An Energy Efficient AES Engine with DPA-Resistance

by

Hye Won Chung

B.S. in Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology, 2007

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

Author ....
Department of Electrical Engineering and Computer Science
September 4, 2009

Certified by...
Anantha P. Chandrakasan
Joseph F. and Nancy P. Keithley Professor of Electrical Engineering
Thesis Supervisor

[illegible]

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

An Energy Efficient AES Engine with DPA-Resistance

by

Hye Won Chung

Submitted to the Department of Electrical Engineering and Computer Science
on September 4, 2009, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

The advent of portable electronics which transmit and receive sensitive data via wireless communication have led to interest in the design of energy-efficient security engines. The hardware implementation of cryptographic algorithms, however, leaks side-channel information about the operations they process. Differential Power Analysis (DPA) is one of the most powerful attacks to disclose secret key of the engine.

This thesis proposes an energy efficient AES-128 engine which is resilient to DPA attacks. A proposed design adopts extensive parallelism and voltage scaling to simultaneously achieve energy efficiency and throughput requirement. Optimized 128-bit architecture and 16 S-boxes placed in the encryption datapath allow the parallel operation of 16 bytes of encryption data at supply voltages in the subthreshold region while maintaining more than tens of $Mbps$ throughput rate. The energy efficient AES core, which does not incorporate techniques to mitigate DPA attack, can operate at $0.35V$ with $54.12pJ/encrypt.$ and $64.6Mbps$.

Before developing countermeasures against DPA, a previously suggested DPA attack methodology in [1] is studied and modified to disclose secret key of our system. The transition power of register is estimated by using the byte-oriented reverse algorithm of AES with the knowledge of a ciphertext and a guess of a secret key. Correlation between the power estimate and the power measurement discloses 12 key bytes (among 16) within $20K$ encryption runs.

A newly proposed AES architecture which balances the Hamming weight of register input can protect the system from DPA attacks. The secured core has been subjected to $100K$ encryptions, $33\times$ more than the number of runs at which attack can disclose a secret key of the unprotected core, but none of its secret key have yet been disclosed. When running the encryption core at $0.4V$ and $10MHz$, power increases by $2\times$ compared to the unprotected core. Maximum throughput at $1.0V$ is reduced by $2/3$ for protected core compared with the unprotected core. This IC performance overhead comes at the cost of the increased security.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Joseph F. and Nancy P. Keithley Professor of Electrical Engineering

Acknowledgments

First, I would like to thank my research advisor, Professor Anantha Chandrakasan, who led me into this exciting research project on security processor design. I really appreciate that he gave me much freedom to develop this project in an interesting direction, and always provided me invaluable insights and inspirations which guided into fruitful outcomes. Thanks to his patience and support, I have been able to enjoy my work and to learn the way of thinking and conducting research in this area.

The past two years at MIT was really great time since I could work with the brilliant and interactive Ananthagroup members. I would like to thank Joyce who has been a wonderful mentor since the beginning of my graduate year. She answered all my questions about CAD tools and design flows, and gave me invaluable comments on my work. Special thanks to Masood, Vivienne, Daniel, Patrick and Payam for all their help in the struggle to tame the CAD tools, to Bonnie and Yildiz for their wonderful jobs as social chairs (I will miss the homemade brownies!), to Marcus and Courtney for getting through the first two years together with helping each other, and to Margaret for making all the paper work run smoothly. Many thanks to all current and former members of Ananthagroup.

My life would have been less colorful without my wonderful friends. Shin Young, thanks for listening my daily rants, and sharing exciting experiences at our new hometown, in Boston. Jessie, thanks for being there for the good times and the bad. Hwajung, we've known each other for more than 10 years! I've been so fortunate to share my dreams and thoughts with such a good friend like you for a long time. Oshanis, thanks for constant reminders to exercise more. Thanks to my KAIST friends for always being on my side. I would also like to thank Korean EECS at MIT for all their helps and advices throughout my life here.

I am grateful to Pastor Dae Sung Choi and his wife Young Mi Choi for their pray and support.

Finally, I would like to thank my parents, Ki Gon Chung and Seung Eem Baek, and my sister, Hye Jin Chung for their constant support and love to me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Emerging mobile and biomedical devices, capable of connecting to the global network via wireless communication, are susceptible to tampering and eavesdropping because wireless network utilizes air itself as the transmission medium. Because of the security concerns, wireless mobile devices need to incorporate cryptographic primitives such as data encryption and user/message authentication. Since energy is highly limited in battery-operated devices, the cryptographic engine should be designed for minimum energy operation. Compared to software-based solutions, dedicated hardware implementations of cryptographic algorithms can be made very energy efficient, thereby making them suitable for energy-constrained applications [8].

However, security hardware is vulnerable to side-channel attacks. In cryptography, side-channel attack is any attack based on information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses in the algorithms. For example, power consumption, delay, or EM radiation can provide an extra source of information which can be exploited to break the system and to uncover secret key stored in the system. Differential Power Analysis (DPA) attack [3] is one of the most effective attacks because the switching power of digital CMOS gates provides attackers with important information about the secret key. There have been efforts to design a secure cryptographic core which defeats DPA attacks by masking the current signature of a system. The increased security is achieved at the cost of power, area, and/or performance overhead. For energy-constrained devices, however, design techniques for DPA-protection should satisfy not only the security specification but also stringent energy constraints. Therefore, most of the previously suggested DPA-protection techniques without specific concern for energy consumption overhead cannot be adopted for energy-limited applications.

The objective of this research is to design a secure 128-bit Advanced Encryption Standard (AES) engine which operates with utmost energy-efficiency while protecting secret key against DPA attack. There are several requirements that a battery-operated AES engine should satisfy. Before delineating the specifications and ideas for an energy-efficient AES engine, the encryption algorithm of AES will be studied first, and then the DPA attack methodology will be explored to understand how attackers can get the critical information about secret key from the power measurement.

1.1 Advanced Encryption Standard

In cryptography, Advanced Encryption Standard (AES) [2] is an encryption standard adopted by U.S. government. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. The AES is one of the most popular algorithms in symmetric key cryptography because it is relatively fast to compute and easy to implement, but still secure enough. As of 2006, the only successful attacks against AES implementation have been side-channel attacks.

1.1.1 Algorithm Specifications

The AES encryption algorithm is a block cipher that converts 128-bit plaintext to the same length of ciphertext by using cryptographic keys of 128, 192, or 256 bits. At the start of cipher, input is copied to the state array of 4 by 4 bytes as shown in Figure 1-1. After mathematical transformation of the data, the final value of the state is copied out as an encrypted data.

Figure 1-2 briefly describes the encryption process. After an initial round key addition, the state array is transformed by implementing a round function repeatedly for 10, 12, or 14 times depending on the key lengths of 128, 192, 256 bits respec-

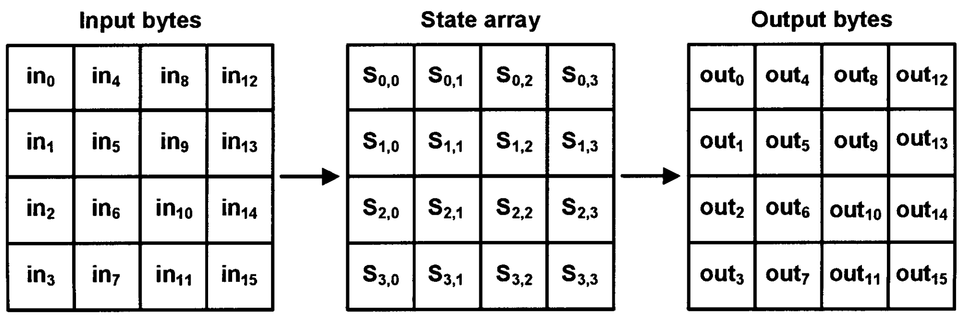


Figure 1-1: State array representation of input and output [2]

tively. All the rounds are identical with the exception of the final round, which does not include MixColumns operation as shown in Figure 1-2. A round function includes 4 sub-transformations : SubBytes, ShiftRows, MixColumns, and AddRoundKey. Round key is generated by a feedback operation of the RoundKeyGeneration block and added to the state array at AddRoundKey block.

The operations performed during a round transformation are described in the following.

- SubByte transformation is a non-linear byte substitution that operates independently on each byte of the State. The byte substitution (S-box) is constructed by composing two transformations, multiplicative inverse in the finite field $GF(2^8)$ and the following affine transformation over $GF(2)$.

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \tag{1.1}$$

where b_i and b_i' are the i^{th} bit of a byte before and after the affine transformation

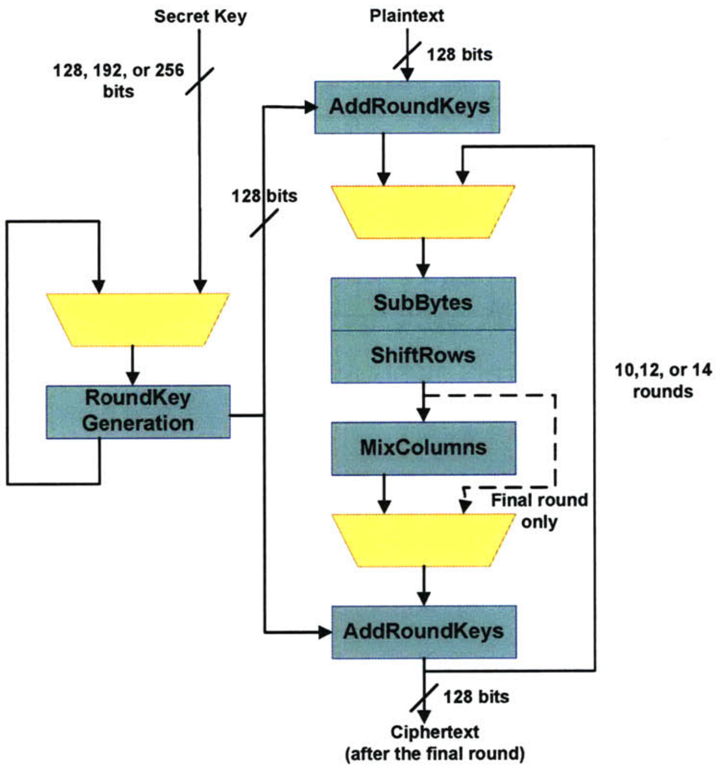


Figure 1-2: Block Diagram of the Advanced Encryption Standard (AES) algorithm

respectively.

- In the ShiftRow transformation, the bytes are cyclically shifted over different numbers of bytes depending on the row number. The first row is not shifted, and the second, third and forth row are cyclically shifted by 1, 2, and 3 bytes respectively.

- MixColumn transformation operates on the State column-by-column. Each column is multiplied modulo x^4+1 with a constant polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ over $GF(2^8)$ where $\{ab\}$ is the hexadecimal notation.

- In the AddRoundKey transformation, a Round Key is added to the State by a simple bitwise XOR operation.

After the final round operation, the State is copied to the output (ciphertext) as shown in Figure 1-1 and the encryption process is finished.

1.1.2 Area, Delay, and Power Characteristics of Implementations of S-box

Cryptographic substitution boxes (S-boxes) are an integral part of modern block ciphers. SubBytes transformation which is defined by S-box introduces non-linearity to the AES algorithm to resist linear cryptanalysis. The S-box is a costly and performance-critical building block of the AES algorithm. S-box lies on the critical path of the round-key scheduler as well as the round datapath so that it limits the maximum clock frequency. Moreover, the S-box also impacts area and power consumption of the AES hardware. There exists several ways to implement the S-box operation by using look-up table [9], calculating block [10], or decoder-permutation-encoder block [7]. The diverse implementations of S-box allow a wide range of trade-offs among timing, silicon area and power consumption, and make it possible to design AES engines suitable for different applications with varying specifications.

In [11], silicon area, critical path delay, and power consumption characteristics are analyzed and compared for the most common standard-cell designs of the AES S-box. Different implementations of AES S-box can be grouped into three basic categories: look-up implementations, calculating implementations, and decoder-permutation-encoder implementations. Look-up table is the simplest and the most straightforward way to implement the SubBytes transformation. It ignores the algebraic structure of the S-box which includes an inversion in GF(2^8) and the following affine transformation, but just maps the input-output relationship which is realized to a straightforward Boolean equation. Hence, the synthesizer has a degree of freedom for optimizing the circuit, which allows a short critical path at the expense of silicon area. In contrast to the look-up table implementation, calculating implementations use algebraic properties of the SubBytes algorithm. The inversion in GF(2^8) is realized by decomposing the finite field into the sub-fields of GF(2^4) and GF(2^2) to reduce the complexity of the inversion calculation. This implementation method allows a very area-efficient design which is suitable for area-constrained applications. However, in this design, it is hard to optimize the critical path compared to the LUT implementation. The last implementation method of S-box, decoder-permutation-encoder block, ignores the algebraic structure of the substitution and simply implements the Boolean equation for the input-output relation as like the LUT implementation. However, it uses a specific structure (decoder-permutation-encoder) and power-saving coding techniques to reduce the signal activity, and is therefore suitable for low power applications. From several synthesis results of the S-box designs in [11], we can find that the most power-efficient design for S-box is the decoder-permutation-encoder block suggested by Bertoni in [7].

Table 1.1 shows area and power trade-offs between 3 different S-box implementations which are synthesized for the same target delay of $5ns$ with the same $0.25\mu m$ process technology. Calculating block [10] is synthesized with the smallest size, but it consumes the largest power among the 3 implementations. Decoder-permutation-encoder [7] achieves the most power efficient implementation of S-box, which consumes only 15% and 27% power compared to other 2 designs, respectively.

Table 1.1: Area and power trade-off between 3 different S-box implementations [11].

	Calculating [10]	LUT [9]	Dec.-Perm.-Enc. [7]
Area(Gates)	496	1352	1399
Power(μW)	1.78	0.97	0.27

1.2 Differential Power Analysis

The hardware implementation of AES algorithm leaks information about the operations being processed. Therefore, even though the cryptographic algorithm is mathematically strong, secret key of the cryptographic engine can be discovered from the leaked information of the encryption process such as delay, power or electromagnetic emanations. Power analysis is one of the most common side-channel attacks because of its simplicity and effectiveness. Two kinds of power analysis, Simple Power Analysis (SPA) and Differential Power Analysis (DPA), are introduced by P. Kocher, J. Jaffe, and B. Jun in [3].

Simple Power Analysis (SPA) involves visual interpretation of power traces, or graphs of electrical activity over time. Power consumption varies depending on the different operations, thereby making it possible for attackers to distinguish which instructions are processed and sometimes even to find out the secret keys from which the order of operations are determined. However, avoiding procedures that use secret intermediates or keys for conditional branch will mask many SPA characteristics. Moreover, the magnitude of variations in power consumption is small so that SPA does not yield key materials for most (but not all) of the hard-wired hardware implementations.

Differential Power Analysis (DPA) is a more advanced form of power analysis which can allow attackers to compute the secret intermediate values by statistically analyzing power consumption measurements collected from the multiple cryptographic operations. When the cryptographic hardware performs operations using secret keys, the attack exploits varying power bias and statistically extract the secret key from measurements, which contain too much noise to be analyzed using SPA. Therefore, DPA is a much stronger attack compared to SPA. We will look further into the methodology of DPA and some countermeasures against DPA attacks.

1.2.1 DPA Attack Methodology

The DPA attack is conducted by monitoring the operations that leak information about secret keys in their power consumption. Attackers have access to ciphertext C (final version of the encrypted data) and guess a secret key K_S in the set of key space to calculate a secret intermediate value L of which Hamming weight is correlated to the power consumption. By finding out the value L which gives the highest correlation with the measured power consumption, the actual secret key K_S can be disclosed to attackers.

For AES-128 algorithm, the input value to the final round (10^{th} round) can be chosen as a secret intermediate value L since the value can be calculated in a reverse algorithmic way with the known ciphertext C and a guessed secret key K_S. Moreover, the value is highly correlated to the power consumption during the 10^{th} round operation. Therefore, once the real value of L is disclosed from correlation attacks, secret key K_S can also be known to the attackers.

The DPA selection function $D(C, b, K_{10})$ is defined as computing the b^{th} bit of intermediate value L where attackers observe the ciphertext C and guess a 10^{th} round key, K_{10}. The secret key K_S can be calculated in a reverse way when the 10^{th} round key K_{10} is found.

To implement the DPA attack, an attacker first observes m encryption operations and captures power traces $T_{1...m}[1 \dots k]$ containing k samples each. Attackers also records the ciphertexts $C_{1...m}$. DPA analysis uses power measurements and selection function $D(C, b, K_{10})$ to determine whether a key block guess K_{10} is correct or not. The following k-sample differential trace $\Delta_D[j], j = 1 \dots k$ calculates difference between the average of the power traces for which $D(C, b, K_{10})$ is one and the power traces for which $D(C, b, K_{10})$ is zero [3].

$$\Delta_D[j] = \frac{\sum_{i=1}^{m} D(C_i, b, K_{10}) T_i[j]}{\sum_{i=1}^{m} D(C_i, b, K_{10})} - \frac{\sum_{i=1}^{m} (1 - D(C_i, b, K_{10})) T_i[j]}{\sum_{i=1}^{m} (1 - D(C_i, b, K_{10}))} \tag{1.2}$$

, where $j = 1 \dots k$.

If K_{10} is an incorrect guess, the bit computed using D will differ from the actual target for about half of the ciphertexts C_i. The selection function $D(C, b, K_{10})$ is effectively uncorrelated to what was actually computed by the target device. Therefore, the difference between the power averages for which $D(C, b, K_{10})$ is 1 and that for which $D(C, b, K_{10})$ is 0 approaches to zero as the number of sample traces goes to infinity.

However, if K_{10} is a correct guess, the computed value for $D(C, b, K_{10})$ will equal to the actual value with probability 1. Therefore, the selection function is correlated to the value of the bit manipulated in the 10^{th} round and $\Delta_D[j], j = 1 \dots k$ function gives the spikes where the effect of the target bit appears on the power consumption.

Figure 1-3 shows the reference current trace (top) averaged over $m = 10^3$ encryptions and three differential traces $\Delta_D[j], j = 1 \dots k$ (bottom) for different key guesses. The first differential trace is produced using a correct secret key and the others are

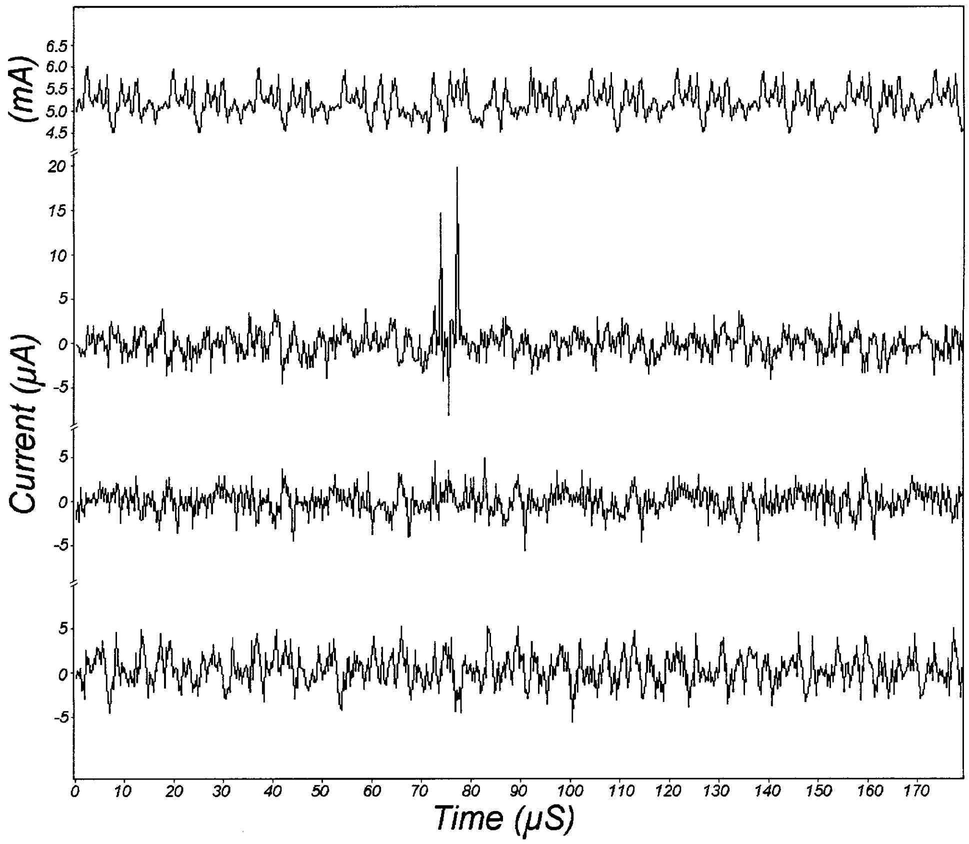


Figure 1-3: Current reference (at the top) and three DPA traces - the first trace with correct key guess and following two traces with incorrect guesses [3]

calculated using incorrect secret keys. We can clearly distinguish the correct key guess with spikes in the differential trace. Even though the differential traces are averaged over 10^3, there still exists a modest amount of noise as shown in the traces.

1.2.2 Previous Works to Defeat DPA Attacks

Techniques to prevent DPA have been suggested and demonstrated at different design levels including algorithm, circuit, logic gate, routing, and system. Even though the details of the ideas are all different, the techniques for DPA protection fall roughly into two categories.

The first approach uses randomization to introduce noise into power consumption measurements. In [12], current trace is scrambled with a randomized clock gating.

The AES engine suggested in [12] incorporates two functionally identical units, and one unit is designed to be more power-efficient than the other. With the randomized clock gating scheme, one and only one of the two units operates at each time. Therefore, the power trace is scrambled, and it makes attackers hard to extract enough information to discover the secret key from power measurements. In [13], Dynamic Voltage and Frequency Scaling (DVFS) scheduler with the previously programmed voltage-frequency pairs is incorporated to randomize the power consumption of a system. However, the value of supply voltage can easily be discovered to attackers by observing the operating frequency of the system. Hence, this technique cannot adequately protect the system from DPA attack. In [14], supply voltage is dynamically scaled with a fixed operating frequency to add some noise in power consumption traces. Even though this idea improves DPA immunity due to the added noise in power trace, none of the power and performance characteristics can be optimized in this system, which is undesirable especially for performance-critical or resource-constrained applications. Unfortunately, all the previously introduced noise-adding techniques moderately reduces the signal-to-noise ratio in power traces. Therefore, these techniques can increase the number of samples required for the disclosure of secret key, but after attackers gather large numbers of samples, the secret key can still be disclosed.

The second approach tries to reduce correlation between secret key and current trace observed at power supply by balancing Hamming Weights and state transitions or by shielding the core logic from power supply at certain period of time. In [1], signal activity is balanced by incorporating dual-rail logic and completely balanced interconnect with specific routing techniques. The method raises the number of measurements to disclosure significantly, but it incurs a 3x area, 4x power and 4x performance overhead. In [15], [16], and [17], the digital AES engine is isolated from the power supply with use of charge-pump circuits or a local switched-capacitor current equalizer. This approach also can make DPA attacks less feasible by increasing MTD by 2500$\times$, but it adds overhead in power consumption by 33%, and reduces maximum throughput by half [16]. It is because there should be periodical charging and discharging phases in this system. At discharging phase, the capacitor is discharged to a known voltage before it is recharged in order to equalize the amount of charge provided by the external power supply, which prevents attackers from getting information about the amount of charge supplied to the core logic.

DPA-protection techniques always incur trade-offs between security (DPA-immunity) and overhead in power, performance, and area. This thesis aims to design a secure AES engine that accommodates both DPA-protection and low energy consumption with wise trade-offs among important specifications of the system.

1.3 Thesis Contribution and Organization

This thesis focuses on developing an energy-efficient AES engine which is resilient to DPA attack. DPA performs a statistical analysis of supply-current measurements and samples of ciphertext to disclose the secret key which consists of 16 bytes. It is a very effective attack for AES algorithm since each byte of a key can be attacked independently, which makes DPA search space to be only 16×2^8. Therefore, the countermeasures of DPA attack should be adopted for secure storage or transmission of private data in portable electronics. Furthermore, to make AES engine suitable for energy-constrained applications, the stringent power budget should be taken into account along with other constraints as well as DPA-resistance.

This thesis proposes the DPA-immune design of an AES engine which maintains energy-efficiency. In Chapter 2, we outline the performance and power constraints of portable mobile and biomedical devices where AES engine will be integrated, and present architecture/logic design techniques to achieve AES engines operating down to power levels of microwatts. An energy efficient architecture will be described and a number of block-design optimizations will be performed to minimize energy consumption per encryption operation. DPA-immunity will not be considered at this phase of design. Simulation results will show that the proposed design meets performance and power constraints of encryption systems suitable for energy-constrained devices.

Chapter 3 will describe the methodology of DPA attacks. A model of the current consumption will be built for each possible key guess by using knowledge of the AES algorithm and ciphertext. The correlation between the model and the measured power will be calculated to disclose the secret key. The AES system designed in Chapter 2 will be attacked by the proposed DPA methodology and it will show how fast DPA analysis can discover the key from an AES engine which does not incorporate DPA-protection design.

Chapter 4 proposes a new design of an AES engine which defeats DPA in a power-efficient manner. DPA-resistant architecture and block optimizations will be described. The DPA attack will be conducted for this system and the results will be compared with those of the unprotected AES design. The area/performance overhead will also be discussed.

Chapter 5 describes the contribution of the thesis, and studies possible future research directions.

Chapter 2

Design of an Energy Efficient AES Engine

The major advantage of AES algorithm is its ability of efficient implementation on various platforms including 8-bit microprocessors, 32-bit processors, or dedicated ASIC hardware. Hardware implementations of AES algorithm focus on different optimization goals such as maximum throughput for networking implementations, power efficiency for radio-powered or energy-harvested devices, and area efficiency for resource-constrained applications like RFID tags. Most of the previous works stress throughput optimization with no energy budget constraints. However, for AES engine to be adopted for mobile or implanted devices, the design goal should be changed to achieve minimum energy operation of the system.

In this chapter, performance and power constraints of an AES engine will be analyzed to integrate the system within biomedical or mobile devices. We will explore the opportunities to optimize the system at the architecture and logic levels. In the course of architecture/block optimization, previous work in literature will be examined to study how the performance/power requirements are met and traded off with varying specifications. Then, an energy-efficient AES engine which incorporates maximum parallelism with voltage scaling will be proposed and described. IC performance results of the system from Nanosim simulation will be provided and compared with those of the state-of-the-art AES engines at the end of the chapter.

2.1 AES Engine Specifications for Energy-Constrained Applications

The efficiency of hardware implementations of AES algorithm allows the use of AES in diverse applications ranging from high-speed networks to passively-powered smart cards. According to the fields of applications, the design specifications and efficient implementations of AES engines can differ greatly. For networking applications, throughput rates in the Gigabit range should be achieved for the first priority. However, in passively-powered or battery-powered electronics, energy is highly limited so that the main design goal is to minimize energy consumption to achieve long enough operation with the same battery. In this work, we consider the design of an AES engine to be integrated in biomedical or mobile devices which are powered by battery or energy-harvester. Therefore, the stringent energy constraints dominate architectural and implementation decisions throughout the design of the AES engine.

Portable devices can receive energy from battery and/or energy harvester. For nonimplanted systems, batteries can be recharged or replaced, but, for the user's benefit, as infrequently as possible. If a portable biomedical device incorporates a low-power general purpose processor, which consumes approximately $10mW$, current battery technology would accommodate approximately 3 days of operation. Alternatively, dedicated solutions, employing specialized low-power design techniques, consumes only about $8\mu W$, achieving more than 10 years of operation with the same battery [18]. Therefore, dedicated low power solutions are indispensable for battery-operated portable electronics.

For implanted systems, however, batteries cannot be easily removed without surgery, and hence alternate wireless energy harvesting approaches must be considered. To supply energy to implanted devices, wireless electromagnetic energy transfer is an effective and commercially proven techniques. However, near-field wireless energy transfer still requires an explicit power transmitter. Alternatively, emerging approaches which exploit true ambient energy do not require any external source because they have the source of energy inherently as part of the system. For example, system which harvests vibrational energy from the user's movement can generate approximately $5\mu W$ on average [19]. The ultimate goal of our system is to operate by the energy derived from an energy-scavenging circuitry, which could deliver upto tens of microwatts. Thus, the power consumption of the AES engine should be constrained to be below tens of microwatts.

Required throughput range of AES implementations varies considerably depending on the applications. For optical networks, throughput range of tens of Gigabit per second (*Gbps*) should be achieved, while throughput range of tens to hundreds of Megabit per second (*Mbps*) is sufficient for secure data transmission at mobile devices. For sensor networks or implanted biomedical devices, an even lower throughput range can support real-time signal transmission since the period of signal sensing is on the order of few seconds. Therefore, the required throughput range of an AES engine is maximally hundreds of Mbps.

From these specifications, it is evident that new methodologies should be found for the design of an energy efficient AES engine with the system-level throughput constraints which are modest as they may be, but which must be met. Fortunately, scaling of device feature sizes, along with the development of high-density and low-parasitic packaging, make it possible to use the increased capability of CMOS gates to achieve the stringent specifications of low-power digital systems [20]. The following sections will suggest low-power techniques and architectural modifications to achieve an energy efficient AES engine with the acceptable throughput range.

2.2 Minimizing Energy Consumption

Low power operation of a system has become a major optimization goal of today's VLSI design. However, low power techniques do not necessarily result in a reduced energy consumption per operation. For example, a serial implementation of a processor can decrease the mean power consumption, but energy consumption is not reduced from this technique. Furthermore, if the serialized system runs at the maximum supply voltage to meet the throughput requirement, it degrades energy-efficiency compared to the parallelized system running at lower supply voltages. Therefore, the difference between power consumption and energy consumption should be considered in adopting adequate techniques for energy-efficient system design.

This section will analyze the power and energy consumption of a digital system and show the existance of minimum energy point. Also, we will discuss how parallelism, combined with voltage scaling, can simultaneously achieve performance requirement and energy efficient operation.

2.2.1 Minimum Energy Point

The dominant source of power in digital circuits is consumed during logic transitions when charge must be transferred from the supply voltage, V_{DD}, to the physical capacitance of a signal carrying circuit node, C_L. The power is called dynamic power, P_{DYN}, and is given by $\alpha C_L {V_{DD}}^2 f_{clk}$, where α is the activity factor and f_{clk} is the operating frequency. For low power operation, each factor in this equation should be minimized. Architectural and algorithmic techniques have been developed and widely used to reduce switching power by minimizing switching factor, α. For example, balancing logic delays from timing-path inputs can avoid glitching and reduce the activity factor. The attempt to reduce C_L by placing less gates on the die also results in the reduced power consumption. However, it might increase the number of clock cycles needed for the completion of operation. Therefore, in the respect of energy consumption, it might not be beneficial. Lowering supply voltage, V_{DD}, reduces power consumption significantly because the power consumption is a quadratic function of V_{DD}. Operating frequency, f_{clk}, can be decreased for reduction of power, but it directly deteriorates the throughput of the system. Therefore, trade-off between power and throughput should be considered and the design methodologies to accommodate both specifications have to be considered.

The total energy consumption of AES engine is broken down into dynamic energy and leakage energy. Dynamic energy is modeled as

$$E_{DYN} = \alpha N C_L {V_{DD}}^2 \tag{2.1}$$

where α is the activity factor, N is the number of clock cycles, C_L is the switched capacitance of the circuit, and V_{DD} is the supply voltage.

The leakage energy, $E_{leakage}$, is given by

$$\begin{aligned} E_L &= V_{DD} I_{LEAK} T_D \\ &= V_{DD} I_S e^{\frac{-V_{th}}{nV_T}} (1 - e^{\frac{-V_{ds}}{V_T}}) T_D \end{aligned} \tag{2.2}$$

where I_S is a technology-dependent scaling parameter, V_{ds} is the drain-to-source voltage, V_{th} is the threshold voltage, V_T is the thermal voltage, n is related to the subthreshold slope, and T_D is the latency of computation [21].

Therefore, the total energy per operation can be modeled by

$$\begin{aligned} E_T &= E_{DYN} + E_L \\ &= \alpha N C_L {V_{DD}}^2 + V_{DD} I_S e^{\frac{-V_{th}}{nV_T}} (1 - e^{\frac{-V_{ds}}{V_T}}) T_D \end{aligned} \tag{2.3}$$

As shown in Equation 2.3, reducing the supply voltage yields significant energy savings. However, lowering V_{DD} increases the latency of computation, T_D, linearly in the saturation region, and exponentially in the subthreshold region because it lowers output currents available to switch the circuit node capacitances [4]. Since leakage energy, E_L, is proportional to T_D, lowering V_{DD} starts to increase the leakage energy after the effect of T_D overwhelms the reduced V_{DD} in Equation 2.2. Since the dynamic energy and the leakage energy scale in an opposite manner with V_{DD} in subthreshold region, there exists a minimum operation point where the energy consumption of a system is minimized.

B. Calhoun solved equations for total energy to provide an analytical solution for the optimum supply voltage, and compared it with the simulation results of FIR filter in [4]. Figure 2-1 shows the existance of minimum energy point for the FIR filter in the subthreshold region. The exponential increase of latency, T_D, makes leakage energy increase exponentially in subthreshold region. The opposing trend of E_{DYN} and $E_{leakage}$ results in the minimum energy point.

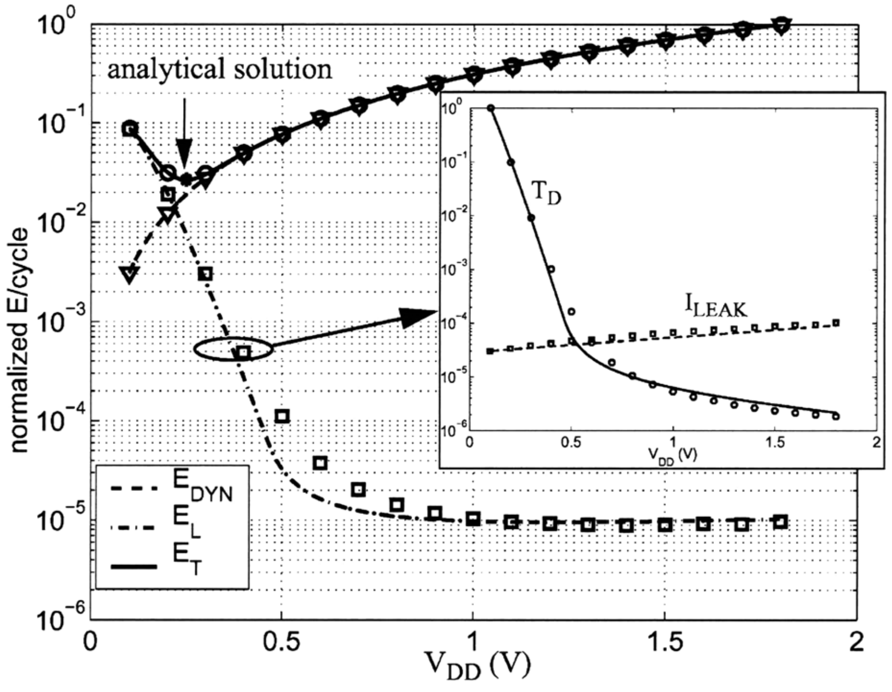


Figure 2-1: Model (lines) versus simulation (makers) of FIR filter show the minimum energy operation point and the contribution from dynamic and leakage energy. Inset shows I_{leak} and T_D effect on E_L [4].

2.2.2 Parallelism and Subthreshold Operation

The previous analysis of energy consumption shows that reducing the supply voltage yields significant energy savings but comes at the cost of reduced performance. In energy-stringent applications such as biomedical devices, the reduced performance that comes with voltage scaling is often acceptable because the accompanying energy savings are so stringent [22]. However, in some real-time applications, system-level throughput constraints, modest as they may be, must be met. For an AES engine, tens to hundreds of *Mbps* throughput rate should be achieved for data encryption and transmission. In such cases, parallelism, combined with voltage scaling, can be employed to simultaneously achieve the required performance and energy efficiency [20].

The AES algorithm processes 128-bit (16 bytes) of data to get the same length of ciphertext. Since the minimum unit for calculation of AES algorithm is a byte, 8-bit, 32-bit, and 128-bit platforms are all suitable for hardware implementation of AES. S-box is a non-linear byte substitution block that operates independently on each byte of the State. Depending on the bus size of datapath, the number of S-boxes placed in the datapath can range from 1 for 8-bit processor to 16 for 128-bit processor. The number of S-boxes placed in a system affects performance, area, and power consumption of the AES engine significantly since it has the longest delay within the critical path, and it is the largest and power-dominant building block of AES hardware. If maximally 16 S-boxes are located in an AES engine, then the substitution operation is conducted for all the 16 bytes concurrently, and each S-box can operate at a frequency reduced by a factor of 16, while maintaining the almost same overall performance as the system which has only one S-box to process 16-bytes of State. The reduced frequency per block enables voltage scaling and improved energy efficiency.

To see delay and energy consumption characteristics of a S-box with respect to supply voltages, a low-power S-box design suggested in [7] was implemented in a 90-nm CMOS process with low threshold devices. Nanosim simulations were performed to determine the maximum frequency and energy per operation at various supply voltages. Figure 2-2 shows the results of these simulations for a S-box. The red curve shows the normalized delay of a S-box and the black curve shows the normalized energy consumption of a S-box with respect to supply voltage. Delay of a S-box increases exponentially as CMOS starts to operate in the subthreshold region. However, the effect of leakage energy on total energy consumption is not significant for this block because a S-box consists of only 839 gates, and thus the increase of total energy in deep-subthreshold region as in Figure 2-1 does not occur here. Therefore, for a S-box, minimum energy point coincides with the minimum operating voltage at which the system can still function well.

From the simulation results of a S-box, it was known that operating at the minimum energy point of 0.35V rather than at the maximum V_{DD} of 1V reduces the energy per operation by 91.15%.

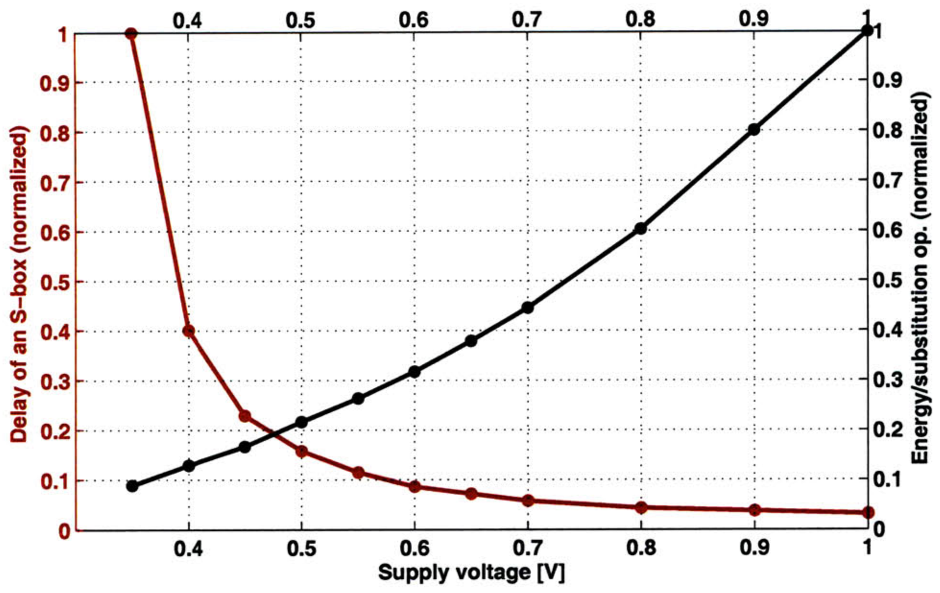


Figure 2-2: Normalized delay (red line) and energy (black line) of a S-box with respect to supply voltage.

To run a S-box at low supply voltage in an energy efficient way while achieving hundreds of *Mbps* throughput requirement of AES engine, S-box should be parallelized in an AES engine. From the synthesized results, we estimated that the delay of S-box is about 46.25% of the critical path delay in encryption datpath of AES engine. Since the synthesized result does not include any parasitics which might increase the delay of critical path, we assumed that the delay of S-box occupies about one third of critical path delay. Table 2.1 shows energy consumption for substitution operation of 16 bytes with respect to the degree of parallelism which is represented as the number of S-boxes in system, N. For efficient use of hardware, it is desirable that the number of S-boxes (N) placed in an AES engine be a multiple of a word size, 4bytes, i.e., 4,8 or 16. The throughput requirement of AES engine is assumed to be $300Mbps$, and the supply voltage is set to satisfy the throughput requirement with the given numbers of S-boxes in a system. From the energy consumption comparison in Table 2.1, it is shown that 16 S-boxes are needed to simultaneously achieve $300Mbps$ throughput and the minimum energy consumption requirements. This analysis will be used in the architectural optimization of AES engine which will be discussed in the next section.

Table 2.1: Comparison of energy consumption for different degrees of parallelism all of which satisfy the same throughput requirement.

N	Throughput [Mbps)]	Delay of a S-box [ns]	V_{DD} [V]	Energy/16 substitutions [$10^{-13}J/16op.$]
4	300	3.56	0.6	2.49
8	300	7.11	0.5	1.70
16	300	14.22	0.4	1.01

2.3 Proposed Architecture

The chosen architecture of a chip design mainly determines the properties of an implementation including power consumption and performance. For a system which is powered by battery or energy-harvester, the stringent energy constraints dominate architectural decisions. Since most operations of AES are byte or word-oriented, they can be executed efficiently on 8-bit, 32-bit, 64-bit, or any word size processors. 128-bit architecture offers the greatest degree of parallelism to increase concurrency of AES computations, and it allows energy-efficient operation while satisfying the throughput requirement.

The AES algorithm encrypts 128-bit blocks of plaintext by repeatedly applying similar round transformations for 10, 12, or 14 times depending on the length of secret keys. Therefore, most AES hardware implementations have one round realized in hardware which is reused to compute all the iterations. To increase throughput of the system, some implementations even unroll the ten iterations of the round transformation and insert pipeline registers between the round transformations. Unfortunately, for modes of operations which have feedback such as CBC mode, pipelining and/or loop unrolling techniques cannot be used as methods for improving throughput. It is because encryption of a plaintext cannot start until processing of the previous text is completed for these modes of operations.

This section will provide previous works of AES hardware implementations and analyze each system with respect to throughput and power consumption characteristics. We will propose a maximally parallelized AES architecture to satisfy energy efficiency as well as throughput requirement, and will optimize the main calculating block to save power consumption of the system further.

2.3.1 Previous Works

The previous hardware implementations of AES algorithm can be categorized into two types depending on the design principles and applications. The first type of implementation is a lightweight design for RFID devices or sensor networks, and the second type of implementation aims for high-speed operation. The lightweight implementation requires low die size and low power consumption while data throughput is of minor importance. In contrast, high-speed implementations optimize system for high data throughput while the required hardware resources and the power consumption are of minor importance. Our design goal of AES engine does not fit perfectly into both of the implementations, but partially coincides with both. Our goal is to design an AES engine suitable for energy-constrained applications such as battery-operated medical devices, but still meet throughput requirements of hundreds of Megabits. Therefore, we will delve into previous works to study how the specifications are traded off with each other, and then we will propose an energy efficient architecture that accommodates the throughput requirement.

For lightweight implementations, design methodologies should be adopted to minimize the number of CMOS gates and to reduce the mean power consumption of a system, since the targeted applications like RFID tags only can support thousands of gates complexity and few μW of power. Therefore, the main design techniques for lightweight implementations aim to realize minimum number of calculating blocks in a system and to try to reuse the block repeatedly by serializing encryption operation and finding out the sub-operation which needs the same calculating block. An example of the lightweight implementation is provided in [5]. In this design, an 8-bit architecture of AES is implemented and only one S-box is realized in the datapath.

The main parts of the AES are the controller, the RAM, the datapath, and the IO module as shown in Figure 2-3. The encryption operation and the key generation are all serialized in order to make it possible to conduct the operations with the given 8-bit datapath. The controller which is realized as a finite-state machine sequences the ten rounds of operations which consists of the sub-operations including SubBytes, ShiftRows, MixColumns, AddRoundKey. Totally, 1032 clock cycles are needed for this architecture to complete the one encryption including the IO operations. This design is optimized for resource-constrained applications so that it consists of only 3400 gates and consumes only $4.5\mu W$. However, it can support only upto $9.9Mbps$ throughput rate. Moreover, the reduction of power consumption does not necessarily result in the reduction of energy consumption. It is because this design reduces the mean power by serializing the required calculations. As a result, it needs more than thousands of clock cycles to complete one encryption and the large number of clock cycles deteriorates the throughput and energy efficiency of the system.

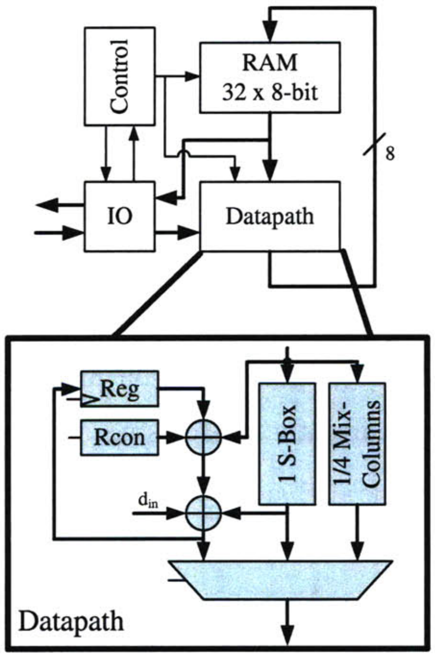


Figure 2-3: Architecture of 8-bit AES module [5]

Next example of an AES implementation requires data rates in the Gigabit range for network applications. For these applications, parallelism, pipelining and loop unrolling are said to be the concepts which increase the performance of a system per second.

$$\begin{aligned} Throughput &= \frac{Block\_size}{Latency} \\ Latency &= \frac{T_{clk} \cdot \#Rounds\_per\_block \cdot \#Pipelined\_stages}{\#Utilized\_stages} \end{aligned} \tag{2.4}$$

Equation 2.4 describes how the throughput of a circuit depends on the block size and latency. Due to the fixed block size of 128 bits in the AES algorithm, latency should be minimized for highest data throughput according to Equation 2.4. T_{clk} is the clock period of the circuit and it depends on the longest critical path through the system. The value *#Rounds_per_block* is the number of rounds required to calculate one block of data. *#Pipelined_stages* stands for the number of pipeline stages in the architecture, and *#Utilized_stages* is the number of pipeline stages which can be used concurrently. Therefore, from this equation, it should be noticed that T_{clk} should be minimized by cutting the longest critical path with pipeline registers, and all the pipeline stages should be utilized in order to maximize throughput of the system. Figure 2-4 shows a fully pipelined and fully parallelized AES processor which achieves tens of *Gbps* [6]. In this design, all 10 rounds are unrolled and pipelined to increase the throughput of the system. Moreover, in each round, one more pipeline register divides the critical path of encryption datapath and round key datapath. Since each pipeline stage is designed with 128-bit architecture, all the 128-bit block of data can be processed concurrently in each stage.

However, it has to be noticed that loop unrolling and pipelining techniques can improve the throughput of a system for limited modes of operations. The recommended modes of operation for a symmetric key encryption are defined by NIST [23]. For applications which demand higher level of security, it is needed to incorporate modes of operations which have feedback, for example Cipher Block Chaining (CBC) and Output Feedback (OFB). In these feedback modes, pipelining and/or loop unrolling cannot be used as methods for improving throughput, because only one round

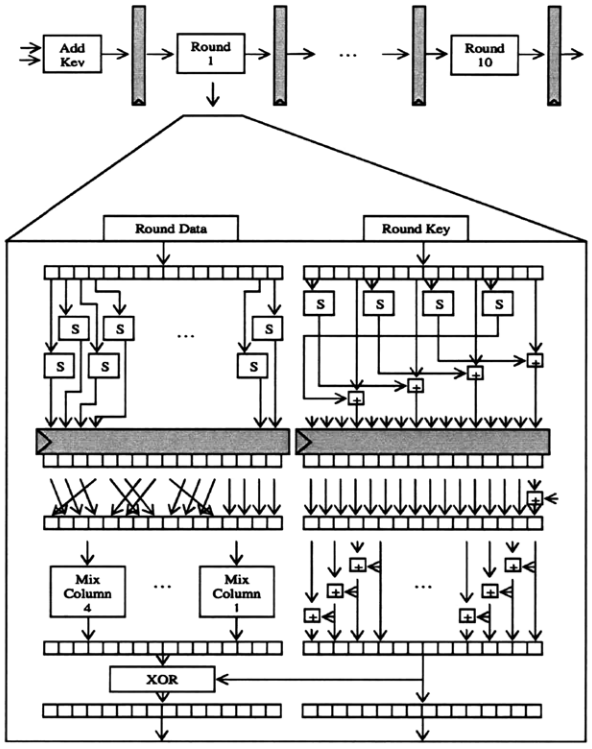


Figure 2-4: Maximizing throughput of AES with inner and outer round pipelining [6]

is active at a time and the pipeline never stay filled. Since the targeting modes of operation in our system also include feedback operation, pipeline cannot be adopted for architectural solution for high throughput requirement.

2.3.2 Architecture Optimizations

From the analysis of previous works for AES hardware implementations, it is found that a 128-bit architecture with parallel realization of 16 S-boxes is optimal to achieve high throughput requirement and energy efficiency simultaneously. Concurrent calculation of 16 bytes of data makes it possible to run each block at lower frequency compared to a non-parallelized architecture, and it allows to lower the supply voltage. Therefore, parallelism with voltage scaling results in energy-efficient operation while satisfying throughput requirement. Pipelining can also be used to increase throughput requirement in an energy-efficient manner, but feedback modes of operation prevents efficient use of pipelining.

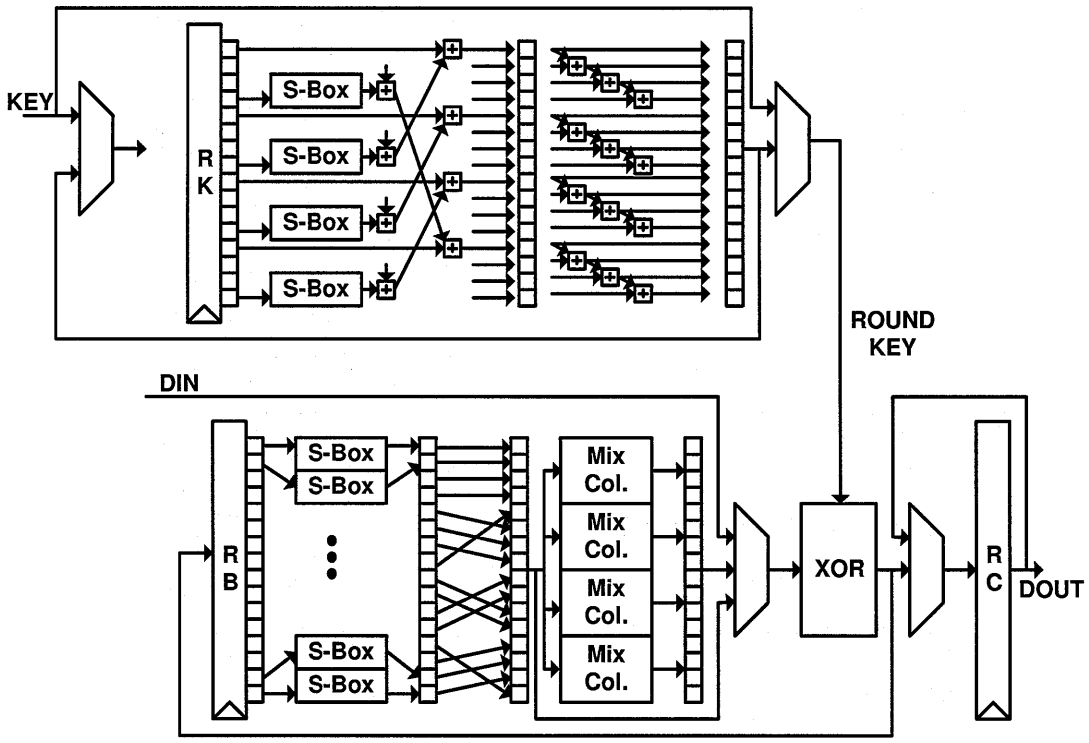


Figure 2-5: Parallel architecture of AES hardware [1]- There exits two datapaths. One is for round key generation (top), and the other is for round transformation of encrypting data (bottom).

Figure 2-5 shows an energy efficient implementation of an AES engine suggested in [1], which realizes one round of the AES-128 encryption algorithm. For AES-128, the key length and the input size are the same as 128 bits, and a round transformation is repeated 10 times. The architecture of one round implementation contains two different datapaths, the encryption datapath and the key scheduling datapath. To reduce the storage requirements, the round key is generated by on-the-fly key scheduling, not by using specialized RAM memory which should store 11 128-bit round keys.

The encryption datapath contains four sequential sub-operations of a round transformation, including SubBytes, ShiftRows, MixColumns, and AddRoundKeys. The substitution (S-box) step is a non linear operation which consists of multiplicative inverse in Galois field of $GF(2^8)$ and the followed affine transformation. This is the most costly sub-operation of each round in the respects of delay, power and area. Therefore, optimizing the critical path and/or developing a power-efficient algorithm for S-box can improve the power/performance characteristics of AES engine significantly. For 128-bit architecture, ShiftRows is performed just by rewiring so that it consumes only a small amount of power due to interconnect capacitance. In the MixColumn block, the bytes of each column are mixed together by multiplying the round data with a fixed polynomial modulo $x^4 + 1$. The MixColumn is implemented using a chain of XORs to minimize delay of this unit. For KeyAddition, the round data is simply XOR-ed with the round key, which is generated from the key scheduling datapath.

The key scheduling datapath consists of substitution operation of a word, and followed by a sequence of XOR operations. It is important to match the critical path delay of the key scheduling datapath and that of the encryption datapath from Substitution to MixColumn operation in order to reduce glitches at round key addition.

In this 128-bit architecture, only 10 clock cycles are needed to complete the encryption of one block data. To increase throughput of the system further, critical path of S-box should be minimized. We will deal with this topic at the next section.

2.3.3 Block Optimizations

As discussed previously, design of a S-box affects characteristics of AES engine significantly because delay, power, and area of a S-box dominate performance characteristics of main datapaths in an AES engine. Moreover, the effect of S-box is magnified especially for the fully parallelized architecture of Figure 2-5, since the number of S-boxes placed in this architecture is 20 times more that that of non-parallelized architecture shown in Figure 2-3.

From the area, delay, and power analysis of diverse S-box implementations in [11], the S-box architecture suggested in [7] is the most power-efficient and comparably fast implementation. The overall idea for this implementation is depicted in Figure 2-6. First, the 8-bit input of an S-box grows to 2^8-bit through a one-hot decoder. It expands the 8-bit input to a group of 2^8 bits output among which the legal combi-

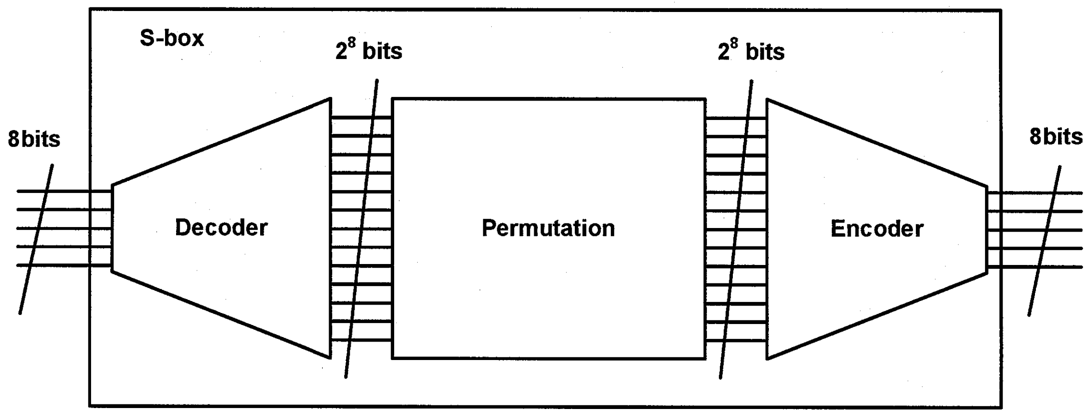


Figure 2-6: Diagram of the S-box with the decoder, permutation, and encoder block [7]

nations of values are only those with a single high (1) bit and all the others of low (0). This decoding technique is called as One-hot coding. Therefore, the location of the only one high bit (among 2^8 bits) indicates what the input data to S-box is. At the permutation block, the one and only one active line will change place inside the string of 2^8 bits to generate an adequate output value for substitution operation, and then the 2^8 bits are re-mapped to 8-bit binary data at the encoder block. Since one and only one line is active for any input data, and since the permutation block rearranges bits just by rewiring, this implementation of S-box consumes very low power compared to LUT or calculating implementations. Moreover, the critical path delay of this design is as short as that of LUT implementations. Therefore, we decide to use this design of S-box for our AES engine as shown in Figure 2-5.

2.4 IC Performance Simulation Results

The silicon implementation of an AES engine described in this chapter was realized using a 90nm CMOS standard-cell library from IBM. The circuit was described and verified in Verilog on register transfer level, and it is synthesized using the Synopsys Design Compiler. Placement and Routing were done with Astro from Synopsys. Continuous testing and verification eliminated errors during the design steps. Back-end verification ensured manufacturability. These tests consisted of static timing analysis, power simulations, LVS and DRC. After place and route, verification of the correct functionality was done in the Nanosim environment. The region of operation regarding supply voltage and clock frequency was evaluated using Nanosim simulations.

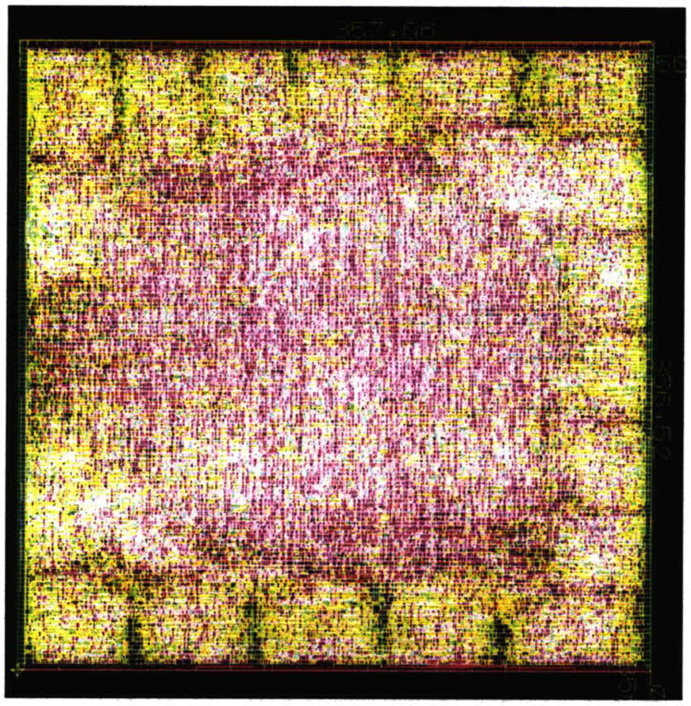


Figure 2-7: Layout of energy-efficient AES engine

Figure 2-7 shows the final layout of AES engine described in this chapter. The core needs a silicon area of $0.13mm^2 (= 0.36mm \times 0.36mm)$ on a $90\mu m$ CMOS, which includes 20854 gates. This number is much bigger than the result from synthesis. The difference in the layout complexity is due to the clock tree, filler cells and other layout overhead. The largest part of the circuit is the combinational logic of 16 S-boxes in encryption datapath. It consumes about 47% of the chip. The controller including the finite-state machine is the second largest portion of the circuit, which consumes 11% of the chip, and flip-flop based registers occupies 10% of the total silicon area.

Nanosim simulation of the extracted AES netlist indicates correct functionality until $0.35V$. R/C parasitics extracted from the layout of AES are included in simulations to improve accuracy of power/delay measurements. At $0.35V$ of supply voltage, a clock frequency of $5.6MHz$ is reached. At the maximum supply voltage of $1V$, the maximum clock frequency of $250MHz$ is reached. In our design, the encryption of one 128-bit data block requires 11 clock cycles including I/O operations. Therefore, the maximum achievable throughput rate is $2.9Gbps$ at $1V$, and is $64.6Mbps$ at $0.35V$.

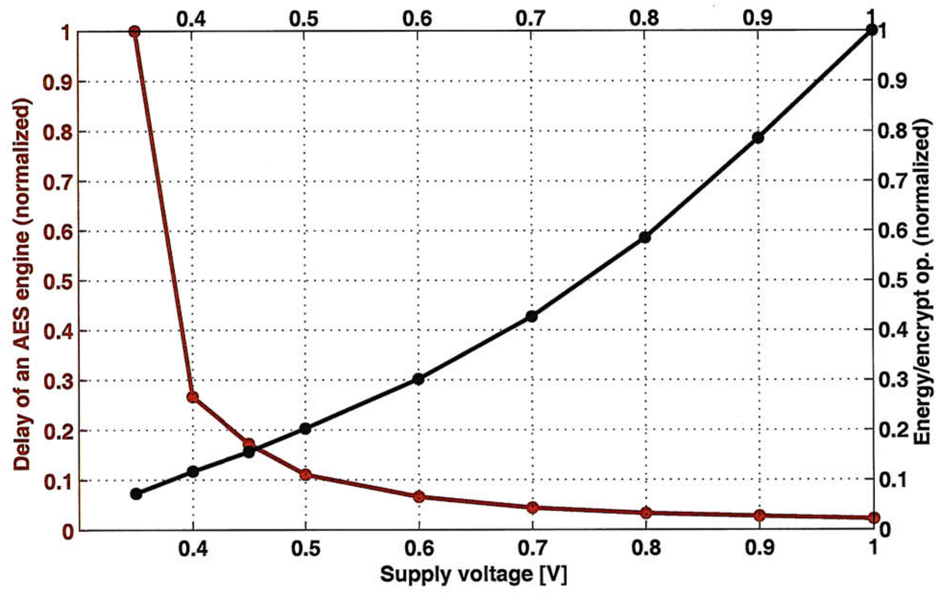


Figure 2-8: Normalized delay (red line) and energy (black line) of the AES engine with respect to supply voltage.

Figure 2-8 shows the normalized delay and energy consumption per encryption operation of the AES engine with respect to supply voltage. Here, the minimum energy point coincides with the minimum operation voltage as in S-box implementation. The increase of energy consumption in deep subthreshold as in Figure 2-1 cannot be observed for energy curve of AES engine. It is because the switching activity of signal nodes is very high for encryption operation. The large α factor in Equation 2.1 makes active energy still dominate total energy in deep subthreshold regions even though leakage energy grows fast due to the exponential increase of delay of a circuit. The switching activity, α, can be calculated by dividing the total number of node toggles in simulation with the total number of cycles multiplied by the total number of nodes in the circuit as shown in Equation 2.5.

$$\alpha = \frac{\#node\_toggles}{\#cycles \times total\#of\_nodes} \quad (2.5)$$

The calculated α for the encryption engine was 0.33. From the simulation results of AES engine, it was known that operating at the minimum energy point of 0.35V rather than at the maximum V_{DD} of 1V reduces the energy per operation by 92.74%.

Table 2.2: Performance comparison of different ASIC AES implementations.

AES-128 Version	Tech. [μm]	V_{DD} [V]	CLK Per.[ns]	Throughput [Mbps]	Power [mW]	CLK cycles /encrypt.	Energy/op. [pJ]
Verbauwhede[24]	0.18	1.8	8	1600	56	11	4923
Satoh[25]	0.13	1.5	1.1	11600	1920	11	23234
Feldhofer[5]	0.35	1.5	12.5	9.9	0.0045	1032	58.05
Our work	0.09	1	4	2900	17	11	747.12
	0.09	0.35	180	64.6	0.027	11	54.12

Table 2.2 shows a performance comparison of the presented work with ASIC AES-128 implementations in [24], [25], and [5]. Many FPGA implementations of AES have also been reported, but it is not straightforward to compare FPGA designs with our ASIC design since design costs and characteristics are very different. The design goal of the first two implementations in Table 2.2, [24] and [25], is to achieve high data throughput upto *Gbps* range. In contrarst, Feldhofer [5] designed an AES engine which is suitable for RFID tags with low power and low die size. Technology and supply voltage are all different for the four designs in Table 2.2, which makes it hard to compare the performance and energy-efficiency in a fair way. However, we can still analyze and compare the different trade-offs between throughput and energy or power, depending on the design goals and implementation characteristics of each design.

Except Felhofer [5], which uses 8-bit architecture, all the AES designs adopt 128-bit architecture to satisfy the high throughput requirement. Satoh [25] used Binary Decision Diagram (BDD) S-Box architecture which is optimized for minimum critical path delay, and thus, it could achieve the highest throughput among the previously reported ASIC implementations of AES. However, it consumes more than $1W$ of power. Verbauwhede [24] succeeded in achieving the high throughput of a few Gbps with much less power compared to Satoh [25]. Our design which runs at the maximum voltage of $1V$ exceeds the throughput of Verbauwhede [24], only with 30% of power compaerd to Verbauwhede's design [24]. Feldhofer [5] consumes the least power among the designs presented in Table 2.2. However, the serialized architecture of this design requires more than thousand clock cycles to complete one encryption, so that low power consumption from the design techniques does not result in low energy consumption per encryption operation. Compared to Feldhofer [5], our design which runs at $0.35V$ achieves $6.5\times$ improved throughput with 7% less energy per encryption operation. Therefore, our system which adopts parallelism with voltage scaling satisfies both the high throughput and energy efficiency simultaneously.

In summary, this chapter proposed a fully parallelized architecture of AES-128 engine. Parallelism, combined with voltage scaling, makes it possible to achieve energy efficiency as well as throughput requirements. IC performance results of the proposed engine showed that it can achieve $64.4Mbps$ with only $54.12pJ/encrypt.$ at $0.35V$, which makes the system suitable for energy-constrained applications. The proposed AES engine is, however, prone to DPA attacks since it does not incorporate any countermeasures against DPA. The next chapter will show how DPA attack can disclose secret key of the system.

Chapter 3

DPA Resistance Test

A differential power analysis attack on the AES core can be performed using a correlation attack on the the transient signature of the core IC power supply. To attack our system suggested in Chapter 2, we pick a transient time at which, the register in encryption datapath is updated to a known value, such as the ciphertext. By using the known ciphertext, we can estimate a switching power of the register at the transient time with a power model based on the AES algorithm and a guessed secret key. Correlation between this power model and the measured transient power can reveal the secret key after large number encryption runs by using the recorded data sets of ciphertext and the corresponding measured power.

In this chapter, DPA attack methodology will be provided in detail to find an effective power model and correlation function to discover the secret keys. The effectiveness of DPA attack will be shown with measurement results of the attack on the AES engine. The number of measurements to disclosure (MTD) would be a metric to indicate how the system is prone to DPA attacks.

3.1 DPA Attack Methodology

To discover secret key of AES engine using a correlation attack, we first have to guess a secret key and estimate a power consumption at a certain transient time based on the key guess. The estimated power is compared with a real measurement of the power consumption to find out the most highly correlated power estimate and the corresponding secret key. The only data which is externally accessible is the ciphertext, the encrypted result of a plaintext. We can use the known ciphertext to model a reasonable estimate of power consumption at registers with knowledge of the AES algorithm and guess of a secret key. In this section, a previously suggested DPA attack methodology is provided and modified to efficiently discover the secret key of AES engine.

3.1.1 Previous work for DPA attack

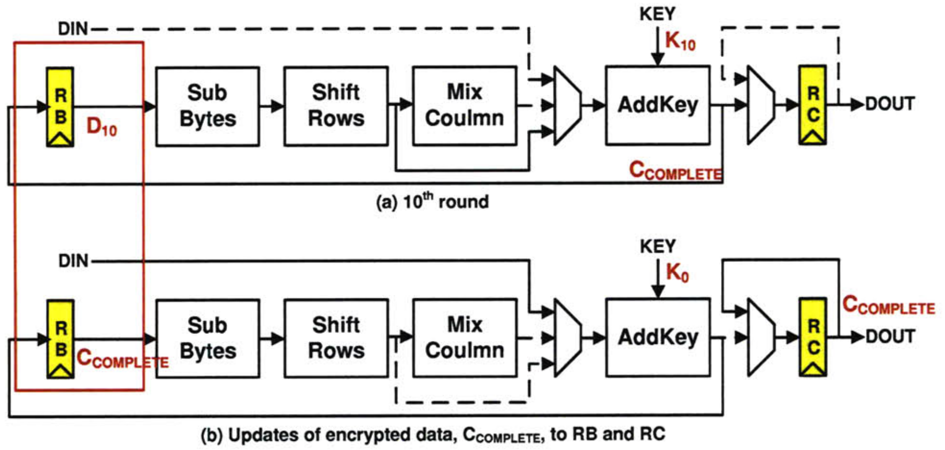


Figure 3-1: AES core [1]: round 10 (top); and round 10+1 (bottom). Encryption data flows through the solid lines at each round.

Figure 3-1 describes one of the most effective attacks on registers suggested by D. Hwang in [1]. During one encryption, the AES core encrypts a 128-bit plaintext, using 11 128-bit round keys (from K_0 to K_{11}), to produce a 128-bit ciphertext $C_{COMPLETE}$ after 11 rounds. At the 10^{th} round, input of the round, D_{10}, is transformed by SubBytes, ShiftRows, and AddRoundKey operations. The output of the round, $C_{COMPLETE}$, is updated to the RB and RC registers at the next clock cycle. D. Hwang modeled the transition power of RB register at the switching time of the 10+1 round, and compared it with a measured power to find a correlation between them.

The power supply current at transition of RB register is modeled by the following steps. First, the value updated to RB at the $(10+1)^{th}$ round is simply the known final ciphertext, $C_{COMPLETE}$. The value stored in RB in round 10 (D_{10}) can be calculated by tracing back the signal obtained after XOR-ing the final ciphertext ($C_{COMPLETE}$) and the round key guess (K_{10}) through both the ShiftRows and SubBytes operations. In digital circuits, power is consumed during logic transitions when charge must be transferred from supply voltage to physical capacitance of a signal carrying circuit node. Therefore, current is supplied from power supply to a circuit node during 0 to 1 transition of the node. Due to the different capacitive load at each node, supply current would be all different for transition of the circuit nodes corresponding to each bit. However, for simplicity, we model the power supply current based on the number of bits where 0 to 1 transitions happen, without considering different capacitive load at each bit positions.

Based on these observations, we model the power consumption at transition time of round 10 to 11 as in Equation 3.1.

$$\begin{aligned} P_{guess} &\propto Dist_{0to1}(D_{10}, C_{COMPLETE}) \\ D_{10} &= SubByte^{-1}(ShiftRow^{-1}(K_{10} \oplus C_{COMPLETE})) \\ P_{msrd} &\propto max(I_{supply,10to11}) \end{aligned} \tag{3.1}$$

With the known ciphertext, $C_{COMPLETE}$, and a guessed secret key of the 10th round, K_{10}, we can calculate D_{10} by reversing sub-operations calculated in the 10th round. Due to the byte-processing structure of AES, the attack can take place byte by byte. Therefore, using the same measured data, each of the sixteen bytes of K_{10} can be hacked separately, so that the key search space of K_{10} is only 16×2^8, as opposed to 2^{128}. In [1], P_{guess} is modeled by Hamming distance between D_{10} and $C_{COMPLETE}$ but we modified it for more accurate power estimate. Here, P_{guees} is proportional to the number of bits at which 0 to 1 transition happen when RB register is updated from D_{10} to $C_{COMPLETE}$. The function, $Dist_{0to1}(A, B)$ in Equation 3.1 measures the specialized distance between two strings of equal length, A and B. The specialized distance between A and B is the number of positions at which the corresponding symbol of A equals logic 0 and that of B equals logic 1. P_{msrd} corresponds to the maximum current supply at the transition time of round 10 to 11, which captures the power consumption of the RB registers.

$$\begin{aligned} Crrl_{xy} &= \frac{\Sigma x_i y_i - n\overline{xy}}{(n-1)s_x s_y} \\ &= \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{\sqrt{n\Sigma x_i^2 - (\Sigma x_i)^2}\sqrt{n\Sigma y_i^2 - (\Sigma y_i)^2}} \end{aligned} \tag{3.2}$$

Correlation between P_{guess} and the measured power, P_{msrd}, is calculated based on the sample correlation in Equation 3.2. In Equation 3.2, x and y corresponds to P_{guess} and P_{msrd} samples respectively, $\overline{x}$ and $\overline{y}$ are the sample means, s_x and s_y are the sample standard deviations. The sum is from 1 to n, when n is the number of ciphertext samples. For each sample of ciphertext, P_{msrd} is measured from power trace and P_{guess} is calculated based on Equation 3.1. A key guess, K_{10}, which gives the highest correlation, $Crrl_{xy}$, between P_{guess} and P_{msrd} is assumed to be a correct round key used in the 10^{th} round of encryption process. Due to the reversible nature of the round key computation, once K_{10} is deduced, we can calculate the original key K_0, and DPA attack is completed.

3.1.2 Modification in Power Estimation

We first used the previously suggested DPA attack methodology to discover secret key of our system. Figure 3-2 shows the logic transitions of CLK, Counter, RB out values and the transient measurement of core supply current, which we get from Nanosim simulation of the extracted netlist of our AES engine. We run large number of encryptions in the Nanosim environment and record current consumption from the power supply. The maximum current at the transition of 10^{th} to 11^{th} round is recorded as P_{msrd}, and it is used to calculate correlation with P_{guess}.

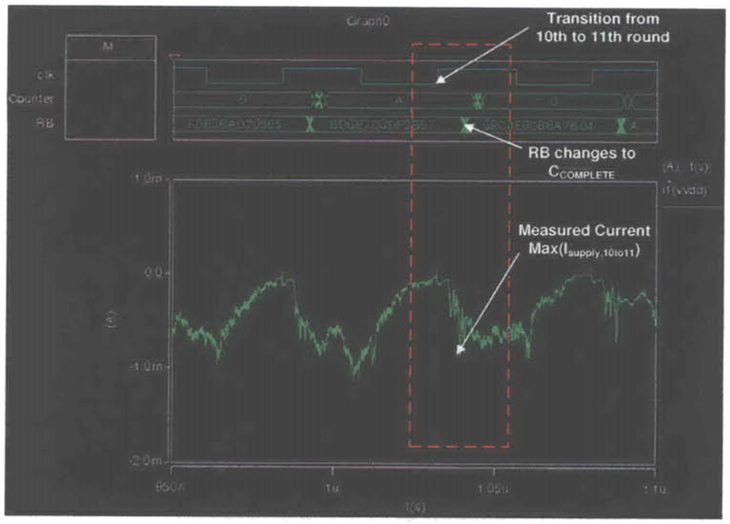
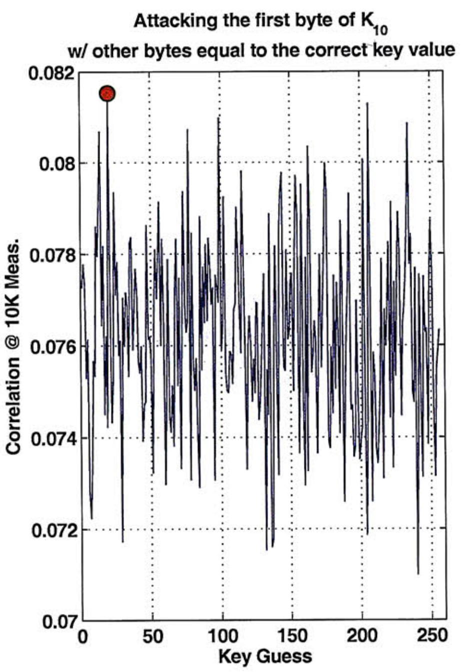


Figure 3-2: Logic transition of CLK, Counter, RB out signals (top) and transient measurement of core supply current (bottom) near 10th to 11th round transition.

Due to the byte-oriented structure of AES, we can attack each byte of K_{10} separately. The possible 256 guesses for the first byte of K_{10}, $K_{10}[0]$, gives different correlation with P_{msrd}, from which we can discover the real secret key. In Equation 3.1, 128-bit K_{10} guess is required to calculated D_{10}, so that we first assume all the bytes of K_{10} except $K_{10}[0]$ as the correct key value, and measure correlation between the measured power and power estimates for 256 different key guesses of $K_{10}[0]$. Then, we try the same attack with different setting for the last 15 bytes of K_{10} as all 0. Figure 3-3 shows the results of two attacks for the same byte, $K_{10}[0]$, with different setting for ($K_{10}[1]...K_{10}[15]$). Both attacks could disclose the correct key of $K_{10}[0]$ after $10K$ encryptions as shown in the graph, at which red marker (the correct key) has the highest correlation among 256 different key guesses. However, the bias of correlation differs a lot depending on the setting for $K_{10}[1]...K_{10}[15]$. When $K_{10}[1]...K_{10}[15]$ are set as the correct key bytes (left graph), the average of correlation for 256 different key guesses is about 0.077, while the average is -0.007 for $K_{10}[1]...K_{10}[15]$ equal to all 0 (right graph). Even though both attacks were successful, the different settings for $K_{10}[1]...K_{10}[15]$ influences calculation of the correlation coefficients for $K_{10}[0]$, and it can result in misinterpretation of DPA attack measurements for the attacking byte.



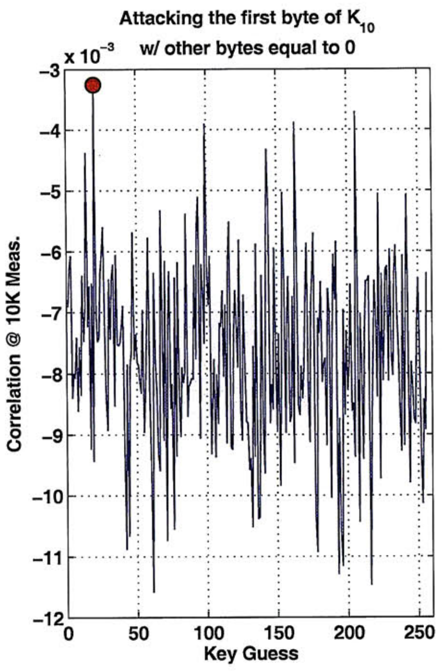


Figure 3-3: Correlation coefficients vs. 256 key guess of $K_{10}[0]$ at $10K$ encryption runs. ($K_{10}[1]...K_{10}[15]$) is set to correct key values (left), and is set to (0...0)(right). Depending on the assumption of ($K_{10}[1]...K_{10}[15]$), correlation has the different bias. The red marker in the graphs shows the correct key $K_{10}[0]$ of AES engine

Therefore, we modified the power estimation model to be only affected by the target attacking byte, but not by the last of key bytes which we do not care when we attack a byte.

$$\begin{aligned}
P_{guess} &\propto Dist_{0to1}(D'_{10}[a], C_{COMPLETE}[b]) \\
D'_{10}[a] &= SubByte^{-1}(K_{10}[a] \oplus C_{COMPLETE}[a]) \\
P_{msrd} &\propto max(I_{supply,10to11}) \\
(a,b) &= \{(0,0),(1,5),(2,10),(3,15),(4,4),(5,9),(6,14),(7,3), \\
&\quad (8,8),(9,13),(10,2),(11,7),(12,12),(13,1),(14,6),(15,11)\} \quad (3.3)
\end{aligned}$$

Equation 3.3 summaries the newly proposed DPA attack methodology which is totally byte-oriented. At the 10^{th} round of AES algorithm, encryption data is transformed by SubBytes, ShiftRows, and AddKeyRound, but not by MixColumns. Therefore, when we calculate D_{10} by reversing the AES algorithm, each byte of D_{10} can be found independently, once we know the corresponding byte of $C_{COMPLETE}$ and the byte of K_{10}. No other bytes are required for calculation of the byte of D_{10}. For example, when $K_{10}[0]$ is a byte we try to discover, we can first calculate $D_{10}[0]$ by reversing a byte, which is obtained after XOR-ing $C_{COMPLETE}[0]$ and a guess $K_{10}[0]$ through SubBytes operation. The estimate of power consumption regarding this key byte, $K_{10}[0]$, is also modeled by the specialized distance between $D_{10}[0]$ and $C_{COMPLETE}[0]$ as defined in Equation 3.1. Because ShiftRows operation shown in Figure 3-4 changes order of the bytes, we have to carefully match the order of the bytes in D_{10} and $C_{COMPLETE}$ to measure the specialized distance in the P_{guess} calculation. In Equation 3.3, D'_{10} is a byte-order changed version of D_{10} and (a, b) give the adequate pairings for D'_{10} and $C_{COMPLETE}$ to calculate P_{guess}.

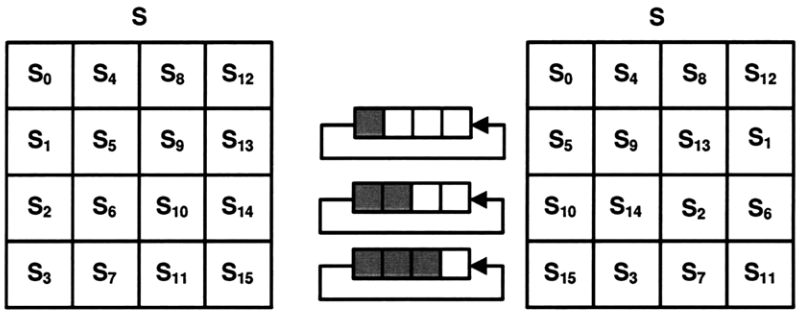


Figure 3-4: ShiftRow operation of AES algorithm

In this DPA attack methodology, when we attack a byte of K_{10}, the other bytes of K_{10} do not need to be set as some default values. Therefore, correlation between P_{msrd} and P_{guess_byte} in Equation 3.3 only captures the effect of the attacking byte on power consumption, which makes it possible to attack each byte of K_{10} totally independently.

3.2 DPA Attack Measurements and Results

The resistance against DPA can be quantified as the number of measurements to disclosure (MTD). The definition of MTD in [1], the crossover point between the correlation coefficient of the correct key and the maximum correlation coefficient of all the wrong key guesses, is used as a measure of DPA resistance.

The energy-efficient AES engine proposed in Chapter. 2, is attacked by the byte-oriented DPA attack methodology described in previous section. Among 16 bytes of secret key, 12 bytes were found easily within $20K$ measurements. Figure 3-5 shows the simulation results of successful DPA attacks which discover the correct key within $5K$ encryption runs. The correlation coefficients are plotted as a function of encryption runs in the left figures. Correlation coefficients for the correct key (Red line), and the max (Blue line)/min (Green line) correlation coefficients among 255 wrong keys are shown in the figure. The number of measurements at which the red curve (the correct key) starts to exceed the blue curve (a incorrect key which has maximum correlation) is an MTD value for the key byte. The 16^{th} key byte requires the minimum MTD of 3100 among 16 key bytes. Right curves show the calculated correlation values for 256 key guesses at $20K$ encryption runs. As shown in Figure 3-5, signal to noise ratio is high enough to distinguish the correct key among 256 key guesses.

Figure 3-6 shows two examples of DPA attacks which need more than $10K$ encryption runs to discover the correct key of AES engine. The 6^{th} (top) and 4^{th} (bottom) key bytes can be found after about $15K$ and $12K$ encryption runs, respectively. SNR of correlation coefficients for these key bytes at $20K$ runs (shown in the right curves) are not as high as those of 16^{th} or 14^{th} key bytes in Figure 3-5, but the correct key can still be distinguished from the curves. The difference in MTD is caused by the different contribution of each key byte in the total power consumption of a system. When we model a power consumption at registers, it is assumed that all the capacitive load corresponding to each bit of the register are similar so that we can estimate power consumption of register with the number of logic transitions. However, this assumption is just for the simple power model. In reality, the capacitive load for each bit position can differ significantly, so that the contribution of each byte in total power consumption also varies depending on the capacitive loads for the byte. Moreover, when we attack a byte of secret key, power consumption at signal carrying nodes associated with all the last 15 bytes is interpreted as noise in the measured power trace. Therefore, if the amount of power consumption related to the attacking key byte is not significant compared to the power consumption associated with the other bytes, signal to noise (SNR) of correlation coefficient might not be good enough to discover the real key value of the byte. Figure 3-7 shows such cases where DPA attack methodology could not discover the correct secret key values even after $20K$ encryption runs. For the 11^{th} and 3^{rd} key bytes, contribution of the bytes in the total power consumption might be trivial, so that the correlation coefficients for all the key guesses are lower than 0.015 and there is no huge peak, which results in an unsuccessful attack. Table 3.1 summarizes DPA attack results for the unprotected AES engine.

Table 3.1: DPA Measurement Results

Measurements to Disclosure	
Min	3100
Max	19000
Mean	9885
Keybytes not found	4(@20K meas.)

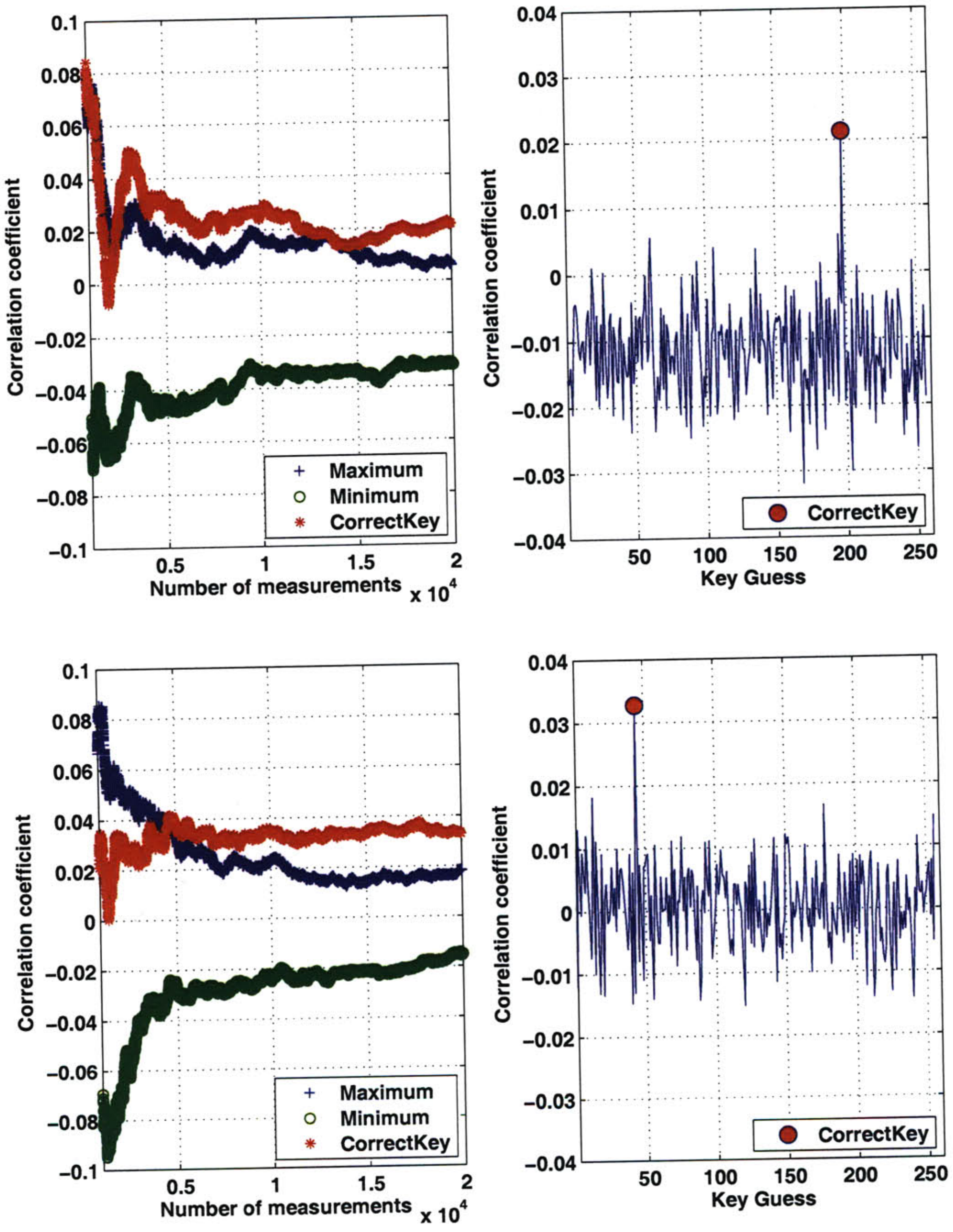


Figure 3-5: Cracking the secret key. Two examples of successful attack with less than 5K encryption runs (top: 16^{th} byte, bottom: 14^{th} byte of secret key). Correlation coefficients as a function of encryption runs (left): Blue for max. correlation among 255 wrong keys, Green for min. correlation among 255 wrong keys, and Red for the correct key. Correlation coefficients for 256 possible key guesses at $20K$ runs (right): Red marker is the correct key

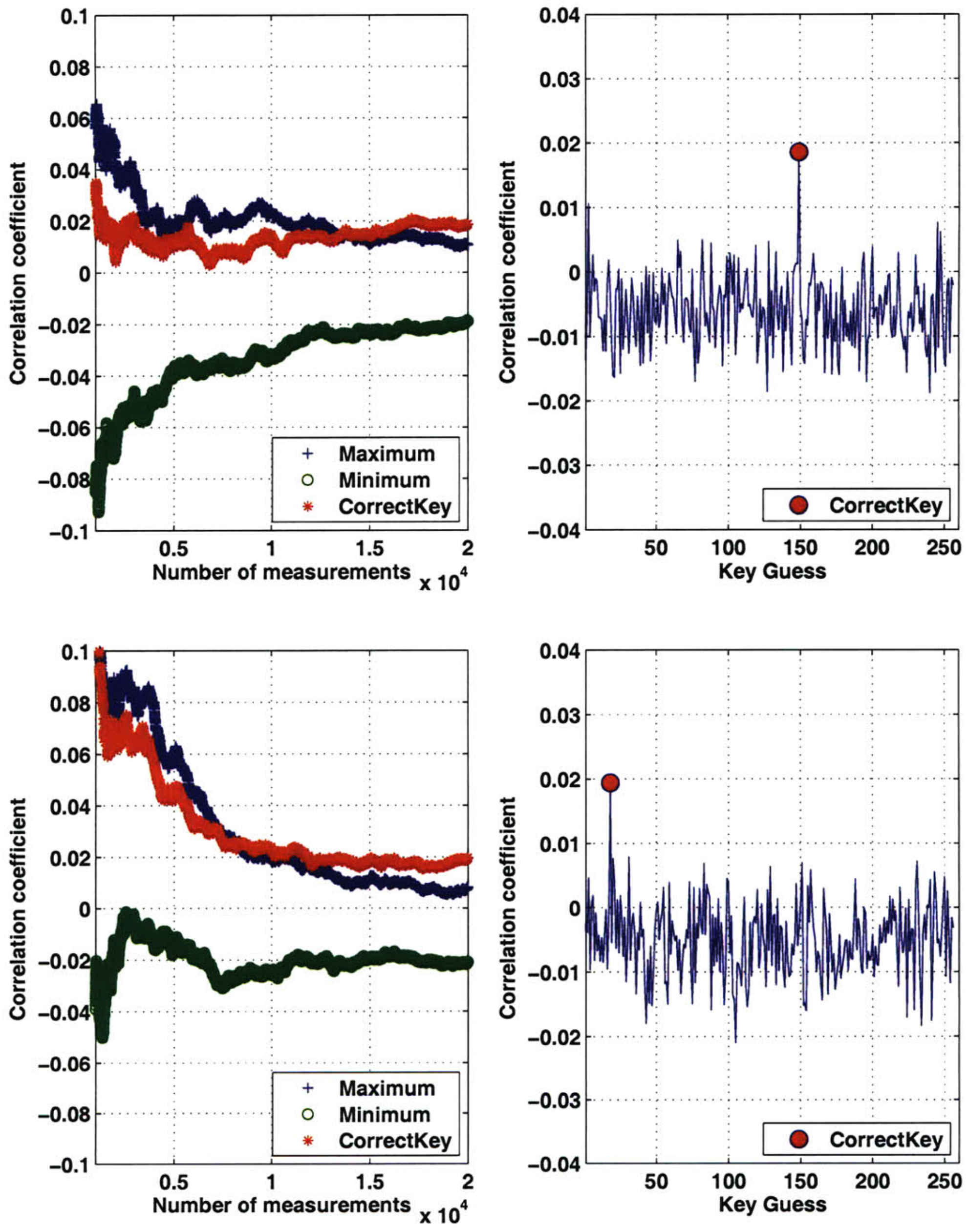


Figure 3-6: Cracking the secret key. Two examples of successful attack with more than 10K encryption runs (top: 6th byte, bottom: 2nd byte of secret key). Correlation coefficients as a function of encryption runs (left): Blue for max. correlation among 255 wrong keys, Green for min. correlation among 255 wrong keys, and Red for the correct key. Correlation coefficients for 256 possible key guesses at $20K$ runs (right): Red marker is the correct key

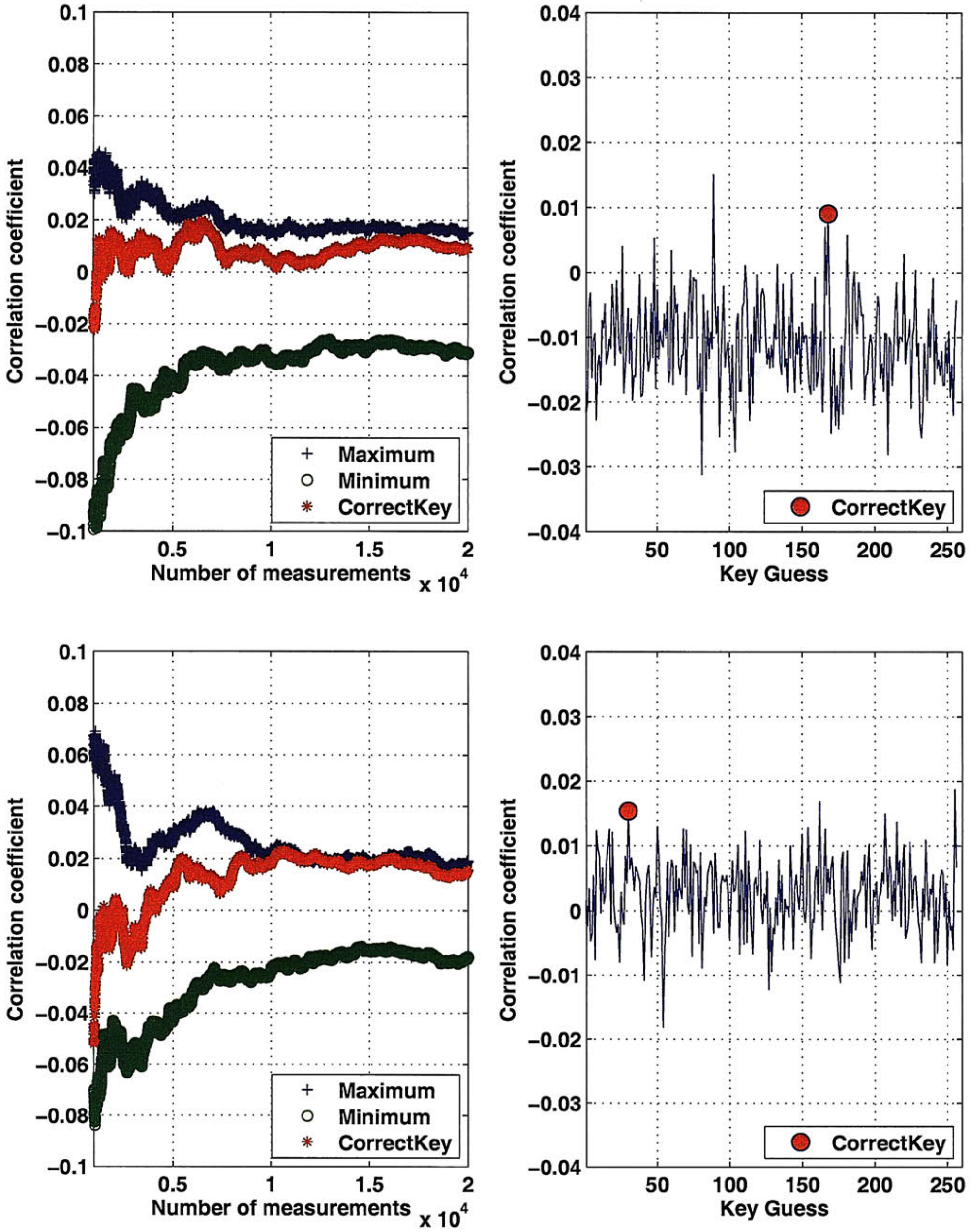


Figure 3-7: Cracking the secret key. Two examples of unsuccessful attack (top: 11^{th} byte, bottom: 3^{rd} byte of secret key). Correlation coefficients as a function of encryption runs (left): Blue for max. correlation among 255 wrong keys, Green for min. correlation among 255 wrong keys, and Red for the correct key. Correlation coefficients for 256 possible key guesses at $20K$ runs (right): Red marker is the correct key

Chapter 4

AES Engine with Resistance to DPA Attacks

In Chapter 3, it is clearly shown that DPA attacks on registers can discover secret key of AES engine when the engine does not incorporate any design techniques to protect against power analysis attacks. Therefore, we decide to modify our energy-efficient AES engine to be resistant to DPA attacks, which is based on the power model of registers.

It should be noted that DPA attacks on logic gates as well as on registers can also be effective to disclose secret keys used in encryption operation [26]. Most logic attacks use power models based on the Hamming weight of output value of the main logic blocks, such as the S-box in an AES processor. In [26], output of an S-box is calculated with a known plaintext and a guess of secret key. Power consumption of S-box is modeled by the Hamming weight of the output value, and DPA attack is conducted to find out one power estimate which is most highly correlated with the measured power, and the corresponding secret key. However, as opposed to register attacks which can find the transition time easily based on clock signal, DPA attacks on logic gates are hard to conduct since transitions occurring at the output of logic gates are very hard to predict. Moreover, because of the complexity of an AES processor, especially for the parallelized engine, the power model based on Hamming weight of one S-box output is not sufficiently correlated to the total power consumption as to be used to discover secret keys. Therefore, we focus on designing AES engine which can be resilient to attacks on registers, which are much more effective compared to attacks on logic gates.

The first section of this chapter will discuss about the idea behind the proposed techniques to protect AES engine against DPA attacks on registers. After that, we will show how to realize and apply the proposed idea to an energy-efficient implementation of AES engine. Then, the DPA measurement results will be provided for the new AES design. At the last of this chapter, IC performance results will be shown and compared with that of unprotected core to discuss about the power, area, and throughput overhead, which is the cost for the increased security.

4.1 Idea Behind the Proposed DPA Protection Techniques

From the previous discussion of DPA attack methodology, it is shown that the different Hamming distance, or the different number of 0 to 1 logic transitions at RB registers in Figure 3-1 (shown again as Figure 4-1) gives attackers side information about the intermediate data, from which secret key of the system can be found based on correlation attacks.

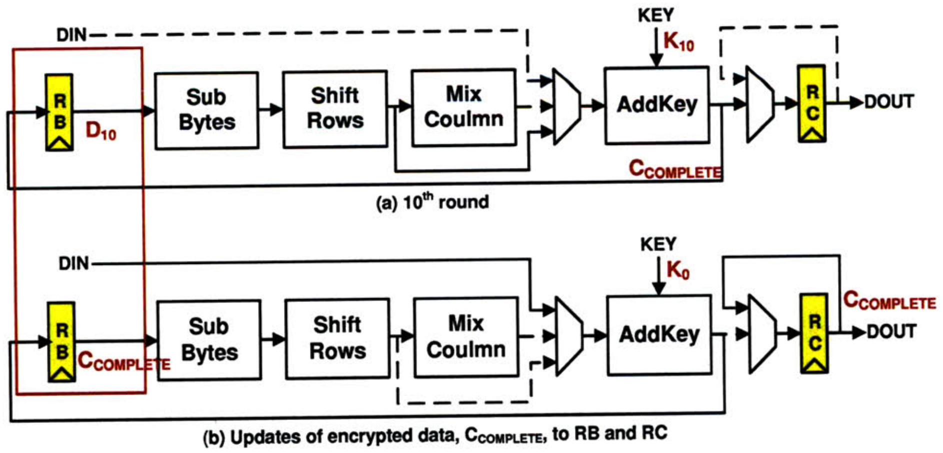


Figure 4-1: AES core: round 10 (top); and round 10+1 (bottom) [1]

Therefore, in order to prevent DPA attacks on registers, the input value to the RB register should be masked or manipulated to make the power consumption of the RB register independent from the encryption data. Since the power estimate for a DPA attack is fundamentally based on the Hamming weight of input and output of the register, an encoder and decoder is inserted in front of, and behind the RB register to make the Hamming weight of input to RB all the same regardless of the changing input data, and to make it to be independent from the real encryption data. In Figure 4-2, encryption data is encoded via the encoder block in front of the RB register to a value whose Hamming weight is a fixed constant. After updating the value at RB register, the data is decoded to a original format in order to be processed by the following sub-blocks.

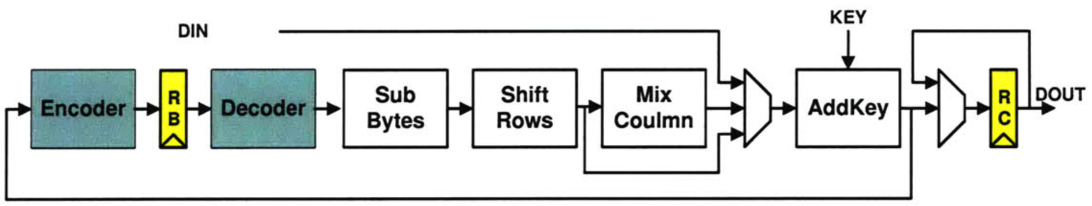
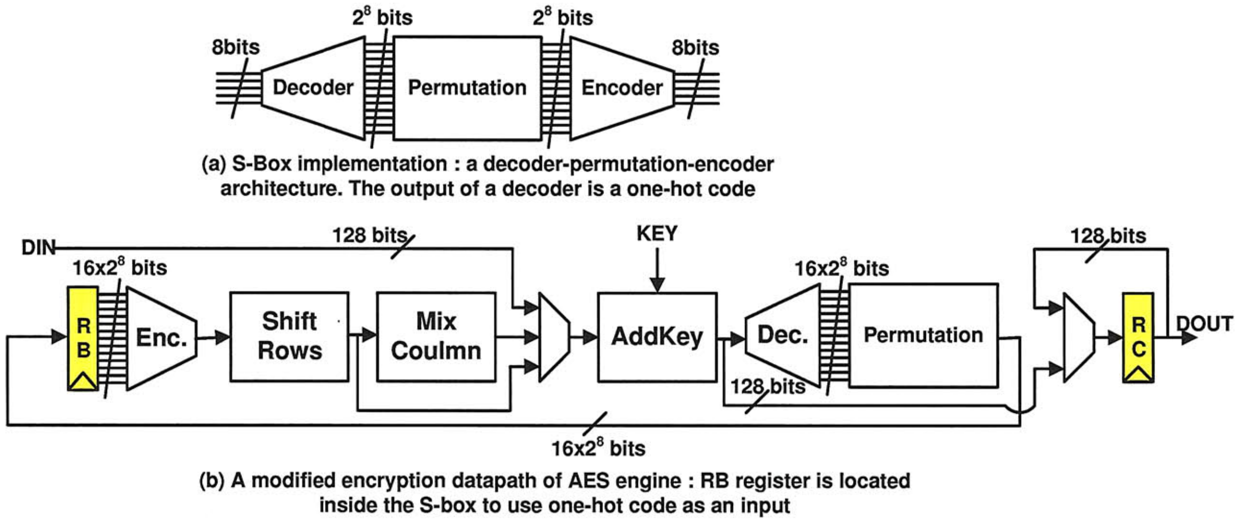


Figure 4-2: RB register is located between encoder and decoder. Encoder masks the encryption data to have a fixed Hamming weight, and decoder recovers data to the original format.

This is the basic idea to protect our system from DPA attacks on registers. Since the encoder/decoder is inserted at the critical path of the system, however, it increases delay of the encryption datapath. The increased delay of critical path directly deteriorates the throughput of the system. In the next section, we will suggest a solution for this problem of the suggested idea and realize the proposed DPA-protection technique in an energy-efficient way.

4.2 Architecture of AES Engine with DPA-Resistance

Addition of encoder and decoder block inside the critical path directly increases delay of the critical path and deteriorates the throughput of the system. Since inserting a new block in the datapth is totally undesirable in the respect of performance, it would be great if any existing blocks can function as like the encoding/decoding block as well. A power-efficient S-box design suggested in Chapter 2, provides the solution for this problem. The design of S-box is re-described in Figure 4-3-(a).



2^8 bits
2^8 bits
8bits
8bits
Decoder
Permutation
Encoder
(a) S-Box implementation : a decoder-permutation-encoder architecture. The output of a decoder is a one-hot code
DIN
128 bits
KEY
128 bits
16x2^8 bits
16x2^8 bits
R B
Enc.
Shift Rows
Mix Coulmn
AddKey
Dec.
Permutation
R C
DOUT
128 bits
16x2^8 bits
(b) A modified encryption datapath of AES engine : RB register is located inside the S-box to use one-hot code as an input

Figure 4-3: A secure AES architecture with resistant to DPA.

The output of a decoder in S-box is a group of 2^8 bits among which a single bit is logic high (1) and others are logic low (0) for any input data. This one-hot code can be used as input to *RB* register since the Hamming weight of one-hot code are all the same, 1, regardless of the changing encryption data. Therefore, if *RB* register is moved inside the S-box, between decoder and encoder block, to use the coded data as input to the register, the Hamming weight of input to *RB* is independent from the encryption data, which is a desirable characteristic to defeat correlation attacks on power measurements of registers.

Figure 4-3-(b) shows the modified encryption datapath of AES engine where *RB* register is moved inside the S-box. The only difference of this design from the energy efficient AES architecture suggested in Chapter 2 is the location and the size of *RB* register. For energy-efficiency and high throughput, 16 S-boxes are parallelized in our system. Therefore, the size of *RB* register is 16×2^8 in this design, which is $32\times$ larger than the size of *RB* register, 16×8, in the previous design, where the register is located in front of S-box.

The increased data-width of RB register results in area and clock power overhead. The capacitive load of RB registers are also magnified, and it might increase the delay of critical path. However, since only 16×1 bits among 16×2^8 bits of input to RB are logic high at every clock cycle, the transition power of RB register is comparable or can be smaller than that of RB in previous design where on average more than 50 logic high bits should be updated at every clock cycle. Moreover, clock power overhead for the increased RB register can be reduced by careful clock gating schemes.

The area, power, energy/encrypt op., and throughput overhead of DPA-protected core will be analyzed and compared with those of unprotected core. The overhead analysis will show the IC performance costs for the increased security of AES engine. Before showing the comparison analysis, we will first provide the DPA attack measurements of the newly proposed AES engine in the next section.

4.3 DPA Attack Measurements and Results

The proposed AES engine is attacked by the byte-oriented DPA attack methodology to demonstrate the improved security of the design. The number of measurements to disclosure (MTD) is used as a measure to quantify the DPA-protection ability of the engine.

The secured core has been subjected to $100K$ encryptions, $33\times$ more than the number of runs at which the same DPA attack could disclose a secret key from the unprotected core, but none of its secret key blocks have yet been disclosed. Figure 4-4 shows the simulation results and DPA measurements for the 16^{th} and 12^{th} key bytes, which were uncovered successfully for the unprotected AES core. The correlation coefficients are plotted as a function of encryption runs in the left figures. Correlation coefficients for the correct key (Red line), and the max (Blue line)/min (Green line) correlation coefficients among 255 wrong keys are shown in the figure. The number of measurements at which red curve (the correct key) starts to exceed the blue curve (an incorrect key which has maximum correlation) is a MTD value for the key byte.

For the protected core, however, red line never exceeds the blue curve, which means that correct key could not be discovered until $100K$ encryptions. The protected core substantially reduces the signal to noise ratio of correlation, as shown by the small correlation peaks in the right curves of Figure 4-4. Therefore, our measurements show that the newly suggested AES engine effectively protects all the sixteen key bytes against DPA attacks on registers.

4.4 IC Performance Comparison between Protected and Unprotected AES Engines

The silicon implementation of the protected AES engine described in this chapter was realized using the 90nm CMOS standard-cell library from IBM. The circuit was described and verified in Verilog on register transfer level, and it is synthesized using the Synopsys Design Compiler. Placement and Routing were done with Astro from Synopsys. Continuous testing and verification eliminated errors during the design steps. Back-end verification ensured manufacturability. These tests consisted of static timing analysis, power simulations, LVS and DRC. After place and route, verification of the correct functionality was done on Nanosim environment. The region of operation regarding supply voltage and clock frequency was evaluated using Nanosim simulations.

Figure 4-5 shows the final layout of the protected AES engine described in this chapter. The core needs a silicon area of $0.42mm^2 (= 0.65mm \times 0.65mm)$ on a $90\mu m$ CMOS, which includes 37583 gates. This ares is $3.2\times$ larger than that of unprotected core. However, the number of gates used in the system are increased only by 80.21%. This means that the density of gates in protected core is much lower than the density of unprotected core. It is because we set the chip area of the protected core with huge margins during place & route flow in order to prevent concentrated capacitive load at signal carrying nodes. The largest part of the circuit for protected core is the flip-flop based registers, which consumes about 51% of the chip. Combinational logic

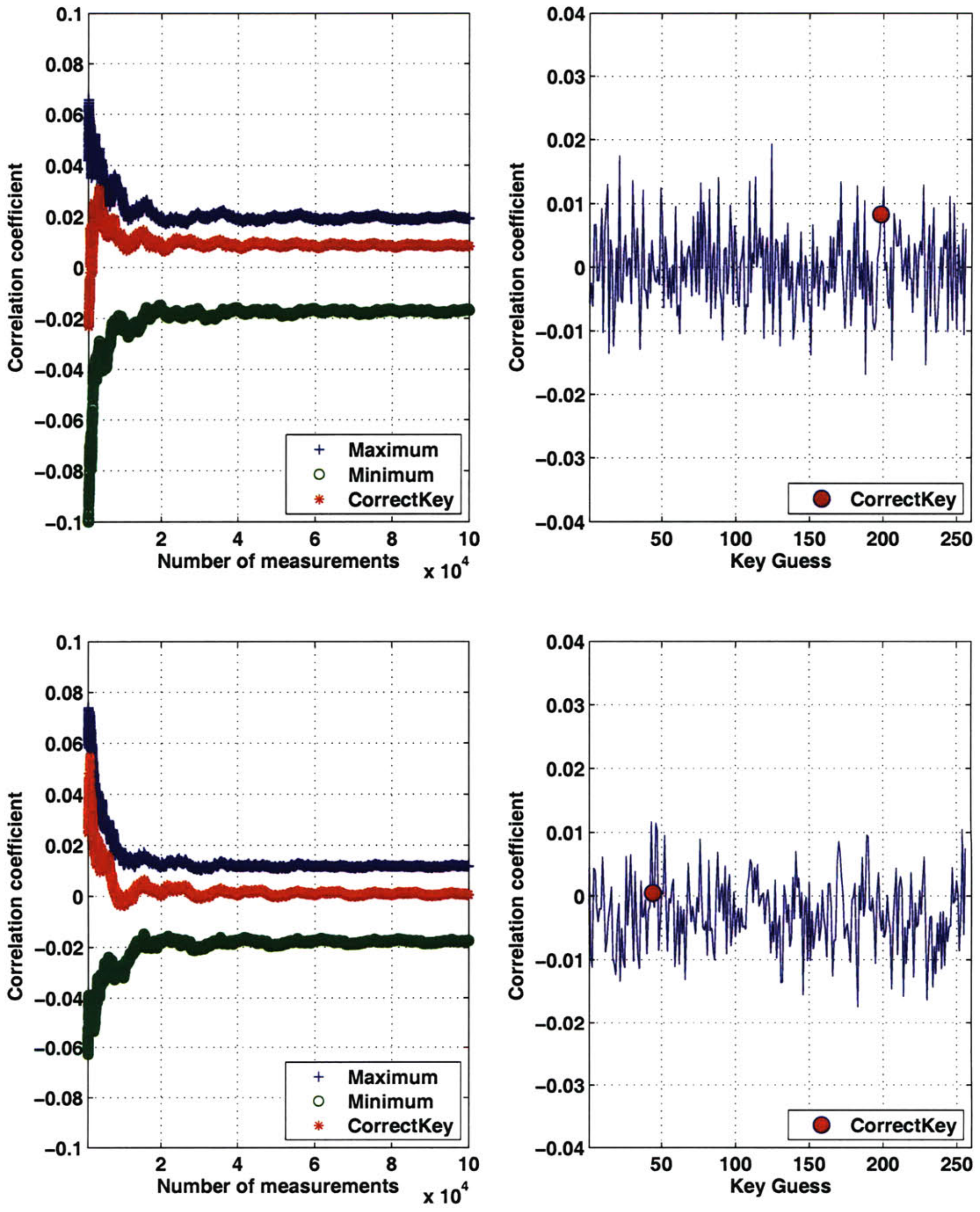


Figure 4-4: Cracking the secret key. Two examples of unsuccessful attacks with more than 10M encryption runs (top: 16^{th} byte, bottom: 12^{th} byte of secret key). Correlation coefficients as a function of encryption runs (left): Blue for max. correlation among 255 wrong keys, Green for min. correlation among 255 wrong keys, and Red for the correct key. Correlation coefficients for 256 possible key guesses at $10M$ runs (right): Red marker is the correct key

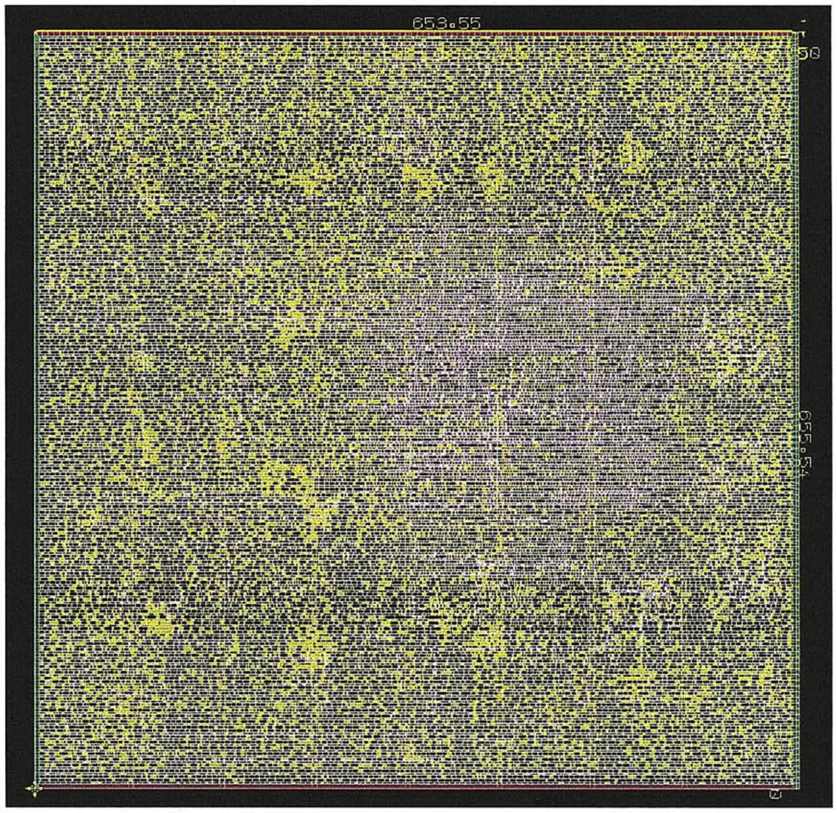


Figure 4-5: Layout of the DPA-protected AES core.

of 16 S-boxes, which consumed the largest portion of area in the unprotected core, occupies only about 14% of the total chip area for the protected core design.

Nanosim simulation of extracted netlist for the protected AES engine indicates correct functionality until $0.4V$. R/C parasitics extracted from the layout of AES are included in simulations to improve accuracy of power/delay measurements. At $0.4V$ of supply voltage, a clock frequency of $12.5MHz$ is reached. At the maximum supply voltage of $1V$, the maximum clock frequency of $167MHz$ is reached. In our design, the encryption of one 128-bit plaintext requires 11 clock cycles including I/O operations. Therefore, the maximum throughput rate for encryption is $1.9Gbps$ at $1V$, and the throughput rate at $0.4V$ is $145Mbps$.

Figure 4-6 shows the normalized delay and energy consumption per encryption operation of the AES engine with respect to supply voltage. From the simulation results of AES engine, it was known that operating at the minimum energy point of

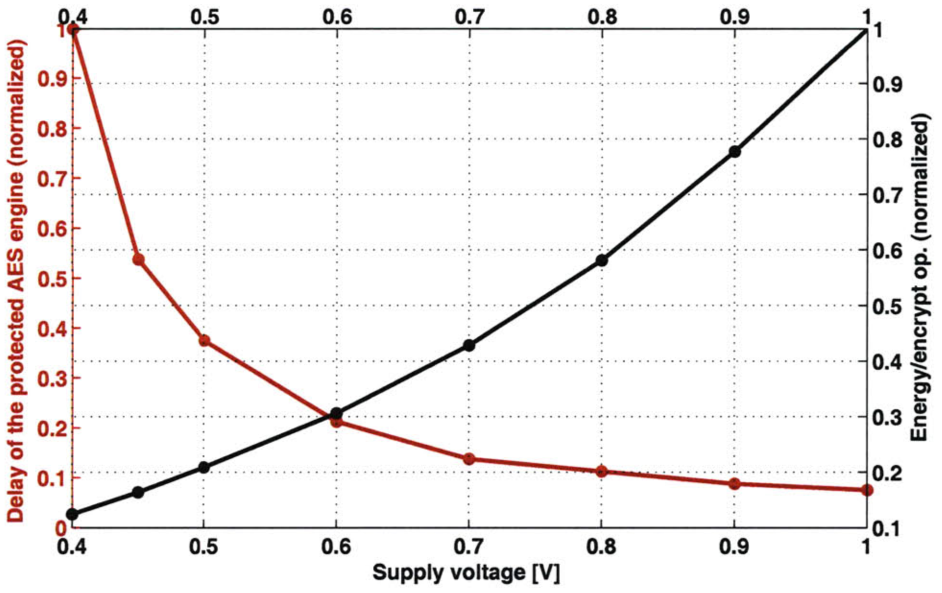


Figure 4-6: Normalized delay (red line) and energy (black line) of the protected AES engine with respect to supply voltage

0.4V rather than at the maximum V_{DD} of 1V reduces the energy per operation by 87.57%.

Table 4.1 summarizes and compares the IC performance results of the unprotected and protected AES cores. The proposed DPA architecture is proven to thwart power attacks by improving DPA resistance orders of magnitude over an unprotected AES core. The secured core has been subjected to $100K$ encryptions, 33× more than the number of runs at which attack can disclose a secret key of the unprotected core, but none of its secret key have yet been disclosed. When running the encryption core at $0.4V$ and $10MHz$, power increases by 2× compared to the unprotected core. Maximum throughput at $1.0V$ is reduced by 2/3 for the protected core compared with the unprotected core. These IC performance overhead is the cost of increased security.

Table 4.1: IC Performance comparison between the unprotected and protected AES engines.

Parameter	Unprotected	Protected
Area [mm^2]	0.13	0.42
Operating Range [V]	0.35 - 1.0	0.4 - 1.0
Power(0.4V,10MHz) [μW]	79.20	158.72
Maximum Throughput(1.0V) [Gb/s]	2.91	1.94
Measurements to Disclosure		
Min	3100	-
Max	19000	-
Mean	9885	-
Keybytes not found	4(@20K meas.)	16(@10M meas.)

Chapter 5

Conclusion

This thesis proposes an energy efficient AES-128 engine which protects against Differential Power Analysis (DPA) attacks. The main contribution of this thesis is the design of an AES architecture which balances the Hamming weight of encryption data in an energy-efficient way to protect the engine from correlation attacks on registers. Although hardware implementation of AES is widely used for secure storage or transmission of data in mobile devices, side-channel information from the switching behavior of CMOS gates allows attackers to find out the secret key. DPA attack is one of the most common side-channel attacks because of its simplicity and effectiveness [3]. In biomedical/mobile applications where the system is powered by a battery or energy harvester, a stringent energy budget poses the greatest challenge in the design of AES engine. Thus, several innovations have been made to design an energy-efficient AES engine and to develop an architecture which can mitigate DPA attacks. The proposed techniques were verified with a silicon implementation of the AES engine, and extensive simulations. Several contributions, and possible improvements, are delineated below.

This thesis proposed a parallelized architecture of AES-128 engine to enable an ultra-low energy operation by lowering the supply voltage while maintaining more than tens of *Mbps* throughput rate. Optimized 128-bit architecture and 16 S-boxes placed in the encryption datapath allow the parallel operation of 16 bytes of encryption data at supply voltage in subthreshold region to simultaneously achieve the required performance and energy efficiency. This resulted in $54.12pJ/encrypt.$ at $64.6Mbps$, a lower energy consumption with an order of magnitude higher throughput or a two orders energy reduction at an order reduced throughput rate compared with the state-of-the-art AES engines [5] and [24], respectively.

The proposed AES engine is, however, prone to the byte-oriented DPA attacks suggested in Chapter 3. This thesis proposed a DPA attack methodology based on a power model at registers in encryption datapath. The transition power of registers is estimated by using the byte-oriented reverse algorithm of AES with the knowledge of a ciphertext and the guess of a secret key. The DPA attack methodology could discover 12 key bytes among 16 key bytes within $20K$ encryption runs.

A new AES architecture is proposed in Chapter 4 to balance the Hamming weight of register input, and thus to protect our system from the DPA attacks on registers. Since adding a new encoding/decoding block on the critical path is not a desirable approach in performance perspective, one-hot code at the output of the decoder in S-box is directly used for the purpose of balancing the Hamming weight of encryption data. For that, register in the datapath is moved behind the decoder block in S-box to use the one-hot coded data as input to the register.

DPA attacks on the newly proposed AES engine could not discover any secret key even after $100K$ encryption runs. However, the improved security comes with the cost of IC performance and chip area. Since the register size is increased by $32\times$ compared to the previous design in Chapter 2, it caused $3\times$ area and $2\times$ power increase, as well as a reduction of maximum throughput by $2/3$. The majority of the power is consumed by the registers, since the increase of data-width leads to large interconnect capacitance and clock signal overhead. However, the power can be reduced by clock gating which can lower the switching capacitance on the clock network.

In summary, this thesis proposed an energy-efficient AES architecture with resistance to DPA attacks. Extensive parallelism makes it possible to run the system at low supply voltages while satisfying the throughput requirements. Parallelism, combined with voltage scaling, can simultaneously provide energy efficiency and high throughput rate. The system also incorporates DPA-protection designs, which mask the encryption data by balancing Hamming weight of input to the register in the datapath. The secured core has been subjected to $100K$ encryptions, $33\times$ more than the number of runs at which a secret key is discovered for the unprotected core, but none of its secret key have yet been disclosed. The trade-off of using such a secure design is an increase in area by three times, an increase in power by two times, and a reduction of maximum throughput by 33%. However, these overheads will be reduced further by clock gating for registers and an efficient control scheme of the system.

Bibliography

[1] D. Hwang, K. Tiri, A. Hodjat, B. Lai, S. Yang, and I. Verbauwhede, "Aes-based security coprocessor ic in 0.18um cmos with resistance to differential power analysis side-channel attacks," *IEEE J. Solid-State Circuits*, pp. 781–791, Apr. 2006.

[2] "Advanced encryption standard (aes), fips 197," *National Institue of Standards and Technology*, Nov. 2001.

[3] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Proc. Advances in Cryptology*, pp. 388–397, 1999.

[4] B. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, SEP 2005.

[5] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "Aes implementation on a grain of sand," *IEE Proceedings on Information Security*, pp. 13–20, Jun. 2005.

[6] A. Hodjat and I. Verbauwhede, "Area-throughput trade-offs for fully pipelined 30 to 70 gbits/s aes processors," *IEEE Trans. on Computers*, pp. 366–372, Apr. 2006.

[7] G. Bertoni, M. Macchetti, L. Negri, and P. Fragneto, "Power-efficient asic synthesis of cryptographic sboxes," *Proceedings of the 14th ACM Great Lakes Symposium on VLSI*, pp. 277–281, 2004.

[8] J. Goodman and A. P. Chandrakasan, "An energy-efficient reconfigurable public-key cryptography processor," *IEEE J. Solid-State Circuits*, pp. 1808–1820, Nov. 2001.

[9] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and performance testing of a 2.29-gb rijndael processor," *IEEE J. Solid-State Circuits*, pp. 569–572, Mar. 2003.

[10] D. Canright, "A very compact s-box for aes," *Cryptographic Hardware and Embedded Systems*, pp. 441–455, 2005.

[11] S. Tillich, M. Feldhofer, and T. Popp, "Area, delay and power characteristics of standard-cell implementations of the aes s-box," *J Sign Process Syst*, pp. 251–261, Jan. 2008.

[12] L. Benini, A. Macii, E. Macii, E. Omerbegovic, M. Poncino, and F. Pro, "Energy-aware design techniques for differential power analysis protection," *Proc. 40th Design Automation Conf.*, pp. 36–41, Jun. 2003.

[13] S. Yang, W. Wolf, N. VijayKrishnan, D. N. Serpanos, and Y. Xie, "Power attack resistant cryptosystem design: A dynamic voltage and frequency switching approach," *Proc. Design, Automation, and Test in Europe, vol.3*, pp. 64–69, 2005.

[14] K. Baddam and M. Zwolinsky, "Evaluation of dynamic voltage and frequency scaling as a differential power analysis countermeasure," *VLSI Design Conference (VLSID)*, pp. 854–862, 2007.

[15] P. Corsonello, S. Perri, and M. Margala, "An integrated countermeasure against differential power analysis for secure smart-cards," *International Symposium on Circuits and Systems (ISCAS)*, pp. 5611–5614, Sep. 2006.

[16] C. Tokunaga and D. Blaauw, "Secure aes engine with a local switched-capacitor current equalizer," *ISSCC Dig. Tech. Papers*, pp. 64–65, Feb. 2009.

[17] P. Rakers, L. Connel, T. Collins, and D. Russel, "Secure contactless smartcards asic with dpa protection," *IEEE J. Solid-State Circuits*, pp. 559–565, Mar. 2003.

[18] L. Wong, S. Hossain, A. TA, J. Edvinsson, D. Rivas, and H. Naas, "A very low-power cmos mixed-signal ic for implantable pacemaker applications," *IEEE J. Solid-State Circuits*, pp. 2446–2456, Dec. 2004.

[19] J. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Pervasive Comput.*, pp. 18–27, Mar. 2005.

[20] A. Chandrakasan, S. Sheng, and R. Brodersen, "Low-power cmos digital design," *IEEE J. Solid-State Circuits*, pp. 473–484, Apr. 1992.

[21] A. Wang and A. Chandrakasan, "A 180-mV subthreshold FFT processor using a minimum energy design methodology," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 310–319, Jan 2005.

[22] A. Chandrakasan, N. Verma, and D. Daly, "Ultralow-power electronics for biomedical applications," *Annual Review of Biomedical Engineering*, pp. 247–274, Aug. 2008.

[23] NIST, "Recommendation for block cipher modes of operation - methods and techniques," *Special Publication 800-38A 2001 ED*, Dec. 2001.

[24] I. Verbauwhede, P. Schaumont, and H. Kuo, "Design and performance testing of a 2.29-gb/s rijndael processor," *IEEE J. Solid-State Circuits*, pp. 569–572, Mar. 2003.

[25] S. Morioka and A. Satoh, "A 10-gbps full-aes crypto design with a twisted bdd s-box architecture," *IEEE Trans. on VLSI Systems*, pp. 686–691, Jul. 2004.

[26] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully attacking masked aes hardware implementations," *Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, pp. 157–171, 2005.