Power Profile Obfuscation using RRAMs to Counter DPA Attacks

Ganesh Chandrakantrao Khedkar

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
Power Profile Obfuscation using RRAMs to Counter DPA Attacks

by

Ganesh Chandrakantrao Khedkar

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Engineering

Supervised by

Dr. Dhireesha Kudithipudi
Department of Computer Engineering
Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
August 2013

Approved by:

Dr. Dhireesha Kudithipudi, Associate Professor
Thesis Advisor, Department of Computer Engineering

Dr. Marcin Łukowiak, Associate Professor
Thesis Co-Advisor, Department of Computer Engineering

Dr. Amlan Ganguly, Assistant Professor
Committee Member, Department of Computer Engineering
Thesis Release Permission Form

Rochester Institute of Technology
Kate Gleason College of Engineering

Title:
Power Profile Obfuscation using RRAMs to Counter DPA Attacks

I, Ganesh Chandrakantrao Khedkar, hereby grant permission to the Wallace Memorial Library to reproduce my thesis in whole or part.

Ganesh Chandrakantrao Khedkar

Date
Dedication

To my family, for their constant love and support
Acknowledgments

I would like to express my deepest appreciation to all countless faculty, staff and friends, who provided me the possibility to complete this work. A special gratitude I give to my thesis advisor, Dr. Dhireesha Kudithipudi for her continual guidance, suggestion and help to coordinate this work. I would like to thank Dr. Marcin Łukowiak for his precious assistance improving my understanding of Side Channel Attacks and also for serving as a committee member. I would also like to thank Dr. Amlan Ganguly for taking time out of his busy schedule to serve as a committee member.

Furthermore I would also like to acknowledge with much appreciation the crucial role of the Mr. Emilio Del Plato and Mr. Richard Tolleson who gave the permission to use all required servers to complete the lengthy simulation tasks. A special thanks goes to Xuan Tran, who helped me for the part of DPA attack. Last but not least, many thanks goes to Cory Merkel, Sundarraman Mohanram, Sam Skalicky, and Gorden Werner for time to time help and comments.
Abstract

Side channel attacks, such as Differential Power Analysis (DPA), denote a special class of attacks in which sensitive key information is unveiled through information extracted from the physical device executing a cryptographic algorithm. This information leakage, known as side channel information, occurs from computations in a non-ideal system composed of electronic devices such as transistors. Power dissipation is one classic side channel source, which relays information of the data being processed. DPA uses statistical analysis to identify data-dependent correlations in sets of power measurements.

Countermeasures against DPA focus on hiding or masking techniques at different levels of design abstraction and are typically associated with high power and area cost. Emerging technologies such as Resistive Random Access Memory (RRAM), offer unique opportunities to mitigate DPAs with their inherent memristor device characteristics such as variability in write time, ultra low power (0.1-3 pJ/bit), and high density (4F2).

In this research, an RRAM based architecture is proposed to mitigate the DPA attacks by obfuscating the power profile. Specifically, a dual RRAM based memory module masks the power dissipation of the actual transaction by accessing both the data and its complement from the memory in tandem. DPA attack resiliency for a 128-bit AES cryptoprocessor using RRAM and CMOS memory modules is compared against baseline CMOS only technology. In the proposed AES architecture, four single port RRAM memory units store the intermediate state of the encryption. The correlation between the state data and sets of power measurement is masked due to power dissipated from inverse data access on dual RRAM memory. A customized simulation framework is developed to design the attack scenarios using Synopsys and Cadence tool suites, along with a Hamming weight DPA attack module. The attack mounted on a baseline CMOS architecture is successful and the full key is recovered. However, DPA attacks mounted on the dual CMOS and RRAM based
AES cryptoprocessor yielded unsuccessful results with no keys recovered, demonstrating the resiliency of the proposed architecture against DPA attacks.
Contents

Dedication .................................................................................................................. iii
Acknowledgments ........................................................................................................ iv
Abstract ....................................................................................................................... v

1 Introduction and Background .................................................................................. 1
  1.1 Resistive Random Access Memory ................................................................. 1
  1.2 Advanced Encryption Standard Algorithm .................................................. 4
    1.2.1 SubBytes .................................................................................................. 5
    1.2.2 ShiftRows ............................................................................................... 6
    1.2.3 MixColumn ............................................................................................. 6
    1.2.4 AddRoundKey ......................................................................................... 7
  1.3 Side Channel Attacks ......................................................................................... 7
  1.4 Power Analysis Attack ...................................................................................... 8
    1.4.1 Simple Power Analysis ........................................................................... 10
    1.4.2 Differential Power Analysis ................................................................... 10
  1.5 Contributions .................................................................................................... 12

2 Related Work and Contributions ............................................................................ 14
  2.1 DPA vulnerability in AES algorithm ............................................................... 15
  2.2 DPA mitigations techniques ............................................................................ 16
    2.2.1 Masking .................................................................................................. 16
    2.2.2 Hiding .................................................................................................... 17
  2.3 Summary .......................................................................................................... 18

3 Hardware Architecture ............................................................................................ 20
  3.1 RRAM .............................................................................................................. 20
    3.1.1 RRAM and other memory technologies ................................................. 20
    3.1.2 RRAM Architecture .............................................................................. 21
  3.2 AES architecture .............................................................................................. 24
3.2.1 AES Encryption Unit .......................... 25
3.2.2 Memory Balancing Logic .................. 27

4 Simulation and Attack Framework .................. 31
  4.1 Simulation Framework ......................... 31
    4.1.1 Power extraction for CMOS implementation .......... 32
    4.1.2 RRAM power extraction .......................... 34
  4.2 Attack Framework ............................... 35
  4.3 Summary ........................................ 37

5 Result and Analysis ................................ 38
  5.1 Memristor ......................................... 38
  5.2 RRAM power dissipation ......................... 40
  5.3 Balance Logic and Inverse State Memory ............ 43
  5.4 DPA Attack Results ................................ 46
  5.5 Summary ........................................ 50

6 Conclusions and Future Work ...................... 51
  6.1 Conclusions ....................................... 51
  6.2 Future Work ...................................... 52

Bibliography ........................................ 54
List of Tables

2.1 Comparison chart depicting the cost incurred due to implementation of each countermeasure when compared with unprotected design. . . . . . . . 19

3.1 Qualitative Comparison between RRAM and other emerging and commercialized memory technologies . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21

4.1 AES design variations under consideration . . . . . . . . . . . . . . . . . . 31
List of Figures

1.1 High-level overview of RRAM block architecture.[48] .................. 3
1.2 Generic Representation for the AES Algorithm .......................... 5
1.3 The ShiftRows Transformations ........................................... 6
1.4 Types of unintended information leakage from cryptographic function implementation ......................................................... 8
1.5 Overview of the power analysis attack flow [32] ....................... 9
1.6 SPA Trace of entire DES algorithm [34] ................................. 11

2.1 Basic overview of Masking approach [57] ............................... 16
2.2 Architectural view of Multiprocessor balancing [6] .................. 18

3.1 Block level view of an NxM RRAM. ...................................... 22
3.2 Power switching using a single power device ............................ 24
3.3 Top view of the proposed AES hardware design ........................ 25
3.4 RTL representation of the AES Encryption Unit ........................ 26
3.5 AES Control Unit State Machine .......................................... 26
3.6 RTL representation of the State Memory with Regular State Memory Bank and Inverse State Memory Bank ......................... 29
3.7 RTL block diagram for the Balancing Logic with the snooper to engage both Regular Memory and Inverse Memory modules. .......... 29

4.1 Top Level Simulation Flow for Power Extraction ..................... 32
4.2 Synthesis flow to generate netlist and gate level simulation model .... 33
4.3 Power modeling flow for AES Encryption Unit. ....................... 34
4.4 Power extraction flow for RRAM state memory model. ............... 35
4.5 Top view of the attack module. ........................................... 36
4.6 Differential Power Analysis overview for a single byte attack ......... 36

5.1 The I-V curve produced by linear memristor model ................... 39
5.2 The I-V curve produced by linear memristor model with temperature variation ................................................................. 39
5.3 Generic representation of an MxN RRAM crossbar[32] ................. 40
5.4 Power dissipation when (a) Writing a one to the RRAM crossbar (b) Writing a zero to the RRAM crossbar (c) Reading across the RRAM crossbar [32].

5.5 Voltage variability analysis for the RRAM crossbar, when writing to 31st bit in the first row. [32].

5.6 Temperature variability analysis for the RRAM crossbar, when writing to 31st bit in the first row. [32].

5.7 Waveform for read/write cycle on regular state memory and inverse state memory.

5.8 Power dissipation for 1 round of AES for (a) Regular CMOS Memory (b) Inverse CMOS State Memory (c) Total CMOS State Memory.

5.9 Power dissipation for 1 round of AES for (a) Regular RRAM Memory (b) Inverse RRAM State Memory (c) Total RRAM State Memory.

5.10 Power comparison of Single/ Dual CMOS and RRAM memory implementation.

5.11 (a) Differential trace for AES with Regular CMOS state memory with 10,000 power traces, all 16 key bytes (b) Confidence ratio for AES with Regular CMOS state memory with 10,000 power traces.

5.12 (a) Differential trace for AES with Regular RRAM state memory with 10,000 power traces, all 16 key bytes (b) Confidence ratio for AES with Regular RRAM state memory with 10,000 power traces.

5.13 (a) Differential trace for AES with inverse CMOS state memory with 40,000 power traces, all 16 key bytes (b) Confidence ratio for AES with inverse CMOS state memory with 40,000 power traces.

5.14 (a) Differential trace for AES with inverse RRAM state memory with 40,000 power traces, all 16 key bytes (b) Confidence ratio for AES with inverse RRAM state memory with 40,000 power traces.
Chapter 1

Introduction and Background

Modern security systems use cryptographic algorithms to provide confidentiality, integrity and authentication of data. These cryptographic algorithms rely on mathematically complex and difficult operations to enhance the security [61]. Research in cryptanalytic techniques has demonstrated that secret information can be extracted by exploiting weaknesses of a cryptographic algorithm’s implementation at hardware/software level [3]. As the conventional hardware implementations are based on CMOS technology, efficacy of countermeasures are confined by the technology limitations. These limitation include high static and dynamic power dissipation. Several emerging technologies such as memristive devices are being commercialized and adopted in the IC design market, it is equally important to understand the effect of these devices on the overall system to improve resiliency against cryptanalytic techniques.

1.1 Resistive Random Access Memory

The Resistive Random Access Memory (RRAM) is a crossbar resistive memory array, in which the storage elements are two-terminal resistive switching element, known as memristor. Memristor is a two terminal, passive circuit device that imposes a non-linear relationship between electrical charge and magnetic flux linkage [13, 15]. From a behavioral point of view, these devices are characterized by a pinched hysteresis current-voltage relationship [14], indicating that their instantaneous resistance depends on the history of applied terminal voltages. The data is stored in the form of high resistance $R_{off}$ and low resistance state
$R_{on}$. The memristor’s resistance states can be read by applying a small non-destructive voltage yielding a non-volatile memory behavior.

Unlike RRAM, many emerging non-volatile technologies such as Phase Change Memory (PCM), Spin-torque transfer random access memory (STTRAM), are being actively researched [66, 78]. Compared to the conventional CMOS memories such as SRAM, these emerging technologies are non-volatile, requiring a little/zero power to maintain the stored state. Memristor based RRAM is a viable technology for future computing with high density, ultra-low static power consumption (limited by CMOS leakage power), low dynamic power consumption ($\approx 0.1-3$ pJ/bit), high retention and endurance [77]. Semiconductor companies such as HP, Toshiba are exploring the possibility of replacing SRAM on-chip memory with the emerging memories like RRAM in the near future [75].

Memristor device modeling is a challenging task due to the diversity of physical implementations and proposed switching mechanisms [76, 77]. A common method, especially in the case of transition metal oxide thin film implementations, is to treat the switching region of the device as two variable resistors in series, where the total resistance is given by $R_m = xR_{on} + (1 - x)R_{off}$, where $x$ is a state variable that ranges from 0 to 1 [48, 69]. In the case of TiO$_2$ switching regions, $x$ is the fraction of the TiO$_{2-x}$ phase present in the switching region. Analytically, this model can be described by

$$R_m(t) = R_{m0} \sqrt{1 - \frac{2\eta \Delta R \phi(t)}{D^2 R_{m0}^2} \mu R_{on}},$$

(1.1)

where $R_{m0}$ is the maximum resistance ($R_{m0} \approx R_{off}$), $D$ is the film thickness, $\eta$ ($\pm 1$) is the polarity of the applied voltage, $Q_0$ the charge required for $x$ to change from 0 to 1, $\Delta R = R_{off} - R_{on}$, $\phi(t)$ is the time integral of the applied voltage, and $\mu$ is the mobility of defects or ion species. Based on equation (1.1) it is clear that the write time, or latency, necessary to achieve target high and low resistance states will vary from device to device. Distributions of film thicknesses, mobilities, and other model parameters effect the write
A generalized illustration of a CMOS/RRAM architecture is shown in Figure 1.1 [48]. The $N \times M$ crossbar array of thin-film memristors is used as the storage medium. Crossbar circuits offer high density and addressability [74]. The gray area represents CMOS/nano interface. All the components outside the gray box are implemented using CMOS, everything inside the box is using nano-CMOS paradigms [47]. A single memristor from the crossbar can be accessed via row and column multiplexers based on the address provided. A read/write control circuit is used to apply a read or (positive or negative) write voltage depending upon value of $\bar{rw}$. RRAM block is isolated using two tri-state buffer and enable signal. Note that this is a bit-addressable block, so we combine $b$ of these blocks in parallel to address a $b$-bit word, striping the word across multiple blocks.
1.2 Advanced Encryption Standard Algorithm

Advanced Encryption Standard (AES) algorithm is the standard for encryption, approved by National Institute of Technology and Standard (NIST) in 2001 [1]. It is a symmetric block cipher that processes 128-bit data blocks and can operate on keys with length of 128, 192, or 256 bits. The number of rounds depends on the key length.

The different transformations of the algorithm architecture operates on the intermediate data blocks, known as State. The State consists of 16 bytes arranged as a rectangular array of four rows and four columns. As a symmetric key encryption algorithm, the identical key is used for the encryption and decryption of data. The equation (1.2) shows the order followed to access data during transformations.

\[
\text{State} = \begin{pmatrix}
00 & 04 & 08 & 0C \\
01 & 05 & 09 & 0D \\
02 & 06 & 09 & 0E \\
03 & 07 & 0A & 0F \\
\end{pmatrix}
\]  

(1.2)

There are several rounds of transformation, dictated by the key length. This number could be 10, 12 and 14 for the key length of 128, 192 and 256 bit respectively. Figure 1.2 shows the generic representation of the AES algorithm. Each round consists of the max four transformations, namely SubBytes, ShiftRows, MixColumns and AddRoundKey as shown in Figure 1.2. The encryption cycle always begins with AddRoundKey transformation and continues with internal rounds with all four transformations. The final round consists of SubBytes, ShiftRows and AddRoundKey transformations only. Figure 1.2 also shows the key expansion unit which generates the key for every round, from the primary key. When all the rounds of transformation are performed, the cipher data of same block size as of input plain data is generated.
1.2.1 SubBytes

The SubByte transformation is the only nonlinear transformation within AES algorithm. The byte substitution operates independently on each byte of the state using a substitution box (S-box). This table is constructed using two transformations: multiplicative inverse and affine transformation. First, the multiplicative inverse in the finite field of GF ($2^8$) is taken, and then an affine transformation over GF(2) is applied. The SubByte can be implemented as simple as Lookup Table (LUT) or can be calculated dynamically during the transformation. The affine transformation for the encryption process is defined as

$$\hat{S}_o = S_i \oplus S_{(i+4)mod8} \oplus S_{(i+5)mod8} \oplus S_{(i+6)mod8} \oplus S_{(i+7)mod8} \oplus C(i) \quad (1.3)$$
where $S_i$ is the $i^{th}$ bit of input byte and $C(i)$ is the $i^{th}$ bit of the byte constant $C$ with the value $C = [01100011]$, as specified in the algorithm [1].

1.2.2 ShiftRows

The rows of the state are cyclically shifted over different offsets. This is done according to the equation (1.4) [10], as shown in below

$$S_{r,c} = S_{r,(c + shift(r,N))modN} \quad (1.4)$$

As a result, the first row is unchanged, while the second, third and fourth row are cyclically shifted to the left by one, two and three respectively as shown in Figure 1.3. This transformation adds the diffusion property to each round transformation to confuse the relationship between the plaintext input and ciphertext output.

1.2.3 MixColumn

The MixColumn transforms maps each of the input state to a new column in the output state. Each input column is considered as a polynomial over $GF(2^8)$ and multiplied with the constant polynomial as shown in equation (1.5a)

![Figure 1.3: The ShiftRows Transformations](image-url)
\[ p(x) = 03x^3 + 01x^2 + 01x + 02 \mod (x^4 + 1) \quad (1.5a) \]

\[ \hat{S}(x) = p(x) \oplus s(x) \quad (1.5b) \]

### 1.2.4 AddRoundKey

At the end of every round, a round key is added to the data using a simple bitwise XOR operation. The actual key is used only at the beginning of the AES encryption before Round 1. During internal rounds AddRoundKey transformation Round keys are derived from the key schedule using the initial cipher key using Rijndael key expansion algorithm. The equation (1.6) illustrates the each round’s subkey computation.

\[
W[i] = \begin{cases} 
  w[i - 4] \oplus w[i - 1] & \text{if } i \mod 4 \neq 0 \\
  w[i - 4] \oplus \text{SubByte}(\text{RotWord}(w[i - 1])) \oplus \text{Rcon}[i] & \text{if } i \mod 4 = 0
\end{cases}
\]

(1.6)

Where \( w[i] \) is the expanded key. The \( \text{RotWord}() \) is a simple cyclic permutation of a word change \([a_0, a_1, a_2, a_3]\) to \([a_1, a_2, a_3, a_0]\). The \( \text{Rcon}(i) \) is the exponentiation of 2 performed in Rijndael’s finite field in polynomial form as shown below.

\[
\text{Rcon}[i] = x^{(i-1)} \mod x^8 + x^4 + x^3 + x + 1
\]

(1.7)

And SubByte will apply S-box value in SubByte transformation to each of key byte.

### 1.3 Side Channel Attacks

The cryptographic algorithms are usually strong against mathematical attacks. The only way to unlock the secret key is to try all possible combinations. A cipher is said to be secure if larger the number of required combinations such that a complete search becomes impossible. For instance, RSA-2048 bits can be used at least till the year 2030 before the expected computing power to do integer factorization is available [70].
Traditional cryptanalysis views a cryptographic algorithm as a black box operation that transforms the plaintext into ciphertext using a secret key and could attempt to exploit the algorithm by analyzing imperfections in its mathematical structure [58]. However, the cryptographic algorithm has to be implemented on a physical device, which will leak additional information related to internal operation through unintended inputs and outputs known as side channels [34]. Figure 1.4 shows such unintended information leakage sources in the form of power, timing, electromagnetic induction etc.

![Figure 1.4: Types of unintended information leakage from cryptographic function implementation](image)

Side Channel Attacks (SCA) attempt to exploit these side channels in order to extract secret information from a cryptographic function implementation [2]. Depending upon the targeted side channel information, attacks can be based on power [34], timing [20], electromagnetic radiations [59].

### 1.4 Power Analysis Attack

In 1999, Kocher showed that instantaneous power consumption of a cryptographic system can be related to secret data within the system [34]. This kind of attack can be mounted in a non-invasive manner using relatively cheap and easily obtained measurement equipment.
These attacks have been implemented against hundreds of devices, including implementations in ASICs, FPGAs, and softwares [35, 43]. The target devices ranges from tiny single-purpose chips to complex devices whose power measurements are noisy and obfuscated by unpredictable parallel operations.

The goal of power analysis is to identify a relationship between the changing internal state of the cryptographic device with respect to instantaneous power consumption. A properly identified intermediate step related to sensitive information will improve the chances of successful extraction [3]. For instance, in an AES implementation, output of the first transformation in the first round is dependent only on a known plaintext input and the secret key. As a result, if a relationship can be found between this intermediate state and circuit’s power consumption, then it could be possible to extract the secret key.

Figure 1.5 shows the overview of the power analysis attack flow. A cryptographic function takes a plaintext and compute an intermediate cipher using a secret key. Generally, a sensitive data will represent the data stored in intermediate registers after computation phase. The associated computational leakage will be observed. This sensitive data will be attempted to predict using different key guess in attack model. In order to correlate the sensitive data with each power trace, a leakage model must be constructed to describe the power consumption of the device. The Hamming Weight and Hamming Distance are some of the most typically used models to describe the leakage of these circuit elements [24].

Figure 1.5: Overview of the power analysis attack flow [32]
The Hamming Weight model is the straightforward representation of a device’s power consumption. It is based on the assumption that power consumption is proportional to the number of bits switched on. Though, power consumption using this model is weakly described, but it is useful when little knowledge about the underlying hardware is available [24]. On the other hand, Hamming Distance assumes that, number of bit transition is proportional to power consumption of the circuit [50].

1.4.1 Simple Power Analysis

Simple Power Analysis (SPA) is a basic power analysis technique which involves direct interpretation of power consumption measurements collected during a cryptographic operation. Number of power traces are observed to identify apparent characteristics that may be useful to reveal information about sensitive operations [34]. This apparent features are SPA weakness caused by operations performed based on key bits. This visible variations results mainly from differences in the power consumption of different operations. Thus, SPA is more helpful in determining the overall structure of an algorithm, not specifically in determining its key.

For instance, Kocher et.al. [34], completed SPA on the Data Encryption Standard (DES) algorithm in order to determine when each round occurs. Figure 1.6 clearly shows all 16 rounds of the DES algorithm. As identification of individual rounds within an encryption operation helps to focus on the samples more likely to correlate thereby improving chances of success of the attack [22]. This makes SPA useful tool for more sophisticated attacks. As, SPA is limited to visual inspection, it is susceptible to noise or measurement errors.

1.4.2 Differential Power Analysis

Differential Power Analysis (DPA) is significantly more powerful statistical analysis capable of detecting smaller scale variations to extract information correlated to the secret keys
The main advantage of DPA over SPA attacks is that no knowledge of the cryptographic system and circuit is necessary [34]. However, DPA requires many traces of the algorithm to work properly, often requiring several thousand traces.

A DPA attack attempts to guess secret information by establishing a relationship between the secret information and the instantaneous power consumption of the device. To achieve this, a state within the system that is dependent upon both the secret and some known quantity is identified. This state is referred to as the sensitive value, and may be estimated by guessing the secret value by applying some known input. If the sensitive value is correlated to the circuit’s power consumption, then a correct guess of the secret will correlated to the power consumption [34, 35].

For instance, to perform a DPA attack on a block cipher, a selection function $D(P, b, K_s)$ is defined for the plaintext (or ciphertext) $P$, for target bits $b$ of sensitive state with key guess $K_s$. The $n$ encryption operations observed will have $n$ corresponding power traces of $k$ samples each, labeled as $T_{1..n}[1..k]$, along with related plaintext (or ciphertext) $P_{1..n}$.

The two average traces are computed, $A_0$ and $A_1$. The average trace $A_1$ is computed as the average of the power traces for which $D(P, b, K_s)$ produces one, and $A_0$ as the average of the power traces for which $D(P, b, K_s)$ produces zero. Then distance between these average traces is calculated producing a differential trace $\Delta_D[1..k]$ [34, 35].
\[
\Delta_D[j] = \frac{\sum_{i=1}^{n} D(P_i, b, K_s)T_i[j]}{\sum_{i=1}^{n} D(P_i, b, K_s)} - \frac{\sum_{i=1}^{n} (1 - D(P_i, b, K_s))T_i[j]}{\sum_{i=1}^{n} (1 - D(P_i, b, K_s))} \\
\approx 2\left(\frac{\sum_{i=1}^{n} D(P_i, b, K_s)T_i[j]}{\sum_{i=1}^{n} D(P_i, b, K_s)} - \frac{\sum_{i=1}^{n} T_i[j]}{n}\right)
\] (1.8)

For incorrect guess of \(K_s\), the bit computed using \(D\) will differ from the actual target bit for about half of the plaintext \(P_i\). Thus, making selection function \(D(P_i, b, K_s)\) completely uncorrelated from the actual computation. As the power traces are divided into two subsets \(A_1\) and \(A_0\) randomly, the difference between the two average traces should approach zero as the number of power traces increases [34, 35] as shown below.

\[
\lim_{n \to \infty} \Delta_D[j] \approx 0
\] (1.9)

However, computed value of \(D(P_i, b, K_s)\) will be equal to actual value of target bit \(b\) with probability 1 if the \(K_s\) is correct. As a result, \(\Delta_D[j]\) will approach to the actual power consumption as \(n \to \infty\).

### 1.5 Contributions

The primary goal of this research is to explore possible role of emerging technologies such as RRAM in improving resiliency against DPAs in order to develop secure hardware designs. This work expands upon an existing attack methodology developed for power analysis attacks on a simulated implementation of the AES block cipher [65]. In order to accomplish this goal,

- The existing RRAM architecture [47] has been altered to make it compatible to existing AES CMOS implementation.
- AES design also been modified to support RRAM and in order to provide a fair and consistent comparison.
• Two architecture with CMOS memory and RRAM are developed and studied in a simulated environment using Synopsys and Cadence tools.

• A DPA attack module is implemented and resiliency with and without countermeasure is been tested.

The rest of the document is organized as follows: the related work of this thesis are discussed in Chapter 2. Chapter 3 discusses the proposed architecture in detail. Simulation and attack framework is discussed in Chapter 4. The results are discussed and analyzed in Chapter 5. The conclusion and future work are presented in Chapter 7.
Chapter 2

Related Work and Contributions

Side channel attacks attempt to reveal the secure data through information extracted from physical implementation of a cryptographic function [49]. The device leaks information through unintentional environmental interactions in the form of power, timing, electromagnetic radiations. Such unintentional information leakage occurs because computations occur on a non-ideal system, composed of electronic devices such as transistors, wires, power supplies, memory, and peripherals. Each of these component have characteristics that vary with the instructions and data being processed. When this variance (side channel information) is measurable, it becomes access point to an otherwise secure system.

One such side channel information is the instantaneous power consumption of a system. Power attacks such as DPAs use power traces to correlate with secure data processed by a cryptographic algorithm implementation [34, 49]. Several successful attacks to extract secure key from a cryptographic system includes memory encryption/decryption schemes [17] and power consumption randomization [19]. Successful attacks on embedded systems such as Virtex-II FPGA [51], Virtex-4/5 FPGA [52] bitstream encryption, and Atmel Cryptomemory non-volatile memory [9] proves that these attacks are practical and lethal. Therefore, it is critical to evaluate countermeasures that hamper power attacks. Countermeasures are mainly focused on achieving a disconnect between the secured data/operation executed and its power consumption [55].

In this chapter, AES algorithms vulnerability towards DPA is discussed first. In subsequent sections, different countermeasures are discussed which is followed by a summary.
2.1 DPA vulnerability in AES algorithm

The encryption cycle of AES algorithm, as discussed in section 1.2, processes plaintext using the round keys derived from cipher key in multiple rounds. At any point within datapath where the state (derived from the plaintext) and the round key (derived from cipher key) enter a logic gate, the dynamic power consumption of this gate depends on both the cipher-key and plaintext. If this information sampled, DPA attack can be mounted successfully. In essence, the output of any transformation in AES could be considered for an attack [57]. Thus, the DPA vulnerability of the intermediate results is greatly dependent on the specific implementation of the datapath.

The ShiftRows function is a simple bit permutation, and is realized using only wiring for 128-bit parallel datapaths and hence, is not suitable for DPA attacks. Any non-linear function increases the efficiency of the statistical attacks such as a DPA [35, 49]. Thus, an efficient attack would target the output of the SubBytes transformation.

The MixColumn transformation is also a non-linear operation and is defined for 32 bits. Attacking this output will require key hypothesis of $2^{32}$. Thus, making it a costlier, in terms of required time and memory, compared to alternatives AES transformations. Finally, the AddRoundKey transformation involves round key XORed with intermediate state. This function is directly related to round key, but it is less efficient to attack than SubByte as operation is linear.

A successful DPA attack on the AES implementation is presented in [44, 54]. Both designs use 128-bit data path and one entire round transformation is performed in a single cycle. Design attacked in [54], is a unprotected implementation of the AES. This attack required 64,000 measurements for a successful extraction of the entire AES key. On the other hand, design in [44] is protected implementation of AES using masking and required 130,000 measurements for successful DPA attack.
2.2 DPA mitigations techniques

DPA Countermeasures attempts to obscure the relationship between the power profile and the data/operation executed. To mitigate DPA, there are two broad categories of countermeasures: masking and hiding. In this thesis, we base our solution on hiding.

2.2.1 Masking

Masking obscures the actual computation information by either adding or multiplying random number’s to algorithm’s input and intermediate output values, making it difficult to build correlation between power consumption and different cryptographic operations [18]. This involves concealing every intermediate value $v$ by a random mask $m$ such that $v_m = v \ast m$ [43] as shown in Figure 2.1. In this way all the intermediate data appears to be nonaligned within the cryptographic system. There are mainly two types of masking: boolean and arithmetic [18]. Boolean masks work by exclusive-oring the data value with the mask given as $v_m = v \oplus m$. Arithmetic masks work by either adding or multiplying the mask to the data using modular addition or multiplication.

![Figure 2.1: Basic overview of Masking approach [57]](image)

Once mask is inserted into the intermediate values, it must be eventually removed as shown in Figure 2.1. Masking the S-Box is a difficult task. This difficulty is associated with non-linearity [67]. Different method have been proposed to achieve masking of S-BOX in [63, 67]. However, implementation of these measures is highly algorithm-specific [18, 28, 46, 57]. Though, this method achieves the goal of providing disconnect between
the intermediate data and the power consumption, but are not most effective against higher order DPA [63].

2.2.2 Hiding

The goal of hiding technique is to level the system’s temporal power profile. In other words, these techniques cause the instantaneous power consumption to be approximately constant, regardless of the cryptographic operation being performed [17, 56, 64]. This can be achieved either by randomizing the power consumption over a given period [28] or by flattening the power consumption so that the power used by device is equal over all operations [53]. Some of the techniques used are randomization [28, 39, 45], dual rail logic [25, 56, 62, 72], current flattening [8, 36, 53], bit-balancing [6, 7].

The main intent behind the randomization is to prevent power trace alignment, therefore thwarting the DPA. This can be achieved by either inserting random access pattern [45] or randomizing the execution pattern of the algorithm [28]. As in [45], random register renaming is used to disassociate the power consumption from its re-accessing. This methodology requires a random number generator and a bigger register file to improve the probability of the desired randomness. In case of [28], instructions are randomly executed in run time to generate random power profile. Both techniques are highly algorithm dependent and require pseudo-randomness. An efficient pseudo-random number generator should have minimal effect on the size and speed of the circuit.

One of the well-known hiding countermeasure is to flatten the power signature of all components, within the circuit’s hardware, independent of data value. An effective method is performed at the cell level using Dual-Rail Precharge (DRP) logic blocks [56]. The concept behind the DRP logic is to create logic cells that make power consumption constant during each clock cycle. Every input and output into cell is paired with its inverse and therefore a constant balance of ‘0’ s and ‘1’ s is maintained at any given point of time [25, 56]. More abstract level method is implemented in [71], where a co-processor uses constant
power dissipation logic for any bit transition. Though leveling power profile proves to be effective, it will double the size of the circuit, increase the power required by 2-3X [6, 28], and finally slows the circuit down by at least 50% due to additional stages [55].

Similar to flattening, bit-balancing attempts to balance bit-flips for every intermediate value. This balancing is achieved by executing similar algorithm processing inverted data in tandem [7] as shown in Figure 2.2. The Hamming Weight for all the intermediate values will be equal when normal and inverted data execution is considered. Although, the architecture should be combined such that footprints related to execution in separate cores cannot be inferred [6]. It is evident that bit-balancing doubles the area and power used by the circuit. However, the overhead circuitry doesn’t change the speed of the algorithm distinctly [6, 7].

2.3 Summary

Two main DPA countermeasures that are effective for AES are hiding and masking. Hiding attempts to camouflage the key message with random (or additive) noise generated from the underlying circuit/logic or use other means such as timing variance to disassociate the information and the power signal [23]. Masking hides the actual computation information by either adding or multiplying random numbers to the data during the encryption process.

![Figure 2.2: Architectural view of Multiprocessor balancing [6]](image-url)
Each of these countermeasures can be applied at multiple levels of integration - algorithm level, logic level, and/or circuit level. Countermeasures such as WDDL [72], current flattening [53], and multiplicative masking [73], have been adopted at different integration levels, albeit at a high cost in performance, power consumption, and area. Table 2.1 compares cost in design parameters introduced by implementing a few of such countermeasures.

<table>
<thead>
<tr>
<th></th>
<th>Masking</th>
<th>Randomization</th>
<th>Dual Rail Logic</th>
<th>Bit Flip</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Logic Level</td>
<td>Architectural/Algorithm Level</td>
<td>Circuit/Logic Level</td>
<td>Architectural level</td>
</tr>
<tr>
<td><strong>Area</strong></td>
<td>1.6X</td>
<td>&lt;1.3X</td>
<td>2X</td>
<td>2X</td>
</tr>
<tr>
<td><strong>Time</strong></td>
<td>1.3X - 1.4X</td>
<td>1.2X - 1.5X</td>
<td>1.5X - 1.7X</td>
<td>1.1X</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>1.4X-1.6X</td>
<td>1.15X-1.3X</td>
<td>2X-3X</td>
<td>2X</td>
</tr>
<tr>
<td><strong>Traces</strong></td>
<td>20,000 [21]</td>
<td>30,000 [ ]</td>
<td>35,000 [ ]</td>
<td>40,000 [ ]</td>
</tr>
</tbody>
</table>

Table 2.1: Comparison chart depicting the cost incurred due to implementation of each countermeasure when compared with unprotected design.

Few of these countermeasures can be applied to only private-key crypto systems and few of them are suitable for public-key crypto systems. Furthermore, all of these countermeasures have been explored as conventional CMOS implementations.

These countermeasures are highly algorithm dependent and require significant design time effort. The effectiveness of any countermeasure is measured by the number of power traces needed to extract a key. The goal is then to create a countermeasure that is a) algorithm independent, b) consumes less power, c) has less effect on execution time, and d) requires minimum design time efforts.

Since several emerging technologies such as memristive devices are being commercialized and adopted in the IC design market, it is equally important to understand the effect of these devices on the overall system and the resiliency to SCAs.
Chapter 3

Hardware Architecture

An AES core module with S-BOX implementation has been used for power analysis attack in this work. At abstract level, the design consists of AES transformation modules along with RRAM based state memory.

This chapter starts with discussing the RRAM and its internal architectural details. In subsequent section, the AES design has been discussed with proposed method to counter the DPA.

3.1 RRAM

In this section characteristics of RRAM are compared with other emerging and commercial memory technologies. In addition, the architecture of the RRAM used in this work is also discussed.

3.1.1 RRAM and other memory technologies

Continued technology migration in to nanometer domain, has initiated research in several hybrid CMOS/nano logic and memory architectures, each of these technology targets high density and low power consumption with little or no performance degradation [16]. The discovery of memristance in nanoscale metal-oxide devices [13, 68] has boosted the work in nanoscale architectures to implement nonconventional logic to improve memory and logic density [30]. Table 3.1 [42, 77] shows a qualitative comparison between RRAM and several emerging and commercialized technologies. RRAM provides high storage capacity
<table>
<thead>
<tr>
<th>Property</th>
<th>RRAM</th>
<th>PCM</th>
<th>STTRAM</th>
<th>SRAM</th>
<th>DRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read Time (ns)</td>
<td>&lt;10</td>
<td>10-50</td>
<td>10-35</td>
<td>0.1-0.3</td>
<td>10</td>
</tr>
<tr>
<td>Write Time (ns)</td>
<td>10</td>
<td>50-500</td>
<td>10-90</td>
<td>01-0.3</td>
<td>10</td>
</tr>
<tr>
<td>Reciprocal Density ((F^2))</td>
<td>&lt;4</td>
<td>4-16</td>
<td>20-60</td>
<td>140</td>
<td>6-12</td>
</tr>
<tr>
<td>Energy per bit (pJ)</td>
<td>0.1 - 3</td>
<td>2.25</td>
<td>0.1 -2.5</td>
<td>0.0005</td>
<td>0.005</td>
</tr>
<tr>
<td>Non-volatility</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y/N</td>
<td>N</td>
</tr>
<tr>
<td>Multi-level capability</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Unknown</td>
<td>Unknown</td>
</tr>
<tr>
<td>Endurance</td>
<td>10^12</td>
<td>10^9</td>
<td>10^15</td>
<td>&gt;10^16</td>
<td>&gt;10^16</td>
</tr>
</tbody>
</table>

Table 3.1: Qualitative Comparison between RRAM and other emerging and commercialized memory technologies

due to smaller cell size. In addition, RRAM offers low operating voltages and multi-level cell storage. A 40 nm 3-bit/cell and 2-bit/cell RRAM operation was demonstrated in [12]. The RRAM has very good write and read speed compare to other emerging technology but lower than conventional SRAMs. Although RRAM has high leakage power as compare to PCM and STTRAM, dynamic power consumption is very low. RRAM can withstand the temperature up to 200° C [38]. However, RRAM has the problem of limited endurance.

RRAM suffers from the problem known as sneak path [40]. Previous work on resistive memories with bidirectional resistive switches employed diodes and transistors to eliminate sneak paths reduce leakage current [26, 27, 37]. More sophisticated circuit level techniques has been proposed to further suppression of sneak path in [42, 31]. Despite the above challenges, the advantages such as low static power, low dynamic power compared to conventional CMOS memories makes a power efficient replacement in this work.

### 3.1.2 RRAM Architecture

A RRAM is the bit-addressable memory as shown in Figure 3.1 and is based on [47]. CMOS/memristor RRAM architecture combines memristor crossbar circuits with additional CMOS circuitry to yield high-density non-volatile RRAM. Figure 3.1 depicts generalize CMOS/memristor hybrid RRAM architecture. In the top-right corner, an NxM crossbar array is used as the physical storage medium. Each memristor stores a single bit of data in the form of a high or low resistance [33]. The crossbar array is fabricated as
back-end CMOS process. The purple color square around the crossbar array represents the CMOS/nano interface. Multiplexers are used to select data from a specific row or column depending on the address given. A single memristor is isolated from the crossbar with the help of address decoder. A read/write control circuit applies a read or write voltage depending on the value of $\bar{rw}$. Two tristate buffers and an enable signal are used to isolate the RRAM block when it is not being used.

**Read Operation**

In the read operation, $en$ should be high and $\bar{rw}$ should be low. This selects the read voltage, $v_r$, to be applied to the positive terminal of the memristor at the row and column specified by the address. The read voltage is small enough such that it doesn’t disturb the state of the addressed memristor. The resulting voltage across $R_{pd, col}$ is compared to a reference.
voltage. The reference voltage is given by the voltage division

\[ v_{ref} = v_{row} \times \frac{R_{pd, col}}{R_{ref,i} + R_{pd, col}} \]  \hspace{1cm} (3.1)

where \( v_{row} \) is the voltage applied to the selected crossbar row, and \( R_{ref,i} \) is either \( R_{ref,r} \), \( R_{ref,w0} \), or \( R_{ref,w1} \), depending upon operation.

**Write Operation**

Write operation is the most power consuming operation. The rate of change in the memristance depends on the voltage applied, and time [68]. There are two regions of operation in case of memristor, linear and nonlinear. The linear mode of operation yields very low energy metrics with a larger write time penalty while the nonlinear mode provides for much faster speed but with higher power consumption [42]. In this work, the linear mode of the operation is used as it is necessary to have low power dissipation and minimum difference of power dissipation as compare to read operation.

The base design proposed in [47], requires two different voltage sources depending on the data to be written. A positive voltage is required if the data signal is high, which forces the memristor in low resistance state. A negative write voltage ensures a high resistance state, when data signal is low.

To support multiple voltages in CMOS circuitry a DC-DC converter is required [11]. The current architectural optimization in CMOS technology has enabled supply voltage in the order of 1V. This constrains the DC-DC circuitry to be highly efficient, low voltage and low-current [11, 29].

In this research, a simple approach has been used to eliminate the requirement of two different power supply. Four control switches has been connected as shown in Figure 3.2. Any two switches will be closed during each operation. To write ‘1’ in the memristor, a positive voltage needs to be applied at \( V_m \) with respect to \( V_n \). To ensure this SW_P1 and SW_P2 are enabled which enables the current flow for positive terminal (+) to negative
Figure 3.2: Power switching using a single power device

terminal (−), as shown by red line. If data to be written is '0', then a negative voltage needs to be applied at $V_m$ with respect to $V_n$. Thus, effectively enabling current in reverse direction. This is achieved by enabling SW.N1 and SW.N2 during data '0' writing, and current direction is depicted by blue line in Figure 3.2.

### 3.2 AES architecture

The AES implementation used in this research is based on [65], designed to encrypt with a 128-bit cipher key. This design is constructed in a structural manner. Figure 3.3 depicts the top view of the AES implementation under consideration. It consists of two main blocks, 1) AES Encryption Unit (AEU) and 2) Memory Balancing Logic.

The AES encryption unit performs transformation at byte level which uses the state memory to store the intermediate state. The balancing logic block generates balancing access pattern by snooping on control bus between AEU and state memory.
3.2.1 AES Encryption Unit

This is a single entity which acts as a execution and control unit for the AES design. The encryption unit has all the execution units for the four transformations of AES: AddRoundKey, SubByte, ShiftRows, and MixColumns.

Figure 3.4 illustrates the RTL representation of the AES Encryption Unit. The execution units for the four transformations of AES are lined up and grouped together along with a control unit. The SubByte unit performs the non-linear inversion in the Galois Field $GF(2^8)$. This is implemented as a look-up table. The ShiftRows transformation is implemented as direct connection, as it changes only the location and not the byte values of the state values during transformation. The control unit reorders the byte during this step of the encryption. The bytes are simply read from a location and then written to a different row address in the State memory.

The MixColumns requires access to a byte in all four rows simultaneously. This is the driving factor behind having separate memory blocks for each row in the state matrix. AddRoundKey is a simple XOR operation between an input byte of the state and a byte of the cipher key or the round key. The round keys are precomputed and stored in Key
Memory Block.

The AEU module also has a simple state machine to drive the datapath using AES control unit. This control state machine is shown in Figure 3.5. The control unit orchestrates the datapath blocks to process one byte of the state matrix every two cycles. One cycle to fetch the data from state memory and second to process the data. Therefore, each AES transformation requires 32 clock cycles.

Once a valid input is received, 16 bytes of plaintext will be loaded in to state memory.
This is followed by initial operation AddRoundKey. For every other clock cycle, each byte will be fetched during AddRoundKey R state and XORed with cipher key in AddRoundKey state. Since the operation is independent between bytes in the state, the output byte is stored by replacing the input byte. SubByte operation is also performed in same fashion. The ShiftRows operation changes the location of the state data of each row of state matrix. This is achieved by relocating bytes from one bank to another bank. During the MixColumns transformation, the four bytes of each column of the state matrix are combined using an invertible linear transformation. This single step will be performed over 8 clock cycles. During which each input byte will affect all four output bytes. Hence, the output bytes are stored in different memory bank in the state memory.

MixColumns transformation is not performed during final round of encryption cycle. So, control is transferred from ShiftRows to AddRoundKey. Finally, the ciphertext is retrieved and stored one byte at a time, requiring an additional 16 cycles. The total encryption requires 1344 clock cycles, as summarized in equation 3.2, where $transf\_cycle$ represents the number of transformation multiplied by clock cycles per transformation.

\[
Total\_cycles = Load\_PT + AddRoundKey + rounds \times transf\_cycle + Store\_CT \quad (3.2)
\]

\[
Total\_cycles = 16 + 32 + 10 \times (4 \times 32) + 16 \quad (3.3)
\]

\[
Total\_cycles = 1344 \quad (3.4)
\]

### 3.2.2 Memory Balancing Logic

Memory balancing logic consists of state memory, dual state memory and balancing logic. There are four identical memories in the state memory and the dual state memory, named as row memory as shown in Fig. 3.6. Each memory unit stores a row of the state matrix. Collectively, the state memory is used to store the previous and next state of the encryption.
The power consumption of the memory is dependent on the data being read or written. For instance, power dissipation for writing data ‘1’ is different than writing data ‘0’. This noticeable difference in power dissipation can serve as a side channel information for the power attacks. The power consumption adds up as multiple bit data accessed. For example, in case of 8-bit data bus, power consumption while writing data B’1111_0000 will be addition of power dissipation of four one’s and four zero’s when accessed separately.

Also it is found that, the power dissipation is additive if multiple memories are accessed simultaneously. The maximum power dissipation occurs when the accessed eight bit data is all ‘1’. Thus, for a data access having less than eight 1, a another data with remaining number of 1’s should be accessed to yield the total eight 1’s. This additional data can be simultaneously accessed from another memory of equal size. This was the motivation for introducing a dual equal size memory in tandem. However, the additional memory will incur cost in terms of power and area. Hence, minimizing this extra cost was one of the motivation for looking beyond the conventional CMOS memory.

Emerging technologies such as RRAM, has high density, and ultra low power [77] making them viable next generation on-chip memory. Additionally, a few research groups are exploring the applicability of RRAM technologies for hardware security enhancement [32, 41, 60]. Thus, a logical choice for alternative memory technology was RRAM which helps to minimize power and area overhead with improved security. Therefore, regular memory and inverse memory are implemented using RRAM. When a row memory from state memory is accessed, then associated row memory from inverse state memory will also be accessed in tandem. The dual memory is interfaced to the AES control block using the balancing logic block (BL).

The balancing logic block is a CMOS-based digital logic block. Fig. 3.7 depicts the internal structure of the balancing logic block. The balancing logic block snoops the access link between the AES control unit and the state memory to detect the type of memory access.
and then generates correlated actions for inverse state memory. The snooping function
monitors the type of operation (read or write), address being accessed, and data. In the case of a write operation a byte is written to the state memory while its complement is written to the dual state memory. The goal is to consume power required for writing all ones at any given point of time. For example, when the write operation on the regular state memory is ”13”(H) then the data written on the dual state memory will be ”EC”(H). Thus, the collective power dissipation of both the memories will be effectively for ”FF”(H).
Chapter 4

Simulation and Attack Framework

This chapter discusses the simulation framework to trigger a DPA attack on the proposed architecture. To mount DPA attack, power traces are generated by simulating the execution of hardware implementation. Four different architectures, as shown in Table 4.1, have been simulated and attacked.

<table>
<thead>
<tr>
<th>Name</th>
<th>AEU</th>
<th>Regular State Memory</th>
<th>Inverse State Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES_CMOS_1</td>
<td>CMOS</td>
<td>CMOS</td>
<td>–</td>
</tr>
<tr>
<td>AES_CMOS_2</td>
<td>CMOS</td>
<td>CMOS</td>
<td>CMOS</td>
</tr>
<tr>
<td>AES_RRAM_1</td>
<td>CMOS</td>
<td>RRAM</td>
<td>–</td>
</tr>
<tr>
<td>AES_RRAM_2</td>
<td>CMOS</td>
<td>RRAM</td>
<td>RRAM</td>
</tr>
</tbody>
</table>

Table 4.1: AES design variations under consideration

In section 4.1, the simulation framework used in this work is discussed. Attack framework is discussed in section 4.2.

4.1 Simulation Framework

A customized flow is designed in this work due to the use of CMOS and RRAM modules in the design. Figure 4.1 depicts the simulation power extraction flow for CMOS and RRAM modules. Power extraction of CMOS involves two steps and is based on the work in [65]. First the hardware design is compiled and synthesized into an gate level implementation. Then this gate level implementation is simulated several times with different plaintext vectors each time. The associated input data and power traces are paired and stored for further
evaluation. This flow utilizes Synopsys tools for compilation, synthesis, and simulation of the hardware design.

Figure 4.1: Top Level Simulation Flow for Power Extraction.

In case of RRAM, power extraction involves simulation using Cadence tools. Power information is extracted and stored along with input plaintext information. Several other scripts are used for supporting the simulated power extraction. A plaintext input generator (not shown in figure) is used to ensure that the similar plaintext is fed into CMOS and RRAM simulation.

This research requires the collection of several thousands of power traces. The fact that, each encryption cycle for a plaintext in independent of one another, opened the possibility of dividing the number of encryption cycles over multiple cores to accelerate the power extraction process. A top level Python based script has been created to control this feature and entire flow.

4.1.1 Power extraction for CMOS implementation

Compilation and synthesis is one time overhead to generate the gate level simulation model. The power extraction is automated using a Makefile which is invoked by top python script. This ensures the dependency tracking for the intermediate resources. The arguments for the Verilog simulation and compiler directives are controlled through top python script.
The modified hardware model of the AES used in this research is written in the SystemVerilog. Figure 4.2 shows the entire flow for synthesis and simulation executable generation for CMOS implementation of AES. DC_Shell reads the HDL along with technology library (120 nm) to generate the gate level simulation model and netlist.

For the purpose of extracting instantaneous power consumption for the AES Encryption Unit, the activity data (VCD) in generated and fed to the power analysis engine directly as shown in Figure 4.3. The piped simulation of the VCS tool with pt_shell reduces the overhead of performing a separate simulation. The SystemVerilog testbench reads the plaintext inputs from the external file. The length of the plaintext file is specified by Verilog plusargs. Testbench is also interfaced with state memory model written in SystemVerilog. This ensures the correctness of encryption without extracting the power for memory.

Each simulation generates two types of output files. A Nanosim out formatted waveform containing the power traces for each operation simulated (power_waveform.out) and a file containing plaintext and ciphertext along with start timestamp (simulation.txt). Listing 4.1 shows one such example of simulation file. The trace loader requires the initial timestamp in order to locate and extract each operation from the power waveform.
Figure 4.3: Power modeling flow for AES Encryption Unit.

Listing 4.1: Simulation timestamp with plaintext and ciphertext

| 1 | 50 d54b6b3b4f95b25b328cb43566101af | 6ceba27832d6ce7f742df828fe5e974c |
| 2 | 21550 59e11bb21c78c469d608764d827d51c | 0a1777abff9f1f522552c49536a7d2f9 |
| 3 | 43050 de4f8f069101b56ca380a674faa0181 | 465bd05353e8d6e308e536305cf1614b |
| 4 | 64550 34f02c1c9a154dcdb7a2709647ac73c5 | 706838ca365c1b89ae841390f729c634 |
| 5 | 86050 bbcdb3228f3e2049868a39cbccff8608c | 6614cef4179f401ce6269e4362b42f1a |
| 6 | 107550 2acbe0bf56a8744da769fd6c1fd2290c | 3ee12b2bd16589f14b2f452c3f74204 |
| 7 | 129050 f94f15a15781bd384628aa995d305d62 | a2fab777bc3046830cefa640865b2 |
| 8 | 150550 0ab7395021af42cd86470e68b4273202 | 07755c83221deb57cca68bb595bb1e15 |
| 9 | ... | ... |

4.1.2 RRAM power extraction

The second part of the extraction flow is for the RRAM section of the design. The memory system including row and column decoders and read/write circuitry is implemented using 45nm CMOS transistors and gates (Berkeley PTM models). The memristor used has following metrics $R_{on} = 10K\Omega$, $R_{off} = 1M\Omega$, width = 40 nm, height=60nm and is been implemented in 60 nm technology [4, 5]. The entire RRAM system, circuits and devices were modeled using Verilog-AMS language. This RRAM is interfaced with AES encryption model implemented using Verilog. Thus, it is necessary to use a tool set with a mixed signal simulation environment. Cadence Virtuoso AMS Designer tool compiles, elaborates and simulates mixed language design as shown in Figure 4.4. The output of the
tool is a waveform database (.trn). A python based script is used to calculate the power information and generate the power waveform similar to Nanosim out file format.

Finally a power extracted from CMOS based AES encryption unit and RRAM is summed up and stored in power_waveform.out format, along with simulation timestamp file.

4.2 Attack Framework

The goal of the attacking framework is to achieve consistency in evaluating each of the simulated designs for power analysis and secret key extraction. The attacking module requires access to a number of instantaneous power consumption traces sampled over the time during which the secret key was used to encrypt the given plaintext.

The main concern is maintaining data precision, in order to avoid losing valuable information for a successful attack. Timing accuracy is less of a concern since it is assumed the power traces are accurately capturing the power consumption when the target execution sequence is exercised. Choosing to operate on the subset of the provided power samples requires the maintaining this accuracy.

A top view of attack module implemented in this research is shown in Figure 4.5. There
could be several power waveform files along with associated simulation timestamp files. File read operation involves disk I/O which is a bottleneck for the performance of the

![Diagram of the attack module]

attack module. Once a trace has been read using a trace reader module, crypto instance segregates them according to key byte guess. These byte level bins can be independently processed by byte level attack instance to calculate the differential trace. Using the final differential trace, a final key byte guess can be made.

A single byte attack has been illustrated in Figure 4.6. For each plaintext out of \( p \) number of plaintexts is combined with \( k \) possible key byte guesses to generate cipher text \( C_{p,k} \). In this case number of possible key byte guesses are 256. These computations associated with power traces and grouped as high and low traces. These high and low traces are labeled as \( A_1 \) and \( A_0 \), and are summed with each new trace based on the output of the power model. Once all traces have been processed, \( A_1 \) and \( A_0 \) are divided by the size of each bin to
compute the average high and low traces. The difference between these two average traces is computed to produce the differential trace. The final output of the DPA attack module is the differential trace.

4.3 Summary

This chapter has discussed the simulation environment for extracting the power of the mixed signal design. A customized simulation environment was designed to simulate and extract power of CMOS and RRAM modules and then to generate the combined power output waveforms.
Chapter 5

Result and Analysis

In this chapter, the memristor behavior is studied. Furthermore, a simple crossbar structure has been analyzed and its power dissipation related to different access operation is discussed. Additionally, functional behavior of the balancing logic along with their effect on overall power dissipation are described in later section. Followed by the result of power analysis attack discussion.

5.1 Memristor

HP’s [68] initial analysis of memristor device behavior led to a simple model with ohmic electronic conductance and linear ionic drift in a uniform field. In this research linear model of a memristor based on a physical metal-oxide device is been used. Equation 5.1 is the current-controlled memristive system [68],

\[ v_m(t) = [R_{on}x + R_{off}(1-x)]i_m(t) \]  

(5.1)

where \( x \) is the state variable, \( R_{on} \) is the memristor resistance when \( x = 1 \), \( R_{off} \) is the memristor resistance when \( x = 0 \), \( v_m(t) \) is the terminal voltage, and \( i_m(t) \) is the current through the memristor.

Figure 5.1 shows the simulation run of a thin-film memristor with \( R_{on} = 10^3 K\Omega \), \( R_{off} = 1 M\Omega \). The hysteretic pattern of the curves is a result of the changing memristance that relates voltage to current. Hence, power dissipation of the memristor is different for the same voltage depending upon the position on the hysteresis curve. The variation in
the temperature also impacts this hysteretic behavior as shown in Figure 5.2. Temperature variations will affect significantly to the power dissipation of the memristor.
5.2 RRAM power dissipation

Countermeasures of DPA are dependent on the accomplishing variations in the power dissipation of a device. Hence, it is important to understand power dissipation trends for the device under study. A crossbar structure as shown in Figure 5.3 has been simulated. The crossbar structure has four bit data bus (M=4) and sixty-four address locations (N=64). Additional resistance in the rows and columns of the crossbar architecture represents the nanowire resistances. Circuitry used to decode the address and data multiplexer/demultiplexer are similar to that used in the RRAM architecture discussed in section 3.1.2.

Different access operations such as writing a '1', writing a '0', and reading a '0' have been performed on this crossbar. Each operation is performed on all sixty-four address locations sequentially. Power dissipation has been measured for each such operation and graphically represented as shown in Figure 5.2. In these plots, X-axis represents the bit address and Y-axis represents power dissipation of the entire crossbar while accessing a particular bit address.

Figure 5.3: Generic representation of an MxN RRAM crossbar[32].
The power dissipation across RRAM crossbar, while writing a ‘1’ to different memristor devices is shown in Figure 5.4(a). It can be observed that the power dissipation is decreasing linearly when farther memristor is accessed. As expected, the power dissipation variation is significant if we compare the best (N=1) and worst (N=64) cases, is as high as 11%. This address dependent power dissipation disparity can be seen in the operations like writing a ‘0’ and reading to/across the crossbar as shown in Figure 5.4(b) and 5.4(c), respectively. It is observed that, the difference within write ‘0’ power and write ‘1’ power
is 90%. Also the power dissipation for write is two orders higher than that of the read.

This implies that, if identical data (one or zero) is accessed from two different address locations, power dissipation will be different. Considering a scenario, where different sets of data are accessed from multiple memories in tandem, the disparity in the address based power dissipation is furthermore deceiving.

Every device has some level of susceptibility towards the variations such as voltage, temperature. RRAM is no exception and has high susceptibility to such variations. The power trends with varying voltages (within ±2% range), when writing to specific bit (31st), are as shown in Figure 5.5. The data is written/sensed at 10us to the 31st bit location,

![Graph](image)

Figure 5.5: Voltage variability analysis for the RRAM crossbar, when writing to 31st bit in the first row.

which generates the power spike at the point. The power remains relatively uniform for the remaining time frame. If any, these variations in the data add noise to the physical leakage of the system.

Also, the power trends with varying temperature, when writing to a specific bit (31st), are shown in Figure 5.6. The variation in the power dissipation is negligible owing to the temperature range. The temperature variation trend provides additional flexibility to the resiliency of the DPA attacks.
Figure 5.6: Temperature variability analysis for the RRAM crossbar, when writing to 31st bit in the first row. [32].

5.3 Balance Logic and Inverse State Memory

To validate the effect of balance logic, first a behavioral simulation is necessary. For this purpose 24X8 bit RRAM based regular memory and dual memory along with balancing logic is used. Cadence SimVision is used to plot the waveforms using output database file (.trn). A multiple read after write has been initiated on every address to ensure correct data is stored in both memories. Figure 5.7 shows the consecutive write and read cycles on address 0. During every write cycle 0x00 is written on the regular state memory. As discussed in section 3.2.2, balancing logic will initiate the similar operation on inverse state memory. When write cycle is initiated data provided to regular state memory will be
inverted and fed to inverse state memory. Hence, 0xFF (inverted 0x00) is written on inverse state memory as shown in Figure 5.7. The read cycle followed by write confirms that data and inverted data is stored in regular and invert state memory, respectively. It has also been ensured that both types of memory, CMOS and RRAM, functionally identical.

Furthermore, power dissipation of regular and invert state memory is observed for both CMOS and RRAM memory modules. Three different power dissipation information has been extracted, regular memory, inverse state memory and total power dissipation. Figure 5.8(a) shows the power dissipation of the 16X8-bit CMOS regular memory in a single round of AES operation. The power dissipation of each transformation has a pattern based on the contiguous read and write operations on memory. The inverse state CMOS memory’s power dissipation when inverted data is accessed for the same round of AES, is shown in Figure 5.8(b). The total power dissipation for the CMOS memory is represented in Figure 5.8(c). It can be observed that the power balancing makes the profile uniform, however the spikes in read and write are still visible without any clear data correlation due to the byte level traces.

A similar analysis has been performed on RRAM memory of same 16X8-bit size. Figure 5.9(a) and Figure 5.9(b) show the power dissipation for the RRAM memory and the inverse state RRAM memory power dissipation respectively. It is observed that the power balancing is achieved in Figure 5.9(c) when both the memories are employed. The initial glitches with both the memory transactions are associated with the write access delays. The slight difference among the balanced peaks seen in Figure 5.9(c) are due to the different addresses accessed for that operation.

Even though balancing can be achieved using CMOS memory, the power consumption for the entire architecture has doubled as shown in Figure 5.10 which makes it inefficient for several embedded systems. A solution with similar effectiveness but with low power consumption is achieved using the RRAM based balancing technique, which exhibits an 80%
Figure 5.8: Power dissipation for 1 round of AES for (a) Regular CMOS Memory (b) Inverse CMOS State Memory (c) Total CMOS State Memory.

reduction in power consumption. The area cost is also significantly lower in the RRAM implementation compared to the CMOS even when the CMOS peripheral circuitry is considered. The drawback with the RRAM implementation is the performance degradation, as much as 5X compared to the CMOS. However, with improved write time latencies this degradation can be easily addressed along with parallelization techniques (owing to the high density)[75, 77].
Figure 5.9: Power dissipation for 1 round of AES for (a) Regular RRAM Memory (b) Inverse RRAM State Memory (c) Total RRAM State Memory

5.4 DPA Attack Results

A measure of success is essential to assess the efficacy of a countermeasure. In this work, confidence ratio metric defined in [65] is used. The confidence ratio for a particular key guess is measured as the maximum value of the differential trace for that key guess divided by the maximum value across all other key guesses.
Results in this section have been generated from the evaluation of simulated power traces for four variations of hardware design. The attack mounted is single bit DPA for all key bytes. The cipher key is: 0x00 0x01 ... 0x0E 0x0F [1]. The hardware is simulated with a 100 Mhz clock cycle for a single round, as only first two transformations power dissipation is necessary for mounting attack. For each attack, two plots were generated: a differential trace and a confidence ratio of each of known key bytes.

A successful DPA attack is mounted on AES with single CMOS state memory by collecting 10,000 power traces. This simulation is used as a baseline for further analysis. The plots are as shown in Figure 5.11. The differential trace for all the correct key byte guesses is shown in Figure 5.11(a). Each color in the plot represents the differential trace for a guess key byte. Only first two transformations are used to mount attack. Execution time taken by these transformation is on X-axis and differential power is on Y-axis. Figure 5.11(b) shows the changes in confidence ratio as more and more power traces are used while calculating

\[ C_k = \frac{\text{maximum(differential trace(key guess))}}{\text{maximum(differential trace(all other key guesses))}} \]  

(5.2)
differential trace. Number of traces used for the total attack is on X-axis and confidence ratio over this power traces is on Y-axis. Confidence jumps to ’1’ right after first 200 power traces. Confidence ratio starts increasing as more power traces are used for computation. Confidence ratio is at its highest level for a key byte at 10,000 power traces.

For comparison purpose, DPA attack is mounted on the AES with single RRAM with 10,000 power traces. Plot for differential trace and confidence ratio are as shown in the Figure 5.4. Attack was unsuccessful with these number of power traces. Differential trace, shown in Figure 5.12(a) is really noisy compared to successful differential trace shown in Figure 5.11(a). One of the reason could be the low power consumption of the RRAM compared to CMOS for each of the memory accesses during each transformation step. Confidence ratio of each key byte, shown in Figure 5.12(b), also doesn’t change much once it reaches a stable value.

DPA attack is also mounted on the hardware with balancing logic implemented using CMOS state memory. This attack was mounted using 40,000 power traces. Figure 5.13(a) shows that, the attack was unsuccessful as differential trace has very insignificant power
Figure 5.12: (a) Differential trace for AES with Regular RRAM state memory with 10,000 power traces, all 16 key bytes (b) Confidence ratio for AES with Regular RRAM state memory with 10,000 power traces.

values. Though the confidence ratio, shown in Figure 5.13(b), started building up for a few

Figure 5.13: (a) Differential trace for AES with inverse CMOS state memory with 40,000 power traces, all 16 key bytes (b) Confidence ratio for AES with inverse CMOS state memory with 40,000 power traces.

key bytes, it wasn’t successfully identified a single key byte. DPA attack may be successful for more number of the power traces.

Figure 5.14(a), shows, the differential power trace for unsuccessful DPA attack mounted on balancing logic with RRAM State memory. 40,000 power traces were analyzed in this
attack. Differential power traces are really noisy. A single successful key byte was not
guesstimated. Unlike balancing logic with CMOS state memory, confidence ratio here, shown in Figure 5.14(b), remains essentially stable for almost all the number of power traces used for mounting the attack. Thus showing a successful implementation of the countermeasure. On the whole, key guesses were incorrect, rendering the DPA attack to be unsuccessful compare to the baseline CMOS design.

5.5 Summary

This chapter presents a bottom up simulation for the proposed architecture. Discussion starts with the variations analysis of the memristor model under consideration. The power dissipation variation for the RRAM crossbar is analyzed. The balancing of the total power dissipation is achieved by inverse state memory. This improved the resiliency against the DPA attack. Power attacks were unsuccessful when analyzed using 40,000 power traces.
Chapter 6

Conclusions and Future Work

6.1 Conclusions

This thesis work proposed and developed a DPA mitigating technique based on RRAM. A regular and inverse state memory architecture is developed which uses a balancing logic to similar operations with inverted data. Power dissipation of the inverted data accessed using inverse state memory adds up to the power dissipation of the regular memory, thereby balancing the overall power dissipation. The balanced power traces has uniform hamming weight, which obstructs the secure key extraction from power information. The technique has been verified by implementing a AES architecture along with regular and inverse state memory. A simulation framework was also developed to extract the power information and mount the DPA attack. The framework combines Synopsys and Cadence simulation environment to extract power information from CMOS and RRAM respectively.

RRAM design was implemented in Verilog AMS was modified to use a single power supply for writing a ‘1’ and ‘0’. This was achieved by arranging switches to change the current flow based on the desired data write operation. Power dissipation behavior has been analyzed by simulating a 1x64 RRAM crossbar array. This analysis showed that there is a gradual decrease in power dissipation as the bit address location is increases. Also, basic functionality of balancing logic was verified and effect on over all power dissipation is assessed.
The balancing logic ensures the balanced power dissipation using both type of memories, CMOS and RRAM. Although, the power balancing could be achieved using CMOS, the total power dissipation was high. Use of RRAM thus, helps to minimize this overall power dissipation. However, the performance of the overall system is lowered by 5X compared to CMOS memory. Recent work by HP labs, Samsung and Hynix demonstrates that, this performance gap is closing. Also recent architectures proposed by these semiconductor companies along with research community are addressing issues such as sneak path, noise tolerance, and nondistruptive read.

Also, a DPA attack module capable of attacking all the sixteen bytes of AES secure key is designed and implemented. Attacks has been mounted on architecture with and without balancing logic. It’s been demonstrated that, key can be extracted from an unprotected design using merely 10,000 power traces. Whereas, attack was unsuccessful on protected design even when 40,000 power traces has been used.

6.2 Future Work

Several avenues exist to extend this thesis work. The thin-film memristor in the RRAM design is assumed to have a linear relationship with applied field. Future work could explore empirical models that represents memristor composed of different materials, switching mechanism, etc. to extract power information of the RRAM.

RRAM architecture used in this work suffers from performance related issues as compare to CMOS memory. Future work should ensure that, performance issues are addressed along with various other crossbar architecture specific issues such as sneak path, and nondistruptive read.

For this research, power attack was mounted using DPA. An extension to this work might focus on mounting high order DPA and more sophisticated power attacks such as CPA. This will ensure the complete resiliency against the power attacks. Security of these
devices is not just vulnerable against power attacks. There are a few more attacks such as timing attacks, poses the similar threat to security. Another extension to this work could modify the simulation framework to analyze these types of attack.
Bibliography


