key expansion algorithm illustrated later. The round key is arranged in the same order as the data block as shown in Fig. 2.1. Then the AddRoundKeys transformation is accomplished by bit-wise XOR of the data array and round key array as shown in Fig. 2.9. Since the inverse of XOR operation is equivalent to itself, the inverse AddRoundKeys transformation is the same as the AddRoundKeys; therefore, no Inv-AddRoundKeys transformation is specified.

2.1.6 Key Expansion

The key expansion algorithm uses the secret key to generate total Nb(Nr+1) words for AddRoundKeys transformation, where Nb = 4 for 128-bit data block. The pseudo code of key expansion algorithm is shown as follows:

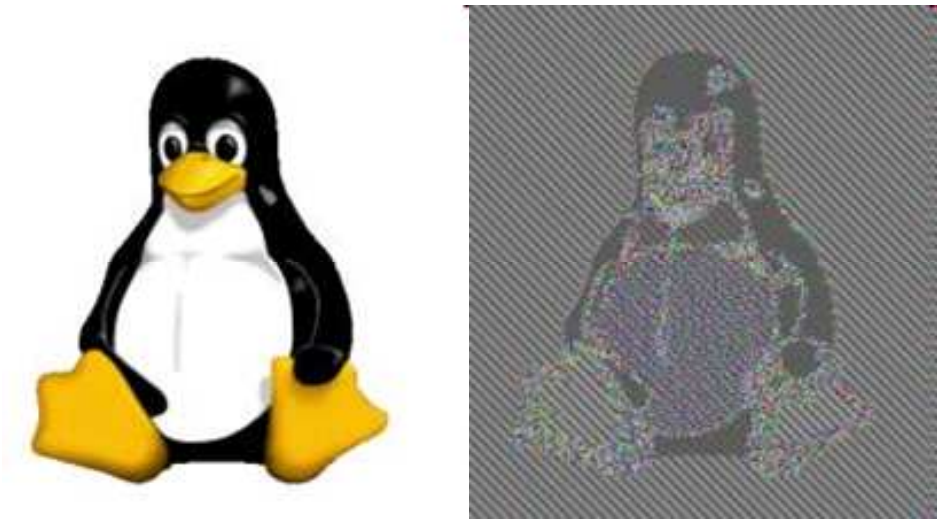
```
Input: byte key[4Nk], Nk
Output: word w[Nb(Nr+1)]
word temp;
i=0;
while i < Nk do
    w[i] = word(key[4i], key[4i+1], key[4i+2], key[4i+3]);
    i = i + 1;
end
while i < Nb(Nr+1) do
    temp = w[i-1];
    if i % Nk == 0 then
        temp = SubWord(RotWord(temp)) XOR Rcon[i/Nk];
    end
    else if (Nk > 6) && (i % Nk == 4) then
        temp = SubWord(temp);
    end
    w[i] = w[i-Nk] XOR temp;
    i = i + 1;
end
```

Algorithm 1: Key Scheduling

Note that Nk = 4, 6, and 8 for key length 128, 192, and 256 bits, respectively. The SubWord is a function that substitutes an input word into an output word by four independent SubBytes transformations. The RotWord performs the cyclically shift left by one byte to input words. The Rcon[i] is a constant word array containing $[x^{i-1}, \{00\}, \{00\}, \{00\}]$. The polynomial x can be represented in hexadecimal format as $\{02\}$.

At the beginning of the key expansion algorithm, the first Nk words are filled with the input key. Then the subsequent words can be obtained by XOR the previous word w[i-1] and Nk previous word w[i-Nk]. If the index i is a multiple of Nk, then the previous word w[i-1] is first applied to the RotWord, SubWord and XORed with Rcon[i/Nk] before the XOR operation with w[i-Nk].

If the key length is 256, or Nk = 8, the key expansion process is slightly different from the other key lengths. If Nk = 8 and i-4 is a multiple of Nk, then the previous word w[i-1] is applied to the SubWord first before the XOR operation with w[i-Nk].



Original **Encrypted**

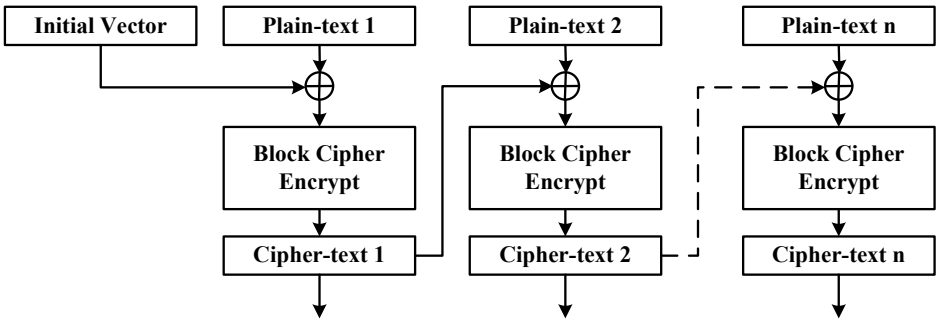
Figure 2.10: The encryption of the ECB mode.

2.2 Modes of operations for Block Ciphers

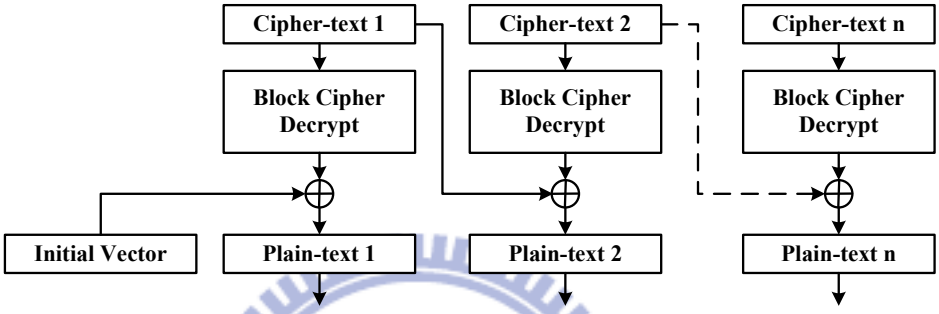
Block ciphers such as the AES algorithm process on data blocks with fixed length at a time using a single key. As a result, the message should be first partitioned into several data blocks for processing. Except directly encrypting these data blocks, the NIST approved several different modes of operation for confidentiality [45, 46], authentication [47], or both [48, 49].

The most intuitive mode of operation is the electronic codebook (ECB) mode. Data blocks are processed separately by block ciphers with the same secret key. This mode is similar to the assignment of code words in the codebook. Plain-texts and cipher-texts have one-to-one mapping in the codebook with the same key under this mode. However, this property is undesirable under some applications. For example, when the picture shown in Fig. 2.10 [1] is encrypted by the ECB mode, the resulting encrypted data can not perfectly hide the information contained in the picture. Therefore, other modes of operation must be used in such applications. In addition to the data confidentiality, the NIST also approves modes for the authentication. All these modes will be briefly introduced in the following subsections.

[1] Picture source: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation



(a)



(b)

Figure 2.11: The data flow of the CBC mode. (a) Encryption. (b) Decryption.

2.2.1 Cipher Block Chaining (CBC) Mode

In the cipher block chaining mode, plain-text blocks are combined with previous cipher-text blocks to form input blocks of the block cipher. For the first plain-text block, an initial vector (IV) is required to form the first input data block. The IV does not have to be secret, but it must be unpredictable as discussed in [45].

To encrypt data in the CBC mode, the first plain-text block is combined with the IV by XOR operation. The combined data block is then encrypted by the block cipher with the secret key to obtain the first cipher-text block. Then the cipher-text block is combined into the next plain-text block by XOR operation to generate the second input block. The second input block is encrypted by the block cipher with the same key to obtain the second cipher-text block. Then following cipher-text blocks can be obtained in the same manner as shown in Fig. 2.11(a).

Fig. 2.11(b) shows the decryption flow in the CBC mode. The first cipher-text block is

decrypted with the secret key. Then the first plain-text block is recovered by combining the cipher output and the IV. For the subsequent blocks, the cipher-text blocks are decrypted and combined with previous cipher-text blocks to obtain plain-text blocks.

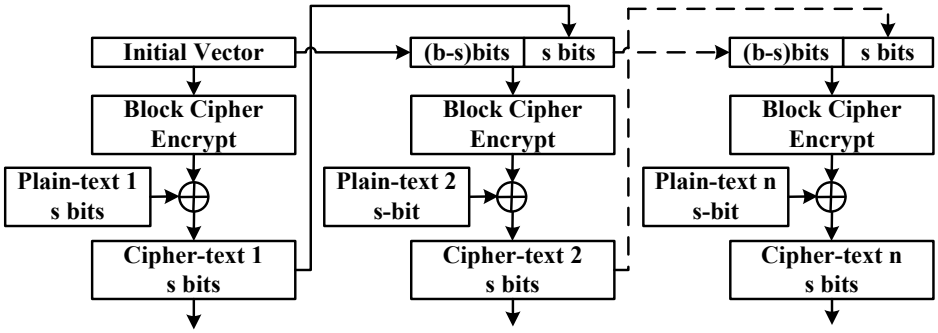
For the encryption in CBC mode, cipher-text blocks depend on all the preceding cipher-text blocks; therefore, the encryption process cannot be done in parallel. On the other hand, since the decryption in CBC mode depends only on the current and previous cipher-text blocks, the decryption can be performed in parallel.

2.2.2 Cipher Feedback (CFB) Mode

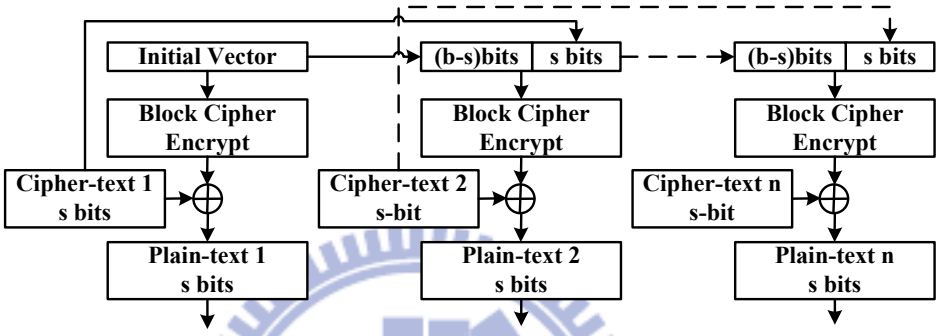
Block ciphers in cipher feedback mode work analogously to stream ciphers, the IV and the secret key are used to generate a series of data blocks and these data blocks are XORed with plain-text blocks to produce cipher-text blocks. Note that in CFB mode a parameter s is defined for the length of one plain-text or cipher-text segment. For example, if s is defined as 64, then each plain-text and cipher-text is of length 64 bits and the name of the mode is called 64-bit CFB mode.

The encryption flow of the CFB mode is illustrated in Fig. 2.12(a). Instead of the first plain-text block, the IV is encrypted by the block cipher to generate an output data block. The most significant s bits are then used to XOR with the first plain-text block to produce the first s-bit cipher-text block. The least significant (b-s) bits of the cipher input is concatenated with the cipher-text block to generate subsequent input blocks.

Since block ciphers in the CFB mode are analogous to stream ciphers, the decryption flow is identical to the encryption flow to generate the same "*key stream*". The IV is encrypted by the block cipher to generate the first output data block. The first cipher-text block is XORed with the most significant s bits to produce the first plain-text block. For following blocks, the least significant (b-s) bits of the previous input block are concatenated with the previous cipher-text block to generate the input block. The most significant s bits are XORed with the cipher-text block to recover plain-text blocks. The decryption in CFB mode is illustrated in Fig. 2.12(b).



(a)



(b)

Figure 2.12: The data flow of the CFB mode. (a) Encryption. (b) Decryption.

Note that in the CFB mode, only the encryption operation of the block cipher is required to accomplish the encryption and decryption flow. Furthermore, since input blocks to the block cipher is dependent on previous cipher-text blocks, the encryption flow can not be performed in parallel. However, the decryption can be done in parallel because the input block only depends on the previous input block and the previous cipher-text block.

2.2.3 Output Feedback (OFB) Mode

Block ciphers in output feedback mode work also analogously to stream ciphers, the IV and the secret key are used to generate a key stream for encryption and decryption. The output data block from the block cipher is used as the input blocks for subsequent operations. To produce cipher-text blocks, output blocks from the cipher is XORed with plain-text blocks, and vice versa.

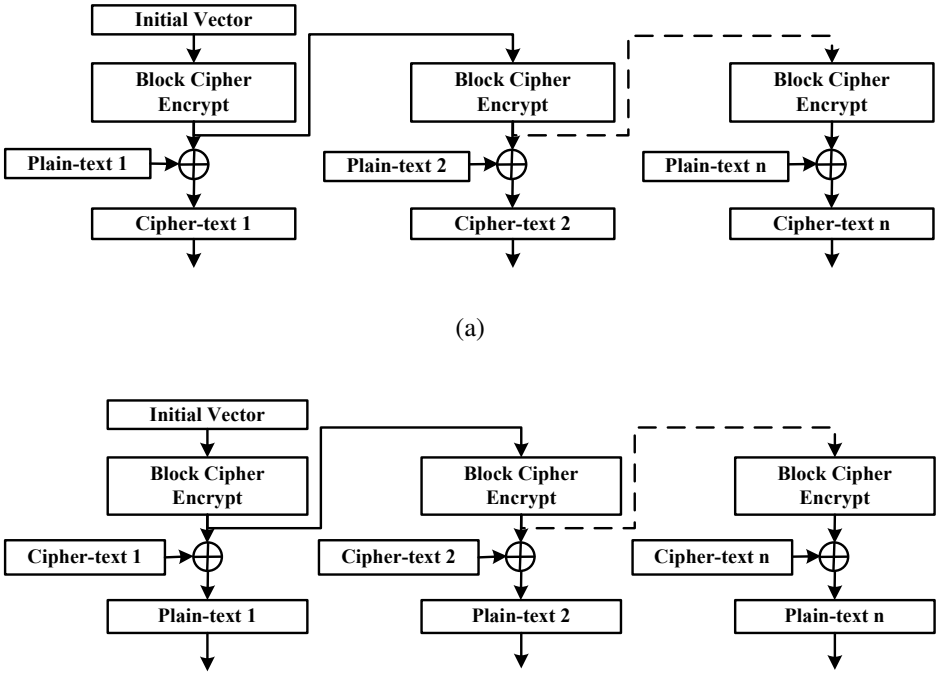


Figure 2.13: The data flow of the OFB mode. (a) Encryption. (b) Decryption.

The encryption flow is shown in Fig. 2.13(a). To encrypt messages in the OFB mode, the IV is encrypted by the block cipher to generate an output data block. The output data block is then XORed with the first plain-text block to produce the first cipher-text block. The output data block is feedback to the block cipher to generate the next output data block for next plain-text block. For the last block, if the plain-text block contains only u bits, which is less than the block size, then only the most significant u bits are XORed with the plain-text block and the remaining bits are discarded.

The decryption flow in the OFB mode is identical to the encryption flow as shown in Fig. 2.13(b). The same IV is encrypted to generate the first output data block. The first plain-text block can be recovered by XORing the output block of the cipher and the cipher-text block. The output data block is used to generate the next output data block for following cipher-text blocks. Note that the decryption in OFB uses only the encryption function of the block cipher.

Since input blocks of the block cipher are dependent on all the previous output blocks,

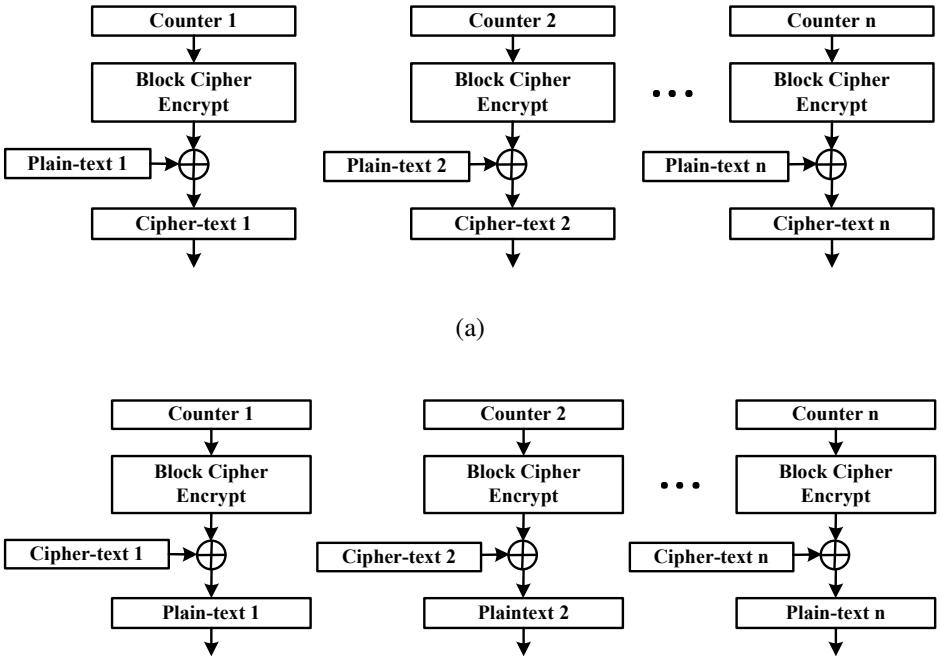


Figure 2.14: The data flow of the CTR mode. (a) Encryption. (b) Decryption.

the encryption and decryption in OFB mode can not be performed in parallel.

2.2.4 Counter (CTR) Mode

In the counter mode, successive blocks, called counters, are applied to the block cipher to generate a sequence of output blocks that are XORed with plain-text blocks to produce cipher-text blocks, and vice versa. Successive input blocks must be different form each other under the same secret key.

The encryption flow of the CTR mode is shown in Fig. 2.14(a). Each counter block is applied to the block cipher to generate a sequence of output blocks. These output blocks are then XORed with plain-text blocks to produce cipher-text blocks. For the last plain-text block, if the length is u bits, which is less that the block size, then the most significant u bits of the last output block is XORed with the plain-text block while the remaining bits are discarded.

The decryption flow of the CTR mode is exactly the same as the encryption flow as shown

in Fig. 2.14(b). Cipher-text blocks are XORed with output blocks from the block cipher to recover plain-text. For the last block, only the most significant u bits are used if the last cipher-text block is of length u-bit.

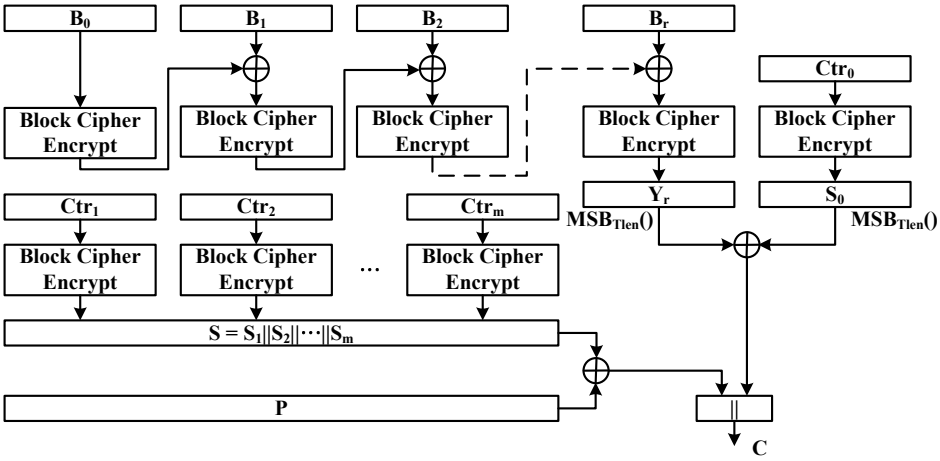
In either encryption or decryption of the CTR mode, each output block only depends on the specific counter block. As a result, both the encryption and decryption can be performed in parallel for high throughput applications.

2.2.5 Counter with CBC-MAC (CCM) Mode

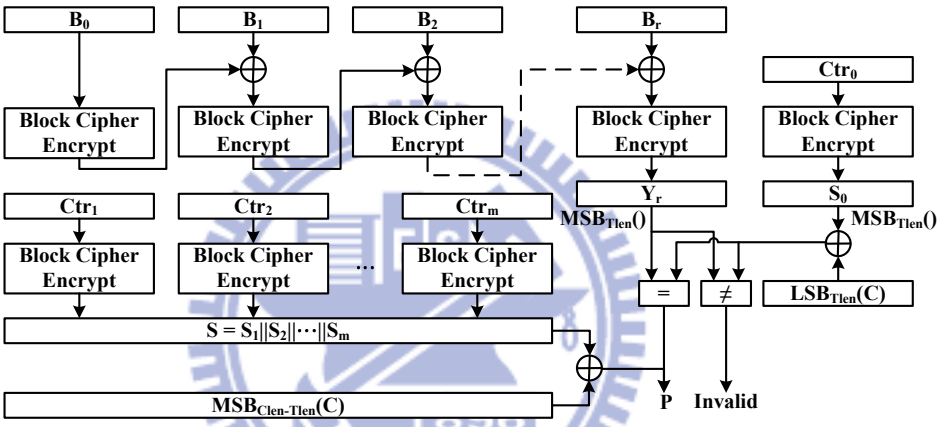
The CCM mode is used to provide confidentiality and authenticity of data by combining techniques of CTR mode and CBC mode. The data that CCM protects consists of a message P with bit length Plen and a associated data A. The confidentiality is provided for the message P and the authenticity is provided for both message P and associated data A. In addition, a nonce N is assigned to each data pair, P and A, to be protected.

The generation-encryption process is shown in Fig. 2.15(a). The input data to the generation-encryption process are a valid nonce N, a message P with Plen bits, and an associated data A. For the authenticity, (N, P, A) are used to generate a series of blocks B_i for CBC operation. The generation of B_i is specified in NIST SP800-38C [48]. B_0 is encrypted and then the output block is XORed with B_1 for cipher chaining. After all B_is are incorporated, the most significant Tlen bits are saved as an internal variable. In addition, the nonce N is used to generate a series of counter blocks Ctr_i. The first counter block Ctr_i is encrypted and XORed with the former internal variable to produce a tag. For the confidentiality, the following counter blocks are encrypted and concatenated as S with length equal to Plen, the size of the message P. Then S and P are bit-wise XORed to produce the encrypted message. At last, the encrypted message is concatenated with the tag to generate the output C.

The decryption-verification process is quite similar to the generation-encryption flow as shown in Fig. 2.15(b). Before starting the decryption-verification process, the length of the encrypted message Clen is checked. If the Clen is less or equal to Tlen, which means received message C is invalid, the INVALID message is returned without further processing.



(a)



(b)

Figure 2.15: The data flow of the CCM mode. (a) Encryption. (b) Decryption.

On the other hand, if the nonce N is valid, N is used to generate a series of counter blocks Ctr_i. These blocks are encrypted by the block cipher to produce output block S_i. The plain-text P can be recovered by XORing the concatenation of these blocks with the most significant Clen-Tlen bits of the received message. For the authenticity, the (N, A, P) is used to produce a series of blocks B_i. These blocks are used to generate an internal variable as that performed in the generation-encryption flow. Then the internal variable is compared with $LSB_{Tlen}(C) \oplus MSB_{Tlen}(S_0)$. If these two variables are equivalent, then the decrypted plain-text P is returned. Otherwise, the INVALID message is returned and the decrypted P and tag should not be revealed.

2.3 Applications of the AES algorithm

The AES algorithm can be used in numerous applications that require data protection, especially in those that have high performance requirements. In this section, applications that use the AES as the data encryption algorithm are briefly introduced.

2.3.1 Wireless Communications

The data security has become more and more important with the development and popularization of wireless communication techniques such as Wireless LAN (WLAN, IEEE 802.11), Person Area Networks (PANs, such as WPAN, ZigBee, and UltraWideband), and Wireless Metropolitan Area Networks (WiMAX). Since data is transmitted over a public environment, any unauthorized party is able to eavesdrop the transmitted information. Therefore, data encryption scheme must be defined in such applications to provide data security.

For WLAN, or IEEE 802.11, security requirements are entity authentication, authorization, data confidentiality and data integrity. The first security solution for WLAN is wired equivalent privacy (WEP). Several security flaws were discovered in WEP and then Wifi-protected access (WPA) is adopted as the security solution. Finally, the IEEE 802.11i standard [50], also known as WPA2, is finalized in 2004 to resolve the security weakness of WEP. IEEE 802.11i specifies two different security architectures, the temporal key integrity protocol (TKIP) and the counter CCM mode protocol (CCMP). The CCMP uses the AES algorithm in CTR mode to provide data confidentiality and in CBC-MAC mode to provide data authentication. The CCMP can provide higher security than RC-4 based TKIP. More details about the security architecture for WLAN can be found in IEEE 802.11i standard.

In addition to the WLAN security, several other applications adopt the AES in CCM mode as the fundamental encryption algorithm. Wireless Person Area Network (WPAN, or IEEE 802.15.3) requires the AES in CTR and CBC mode for data encryption and message authentication. ZigBee (IEEE 802.15.4) is a low power wireless standard which uses the AES CCM mode for encryption and authentication. Other wireless applications such as

WiMAX (IEEE 802.16e) and UltraWideband also adopt AES in CCM mode for data encryption and authentication.

2.3.2 Network Protocols

In addition to wireless communications, security is also provided in several network communications such as the Ethernet, the Fibre Channel, or the Ethernet Passive Optical Network (EPON). IEEE 802.1ae [51], also known as MACsec, specifies provision of connectionless user data confidentiality, frame data integrity, and data origin authenticity by media access independent protocols and entities that operate transparently to MAC clients. IEEE 802.1ae adopts the AES in Galois/Counter mode (GCM) [49] with key length 128 and 256 bits for data encryption. The GCM mode differs from the CCM mode in the authentication tag generation. The GCM mode defines a new scheme to produce the authentication tag instead of using CBC-MAC.

For Fibre Channel applications such storage area networks (SAN), the security is also specified in Fibre Channel - Security Protocol (FC-SP) [52]. FC-SP specifies protocols to enhance the authentication of Fibre Channel devices, secure key exchange, and secure communication between Fibre Channel devices. The AES CBC mode and GCM mode are adopted in this standard for data encryption.

The Transport Layer Security (TLS) and Secure Socket Layer (SSL) are cryptographic protocols to provide secure communication over the Internet. The AES algorithm is also defined in TLS and SSL for data confidentiality.

2.3.3 Tamper Resistant Applications

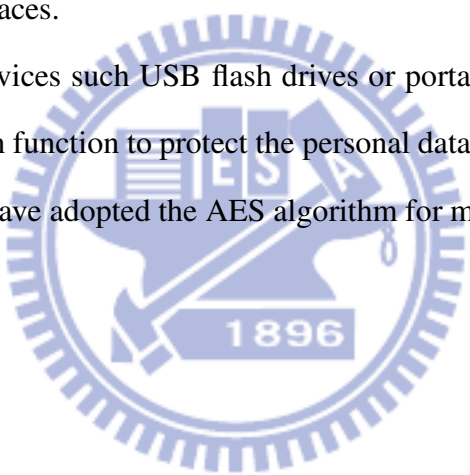
For several applications, the content stored in the chip must be keep security and cannot be accessed by unauthorized users. For example, a tamper resistant smart card can prevent the attacker to retrieve or modify the private information such as the private key or the electronic credit stored in the chip. Other applications such as digital rights management (DRM) or

storage devices also need the private data kept secret.

A smart card is a plastic card with an embedded chip, which contains memory or microprocessor, to store and transact data. The cryptographic algorithm can be implemented in software or hardware to provide secure transaction for banking systems or E-commercial applications. The most common block cipher for the smart card today is the DES or triple-DES; however, for stronger security, the AES is also provided in modern smart cards.

DRM is used by copyright holders, publishers to limit the usage of digital contents such as films, televisions, musics, computer games, or E-books. Digital contents are encrypted by ciphers such as the AES and then published. Only those who are authorized can access to the digital contents in limited ways. For example, the contents can be accessed only in limited periods or at limited places.

Modern storage devices such USB flash drives or portable external hard drives usually include data encryption function to protect the personal data stored in these devices. Several commercial products have adopted the AES algorithm for mass data encryption.



Chapter 3

Design of AES Crypto Engines

As mentioned in the last chapter, the AES algorithm can be adopted in several different applications for data encryption. Therefore, implementations of the AES algorithm have been well discussed in several literatures [20–39]. In this chapter, previous works on the AES algorithm are briefly reviewed and then three different architectures are proposed for different targeted applications. At first, an architecture for ultra high throughput applications such as storage devices or 10G Ethernet is presented. The only design consideration is the working frequency and the throughput. After that, an architecture optimized for ultra low cost applications such as smart cards or RFIDs is presented in the following section. The throughput in such applications is not a big issue, thus the hardware cost is optimized to be minimum. At last, an efficient architecture is presented for applications that require both high throughput and low cost. A brief summary of different architectures is given at the end of this chapter to provide some inspections about the AES crypto engines.

3.1 Previous Works on the AES algorithm

The AES algorithm is composed of four different transformations and implementations of these transformations are critical to the AES crypto engine. Therefore, several literatures focus on the optimization of these transformations [20, 53–56]. The ShiftRows transformation re-orders data bytes of the data block and this transformation can be done by wire

connections. The AddRoundKeys transformation adds round keys into data blocks by simple XOR operation. Therefore, optimizations of these two transformations are less addressed. The SubBytes transformation is the only non-linear operation and is the most complicated component in the AES algorithm. Several different implementations of the SubBytes have been proposed [20,27,53–55]. The MixColumns transformation can be performed by matrix operation and it can be reused in the Inv-MixColumns transformation by factoring [20]. Implementations of SubBytes and MixColumns are briefly introduced in the following sections.

3.1.1 SubBytes Transformation

The SubBytes can be implemented in two types: combinational look up table (LUT) based and composite field based. The LUT based method is usually adopted in high throughput architectures because the SubBytes transformation is implemented in logic level and the critical path can be optimized by truth table or Karnaugh-Map. The composite filed method uses arithmetics in finite field to optimize the SubBytes in terms of hardware complexity. A finite field arithmetic unit is used for SubBytes to reduce required hardware resources. Therefore, the composite field based method is usually adopted for low cost architectures. In this subsection, previous works on the composite field based SubBytes are illustrated.

As mentioned earlier, the SubBytes can be decomposed into multiplicative inversion followed by affine transformation. The multiplicative inversion in $GF(2^8)$ can be computed by extended Euclidean algorithm. However, the calculation of inverse in $GF(2^8)$ is quite complicated while the calculation of inverse in $GF(2^2)$ is relative easy, as suggested by Rijmen [57]. Therefore, field elements can be transformed to composite field $GF((2^4)^2)$ [22,31–33,38,53] or $GF(((2^2)^2)^2)$ [20,54] to reduce the hardware complexity.

An element G in $GF(2^8)$ can be represented over $GF(2^4)$ as $G = \gamma_1 y + \gamma_0$ with irreducible polynomial $r(y) = y^2 + \tau y + \nu$. Note that coefficients γ_1 and γ_0 are both 4-bit elements in subfield $GF(2^4)$. In this way, the pair $[\gamma_1, \gamma_0]$ can be used to present the element G in field $GF((2^4)^2)$. The element can be represented using the polynomial basis $[Y, 1]$ or using the normal basis $[Y^{16}, Y]$, where Y^{16} and Y are roots of $r(y)$. Note that in normal

basis, Y^{16} and Y are both roots of $r(y)$. As a result,

$$r(y) = y^2 + \tau y + \nu = (y + Y)(y + Y^{16}), \tag{3.1}$$

which means $\tau = (Y + Y^{16})$ is the trace and $\nu = (Y)(Y^{16})$ is the norm of Y.

Similarly, coefficients in $GF(2^4)$ can be represented over $GF(2^2)$ as $\Gamma_1 z + \Gamma_0$ with irreducible polynomial $r(z) = z^2 + Tz + N$, where Γ_1 and Γ_0 are both in subfield $GF(2^2)$. Again, the element can be represented using the polynomial basis $[Z, 1]$ or using the normal basis $[Z^4, Z]$. Note that $T = Z + Z^4$ is the trace and $N = (Z)(Z^4)$ is the norm of Z if they are represented in normal basis.

At last, the element in $GF(2^2)$ can be also represented over $GF(2)$ as $g_1 w + g_0$ with irreducible polynomial $r(w) = w^2 + w + 1$, where g_1 and g_0 are both in $GF(2)$, or single bits. The polynomial basis $[W, 1]$ and the normal basis $[W^2, W]$ can be used to represent the pair $[g_1, g_0]$. The above decomposition can simplify the operation in $GF(2^8)$ to $GF(2^4)$, which in turn can be simplified further over $GF(2^2)$ and $GF(2)$.

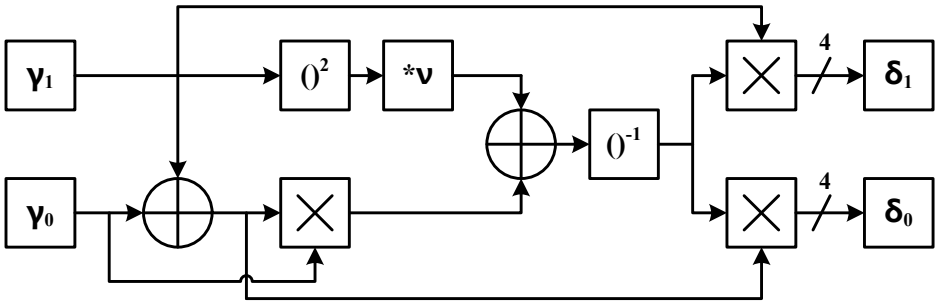
Polynomial Basis

The multiplication in $GF(2^8)$ can be mapped to $GF((2^4)^2)$ modulo $r(y)$ as

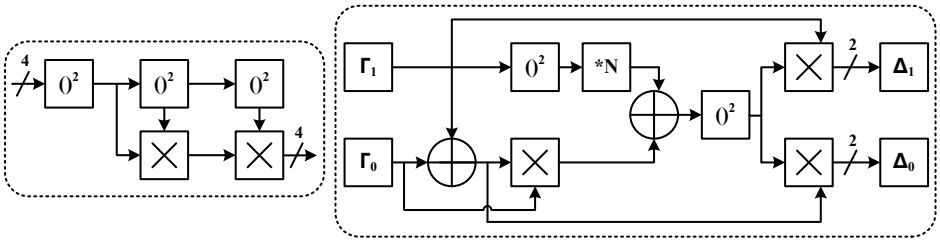
$$(\gamma_1 y + \gamma_0)(\delta_1 y + \delta_0) = (\gamma_1\delta_0 + \gamma_0\delta_1 + \gamma_1\delta_1\tau)y + (\gamma_0\delta_0 + \gamma_1\delta_1\nu). \tag{3.2}$$

Thus, the multiplicative inverse can be computed by making the right hand side of equation 3.2 equal to 1. In this way, $\delta_1 y + \delta_0$ is the multiplicative inverse of $\gamma_1 y + \gamma_0$. The multiplicative inverse can be found by solving following equations:

$$\begin{aligned} \gamma_1\delta_0 + \gamma_0\delta_1 + \gamma_1\delta_1\tau &= 0 \\ \gamma_0\delta_0 + \gamma_1\delta_1\nu &= 1. \end{aligned} \tag{3.3}$$



(a)



(b)

Figure 3.1: Polynomial basis inverter (a) Over $GF(2^8)$. (b) Over $GF(2^4)$.

The multiplicative inverse is then given by

$$(\gamma_1 y + \gamma_0)^{-1} = (\delta_1 y + \delta_0) = [\theta^{-1}\gamma_1]y + [\theta^{-1}(\gamma_0 + \gamma_1 \tau)] \tag{3.4}$$

where $\theta = \gamma_1^2 \nu + \gamma_1 \gamma_0 \tau + \gamma_0^2$.

The multiplicative inverse in $GF(2^8)$ is then decomposed into a series of multiplications, additions, and a multiplicative inverse in $GF(2^4)$. The equation 3.2 and 3.4 then can be modified to decompose the operation in $GF(2^4)$ into $GF(2^2)$. Once the operation is reduced over $GF(2^2)$, the inverse operation is identical to the square operation because for $\Gamma \in GF(2^2)$, $\Gamma^4 = \Gamma$, and therefore $\Gamma^2 = \Gamma^{-1}$.

Fig. 3.1(a) shows the multiplicative inverter over the polynomial basis in $GF((2^4)^2)$. The inverter in field $GF(2^4)$ can be implemented by a series of multiplication such that $x^{-1} = x^{14}$, by combinational LUT with 16 entries, or by further decomposition of the inverter over $GF(2^2)$. Fig. 3.1(b) shows different implementations of the inverter in $GF(2^4)$.

Normal Basis

The operation in normal basis $[Y^{16}, Y]$, where Y^{16} and Y are roots of $r(y) = y^2 + \tau y + \nu$, uses properties $\tau = Y^{16} + Y$, $\nu = (Y^{16})(Y)$, and $1 = \tau^{-1}(Y^{16} + Y)$. The multiplication in $GF(2^8)$ then can be decomposed into $GF((2^4)^2)$ modulo $r(y)$ as:

$$\begin{aligned}(\gamma_1 Y^{16} + \gamma_0 Y)(\delta_1 Y^{16} + \delta_0 Y) &= \gamma_1\delta_1 Y^{32} + (\gamma_1\delta_0 + \gamma_0\delta_1)Y^{17} + \gamma_0\delta_0 Y^2 \\ &= \gamma_1\delta_1(\tau Y^{16} + \nu) + (\gamma_1\delta_0 + \gamma_0\delta_1)\nu + \gamma_0\delta_0(\tau Y + \nu) \\ &= \gamma_1\delta_1\tau Y^{16} + \gamma_0\delta_0\tau Y + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}(Y^{16} + Y) \\ &= (\gamma_1\delta_1\tau + \theta)Y^{16} + (\gamma_0\delta_0\tau + \theta)Y\end{aligned} \tag{3.5}$$

where $\theta = (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}$. Again, the multiplicative inverse can be calculated by making $(\gamma_1 Y^{16} + \gamma_0 Y)(\delta)_1 Y^{16} + \delta_0 Y) = 1 = \tau^{-1}Y^{16} + \tau^{-1}Y$. Then the inverse can be found by solving following equations:

$$\begin{aligned}[\gamma_1\delta_1\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}] &= \tau^{-1} \\ [\gamma_0\delta_0\tau + (\gamma_1 + \gamma_0)(\delta_1 + \delta_0)\nu\tau^{-1}] &= \tau^{-1}\end{aligned} \tag{3.6}$$

The multiplicative inverse in normal basis is then given by

$$(\gamma_1 Y^{16} + \gamma_0 Y)^{-1} = (\delta_1 Y^{16} + \delta_0 Y) = [\theta'^{-1}\gamma_0]Y^{16} + [\theta'^{-1}\gamma_1]Y, \tag{3.7}$$

where $\theta' = \gamma_1\gamma_0\tau^2 + (\gamma_1^2 + \gamma_0^2)\nu$.

The multiplicative inverse in $GF(2^8)$ can be decomposed into a series of multiplications, additions, and a inverse in $GF(2^4)$. Fig. 3.2 shows the inverter in $GF(2^8)$ with normal basis. The inverter consists of three multipliers, two adders, one inverter, and one squarer multiplied by constant ν. The multiplication in $GF(2^4)$ is analogous to that in $GF(2^8)$ as

$$(\Gamma_1 Z^4 + \Gamma_0 Z)(\Delta_1 Z^4 + \Delta_0 Z) = (\Gamma_1\Delta_1 T + \Theta)Z^4 + (\Gamma_0\Delta_0 T + \Theta)Z \tag{3.8}$$

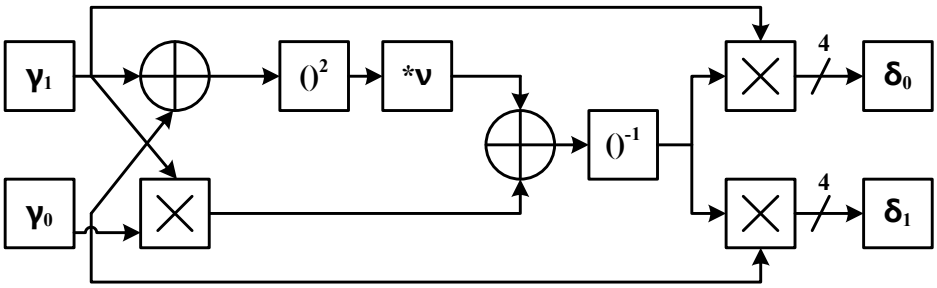


Figure 3.2: Normal basis inverter in $GF(2^8)$.

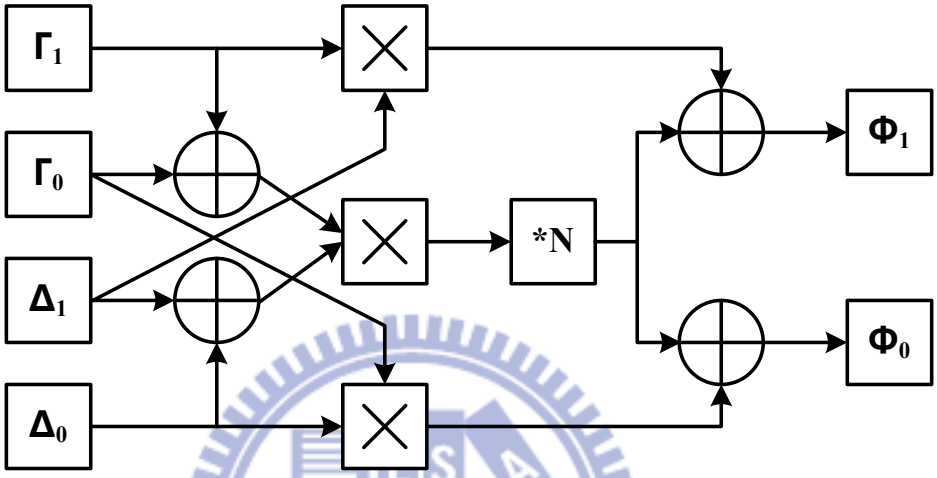


Figure 3.3: Normal basis multiplier in $GF(2^4)$.

where $\Theta = (\Gamma_1 + \Gamma_0)(\Delta_1 + \Delta_0)NT^{-1}$. The architecture of multiplier in $GF(2^4)$ is shown in Fig. 3.3. Note that multiplications and additions are performed over $GF(2^2)$. The multiplication in $GF(2^2)$ has the same structure, except that it lacks of scaling by norm, and in $GF(2)$ the multiplication is identical to AND operation.

In addition to multipliers and adders, another operation needed in $GF(2^4)$ is the combined operation of squaring followed by scaling ν as shown in Fig. 3.2. The combined operation can be represented as:

$$(\Gamma_1 Z^4 + \Gamma_0 Z)^2 \times \nu = [(\Gamma_1 + \Gamma_0)^2]Z^4 + [(N \times \Gamma_0)^2]Z. \tag{3.9}$$

The operation in $GF(2^4)$ now can be performed with addition, multiplication, and squaring in $GF(2^2)$. Note that in $GF(2^2)$ the inversion is the same as the squaring and can be

represented as:

$$\begin{aligned}(g_1W^2+g_0W)^{-1}=(g_1W^2+g_0W)^2 &= g_1^2W^4+g_0^2W^2\\ &= g_1^2(W^2+1)+g_0^2W2\\ &= g_1^2W^2+g_1^2(W^2+W)+g_0^2W^2\\ &= g_0^2W^2+g_1^2W\\ &= g_0W^2+g_1W,\end{aligned} \tag{3.10}$$

indicating that the squaring or inversion in $GF(2^2)$ is free by swapping the bit positions.

The remaining operation needed in $GF(2^4)$ multiplier is multiplication in $GF(2^2)$ and then scaling by $N=W^2$. The combined operation can be represented as:

$$\begin{aligned}(g_1W^2+g_0W)(d_1W^2+d_0W)\times N &= g_1d_1W^6+(g_1d_0+g_0d_1)W^5+g_0d_0W^4\\ &= g_1d_1+(g_1d_0+g_0d_1)W^2+g_0d_0W\\ &= g_1d_1(W^2+W)+(g_1d_0+g_0d_1)W^2+g_0d_0W\\ &= (g_1d_1+g_1d_0+g_0d_1)W^2+(g_1d_1+g_0d_0)W\\ &= [g_0d_0+(g_1+g_0)(d_1+d_0)]W^2+(g_1d_1+g_0d_0)W\end{aligned} \tag{3.11}$$

Inv-/SubBytes Sharing

For SubBytes and Inv-SubBytes transformations, the multiplicative inversion can be shared to reduce hardware resources. Fig. 3.4 shows the data flow of forward SubBytes transformation. The input byte is applied to a field transformation matrix δ first and then the element is applied to the multiplicative inversion over composite field $GF((2^4)^2)$ or $GF(((2^2)^2)^2)$. Then the field element is mapped back to $GF(2^8)$ by the inverse of field transformation matrix δ^{-1}. At last, the element is applied to the affine transformation to finish the SubBytes transformation. Note that the inverse filed transformation matrix and the affine transformation can be combined to reduce the number of matrix operation. Fig. 3.4 also shows the flow

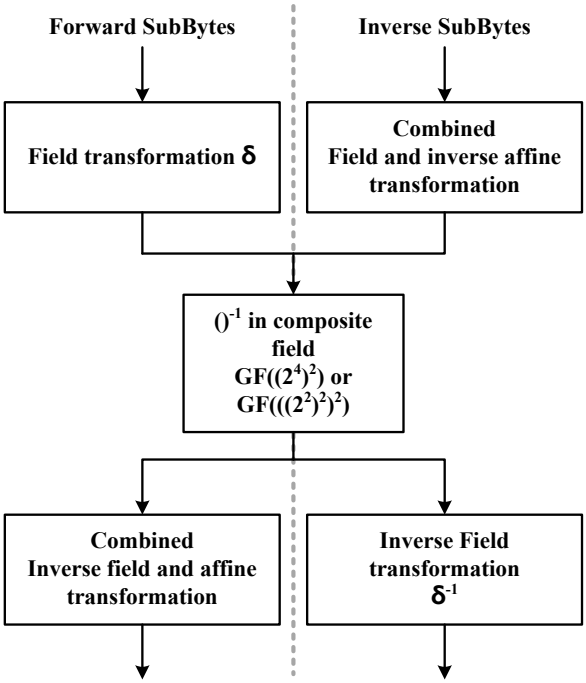


Figure 3.4: Inv-/SubBytes sharing structure.

of the Inv-SubBytes transformation. The input data byte is first applied to a inverse affine transformation and then the element is mapped in to composite field by δ. The final result can be obtained by applying the inverse field transformation to the multiplicative inverse. Note that the inverse affine transformation and the field transformation matrix can also be combined to reduce the number of matrix operation. With this structure, the multiplicative inversion can be shared in both encryption and decryption process.

The result from [54] shows that total 180 and 182 gates are required for the SubBytes and Inv-SubBytes, respectively. The total number of gates for both encryption and decryption would be 362 gates. By the hardware sharing of the multiplicative inverse unit, the combined SubBytes/Inv-SubBytes only requires 234 gates.

3.1.2 MixColumns Transformation

The MixColumns transformation can be written as matrix operation as:

$$\begin{aligned}\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} &= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \\ &= \begin{bmatrix} 00 & 01 & 01 & 01 \\ 01 & 00 & 01 & 01 \\ 01 & 01 & 00 & 01 \\ 01 & 01 & 01 & 00 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} 02 & 02 & 00 & 00 \\ 00 & 02 & 02 & 00 \\ 00 & 00 & 02 & 02 \\ 02 & 00 & 00 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}\end{aligned} \tag{3.12}$$

Common terms can be reduced by factoring to obtain equations for the MixColumns as follows:

$$\begin{cases} b_0 = 02X_0 + X_1 + a_3 \\ b_1 = 02X_1 + X_2 + a_0 \\ b_2 = 02X_2 + X_3 + a_1 \\ b_3 = 02X_3 + X_0 + a_2 \end{cases} \begin{cases} X_0 = a_0 + a_1 \\ X_1 = a_1 + a_2 \\ X_2 = a_2 + a_3 \\ X_3 = a_3 + a_0 \end{cases} \tag{3.13}$$

For the Inv-MixColumns transformation, the matrix operation can be decomposed into

three matrix operations as follows:

$$
\begin{aligned}
\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} &= \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \\
&= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \\
&+ \begin{bmatrix} 08 & 08 & 08 & 08 \\ 08 & 08 & 08 & 08 \\ 08 & 08 & 08 & 08 \\ 08 & 08 & 08 & 08 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} 04 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \\ 04 & 00 & 04 & 00 \\ 00 & 04 & 00 & 04 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}
\end{aligned}
\tag{3.14}
$$

As shown in equation 3.14, the Inv-MixColumns transformation consists of a forward MixColumns transformation and two more matrix operations. Therefore, the Inv-MixColumns and the MixColumns can be combined to reduce hardware resources. The result from [20] shows that the MixColumns requires 152 XOR gate and the Inv-MixColumns requires 440 XOR gates, which results in 592 XOR gates for both encryption and decryption. If the MixColumns and Inv-MixColumns are combined, only 195 XOR gates are required. However, the path delay is increased from 5 XOR gates to 7 XOR gates due to the decomposition of Inv-MixColumns transformation.

3.2 High Throughput AES Engine for 100 Gigabit Ethernet

Nowadays, several applications require throughput over 5 Gb/s such as USB 3.0 or SATA3 or even 10 Gb/s such as 10 Gigabit Ethernet (10 GbE). Moreover, the 40 GbE and 100 GbE were ratified in June 2010. The throughput requirement is further increased up to 100 Gb/s. In order to meet the requirement of these applications, parallel processing is unavoidable. Therefore, only AES in the ECB mode and the CTR mode can be used for high throughput architecture. This is because that in other modes, each data block is dependent on previous output blocks. As discussed in chapter 2, the AES in CTR mode can provide higher security than that in ECB mode. Moreover, only the encryption function of the AES is required when operating in the CTR mode. Therefore, the AES engine can be optimized in terms of the critical path delay to increase the throughput if only the encryption function is needed.

In this section, a high throughput AES architecture which consists of a pipelined data-path unit and an off-line key expansion unit is presented. The only design consideration for this architecture is the throughput because the hardware cost is usually a less important issue for high throughput applications. The data-path unit for the high throughput architecture will be presented first and then the off-line key expansion unit is presented latter.

3.2.1 Data-path Unit

The throughput for the AES engine can be calculated by equation:

$$Throughput = \frac{Frequency \times 128}{Latency}. \tag{3.15}$$

For the AES algorithm, the latency with key length 128 bits is 11, which includes 10 round functions and the initial round for adding key. To achieve 10 Gb/s, the operating frequency must be higher than 860 MHz. However, for the cell-based design, it is not easy to achieve such high operating frequency. Therefore, increasing the level of parallelism by loop un-

rolling is more feasible. The level of parallelism can be determined by the throughput requirement. For example, if 2 round functions are unrolled, the latency would be reduced to 6. In this way, the throughput 10 Gb/s can be achieved with operating frequency 470 MHz, which is more feasible for cell-based design. The throughput requirement of the 40 GbE can be achieved by using 4 independent AES engines or fully unrolling all the round functions. If round functions are fully unrolled, then the latency would be reduced to 1. The throughput 40 Gb/s then can be achieved with operating frequency 313 MHz. For 100 GbE, the operating frequency must be higher than 782 MHz to meet the throughput requirement even if the round functions are fully unrolled. Again, it is not easy for cell-based design to operate at 782 MHz, so a better solution for 100 Gb/s is using two 50 Gb/s AES engines operating at 391 MHz.

As mentioned above, the operating must be higher than 470 MHz and 391 MHz to achieve throughput 10 Gb/s and 50 Gb/s, respectively. That is, the critical path should be less than 2.12 ns and 2.55 ns. Since the SubBytes transformation is the most complicated, implementation of the SubBytes transformation dominates the overall critical path delay. Although the composite field based SubBytes can largely reduce the hardware complexity, the critical path is also increased significantly. Therefore, LUT based is more suitable for high throughput architectures. Fig. 3.5 shows the comparison between the composite field based and LUT based SubBytes in 90 nm CMOS technology. The minimum path delay of the LUT based method can be optimized to only 0.5 ns while that of composite field based method can only be optimized to 1.5 ns. The critical path delay of LUT based method is three times better with less than three times larger hardware cost.

Traditionally, usually 10% design margin is left for the back-end design flow. However, the wire delay sometimes dominates over the gate delay in deep sub-micron technology such as 90 nm CMOS. Therefore, the design margin is better increased to 30% - 50%, especially for operating frequency higher than 400 MHz. In this way, the critical path of the AES engine should be less than 1.5 ns to meet the throughput requirement when considering the design margin. As a result, the composite field based method cannot be used in ultra high

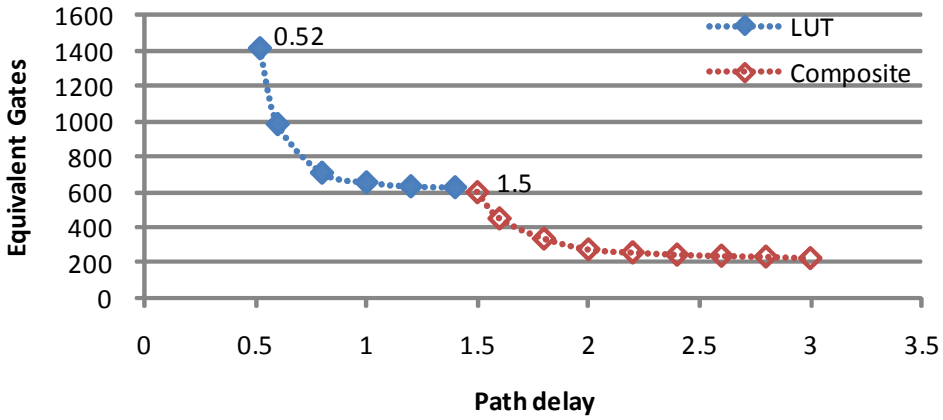


Figure 3.5: Implementation results of different SubBytes architecture.

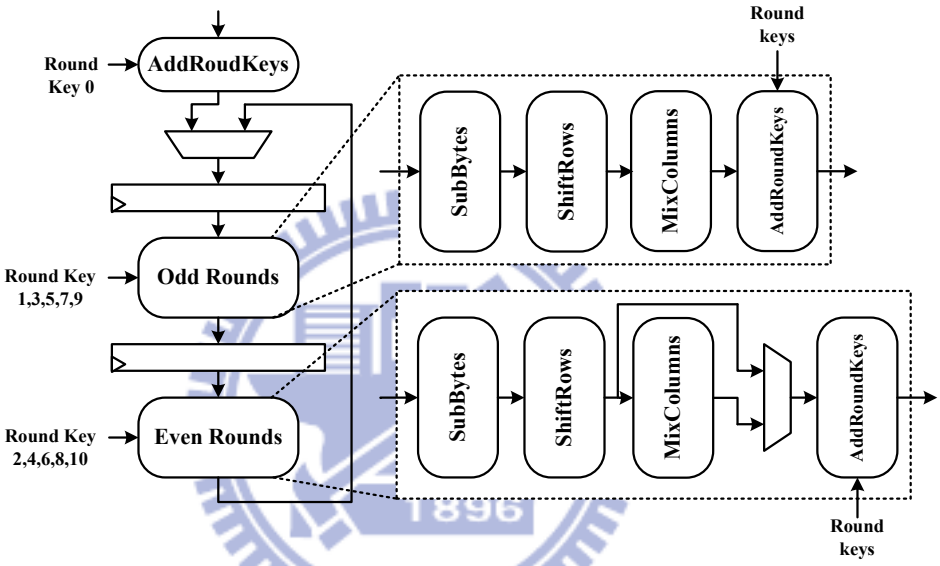


Figure 3.6: Data path for 10 Gb/s throughput.

throughput architecture without pipelining architecture.

Once the operating frequency can be higher than 470 MHz, the throughput requirements of 10 Gb/s and 50 Gb/s can be achieved by different levels of parallelism. Fig. 3.6 shows the data path with 2 rounds unrolled for 10 Gb/s throughput requirement. The initial round is performed during the data loading phase. The odd round functions contain four transformations and the even round functions contain one additional multiplexer for the last round to skip the MixColumns. With this architecture, the average output per clock cycle can be increased to 21.33 bits instead of 11.63 bits for non-unrolled architecture.

For throughput requirement higher than 50 Gb/s, the data path must be fully unrolled as mentioned above. Fig. 3.7 shows the fully unrolled and pipelined architecture. The ini-

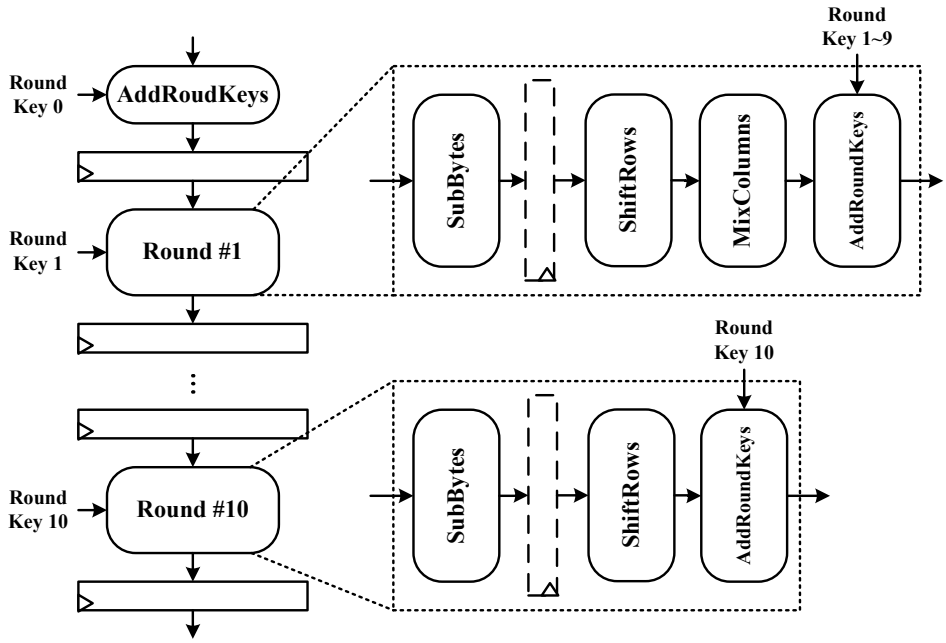


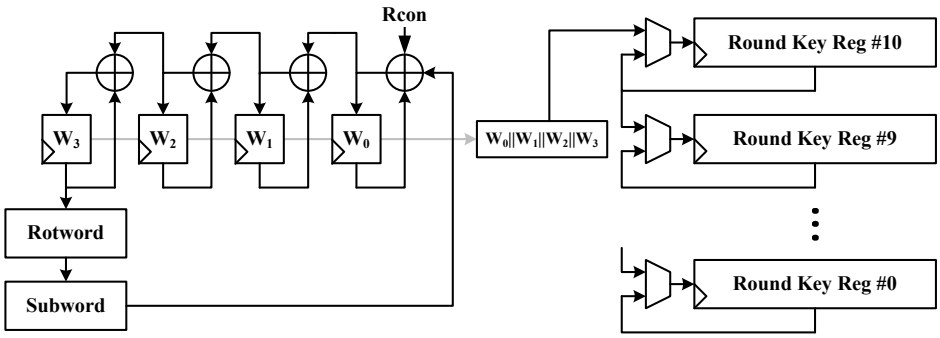
Figure 3.7: Data path for 50 Gb/s throughput.

tial round is still performed during the data loading phase and other round functions are performed with dedicated pipeline stage. Note that the MixColumns transformation is eliminated in the last round function. The throughput can be further increased by sub-stage pipelining, that is, each stage is divided into two phases. To balance the path delay of each sub-stage, the pipeline register is inserted right after the SubBytes transformation. The effect of the sub-stage pipelining method is discussed latter in this section. With fully unrolling scheme, the output per clock cycle can reach 128 bits.

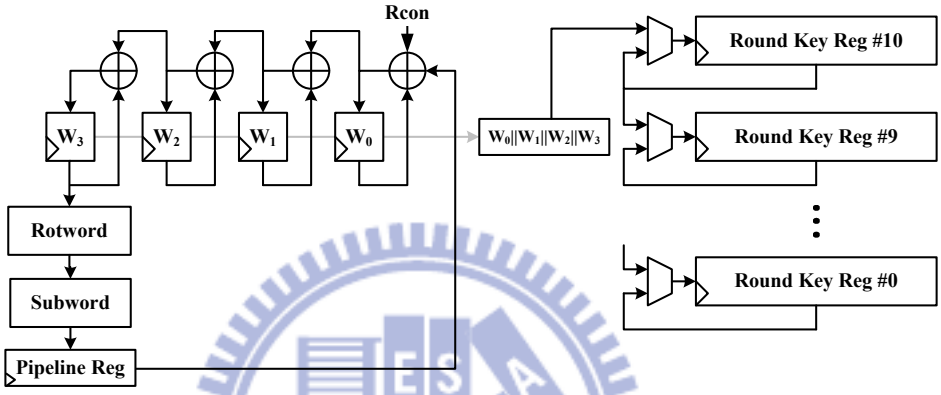
3.2.2 Key Expansion Unit

Round keys required for the AES algorithm is expanded from the secret key by the key scheduling algorithm specified in the standard [8]. The same secret key would always generate the same sequence of round keys. Since the key exchange process is usually time consuming, the secret key for high throughput applications is less frequently changed. Therefore, the round keys can be generated off-line and stored in registers or memories. The key expansion process is invoked only when the secret key is changed.

Fig. 3.8(a) shows the architecture of the off-line key expansion unit for data path without



(a) No sub-stage pipelining



(b) 2 sub-stage pipelining

Figure 3.8: The off-line key expansion unit.

sub-stage pipelining. The 128-bit secret key is loaded into four 32-bit registers W_0, W_1, W_2, and W_3. These four 32-bit registers are concatenated as a 128-bit round key and then stored into the round key registers. All the 10 round key can be generated with following equations:

$$\begin{aligned} W'_0 &= Subword(Rotword(W_3)) \oplus Rcon \oplus W_0 \\ W'_1 &= W'_0 \oplus W_1 \\ W'_2 &= W'_1 \oplus W_2 \\ W'_3 &= W'_2 \oplus W_3 \end{aligned} \tag{3.16}$$

where $Rotword()$ cyclically shifts the word left by one byte and $Subword()$ performs 4 independent SubBytes transformations to the 4-byte word. Rcon is a constant array as specified in the standard [8]. The key expansion unit can generate 128 bits round key in every cycle and shift it into the round key registers. After all the round keys are generated, the key

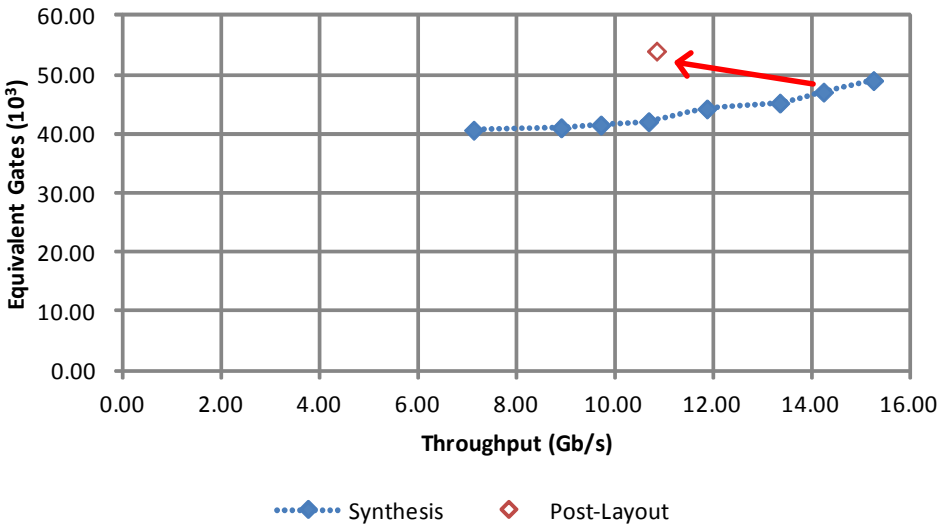


Figure 3.9: Implementation results for 10 Gb/s throughput.

expansion unit halts and all required round keys are stored in round key registers.

Fig. 3.8(b) shows the off-line key expansion unit for 2 sub-stage pipelining data path. Since the longest delay of the 2 sub-stage data path is the same as the delay of the SubBytes transformation, the pipeline register is inserted right after the $Subword()$ to balance the critical path delay. With this architecture, it requires 22 clock cycles to generate all the 11 round keys but it is not a big issue if the secret key is rarely changed.

3.2.3 Implementation Results

The AES engine which targets at 10 Gb/s is implemented using UMC 90 nm CMOS technology. In Fig. 3.9, solid marks indicate synthesis results under different timing constraints. The y-axis is the number of equivalent 2-input NAND gates and the x-axis is the throughput in Gb/s. With 30% timing margin left for back-end flow on the safe side, the timing constraint for the front-end design should be less than 1.5 ns. As a result, the synthesis result using timing constraint 1.5 ns is used to implement the AES engine targeted at 10 Gb/s. The hollow mark indicates the implementation result after back-end design flow. The throughput is degraded by 24%, which is within the 30% design margin, and the area is increased by 15%.

The fully unrolled AES engine that targets at 50 Gb/s is also implemented using UMC

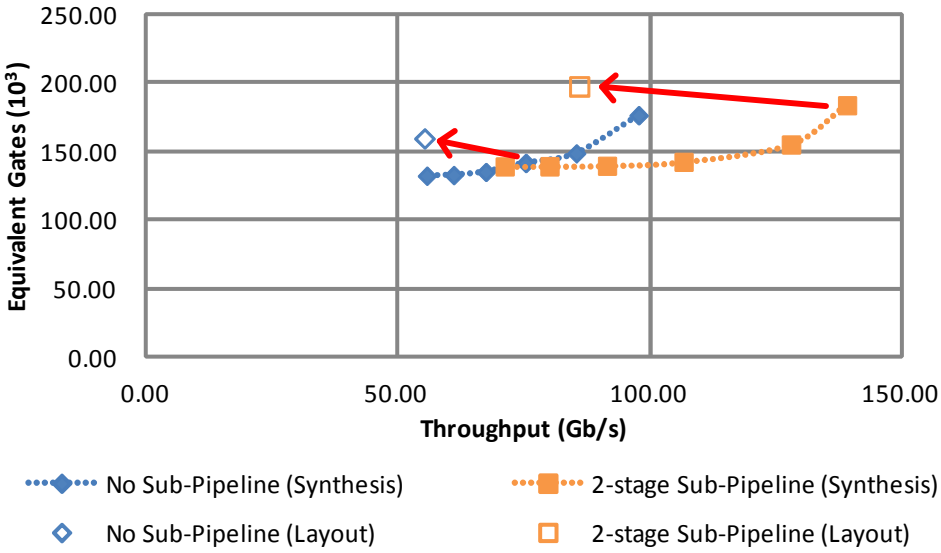


Figure 3.10: Implementation results for 50 Gb/s throughput.

90 nm CMOS technology. The synthesis results for architectures with 1 pipeline stage and 2 pipeline stages per round function are shown in Fig. 3.10. These two architectures are also optimized with different timing constraints and trade-offs between hardware cost and throughput can be illustrated in this figure. To achieve 50 Gb/s, the operating frequency must be higher than 391 MHz, or the critical path delay is less than 2.55 ns. With 30% design margin, the synthesis result of timing constraint 1.7 ns is used to implement the AES engine for 50 Gb/s throughput. The implemented AES can achieve throughput 55.17 Gb/s with 160k equivalent 2-input NAND gates. The throughput is degraded by around 27%, which is also within 30% design margin, and the area is increased by 13%.

The design goal of 2 pipeline stage architecture is to achieve 100 Gb/s using a single core. Under this specification, the operating frequency must be higher than 782 MHz, that is, the critical path delay must be shorter than 1.28 ns. With 30% design margin, the synthesis result with timing constraint 0.9 ns is used to implement the AES engine. However, the implementation result shows that more than 50% design margin is required for such high operating frequency. This is reasonable because in nano-meter technology the wire delay would dominate over gate delay. The throughput can be further increased by using 3 stage pipeline architecture, but such high operating frequency is not easy achieved with cell based design and the design margin left has to be more than 50%. Therefore, two 50 Gb/s AES

Table 3.1: Comparison between AES engines over 10 Gb/s

Synthesis Results						
Design	Technology	Frequency (MHz)	Throughput (Gb/s)	Area (mm^2)	Gates (10^3)	Power (mW)
2-round unrolled	90 nm	666	14.22	0.133	47.08	-
fully unrolled - 1[1]		588	75.29	0.400	141.81	-
fully unrolled - 2[2]		1087	139.13	0.519	183.95	-
Morioka [27]	0.13 μm	909	11.6	-	167.57	1920
Hodjat [28]	0.18μm	245	15.7	-	116	-
		467	59.7	-	313	-
Implementation Results						
Design	Technology	Frequency (MHz)	Throughput (Gb/s)	Area (mm^2)	GEs (10^3)	Power (mW)
2-round unrolled	90 nm	508	10.83	0.152	54.01	39.43
fully unrolled - 1[1]		431	55.17	0.451	159.68	152.06
fully unrolled - 2[2]		671	85.91	0.558	197.59	180.40
Mathew [33]	45 nm	2100	53	0.026	-	125

[1] 1 pipeline stage per round function
[2] 2 pipeline stage per round function

engines would be a better solution for 100 Gigabit Ethernet.

Table 3.1 shows the comparison between AES engines with throughput higher than 10 Gb/s. Note that synthesis results and implementation results are compared separately. Among synthesis results, our design has the lowest hardware cost when targets at over 10 Gb/s throughput. Since different technologies are used, the hardware cost can be compared using the number equivalent 2-input NAND gates.

Mathew *et al.* proposed the first fabricated AES engine with throughput higher than 50 Gb/s [33] with only 0.026 mm^2 silicon area in Intel 45 nm technology. However, the key expansion unit is not included in this design and round keys should be computed by users. Moreover, the 50 Gb/s throughput is achieved when the operating frequency is 2.1 GHz, which is unfeasible for standard cell-based design.

3.3 Low Cost AES Engine for Smart Cards or RFIDs

Throughput requirement in some applications such as smart cards or RFIDs is not so important, instead, the hardware cost is the main consideration. Moreover, the operating frequency

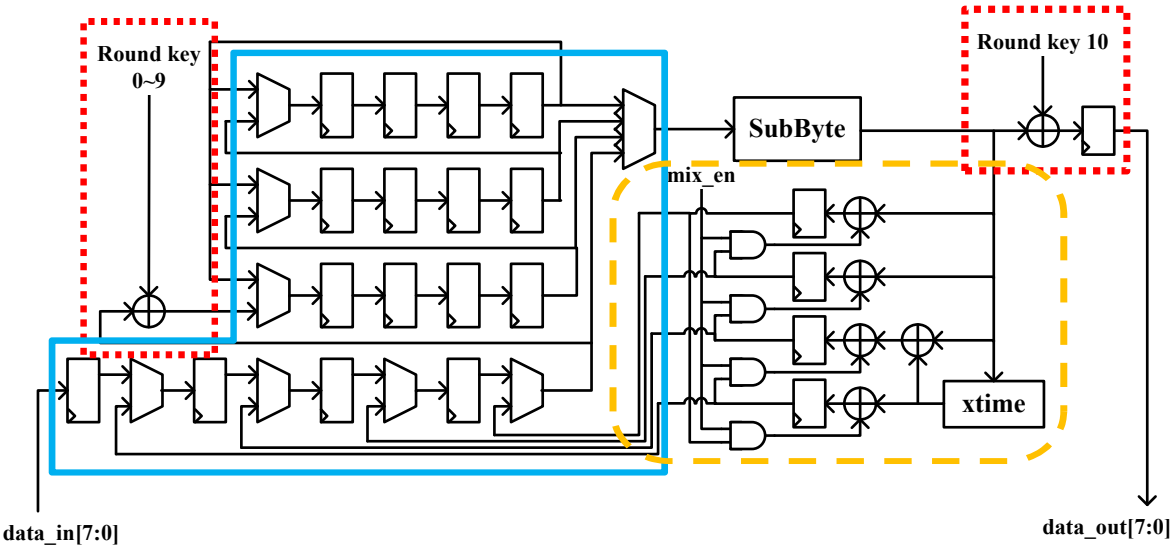


Figure 3.11: 8-bit data-path for low cost AES engine.

in these applications is also much slower than that in high throughput applications, so it does not make sense to optimize the throughput. As introduced in chapter 2, only encryption function is required when the AES algorithm operates under OFB, CFB, and CTR modes and it is sufficient to perform data encryption and decryption. Therefore, the hardware cost can be minimized by designing a very compact AES encryptor.

In this section, a very low cost AES engine with 8-bit data-path unit and key expansion unit is presented. The only design consideration is the hardware cost because the throughput is less important for very low cost applications. The 8-bit data-path unit is presented first and followed by an 8-bit on-the-fly key expansion unit. Implementation results and comparison are given in the last sub-section.

3.3.1 Data-path Unit

Although the data block for the AES algorithm is specified as 128 bits, transformations on the data block can be done byte-by-byte. The hardware cost can be significantly reduced if the data-path width is implemented as 8-bit. The data-path reduction can be done easily for AddRoundKeys and SubBytes because these two transformations are byte-oriented. However, the MixColumns transformation is performed on 32-bit data and the ShiftRows transformation is performed on whole 128-bit data. Fig. 3.11 shows the 8-bit data-path unit for the low

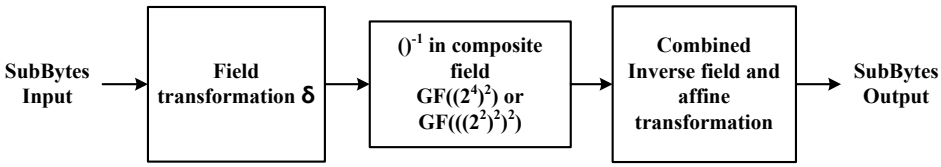


Figure 3.12: The low cost SubBytes transformation.

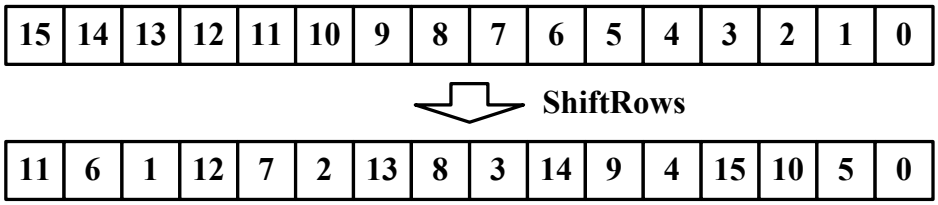


Figure 3.13: The processing order of data bytes after ShiftRows transformation.

cost AES engine. The four transformations used in the AES algorithm is marked by different colors. The AddRoundKeys is marked by red lines, and the MixColumns is marked by the yellow line. The ShiftRows is performed by several multiplexers and is marked by the blue line. Details of the these transformations are given as follows.

AddRoundKeys

The 8-bit data-path contains a dedicated AddRoundKeys for the last round of the AES process. This additional AddRoundKeys consists of 8 2-input XOR gates, which is smaller than an 8-bit 2-to-1 multiplexer. In addition, no extra control logic is required if a dedicated circuit is used for the AddRoundKeys in the last round.

SubBytes

For the 8-bit data-path architecture, only one SubBytes module is required and all the 16 bytes can be performed in serial. For the hardware cost consideration, the composite field based architecture for the SubBytes transformation is adopted. In addition, since the low cost AES requires only the encryption function, the inverse SubBytes transformation can be eliminated to further reduce the hardware cost. Fig. 3.12 shows the block diagram for the SubBytes transformation. Details of the implementation is described in chapter 3.1.1.

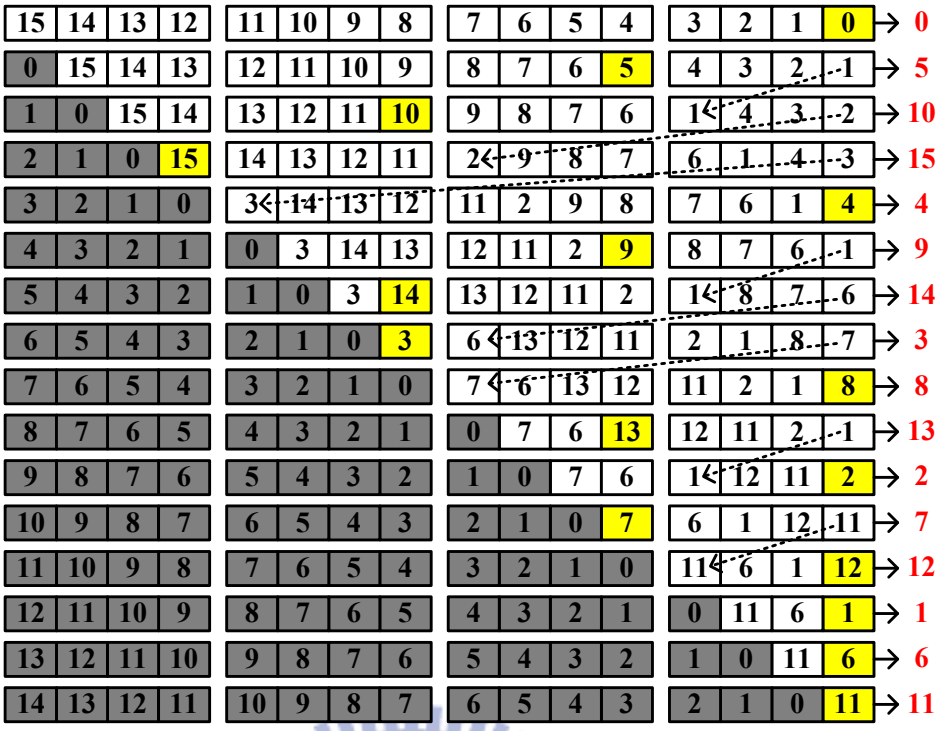


Figure 3.14: The operation of the ShiftRows transformation for the 8-bit data-path.

ShiftRows

The processing order of data bytes is rearranged after the ShiftRows transformation. Fig. 3.14 shows the processing order of all the 16 bytes with the 8-bit data-path architecture. According to the ShiftRows transformation, data bytes in one round is processed with the order 0, 5, 10, 15, 4, 9, 14, 3, 8, 13, 2, 7, 12, 1, 6, 11 instead of 0, 1, ..., 15 after the ShiftRows transformation. Therefore, the ShiftRows in an 8-bit data-path architecture can be performed by shift registers and multiplexers as shown in Fig. 3.11.

Fig. 3.14 shows the operation of shift registers and multiplexers for the ShiftRows transformation. In this figure, data bytes marked in yellow are selected as the output of the ShiftRows and that marked in gray are new data bytes for next round function. These data bytes can be divided into four groups, each of which consists of four data bytes. Note that the output is selected from the least significant byte of these four groups and a 4-to-1 multiplexer can be used to performed this operation. If the output is not selected from the rightmost data byte, then the rightmost data byte is shifted back to replace the selected output byte. For example, in the second cycle, the data byte 5 is selected as the output and the rightmost data

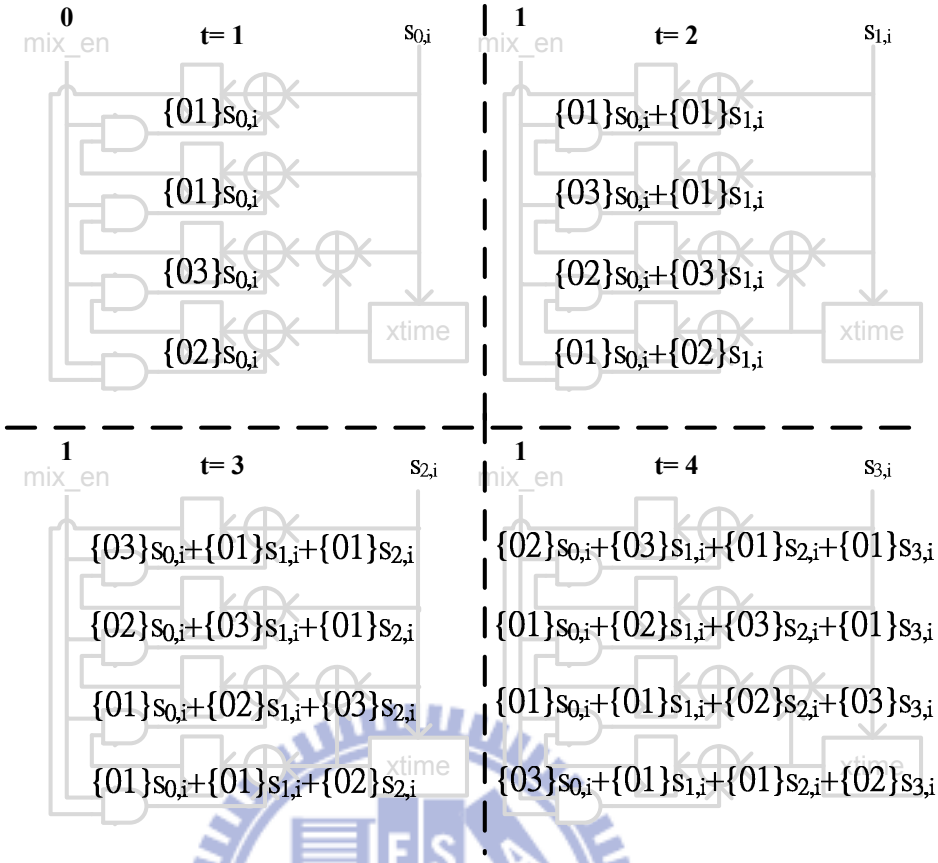


Figure 3.15: The operation of the MixColumns transformation for the 8-bit data-path.

byte 1 is shifted back to the left of data byte 4 to replace data byte 5. The shift-back operation can be done by the 2-to-1 multiplexers prior to each 4-byte group.

MixColumns

The MixColumns transformation is a word-oriented operation and according to equation (2.15), the output of the MixColumns can be written as following equations:

$$\begin{aligned} s'_{0,i} &= \{02\}s_{0,i} + \{03\}s_{1,i} + \{01\}s_{2,i} + \{01\}s_{3,i} \\ s'_{1,i} &= \{01\}s_{0,i} + \{02\}s_{1,i} + \{03\}s_{2,i} + \{01\}s_{3,i} \\ s'_{2,i} &= \{01\}s_{0,i} + \{01\}s_{1,i} + \{02\}s_{2,i} + \{03\}s_{3,i} \\ s'_{3,i} &= \{03\}s_{0,i} + \{01\}s_{1,i} + \{01\}s_{2,i} + \{02\}s_{3,i} \end{aligned} \tag{3.17}$$

Since $s_{0,i}$, $s_{0,i}$, $s_{0,i}$, and $s_{0,i}$ are available in serial, the MixColumns can be performed in four cycles. The contents of four registers at each cycle are shown in Fig. 3.15. Note that the

mix_en is set to 0 for the first cycle and set to 1 for remaining cycles.

3.3.2 Key Expansion Unit

Since the data-path requires only 8-bit round key at each cycle, the data-path width of the key expansion can also be implemented as 8-bit to reduce hardware complexity. The 32-bit operation of equation (3.16) can be modified to 8-bit operation as follows:

$$
\begin{aligned}
B'_0 &= SubBytes(B_{13}) \oplus Rcon[i] \oplus B_0 \\
B'_1 &= SubBytes(B_{14}) \oplus B_1 \\
B'_2 &= SubBytes(B_{15}) \oplus B_2 \\
B'_3 &= SubBytes(B_{12}) \oplus B_3 \\
B'_4 &= B'_0 \oplus B_4 \\
B'_5 &= B'_1 \oplus B_5 \\
B'_6 &= B'_2 \oplus B_6 \\
B'_7 &= B'_3 \oplus B_7 \\
B'_8 &= B'_4 \oplus B_8 \\
B'_9 &= B'_5 \oplus B_9 \\
B'_{10} &= B'_6 \oplus B_{10} \\
B'_{11} &= B'_7 \oplus B_{11} \\
B'_{12} &= B'_8 \oplus B_{12} \\
B'_{13} &= B'_9 \oplus B_{13} \\
B'_{14} &= B'_{10} \oplus B_{14} \\
B'_{15} &= B'_{11} \oplus B_{15}
\end{aligned}
\tag{3.18}
$$

Fig. 3.16 shows the architecture of the 8-bit key expansion unit. Since equations in equation (3.18) have shifting property, shift registers are used to reduce storage spaces. The 128-bit secret key is first loaded byte-by-byte into 16 8-bit registers, each of which stores a

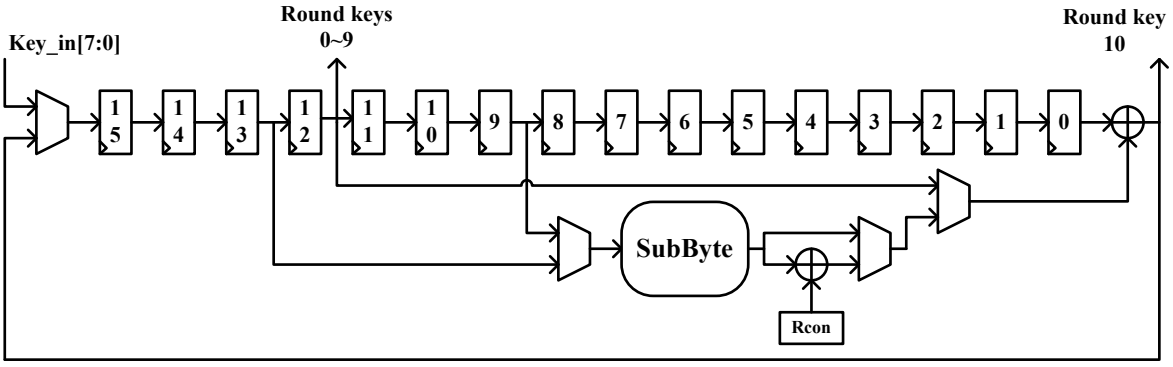


Figure 3.16: The 8-bit key expansion unit.

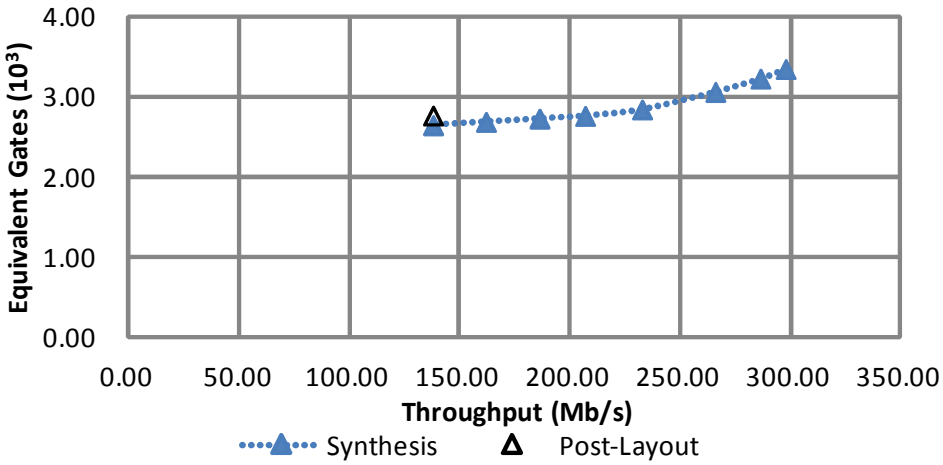


Figure 3.17: Implementation results of low cost AES engine.

key byte. Then the first key byte of the next round key can be generated by the first equation in equation (3.18). The second and third key bytes can be generated in the same way except the operation XOR with Rcon[i]. B_{12} and B_3 are used to generate the forth key byte. After 4-cycle shifting, B_{12} and B_3 will be stored in register byte 9 and 0, respectively. Following key bytes can be obtained by XORing the data bytes in the register byte 12 and 0. Note that a dedicated output is used for the last round key as that described in the data-path unit.

3.3.3 Implementation Results

The low cost AES engine is implemented in UMC 90 nm CMOS technology and implementation results are shown in Fig. 3.17. Synthesis results under different timing constraints are marked in solid and implementation results after back-end design flow is marked in hollow. For low cost AES designs, the throughput is less important and therefore no design margin

Table 3.2: Comparison between low cost AES engines

Design	Technology	Frequency (MHz)	Throughput (Mb/s)	Gates (10^3)	Power (μW)
Proposed	90 nm	185	137.8	2.76	35.2 @1V,1MHz
Feldhofer [35]	0.35 μm	80	9.9	3.40	4.5 @1.5V,100KHz
Hämäläinen [37]*	0.13 μm	152	121.0	3.10	37.0 @1.2V,1MHz
Good [39]	0.13 μm	12	4.3	5.50	0.692 @0.75V,100KHz
Satoh [20]*	0.11 μm	131	311.1	5.40	-

*Synthesis results

is left for the back-end design. The synthesis result with smallest silicon area is used to implement the low cost AES engine. The implementation result shows that the hardware area is slightly increased from 2640 equivalent gates to 2760 gates after back-end design. The maximum throughput remains the same because the timing constraint is much looser.

Table 3.2 shows the comparison with state-of-the-art designs that also target at low cost. The proposed low cost AES has the smallest hardware cost in terms of equivalent gate counts. Throughputs of different designs vary a lot due to different architectures. For example, Feldhofer [35] adopts memory based architecture to store round keys and therefore results in very long latency for one AES encryption process.

The power consumption of the proposed low cost is slightly higher than state-of-the-art designs because of the technology used. The leakage power consumption in deep sub-micron technology usually dominates over the switching power consumption. The leakage power for the proposed AES design is 24.5 μW, which is 70% of total power consumption.

3.4 Median Throughput AES Engine for WLAN

The architecture of median throughput and low cost AES engine shown in Fig.3.18 contains an AES crypto core and an IO buffer. The AES crypto core consists of three major blocks: control unit, key expansion unit, and integrated data process unit. The IO buffer is designed

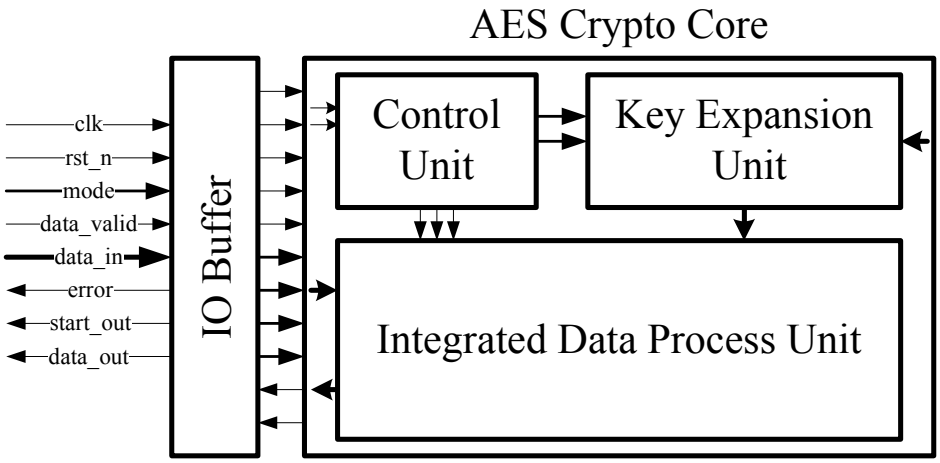


Figure 3.18: Block diagram of the median throughput AES architecture

due to the limitation of pin allocation.

The key expansion unit can be implemented with buffer memories or separate key expansion data-paths [26, 32]. Both approaches require additional cost for decryption. In addition, different key expansion flows for different key lengths would further raise the hardware cost, leading to the difficulty in implementing full-key-length AES.

On the other hand, the data process unit must be able to perform both encryption and decryption. The most straight forward method is to utilize two dedicated data process units, one for encryption and the other for decryption [26, 32]. In our proposed architecture, the encryption and the decryption data-paths are integrated as one data process unit with minimum area overhead.

3.4.1 Data-path Unit

As shown in Fig.3.19, the integrated data process unit contains four basic transformation modules and controlled multiplexers to switch between encryption or decryption. Different data flows are indicated in Fig.3.19 with different lines. The black line indicates data flow for encryption and the gray line indicates data flow for decryption. The dotted line is used to support different operation modes.

To reduce hardware cost, the decryption data-path is combined with the encryption data-path. The SubBytes used in encryption is different from that used in decryption; therefore,

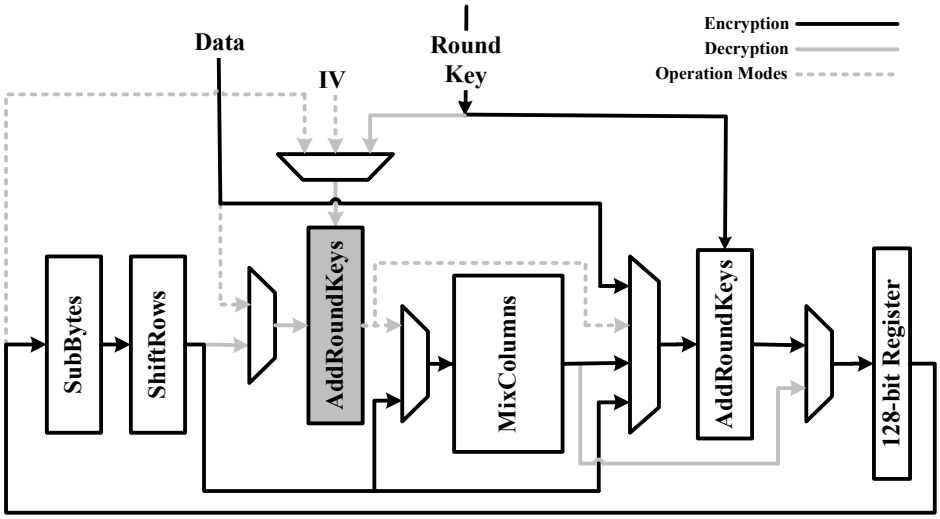


Figure 3.19: Architecture of integrated data process unit

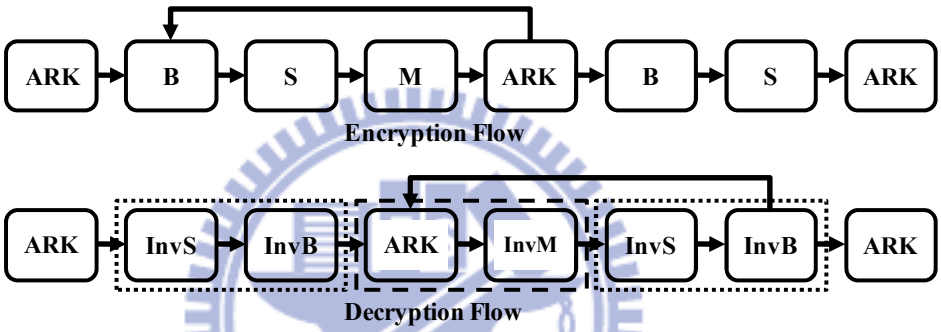


Figure 3.20: The data flow of encryption and decryption process.

total 32 SubBytes modules are required if they are implemented based on LUT. As a result, composite field based SubBytes is a better method to integrate the encryption and decryption data-path [20,54] because the multiplicative inversion in SubBytes can be shared in SubBytes and Inv-SubBytes. To further reduce cost of the integrated data process unit, the data flow of decryption must be modified to merge the data-path of encryptor and decryptor.

To integrate data-paths of encryption and decryption for hardware resource sharing, the data flow of the decryption process is modified. As shown in Fig. 3.20, the data flow of decryption can be exactly the same as that of encryption by changing the processing order of: 1) inverse SubBytes (InvB) and inverse ShiftRows (InvS), and 2) inverse MixColumns (InvM) and AddRoundKeys (ARK). The first modification can be done without additional overhead since both transformations are byte-oriented. However, the swapping of InvM and ARK requires an additional MixColumns transformation in the key expansion unit. This

is because the InvM is defined over field $GF((2^8)^4)$ but the ARK is defined over $GF(2)$. Combined equations of ARK and Inv-/MixColumns in encryption and decryption can be expressed as

$$\begin{aligned} &Encryption \quad (s(x) \times a(x)\ mod\ x^4 + 1) + k(x) \\ &Decryption \quad (s(x) + k(x)) \times a^{-1}(x)\ mod\ x^4 + 1 \end{aligned}$$

Note $a(x)$ and $a^{-1}(x)$ are constant polynomials over $GF((2^8)^4)$ defined in FIPS-197, and $s(x)$ and $k(x)$ represent 32-bit data blocks of processed data and round keys, respectively. The MixColumns transformation can be represented by $s(x) \times a(x)\ mod\ x^4 + 1$ and AddRoundKeys can be expressed by $s(x) + k(x)$. By the distributive law, the equation for decryption can be reformulated as

$$(s(x) \times a^{-1}(x)\ mod\ x^4 + 1) + (k(x) \times a^{-1}(x)\ mod\ x^4 + 1).$$

In this way, the data flow of decryption and encryption would be exactly the same with an additional MixColumns transformation applied to round keys. To eliminate the MixColumns, the processing order of InvM and ARK in the decryption is left unchanged. Note that the AddRoundKeys module in gray shown in Fig.3.19 is used for decryption. This additional AddRoundKeys can be reused to support different operation modes.

3.4.2 Key Expansion Unit

The proposed key expansion architecture is shown in Fig.3.21. It can be applied in both encryption and decryption with different key lengths: 128-, 192-, and 256-bit. The key gen-

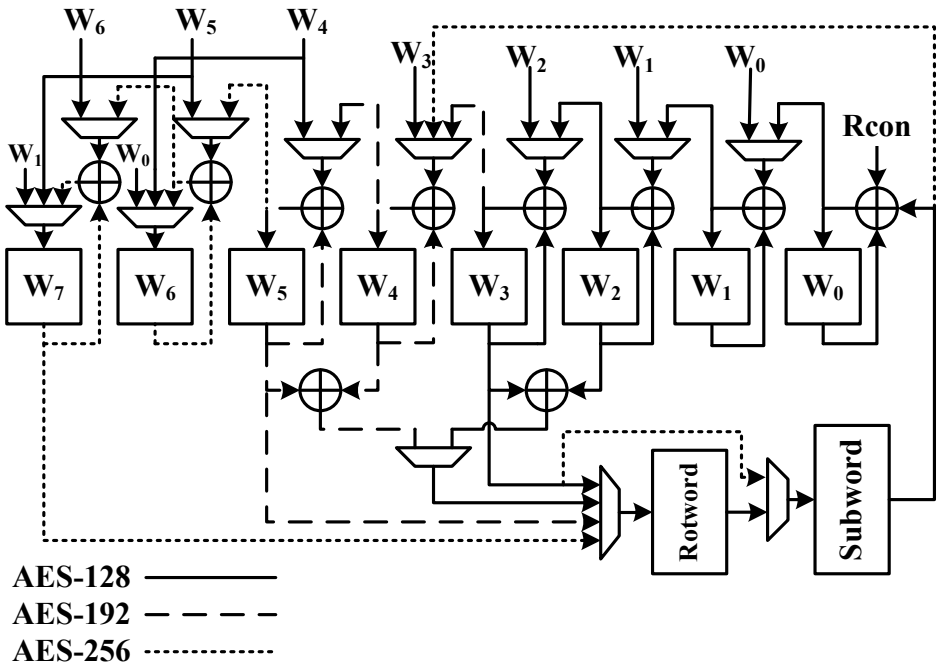


Figure 3.21: On-the-fly key expansion unit for median throughput

erating algorithm of key length 256 for encryption can be modeled as following equations:

$$
\begin{aligned}
W'_0 &= Subword(Rotword(W_7)) \oplus Rcon \oplus W_0 \\
W'_1 &= W'_0 \oplus W_1 \\
W'_2 &= W'_1 \oplus W_2 \\
W'_3 &= W'_2 \oplus W_3 \\
W'_4 &= Subword(W'_3) \oplus W_4 \\
W'_5 &= W'_4 \oplus W_5 \\
W'_6 &= W'_5 \oplus W_6 \\
W'_7 &= W'_6 \oplus W_7
\end{aligned}
\tag{3.19}
$$

Note that W'_n are next round key words and W_n are current round key words, and each 128-bit round key contains four round key words. Subword contains four S-boxes to substitute a word and Rotword shifts the input left by one byte. Rcon is a constant array defined in FIPS-197 [8]. Because round keys used in the decryption flow are in reverse order, the key expansion unit needs to on-the-fly compute these reversely ordered round keys. To generate

such reversely ordered round keys, the last round key is needed and then the following round keys can be generated. That is, it needs to compute W_n from W'_n. The round key expansion process for decryption can be written as follows:

$$
\begin{aligned}
W_0 &= Subword(Rotword(W'_6 \oplus W'_7)) \oplus Rcon \oplus W'_0 \\
W_1 &= W'_0 \oplus W'_1 \\
W_2 &= W'_1 \oplus W'_2 \\
W_3 &= W'_2 \oplus W'_3 \\
W_4 &= Subword(W'_3) \oplus W'_4 \\
W_5 &= W'_4 \oplus W'_5 \\
W_6 &= W'_5 \oplus W'_6 \\
W_7 &= W'_6 \oplus W'_7
\end{aligned}
\tag{3.20}
$$

Note that the first round key used in decryption is the same as the last round key used in encryption. If the key length is 256-bit, at most 14 cycles are required to produce the initial round key in the first decryption operation. To speedup the decryption process, the last round key could be stored in a buffer, then the following decryption process can start immediately when the AES crypto core receives ciphertext blocks.

As defined in FIPS-197 [8], key expansion processes for key length 128 and 192 are quite similar. The data flow of AES-128 is the solid line shown in Fig.3.21, and the dash line is the additional data flow for AES-192. The complexity of key expansion unit is raised significantly when considering key length 256. As shown in above equations, the round key expansion process needs two Subword modules in AES-256. The additional Subword leads to higher hardware cost and also increases the critical path. Since only 128 bits are required as round keys, the key expansion process of AES-256 can be divided into two phases. In encryption, $\{W'_0, W'_1, W'_2,$ and $W'_3\}$ are computed in the first phase and $\{W'_4, W'_5, W'_6,$ and $W'_7\}$ are generated by using the same Subword module in the second phase. In decryption, $\{W_4, W_5, W_6, W_7\}$ are generated in the first phase and $\{W_0, W_1, W_2, W_3\}$ are generated

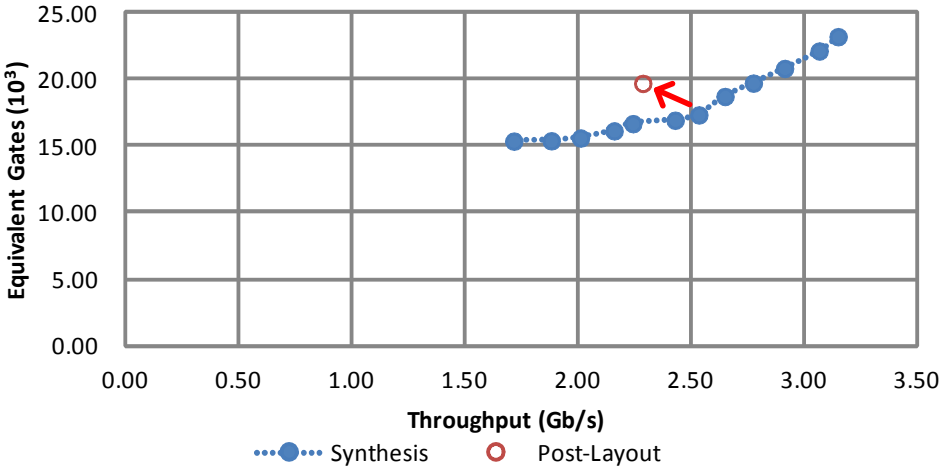


Figure 3.22: Implementation results of median throughput AES engine.

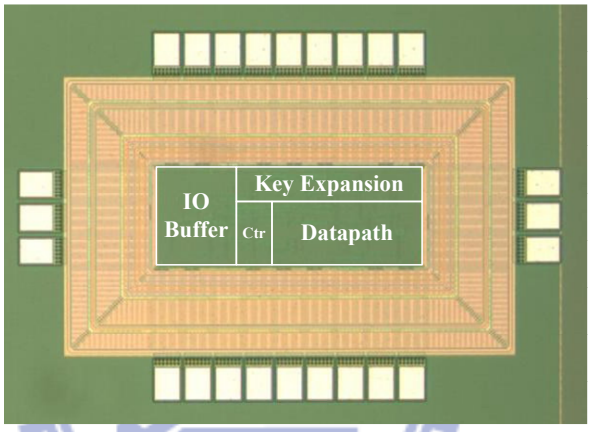


Figure 3.23: Die micrograph for median throughput AES engine

in the second phase. Note that the required $W_6' \oplus W_7'$ when computing W_0 have already been stored in W_7 in the first phase. The dotted line in Fig.3.21 indicates the data flow in AES-256.

3.4.3 Implementation Results

The median throughput and low cost AES engine is implemented in UMC 90 nm CMOS technology and implementation results are shown in Fig. 3.22. Synthesis results under different timing constraints are marked in solid circles and implementation results after back-end flow is marked in hollow circle.

This design is also fabricated in UMC 90 nm technology and the die micrograph is shown in Fig. 3.23. The core area is $0.069mm^2$ where 63% of the fabricated chip is the AES

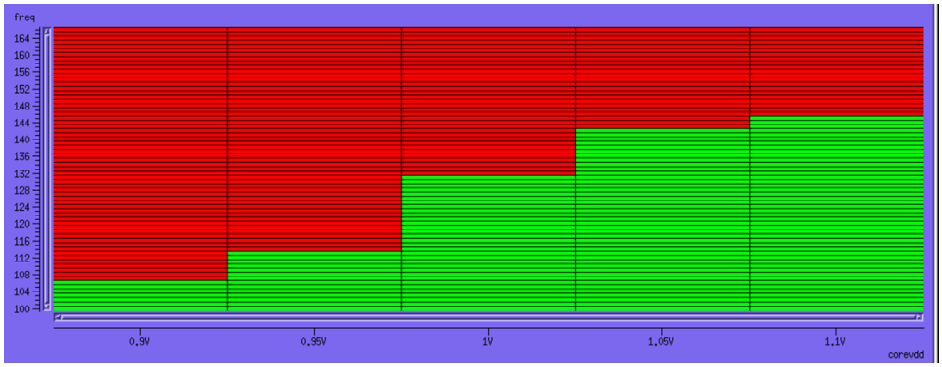


Figure 3.24: Shmoo plot for median throughput AES engine

Table 3.3: Comparison between median throughput AES engines

Design	Technology	Frequency (MHz)	Throughput (Gb/s)	Gates (10^3)	Power (mW)	Mbps /K-gates
Proposed	90 nm	131.8	1.69	15.58	5.02	108.5
Gürkaynak [26]	0.25μm	166	2.12	119	600	17.8
Hodjat [29]	0.18μm	330	3.84	54	79	48.6
Lin [32]	0.13μm	333	4.27	40.9	86.2	49.5

core and 37% is the IO buffer. The average power consumption of this chip is 5.02 mW when operating at 131.8 MHz. Fig. 3.24 shows the Shmoo plot for the maximum operating frequency under different conditions. The maximum operating frequency can be 145 MHz when the supply voltage is 1.1V and the maximum operating frequency is 106 MHz with 0.9V supply voltage. All FIPS-197 test patterns and random patterns are fully tested.

In Table 3.3, the proposed design is compared with some other designs in terms of throughput and hardware cost. Only designs with measured results are listed in this table. The performance metric, Mbps/K-gate, is also listed to show the efficiency and the normalized performance is also given in the table for comparison.

Gürkaynak [26] proposed a full-duplex design which can achieve throughput up to 4.24 Gb/s in ECB and CBC modes and up to 2.12 Gb/s in OFB and CFB modes. However, the encryptor and the decryptor are not easy to operate in parallel because it requires at least 768 IO pins to make both encryptor and decryptor operate at the same time. Hodjat [29] proposed a design adopting table look-up based S-boxes to reduce the critical path but also leads to higher hardware cost. Lin [32] proposed a two-stage pipelining architecture

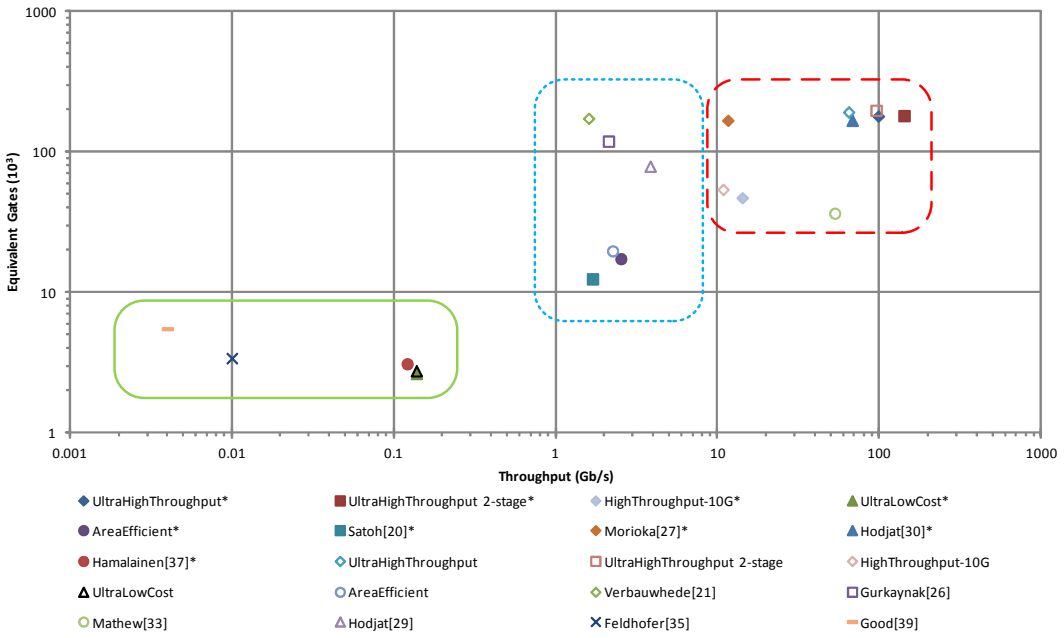


Figure 3.25: Summary of area throughput trade-offs of AES engines.

with high throughput. However, separately designed encryptor and decryptor result in much higher hardware cost. Moreover, the pipelining architecture also limits the implementation of feedback operation modes such as OFB and CFB.

3.5 Summary

In this chapter, different architectures of AES engines are proposed for different applications. Fig. 3.25 shows the area and throughput trade-offs for different architectures. The detailed implementation technology and results can be found in previous sections. Note that designs with synthesis results are indicated by solid marks while designs with implementation results are indicated by hollow marks.

For designs with throughput higher than 10 Gb/s, the proposed design can achieve the highest throughput without considering the hardware cost. For designs with median throughput, the proposed AES engine has almost the smallest hardware cost. Satoh's [20] design is smaller than the proposed because only AES-128 and ECB mode is supported. However, the proposed design can support AES-128, AES-192, and AES-256 under all different modes of operation. As last, for the low cost AES engine, the proposed can also outperform others in

Table 3.4: Design summary on different AES architectures

Implementation Technique	High Throughput	Area Efficient	Low Cost
Unrolling	2 - fully	Non	Non
Pipelining	Yes	No	No
Data Width	128	128	8
Key Expansion	Off-line	On-the-fly	On-the-fly
SubBytes Implementation	LUT	Composite Field	Composite Field

terms of equivalent gate counts.

In this section, we also give a brief summary on design selection for different considerations in Table 3.4. This table gives the prospective selection of implementation techniques for different architectures.

Chapter 4

Power Analysis Attacks

Side-channel attacks exploit leaked physical information from chips to analyze possible keys and have become efficient ways to attack cryptographic devices. In 1996, Kocher proposed attacks that utilize the timing or power information with controlled data from attacked devices [18]. Since the power information can be easily obtained by existing equipments, power analysis has become the most common attacking methods.

The security weakness of hardware-based crypto chips is further investigated by Kocher *et al.* in 1999 using simple power analysis (SPA) and differential power analysis (DPA) [19]. In [58], the SPA and DPA are more thoroughly introduced and discussed. For the SPA, attackers observe a single power trace of attacked devices to guess a part of the secret key. Because SPA utilizes key-dependent characteristics of power trace, this kind of attack is more suitable to attack asymmetric encryption algorithms. However, in symmetric encryption algorithms such as the AES algorithm, characteristics of power traces are independent of secret keys. The DPA attack can more efficiently disclose secret keys by the power consumption information leaked from cryptographic devices.

In this chapter, the SPA and DPA are briefly introduced in the first two sections. Since the SPA is rarely used for the AES chips, only DPA attack results on the fabricated chip are shown in the last section.

4.1 Simple Power Analysis

The SPA attack is described by Kocher *et al.* in [19] as: ”SPA is a technique that involves directly interpreting power consumption measurements collected during cryptographic operations.” That is, the attacker tries to disclose the secret key of cryptographic devices by observing collected power traces. However, the attacker must have detailed knowledge about the cryptographic device under attacking, including the hardware architecture, operation timing and so on. In addition, the noise from the measurement environment or equipments would also make the SPA become more challenging.

4.1.1 General Description

SPA is a method to disclose the secret key when only very small number of power traces are available. For example, consider the scenario when a user uses a smart card for payment. A malicious card reader can record the power consumption of the smart card in the payment process. The SPA can be categorized into single-shot SPA and multiple-shot SPA by the number of power traces used to disclose the secret key. In single-shot SPA attack, only one power trace is recored and this is the only information for secret key disclosure. In multiple-shot SPA attack, a few power traces are recorded and used to disclose the secret key.

For the multiple-shot SPA, power consumption of the same pattern can be recorded multiple times, or even different patterns can be recorded. The most important advantage of multiple-shot SPA is that noises from measurement environment or equipments can be reduced by computing the mean of collected traces.

Although the single-shot and multiple-shot SPA differs by the number of power traces recorded, the fundamental idea of SPA attacks is the same. Attackers observe the single or averaged power trace and disclose the secret key of cryptographic devices directly or indirectly.

Figure 4.1: The power trace of a straightforward RSA implementation.

4.1.2 SPA on Asymmetric Ciphers

Since SPA extracts the secret key by observing a single or multiple power traces, the secret key can be disclosed if the power trace is key dependent. That is, different secret key leads to different characteristics of the power trace. Therefore, SPA is more suitable for asymmetric ciphers such as RSA or ECC. This is because the fundamental operation of asymmetric is exponentiation, which can be decomposed into a series of multiplication and squaring as follows:

Input: Message M, Secret key E=$\{E_{n-1}E_{n-2}...E_0\}_2$
Output: Ciphertext C = M^E
P = M, C = 1;
for *i = n-1 down to 0* **do**
    C = C $\times$ C mod N;
    if E_i == *1* **then**
        C = C $\times$ P mod N;
    end
end
return C;

Algorithm 2: Exponentiation

As shown in this algorithm, the key bit E_i == 0 and E_i == 1 will lead to different operation. If the key bit equals to 1, then a modular multiplication will be performed and this additional operation can result in different power consumption characteristics. Fig. 4.1 shows the power trace of an exponentiation operation from [59]. From the power trace, the key bit 1 and 0 can be easily distinguished by different power consumption characteristics.

Techniques for resisting SPA are usually quite simple to implement. Once procedures that require secret key for conditional branch can be avoided, then the power consumption characteristics can be masked.

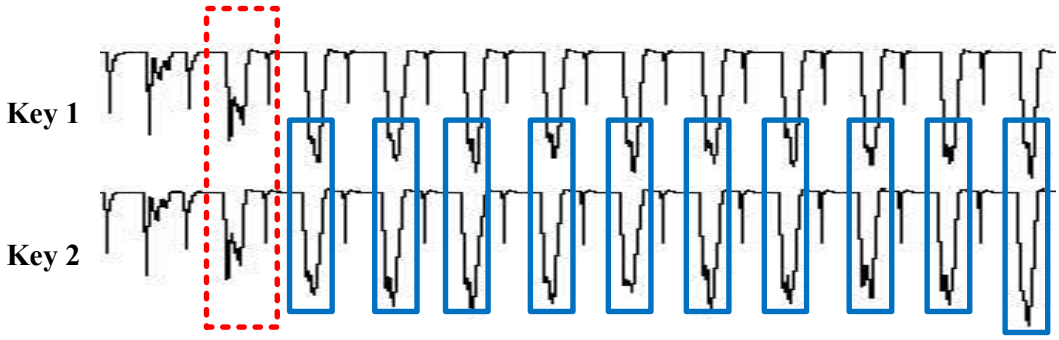


Figure 4.2: The power trace of an AES implementation.

4.1.3 SPA on Symmetric Ciphers

Operations of symmetric ciphers are usually independent of the secret key and then the SPA is not easy to disclose the secret key by observing power traces. Fig. 4.2 shows the power trace of one AES encryption operation. The period for the initial round for the AES algorithm is enclosed by the red rectangle and that for the other 10 rounds can also be easily distinguished as enclosed by blue rectangles. The power consumption characteristic for the initial round is significantly different from others because only AddRoundKeys is performed in the initial round. From this power trace, operation of the AES can be distinguished but the secret key can not be disclosed by observing the power trace.

To demonstrate the SPA resistance, the power traces of the same plaintext with two different secret keys are recorded. The characteristic of the peak for each round is slight different if different keys are used. However, this difference is affected by all the 128 key bits and the attacker must be able to distinguish all 2^{128} possible characteristics to disclose the secret key. As a result, the SPA is less important for the symmetric cipher implementation and the SPA resistance is not addressed in this dissertation.

4.2 Differential Power Analysis

DPA attacks are based on statistics to find the correlation between the measured power consumption and the predicted power consumption. Because the power prediction model takes into account both secret keys and processed data blocks, the statistical calculation result can

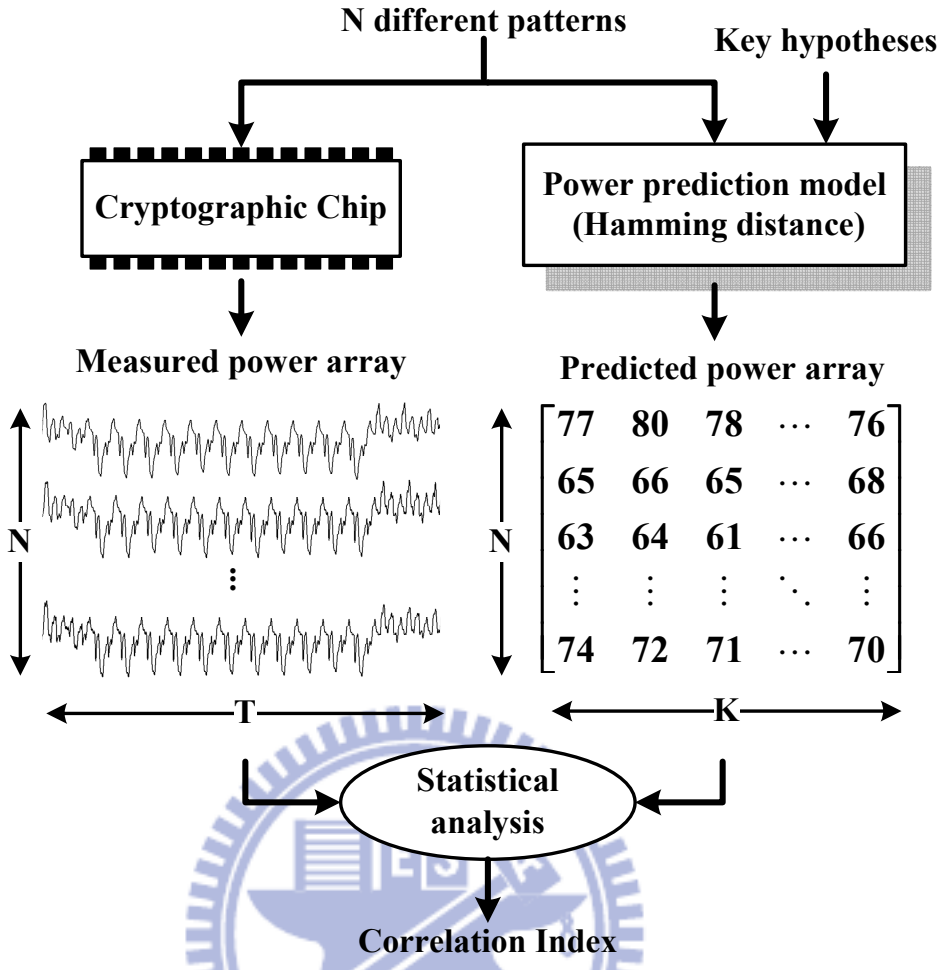


Figure 4.3: The flow of the DPA attack.

be used to disclose the possible secret key in the cryptographic device.

4.2.1 DPA Attack Flow

A brief flow of the DPA attack is shown in Fig. 4.3. The attacker prepares N different patterns for en-/decryption and records power traces of these patterns. These N power traces, which consist of T sample points, are firstly arranged as a N-by-T measured power array for further processing. The same plaintexts and all possible key hypotheses are used to generate predicted power values by an appropriate power prediction model. The power prediction model is a method to determine possible power consumption either in the behavior or in the algorithm level. The most common used power models are the Hamming-distance model and the Hamming-weight model. The DPA attack efficiency is largely dependent on the power model used. Since the AES algorithm is byte-oriented, the 128-bit secret keys can be

divided into 16 8-bit sub-keys, which largely reduce the key space from 2^{128} to 16×2^8. As shown in Fig. 4.3, power values are arranged as a N-by-K array, where N is the number of plaintexts and K is the number of all possible key hypotheses. Every column of the array indicates predicted power values for all N plaintexts of a specific key hypothesis.

The final step of the DPA attack is to find the correlation by statistical calculation such as difference-of-means or correlation coefficient. For the difference-of-means, introduced by Kocher *et al.* in [19], power traces are divided into two groups depending on corresponding power values. The difference of the average of these two groups then indicates the correlation between power traces and power values. The correlation coefficient, which is proposed by Brier *et al.* [60], considers not only the means but also the variances to reduce the required the number of measurements.

4.2.2 Power Models

For DPA attack, intermediate values of each key hypothesis are mapped to predicted power values for analysis. This is a kind of power simulation of cryptographic devices and obtained power values may be in some way related to the actual power consumption. The Hamming-weight (HW) and Hamming-distance (HD) are two most often used power models and they are briefly introduced in this sub-section.

Hamming-Weight Model

The HW model is a simple model that applied if attacker have less knowledge about the cryptographic device. In case of the HW model, the power consumption is assumed to be proportional to the number of bits that are set in the intermediate value. As a result, the HW model is not suitable for describing the power consumption of CMOS circuits because the power consumption of CMOS circuits depends on the number of transitions instead of the processed value.

In practice the Hamming weight of an intermediate value is still somewhat related to the power consumption. For example, a pre-charged or pre-discharged data bus will be all 1s or

0s before processing the data. Then the Hamming weight of the processed data is related to the number of transitions.

However, it is not always the case that the data bus will be pre-charged or pre-discharged and the HW model is not sufficient for these cases. HD model then must be used for interpreting intermediate values to predicted power values.

Hamming-Distance Model

The basic idea of the HD model is to count the number of transitions, including $1 \rightarrow 0$ and $0 \rightarrow 1$, in the circuit during a specific period. Then the number of transitions can be used to describe the power consumption of the circuit in this period. Although the number of transitions is not the actual power consumption in Watt, it is still proportional to the actual power consumption.

Two assumptions are made when using the HD model to predict the power consumption of the circuit. First, all $1 \rightarrow 0$ and $0 \rightarrow 1$ transitions contribute to the same power consumption. Although in practice the power consumption of different transitions is slightly different, it make the power model easier with this assumption. Second, all $0 \rightarrow 0$ and $1 \rightarrow 1$ contribute equally to the power consumption. Parasitic capacitances of wires and cells are eliminated in the HD model. The static power consumption such as internal power or leakage power are also ignored for simplicity.

Since the HD model is simple and can better describe the CMOS circuit, it is commonly used for power simulations. The power simulation based HD model can provide a rough estimation of the power consumption of real chip. The HD model can be formalized by $HD(v_0, v_1) = HW(v_0 \oplus v_1)$, where v_0 and v_1 are two successive values that appear on a data bus in different time instance.

4.2.3 Statistical Analysis

The most commonly used statistical analysis methods are the difference-of-means [19] and correlation coefficients [60]. The difference-of-means is introduced first and then followed

by the correlation coefficients in this sub-section.

Difference-of-Means

The basic idea of difference-of-means is to determine the relationship between columns of measured power array and predicted power array as shown in Fig. 4.3. In order to check if a key hypothesis K_i is correct or not, the power trace array is divided into two sets according to power values of the key hypothesis. That is, N predicted power values for the key hypothesis K_i is used to categorize N power traces. If the power value is larger than a threshold, then the corresponding power trace is grouped to set 1. If the power value is lower than a threshold, the corresponding power trace is grouped to set 0. The means of these two sets are calculated by following equations:

$$\begin{aligned} m_{1i,j} &= \frac{1}{n_{1i}} \cdot \sum_{l=1}^{N} \left\lfloor \frac{h_{l,i}}{\theta_h} \right\rfloor \cdot t_{l,j} \\ m_{0i,j} &= \frac{1}{n_{0i}} \cdot \sum_{l=1}^{N} (1 - \left\lfloor \frac{h_{l,i}}{\theta_h} \right\rfloor) \cdot t_{l,j} \end{aligned} \tag{4.1}$$

where i and j are indices for key hypothesis K_i and sample point of the power trace. θ_h is a pre-determined threshold, which is usually set to half of the maximum of $h_{l,i}$, used to group power traces. n_{1i} and n_{0i} are the number of power traces in set 1 and set 0, respectively. The vector m_{1i} and m_{0i} are used to denote the mean of rows in set 1 and set 0, respectively. If the key hypothesis K_i is incorrect, then the grouping is somewhat similar to a random method and these two sets would have similar means. If the key hypothesis K_i is correct, then the difference of m_{0i} and m_{1i} would be significant at some point in time.

The difference between m_{0i} and m_{1i} indicates the correlation between power values of key hypothesis K_i and power traces at some time. The difference will be significant at the point when the data and the secret key is processed. For the other time instances, the difference between m_{0i} and m_{1i} will approach zero. If the key hypothesis is incorrect, then the difference will essentially be zero for all time instances.

Correlation Coefficient

The correlation coefficient is a common statistical method to determine the relationship between data. Therefore, the relationship between the measured power and predicted power can also be determined by this method. Brier *et al.* proposed the method which is suitable for the DPA attack. The equation used to find the correlation coefficient is as follows:

$$r_{i,j} = \frac{\sum_{n=1}^{N}(h_{n,i} - \bar{h_i}) \cdot (t_{n,j} - \bar{t_j})}{\sqrt{\sum_{n=1}^{N}(h_{n,i} - \bar{h_i})^2 \cdot \sum_{n=1}^{N}(t_{n,j} - \bar{t_j})^2}}, \tag{4.2}$$

where $h_{n,i}$ is the power value of key hypothesis K_i for the n-th input pattern; $t_{n,j}$ is the measured power at sample time j for the n-th input pattern; $\bar{h_i}$ and $\bar{t_j}$ are mean values of $h_{n,i}$ and $t_{n,j}$ for total N input patterns. The correlation coefficient is thus an index used for disclosing the secret key. For example, if the key hypothesis is wrong, predicted power values and the measured power would be independent and the correlation coefficient would approach 0 at all sampling time. On the contrary, if the key hypothesis is correct, the correlation coefficient would be much higher at the sample time performing related operations.

4.3 Security Evaluation

In this section, the practical environment for the DPA attack are introduced, including how to set up the measurement environment and how to build the power models. After that, the DPA attack on the AES circuit is conducted with different statistical analysis methods. Moreover, the DPA attack results on the real chip is also given in the this sub-section. At last, an estimation method for measurements to disclosure is illustrated in the last sub-section.

4.3.1 Measurement Environment

Fig. 4.4 shows the block diagram for the measurement environment. Each block are briefly described and equipments used are also specified.

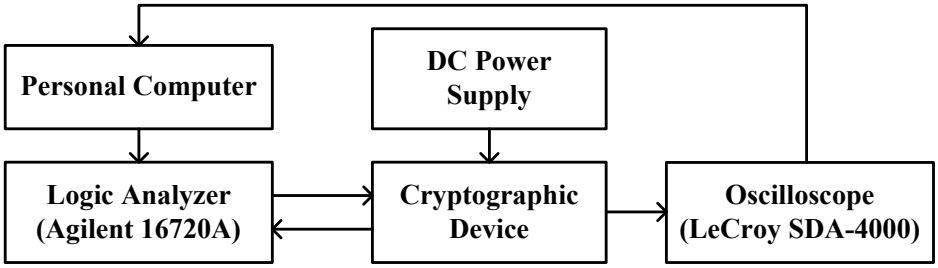


Figure 4.4: The block diagram of the measurement environment.

- **Cryptographic Device** - The cryptographic device consists of hardware implementation of cryptographic algorithms such as the AES algorithm. The device provide an interface to communicate with the logic analyzer and oscilloscope for verification and power recording. I/O pins of the device are connected to the logic analyzer and a small resist is shunt between the power supply and the device for the oscilloscope measuring the current consumption.

- **Logic Analyzer** - The logic analyzer is an equipment that supplies input vectors to the cryptographic device and captures output signals from the device for verification. These input and output vectors are in the csv format and can be derived in the simulation phase. The logic analyzer can also capture outputs of the cryptographic device and compare the captured data with that in the csv file.

- **Power Supply** - The power of the cryptographic device can be supplied by external DC power supplies. The power supply can be configured to supply 2.5 V and 1 V for the pad and core of the cryptographic device.

- **Oscilloscope** - The digital oscilloscope uses a probe to measure the cross voltage of the resistor in between the power supply and the cryptographic device. Stored power traces can be transfered to the personal computer via network or portable storage devices for further processing and analysis.

- **Personal Computer** - The personal computer receives power traces recorded by the oscilloscope and analyzes these power traces by a C/C++ program. The personal computer also generates the csv file for the logic analyzer to verify the function of the

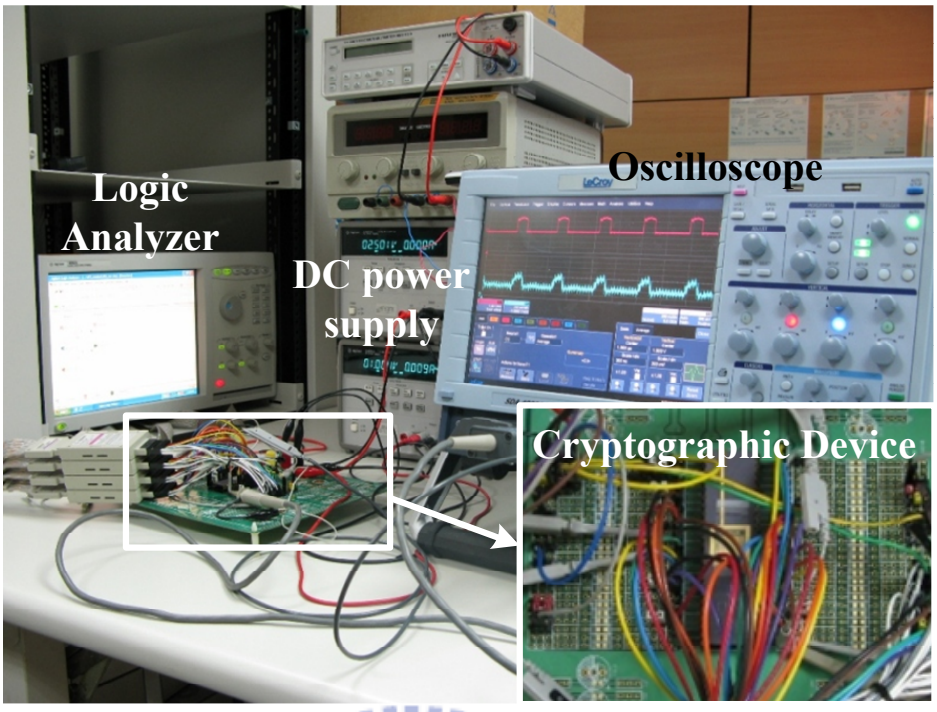


Figure 4.5: The test chip measurement and analysis environment setup.

cryptographic device.

The practical measurement and analysis environment setup is shown in Fig. 4.5. The power consumed by the AES crypto core is measured by the cross voltage of a small resistor between the DC power supply and the test chip. Input vectors are supplied by the Agilent 16720A pattern generator and output signals are analyzed by the Agilent 16902A logic analyzer. The cross voltage of the resistor is measured and recorded by the LeCroy SDA-4000 with 5 GHz sampling frequency. The recorded power traces are then transfered to a personal computer for analysis.

4.3.2 Power Models

In practical scenarios, the user can only access to the ciphertext of a cryptographic device. As a result, the round key for the last round is disclosed and then the secret key can be found by the inverse key scheduling algorithm. To disclose the last round key, the power model is build to predict the power consumption between the 10^{th} round and 11^{th} round as shown in Fig. 4.6. Variables used to build the predicted power array are the known ciphertext and the

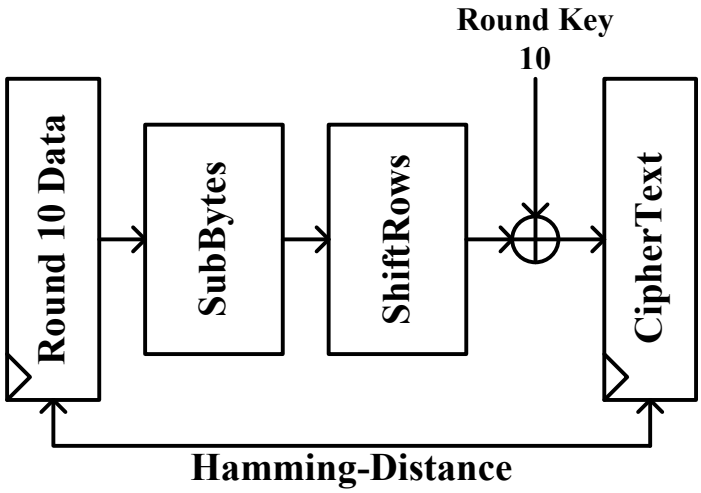


Figure 4.6: The last round model for power simulation.

last round key.

In the AES algorithm, each data byte is processed independently, and therefore the secret key can be disclosed byte-by-byte. All the 128-bit secret key can be disclosed by 16 DPA attacks. However, although only 8-bit secret key is disclosed at a time, the power consumption measured from the chip is still contributed from all the 128-bit secret key. The Hamming distance of data registers between the 10^{th} round and last round is used as the predicted power value. Since the data register in the last round contains the ciphertext, the predicted power value can be obtained if the data register in the 10^{th} round is known. To calculate the 128-bit data of 10^{th} round, the 128-bit round key, which contains the 8 guessed bits and other 120 random bits, is built for a series of inverse transformations. Each ciphertext is used to generate 256 power values by the 256 key hypotheses and then the predicted power array can be built from N ciphertexts and 256 key hypotheses.

4.3.3 DPA Results

To demonstrate the efficiency of the DPA attack, the attack flow is applied to the proposed AES engine presented in chapter 3. To evaluate the DPA resistance of AES engine before chip fabrication, a simulation-based platform is set up for early stage estimation. Since the estimation is based on transistor level simulation in a noiseless environment, a lower bound

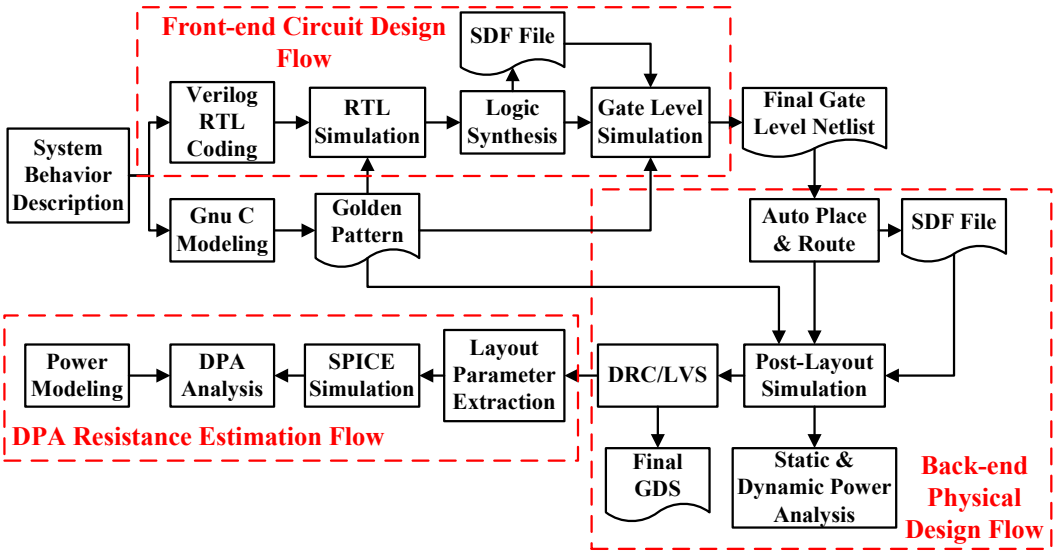


Figure 4.7: The design and estimation flow for the AES engine.

of the DPA resistance for a fabricated chip can be obtained.

Fig. 4.7 shows the design and estimation flow for the AES engine. The design flow can be partitioned into three major parts: 1) front-end circuit design flow, 2) back-end physical design flow, and 3) DPA resistance estimation flow. The major modification is the DPA resistance estimation flow. For the DPA resistance estimation design flow, the SPICE model of the AES engine is extracted from the layout and then simulated power traces can be obtained by transistor-level simulation. Then simulated power traces and the predicted power model are used to perform the DPA attack. Since power traces from transistor level simulation represent the ideal case of the power consumption, the DPA resistance obtained from these power traces can be considered as the lower bound for a real chip. The simulation based analysis results based on both the different-of-means and the correlation coefficients are shown first and then followed by the real chip attack.

Results of Difference-of-Means

Fig. 4.8 shows the analysis results from transistor level simulation based on the difference-of-means. Fig. 4.8(a) shows the difference-of-means for all 256 key hypotheses with 8000 power traces. The result for incorrect key hypotheses are plotted in gray bars and that for the correct one is highlighted in black. The difference-of-means of the correct key hypothesis is

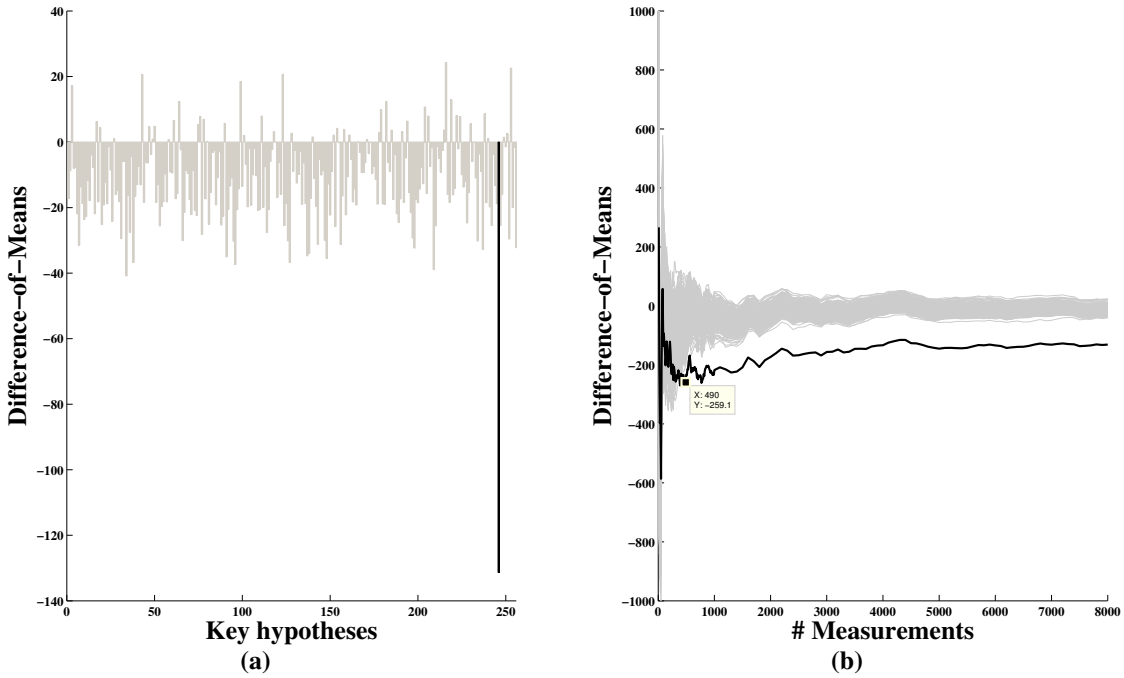


Figure 4.8: The DPA results based on difference-of-means.

significantly higher than all other incorrect ones and therefore the attacker can disclose the secret key byte by identifying the peak from this figure.

Fig. 4.8(b) shows the relationship between the difference-of-means and the number of measurements. The difference-of-means of all incorrect key hypotheses with different number of measurements are plotted in gray lines while that of the correct one is highlighted in black. This figure shows that only 490 measurements are required to disclose the key byte in a noiseless environment. This figure also shows that the correct key byte becomes easier to be distinguished with the number of measurements increased.

Results of Correlation Coefficient

In addition to the difference-of-means, the same power traces are also analyzed by the correlation coefficient method. Fig. 4.9 shows analysis results from transistor level simulation based on the correlation coefficient. In Fig. 4.9(a), the correlation coefficients of all 256 key hypotheses are obtained with the same 8000 power traces. The correlation coefficients of incorrect key hypotheses are again plotted in gray bars and that of the correct key is highlighted in black. This figure also shows that the correct key is more easier distinguished when the

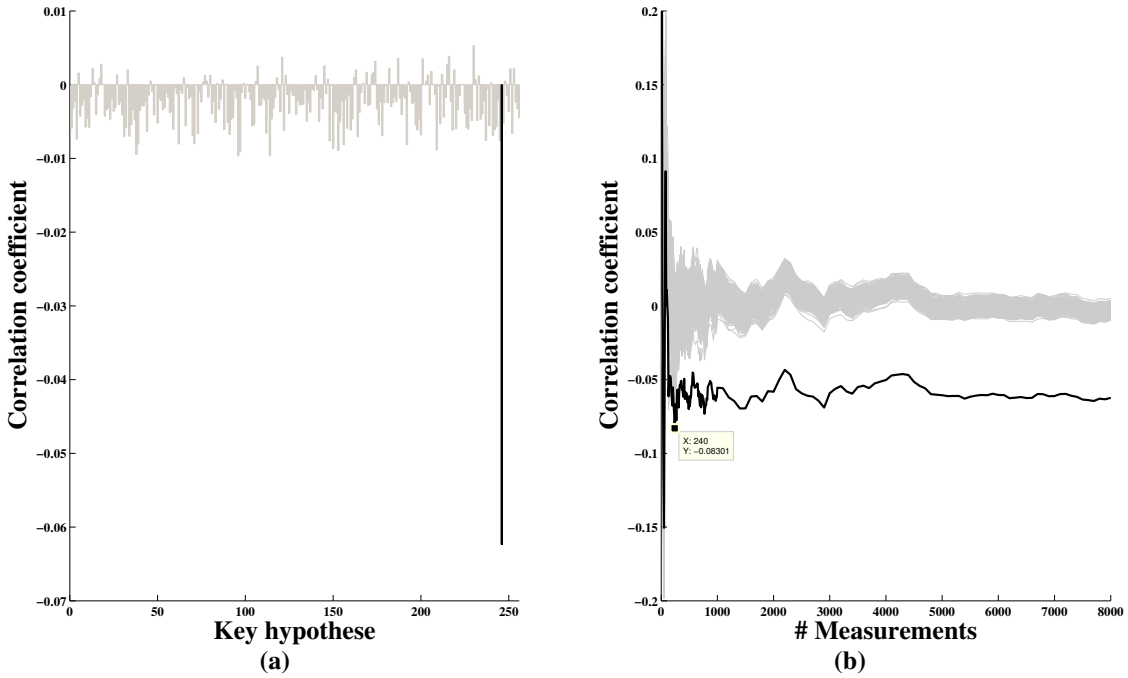


Figure 4.9: The DPA results based on correlation coefficient.

correlation coefficient is used for statistical analysis with the same power traces.

Fig. 4.9(b) shows the relationship between correlation coefficients and the number of measurements. Correlation coefficients of all incorrect key hypotheses with different number of measurements are again plotted in gray lines while that of the correct key is highlighted in black. This figure shows that the minimum number of measurements to disclose the secret is reduced to only 240 power traces, which is about 50% more efficient than the difference-of-means method. When the number of measurements increased, the difference between the correct key and incorrect keys are also more significant. As a result, the correlation coefficient method has higher efficiency than the difference-of-means method and the following analysis results are based on the correlation coefficient for DPA attack.

Real Chip Analysis Results

The DPA attack on a real AES chip is shown in Fig. 4.10. Fig. 4.10(a) shows correlation coefficients of all 256 key hypotheses with 10000 measurements. Correlation coefficient of the correct key hypothesis is reduced from -0.06 to -0.03 because of the non-ideal power supply and noises in the measurement environment. Although the correlation coefficient

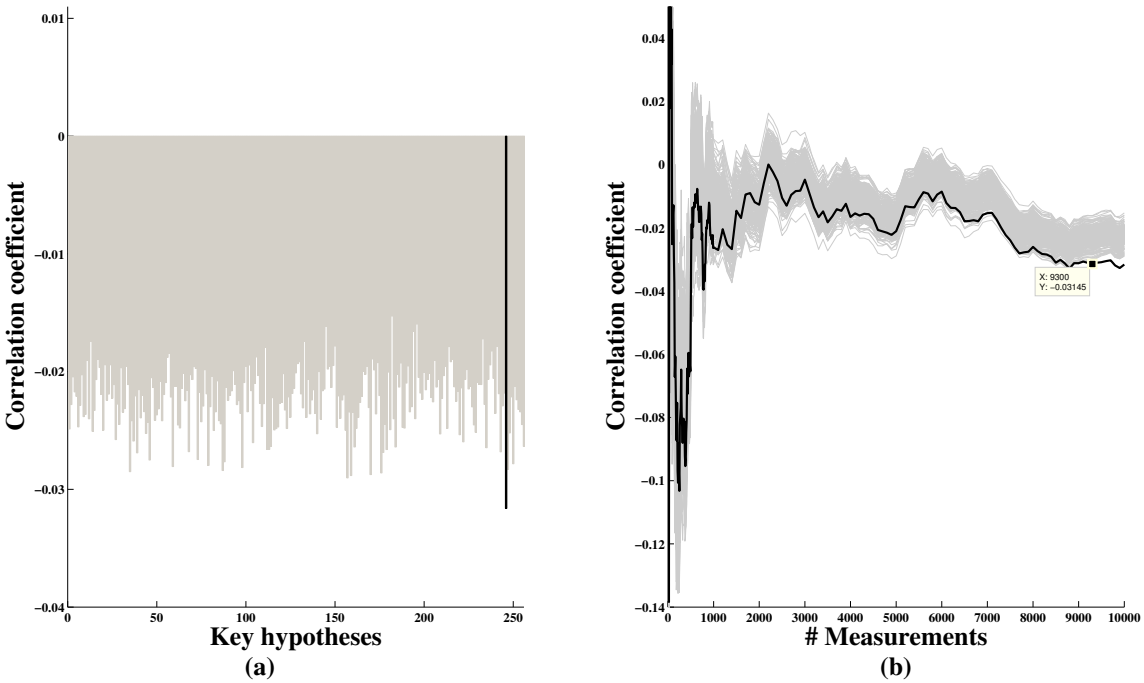


Figure 4.10: The DPA attack results on real AES chip.

is significantly reduced, it is still possible to distinguish the correct key hypothesis from incorrect ones with 10000 measurements.

In Fig. 4.10(b), correlation coefficients versus the number of measurements of the correct key hypothesis is plotted as the black line while that of other incorrect key hypotheses are plotted as gray lines. When noises in the measurement environment are involved, the number of measurement to disclose the secret key is increased largely from 240 measurements to about 9300 measurements, which is about 38 times higher than the noiseless simulation environment.

4.3.4 Accessing the Number of Needed Power Traces

In DPA attacks, the more power traces acquired, the more precise the DPA result is. However, how many traces does the attacker need to disclose the secret key is an important issue. Since the DPA attack is based on statistical analysis, it is not easy to give a precise answer. In fact, only a rough estimation can be provided based on statistical calculations. The estimation is based on two assumptions: 1) the DPA attack is successful if there is a significant peak for correlation index $r_{i,j}$, and 2) the number of traces that are needed to see a peak exclusively

Table 4.1: Quantiles $z_{1-\alpha}$ of the normal distribution for different $1-\alpha$.

$1-\alpha$	$z_{1-\alpha}$	$1-\alpha$	$z_{1-\alpha}$
0.850	0.842	0.990	2.326
0.900	1.036	0.995	2.576
0.950	1.282	0.999	3.090
0.975	1.960	0.9999	3.719

Table 4.2: The estimated number of traces for different $\rho_{ck,ct}$.

$\rho_{ck,ct}$	$n_{\alpha=0.0001}$	$\rho_{ck,ct}$	$n_{\alpha=0.0001}$	$\rho_{ck,ct}$	$n_{\alpha=0.0001}$
0.900	16	0.090	3400	0.009	341493
0.800	26	0.080	4307	0.008	432206
0.700	40	0.070	5630	0.007	564519
0.600	64	0.060	7668	0.006	768378
0.500	95	0.050	11049	0.005	1106471
0.400	157	0.040	17273	0.004	1728870
0.300	292	0.030	30720	0.003	3073559
0.200	676	0.020	69140	0.002	6915526
0.100	2751	0.010	276606	0.001	27662152

depends on simulation based correlation coefficient $\rho_{ck,ct}$, where ck is the index of the correct key and ct is the index of the correct time.

The number of traces needed to distinguish the correlation coefficient $\rho_{ck,ct}$ from $\rho = 0$ with confidence α can be assessed by the following equation:

$$n = 3 + 8\frac{z_{1-\alpha}^2}{ln^2\frac{1+\rho_{ck,ct}}{1-\rho_{ck,ct}}}, \tag{4.3}$$

where $z_{1-\alpha}^2$ the quantile of the normal distribution for some values of α as listed in Table 4.1 and $1-\alpha$ is error probability. To illustrate the relationship between the estimated number of traces and the $\rho_{ck,ct}$, Table 4.2 shows calculated values according to equation (4.3) with error probability $\alpha = 0.0001$.

We now can assess the number of power traces for the AES engine based on the simulation based correlation coefficient $\rho_{ck,ct}$. In Fig. 4.9, we can see that the correlation coefficient for the correct key is stabilized after around 5000 measurements and the simulated correlation coefficient is about -0.06 (negatively related). Then we can estimate the number of traces needed by looking up Table 4.2 and the result shows that 7668 traces are needed to disclose

the secret key. Compared with the DPA result of the real chip, total 9300 traces are required to disclose the secret key. Hence, the estimated value could be used as a rough assessment.

Chapter 5

Design of DPA Resistant AES Engines

In this chapter, brief introductions to power-masking and power-hiding methods are given first. Then the proposed DPA countermeasure circuit based on pseudo random number generator is presented. A security issue of the pseudo random based architecture is analyzed and then followed by an improved architecture based on true random like number generator.

5.1 Previous Works on DPA Countermeasure

In the last decade, several methods in algorithm level or circuit level have been proposed [42, 61–70] to counteract the DPA attacks. In general, these methods can be categorized as power-masking or power-hiding. The fundamental concept of these methods is to break the correlation between the real power consumption and the predicted power consumption. Power-masking methods [61–67] incorporate a random mask into processed data blocks to make the power consumption of cryptographic devices unpredictable. The random mask is added into data blocks at the beginning of the encryption and is removed at the end of the encryption. Although power-masking methods can effectively increase the DPA resistance, the hardware cost is increased by at least 2 times and the performance is degraded by at least 50% [71–73]. Power-hiding methods use new logic cells [68–70] or switching capacitors [42] to make the power consumption of different transitions be equal. That is, different plaintexts would lead to the same power consumption characteristics. The DPA resistance of

these methods can also be effectively increased, but the cost is also 2 times higher with more than 50% performance degradation.

5.1.1 Power Masking Methods

For power-masking methods, data blocks can be masked either in the algorithm level or in the circuit level. In the algorithm level masking, the non-linear transformation, SubBytes, has to be modified to facilitate the recovery of correct data at the end of the encryption. The first masked SubBytes transformation is proposed by Akkar and Giraud [61]. In addition to the random mask used at the beginning of the encryption process, another independent mask is added into data blocks before performing the SubBytes transformation and removed at the end of the SubBytes transformation. Trichina *et al.* proposed a simplified method based on Akkar and Giraud's by reusing the original random mask instead of using a new random mask [62]. However, an all-zero data byte would not be masked and therefore these two masked SubBytes are still vulnerable to the DPA attack. As a result, different masked SubBytes are proposed to solve the all-zero case [64–66]. Instead of the algorithm level modification, Trichina proposed a SubBytes masked in the circuit level by using masking cells [63]. Later, Suzuki proposed another masking cells called random switching logic to reduce the area overhead and shorten the critical path [67].

Masking in Algorithm Level

Fig. 5.1 illustrates the concept of the masking methods in the algorithm level. A random mask is added into data blocks by bit-wise XOR operation for data encryption. The random mask on linear operations can be easily removed because $f(A \oplus X) = f(A) \oplus f(X)$. However, the mask on non-linear operations can not be easily removed because $f(A \oplus X) \neq f(A) \oplus f(X)$. As a result, the data recovery circuit contains all the linear operations of the AES algorithm to remove the masked data. The non-linear part of the AES algorithm is modified as shown in Fig. 5.2. Since the mask can not be easily removed, the modified inversion preserves the random mask X after the operation. The modified inversion computes

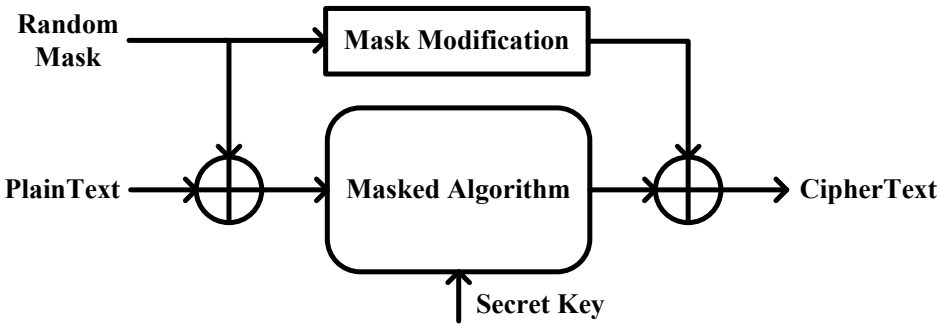


Figure 5.1: The concept of the masking methods in algorithm level.

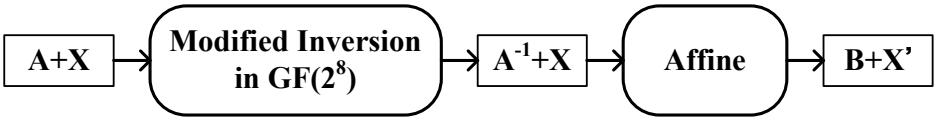


Figure 5.2: The modification of non-linear parts in AES algorithm.

$A^{-1} + X$ instead of $(A + X)^{-1}$. That is, the masking in algorithm level uses mathematical methods to obtain $A^{-1} + X$ from $A + X$.

Fig. 5.3 shows the method proposed by Akkar and Giraud [61]. The $A+X$ is transformed to $A \times X$ before the inversion operation. The $(A \times X)^{-1}$ is then transformed to $A^{-1} + X$ after the inversion operation. Another independent random mask Y is required to mask the power of the inversion operation.

Oswald *et al.* try to find the $A^{-1} + X$ by direct finite field arithmetic [65]. The multi-

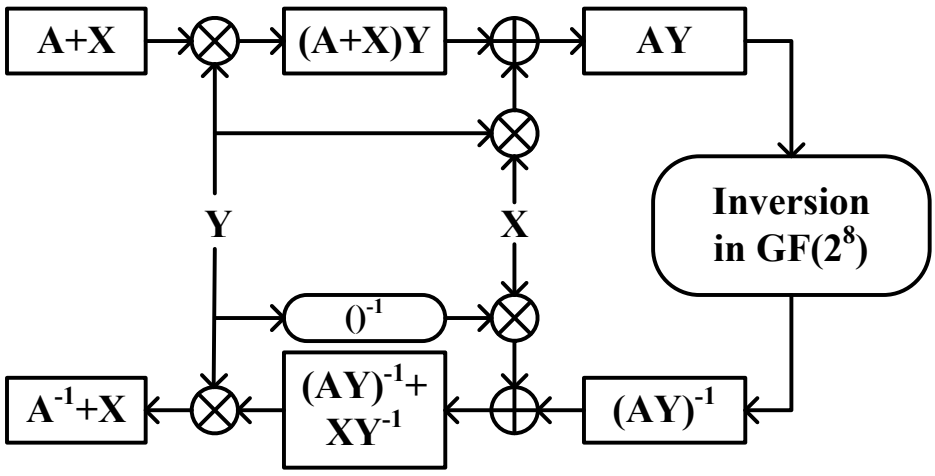


Figure 5.3: The modified SubBytes by Akkar and Giraud.

plicative inverse over $GF((2^4)^2)$ can be computed by the following equations:

$$\begin{aligned}
(a_h x + a_l)^{-1} &= a'_h x + a'_l \\
a'_h &= a_h \times d' \\
a'_l &= (a_h + a_l) \times d' \\
d' &= (a_h^2 \times p_0) + (a_h \times a_l) + a_l^2 \\
d &= d^{-1}
\end{aligned} \tag{5.1}$$

In the similar manner, the masked multiplicative inversion over $GF((2^4)^2)$ can be computed by the following equations:

$$\begin{aligned}
((a_h + m_h)x + (a_l + m_l))^{-1} &= (a'_h + m'_h)x + (a'_l + m'_l) \\
a'_h + m'_h &= f_{ah}((a_h + m_h), (d' + m'_d), m_h, m'_h, m'_d) \\
&= a_h \times d' + m'_h \\
a'_l + m'_l &= f_{al}((a'_h + m'_h), (a_l + m_l), (d' + m'_d), m_l, m'_h, m'_l, m'_d) \\
&= (a_h + a_l) \times d' + m'_l \\
d + m_d &= f_d((a_h + m_h), (a_l + m_l), p_0, m_h, m_l, m_d) \\
&= a_h^2 \times p_0 + a_h \times a_l + a_l^2 + m_d \\
d' + m'_d &= f_{d'}(d + m_d, m_d, m'_d) \\
&= d^{-1} + m'd
\end{aligned} \tag{5.2}$$

Masking in Circuit Level

Instead of modifying the AES algorithm, the masking in the circuit or cell level can be achieved with new logic cells. Fig. 5.4 shows the masked-AND logic proposed by Trichina to perform masked operation [63]. The a_x and b_y are masked by x and y, respectively. With this masked-AND logic, a masked result ab_z (a AND b masked by z) can be obtained without knowing values of a and b. Since the operation of the AES is over finite field, XOR and AND

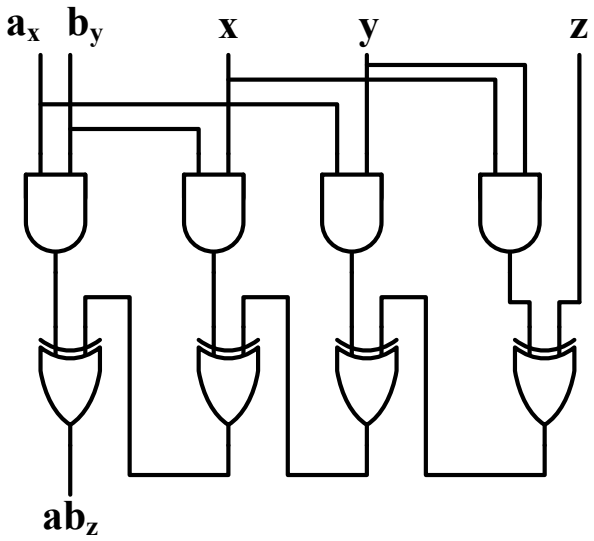


Figure 5.4: The masked-AND logic.

gates are sufficient to implement the AES engine. Therefore, only masked-AND is required to resist the DPA attack.

5.1.2 Power Hiding Methods

For power-hiding methods, new cells are adopted to balance the power consumption of different transitions. The sense amplifier based logic (SABL) [68], wave dynamic differential logic (WDDL) [69], dual-rail circuit [70], and switching capacitors [42] are of this kind of countermeasure methods. Dual-rail cells balance the power consumption by redundant complementary logic cells. In addition, since these new logic cells have complementary outputs, differential routing is also required in physical implementation, which leads to much higher design effort. The switching capacitors proposed by Tokunaga and Blaauw [42] uses an array of capacitors to isolate the current supplied by the power supply and that consumed by cryptographic devices. Since power-hiding methods are algorithm independent, they can also be applied to other encryption algorithms such as RSA or ECC to resist the DPA attacks.

Wave Dynamic Differential Logics (WDDL)

To make the power consumption be constant for different transitions, two conditions must be satisfied: 1) a logic gate has exactly one charging event per clock cycle and 2) the logic

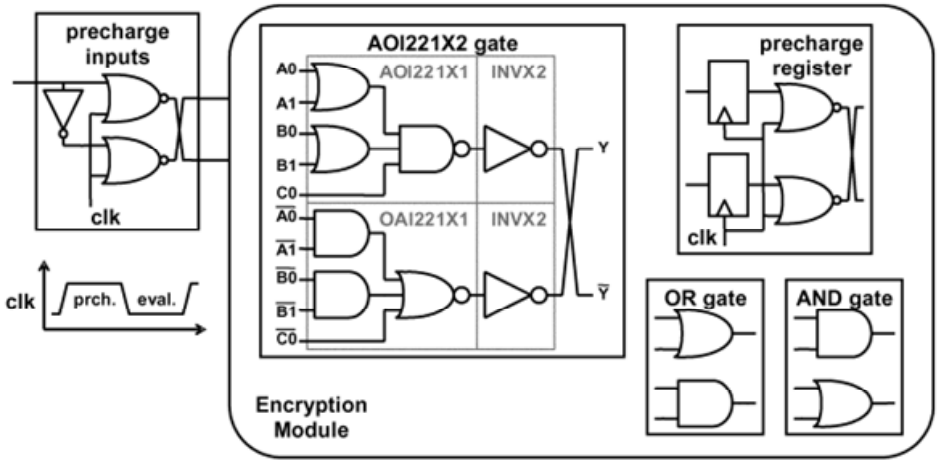


Figure 5.5: The wave dynamic differential logic.

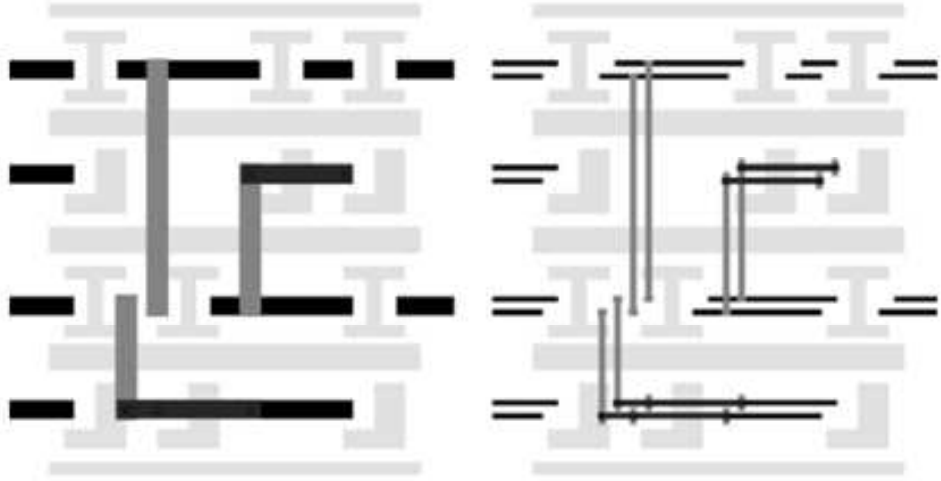


Figure 5.6: The differential routing technique.

gate charges a constant capacitance in that event. The first condition can be achieved by the WDDL cells as shown in Fig. 5.5 and the second condition can be achieved by the differential routing as shown in Fig. 5.6 [41].

The WDDL gates consist of two positive but complementary gates as shown in Fig 5.5. The complementary gate computes the false output by using complementary input signals. For example, the AND gate now consists of an AND gate and a OR gate. The AND gate works for the original function and the OR gate generates a false output $f = a + b$. In this way, there will be exactly one transition among two complementary outputs.

In addition to the same transition numbers per clock cycle, the amount of capacitance charged must be the same. That is, the output loading for both complementary outputs must

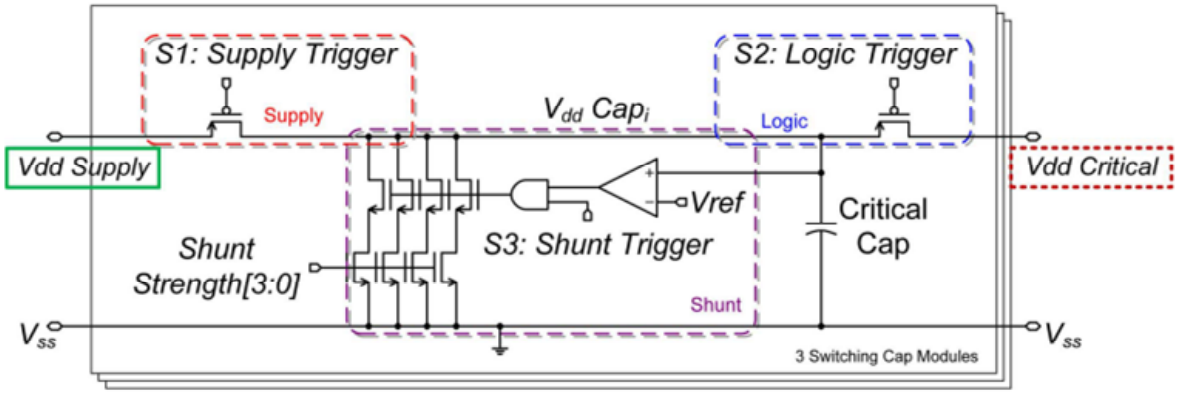


Figure 5.7: The switching capacitors.

be equivalent. The loading capacitance is composed of the output capacitance of the gate, the wire parasitic capacitance, and the input capacitance of the load. Since the output capacitance and input capacitance for the gates can be balanced by selecting appropriate cells, the wire parasitic capacitance is the most challenging part to balance the total output loading. Fig. 5.6 shows the method called differential routing technique to balance the wire load presented in [41].

Switching Capacitors

Switching capacitors are used to make the power consumption of different operations constant to resist the DPA attack. Fig. 5.7 shows the circuit proposed by Tokunaga and Blaauw to equalize the current consumption. The switching cycle starts with supply trigger S1. In this cycle, the capacitor is charged to full potential by directly connecting the power supply to the capacitor. In the next cycle, supply trigger is opened and logic trigger is closed. Then the current demanded by the circuit is now supplied by capacitors. At last, the shut trigger is closed for the capacitor reaching a known state. These three states are performed repeatedly and can be done in parallel by an array of capacitors.

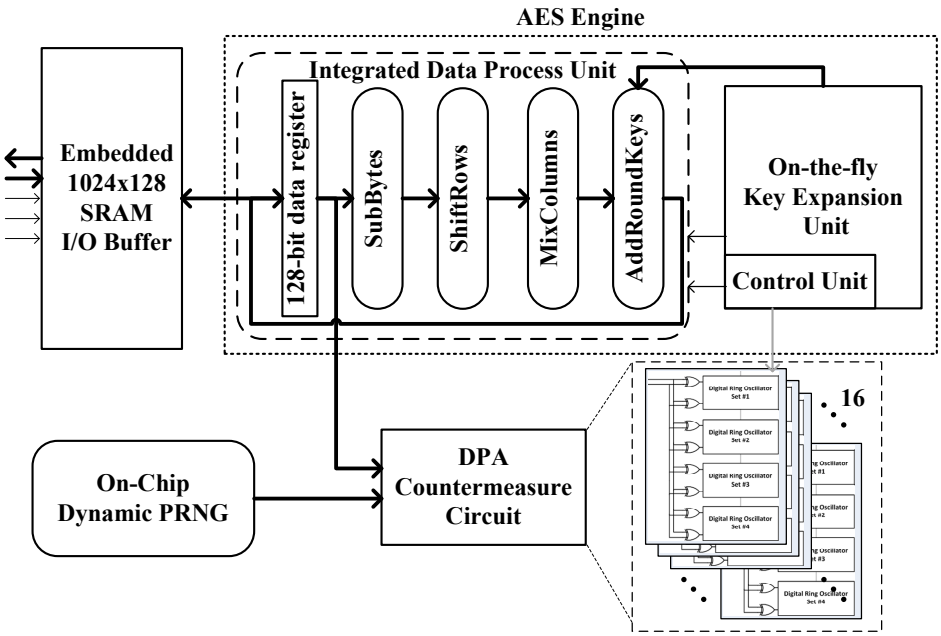


Figure 5.8: Block diagram of the pseudo random based DPA-resistant AES chip.

5.2 Pseudo Random Based DPA Countermeasure Circuit

Block diagram of the proposed DPA-resistant AES chip, including an I/O buffer, an AES engine, an on-chip dynamic pseudo random number generator (PRNG), as well as the proposed DPA countermeasure circuit, is shown in Fig. 5.8. The I/O data buffer is a 1024×128 embedded SRAM with 512 entries for plaintext and 512 entries for ciphertext. The AES engine can support encryption and decryption for key lengths 128, 192, and 256 specified in FIPS-197 [8]. The on-chip dynamic PRNG and the DPA countermeasure circuit work together to counteract the DPA attacks. Details of these components are given in the following subsections.

5.2.1 Ring Oscillator Based DPA Countermeasure Circuit

Both DPA countermeasure and on-chip PRNG circuits are designed to dynamically change power signatures of the AES crypto core. Unlike power-masking methods that change data blocks by adding random masks, the proposed DPA countermeasure circuit and the on-chip PRNG can work in parallel with the AES core, implying that no extra delay will be induced in the critical path.

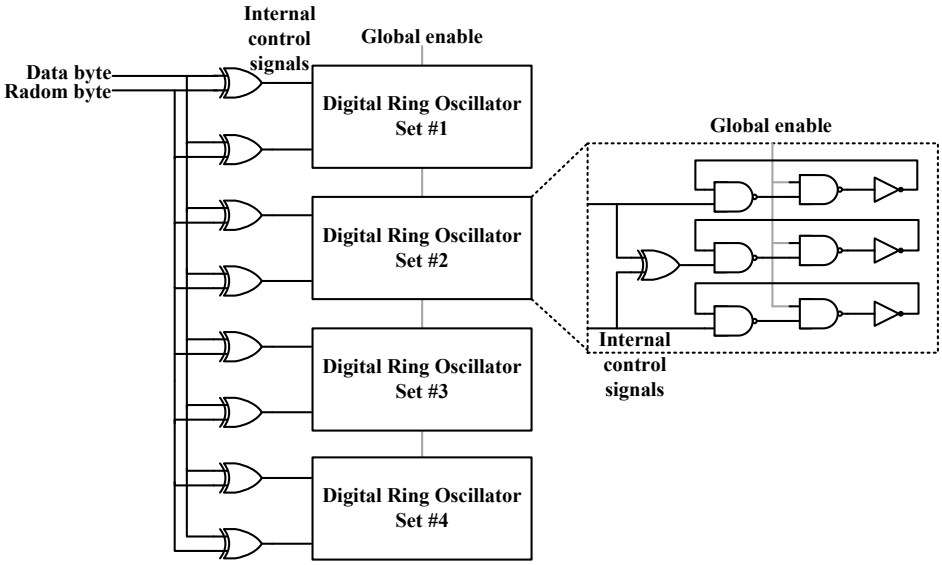


Figure 5.9: Detailed structure of the DPA countermeasure sub-circuit.

The DPA countermeasure circuit is composed of 16 identical sub-circuits and each sub-circuit is based on digitally controlled ring oscillators as shown in Fig. 5.9. All these 16 sub-circuits will be enabled at the same time when the AES core is activated. Each sub-circuit consists of 4 digital ring oscillator sets and each of which is composed of 3 ring oscillators. Two of them are directly activated by internal control signals, which are obtained by bit-wise XORing a random byte from the PRNG and a data byte from the AES core. Data bytes from the AES core are incorporated to generate more random internal control signals because the PRNG always starts from a deterministic state after system reset. If data bytes from the AES core are incorporated, different internal control signals can be generated with different data blocks. Remaining ring oscillators are indirectly activated by the combination of internal control signals. Furthermore, a global enable signal controlled by the AES core can be used to turn off these ring oscillators to reduce the power consumption. The amount of additional power depends on the number of activated ring oscillators. As a result, overall power of the chip at every cycle can be randomly changed to make it independent of predicted power models.

To minimize area overhead of the DPA countermeasure circuit, the number of inversion stages of a ring oscillator should be as short as possible. Because ring oscillators must be initialized and controlled by internal and global control signals, at least three inversion stages

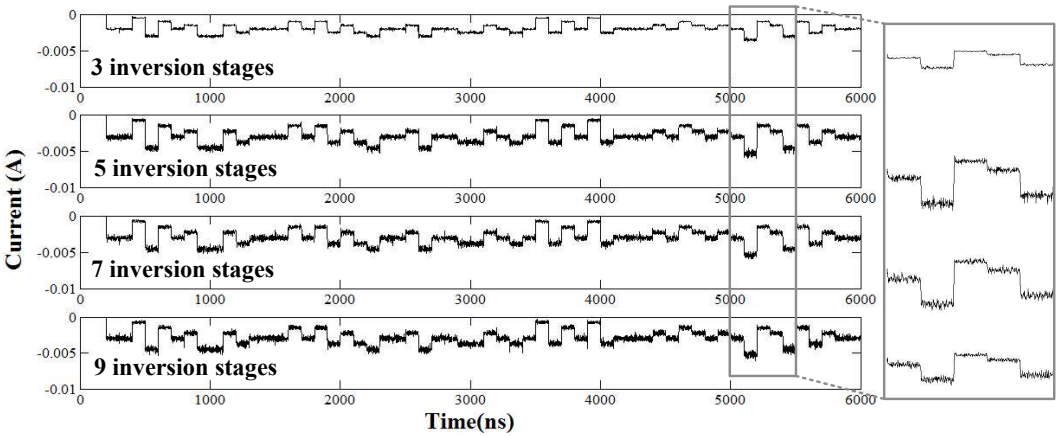


Figure 5.10: The power traces of ring oscillators with different inversion stages.

are required. As shown in Fig. 5.9, internal control signals and their combination are applied to one NAND gate and the global enable signal is applied to the other NAND gate. A ring oscillator will be activated only when the corresponding internal control signal is logic one. Furthermore, the DPA countermeasure circuit would be activated only when the AES core is working. Fig. 5.10 shows power signatures of ring oscillators with 3, 5, 7, and 9 inversion stages. All ring oscillators are controlled by the same data bytes and random bytes, and power signatures shown in Fig. 5.10 are quite similar except power values. A section of wave is zoomed in to show that relative changes of the current are maintained with different inversion stages. Therefore, 3 inversion stages are adopted in the DPA countermeasure circuit to minimize area overhead. Note that the pulse width would be too short that internal signals of ring oscillators may not reach full rail. However, since ring oscillators still consume additional power, the DPA resistance of our proposal can be increased even they cannot reach full rail.

The number of ring oscillators in each DPA countermeasure sub-circuit is another design consideration for area overhead. Since each sub-circuit is controlled by a data byte and a random byte, the most intuitive way is to adopt 8 ring oscillators in a sub-circuit. However, this architecture is still vulnerable to the DPA attack because random power consumption of the proposed DPA countermeasure circuit cannot dominate over total power consumption. From equation (4.2), it is obvious that the correlation coefficient would approach zero if

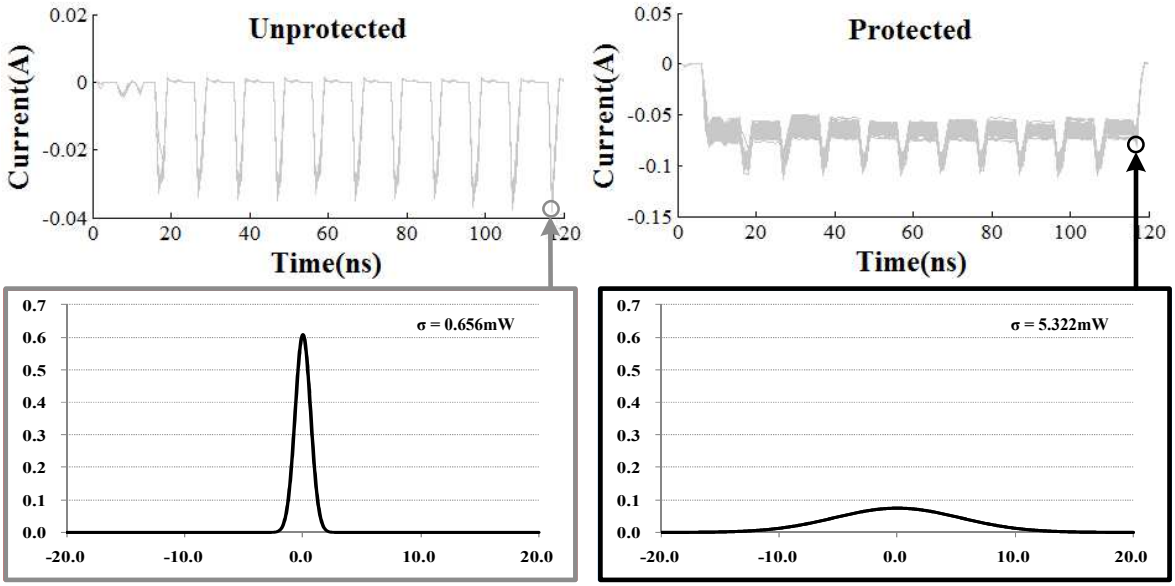


Figure 5.11: The power distribution of AES engines.

variables $h_{n,i}$ and $t_{n,j}$ are independent. Once power consumption of the DPA countermeasure can dominate, the standard deviation of $t_{n,j}$ will be increased. Therefore, the correlation coefficient of correct key hypothesis would also approach zero. From SPICE simulation in 90 nm CMOS technology, power consumption of a single ring oscillator is around 90 μW and that of one sub-bytes module, the most power consumptive component, is around 150 μW. As a result, at least two ring oscillators are activated at the same time in order to dominate over power consumption of one sub-bytes module. As shown in Fig. 5.9, 2 ring oscillators are activated even when Hamming weight of internal control signals is 1. For Hamming weight more than 1, more than 2 ring oscillators would be activated.

Fig. 5.11 shows simulated power traces of encryption operations with the same data block repeatedly applied to both the unprotected and protected AES core, respectively. Power traces of the unprotected and protected core are superimposed to show the effect of proposed DPA countermeasure circuit. As shown in this figure, power traces of the protected core have quite different characteristics even when the same data block is applied. The probability density function (PDF) at an interested time in the last round is also provided in Fig. 5.11. The PDF plot shows that power distribution of the unprotected core is quite centralized, indicating that the DPA attack can be used to disclose secret keys. On the contrary, PDF

plot of the protected AES core at the same time instance is also shown in Fig. 5.11 enclosed by black rectangular. The standard deviation is increased and the distribution becomes more flattened than that of the unprotected AES core. Because it is not easy to measure power consumption of individual modules of the protected AES core, the uncorrelated power of ring oscillators is used to increase the standard variation of total power consumption. Once the standard variation is increased, correlation coefficients can therefore be reduced according to equation (4.2) and therefore that of the correct key hypothesis would not lead to a significant peak. Hence, it is much difficult to disclose secret keys and the security level of the AES chip can be largely enhanced.

5.2.2 Dynamic Pseudo Random Number Generator

An on-chip PRNG is used to control the DPA countermeasure circuit. The most common method to implement a PRNG is based on linear feedback shift registers (LFSR), but characteristic polynomials can be broken easily due to regularity of the output sequence. A dynamic PRNG (DPRNG) [74] can change the feedback configuration of a LFSR to produce a more unpredictable output sequence. Block diagram of an 8-bit DPRNG is shown in Fig. 5.12. In order to generate 8 random bits at the same time, the DPRNG consists of 8 LFSRs based on [74] for each random bit. In addition, in order to further increase period of the 8-bit random byte, the length of each LFSR is chosen to be different. That is, characteristic polynomials with different degrees are used for each LFSR. With this architecture, period of the 8-bit sequence can be much higher than the 1-bit architecture.

The feedback configuration of each LFSR is dynamically selected from 4 characteristic polynomials by a 5-bit LFSR and a decoder. The purpose of the 5-bit LFSR and the decoder is to randomly configure the feedback network of each LFSR and make the output sequence meet the FIPS SP800-22 requirements [75]. To reduce the hardware cost, only polynomials with minimum terms are used as feedback configurations for each LFSR. With these minimum-term characteristic polynomials, the number of XOR gates for the feedback configuration is minimized. For example, characteristic polynomials used in the longest 17-bit

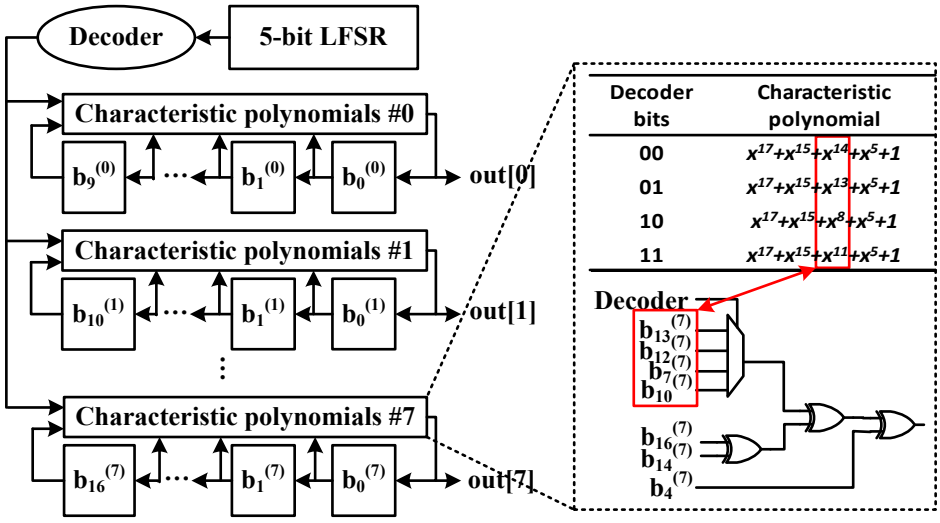


Figure 5.12: The block diagram of the on-chip dynamic PRNG.

LFSR is provided in Fig. 5.12. Since terms x^{17}, x^{15}, x^5, x^0 can be shared among these 4 characteristic polynomials, the feedback circuit requires only one additional 4-to-1 multiplexer instead of 4 combinational circuits for each configuration as also shown in Fig. 5.12.

5.2.3 Security Evaluation on SubBytes

In this sub-section, security analysis results of AES engines with LUT based and composite field based SubBytes are presented. The proposed DPA countermeasure circuit can be applied to different implementation architecture to provide the DPA resistance.

Fig. 5.13(a) shows the analysis result of a LUT based SubBytes implementation with proposed DPA countermeasure circuit. The correct key now does not result in the highest correlation and is now hidden in the analysis result. Therefore, even if attackers can find a peak in the analysis result, they still cannot find the correct key. As shown in Fig. 5.13(b), the correlation of the correct key is still lower than some other key hypotheses after one million traces are used for analysis. Besides, our proposed DPA countermeasure circuit can also efficiently counteract DPA attacks along with a composite field based SubBytes as shown in Fig. 5.14(a). The correlation of the correct key is always lower than some other key hypotheses, so the correct key can also be hidden under the protection of our DPA countermeasure circuit. Fig. 5.14(b) shows that the correct key again cannot be found even

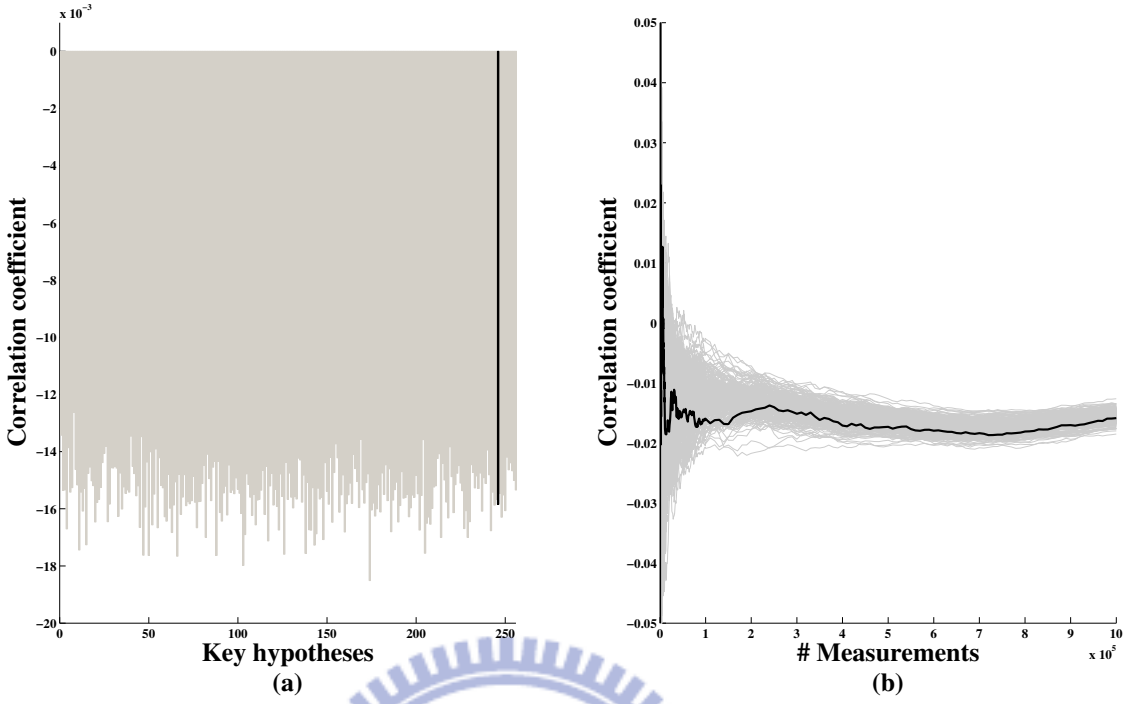


Figure 5.13: DPA results for LUT based SubBytes.

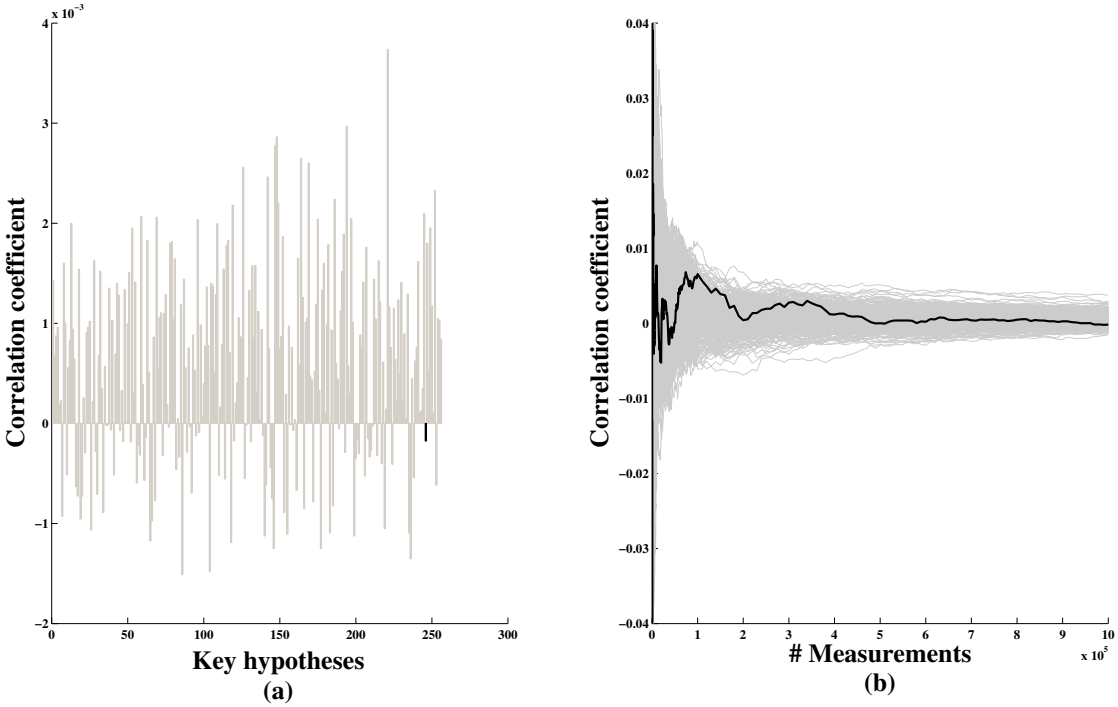


Figure 5.14: DPA results for composite field based SubBytes.

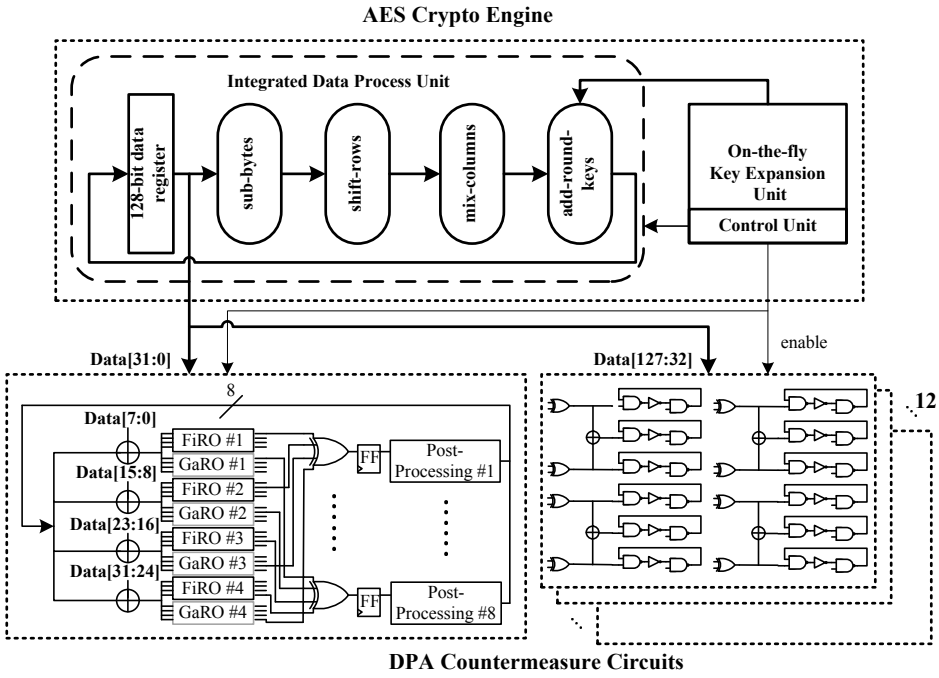


Figure 5.15: Block diagram of true random like based DPA-resistant AES engine.

one million power traces are used for analysis.

5.3 True Random Like Based DPA Countermeasure Circuit

Fig. 5.15 shows block diagram of the proposed true random like based DPA-resistant AES engine. The AES engine contains an integrated data process unit for data encryption/decryption and a on-the-fly key expansion unit for round key scheduling. The DPA countermeasure circuit is controlled by the AES engine when the data is processed in the data process unit. When the AES engine is in the idle state, the DPA countermeasure circuit is turned off to eliminate unnecessary power consumption. The 128-bit data path is connected to the DPA countermeasure circuit to dynamically enable different number of ring oscillators to resist the DPA attack. The least significant 32 bits are connected to the module with self generated random sequence and the remaining bits are connected to 12 ring oscillator based DPA countermeasure modules.

In this section, a serious issue of pseudo random based architecture is analyzed first. This

issue is a common problem if a pseudo random number generator is adopted. The improved architecture based on true random like number generator is presented to self generate a true random like sequence by reusing ring oscillators of the DPA countermeasure circuit.

5.3.1 Security Issue on PRNG

To demonstrate effect of the DPA countermeasure circuit, power traces recorded by SPICE simulation are illustrated in Fig. 5.16(a). This figure shows power traces of an unprotected AES circuit with the same input pattern but two different secret keys. The same input data is repeatedly encrypted for 100 times and those two secret keys differ only in the least significant bit. The distribution of the power consumption at a specific time instance is shown in Fig. 5.16(b). Standard deviations for both keys are relatively small and normal distributions are quite centralized. Since normal distributions of different secret keys can be easily distinguished, the DPA attack can use the statistical analysis to disclose secret keys with such kind of distribution.

On the contrary, power traces of the protected AES circuit with the same input pattern and secret key are shown in Fig. 5.17(a). Normal distributions of these two different secret keys are shown in Fig. 5.17(b). The standard deviation is largely increased compared with the unprotected AES circuit, leading to more flattened distributions. In this way, distributions of two different secret keys become very difficult to be distinguished and the DPA resistance can be largely increased.

However, there is still a security issue with this pseudo random based architecture. The random byte from the pseudo random number generator would be the same after the system is reset. Therefore, the additional power consumption added by the DPA countermeasure circuit in each cycle would be the same if the attacker resets the system before recording each power trace. Fig. 5.18 shows power traces and distributions if the system is reset before every operation. It is clear from the figure that the same amount of power consumption is added in each cycle. Power distributions with two different secret keys become easily distinguishable again, indicating that the DPA resistance is decreased.

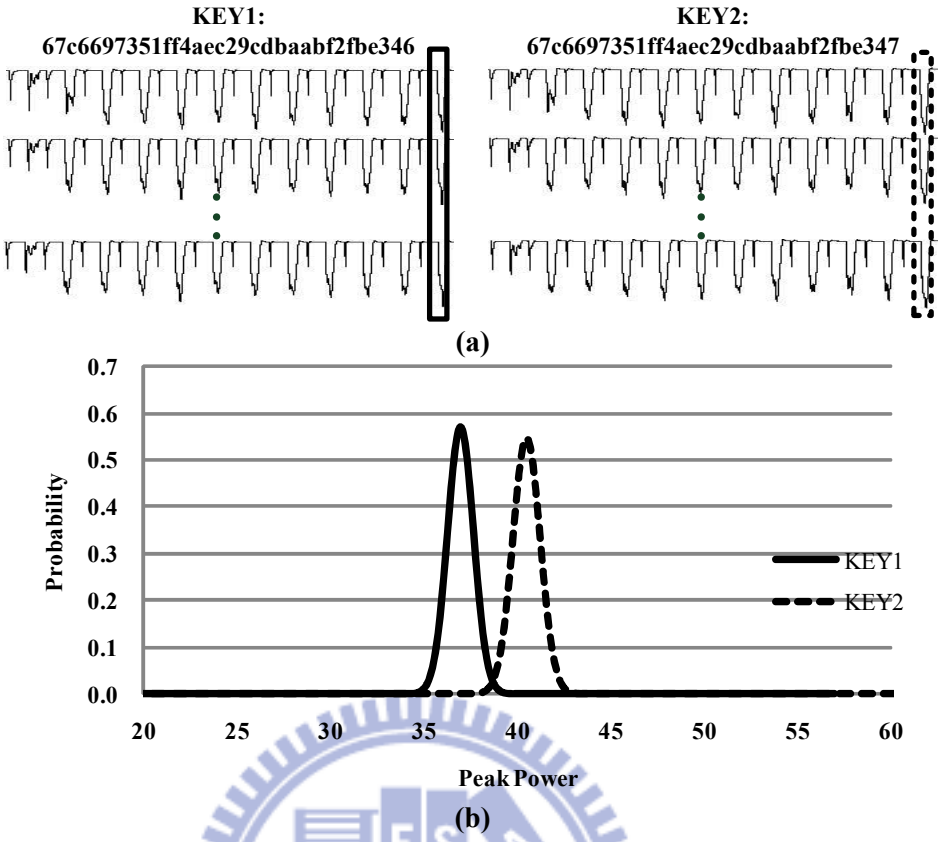


Figure 5.16: Power distributions of the unprotected AES engine.

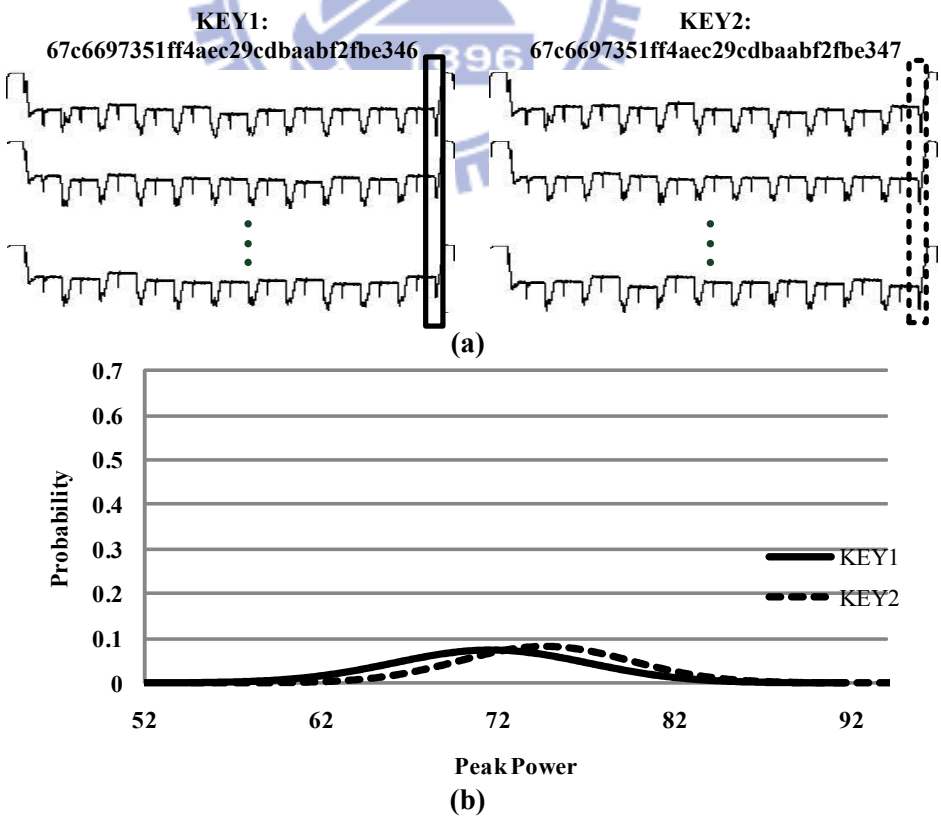


Figure 5.17: Power distributions of the pseudo random based architecture.

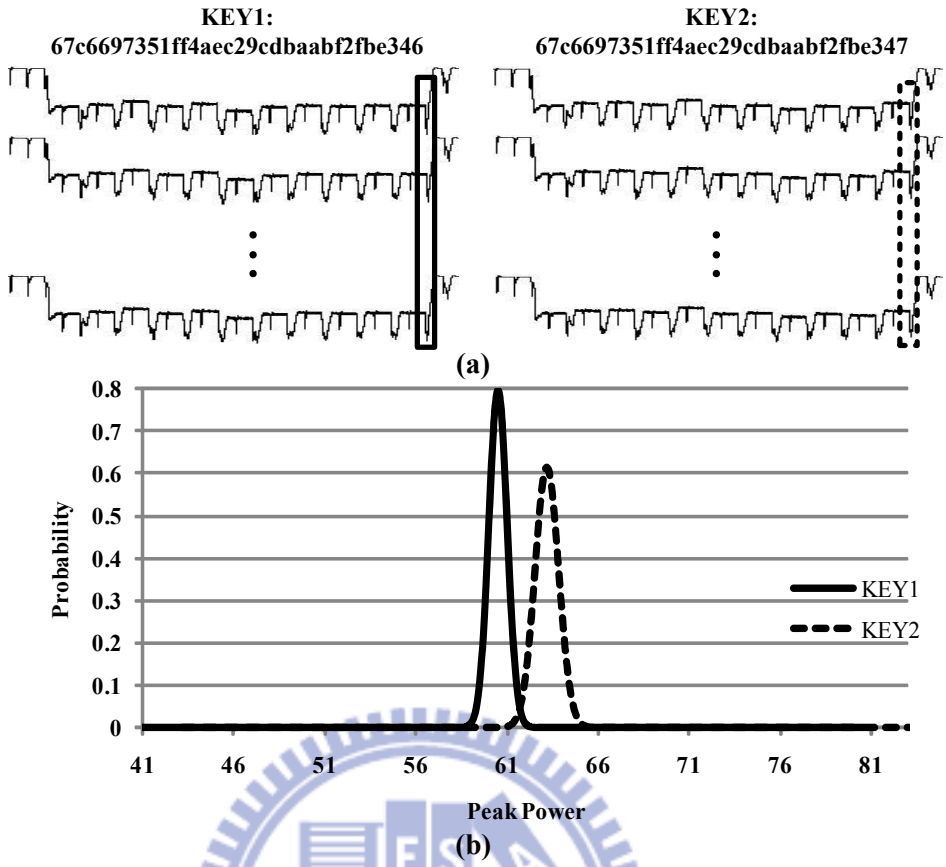


Figure 5.18: Power distributions of DPA-resistant AES engine with reset.

5.3.2 DPA Countermeasure with True Random Like Sequence

To solve the security issue in the pseudo random based architecture, a true random like sequence for the DPA countermeasure circuit is required. However, most true random like number generators are analog circuits with much higher power consumption. Golić proposed a digital method to generate random data by using ring oscillators in fibonacci or Galois configuration [76]. As shown in Fig. 5.19, fibonacci and Galois ring oscillator consist of a series of inverters connected with feedback polynomial $f(x) = \sum_{i=0}^{r} f_i x^i$, where $f_0 = f_r = 1$. The coefficient $f_i = 1$ indicates that the path is connected while $f_i = 0$ indicates no connection.

Instead of designing an extra true random like number generator, Fig. 5.20 shows the proposed DPA countermeasure circuit that can generate a true random like sequence of itself. Since the DPA countermeasure circuit is composed of several digital ring oscillators, these oscillators can be reused as random sources of the true random like number generator. Note

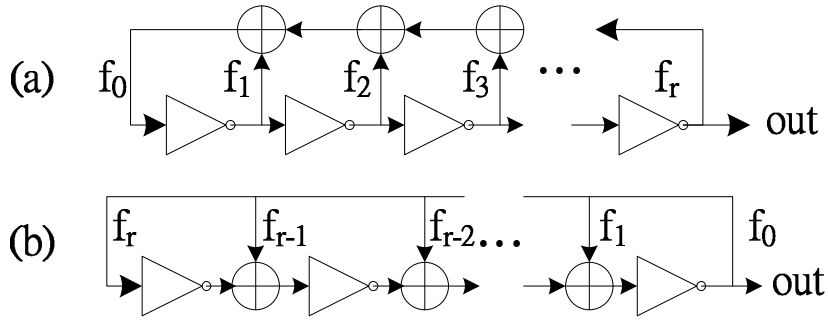


Figure 5.19: (a) Fibonacci ring oscillator (b) Galois ring oscillator.

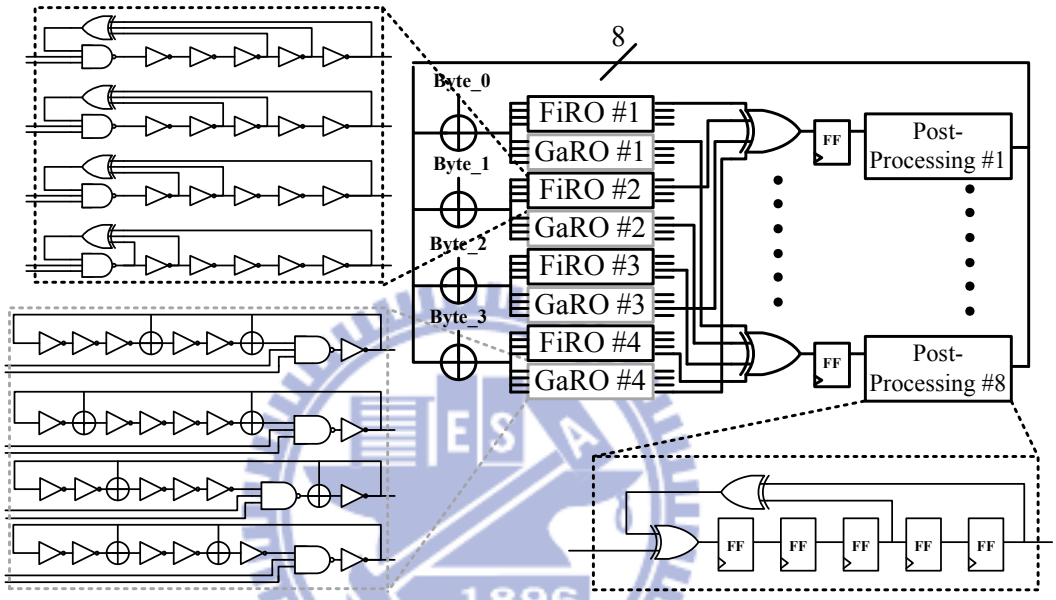


Figure 5.20: DPA countermeasure circuit with true random like sequence.

that the proposed DPA countermeasure circuit consists of 4 fibonacci ring oscillators sets (FiRO), 4 Galois ring oscillators sets (GaRO), and 8 post-processing circuits. The FiRO and GaRO are composed of 4 fibonacci and Galois ring oscillators, respectively.

The combination of 2 fibonacci ring oscillators and 2 Galois ring oscillators are used as the random source to generate one random sequence. These 4 ring oscillators are selected from the FiRO and GaRO randomly. In order to generate 8 independent random bits for each data byte, total 32 ring oscillators (including fibonacci and Galois ring oscillators) are required in the DPA countermeasure circuit. These 8 random sources are sampled by flip-flops for further post-processing. After post-processing, these 8 random bits are XORed with data bytes from the cryptographic circuit to dynamically enable oscillators in the FiRO and GaRO. The FiRO and GaRO now work not only as random sources in [76] to generate

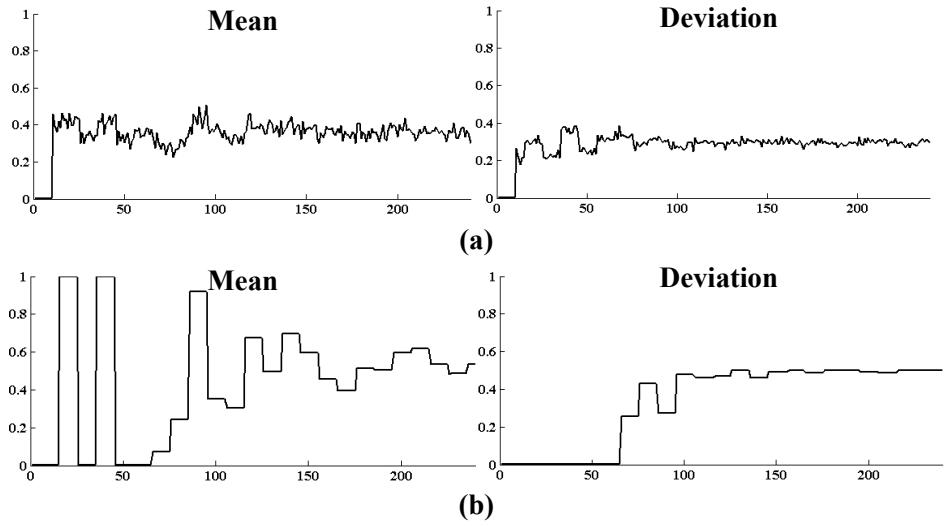


Figure 5.21: The randomness analysis of random sequence.

random data but also as the digitally controlled ring oscillators in the proposed DPA countermeasure circuit.

As discussed in [76], the fibonacci ring oscillator will not have a *fixed point* if and only if $f(x) = (1 + x)h(x)$ and $h(1) = 1$, where $f(x)$ is the polynomial presentation of the feedback configuration for fibonacci ring oscillator and $h(x)$ is a primitive polynomial. Note that a fixed point is a state that the output vector of inverters is an alternating string of 1 and 0 ($\{01010\cdots\}$ or $\{10101\cdots\}$). Since each random source is from the combination of four different ring oscillators, at least four different $h(x)$ are required. To have four different forms of $h(x)$, the minimum degree of $f(x)$ for the fibonacci ring oscillator is 6. Similarly, the condition for the Galois ring oscillator having no fixed point is $f(1) = 0$ and the degree of $f(x)$ must be odd [76]. Again, in order to have four different configurations, the minimum degree of $f(x)$ for Galois ring oscillators must be 7. The selected four fibonacci and Galois ring oscillators are shown in Fig. 5.20 for minimum hardware cost consideration.

The post-processing circuit is composed of LFSRs with different initial seeds. The purpose of the post-processing circuit is to remove bias of random sources. In each post-processing circuit, the feedback value is XORed with that from ring oscillators. In this way, even the post-processing circuit starts from a deterministic state after the system is reset, the generated random sequence would not be the same because the random source is incorpo-

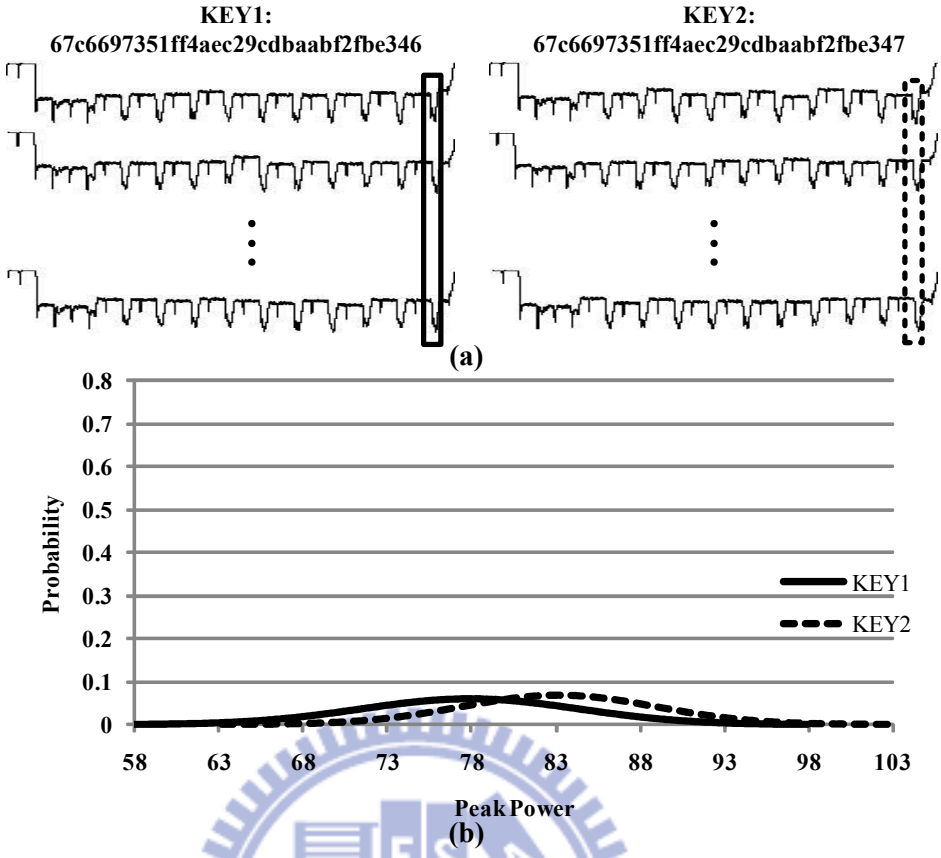


Figure 5.22: Power distributions of true random like based architecture.

rated into the feedback of LFSRs. Fig. 5.21(a) shows means and deviations of one random source for 100 system reset. Fig. 5.21(b) shows means and deviations of the random bits after post-processing circuit. The means show that the random sequence would be VDD/2 and standard deviations show that the generated sequence is true random like after a warm-up time. Although standard deviations are zero in the first few cycles, which means the generated bits in these cycles would be always the same after the system is reset. However, after the warm-up time, standard deviations would increase significantly and the random number generator would enter the true random like state.

To demonstrate the effect of the true random like based DPA countermeasure circuit, power traces of the AES engine with proposed DPA countermeasure circuit are shown in Fig. 5.22(a). The same conditions are used as previous analysis. Power traces now have different power consumption characteristics even the system reset is asserted before every operation. Normal distributions shown in Fig. 5.22(b) become flattened again due to the true

random like sequence. These two distributions can not be easily distinguished and therefore the DPA resistance of the true random like based architecture is increased even if the system is reset before every operation.

Chapter 6

Implementation Results and Comparison

This chapter presents chip implementation results of pseudo and true random like based DPA-resistant AES engines in UMC 90 nm CMOS technology. In addition, the DPA resistance of these two AES engines are also analyzed to demonstrate the effect of the proposed DPA countermeasure circuit. As last, the proposed DPA-resistant AES engines are compared with state-of-the-art designs in terms of area, throughput, power consumption, and DPA resistance.

6.1 Chip Implementation Results

Chip implementation results for the pseudo random based and true random like based architectures are provided in the following two sub-sections.

6.1.1 Pseudo Random Based Architecture

The summary of chip implementation results of the pseudo random based architecture is listed in TABLE 6.1. The first part provides the physical specification of the fabricated chip and the second part provides implementation results of the test chip.

The cell area for the unprotected AES core is 0.0885 mm^2 with 31.6 k equivalent 2-input NAND gates. The cell area for the DPA countermeasure circuit and the on-chip PRNG is

Table 6.1: Chip summary of pseudo random based architecture

	Unprotected	Protected
Technology	90 nm	
Core Voltage	1 V	
IO Voltage	2.25 V	
Package	48-Pin DIP	
Chip Area	1220×926 μm^2	
Core Area	944×650 μm^2	
Cell Area	0.0885 mm^2	0.0975 mm^2
Max Frequency	237 MHz	237 MHz
Max Throughput	2.76 Gbps	2.76 Gbps
Power (1V,237MHz)	39.5 mW	45.6 mW

0.009 mm^2. The area overhead of the DPA countermeasure circuit and the on-chip PRNG to a single AES core is 10.2%. However, when considering the I/O buffer, the area overhead of the DPA countermeasure circuit would be reduced to only 2.6%. The maximum measured operating frequency is 237 MHz and the throughput of AES-128 at this operating frequency is 2.76 Gbps. Since the DPA countermeasure circuit works in parallel with the AES core, the protected core can achieve zero throughput degradation. The power consumption of the unprotected AES core and the protected AES core is 39.5 mW and 45.6 mW, respectively. The power overhead of the DPA countermeasure circuit is 15.4%, indicating our design can be used in mobile devices requiring both low power consumption and high security. Fig. 6.1 shows the power breakdown for the unprotected AES operation. The AES core consumes 50.44% of total power and the embedded memory consumes almost 40% of total power. As mentioned earlier, the power consumption of our proposed 3-stage ring oscillator is 90 μW and each digital ring oscillator set is activated with probability 75%. That is, on average, 6 ring oscillators in a DPA countermeasure sub-circuit will be activated at each cycle. Therefore, a DPA countermeasure sub-circuit will consume 540 μW. Since the DPA countermeasure circuit consists of 16 identical sub-circuits, 8.64 mW additional power is required to protect the AES core and power overhead would be 43% without considering infrastructure for test chip fabrication. However, when power consumption of the embedded memory for data IO is included, power overhead is reduced to around 21.7%. Moreover, the DPA countermeasure can be disabled during loading phases of input and output data

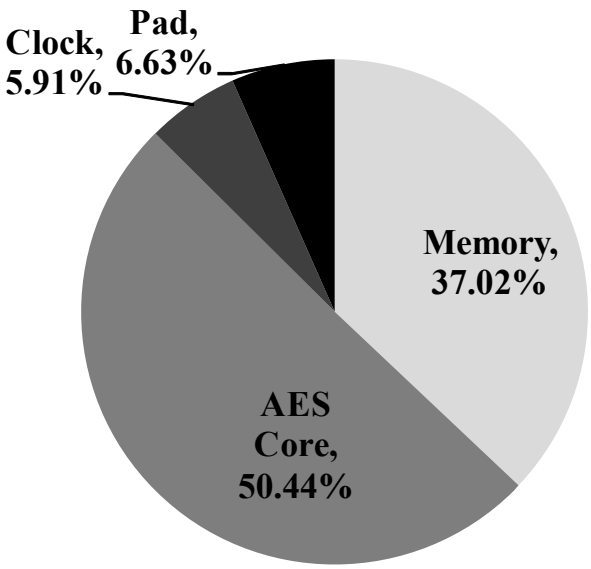


Figure 6.1: Power breakdown of the test chip.

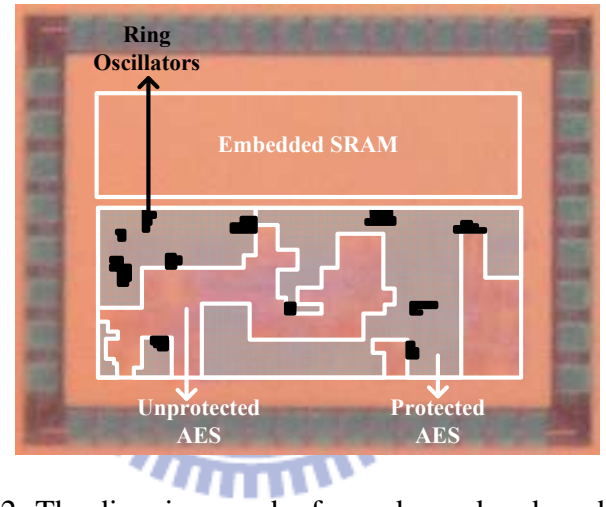


Figure 6.2: The die micrograph of pseudo random based test chip.

blocks. For this test chip, when the data loading phase is included, the power overhead can be reduced to about 15%. B

Fig. 6.2 shows the die micrograph of the test chip. The test chip contains a memory, an unprotected AES engine, and a protected AES engine as partitioned in this figure. Detailed location of the protected AES core is specified in this figure, where the location of ring oscillators is also specified by black blocks. Since placement is done by EDA tool automatically, it is not easy to identify the DPA countermeasure circuit from real silicon. As a result, the invasive alternation such as FIB to cut connections from ring oscillators or DPRNG to the AES core is hard to be achieved.

Table 6.2: Chip summary of true random like based architecture

	Unprotected	Protected
Technology	90 nm	
Core Voltage	1 V	
IO Voltage	2.25 V	
Package	40-Pin DIP	
Chip Area	787×787 μm^2	
Core Area	470×470 μm^2	
Cell Area	0.0979 mm^2	0.104 mm^2
Max Frequency	255 MHz	255 MHz
Max Throughput	2.97 Gbps	2.97 Gbps
Power (1V,255MHz)	7.68 mW	8.70 mW
Power (1V,100MHz)	3.35 mW	4.71 mW

6.1.2 True Random Like Based Architecture

The summary of chip implementation results of the true random like based architecture is listed in TABLE 6.2. The first part again provides physical specifications of the test chip and the second part provides implementation results of the chip.

The cell area for the AES engine is 0.0979 mm^2 and that of the DPA countermeasure circuit is 0.0061 mm^2, indicating the area overhead is around 6.2%. The area overhead is slightly reduced from 10.2% to 6.2% for the true random like based architecture because ring oscillators are reused to generate a random sequence and the PRNG can be eliminated. The maximum operating frequency for the AES engine is 255 MHz with throughput up to 2.97 Gb/s. Note that when the DPA countermeasure circuit is enabled to resist the DPA attack, the maximum operating frequency of the AES engine is not degraded, which means no throughput degradation when the DPA resistance is provided. The power consumption for the AES engine working under 255 MHz with the DPA countermeasure circuit enabled and disabled is 7.68 mW and 8.7 mW, respectively. If the operating frequency is decreased to 100 MHz, the power consumption is reduced to 3.35 mW and 4.71 mW, respectively.

Fig. 6.3 shows the die micrograph of the test chip. The test chip contains the proposed true random like based DPA-resistance AES engine as highlighted in this figure. Note that another irrelative design is also included for other purpose.

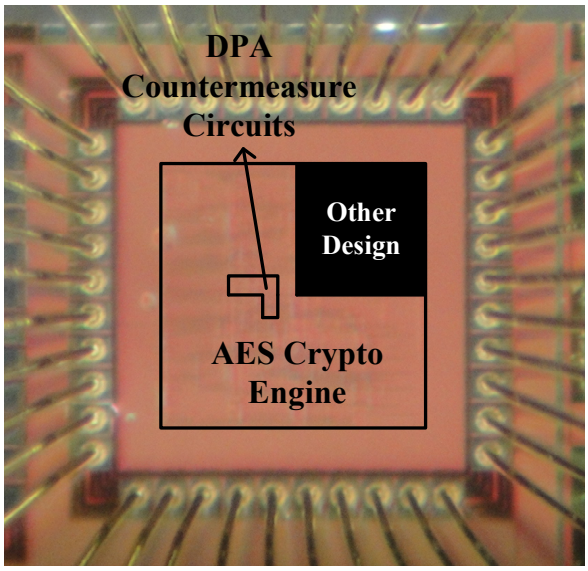


Figure 6.3: The die micrograph of true random like based test chip.

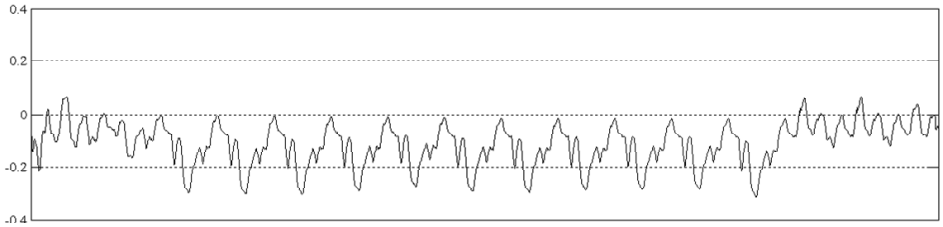
6.2 Security Analysis Results

The DPA attack is performed on both test chips to show the DPA resistance of the proposed DPA countermeasure circuit in this section.

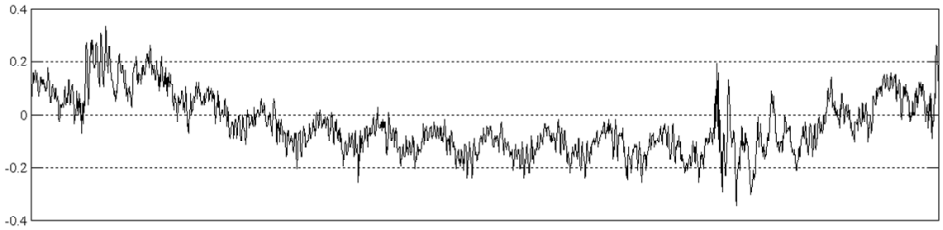
6.2.1 Pseudo Random Based Architecture

Power traces of the test chip for one encryption operation are shown in Fig. 6.4. Fig. 6.4(a) shows a measured power trace of the unprotected AES core for one encryption operation. 11 rounds of the AES encryption process can be easily distinguished and the peak current consumption is corresponding to round keys and data blocks. Fig. 6.4(b) shows the power trace of the protected AES core for the same encryption operation. Power consumption characteristics are now changed by the DPA countermeasure circuit.

DPA attack results for the least significant key byte are shown in Fig. 6.5. Fig. 6.5(a) shows that the correlation coefficient of the correct key hypothesis is still not distinguishable from other incorrect key hypotheses with 10^7 measurements. Compared with simulation-based DPA attack results, the correlation coefficient of the correct key hypothesis is reduced from -0.003 to -0.0003. If environment noises are considered, the measurement to disclosure (MTD) is increased by 38 times as discussed earlier. As a result, since the MTD of the



(a) Unprotected core



(b) Protected core

Figure 6.4: Power traces of the test chip for one encryption operation.

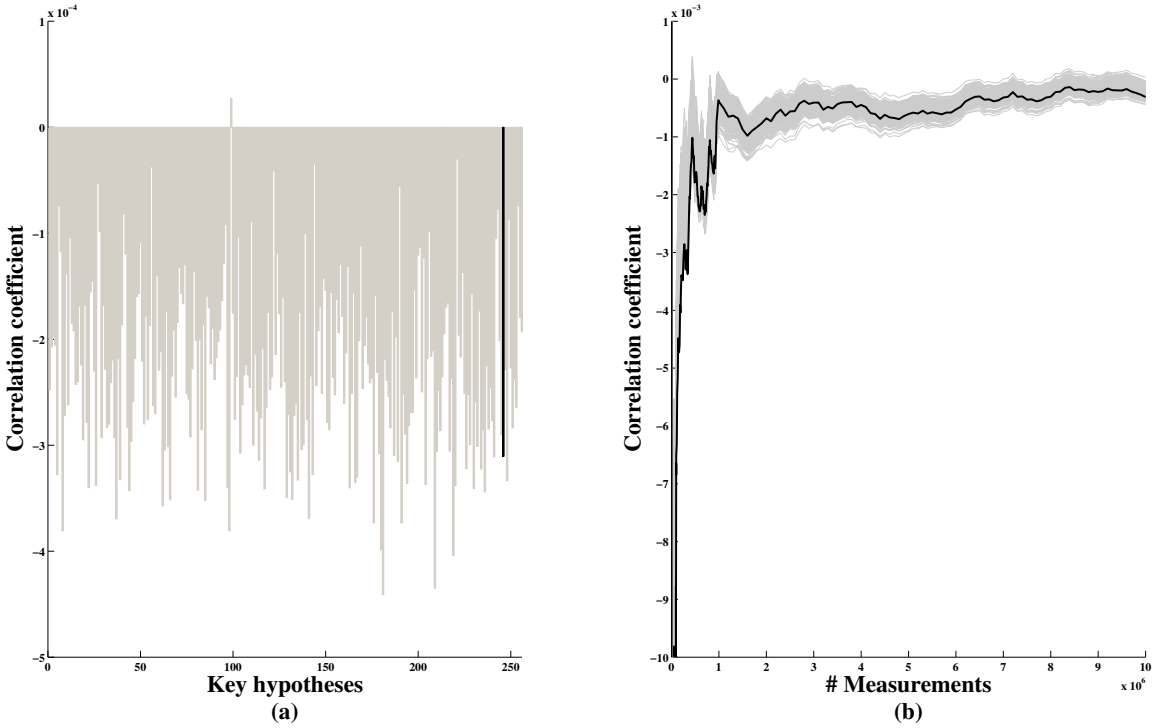
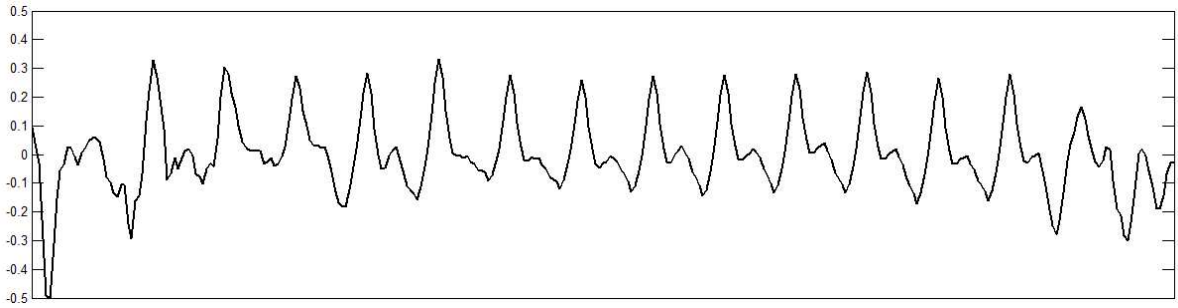
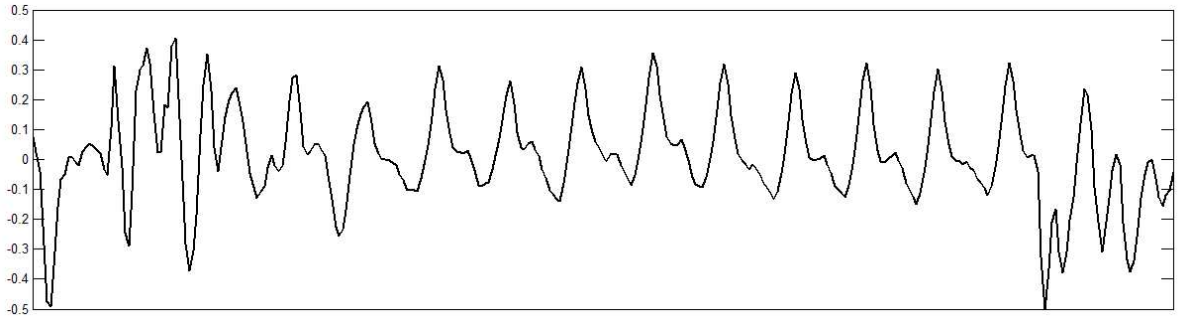


Figure 6.5: DPA results for pseudo random based test chip.



(a) Unprotected core



(b) Protected core

Figure 6.6: Power traces of the test chip for one encryption operation.

protected AES core is more than 10^6 measurements in simulation, the estimated MTD of the protected AES core could be more than 3.8×10^7 measurements when considering these non-ideal effects. However, from the DPA attack on the test chip, the MTD of the protected AES core is at least 10^7 measurements as shown in Fig. 6.5(b), which is sufficient to resist the DPA attack.

6.2.2 True Random Like Based Architecture

Power traces of the test chip for one encryption operation are shown in Fig. 6.6. Fig. 6.6(a) is the measured power trace of the unprotected AES core for one encryption operation. The power trace of the protected AES core is also given in Fig. 6.6(b) for comparison. The peak power consumption of every clock cycle is now changed by the DPA countermeasure circuit, which leads to the DPA resistance of the AES engine.

DPA attack results for the least significant key byte is shown in Fig. 6.7. Note that the

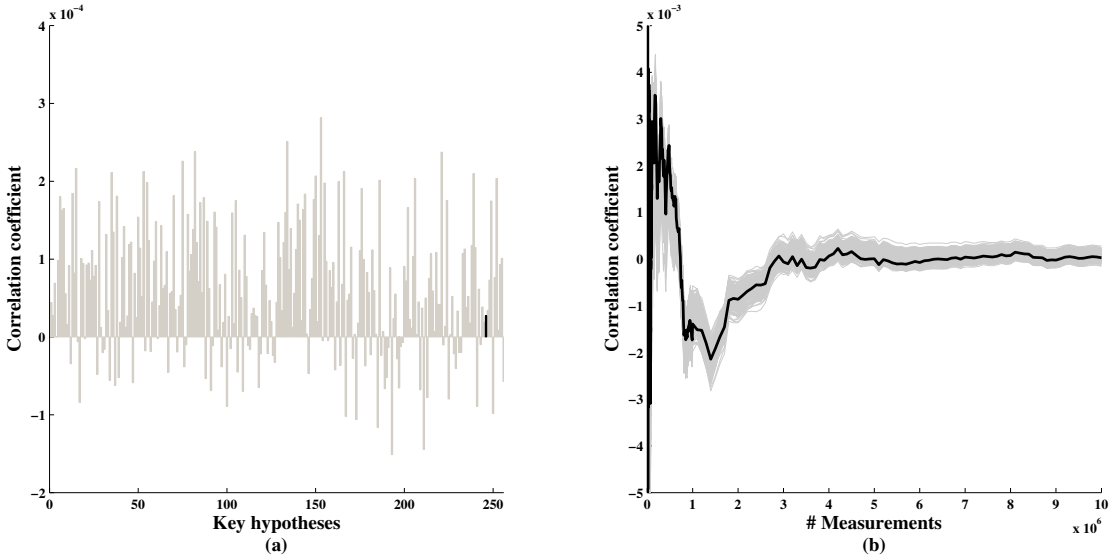


Figure 6.7: DPA results for true random like based test chip.

system reset is asserted before every encryption operation to demonstrate the effect of the true random like based architecture. Fig. 6.7(a) shows correlation coefficients of all key hypotheses with 10^7 measurement. As shown in this figure, the correlation coefficient of the correct key hypothesis can not be distinguished from other wrong key hypotheses. That is, the attacker can not disclose the correct key byte from the analysis result.

To illustrate the security in terms of MTD, correlation coefficients of all key hypotheses for different number of measurements are given in Fig. 6.7(b). In this figure, the correlation coefficient of the correct key byte is plotted in black while that of wrong key hypotheses are plotted in gray. This figure shows that the correct key hypothesis can not be disclose with less than 10^7 measurements. The security of the true random like based architecture now can provided the same security level even if system reset is asserted before every operation.

To further assess the number of required power traces to disclose the secret key, equation (4.3) and Table 4.1 is used to estimation the number of power traces to distinguish correlation value between 0.0003 and zero. Note that 0.0003 is used as $\rho_{ck,ct}$ obtained from the analysis result. With $\rho_{ck,ct}$ equals to 0.0003, the assessed MTD for the protected AES engine could be around 3×10^8 measurements.

Table 6.3: Comparison for hardware cost

Design	Technology	DPA	Core Area	Overhead$_A$
True Random Like Based	90 nm	N	0.0979 mm^2	-
		Y	0.104 mm^2	6.2%
Pseudo Random Based	90 nm	N	0.0885 mm^2	-
		Y	0.0975 mm^2	10.2%
WDDL [41]	0.18 μm	N	0.79 mm^2	-
		Y	2.45 mm^2	210.1%
Switching Capacitors [42]	0.13 μm	N	0.35 mm^2	-
		Y	0.44 mm^2	27.1%

6.3 Comparison

The comparison to state-of-the-art designs in terms of area overhead, throughput degradation, power overhead, and DPA resistance are presented in this section. Note that only silicon proven designs are listed for conciseness. Table 6.3 lists the comparison in terms of the hardware area. The WDDL cells are usually 2-3 times larger than the standard cells and therefore the protected AES engine proposed by Hwang *et al.* results in 210% area overhead. Switching capacitors proposed by Tokunaga and Blaauw can significantly reduce the area overhead to 27.1%. Our proposed architectures require only 10.2% and 6.2% more silicon area to resist the DPA attack. This factor indicates that the proposed DPA countermeasure circuit can be widely adopted in area limited applications such as smart cards or secure RFIDs.

Table 6.4 lists the comparison in terms of the throughput. The maximum operating frequency of different AES engines differ due to different architectures and technologies. It is more fair to compare the performance index in terms of throughput degradation. That is, how much will the DPA countermeasures affect the throughput. The WDDL cells result in longer cell delay and the maximum operating frequency is degraded by 74% when the DPA resistance is provided. For switching capacitors, the current consumed by the AES engine is now supplied by an array of capacitors. The maximum operating frequency is also degraded by 50% due to the operation of these capacitors. Since the proposed DPA countermeasure circuit works in parallel with the AES engine, no extra delay in the AES engine is induced when the DPA resistance is provided. Therefore, the maximum achievable throughput is not

Table 6.4: Comparison for throughput

Design	Technology	DPA	Frequency	Throughput	Overhead$_T$
True Random Like Based	90 nm	N	255 MHz	2.97 Gb/s	-
		Y	255 MHz	2.97 Gb/s	0%
Pseudo Random Based	90 nm	N	237 MHz	2.76 Gb/s	-
		Y	237 MHz	2.76 Gb/s	0%
WDDL [41]	0.18 μm	N	330 MHz	3.84 Gb/s	-
		Y	85.5 MHz	0.99 Gb/s	74.2%
Switching Capacitors [42]	0.13 μm	N	200 MHz	2.56 Gb/s	-
		Y	100 MHz	1.28 Gb/s	50%

Table 6.5: Comparison for power

Design	Technology	DPA	Power	Condition	Overhead$_P$
True Random Like Based	90 nm	N	7.68 mW	1V, 255MHz	-
		Y	8.70 mW	1V, 255MHz	13.3%
Pseudo Random Based	90 nm	N	39.5 mW	1V, 237MHz	-
		Y	45.6 mW	1V, 237MHz	15.4%
WDDL [41]	0.18 μm	N	54 mW	1.8V, 50MHz	-
		Y	200 mW	1.8V, 50MHz	270.4%
Switching Capacitors [42]	0.13 μm	N	33.32 mW	1.2V, 100MHz	-
		Y	44.34 mW	1.2V, 100MHz	33%

affected. This factor indicates that the proposed DPA countermeasure circuit can also be widely adopted in high throughput applications such as secure storage devices or high speed communications.

Table 6.5 shows the comparison in terms of the power consumption. Again, since the power consumption largely depends on architectures and technologies, the power consumption overhead is more important than absolute power consumption values. The WDDL method requires a lot of redundant gates to balance the power consumption of different transitions. Therefore, the resulting power overhead is more than 270%. For switching capacitors, extra logic to control the array of capacitors is required and therefore the power overhead is around 33%. Note that in [42] the power consumption of other circuits such as I/O memory is included. That is, the power overhead will be much higher if only the AES engine is considered. The proposed DPA countermeasure circuit can result in only 13.3% power overhead.

Table 6.6 shows the result that considers the area overhead, throughput degradation, and

Table 6.6: Comparison for figure of merit

Design	Overhead$_A$	Overhead$_T$	Overhead$_P$	Figure of merit
True Random Like Based	6.2%	0%	13.3%	1.20
Pseudo Random Based	10.2%	0%	15.4%	1.27
WDDL [41]	210.1%	74.2%	270.4%	20.01
Switching Capacitors [42]	27.1%	50%	33%	2.54

power overhead at the same time. The figure of merit is defined as $(Overhead_A + 100\%) \times (Overhead_T + 100\%) \times (Overhead_P + 100\%)$. The smaller value indicates less impact on the AES engine when the DPA resistance is provided. This table shows that our proposal outperforms other state-of-the-art designs.

Chapter 7

Conclusion and Future Works

In this chapter, conclusions on the implementation of different AES engines and DPA-resistant AES engines are given first. Then some possible research directions are discussed in the following section.

7.1 Conclusion

The AES algorithm has been widely adopted in different applications to provide data security. However, design considerations for different applications differ a lot. The AES algorithm can be implemented with different architectures to meet requirements for different applications. As a result, the implementation of AES engines are discussed first in this dissertation.

For applications that require throughput higher than 10 Gb/s, the AES can be implemented using loop unrolling and pipelining. The 10 Gb/s throughput requirement can be met by unrolling two rounds of the AES algorithm with a 2-stage pipelining architecture. The implementation result shows that 10 Gb/s throughput can be achieved with 0.152 mm^2 silicon area, or 54K equivalent 2-input NAND gates, in 90 nm CMOS technology. For applications such as 40 GbE, the throughput requirement can be met by a fully unrolled and pipelined architecture. The implementation result shows that 50 Gb/s throughput can be achieved with 0.451 mm^2 silicon area, or 160K equivalent 2-input NAND gates, in 90 nm

CMOS technology. For 100 GbE, the throughput requirement can be met by using two 50 Gb/s AES engines because it is not easy for cell based design to achieve 100 Gb/s in a single core.

For area limited applications such as smart cards or RFIDs, the throughput requirement is less important. As a result, data-path width of the AES engine can be reduced from 128-bit to 8-bit to largely reduce the hardware cost. Only 8 bits are processed in each cycle, and therefore the latency is increased from 11 cycles to 172 cycles. Moreover, only the encryption function of the AES with key length 128 bits is implemented to reduce hardware complexity. The data encryption and decryption can be achieved by modes of operation for block ciphers. The implementation result shows that the silicon area can be minimized to 7800 μm^2, or 2764 equivalent 2-input NAND gates, in 90 nm CMOS technology. The throughput can be up to 137 Mb/s while operating under frequency 185 MHz, which is quite sufficient for smart cards or RFIDs.

For applications that require both high throughput and low cost such as high speed mobile devices, an area efficient architecture is also presented. To achieve Gb/s throughput, 128-bit data-path is required. To reduce the hardware cost, no loop unrolling and pipelining architecture is adopted. The implementation result shows that 1.69 Gb/s throughput can be achieved with only 0.044 mm^2 silicon area, or 15.58K equivalent 2-input NAND gates, in 90 nm CMOS technology.

In addition to the architecture design of the AES algorithm, the security issue of cryptographic devices is also discussed in this dissertation. The power analysis, especially the DPA attack, is used to attack the proposed AES engine. The DPA attack result shows that only 9300 measurements are sufficient to disclose the secret key of an AES engine. Compared with the brute force method, which requires 2^{128} attempts to disclose the secret key, the efficiency is significant improved.

To resist the DPA attack, a DPA countermeasure circuit based on digitally controlled ring oscillators is presented. The statistical analysis of power traces for both pseudo and true random based architectures is discussed. Although the pseudo random based method has

the advantage of easy implementation, the DPA resistance is largely reduced if the system is reset before recording the power trace. Therefore, a true random based architecture utilizing ring oscillators is proposed to eliminate this security problem. As compared with the pseudo random based architecture, the hardware overhead is slightly improved since ring oscillators are shared by the DPA countermeasure circuit and the true random number generator. The area overhead for the true random based architecture is largely reduced to only 6.2% while no throughput degradation is induced. As a result, the proposed DPA countermeasure circuit can be widely applied all applications that require both high security and low cost or high throughput.

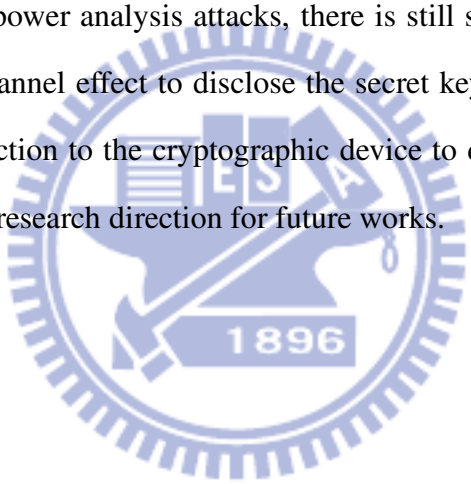
7.2 Future Works

There are several possible research directions for the power analysis or even side-channel analysis attacks. In this dissertation, the DPA is conducted based on single time instance, which is called as the first-order DPA attack. For the first-order DPA attack, attackers use the measured power traces to disclose secret keys directly. However, several literatures work on second-order or high-order DPA attack to disclose the secret key for the data-masking methods [77, 78]. In data masking methods, the random masks may be reused to save the required random sequence. As a result, by proper operations on power traces, such as averaging or subtraction, the power information of random masks could be removed. Therefore, instead of performing direct analysis on power traces, high-order DPA attacks analyzing the manipulated power traces. Since the proposed DPA countermeasure circuit is based on data-masking method, the high-order DPA attack on the DPA-resistant AES engine should be further analyzed.

Another possible research direction is to reduce the power overhead of the DPA-resistance AES engine for low power applications. Although the proposed DPA countermeasure can reduce the power consumption overhead to 18%, the power overhead is still too high for applications such wireless sensor networks. One possible solution is to utilize the concept

of dynamic voltage and frequency scaling (DVFS). For DVFS, the supply voltage can be changed by controlling power gating cells and the number of turned on cells is determined by a specific algorithm. Instead of this specific algorithm, the power gating celling can be turned on by a random sequence. The number of ones in this random sequence indicates the number of turned on power gating cells. As a result, the number of turned on cells would be different for every operation. In this way, the power consumption can also be dynamically changed for each operation. The power consumption would also reduce due to lower supply voltage and lower operating frequency. For low power applications, the throughput is usually not an important consideration and the DVFS based method may provide an excellent solution to reduce the power consumption.

In addition to the power analysis attacks, there is still some other side-channel attacks that utilize the side-channel effect to disclose the secret key. For instance, the fault attack relies on the fault injection to the cryptographic device to disclose the secret key [79–81]. This is also a possible research direction for future works.



Bibliography

[1] H. Wu, "The stream cipher hc-128," in *New Stream Cipher Designs*, ser. Lecture Notes in Computer Science, vol. 4986. Springer Berlin / Heidelberg, 2008, pp. 39–47.

[2] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: A new high-performance stream cipher." in *FSE*, 2003, pp. 307–329.

[3] D. Bernstein, "Salsa20," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/025, 2005, http://www.ecrypt.eu.org/stream.

[4] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, "Sosemanuk, a fast software-oriented stream cipher," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/027, 2005, http://www.ecrypt.eu.org/stream.

[5] M. Hell, T. Johansson, and W. Meier, "Grain - a stream cipher for constrained environments," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005, http://www.ecrypt.eu.org/stream.

[6] S. Babbage and M. Dodd, "The mickey stream ciphers," in *New Stream Cipher Designs*, ser. Lecture Notes in Computer Science, vol. 4986. Springer Berlin / Heidelberg, 2008, pp. 191–209.

[7] C. D. Canniére and B. Preneel, "Trivium - a stream cipher construction inspired by block cipher design principles," eSTREAM, ECRYPT Stream Cipher Project, Report 2005/010, 2005, http://www.ecrypt.eu.org/stream.

[8] *Federal Information Processing Standards Publication 197 - Advanced Encryption Standard*, National Institute of Standards and Technology, Nov. 2001.

[9] *Federal Information Processing Standards Publication 46-3 - Data Encryption Standard*, National Institute of Standards and Technology, Oct. 1999.

[10] *NIST Special Publication 800-67 - Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*, National Institute of Standards and Technology, May 2008.

[11] *Federal Information Processing Standards Publication 185 - Escrowed Encryption Standard (EES)*, National Institute of Standards and Technology, Feb. 1994.

[12] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *FSE*, 1993, pp. 191–204.

[13] E. Biham, R. J. Anderson, and L. R. Knudsen, "Serpent: A new block cipher proposal," in *FSE*, 1998, pp. 222–238.

[14] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644 – 654, Nov. 1976.

[15] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, February 1978.

[16] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. pp. 203–209, 1987.

[17] V. S. Miller, "Use of elliptic curves in cryptography," in *Lecture Notes in Computer Science; 218 on Advances in cryptology—CRYPTO 85*. Springer-Verlag New York, Inc., 1986, pp. 417–426.

[18] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proceedings of the* 16^{th} *Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 1996, pp. 104–113.

[19] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proceedings of the* 19^{th} *Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 1999, pp. 388–397.

[20] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with s-box optimization," in *Proceedings of Advances in Cryptography*, ser. LNCS, vol. 2248, 2001, pp. 239–254.

[21] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and performance testing of a 2.29-GB/s Rijndael processor," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 3, pp. 569–572, Mar. 2003.

[22] C.-P. Su, T.-F. Lin, C.-T. Huang, and C.-W. Wu, "A high-throughput low-cost AES processor," *IEEE Communications Magazine*, vol. 41, no. 12, pp. 86 – 91, Dec. 2003.

[23] K. U. Järvinen, M. T. Tommiska, and J. O. Skyttä, "A fully pipelined memoryless 17.8 gbps aes-128 encryptor," in *Proceedings of the 2003 ACM/SIGDA eleventh international symposium on Field programmable gate arrays*, ser. FPGA '03. ACM, 2003, pp. 207–215.

[24] G. Saggese, A. Mazzeo, N. Mazzocca, and A. Strollo, "An FPGA-based performance analysis of the unrolling, tiling, and pipelining of the AES algorithm," in *Field-Programmable Logic and Applications*, ser. Lecture Notes in Computer Science, vol. 2778. Springer Berlin / Heidelberg, 2003, pp. 292–302.

[25] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 9, pp. 957 – 967, Sept. 2004.

[26] F. K. Gürkaynak, A. Burg, N. Felber, W.Fichtner, D. Gasser, F. Hug, and H. Kaeslin, "A 2 Gb/s balanced AES crypto-chip implementation," in *Proceedings of the* 14^{th} *ACM Great Lakes symposium on VLSI*, Boston, MA, USA, Apr. 2004, pp. 39–44.

[27] S. Morioka and A. Satoh, "A 10-Gbps full-AES crypto design with a twisted BDD S-Box architecture," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 7, pp. 686 – 691, July 2004.

[28] T. A. Processor and A. Hodjat, "Speed-area trade-off for 10 to 100 Gbits/s throughput AES processor," in *37^{th} Asilomar Conference on Signals, Systems, and Computers*, 2003.

[29] A. Hodjat, D. D. Hwang, B. Lai, K. Tiri, and I. Verbauwhede, "A 3.84 Gbits/s AES crypto coprocessor with modes of operation in a 0.18-μm CMOS technology," in *Proceedings of the 15^{th} ACM Great Lakes symposium on VSLI*, Chicago, Illinois, USA, 2005, pp. 60–63.

[30] A. Hodjat and I. Verbauwhede, "Area-throughput trade-offs for fully pipelined 30 to 70 Gbits/s AES processors," *IEEE Transactions on Computers*, vol. 55, no. 4, pp. 366–372, Apr. 2006.

[31] M. Alam, S. Ray, D. Mukhopadhayay, S. Ghosh, D. RoyChowdhury, and I. Sengupta, "An area optimized reconfigurable encryptor for AES-Rijndael," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '07. San Jose, CA, USA: EDA Consortium, 2007, pp. 1116–1121.

[32] S. Y. Lin and C. T. Huang, "A high-throughput low-power AES cipher for network applications," in *Proceedings of Asia and South Pacific Design Automation Conference*, Jan. 2007, pp. 595–600.

[33] S. Mathew, F. Sheikh, A. Agarwal, M. Kounavis, S. Hsu, H. Kaul, M. Anders, and R. Krishnamurthy, "53Gbps native GF$(2^4)^2$ composite-field AES-encrypt/decrypt accelerator for content-protection in 45nm high-performance microprocessors," in *2010 IEEE Symposium on VLSI Circuits (VLSIC)*, June 2010, pp. 169 –170.

[34] S. Mangard, M. Aigner, and S. Dominikus, "A highly regular and scalable AES hardware architecture," *IEEE Transactions on Computers*, vol. 52, no. 4, pp. 483 – 491, April 2003.

[35] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "AES implementation on a grain of sand," in *IEE Proceedings of Information Security*, vol. 152, no. 1, Oct. 2005, pp. 13–20.

[36] N. Pramstaller, S. Mangard, S. Dominikus, and J. Wolkerstorfer, "Efficient aes implementations on ASICs and FPGAs," in *Advanced Encryption Standard - AES*, ser. Lecture Notes in Computer Science, vol. 3373. Springer Berlin / Heidelberg, 2005, pp. 571–571.

[37] P. Hämäläinen, T. Alho, M. Hännikäinen, and T. Hämäläinen, "Design and implementation of low-area and low-power AES encryption hardware core," in *9^{th} EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2006, pp. 577 –583.

[38] S.-F. Hsiao, M.-C. Chen, and C.-S. Tu, "Memory-free low-cost designs of advanced encryption standard using common subexpression elimination for subfunctions in transformations," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, no. 3, pp. 615 –626, March 2006.

[39] T. Good and M. Benaissa, "692-nW Advanced Encryption Standard (AES) on a 0.13-μm CMOS," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 12, pp. 1753 –1757, Dec. 2010.

[40] S. B. Örs, F. Gürkaynak, E. Oswald, and B. Preneel, "Power-analysis attack on an ASIC AES implementation," in *Proceedings of the International Conference on Information Technology: Coding and Computing*, vol. 2. Washington, DC, USA: IEEE Computer Society, 2004, p. 546.

[41] D. D. Hwang, K. Tiri, A. Hodjat, B.-C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis side-channel attacks," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 4, pp. 781–792, Apr. 2006.

[42] C. Tokunaga and D. Blaauw, "Securing encryption systems with a switched capacitor current equalizer," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 23 –31, Jan. 2010.

[43] J. Daemen and V. Rijmen, "The block cipher rijndael," in *Proceedings of the The International Conference on Smart Card Research and Applications*. London, UK: Springer-Verlag, 2000, pp. 277–284.

[44] A. Menezes, P. V. Ooschot, and S. Vanstone, *Handbook of Applied Cryptography*. New York: CRC Press, 1997.

[45] *NIST Special Publication 800-38A - Recommendation for Block Cipher Modes of Operation*, National Institute of Standards and Technology, Dec. 2001.

[46] *NIST Special Publication 800-38E - Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*, National Institute of Standards and Technology, Jan. 2010.

[47] *NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*, National Institute of Standards and Technology, May 2005.

[48] *NIST Special Publication 800-38C - Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*, National Institute of Standards and Technology, May 2008.

[49] *NIST Special Publication 800-38D - Recommendation for Block Cipher Modes of Operation Galois/Counter Mode (GCM) and GMAC*, National Institute of Standards and Technology, Dec. 2007.

[50] *IEEE Standard for Information technology - Telecommunication and information exchange between systems - Local and metropolitan networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Amendment 6: Medium Access Control (MAC) Security Enhancements*, IEEE Computer Society, 2004.

[51] *IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Security*, IEEE Computer Society, Aug. 2006.

[52] *FIBRE CHANNEL - SECURITY PROTOCOLS*, American National Standard for Information Technology, Feb. 2006.

[53] J. Wolkerstorfer, E. Oswald, and M. Lamberger, "An ASIC implementation of the AES SBoxes," in *Topics in Cryptology - CT-RSA 2002*, ser. Lecture Notes in Computer Science, vol. 2271. Springer Berlin / Heidelberg, 2002, pp. 29–52.

[54] D. Canright, "A Very Compact S-Box for AES," in *Proceedings of CHES 2005*, ser. LNCS, vol. 3659. Springer-Verlag, 2005, pp. 441–455.

[55] X. Zhang and K. Parhi, "On the optimum constructions of composite field for the AES algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 10, pp. 1153 –1157, Oct. 2006.

[56] H. Li and J. Li, "A new compact architecture for AES with optimized shiftrows operation," in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007.*, May 2007, pp. 1851 –1854.

[57] V. Rijmen, "Efficient implementation of the Rijndael s-box," 2001, available at http://www.esat.kuleuven.ac.be/~rijmen/rijndael/sbox.pdf.

[58] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer Science+Business Media, LLC, 2007.

[59] G. Hollestelle, W. Burgers, and J. den Hartog, "Power analysis on smartcard algorithms using simulation," Eindhoven, 2004.

[60] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Cryptographic Hardware and Embedded Systems - CHES*. Springer Berlin / Heidelberg, 2004, pp. 16–29.

[61] M.-L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," in *Cryptographic Hardware and Embedded Systems - CHES*. Springer Berlin / Heidelberg, 2001, pp. 309–318.

[62] E. Trichina, D. D. Seta, and L. Germani, "Simplified adaptive multiplicative masking for AES," in *Cryptographic Hardware and Embedded Systems - CHES*. Springer Berlin / Heidelberg, 2003, pp. 71–85.

[63] E. Trichina, "Combinational logic design for AES subbyte transformation on masked data," IACR report, Tech. Rep., 2003.

[64] J. Blömer, J. Guajardo, and V. Krummel, "Provably secure masking of AES," in *Selected Areas in Cryptography*. Springer Berlin / Heidelberg, 2005, pp. 69–83.

[65] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen, "A side-channel analysis resistant description of the AES s-box," in 12^{th} *International Workshop on Fast Software Encryption*. Springer Berlin / Heidelberg, 2005, pp. 413–423.

[66] D. Canright and L. Batina, "A very compact "perfectly masked" s-box for AES," in *Applied Cryptography and Network Security*. Springer Berlin / Heidelberg, 2008, pp. 446–459.

[67] D. Suzuki, M. Saeki, and T. Ichikawa, "Random switching logic: A countermeasure against DPA based on transition probability," on Transition Probability, IACR ePrint, Tech. Rep., 2004.

[68] 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* 28^{th} *European Solid-State Circuits Conference*, Sept. 2002, pp. 403–406.

[69] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, Feb. 2004, pp. 246–251.

[70] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Design and analysis of dual-rail circuits for security applications," *IEEE Transaction on Computer*, vol. 54, no. 4, pp. 449–460, 2005.

[71] N. Pramstaller, F. Gürkaynak, S. Haene, H. Kaeslin, N. Felber, and W. Fichtner, "Towards an AES crypto-chip resistant to differential power analysis," Sept. 2004, pp. 307–310.

[72] E. Trichina, T. Korkishkoand, and K. H. Lee, "Small size, low power, side channel-immune AES coprocessor: Design and synthesis results," in *Advanced Encryption Standard - AES*, ser. Lecture Notes in Computer Science, vol. 3373. Springer Berlin / Heidelberg, 2005, pp. 113–127.

[73] M. Saeki, D. Suzuki, K. Shimizu, and A. Satoh, "A design methodology for a DPA-resistant cryptographic LSI with RSL techniques," in *Cryptographic Hardware and Embedded Systems - CHES*, ser. Lecture Notes in Computer Science, vol. 5747. Springer Berlin / Heidelberg, 2009, pp. 189–204.

[74] R. Mita, G. Palumbo, S. Pennisi, and M. Poli, "A novel pseudo random bit generator for cryptography applications," in 9^{th} *International Conference on Electronics, Circuits and Systems*, vol. 2, 2002, pp. 489–492.

[75] *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, National Institute of Standards and Technology, Apr. 2010.

[76] J. D. Golić, "New methods for digital generation and postprocessing of random data," *IEEE Transactions on Computers*, vol. 55, pp. 1217–1229, Oct. 2006.

[77] M.-L. Akkar and L. Goubin, "A generic protection against high-order differential power analysis," in *Fast Software Encryption*, ser. Lecture Notes in Computer Science, vol. 2887, 2003, pp. 192–205.

[78] J. Waddle and D. Wagner, "Towards efficient second-order power analysis," in *Cryptographic Hardware and Embedded Systems - CHES 2004*, ser. Lecture Notes in Computer Science, vol. 3156. Springer Berlin / Heidelberg, 2004, pp. 1–15.

[79] G. Piret and J. jacques Quisquater, "A Differential Fault Attack Technique Against SPN Structures, with Application to the AES," in *AES and KHAZAD", CHES 2003, LNCS 2779*. Springer-Verlag, 2003, pp. 77–88.

[80] A. Moradi, M. Shalmani, and M. Salmasizadeh, “A Generalized Method of Differential Fault Attack Against AES Cryptosystem,” in *Cryptographic Hardware and Embedded Systems - CHES 2006*, ser. Lecture Notes in Computer Science, vol. 4249. Springer Berlin / Heidelberg, 2006, pp. 91–100.

[81] A. Barenghi, G. Bertoni, L. Breveglieri, M. Pellicioli, and G. Pelosi, “Fault attack on AES with single-bit induced faults,” in *Information Assurance and Security (IAS), 2010 Sixth International Conference on*, Aug. 2010, pp. 167 –172.