3-22-2012

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

Austin W. Fritzke

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

THESIS

Austin W. Fritzke, Second Lieutenant, USAF

AFIT/GE/ENG/12-15

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/12-15

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Austin W. Fritzke, B.S.E.E.

Second Lieutenant, USAF

March 2012

AFIT/GE/ENG/12-15

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

Austin W. Fritzke, B.S.E.E.
Second Lieutenant, USAF

Approved:

/signed/	22 Feb 2012
Maj Todd R. Andel, PhD (Chairman)	date
/signed/	22 Feb 2012
Lt Col Jeffrey W. Humphries, PhD (Member)	date
/signed/	22 Feb 2012
Maj Mark D. Silvius, PhD (Member)	date

Abstract

The transfer of information has always been an integral part of military and civilian operations, and remains so today. Because not all information we share is public, it is important to secure our data from unwanted parties. Message encryption serves to prevent all but the sender and recipient from viewing any encrypted information as long as the key stays hidden. The Advanced Encryption Standard (AES) is the current industry and military standard for symmetric-key encryption. While AES remains computationally infeasible to break the encrypted message stream, it is susceptible to side-channel attacks if an adversary has access to the appropriate hardware. The most common and effective side-channel attack on AES is Differential Power Analysis (DPA). Thus, countermeasures to DPA are crucial to data security. This research attempts to evaluate and combine two hiding DPA countermeasures in an attempt to further hinder side-channel analysis of AES encryption. Analysis of DPA attack success before and after the countermeasures is used to determine effectiveness of the protection techniques. The results are measured by evaluating the number of traces required to attack the circuit and by measuring the signal-to-noise ratios.

Acknowledgements

I would first and foremost like to thank my Lord and Savior Jesus Christ. To Him be the glory. Secondly, I would like to thank my family and friends, with whom I could ask for no better. You have been so very encouraging and helpful throughout the good and the bad. A special thanks to the gentlemen in the lab who kept me pointed in the right direction and made my research experience enjoyable. To my advisor and teachers, thank you for your guidance and instruction. I am honored to have worked with you.

Austin W. Fritzke

Table of Contents

Page

Page

List of Figures

List of Tables

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

I. Introduction

We are now living in the information age. With current computing technology and networking capabilities, the transfer of large amounts of information is capable of being completed quickly and efficiently. In today's computing environment, most information is stored on computer systems, and most transactions are conducted between the computers along physically insecure channels. However, society has become extremely dependent on the security and protection of information transferred between computing systems, creating a need for information security. Protection of information transfer from one party to another is both crucial and challenging. Whether it is military operations, banking information, online purchases, or any number of sensitive items, it is important that the transferred information remain private between the sender and the receiver. More and more, our lives are dependent on electronic transfers of information. Information security is not only useful for peace of mind, but also for society to operate in the organized, lawful manner that it does. The United States (U.S.) military is no exception to living in the information age. In fact, the U.S. military has become an information age organization where information transfer has become a critical component for mission success [8]. Because most information transferred by the U.S. military is not public, information security is a great concern and has become a significant focus of the Department of Defense (DoD).

1.1 Motivation

As given in the Federal Information Security Management Act (FISMA) of 2002, information security is critical to the defense and security of the nation [9]. One of the main methods used to keep information safe is through cryptographic algorithms. Cryptographic algorithms allow the secure transfer of information over an

insecure channel, free from concern of a third party accessing the information [10]. Cryptographic algorithms were developed by government agencies and private industry to provide information confidentiality and integrity.

As our adversaries become more advanced, it is important for our information to be kept secure. In 2009, Iraqi insurgents were able to hack unprotected Predator drone feeds, gaining vital information on U.S. interests and targets within the area [11]. This vulnerability could have easily been avoided had the feeds been encrypted. Also, it is important to protect the information and technology stored on computer systems that falls into the wrong hands. Between 1990 and 2000, The United States Air Force (USAF) lost 17 aircraft in combat [12]. Many of those aircraft were lost in enemy territory. It is important for critical information to remain secure even after an adversary gains access to a system. An effective way to accomplish both information storage and information transfer security is through encryption.

By encrypting data, a user prevents all but the authorized parties from accessing the protected information. Encryption is defined as the process of transforming information using an algorithm to make it unintelligible to anyone except those possessing a secret key [10]. There are two main types of cryptographic algorithms commonly used today: public and symmetric key encryption. Public key encryption requires two keys and is useful when two parties cannot easily and securely share a single symmetric key. However, public key encryption requires significantly more time and overhead than equivalent symmetric key systems. Symmetric key encryption requires both parties to already share the same key through some predetermined method and is useful for the transfer of large blocks of information.

This research focuses on symmetric key cryptography. The current standard for symmetric key cryptography is the Advanced Encryption Standard (AES) [2]. The algorithm is based on computational security, meaning that the encrypted message can be broken given sufficient time and resources. Fortunately, at our current level of technology, the time and/or resources needed to crack the algorithm are not within a polynomial bounded timeframe. However, there have been ways found to circumvent

the computational security of cryptographic algorithms and to decipher the messages without significant difficulty. Often overlooked when designing secure algorithms is that the hardware used to process the encryption is vulnerable to side channel analysis (SCA), an unobtrusive attack capable of defeating even the most computationally secure systems. Attacks such as these only exemplify the need to provide information defenses in both the software and hardware of information systems.

Discovered by Paul Kocher et al. [4], power analysis attacks are effective against both public and symmetric key encryption algorithms. While the algorithms may be secure on paper, power analysis evaluates the power used by circuits performing the encryption or decryption process. Power usage can be measured on a transistor level and is heavily dependent upon the data. Because of this ability, patterns form which correlate to the message trying to be protected. The benefits of power analysis are that it is fast and unobtrusive. Electromagnetic analysis works similarly to power analysis but requires no contact with the circuit. All an adversary has to do is achieve close proximity to the device performing the encryption for a given period of time. This vulnerability is too great to ignore and must be addressed before true information security can be achieved.

Fortunately, countermeasures to power analysis attacks exist. While there are no ways to absolutely prevent power analysis attacks, countermeasures can increase the time and/or resources needed to determine the secret messages. For AES, the two main side-channel countermeasure types are hiding and masking. Hiding attempts to cover the message with noise from the circuit or use other means such as timing variance to create independence between the information and the power signal [5]. Masking conceals the information by either adding or multiplying random numbers to the data during the encryption process [5]. Both techniques are effective in increasing the difficulty attacking a cryptographic system. The challenge becomes implementing the countermeasures without decreasing the speed, increasing the power consumption, or increasing the area of the cryptographic algorithm beyond reasonable limits.

This research directly supports the USAF mission to protect the U.S. and its

global interests by providing information security. The cryptographic algorithms may be computationally secure, but once an adversary gains access to the hardware containing or receiving the information, the system can be compromised. Power analysis attack countermeasures have been shown to compound the difficulty of successful attacks and are therefore beneficial to the security of the nation.

1.2 Problem Statement

The ultimate goal of side-channel attack countermeasures is to increase the difficulty of finding the secret key used to encrypt the data beyond a reasonable time frame while minimizing the added area, power consumption, and delay on the chip. There has been extensive research done on effective hiding and masking countermeasures for protection of the AES algorithm. However, a large number of implemented countermeasures create unreasonable increases in circuit power consumption, area, or delay. There have been many designs that have attempted to meet these requirements, but some of the most promising countermeasures have never actually been implemented on a chip [1, 13–17]. Of course, no "perfect" countermeasure implementation exists, but there is still work to be done in optimizing the costs versus benefits of countermeasures.

Another area of concern is that countermeasures often emphasize security against specific types of power analysis attacks but remain vulnerable to others. While one countermeasure may be impervious to one type of side-channel attack, the countermeasure is still only as strong as its weakest link. Is it possible to create a countermeasure robust enough to withstand multiple types of power analysis attacks? Or if not, does a combination of different countermeasures integrate in such a way that their benefits combine to withstand multiple types of attacks?

1.3 Research Objectives and Contributions

The desired objective of this research is the testing and combination of two specific AES power analysis countermeasures useful in obfuscating the power and

electromagnetic emanations from the cryptographic circuit. This research seeks to combine two hiding techniques previously untested on an FPGA in hopes robust protection can be achieved with minimal cost from added area, power consumption, and delay in the circuit. The two techniques include randomizing the clock frequency of the algorithm and simultaneously performing bit-balancing through the encryption (or decryption) on the inverse of the data.

1.3.1 Random Clocking to Prevent Trace Alignment. Almost all AES power analysis attacks require power or electromagnetic emanations from more than one iteration of the algorithm. When evaluating, the emanation traces are aligned and evaluated for correlation to the information. By randomizing the clock for each cycle of the AES algorithm, alignment of traces becomes difficult, if not impossible to complete [17]. This randomization greatly increases the difficulty of performing power analysis with only a minimal cost in circuit delay. While this idea is promising in simulation, its effectiveness must be evaluated on an actual circuit to determine true effectiveness and to determine any potential challenges of the design.

1.3.2 Bit-Balancing with Dual Algorithms Processing Inverse Data. One of the most common power attacks seeks to capitalize on the varying power consumption on the transistor level when evaluating a '0' or a '1'. Hamming Weight and Hamming Distance attacks take advantage of these differences. A common hiding countermeasure is to process the message and the inverse of the message (both a '0' and a '1' at all times) simultaneously at the gate level [5]. However, this method is relatively complex and resource intensive. A less costly implementation is to process the inverse of the message at the system level by running two AES algorithms simultaneously on the message and the inverse of the message [1]. While this design theoretically doubles the area and power consumption of the overall AES design, it is still much more efficient than gate level balancing. The difficulty with implementation will be to mix the two algorithms in such a way that they cannot be isolated from each other.

1.3.3 AES Implementation with Combined Countermeasures. An implementation of AES is designed and synthesized on an FPGA using both countermeasures. A baseline AES system without power analysis attack resistance is evaluated against side-channel attacks and then compared to the obfuscated design of this research. Evaluation is based on an increase in benefits and a limitation of costs. Benefits are measured by comparing the resistance of the new design to commonly used power analysis attacks when compared to the strength of the baseline system. Trade-offs are compared using area, power consumption, and delay metrics against the baseline system. By creating a highly robust and SCA resistant AES system with minimal cost to the user, this thesis provides a significant contribution to the U.S. Air Force and to the information security community in general.

1.4 Thesis Organization

The work presented in this thesis is divided into five main sections. Chapter 2 provides a background on the AES algorithm, power analysis attacks, and related research. Additionally, Chapter 2 includes several of the most common countermeasures used to obfuscate the AES algorithm. Chapter 3 covers the methodology of the approach and implementation of this thesis based on the stated objectives. Chapter 4 details the results of the research and any adjustments made to meet the objectives. Finally, Chapter 5 provides a summary of the work accomplished and includes the major contributions and recommendations for future work of this research.

II. Background

This chapter serves to provide background information to the reader on symmetric key encryption, power attacks and side-channel analysis countermeasures. The chapter is structured as follows: Section 2.1 covers the AES algorithm and how it works. Section 2.2 gives an overview of Simple Power Analysis. Section 2.3 covers Differential Power Analysis, to include a step-by-step theoretical methodology to performing an effective power attack. Section 2.4 covers the actual hardware implementation strategies of a power attack. Finally, Section 2.5 describes several current countermeasures to power attacks and their implemenentations. While only two specific countermeasures are used in this research, it is important to understand the basic concepts of all general countermeasures.

2.1 The AES Algorithm

Understanding of all intermediate steps of the cryptographic system is helpful to determine potential weaknesses by side-channel attacks. The AES algorithm is a form of the Rijndael algorithm and is the current standard for symmetric key cryptography [2]. It is a symmetric block cipher that processes 128-bit data blocks and can operate on keys with lengths of 128, 192, or 256 bits. Each data block consists of 16 bytes and is four rows deep and four columns wide. For this research, when referring to the AES algorithm, we assume a key length of 128 bits to ensure it is implementable on the hardware used in the study. The AES algorithm is computationally secure and consists of 10, 12, or 14 rounds depending on the key size.

A pictorial overview of AES is given in Figure 2.1. Each round consists of four individual transformations of the data: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The algorithm begins by adding the RoundKey and continues into the rounds. The last round does not contain the MixColumns function. For 128-bit key AES, our focus in this paper, there are a total of 10 rounds.

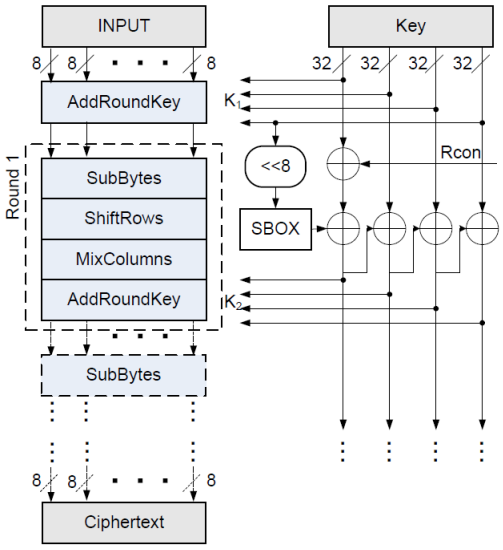

Figure 2.1: The AES Algorithm [1]

2.1.1 SubBytes. The subbytes transformation is the only non-linear transformation within the AES algorithm and therefore subject to great interest for power attacks [18]. The byte substitution operates independently on each byte using a substitution table [2]. The table is constructed using two transformations: first take the multiplicative inverse in the finite field GF(2^8), then apply an affine transformation over GF(2). Figure 2.2 gives an example of an S-box in hexadecimal format. The input data is used to choose the row and column of the output byte. For example, an input byte "01" points to output "7c" in the S-box below.

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

Figure 2.2: AES S-box [2]

2.1.2 ShiftRows. The ShiftRows function is a linear transformation where the last three rows of the block are cyclically shifted [2]. Each byte is shifted by rotating to the left. The first row does not shift, the second row shifts by one, third row by two, and fourth row by three.

2.1.3 MixColumns. The MixColumns function is a linear tranformation that operates on the block column by column. It performs a constant matrix multiplication on the four input bytes [18]. MixColumns is the only transformation not performed in the final round of AES.

2.1.4 AddRoundKey. In the final process of a given round, a Round Key is added to the data using a simple bitwise XOR operation [2]. The Round Keys are derived from the key schedule which is derived from the original 128-bit key. The Round Keys are calculated during the key expansion process of the AES algorithm and are 1408 bits each for a 128-bit key. The actual key is used only at the very beginning of AES encryption before Round 1 where the original key is initially XORed with the data.

The full AES algorithm pseudocode is given in Algorithm 1 as described in the AES documentation. Of the four different functions, the SubBytes and AddRoundKey functions are the most vulnerable to leaking information to attackers. The SubBytes non-linearity provides significant leakage among the power spectrum, and the AddRoundKey function is directly related to the secret key and therefore a choice target for Simple and Differential Power Analysis attacks.

The number of rounds is represented by N_r and the number of 32-bit words in the Cipher Key is represented by N_k. When dealing with a 128-bit key, $N_r = 10$. Else, $N_r = 12$ for a 192-bit key and $N_r = 14$ for a 256-bit key. The variable N_b represents the data block size (in words) and is always 4.

Algorithm 1 Pseudo Code for the AES Algorithm

Cypher
INPUT: byte $in[4 * N_b]$, word $w[N_b * (N_r + 1)]$
OUTPUT: byte $out[4 * N_b]$

byte $state[4, N_b]$
state = in;
AddRoundKey($state, w[0, N_b - 1]$)

for $round = 1$, $step$ 1 to $N_r - 1$ **do**
    SubBytes($state$)
    ShiftRows($state$)
    MixColumns($state$)
    AddRoundKey($state, w[round * N_b, (round + 1) * N_b - 1]$)
end for

SubBytes($state$)
ShiftRows($state$)
AddRoundKey($state, w[N_r * N_b, (N_r + 1) * N_b - 1]$)

$out = state$

2.2 Simple Power Analysis

The main concept behind both Simple Power Analysis and Differential Power Analysis is to extract the secret key from a cryptographic device [3]. Figure 2.3 gives a broad overview of the cryptographic system. By using side-channel analysis we can measure the power consumption and/or electromagnetic (EM) radiation emanating from the circuit. From this data, correlation between the leaked information and

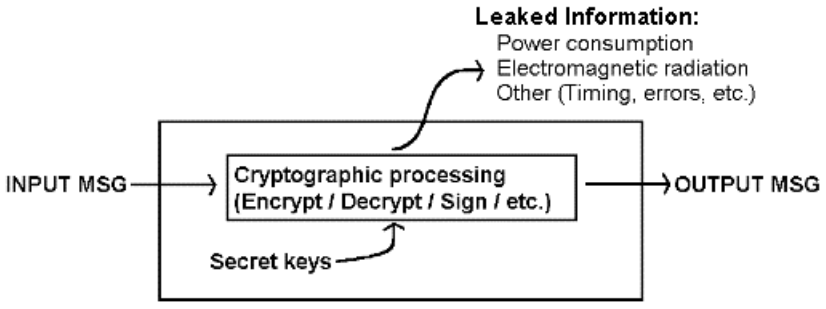

Figure 2.3: Cryptographic System [3]

the secret keys becomes apparent. Other attack methods such as timing analysis or insertion of glitches[1] are also effective but are not evaluated in this paper. Instead, the primary focus is on simple power analysis, differential power analysis, and their respective common countermeasures.

Simple Power Analysis is a tool for interpreting power traces collected during the operation of a cryptographic algorithm [5]. Simple Power Analysis (SPA) is useful if the attacker only has access to a few traces, but in general is difficult to use against AES. Most SPA attacks are conducted by visual inspections of a trace. When viewing the AES algorithm, it is almost impossible to determine the key from visual inspection. Instead, SPA is more helpful in determining the overall structure of an algorithm, not specifically in determining its key.

When evaluating any cryptographic algorithm, SPA is helpful for preliminary evaluation to determine when repetitive operations occur over the entire algorithm's period. For example, Kocher et al. [4] completed SPA on the Data Encryption Standard (DES) algorithm in order to determine when each round occurs. In Figure 2.4 one can clearly differentiate between the 16 rounds of the DES algorithm.

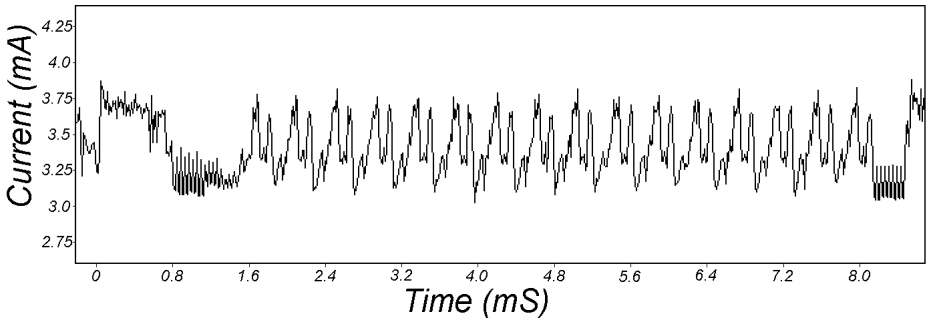

Figure 2.4: SPA Trace of Entire DES Algorithm [4]

Focusing back on AES, the signal can be extracted using SPA after complex statistical methods have been used, but for the most part SPA is only helpful for

[1]Glitches are the transitions at the output of a gate that occur before the gate switches to the correct output after the input(s) change. Glitches are a significant source of power consumption for CMOS circuits and therefore vulnerable to power analysis attacks [19].

alignment and locating important functions in the algorithm. Often small signal-to-noise ratios make key extraction infeasible using SPA on AES. However, if it is possible to obtain more than one power trace of the AES algorithm, attackers can perform the more powerful Differential Power Analysis attack.

2.3 Differential Power Analysis

While SPA is limited to visual inspection, Differential Power Analysis (DPA) is a significantly more powerful attack that uses statistical analysis and error correction techniques to extract information correlated to the secret keys [3]. The main advantage of DPA over SPA attacks is that no detailed knowledge of the cryptographic system and circuit is necessary [5]. However, DPA requires many traces of the algorithm to work properly, often requiring many thousands of traces. If the collection of many traces is feasible, a large amount of information may be gained that was not previously visually apparent using SPA.

There are two main phases of a DPA attack: data collection and data analysis [3]. Data collection is performed by sampling a circuit's power consumption or EM radiation during the cryptographic operations. Data analysis is the more involved phase of DPA and often involves correlation of the collected power traces to a hypothetical power model created by the attacker.

2.3.1 First Order DPA. To better understand how DPA works, it is easiest to follow step-by-step how an attack is performed as described by [5]. A visualization of the steps is given in Figure 2.5.

Step 1: Choose an Intermediate Result of Algorithm to Attack. Before any traces are collected or analyzed, it is critical to choose the intermediate value that will best reveal the secret key. This step can be done by making sure the intermediate result is a function $f(d, k)$ of both the data d and a small part of the key k. For AES, the input or output of the SubBytes function is an optimal intermediate

target.

Step 2: Measure the Power Consumption. During this step, the power traces are collected and stored. It is important that the attacker knows the data values (plaintext or ciphertext) that correspond to each power trace. In this way, the attacker can calculate the expected intermediate value corresponding to each trace. Given that the length of each trace is T, and there are D traces each with corresponding d_i data values, we can create a matrix P of size $D \times T$. It is crucial that all power traces be aligned correctly. To do this alignment, a trigger signal for the oscilloscope can be generated. Otherwise, the traces must be aligned using alternate techniques.

Step 3: Calculate the Hypothetical Intermediate Values. The attacker must then calculate a hypothetical intermediate value for every possible choice of k. The possible choices for k are stored in a vector K units long. Then based on the previously recorded plaintexts or ciphertexts, the attacker calculates the hypothetical intermediate values for each key and each data value. The results are stored in a matrix V of size $D \times K$. The goal of DPA attacks is to determine which column in V correctly matches to the actual data. When this informed key guess occurs, the hypothetical key value k_i will match the real secret key.

Step 4: Map Intermediate Values to Power Consumption Values. In this step, the attacker maps the hypothetical intermediate values to hypothetical power consumption values. In essence, the attacker guesses at what the power consumption of the circuit should be for each hypothetical intermediate value in matrix V. There is no standard way to complete this step, but instead an attacker can choose between many power models to derive a hypothetical power consumption matrix H. Some of the more common power models are the Hamming Weight (HW), Hamming Distance (HD), Zero-Value (ZV), and Bit models.

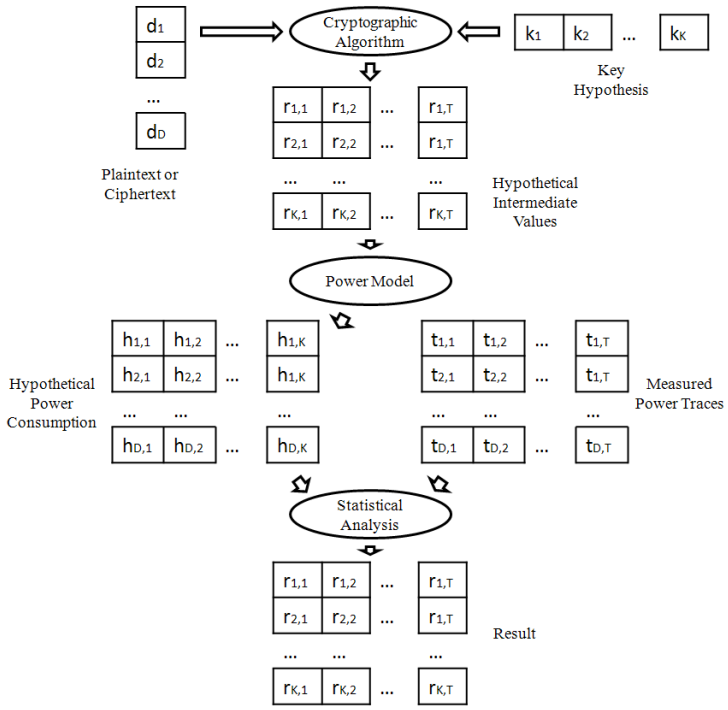

Figure 2.5: Steps of DPA Attack [5]

Step 5: Compare the Hypothetical Power Consumption Values with the Actual Power Traces. Once the matrices P and H have been created, they can be compared. Each column h_i is compared with each column t_i. The attacker is comparing the hypothetical power consumption of each key guess with the actual power consumption at every position. By doing this step, the attacker creates a matrix R with size $K \times T$ where each element $r_{i,j}$ contains the result of the comparison between the h_i and t_i column values. The indexes of the highest values of the matrix R reveal the positions at which the hypothetical key guess matches with the actual key used by the cryptographic algorithm. The matrix R can be plotted in K different plots where each plot (a single row) matches with each key guess, and the x-axis of each plot correlates to specific times during the algorithm. A correct key guess will have a spike in the graph at the point in the algorithm where the chosen intermediate value is processed. A visual breakdown of the five main steps of DPA is given in Figure 2.5.

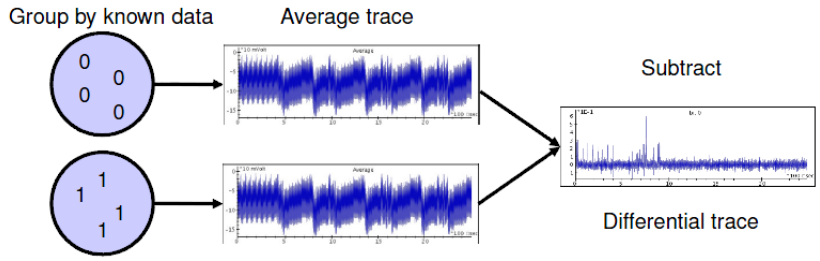

Figure 2.6: Bit Model DPA Attack

2.3.1.1 Bit Comparison. The method shown in Figure 2.5 gives an overview of the general steps of correlation DPA. Another common, simpler method is the bit model where the recorded traces are combined into two separate groups: traces where the hypothetical intermediate value should be 0, and traces where the hypothetical intermediate value should be 1 for a given key guess. The bit model is best explained by the Riscure Inspector training slides [20], describing the bit model DPA attack in Figure 2.6. The bit model is effective, but limited to only evaluating one bit at a time. This process leads to either poor correlation (when attempting to guess a key based on one bit) or slow implementation (one key-bit at a time). Instead, correlations models, as described in the following section, are more commonly used.

2.3.1.2 Correlation. In Figure 2.5, the attacker uses a power model to convert the hypothetical intermediate values into hypothetical power consumption traces. The most common power models used today are given below:

1. Hamming Weight (HW) Model
2. Hamming Distance (HD) Model
3. Zero-Value (ZV) Model

The HW model counts the number of '1's in a binary number and assigns that to its value [5]. For example, 00011000 would have a HW of two. The HW model is effective in modeling power traces because of the difference in power drawn from a circuit evaluating a '1' compared to a '0'.

The HD model is even more effective than the HW model, but requires knowledge of either the preceding or following intermediate value as well as the current hypothetical intermediate value [5]. The HD is the difference in '1's between two binary numbers. For example, if a number were to change from 00011000 to 11111111, the HD would be six. There is usually a significant consumption of power when a device in a circuit transitions its output value from '0' to '1' or vice versa. In this way, the HD model can be a very effective model for hypothetical power consumption.

Finally, the ZV model assumes that the power consumption for the data value 0 is lower than the power consumption for all other values [5]. The data within a ZV model is set as either a 0 or 1 depending on whether the data evaluated is equal to 0.

$$h_{i,j} = ZV(v_{i,j}) = \begin{array}{ll} 0 & \text{for } v_{i,j} = 0 \\ 1 & \text{for } v_{i,j} \neq 0 \end{array}$$

The ZV model is particularly useful against multiplicative masking because any value multiplied by zero equals zero. Therefore, multiplicative masks cannot mask data values equal to zero and therefore the ZV model remains effective. However, the ZV model generally requires several times more traces than the other models in order to obtain then same results [13].

Once the attacker has the hypothetical power traces, they must be correlated with the actual power traces in order to determine a match. This process is most often completed by calculating the coefficient correlation according to Equation 2.2, where r is the coefficient of correlation, D is the number of traces, h is the current

power model value being used (HW in this case), $\bar{h}$ is the average of the power model values, t is the power trace sample, $\bar{t}$ is the average power, and i is the number of samples in j power traces.

$$r_{i,j} = \frac{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i) \cdot (t_{d,j} - \bar{t}_j)}{\sqrt{\sum_{d=1}^{D}(h_{d,i} - \bar{h}_i)^2 \cdot \sum_{d=1}^{D}(t_{d,j} - \bar{t}_j)^2}} \tag{2.2}$$

The two matrix columns are summed and then multiplied by each other, producing a correlation value. If the key guess was correct, there will be a correlation spike. If incorrect, there will not be anything similar between the two traces.

2.3.2 Higher Order DPA. Higher order DPA attacks work in the same manner as first order DPA except they take advantage of several intermediate values, not just one [5]. Masking AES is a popular countermeasure and is performed by either adding or multiplying a random mask to the intermediate values. First order DPA is then thwarted by the intermediate values' independence from the data, but higher order DPA defeats this countermeasure. For example, say that intermediate value u is masked with mask m. A second intermediate value v is also masked with m. We can combine the two traces and calculate the hypothetical value of the two intermediate values when combined as shown in Equation 2.3.

$$w = comb(u, v) = u \oplus v = u_m \oplus v_m \tag{2.3}$$

By following this approach, we can calculate w without ever having to know the mask. This process is the basic idea behind higher order DPA. Preprocessing of the power traces is required because the intermediate values do not usually occur at the same period of time along the traces. Therefore, the two or more points along the trace must be combined by one form or another.

2.4 Methods to perform SPA and DPA

Now that the procedure of evaluating the collected power traces is known, it is important to take a step back and understand how an attacker actually obtains power traces. If the procedure were heavily intrusive, side-channel countermeasures could easily be implemented through external hardware. Unfortunately, power measurement is easy to set up and requires little to no intrusion on the actual circuit. The two main methods include placing a resistor in series with the circuit's power supply/ground or by using an electromagnetic probe.

2.4.1 Resistors in Series. Arguably the most basic of methods is inserting a resistor in series with either ground or the power supply to the board. This method is used in several attacks, including [6, 21]. The voltage across the resistor corresponds to the power based on $P = V^2/R$. The size of the resistor is important for two reasons: it must be limited in the effect the resistor will have on the circuit while at the same time provide a large enough power signature to the oscilloscope. Generally, resistors in the range of 0.1 to 20 ohms are used. Inserting a resistor in series with the circuit is easy to do but is also susceptible to noise from the power supply. Often, a low pass filter is also implemented to remove frequencies significantly greater than the clock frequency. A common setup of an attack using a resistor in series is shown in Figure 2.7. The setup is composed of a computer, digital oscilloscope, circuit (FPGA board in this case), and power supply.

2.4.2 Probes. The second trace collection option is to use a probe to measure the power. Electromagnetic (EM) probes are a common option and have been used in attacks such as documented by [22]. Magnetic flux is proportional to the current as described by Faraday's law, and therefore power is proportional to the electromagnetic spectrum as well. EM probes are effective because they can be moved to pinpoint the exact location on the chip the attacker wishes to focus upon. This feature can potentially minimize unwanted noise. The drawback to EM probes is that they are significantly more expensive than buying a single resistor. However, their cost is

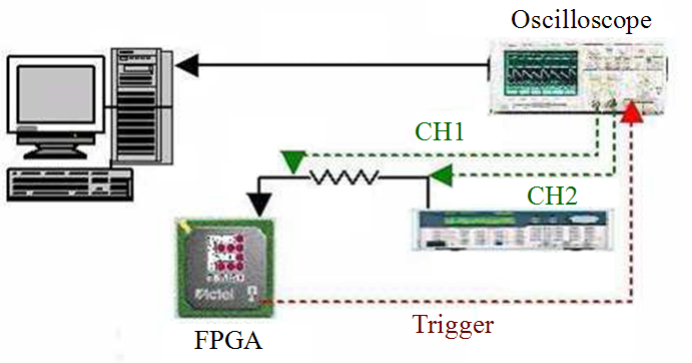

Figure 2.7: Experimental Setup Using a Resistor in Series [6]

low enough to be inconsequential compared to the other supplies needed (computer, digital oscilloscope, etc).

2.5 Countermeasures

Because of the effectiveness of side-channel attacks in obtaining the secret key from cryptographic devices, attention has moved to developing countermeasures for power attacks. The main countermeasures for protecting the AES algorithm are hiding and masking techniques. Each have their own strengths and weaknesses, and no countermeasure is implemented without a cost. The main costs to each countermeasure are listed below.

1. Increased circuit area
2. Increased power consumption
3. Increased execution time

Of course, the benefits include greater obfuscation of the algorithm against attacks. Currently, there is no perfect countermeasure that can protect a circuit against all known attacks. More realistically, a countermeasure's effectiveness can be measured in either the number of increased power traces needed to extract a key or an increase in order of the DPA attacks needed to circumvent a mask. The goal is then to create a countermeasure that makes the circuit impractical to polynomial-time bounded side-channel attackers.

2.5.1 Hiding. The goal of hiding the AES circuit is to make the power consumption of cryptographic devices independent of the intermediate values [5]. This goal can be done either by randomizing the power consumption over a given period or by flattening the power consumption so that the power used by all devices is equal over all operations. While there are many ways to hide a circuit, this research focuses on the most common implementations: delay insertion, dual rail logic, flattening, clock variance, and bit-balancing techniques.

2.5.1.1 Delay Insertion. The main idea behind randomly inserting delay cells into a cryptographic system is to prevent power trace alignment, therefore thwarting the DPA procedure. Delay cells also help prevent glitch attacks by halting the propagation of the glitch through the rest of the circuit. As in [7], delay cells are randomly inserted in areas of the algorithm where leakage is significantly high. It is important that the delay cells be as close to the target intermediate values as possible. Otherwise, an attacker could simply "cut out" the part of the recorded power signal before and after the target area and be left with traces of the intermediate values that do not contain any delay cells and are therefore aligned. Also, consideration should be taken to keep the power signature of the delay cells either random or similar to the original algorithm. It is not desirable to create delay cells that are easily apparent with SPA.

Randomly inserting delay cells into a cryptographic algorithm adds very little to the area and power required by the circuit. Compared to the overall time it takes to complete the algorithm, the delay created by the delay cells is minimal. The main concern for delay insertion is creating pseudo-random numbers quickly so that a random number of delay cells are generated per round of the algorithm. An efficient pseudo-random number generator should have minimal effect on the size and speed of the circuit, but is usually not the case. Also, it is important that the pseudo-random numbers be cryptographically secure [23]. Because of this requirement, considera-

tion should be taken in producing acceptable pseudo-random values that an attacker cannot predict.

2.5.1.2 Clock Variance. Going a step further, clock variance seeks to randomly change the period of the clock over a given number of clock cycles. Similar to delay insertion, clock variance makes trace alignment difficult [17]. It becomes nearly impossible to pre-process the traces for DPA because the random clocking alters the composition of each separate trace. The clock frequency can be varied every cycle or after a given number of cycles depending on preference of the user. Attempts at clock variance has been made on smart cards [24], and has shown robustness against resynchronization attempts. Similar results can be expected for FPGA implementation, making clock randomization a viable countermeasure.

Similar to delay insertion, the size of the algorithm on the circuit does not change significantly. However, the speed of the algorithm will change as the clock frequency changes. For example, if the clock frequency is varied randomly from 50% to 100%, the algorithm will run 25% slower on average. Also, a pseudo-random number generator outputting up to every clock cycle must be created and may have a significant effect on the circuit's speed and size depending on the system.

2.5.1.3 Dual Rail Logic and Flattening. One of the more popular hiding countermeasures is the attempt to flatten the power signature of all components within the circuit's hardware for all values of data. An effective method is performed at the cell level using Dual-Rail Precharge (DRP) logic blocks [5]. The idea behind DRP logic is to create logic cells that make power consumption constant during each clock cycle. Every input and output into a cell is paired with its inverse and therefore a constant balance of '0's and '1's are entering and exiting the cell at all times. Figure 2.8 gives an example of a dual rail AND gate.

There are additional methods to flatten the power signature other than DRP logic. In [25], a low power S-Box has been designed. By minimizing the power output

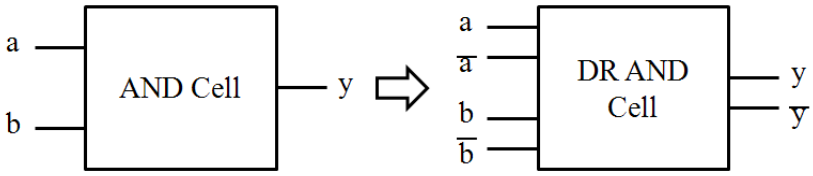

Figure 2.8: Dual Rail AND Gate

of the circuit's key components, the signal-to-noise ratio is decreased making side-channel attacks more difficult to implement.

While eliminating the power signature appears to be an excellent method for AES obfuscation, flattening techniques are not perfect and can be quite costly. DRP logic will double to triple the size of a circuit, increase the power required by 2-3X, and finally slow the circuit down by at least 50% due to added stages and/or precharging required for each cell [5]. Other flattening techniques are not as costly, but are also usually not as effective.

2.5.1.4 Bit-Balancing. Similar to flattening, bit-balancing attempts to balance bitflips for every intermediate value. This balancing is accomplished by running the AES algorithm and an equivalent algorithm processing inverted data in parallel [1]. The idea is centered around the HW model. By inverting the data, each Hamming Weight for all intermediate values will be equal when considering both circuits. Of course, the two circuits must be combined in such a way that an attacker cannot separate the power emanating from each. In many ways, bit-balancing is DRP at the circuit level instead of at the logic gate level. Both implementations are especially effective against the HW model for all orders of DPA.

As expected, bit-balancing doubles the area and power used by circuit. However, the speed of the algorithm is not noticeably changed. The challenge comes from guaranteeing the two parallel circuits cannot be separated in any way by an attacker.

Hiding AES is effective in confounding attacks on the algorithm, but is not guaranteed. Even the best hiding techniques can only make DPA attacks more difficult

to implement, but not impossible. Using the same concept, it may be worthwhile to simply insert noise generators into the circuit as done by [15]. By lowering the signal-to-noise ratio below a given threshold, it becomes infeasible for an attacker to perform a DPA attack given the amount of noise in the power traces.

2.5.2 Masking. In masking, every intermediate value v is concealed by a random value m such that $v_m = v * m$ [5]. In this way, all intermediate values appear to be independent of the data within the cryptographic system. In general there are two types of masks: boolean and arithmetic masks. 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. Multiplicative masking given as $v_m = v \times m$ is frequently used in masked AES. Both methods provide complete independence of the intermediate data from the power trace, preventing first order DPA from being effective.

2.5.2.1 Masking the S-Box. Once a mask is inserted into an intermediate value, it must eventually be removed, thus masking the S-Box is a difficult task. The AddRoundKey, ShiftRows, and MixColumns operations are all linear operations and therefore masking is quite trivial. The difficulty in masking AES is masking the SubBytes operation due to its non-linearity [7]. It can be done as seen in Figure 2.9, but requires an extensive network of masking. Essentially, the linear parts of the S-Box inversion must all be individually masked for the entire inversion process to be covered. A second method was proposed by [13]. Instead of using a lookup table for the S-Box, the data would be squared and multiplied using masked squaring and masked multiplication. The pseudocode for masked squaring and multiplication are given in Algorithms 2 and 3.

An alternate to both methods of masking the S-Box is to pre-calculate the S-Box lookup table for each new mask. While all the different methods to mask the S-Box are effective, they all require a significant amount of resources. Masking generally more than doubles the circuit size and cuts a circuit's speed by at least

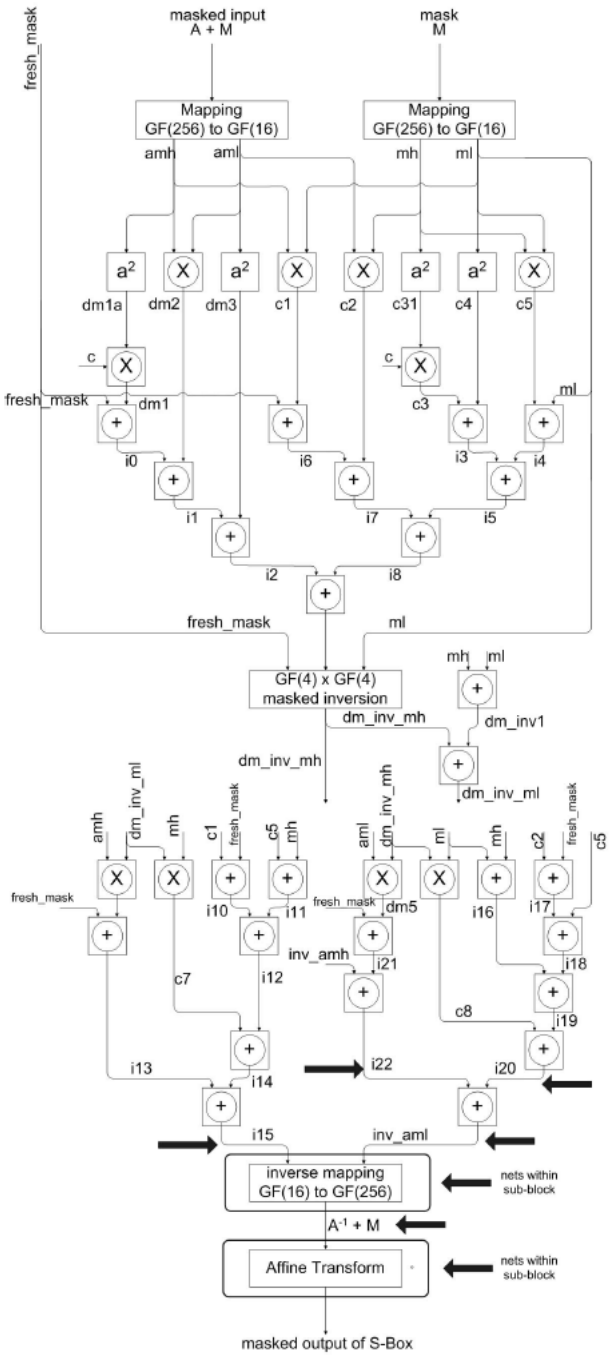

Figure 2.9: Masked S-Box Schematic [7]

Algorithm 2 Perfectly Masked Squaring [13]

Input: $x = u^e + r_{1,i-1}$
Output: $u^{2e} + r_{1,i}$

1: $f_i \leftarrow x^2$
2: $s_{1,i} \leftarrow r_{1,i-1}^2 + r_{1,i}$
3: $t_i \leftarrow f_i + s_{1,i}$

Algorithm 3 Perfectly Masked Multiplication [13]

Input: $x = u^e + r_{1,i-1}$, $x' = u + r_{2,i}$
Output: $u^{e+1} + r_{1,i}$

1: $f_i \leftarrow x \cdot x'$
2: $v_{1,i} \leftarrow x' \cdot r_{1,i-1}$
3: $v_{2,i} \leftarrow v_{1,i} + r_{1,i}$
4: $s_{1,i} \leftarrow v_{2,i} + r_{1,i-1} \cdot r_{2,i}$
5: $s_{2,i} \leftarrow x \cdot r_{2,i}$
6: $t_{1,i} \leftarrow f_i + s_{1,i}$
7: $t_i \leftarrow t_{1,i} + s_{2,i}$

half. Also, multiplicative masking is still vulnerable to ZV model attacks because any intermediate value that equals 0 will still be 0 even after it is masked because $m\times$ 0 = 0.

2.5.2.2 Masking Against High-Order DPA. Even when the data is perfectly masked, higher-order DPA methods can still extract the intermediate values by comparing two intermediate values that share the same mask. Fortunately, more masks can be added to the cryptographic system to the point where each intermediate value has its own mask [16]. However, higher-order masking increases the area of the circuit exponentially for every order attack it is guaranteed to prevent. This side effect also follows for the speed of the circuit. According to the design in [16], a third order masking implementation of AES would be 150 times slower than an unprotected implementation of the AES algorithm.

Masking can be proven secure against low orders of DPA, but at a great cost. It is difficult to mask the SubBytes function in AES, and the number of pseudo-random masks needed grows exponentially to protect against higher-order DPA.

2.6 Conclusion

While the AES algorithm is computationally secure in its input/output relationship, physical implementations are vulnerable to side-channel attacks - specifically SPA and DPA. While it is difficult to obtain the secret key using SPA, DPA is a powerful tool capable of attacking AES, given enough collected power traces. In response, several countermeasures have been developed. Hiding techniques seek to randomize or flatten the power consumption of the intermediate values. Masking techniques conceal the intermediate values with a random mask leading to independence between the intermediate values and the data. While not perfect, countermeasures are effective in greatly increasing the difficulty to perform DPA on a cryptographic system. However, countermeasures cannot be implemented without a cost. Increases in circuit area, power consumption, and delay in the system are seen and vary for each countermeasure. When choosing a prevention method, its obfuscation effectiveness must be considered against its cost so in the end the right countermeasure implementation is chosen for a specified system. The two countermeasures used in this research are both hiding techniques with their focuses on clock variance and bit-balancing.

III. Methodology

3.1 Problem Definition

3.1.1 Goals and Hypothesis. Every AES system can be cracked given enough time and resources. The challenge then becomes increasing the time and resources required to an unreasonable level for an attacker. Many AES implementations run on chips that do not provide an adequate amount of difficulty to attack. Cryptographic circuits such as the AES algorithm are particularly susceptible to power attacks such as differential power analysis (DPA). Therefore, it is desirable to incorporate DPA countermeasures within the AES implementation to prevent an attacker from determining the key given a reasonable amount of time and effort. Unfortunately, every countermeasure has its drawbacks. The main drawbacks of adding extra software or hardware to a cryptographic algorithm include:

1. Increased circuit area
2. Increased power consumption
3. Increased execution time

The goal of this research is to determine whether certain AES DPA countermeasures are capable of hiding the secret key from an attacker while minimizing added delay, area, or power consumption. The hypothesis of this research is that theoretically proven countermeasures increase the time and/or resources needed by an attacker to determine the secret AES key in a cryptographic system. Because no countermeasure is perfect, defining the ability to hide the secret key is somewhat ambiguous. For this research, the cryptographic strength of a given circuit is measured against a baseline system with no countermeasures. An effective system requires several orders more time and/or resources for an attacker to break than the baseline system.

3.1.2 Approach. The goal of this research is to develop and analyze the effectiveness of two countermeasures to hide data within the cryptographic system from side-channel attacks. A combination of two theoretical countermeasures, each

designed to hide data from an attacker are used. The first is a system-level bit-balancing scheme that obfuscates the data from Hamming Weight attacks. The second randomizes the clock frequency each clock cycle to prevent the alignment of traces. The countermeasures are compared to a baseline AES system without any side-channel defense.

3.1.2.1 Establish Baseline. To determine the effectiveness of the final design, baseline values in the area of speed, size, power, and the number of traces needed for a successful attack are established. The basic AES algorithm is an iterative hardware implementation. Once established, the circuit is characterized to determine baseline values for the "standard" AES system.

3.1.2.2 Bit-Balancing. Bit-balancing balances bit-flips for intermediate values of the AES algorithm. In general, the power drawn by the circuit changes depending on the value of the bit being evaluated. By evaluating every bit and its inverse simultaneously, the power signature should theoretically be independent of the value of the bit. A system level approach to bit-balancing is shown in Figure 3.1. By interleaving the two algorithms, an attacker should not be able to correlate the intermediate values and the power traces.

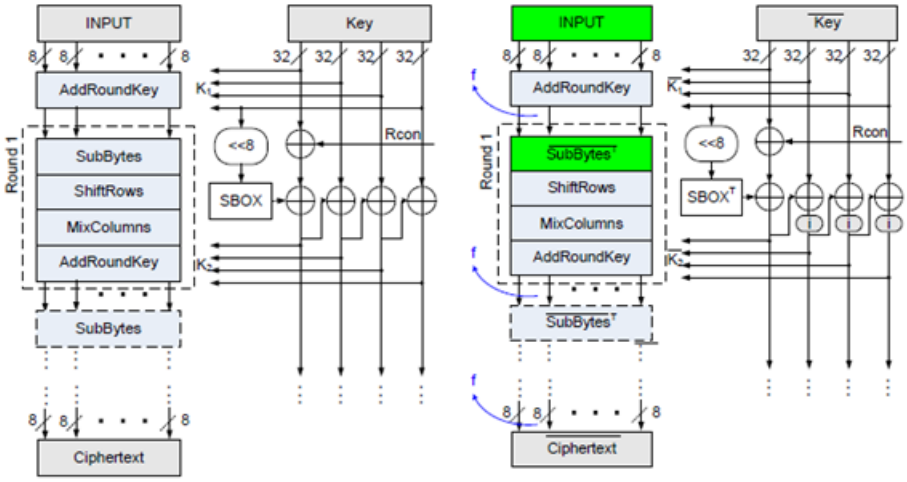

Figure 3.1: Dual AES Algorithms [1]

3.1.2.3 Randomize Clock Frequency. Because DPA requires a large number of traces to correlate a key guess with power, it is important that the traces be aligned. If not, each trace must be evaluated individually. To force this issue, the clock frequency can be randomized every clock cycle. The clock pulse width is determined pseudo-randomly using a linear feedback shift register (LFSR) [17]. An LFSR can provide a pseudo-random stream of zeros and ones. The LFSR is polled twice and the clock cycle varies depending on the LFSR output, therefore providing pseudo-random clocking of the circuit. This randomization should prevent the alignment of traces and therefore hinder a power attack.

Both countermeasures are evaluated individually and then in combination. The results are compared to the baseline AES system to determine effectiveness of the design. This design approach allows each countermeasure's contribution to be monitored and compared against the standard AES design.

3.2 System Boundaries

The system under test (SUT) is the DPA Resistant AES System with the two hiding countermeasures. Figure 3.2 gives a visual representation of the SUT. There are four components within the SUT. They are the AES algorithm, the FPGA, the processor, and finally the AES configuration countermeasures. The component under test (CUT) is the AES configuration countermeasures.

3.2.1 AES Algorithm. The AES encryption algorithm is a form of the Rijndael algorithm and is the current standard for symmetric key cryptography [2]. It is a symmetric block cipher processing 128 bit blocks of data and supports 128, 192, or 256 bit length keys. This research assumes a key length of 128 bits, thereby restricting the algorithm process to 10 rounds. The algorithm can be implemented in software or hardware, and can vary in the amount of pipelining being performed. For this research, an iterative hardware implementation is used.

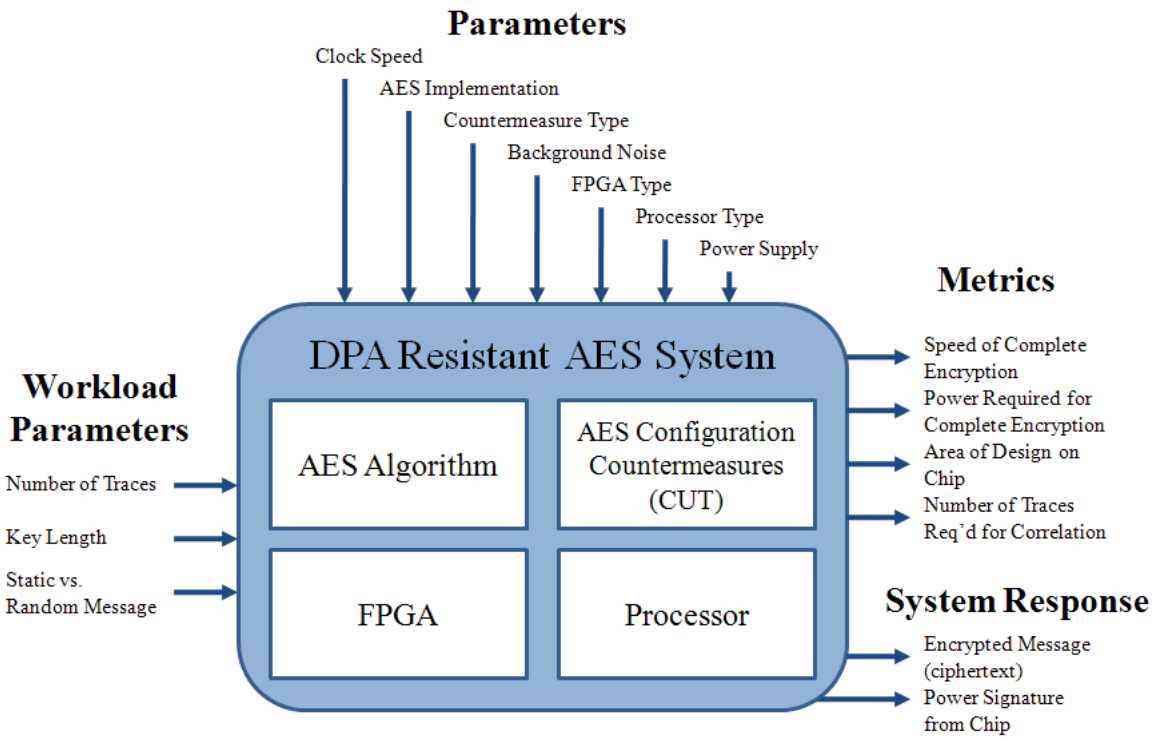

Figure 3.2: DPA Resistant AES System

3.2.2 Xilinx Virtex-5 FPGA. The Xilinx Virtex-5 FPGA is a field programmable gate array. The designs are initially coded using the Verilog hardware description language before being synthesized by the Xilinx software and transferred to the chip via a bitstream.

3.2.3 Processor. A PowerPC 440 processor core serves as a central control unit for the system. While the actual AES algorithm is hardware-based, the input and output handling is controlled by the PowerPC.

3.2.4 AES Configuration Countermeasures. The countermeasures implemented are system-level bit-balancing and randomizing the clock frequency every clock cycle. System-level bit-balancing is done by running two nearly identical AES algorithms together with one algorithm processing the correct data and the other processing the inverse of the data. This inversion is accomplished by inverting the input data and adjusting the S-Box so that all intermediate values within the modi-

fied algorithm are inverted. The result of inverted intermediate values is an inverted ciphertext. When the normal and inverted algorithms run side by side, it results in system-level bit-balancing. The randomized clock frequency is generated by using an external clock and a pseudo-random pattern generator to choose a random frequency every clock cycle during the encryption process.

3.3 System Services

The system services provided by the DPA Resistant AES System are as follows:

3.3.1 Data Encryption and Decryption. The system's primary function is the encryption and decryption of data. It either outputs the correct ciphertext or plaintext or it does not. The encryption and decryption output is verified against established software versions of AES.

3.3.2 Side-Channel Analysis Resistance. The system also provides the service of resistance to power analysis attacks. Any increase in difficulty for an attacker to determine the secret key of the system is considered successful for this research. The degree of success is measured by the increase in the number of traces required to correlate the power traces to the data versus the number of traces needed for the baseline system.

3.4 Workload

The workload parameters of the system characterize the requests for services from the DPA Resistant AES System. The parameters describe the data to be encrypted or decrypted, and in the case of AES encryption, the size of the key.

3.4.1 Number of traces. The AES algorithm only accepts input data as 128 bit blocks at a time, with the size of the complete message determining the number of traces the algorithm can produce. The larger (in number of bits) the plaintext or ciphertext is, the greater the number of traces the AES system generates. Therefore,

the number of traces, and message blocks, is directly correlated to the workload and output of the system. The number of traces equals the size of the message (in bits) divided by 128. Each iteration of the algorithm provides one power trace during power analysis. The workload is varied based on the number of traces required to establish a correlation between the power traces and the data. The number of traces required to extract the key varies between 1,000 and three million depending on the effectiveness of the current implemented countermeasure.

3.4.2 Random Versus Static Message. A workload representative of actual data should be fairly random. While there are definite patterns within plaintext, the possible number of unique plaintexts is large. Random data fed into the system is a good representation of a real world workload. For the system, random input data requires a random number generator, which reduces the speed of the system according to the time it takes to transfer and store the input data to memory every iteration. A static message only requires the system to store the message once per data collection cycle. For this experiment, random messages are used for encryption.

3.4.3 Key Length. The key length for the AES algorithm increases the computational security of the system but also changes the number of rounds required to complete an encryption. The longer the key length, the more rounds used during the encryption process. By increasing this workload parameter, the encryption execution time increases. For this experiment, the key length is set to 128 bits - minimizing the number of rounds required to complete the AES algorithm.

3.5 Metrics

The performance metrics of the DPA Resistant AES system measure system performance. They are generally effective when comparing similar systems or when measuring benefits or costs of specific designs. For this experiment, the metrics are time required to complete the one encryption process in microseconds, the power required to complete the full encryption process (measured in millivolts recorded by

the electromagnetic probe), the design area on the chip measured by the number of utilized registers, look up tables (LUTs), FFs, and BRAMS used, and the number of traces required to achieve adequate data correlation.

3.5.1 Speed of Complete Encryption Process. The time required to complete one encryption process is important for the overall speed of the system. Often, data transfers and processing measures are limited in speed by the encryption process. It is therefore important to minimize the time it takes to encrypt data. The time is measured from the instant the user sends data to the encryption algorithm and ends when the user receives the entire ciphertext back from the system. The time is measured in microseconds.

3.5.2 Power Required for Complete Encryption. It is not just stationary computers that require the use of encryption, but also small, power limited embedded systems. Whichever system it may be, it is desirable to minimize the power used by the encryption process. Power used by the encryption process is measured by taking the integral of the power trace measured by the electromagnetic probe over the entire encryption process. The electromagnetic emanations from the circuit are directly proportional to the power consumed by the circuit and are therefore a viable output to measure when determining power usage. The electromagnetic probe measures millivolts emanating from the chip. While this measurement is not a direct measurement of the power drawn by the circuit, it is useful when comparing similar circuits and evaluating against a baseline.

3.5.3 Chip Design Area. Chip area is often limited on FPGAs and the encryption algorithm is normally not the only system on the chip. It is therefore desirable to minimize the DPA resistant AES system's footprint on the Virtex-5. Area is measured by using the Xilinx XPS software to pinpoint and quantify the hardware used for the AES system. The Xilinx software provides a summary of the number of registers, LUTs, FFs, and BRAMS currently in use on the device.

3.5.4 Number of Traces Required for a Successful Attack. The number of traces required to determine the desired information measures the effectiveness of the countermeasures. An effective DPA resistant AES system maximizes this value, thereby increasing the difficulty for an attacker to determine the secret key. A baseline system determines the baseline value and different system designs are compared to this standard.

3.6 System Parameters

The system parameters affect the performance of the SUT. If changed, the variation in the system parameters is reflected in the metrics and/or response of the system.

3.6.1 AES Implementation. While the outputs and basic functionality of the AES algorithm remain the same, there are several different methods to implement the encryption system. The two main difference factors are software vs. hardware implementation, and pipelined vs. iterative implementations. Software (SW) AES is performed solely on a general purpose processor. Hardware (HW) AES performs the encryption at the gate level and is usually faster and more efficient. The speed, power usage, and area on chip all vary greatly depending on whether it is a SW or HW version of AES. Pipelined AES is faster and more difficult to evaluate using power analysis, while iterative AES runs in consecutive steps. There are varying degrees of how pipelined the code can be. This research uses an iterative HW version of AES as the core algorithm.

3.6.2 FPGA Type. There is a wide range of variation between different types of FPGAs. The Xilinx Virtex-5 FPGAs have a flip-chip design and are generally more vulnerable to power attacks on their undersides. They are built using 65 nm technology. Many newer chips are built with smaller process widths, decreasing power consumption. Changes in design, process technology, and general hardware schematics

all affect the speed, size, power consumption, and traces required for correlation of the system.

3.6.3 Processor Type. The PowerPC 440 processor is physically embedded on the Virtex-5 chip and is capable of processing the SW version of AES as well as data input, output, and system controls. Because data is processed on the PowerPC, the power signature of the processor is important for power attack prevention. Therefore, the type of processor used has influence on the system speed, power consumption, and side-channel analysis (SCA) resistance.

3.6.4 Power Supply. The power supply for the chip can minimize the variation and therefore has influence on the power consumption of the system. For this research, the laboratory grade power supply voltage is constant at the manufacturer's recommended level. The power supply may also add noise to the system.

3.6.5 Background Noise. Background noise is created by various sources, many which are located on the board itself. Because correlation is a function of the signal to noise ratio of the power traces, background noise greatly affects the number of traces required for correlation. For the SUT, background noise is minimized as much as possible.

3.6.6 Clock Speed. The clock frequency is highly variable and is directly correlated to the speed of the encryption/decryption process. Increasing the clock frequency also increases the power drawn by the circuit over a given period of time.

3.6.7 Countermeasure Types. Different countermeasure implementations all have different speed, power consumption, area, and SCA resistance characteristics. Depending on the type of countermeasure used, they can effect the performance of the system. This research determines the benefits and drawbacks of each countermeasure or countermeasure pair against the baseline system.

Table 3.1: Factors and Levels

Factors	*Levels*
Countermeasure Type	None/Random Clocking/Bit-Balancing/Combined
Number of Traces	1k/2k/3k/4k/5k/6k/7k/8k/9k/10k/20k/30k/40k/50k 60k/70k/80k/90k/100k/150k/200k/250k/300k/350k 400k/450k/500k/600k/700k/800k/900k/1,000k/1,500k 2,000k/2,500k/3,000k (No more than 5 levels ever chosen)

3.7 Factors

The factors in this experiment are the parameters that are varied to evaluate the response of the system. Table 3.1 contains the two factors of this research.

3.7.1 Countermeasure Type. The research is based on the performance of four different levels of protection: no protection, randomizing the clock every cycle, system bit-balancing, and combined random clocking and bit-balancing. Benefit is detected by an increase in the number of traces for correlation compared to an unprotected system. Drawbacks are any increase in area, power consumption, and delay of the system when compared to an unprotected system.

3.7.2 Number of Traces. Power attacks are dependent on correlation between data and the power traces and are sensitive to noise and variation within the system. To improve correlation, the number of traces currently evaluated can be increased if additional traces are available. One trace is equivalent to a message size of 128 bits. Therefore, the number of traces can be increased by increasing the message size fed into the system. The minimal number of traces recorded by each iteration of the encryption system is determined to achieve a correlation between the input data and the power traces. While there are 36 potential levels for the number of traces, a binary search during testing narrows the number of levels ever evaluated to a maximum of five [26]. Of course it may be beneficial to include all levels in the results for graphical visualization purposes.

3.8 Evaluation Technique

The evaluation technique used to determine the performance of the countermeasures is the measurement of an actual system. Both randomizing the clock frequency every clock cycle and system-level bit-balancing have been attempted in simulation. What makes this research unique is the system-level bit-balancing design, the combination of the two countermeasures, and real world implementations and results. Simulation is used initially to determine the expected outputs, but the final results are measured from a system on the Xilinx Virtex-5 FX evaluation board. Random input values are provided by the Riscure Inspector 4.2 Software to the algorithm on the board. Power traces are measured using a Lecroy WavePro 725Zi oscilloscope and a Riscure high sensitivity electromagnetic probe. A start and finish trigger is provided by software within the system, and the output data and the traces are recorded in real-time by Riscure's Inspector test software for later evaluation. A graphical representation of the evaluation setup is given in Figure 3.3. The power traces recording process begins when the Riscure software sends a pseudo-random plaintext or ciphertext value to the Virtex-5 board. When the board recieves the message, it triggers the oscilliscope to begin recording electromagnetic emanations with the EM probe. When the encryption/decryption is complete, the AES system sends the output data back to the Riscure software and once again triggers the oscilloscope to stop recording. This process is repeated until the desired number of traces are collected.

The effectiveness of the results are evaluated based on the information stored by the Inspector software. Encryption success is determined by comparing the plaintext, key, and created ciphertext to determine correct functionality. Encryption speed is measured by evaluating the power trace where the start flag is triggered until the finish flag is set. Power required is evaluated using Inspector's modules to determine the overall power of each measured power trace. Circuit area is determined by the design of the software. The resistance of the circuit against attack is determined by performing DPA attacks on the recorded power traces using simple correlation functions using MATLAB R2009b technical computing software. For each byte within the

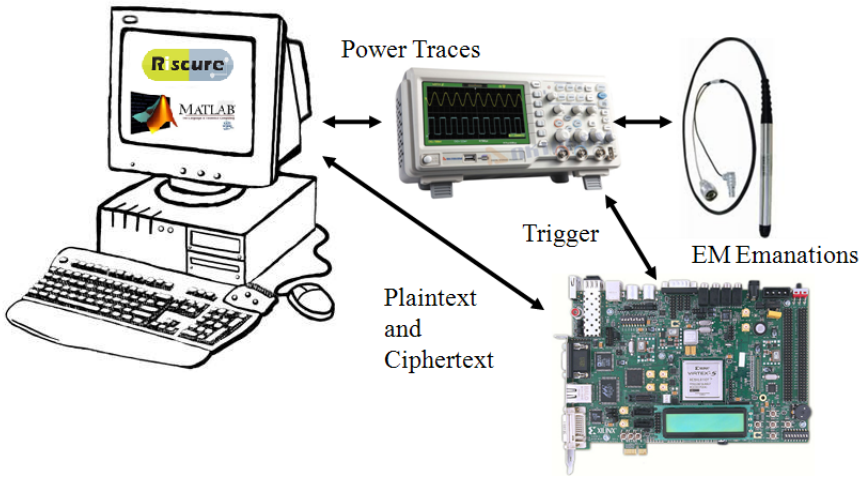

Figure 3.3: Evaluation Setup

AES key, there are 256 (2^8) potential key guesses. The final design is validated when the minimal number of traces required to provide a measurably greater correlation with the correct key guess is determined, and the required number of traces is less than an equivalent encryption system with no countermeasures. During the correlation process, there is a peak correlation for each key guess somewhere between 0 and 1. DPA is successful when the correlation between a correct key guess is greater than all other key guesses for every byte of the key. Validation also occurs by comparing the measured results to the analytical expectations of each countermeasure's improvement upon DPA resistance. The countermeasures are expected to measurably increase the number of traces required to perform an effective DPA attack.

3.9 Experimental Design

A full factorial design is conducted for this research. There are 2 factors; one has 4 levels while the other has 36 levels. To minimize the number of required tests, the number of levels can be minimized for the second factor. The objective is to determine the number of traces required for adequate correlation. The number of traces is incremented in varying steps until the total number of traces reaches three million. However, a binary search method is used. This search is accomplished by

choosing a middle value (100,000 traces to begin with) and determining if the secret key can be discovered. If so, eliminate all trace levels above the chosen trace level. If the key cannot be determined, eliminate all trace levels below the chosen trace level. Repeat by choosing a middle level until there is only one trace level left. Therefore, no more than five tests are required to determine the number of traces needed for correlation. Because of the high number of traces required for each test and the fact that correlation requires an averaging of the traces, the expected variation between tests is minimal. The purpose of the correlation is to remove the variation for all factors except the secret key. Each test is replicated three times, so there are a total of 60 experiments. A 95% confidence level is used to evaluate the data during processing of the traces. The confidence interval units are number of traces. A 95% confidence level is sufficient to determine if the final design requires more traces to evaluate than the baseline AES system with the desired level of certainty.

3.10 Methodology Summary

This chapter defines the experimental methodology of the DPA resistant AES system. The goal of this research is to evaluate AES DPA countermeasure capable of hiding the secret key from an attacker. The system under test includes the AES algorithm, processor, FPGA, and countermeasures. The two countermeasures tested include varying the clock frequency every clock cycle and performing system-level bit balancing. Performance is measured by the circuit's area, speed, power consumption, and the number of traces required to receive adequate correlation. There are a total of 60 experiments where the user varies the countermeasures used and the number of traces measured for post-processing.

IV. Design and Results

This chapter begins by discussing the experimental setup and methodology used for performing DPA on the system in Sections 4.1 and 4.2. Section 4.3 describes the development and implementation of the first countermeasure - the randomized clock. Section 4.4 explains the process of developing the bit-balancing countermeasure. The experimental results of this research are given in Sections 4.5, 4.6, 4.7. The results from the baseline AES system and each countermeasure are compared for all 4 categories (DPA resistance, area of circuit, delay of circuit, and power used). Section 4.8 summarizes the results and comparisons between all the AES systems used and/or developed in this research.

4.1 Experimental Setup of DPA attack

For this paper, a simple, iterative version of AES is used for all the experiments [27]. Power traces are collected using the Riscure Inspector side-channel test tools, including an EM high sentitivity probe, an EM Probe Station, and the Riscure Inspector software. Power traces are measured using a Lecroy WavePro 725Zi oscilloscope, and the algorithm is defined in Verilog and loaded onto a Xilix Virtex-5 FX FPGA evaluation board. A visual overview of the experimental setup is given in Figure 4.1.

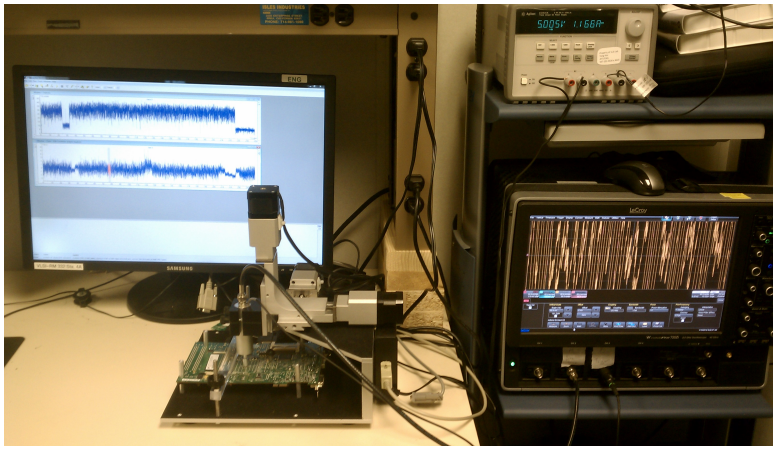

Figure 4.1: Experimental Setup

The frequency of the bus is set to 100 MHz. Therefore, to minimize unwanted noise, the traces are filtered using a bandpass filter between 90 and 210 MHz. Each trace uses the same key given in Equation 4.1, and the input data is randomized for every trace using Java's randomize function. An example of a power trace is given in Figure 4.2. As is visible, the actual AES encryption occurs between 1.1 and 1.7 μsec.

$$\begin{aligned} \texttt{Key} &= \texttt{0001 0203 0405 0607 0809 0A0B 0C0D 0E0F} \\ \texttt{Last Round Key} &= \texttt{1311 1D7F E394 4A17 F307 A78B 4D2B 30C5} \end{aligned} \tag{4.1}$$

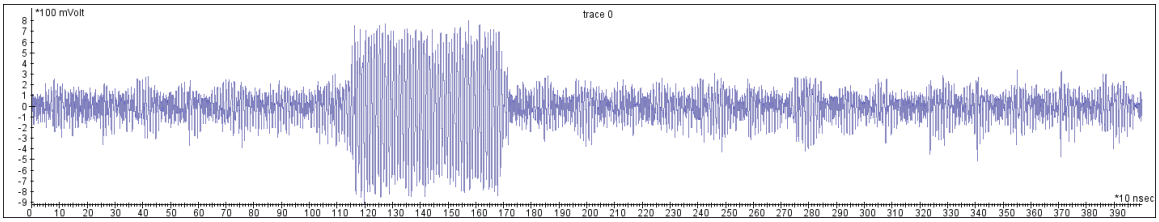

Figure 4.2: Power Trace of Hardware AES

To maximize signal from the probes, the Virtex board is flipped over and the probes are situated on the bottom of the board. The Virtex-5 FPGA contains an array of internal power coupling capacitors as seen in Figure 4.3. Because these capacitors provide primary internal power to the FPGA, they tend to leak significant amount of EM information. Because the AES algorithm is the only system currently running on the chip, the capacitors are an optimal feature of the board to monitor with the EM probe.

The power traces are recorded using the oscilloscope and stored using the Riscure inspector software. Also, the corresponding plaintext and ciphertext are stored with each power trace. For this research, both the plaintext and ciphertext are stored along with the traces and the key is already known. In practice, an adversary would not know the key, and would only need one (plaintext or ciphertext) side of the data. In fact, the adversary is often motivated to determine the key to uncover

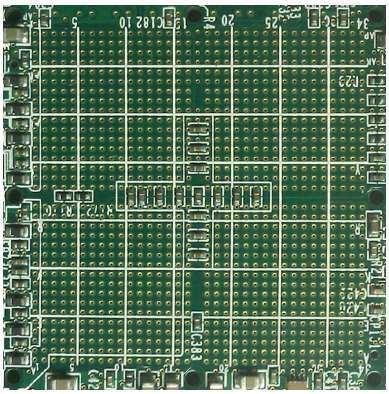

Figure 4.3: Capacitors on the bottom of the Virtex-5 FPGA

any plaintext or ciphertext they do not know. The attacks used in this research work equally well with the normally missing pieces of data, but the added information is helpful in determining effectiveness. The Riscure software outputs the data for each trace to the console. An example output is shown in Figure 4.4. Along with the input data and key, Riscure sends commands to the system to inform whether the key or plaintext is being sent, and also when to start the algorithm. The console output is convenient to confirm correct algorithm functionality, especially after the system is modified with countermeasures.

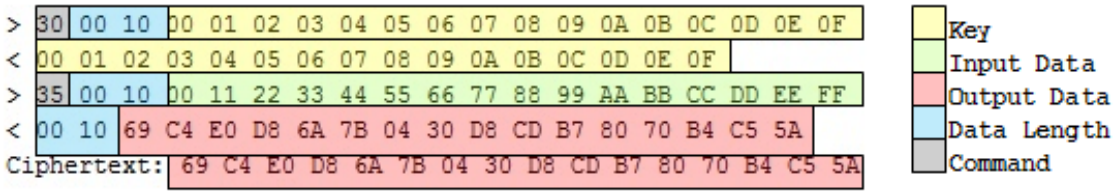

Figure 4.4: Riscure Inspector's Console Output for AES Encryption

During DPA attacks, the plaintext is randomized but the key remains the same. The randomization is performed in Riscure Inspector's Java user modules using Java's randomize function.

4.2 DPA Attack Methodology

Performing a successful attack requires two processes: data collection and data evaluation/processing. During data collection, it is important to pinpoint location(s)

on the board where the AES algorithm leaks a strong EM signal. Data evaluation and processing attempts to minimize any noise within the signal and then compare the traces to a hypothetical power model to determine correlation with key guesses.

4.2.1 Data Collection. To determine the best location on the chip for data collection, the Riscure EM Probe Station is used. The software that works with the station takes traces in a tiled pattern over the surface of the chip. It can then graphically depict correlation strength of the power traces to the data. An example of this is shown in Figure 4.5.

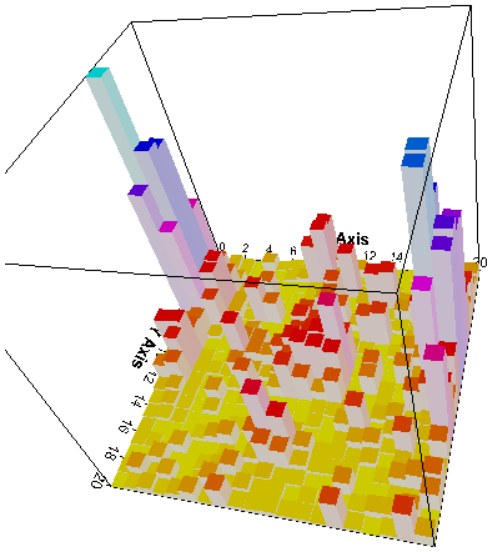

Figure 4.5: Correlation Strength at Different Locations on the Chip

While the graphical output similar to the one in Figure 4.5 is helpful, a visual scan of the different traces across the chip was most effective in determining where the strongest EM leakage is for this research. Once an ideal probe location is determined, traces can be collected.

Because the frequency of the system is set to 100 MHz, the sampling frequency is at 1 GHz. This allows an accurate picture of the EM emanations without creating an overwhelming amount of data to be stored and analyzed. It also allows for ease of processing within the Inspector software as opposed to a 200 MHz sampling frequency.

From here, several million traces are collected and stored using the Riscure Inspector software.

4.2.2 Data Processing and Evaluation. Once the traces have been collected, a band-pass filter is applied between the reasonable frequencies of the circuit. For the baseline system running at 100 MHz, a bandpass filter within the Inspector software is set between 90 and 210 MHz. While a narrower frequency band would slightly increase the signal to noise ratio, a wider band was selected to improve processing performance for the Riscure software. After the traces are filtered, they are aligned by shifting the data left or right so that the entire AES process is synced on the time scale for all EM traces. Finally, all data before and after the AES run (as seen by the apparent voltage increase in Figure 4.2) is trimmed, leaving only the AES algorithm to be evaluated.

Once the traces have been processed, they can then be compared to power models as described in Chapter 2. To effectively attack the AES algorithm, several different models were used on both the first and last round of the algorithm. The reason only the first and last rounds of the AES algorithm are attacked is because of the MixColumns function. The outputs bytes of the other modules within the AES algorithm only depend on one input byte, and therefore also only one key byte [2]. However, the output bytes of MixColumns depend on four input bytes. This dependency increases the complexity of intermediate value calculation beyond reasonable levels. Instead of having only one byte of the key to evaluate, an attacker would have to evaluate four bytes of the key. This problem only compounds as the number of rounds increases, therefore exponentially increasing the number of key guesses as the number of rounds increases. Instead, only the data before the first MixColumns function or after the last MixColumns function is evaluated. Figure 4.6 shows how the iterative version of AES used in this research works. Each round is evaluated in one clock cycle [27].

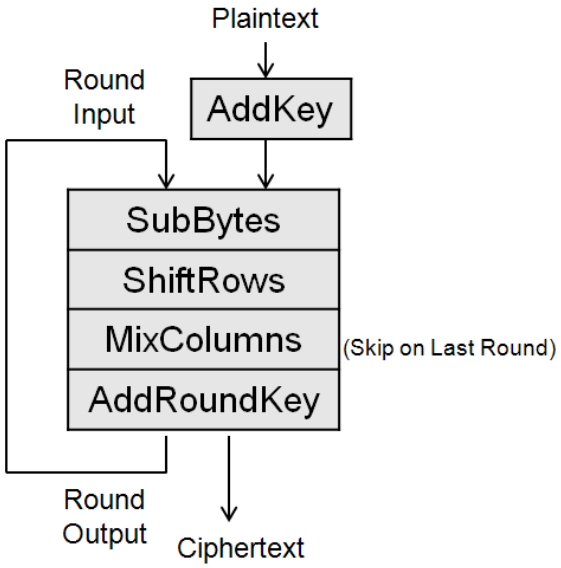

Figure 4.6: Iterative Version of AES

Because HW and HD models evaluate the difference in power between a 0 and a 1, it is desireable to attack where intermediate values are temporarily stored. Based on how the iterative version of AES in this research works, an attack appears to be most effective on data before or after the entire round (because the data must be saved and loaded into the next round). However, to fully determine the most effective HW or HD attack, four intermediate values were chosen for HW, and three bit changes were chosen for HD.

It is important to consider the difference between iterative and pipelined versions of AES and whether the countermeasures are applicable to both. In general, the main benefit to using an iterative version of AES is the area required for the circuit [28]. Figure 4.6 shows an iterative approach to AES. The algorithm completes after n clock cycles, where n is the number of rounds. The next block of input data cannot be evaluated until the previous has completed. On the other hand, pipelined AES unrolls the rounds, allowing each round to be processing a different block of input data concurrently. This unrolling speeds up the process, but generally requires more memory and space for the circuit [28]. Figure 4.7 shows a graphical view of the pipelined approach.

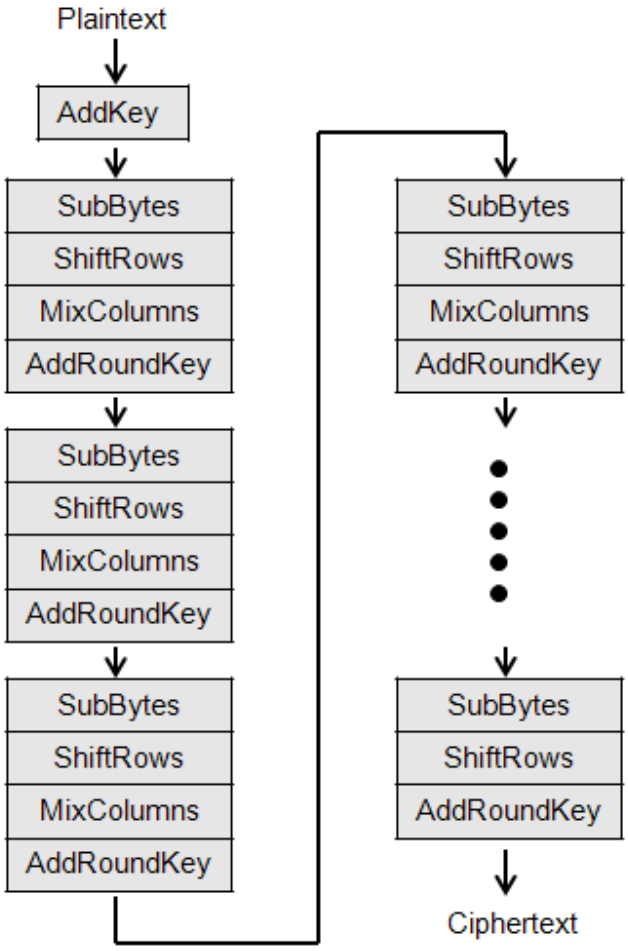

Figure 4.7: Pipelined Version of AES

The main difference when performing DPA on the two types of AES is the noise level. For iterative AES, each round and therefore each intermediate value can be isolated to specific points along the time scale. Also, no other data is being evaluated at that specific point in time, therefore minimizing the background noise. For pipelined AES, up to 10 rounds can be evaluated in one clock cycle (for 128 bit AES). Therefore, the power trace will contain the desired intermediate value plus nine other intermediate values. In this case, the other rounds act as noise. However, bit-balancing and random clocking work for both implementations. Because the noise level for iterative AES is lower, it has been chosen for this research. By taking this approach, it minimizes the number of traces required to attack the implementations, saving both time and resources. However, the two countermeasures proposed in this research should work effectively for pipelined AES as well.

When performing HW and HD attacks on the circuit, seven different intermediate values or intermediate value pairs were chosen:

1. HW: First Round Before SBOX
2. HW: First Round After SBOX
3. HW: Last Round Before SBOX
4. HW: Last Round AFter SBOX
5. HD: First Round Between Values Before and After SBOX
6. HD: Last Round Between Values Before and After SBOX
7. HD: Last Round Between Values Before SBOX and Round Output

When evaluating the last round, the original key cannot be used, but instead the last round key as given in Equation 4.1.

The correlation is performed using MATLAB software. After evaluating each modeling technique on the different intermediate values, it appears the HD on the last round between the intermediate value before the SBOX and the ciphertext (choice 7) provides the highest correlation. This result makes sense because if the input and

output data to the rounds are being stored in the same register, then the values in those registers would change every clock cycle.

4.3 Randomized Clock Countermeasure Development

In order to randomize the clock, a 16-bit linear feedback shift register (LFSR) is polled at the end of each clock cycle to pseudo-randomly choose between four clock cycle lengths. The LFSR is seeded with a new pseudo-random value at the beginning of each encryption process to ensure variance between algorithm runs. Optimally, the clock should vary between 100% and 50% of the original frequency. However, for this research, the clock varies between 33% and 16.7%. Initially, the idea was to simply increase the clock frequency on the Virtex-5 bus by 300%, but the clock was already set near its maximum frequency (100 MHz with a limit of 125 MHz). An external clock could be added to the system, but that is out of the current scope of this research. A graphical depiction of the random clock module is given in Figure 4.8.

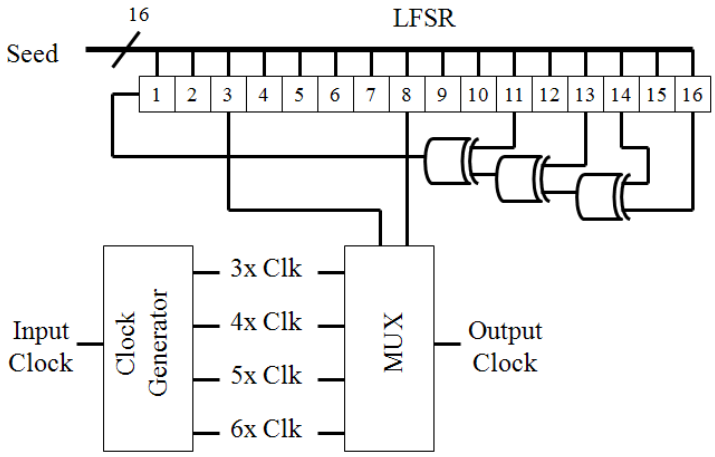

Figure 4.8: Random Clock Countermeasure Module

The reason the clock period varies between 3x to 6x longer than the original clock is because of the difficulty of working with partial clock cycles. The new clock period needs to be divisible by the original clock period. Because 4 different clock frequencies are desired for this research, the fastest clock periods that can be chosen that create clock cycles that vary by 100% are 3x, 4x, 5x, and 6x. The clock was

defined in Verilog and simulated in ModelSim. Simulation results of the randomized clock module can be seen in Figure 4.9. With a LFSR seed of 0x7575, it is clear that the clock cycles vary as expected.

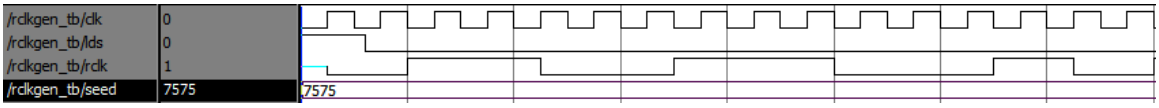

Figure 4.9: Random Clock Simulation using ModelSim

After simulation, the module is added to the AES system and tested on the Virtex-5 board. The correct output is confirmed, as expected. A power trace of the algorithm running on a randomized clock is shown at the bottom of Figure 4.10. It is compared against the baseline power trace to emphasize the differences in EM amplitudes. In fact, the power signature varies greatly for each iteration when there is a randomized clock, making each trace relatively unique. This confirms that the clock is being randomized. With the power traces being anything but uniform, trace alignment should be nearly impossible to accomplish.

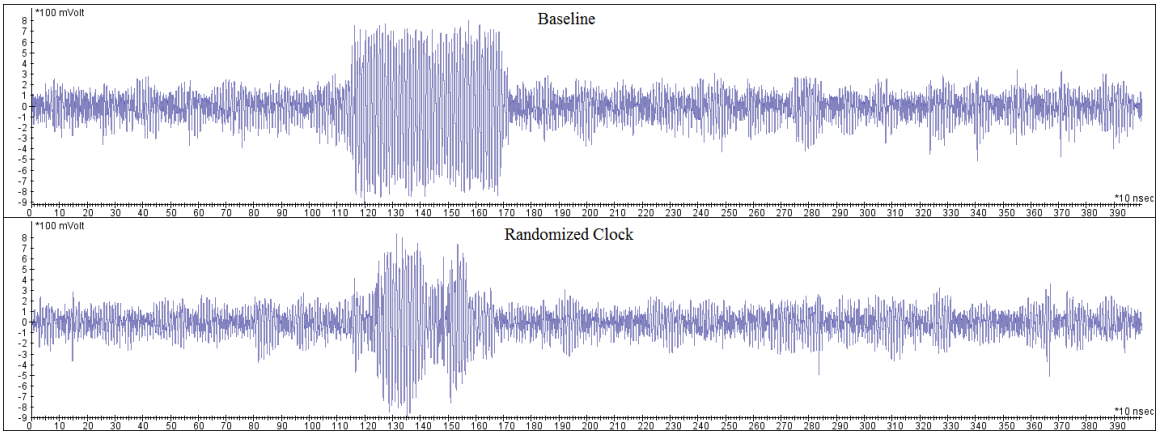

Figure 4.10: Power Trace with Random Clocking vs Baseline

This randomization of the power traces is both essential and expected for effective DPA resistance. To perform DPA, alignment of the traces is attempted. The traces are aligned at the beginning of the encryption process in order to hopefully minimize the effect of the random clock. However, as seen in the results of a DPA attack on AES with a randomized clock in Section 4.5, the design remains highly resistant to

general DPA attacks. An attacker would have to pinpoint the exact clock cycle they plan on correlating against, and even then they would need four times the number of traces that they did originally. Therefore, randomizing the clock trace makes the circuit at least 4x more difficult to attack, and that is only given if an attacker can pinpoint an exact clock cycle (and they know which clock cycle to pinpoint). Both assumptions are very difficult to accomplish, increasing the difficulty of performing DPA even further.

4.4 System-Level Bit-Balancing Countermeasure Development

For this research, a second countermeasure using a system-level bit-balancing design was developed to obfuscate the circuit against HW and HD attacks. The bit-balancing design is similar to [1], but differs in several key areas. First, the key remains unchanged as opposed to [1]. Second, the key schedule is left untouched and the round keys remain the same. Third, the input data is inverted before entering the AES algorithm, unlike the design proposed by Ambrose et al. The design for the inverted circuit is given in Figure 4.11. Due to their linearity, the AddRoundKey, ShiftRows, and MixColumns components of the AES algorithm retain the inversion of the data (inverted input = inverted output from the components). However, the main problem comes from the non-linear SubBytes component. Finally, this design includes HD resistance, while [1] does not.

In order for the output data from the SBOX to be inverted when provided an inverted input, it is helpful to look at how the SBOX handles the data. For the specific AES algorithm used in this research, a lookup table is used for the SubBytes function. The SBOX is shown again below in Table 4.1.

Each cell is indexed by the input value. For example, an input of 0x03 would output 0x7b. Because the output needs to be inverted, each value within the SBOX is replaced with its inverted value (0x7b would be replaced with 0x84) for the inverted AES algorithm. However, one more step must be taken to ensure an inverted output. Because the input data will be inverted, indexing must also be inverted. Therefore,

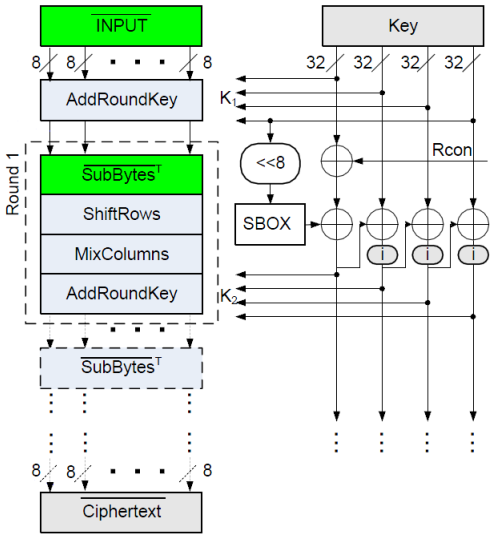

Figure 4.11: System Level Bit-Balancing Design

what was in the top left cell must now be moved to the bottom right, etc. Once this index "rotation" is accomplished, the new SBOX is ready to output inverted data given an inverted input. The new SBOX for the inverted circuit is shown in Table 4.2. Note that 0x84 (the inverse of 0x7B) is now indexed by 0xFC, the inverse of the original index 0x03.

Using the new SBOX, complete intermediate value inversion can be achieved for the entire algorithm. To confirm functionality, the design is simulated in ModelSim. The results of the original algorithm are shown in Figure 4.12, while the complete run is given in Appendix A in Figure A.1. The inverted algorithm is seen in Figure 4.13, while the complete run is seen in Appendix A Figure A.2.

It is important to note that the intermediate values of the inverted AES algorithm are inversions of the intermediate values for the original algorithm. Table 4.3 gives the complete values of all the intermediate values that appear for the AES algorithm chosen for this research and its inverted version.

However, inverting all the intermediate values only protects against HW attacks and not HD attacks. The HD between intermediate value 1 and intermediate value 2

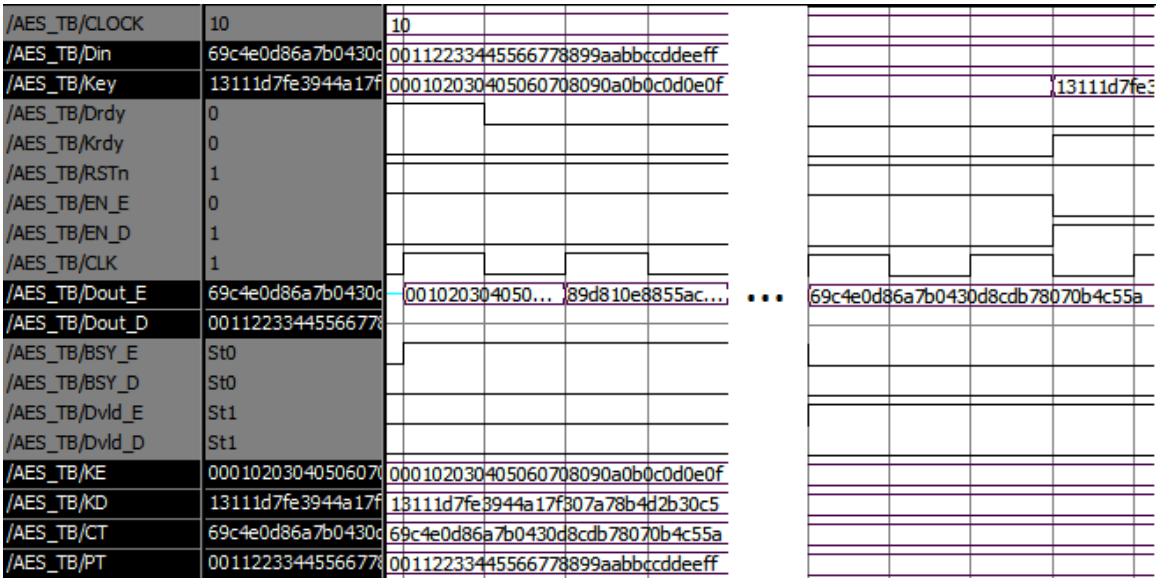

Figure 4.12: Simulation of AES Algorithm Using ModelSim

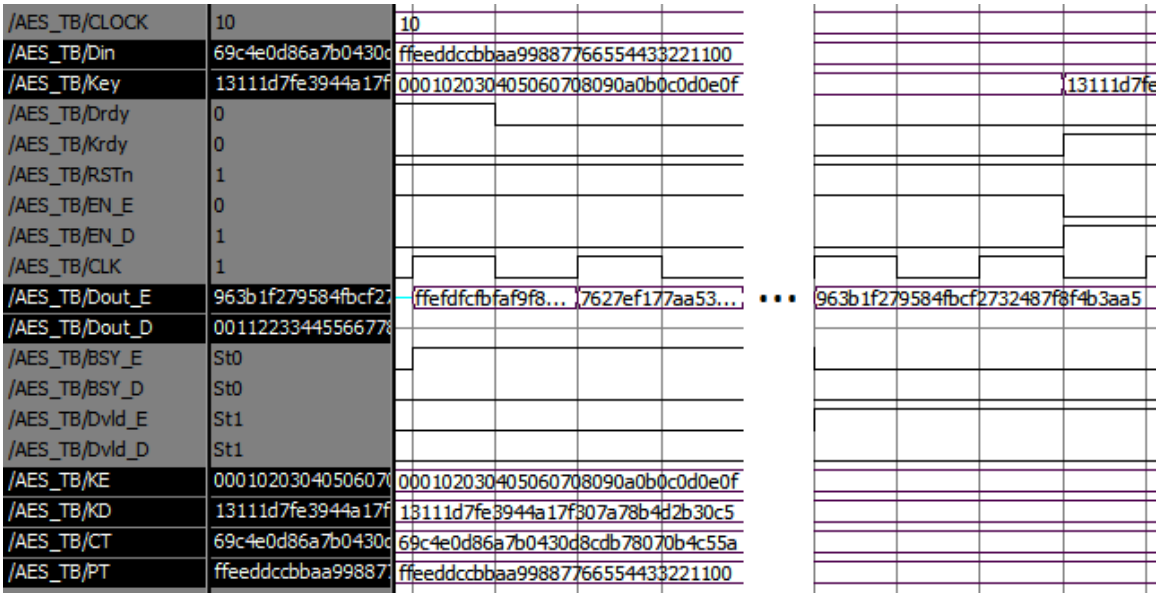

Figure 4.13: Simulation of Inverted AES Algorithm Using ModelSim

Table 4.1: SBOX

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

Table 4.2: Inverted Rotated SBOX

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

happens to be the same as the HD between inverse intermediate value 1 and inverse intermediate value 2. An example of this can be seen in Equation 4.2.

Table 4.3: Intermediate Values of Bit-Balancing Design

Original Intermediate Values	Inverted Intermediate Values
00102030405060708090a0b0c0d0e0f0	ffefdfcfbfaf9f8f7f6f5f4f3f2f1f0f
89d810e8855ace682d1843d8cb128fe4	7627ef177aa53197d2e7bc2734ed701b
4915598f55e5d7a0daca94fa1f0a63f7	b6eaa670aa1a285f25356b05e0f59c08
fa636a2825b339c940668a3157244d17	059c95d7da4cc636bf9975cea8dbb2e8
247240236966b3fa6ed2753288425b6c	db8dbfdc96994c05912d8acd77bda493
c81677bc9b7ac93b25027992b0261996	37e98843648536c4dafd866d4fd9e669
c62fe109f75eedc3cc79395d84f9cf5d	39d01ef608a1123c3386c6a27b0630a2
d1876c0f79c4300ab45594add66ff41f	2e7893f0863bcff54baa6b5229900be0
fde3bad205e5d0d73547964ef1fe37f1	021c452dfa1a2f28cab869b10e01c80e
bd6e7c3df2b5779e0b61216e8b10b689	429183c20d4a8861f49ede9174ef4976
69c4e0d86a7b0430d8cdb78070b4c55a	963b1f279584fbcf2732487f8f4b3aa5

```
First Intermediate Value:  0x86 (1000 0110) HW = 3
Second Intermediate Value:  0x3F (0011 1111) HW = 6
Hamming Distance Between Values:  5

Inverted First Value:  0x79 (0111 1001) HW = 5
Inverted Second Value:  0xC0 (1100 0000) HW = 2
Hamming Distance between Inverted Values:  5
```
(4.2)

This vulnerability is a problem, especially since HD attacks are commonplace and relatively easy to perform. The system-level bit-balancing design for this research also differentiates itself by including features to resist HD attacks along with its HW resistance. In order to protect against HD attacks, gate level pre-charging has been used in the past [5, 29, 30]. However, this research uses system-level bit-balancing, so system-level pre-charging is needed. In order to accomplish the pre-charging, it is important to understand where the AES implementation is most vulnerable. Because each round occurs in one clock cycle, the intermediate values between clock cycles are

most vulnerable to HD attacks. To minimize the vulnerability, 10 buffer cycles are added between the rounds to clear the intermediate values within the rounds. This system-level precharging is accomplished by sending a 0 as the input data and key into the round. This buffering allows for the HW bit-balancing to effectively prevent against HD attacks as well, as seen in Equation 4.3.

```
First Intermediate Value:  0x86 (1000 0110) HW = 3
            Buffer Value:  0x00 (0000 0000) HW = 0
  Hamming Distance Between Values:   3
Second Intermediate Value:  0x3F (0011 1111) HW = 6
  Hamming Distance Between Values:   6

Inverted First Value:  0x79 (0111 1001) HW = 5
            Buffer Value:  0x00 (0000 0000) HW = 0
  Hamming Distance Between Values:   5
Inverted Second Value:  0xC0 (1100 0000) HW = 2
  Hamming Distance between Values:   2
```
(4.3)

Of course, the output of the rounds must be stored between clock cycles while the system is clearing the intermediate values. This register storage creates a vulnerability. To minimize the potential for an attacker to perform HD correlation on the registers storing the intermediate values between rounds, an extra feature is added to the design. An LFSR is connected to a multiplexer, and the outputs of the rounds are stored in 1 of 4 locations randomly chosen by the LFSR. In this way, there is only a 25% chance of the HD recorded by the EM probe matching the data. Therefore, the HD leakage is effectively minimized. The final bit-balancing design is simulated

in ModelSim to confirm functionality. The inverted module simulation is shown in partial form in Figure 4.14, and complete in Figure A.3 in Appendix A.

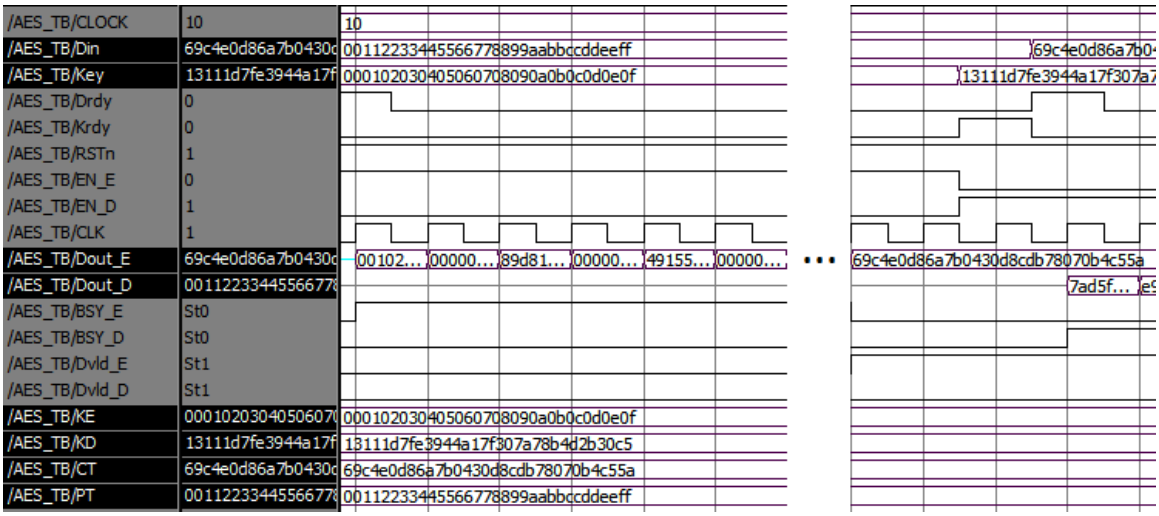

Figure 4.14: Simulation of AES Algorithm with HD Resistance Using ModelSim

As seen in Figure A.3, 10 extra clock cycles are added to the design. After simulation, the design is loaded to the Virtex-5 board. A power trace of the system-level bit-balancing design is shown on the bottom of Figure 4.15.

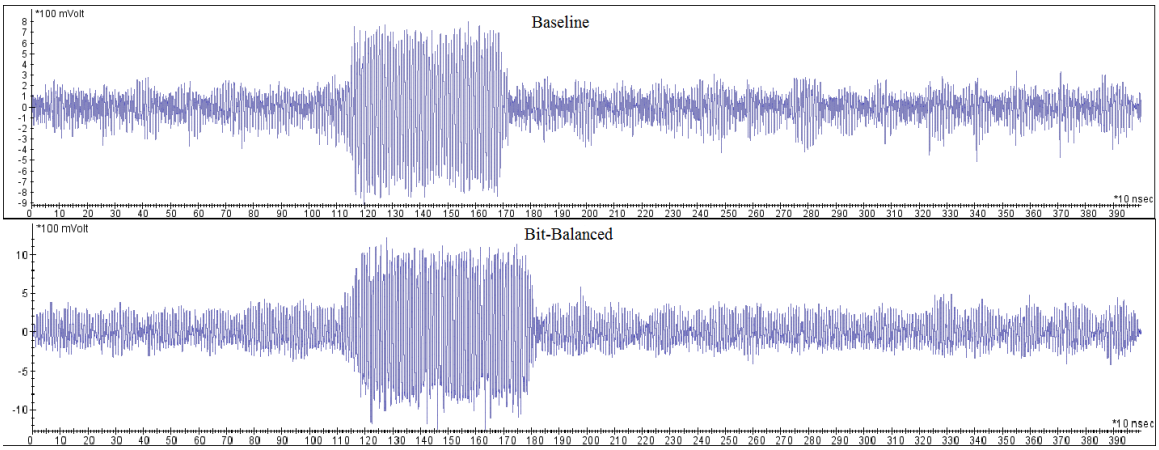

Figure 4.15: Power Trace of Hardware AES with System-Level Bit-Balancing

Compared to the power trace of the baseline design, it is obvious that the countermeasure uses more power and adds a small amount of delay to the system. However, the overall shape remains the same.

Finally, unlike the design in [1], both the normal and inverted versions of the AES algorithms are implemented on the same chip and their components jumbled together according to Xilinx's place and route software preferences. This prevents an attacker from differentiating each algorithm individually. Preventing the separation of the two algorithms is critical in order to achieve effective system-level bit-balancing.

4.5 DPA Resistance Results

While area, power, and speed of circuit are all important, the main focus of this research is to provide DPA resistance for the AES algorithm. DPA resistance can be defined as the difficulty of determining the correct key from the power traces. Because there are many factors that affect DPA effectiveness, only one factor, the number of traces collected, is varied so a comparison can be made. The rest of the factors (probe used, filtering techniques, power model used, etc.) remain constant. The four implementations tested are AES without countermeasures, AES with random clocking, AES with system-level bit-balancing, and AES with both countermeasures combined. The AES without countermeasures is used as the baseline model for comparison.

4.5.1 DPA Results on Baseline AES. Following Table 3.1, DPA is performed using a varying number of traces. The power model used for correlation is the HD model, and the intermediate values observed are the HD between the input of round 10 and the ciphertext. The first byte of the key is attacked - or in this case the first byte of the last round key. Due to the properties of key expansion, we can assume that by knowing the last key round the original key can easily be determined [31]. For this research, DPA is considered successful if correlation for the correct key guess is higher than any other key guess.

For the baseline AES system, correlation with the correct key guess occured with 500,000 traces (or more). Figure 4.16 shows the maximum correlation observed for each key guess for 600,000 traces. The correct key guess (19) has the highest spike, and is therefore the most reasonable choice. It is important to note that because

MATLAB does not index arrays starting with 0 but instead at 1, the key guess on the MATLAB graph appears at 20 (19+1).

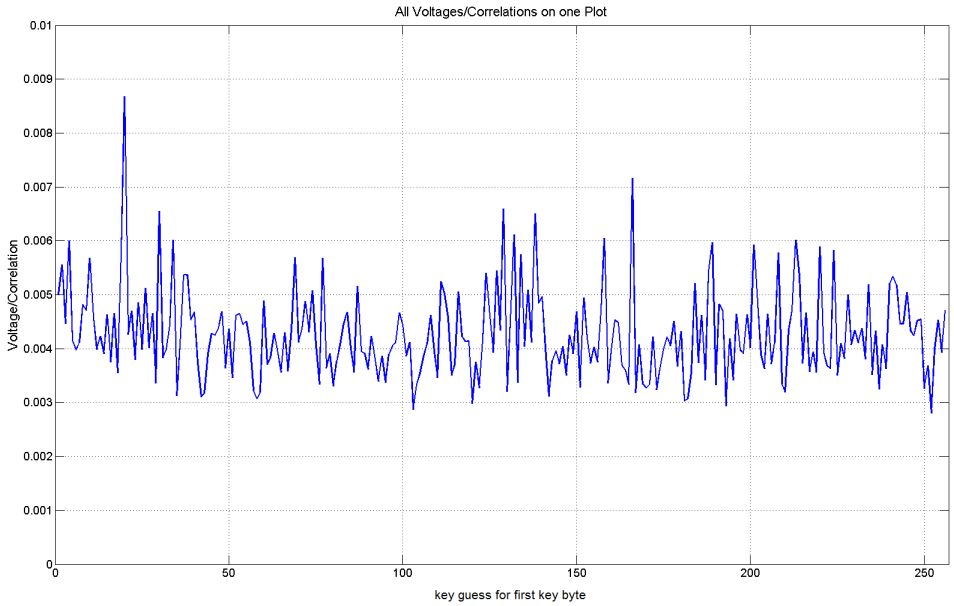

Figure 4.16: Max Correlation of all 256 Key Guesses for Baseline AES for 600,000 Traces

In order to determine that 500,000 traces are needed to successfully attack the baseline AES system, several DPA attacks are performed with varying number of traces analyzed. Figure 4.17 gives a graphical view of the correlation between the traces and the power model. Because correlation is achieved, no more than 900,000 traces are evaluated.

The average noise level is the average correlation for all 256 key guesses. The peak false positive is the highest correlation that is not the correct key guess. As it is clearly seen, DPA is successfully performed when the number of traces is greater than or equal to 500,000 traces, as indicated when the correlation of the correct guess exceeds the peak false positive.

4.5.2 DPA Results on Random Clocked AES. Now that it has been determined that the baseline AES system can be successfully attacked using 500,000

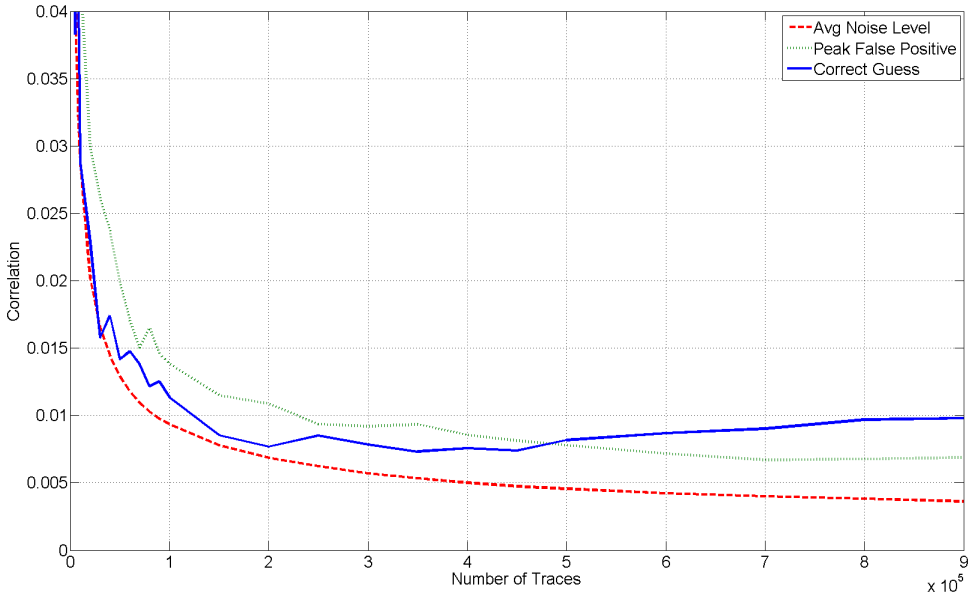

Figure 4.17: DPA Results Based on Number of Traces for Baseline AES

traces, the objective for the countermeasures is to require more than 500,000 traces to attack if all other factors remain the same.

The randomized clock changes the clock frequency for every round of the algorithm and makes alignment of the traces incredibly difficult. Trace alignment is attempted using Riscure Inspector's alignment software, but even then there is misalignment. Up to three million traces were collected. The correlation of each key guess for three million traces is shown in Figure 4.18.

From the graph, it is not apparent that the correct key guess is 19. Randomizing the clock appears to effectively protect the circuit from DPA attacks. Figure 4.19 gives a graphical view of the correlation between the traces and the power model as the number of traces varies.

The correct key guess never stands out above the noise floor. This obfuscation could be remedied by aligning the exact clock cycle being attacked. Even then, the traces would have to be split into four groups for the four different potential clock frequencies. Therefore, even if an attacker were able to pinpoint and align the traces

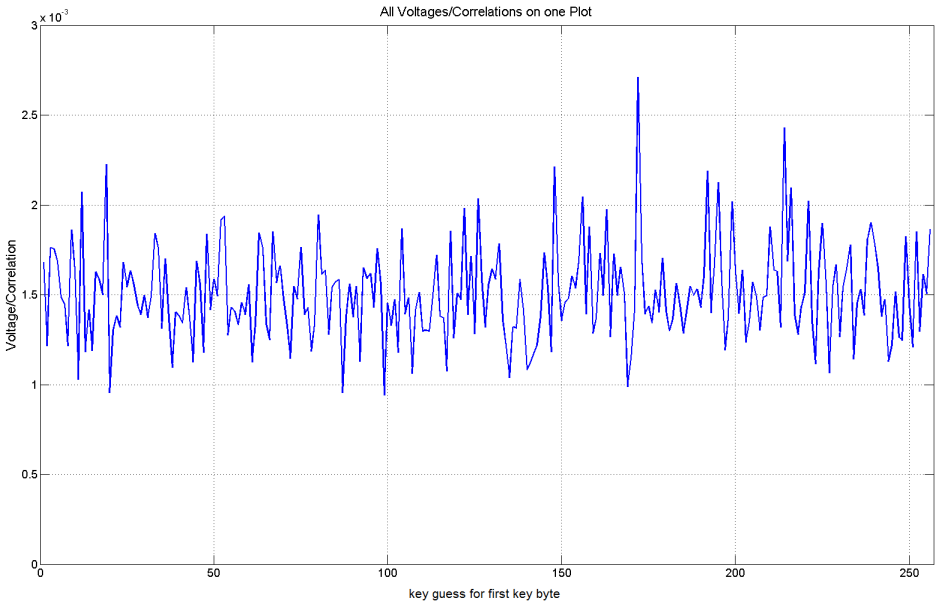

Figure 4.18: Max Correlation of all 256 Key Guesses for Randomly Clocked AES for 3 Million Traces

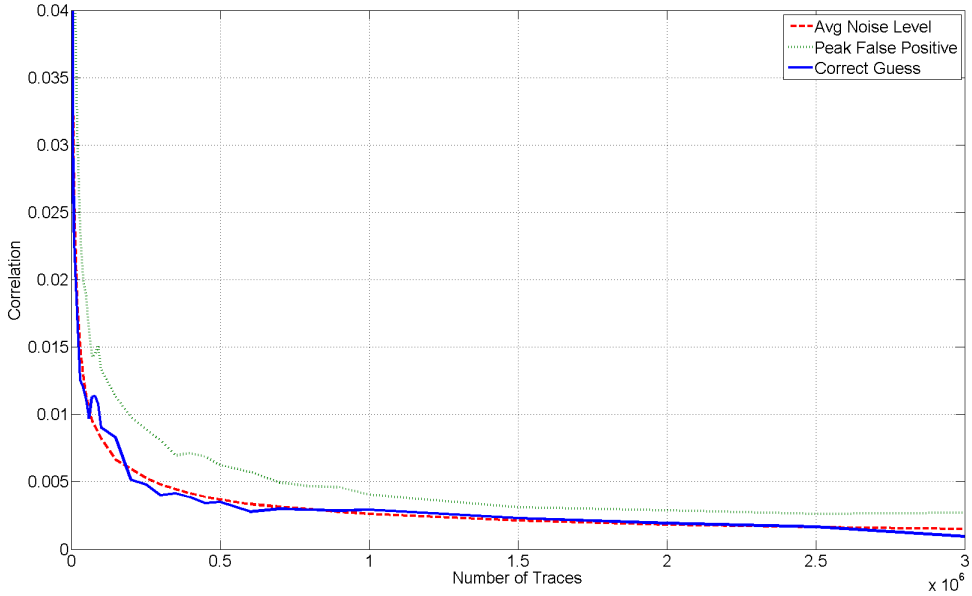

Figure 4.19: DPA Results Based on Number of Traces for Randomly Clocked AES

on the exact clock cycle being evaluated, the attacker would still need four times as many traces to determine the correct key than with the baseline AES system.

4.5.3 DPA Results on System-Level Bit-Balanced AES. For attacking the system-level bit-balanced system, all factors are kept the same and only the number of traces varies. This attack style consistency works because the bit-balancing is designed with HW attacks in mind, but can also protect against HD attacks. Because the system is specifically designed for HW balancing, HD attacks have the greatest potential for success since the intermediate values after each round are stored on the chip. However, the randomized register storage implementation is designed to mitigate this risk.

The objective is to increase the number of traces required to successfully attack the circuit. Correlation with the correct key guess never becomes apparent even after 3 million traces are collected. Figure 4.20 shows the maximum correlation observed for each key guess for three million traces. The spike seen in Figure 4.20 is a false positive key guess of 15.

Figure 4.21 gives a graphical view of the correlation between the traces and the power model as the number of traces increases. Up to three million traces are evaluated.

Even with a large amount of traces, it is clearly seen that the correlation for the correct key guess does not stand above the noise floor. False positives effectively mask the correct key guess from standing out. Like the randomized clock implementation, it is clear that an attacker would need significantly more than three million traces to successfully attack the bit-balanced system using the standard DPA attacks.

4.5.4 DPA Results on Random Clocked and System-Level Bit-Balanced AES. The combination of both countermeasures is easily done due to their relative independence. The system is mainly composed of the system-level bit-balancing core,

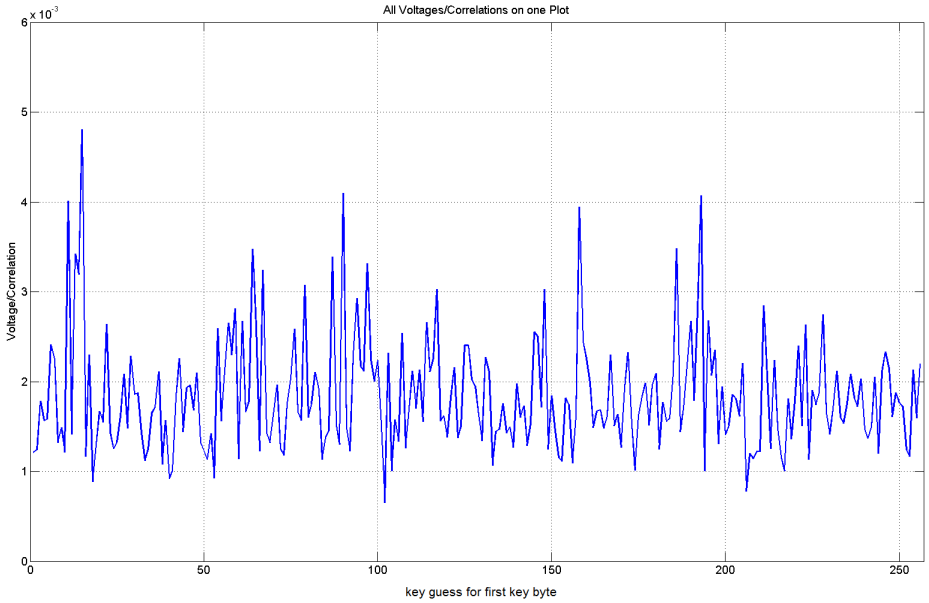

Figure 4.20: Max Correlation of all 256 Key Guesses for Bit-Balanced AES for 3 Million Traces

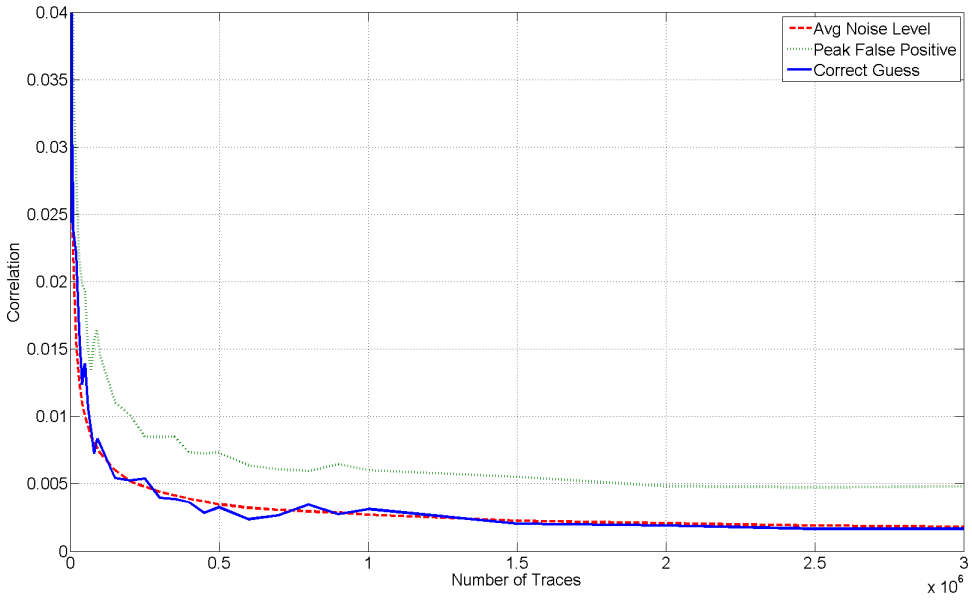

Figure 4.21: Max Correlation of all 256 Key Guesses for Bit-Balanced AES

but now with a pseudo-random clock. Figure 4.22 shows the correlation for each key guess after three million traces are collected and evaluated.

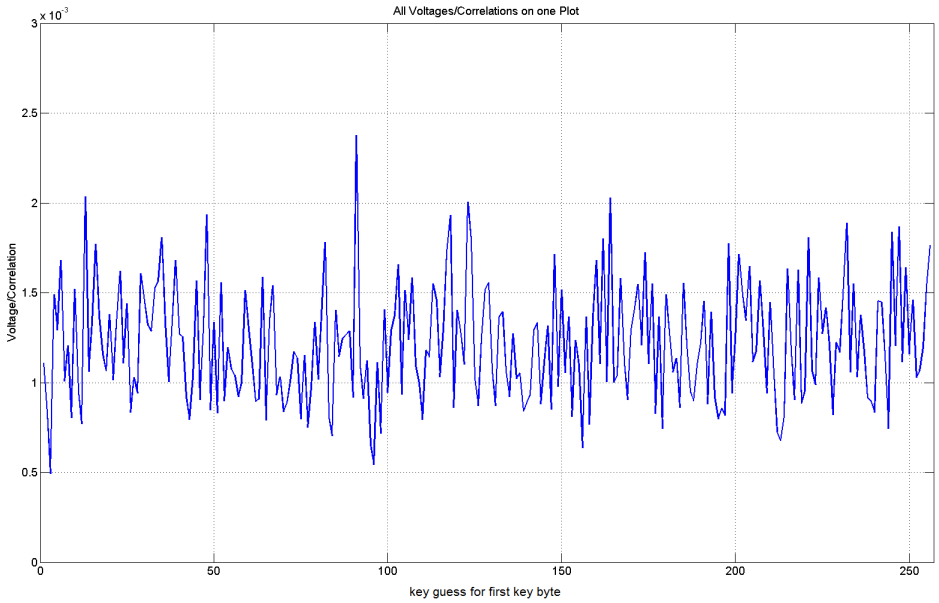

Figure 4.22: Max Correlation of all 256 Key Guesses for Combined AES for 3 Million Traces

As expected, there is no clear indication that the correct guess is 19. Figure 4.23 gives a graphical view of the correlation between the traces and the power model as the number of traces increase to three million. As expected, the correct key guess never separates itself from the noise floor. The combination of both countermeasures provides effective resistance for the AES circuit against the standard power attacks.

4.6 Area and Power Used Results

While DPA resistance is the main focus, it is also important to minimize size of the circuit and the power drawn by the circuit. By doing this, the cost of adding the countermeasures to the original AES algorithm can remain within reasonable levels for actual use in the field.

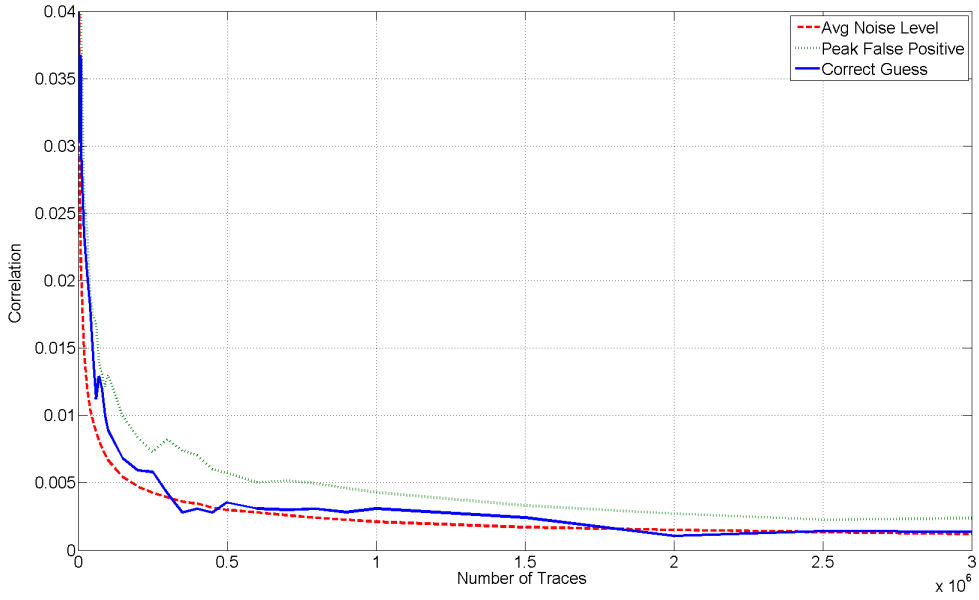

Figure 4.23: Max Correlation of all 256 Key Guesses for Combined AES

4.6.1 Area of Circuits on the Chip. Because the design is simulated and tested on an FPGA, the AES component is built using look-up-tables (LUTs), flip-flops (FFs), and BRAMs. The designs are built using Xilinx Platform Studio, and the output files include a summary of the design size on the chip. Table 4.4 details the number of components used in the design of each circuit.

Table 4.4: Area of the Different Designs on the FPGA Board

	FF Used	**LUTs Used**	**BRAMs Used**	**Slice Registers**	**Slice LUTs**	**Occ. Slices**	**% Regs. and % LUTs**
Baseline	850	1,131	5	6,133	5,834	3,281	13% 13%
Random Clock	870	928	5	6,153	5,620	3,596	13% 12%
Bit-Balancing	3,337	3,969	10	8,620	8,674	5,249	19% 19%
Combined	3,357	3,376	10	8,640	8,062	5,084	19% 17%

The baseline AES system is used as a reference point. Comparing, the addition of the randomized clocking countermeasure has an insignificant effect on the size of the

system on the chip. This minuscule increase in area is expected since the randomized clock design is minimal. In fact, it may be said that the circuit area is now less, but this is most likely due to the fundamental change in the circuit's handling of the clock. However, the bit-balancing design is almost 3.2x greater than the baseline system. A significant portion of the increased area comes from the added registers needed to randomize the storage of the intermediate values between rounds.

The combined circuit is similar in size to the bit-balanced design, as expected. It is around triple the size of the original circuit.

4.6.2 Power Consumed by Circuits on Chip. Calculating power consumed by the FPGA poses challenges when compared to an Application Specific Integrated Circuit (ASIC) design. An FPGA contains reconfigurable logic blocks containing registers and look-up-tables (LUTs) that are reconfigured for each design. However, the power consumed by the FPGA board is less affected by the size of the circuit than for an ASIC design because the logic blocks are placed during manufacturing and remain on the chip whether they are a part of the design or not. Also, the static power of the circuit is mostly dependent on the FPGA itself and not the circuit design [32]. Of the two types of power consumption, static and dynamic, dynamic power makes up most of the total power used by the circuit [32]. Because of this fact, switching activity on the board becomes important because it is a major factor in determining the dynamic power consumed [33]. Also, results show that most of the power is dissipated by on-chip long-wire connecting the different functional blocks of the design [33]. Taking in all the factors of power on an FPGA, it can be concluded that power is only partially correlated to the size of the design on the chip.

Because the clock frequency remains the same for the baseline and bit-balancing designs, but the bit-balancing design uses twice as many components as baseline AES, it can be estimated that the power consumed in an ASIC should double. The randomized clock decreases the clock frequency but does little to the circuit size and design. Based on the equation for dynamic power, the power consumed holds a linear

relationship with the frequency of the clock [32]. Therefore, the randomized clock design should consume less dynamic power based on the change in clock frequency.

To begin, the four circuit designs are simulated using the Xilinx Xpower Analyzer simulation tool. The simulator takes into account the components used by the design on the board, the clock frequency, and a ModelSim simulation of each design. Unfortunately, the Xpower Analyzer showed little to no variation between the designs. The results of the simulator are shown in Table 4.5.

Table 4.5: Simulated Power Consumption Results (Watts)

On-Chip	**Baseline**	**Random Clock**	**Bit-Balancing**	**Combined**
Clocks	0.201	0.223	0.255	0.246
Logic	0.002	0.002	0.002	0.002
Signals	0.004	0.005	0.004	0.005
BRAMs	0.087	0.087	0.087	0.087
PLLs	0.108	0.108	0.108	0.108
DCMs	0.068	0.068	0.068	0.068
PPC440s	0.649	0.649	0.649	0.649
IOs	1.901	1.900	1.900	1.900
Leakage	1.590	1.591	1.592	1.592
Total	**4.610**	**4.633**	**4.667**	**4.657**

From the simulated results, it is clear that static power dominates the design on the Virtex-5 FPGA. It can be concluded that the differences in designs have very little effect on the overall power drawn by the board. However, this research emphasizes minimizing the power consumed by the AES algorithms, not FPGA boards. Therefore, an alternate strategy to determining power consumption is required to confirm power consumption assumptions of the four designs. To do this, a wide-angle EM probe is used to measure the overall EM signal emanating from the chip. The reason the more focused Riscure high sensitivity probe is not used is because the location of the algorithm on the FPGA does not remain constant for every design. Future work using Xilinx's PlanAhead software may alleviate this problem, but for now the Willtek wide-angle EM probe works well without the need for clear specification of the design's location. The 'actual' power consumption values are determined using

EM estimation and are therefore more qualitative than quantitative in nature. Unlike the ML507 evaluation platform used in this research, many evaluation boards have built-in power measurement features to provide better accuracy when measuring the power consumption of a circuit.

Power consumption comparison is made by averaging 1000 traces of the algorithms and comparing the voltage magnitudes as the AES algorithms run. Figure 4.24 shows the average trace for the baseline AES system.

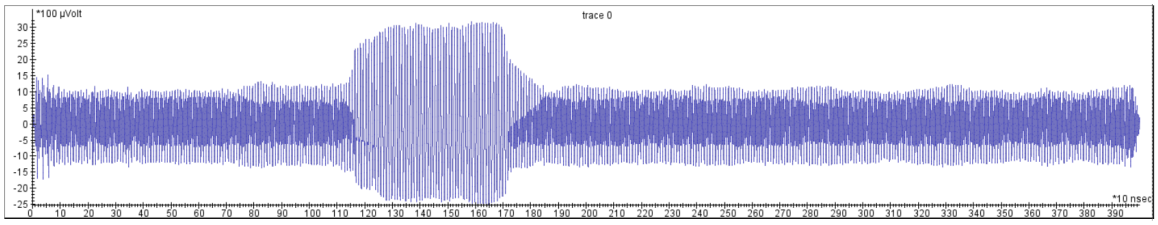

Figure 4.24: Average Trace of the Baseline AES System using the Willtek Probe

The voltage reading reaches a peak of 3.2 mV. Figure 4.25 shows the average trace for the bit-balancing AES system. The max voltage recorded is nearly quadruple the baseline AES system.

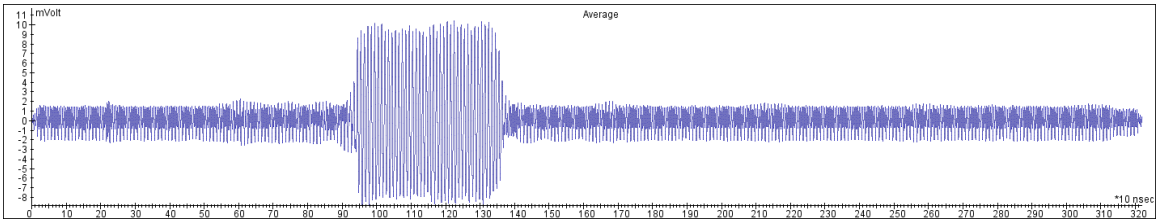

Figure 4.25: Average Trace of the Bit-Balancing AES System using the Willtek Probe

The voltage reaches 11.0 mV for the bit-balancing AES system. This voltage increase is expected given the circuit triples in size. For the randomized clock and combined systems, however, the power is more difficult to determine. For one, averaging the traces tends to flatten the signal because of the clock randomization. Secondly, the power signature should be 3-6x lower because of the decrease in clock frequency. Figure 4.26 shows the average trace for the randomized clock AES sys-

tem, while Figure 4.27 shows the average trace for the combined countermeasure AES system.

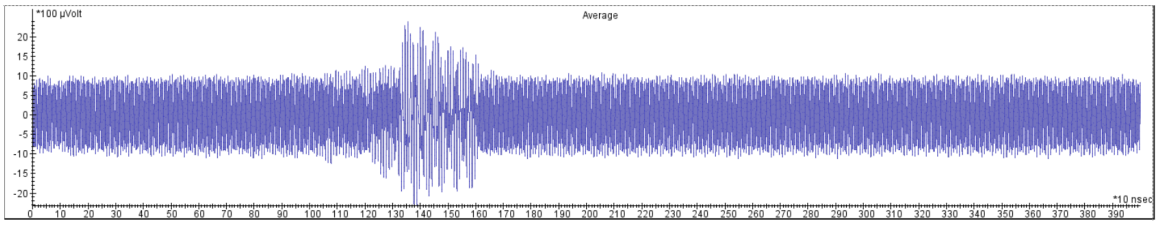

Figure 4.26: Average Trace of the Randomized Clock AES System using the Willtek Probe

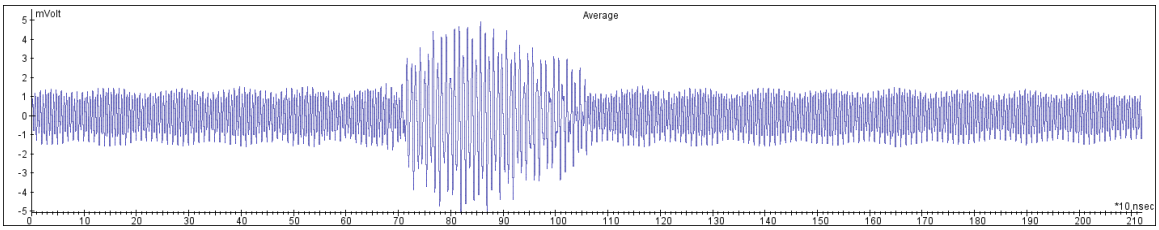

Figure 4.27: Average Trace of the Combined AES System using the Willtek Probe

Both the randomized clock and combined systems appear to have a voltage magnitudes less than their non-randomized counterparts. It is better to conclude that if the clock frequency were to remain constant, the power consumption of the randomized clock system should be similar to the baseline AES system. This assumption is made because the actual AES algorithm remains unchanged, and the added hardware to create a pseudo-random clock signal is minimal. The same can be concluded for the combined system when comparing it to the bit-balancing AES system. Therefore, the randomized clock reduces power to the effect that dynamic power is linearly correlated with the clock frequency. Table 4.6 summarizes the amplitudes of the EM measurements taken. While the EM measurements are measured in voltage, the EM measurements are directly correlated to power emissions and are therefore a good indicator of power consumed by the different designs.

Table 4.6: Hardware Power Consumption Results

System	**Voltage**	**Normalized to Baseline**	**Estimated Normalized to Baseline Excluding Clock Frequency Differences**
Baseline	3.2 mV	1.00x	1.00x
Random Clock	1.7 mV	0.53x	1.00x
Bit-Balancing	11.0 mV	3.43x	3.43x
Combined	5.0 mV	1.56x	3.43x

4.7 Circuit Delay Results

Finally, it is important to evaluate the speed of the cryptographic circuit due to the fact the AES algorithm needs to evaluate large amounts of input data in a timely manner. The amount of time required to complete an AES encryption is evaluated both in simulation and practice. The clock frequency is set to 100 MHz, and therefore the clock period is 10 ns. The algorithm completes after several stages listed below:

1. The AES key is sent to the system: This process can be completed in one clock cycle at its fastest.
2. The input data is sent to the system: This process can be completed in one clock cycle at its fastest.
3. Complete the rounds and output ciphertext: Each round takes one clock cycle, totaling 11 clock cycles for 128-bit AES.

In total, the entire process for the baseline system is capable of completing in 13 clock cycles. This process is reflected in the simulations using ModelSim.

4.7.1 Simulated Timing Results. Table 4.7 gives the simulated time required to complete the encryption processes.

Of course, if the clock randomization was set to 1-2x the original clock frequency, there would be substantial improvement on the time required to complete the algorithm using random clocking. Also, the added 10 clock cycles to the bit-balancing

Table 4.7: Simulated Circuit Speed Results

System	Clock Cycles	Time	Normalized to Baseline
Baseline	13	130 ns	1x
Random Clock	13	390 ns - 780 ns	3x - 6x
Bit-Balancing	23	230 ns	1.77x
Combined	23	690 ns - 1380 ns	5.31x - 10.62x

design are significant when realizing the algorithm does not need more than 13 clock cycles to complete in the first place. However, as seen in the hardware timing results, the simulated values do not take into account the fact that sending and receiving data to and from the system requires significantly more than one clock cycle.

4.7.2 Actual Timing Results. The actual timing results on the Virtex-5 FPGA board are heavily dependent on the Riscure Inspector Software and the C-code used to send and receive data. Because of this dependency, a hardware-based timing analysis is relatively inconclusive due to the fact that the timing appears to be driven by the C-code and serial communication speed.

The C-code used to send and receive data to the system is shown below.

Code IV.1:

```
AES_VERILOG_mWriteReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG0_OFFSET,0x00000000); //reset
enableTrigger(); // Turn on LED 0 to Trigger O-Scope
AES_VERILOG_mWriteReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG0_OFFSET,0x2C000000); //send key
AES_VERILOG_mWriteReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG0_OFFSET,0x2A000000); //send data
AES_VERILOG_mWriteReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG0_OFFSET,0x28000000); //enable ...
     system

// Read ciphertext from Hardware AES
cipher3 = AES_VERILOG_mReadReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG11_OFFSET);
cipher2 = AES_VERILOG_mReadReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG12_OFFSET);
cipher1 = AES_VERILOG_mReadReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG13_OFFSET);
cipher0 = AES_VERILOG_mReadReg(XPAR_AES_VERILOG_0_BASEADDR,AES_VERILOG_SLV_REG14_OFFSET);

disableTrigger(); // Turn off LED 0 to signify end of Triggering O-Scope
```

The trace collected is of the system between the enabletrigger() and disabletrigger() commands. Theoretically, this should cover the entire algorithm and nothing

more, but instead the C-code takes approximately 4 μs (4000ns) to complete. The trace of the entire process is below in Figure 4.28.

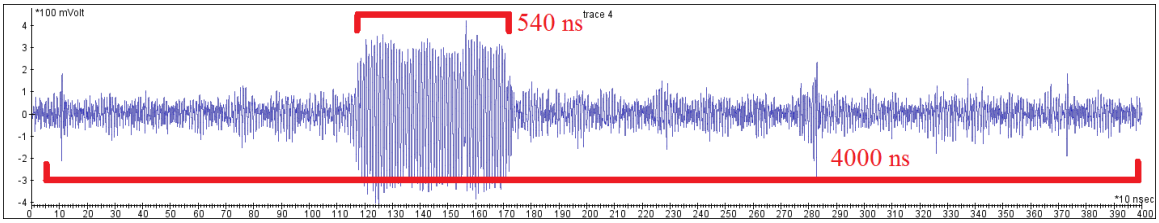

Figure 4.28: Timing Analysis of Baseline AES

It is clearly seen on the graph when the AES algorithm is enabled and when the algorithm is finished processing data. However, the 540 ns that the algorithm takes does not match with the 130 ns estimated during simulation. The hardware timing results are below in Table 4.8.

Table 4.8: Actual Circuit Speed Results

System	**Overall System Speed**	**Average Algorithm Runtime**	**Normalized to Baseline**
Baseline	4000 ns	540 ns	1x
Random Clock	4000 ns	365 ns	0.68x
Bit-Balancing	4000 ns	610 ns	1.13x
Combined	4000 ns	365 ns	0.68x

What is interesting to note is that the randomized clock implementation appears to complete faster than the baseline system. This outcome only further confirms that the timing of the circuit on the chip is more dependent on the transmission and reception of data than the actual processing. In this case, the simulation results provide a more accurate assessment of the delays of the different designs.

4.8 Results Summary

The initial AES algorithm design proves to be vulnerable to DPA and can be attacked after sampling half a million traces. An iterative version of AES is used

to maximize the signal to noise ratio, but the two countermeasures discussed in this research are designed to work with a pipelined AES version as well. After adding the countermeasures, the resistance to DPA increased by a factor of at least six for the bit-balancing and for randomized clock design. DPA was never successfully performed against either of the countermeasures due to the equipment constraints of this research.

While DPA resistance is the most important factor when designing countermeasures, there are several other factors that must be considered for the final design. The circuit size, power consumption, and speed are all important when considering real world application of the countermeasures. The objective of this research is to increase the DPA resistance of the circuit by several orders of magnitude compared to a baseline AES system. Once DPA resistance is accomplished, focus is then to minimize the costs to the user in increased circuit size, power consumption, and a decrease in speed. Table 4.9 provides an overview of the metric results evaluated for this research.

The circuit area for the bit-balancing and combined designs are around 3x the original circuit. While the goal was <= 2x, extra features were added to the system-level bit-balancing design to provide resistance to HW attacks as well as HD attacks. Power consumption, while challenging to measure, mirrors the circuit area used. Of course, it is important to note that the normalized power consumption has been modified to take into account the change in clock frequency. The normalized values are for systems whose average frequency is equal. Finally, the circuit speed is the one metric that did not meet expectations. The two reasons for this are the 10 extra clock cycles needed in the bit-balancing design to provide HD resistance, and also the clock period has been increased by 3x to 6x (instead of 1x to 2x). The clock frequency issue can be resolved depending on the system being used. In order to minimize the delay within the circuit from the randomized clock, the user only needs to triple the clock frequency entering the random clock generator. However, the Virtex-5 FPGA used

Table 4.9: Experimental Results for all Designs

Metric	**Baseline**	**Random Clock**	**Bit-Balancing**	**Combined**
DPA Resistance	500k Traces	>3,000k Traces	>3,000k Traces	>3,000k Traces
Norm. DPA Resistance	1x	>6x	>6x	>6x
Circuit Area (FF/LUTs/BRAMs)	850/1,131/5	870/928/5	3,337/3,969/10	3,357/3,376/10
Norm. Circuit Area	1x	0.95x	3.15x	2.98x
Dynamic Power Consumed	3.2 mV	1.7 mV	11.0 mV	5.0 mV
Norm. Power Consumed	1x	1x	3.43x	3.43x
Circuit Speed (Simulated)	130 ns	390 ns-780 ns	230 ns	690 ns-1380 ns
Norm. Circuit Speed	1x	3x - 6x	1.77x	5.31x - 10.62x

in this research has a bus clock limit of 125 MHz, and the system is already running at 100 MHz.

The countermeasures used in this research were tested thoroughly on the Virtex-5 FPGA with the intention of preventing DPA attacks on the AES algorithm. The countermeasures have shown to provide increased DPA resistance with only small increases in costs to the user. Using the same equipment that was used for this research, a single attack on one key byte using three million traces took on average four days to complete. The attack used over 100 GB of hard drive space, and required at least 16 GB of memory to perform the computations. While the countermeasures do not make the circuit immune to the side-channel attacks performed in this research, they do significantly increase the time and resources needed by an attacker.

V. Conclusions

The countermeasures developed and tested in this research are effective in obfuscating the key and intermediate data of the AES algorithm from standard DPA attacks. Those attacks include correlation with bit, HW, HD, and zero-value models used to determine the secret key. This chapter summarizes the research effort by covering the completed objectives, discussing the conclusions inferred from the results, presents contributions given to the field of study, and finally provides ideas for future work in the area.

5.1 Completed Objectives

This research was guided by a main objective, several secondary objectives, and also several derived objectives required to complete the main objective.

5.1.1 Main Objective: Side-Channel Obfuscation. The primary objective of this research is to provide protection for the AES algorithm against several of the most common side-channel attacks. Two hiding countermeasures were developed and implemented on the FPGA. Both countermeasures resulted in a significant increase in time and/or resources required to attack the circuit. Due to their relative independence, the combination of the countermeasures only adds to the difficulty of performing a successful DPA attack. Correlation between the power traces and the data was not found using the current resources of this thesis when the countermeasures were enabled.

5.1.2 Secondary Objective: Minimize User Cost. Because of the high number of side-channel hiding countermeasures available for the AES algorithm, consideration of user costs is important for the countermeasures developed in this research to remain viable anti-tamper options. The developed countermeasures do not create significant increases in circuit area, power consumed, or delay to the system.

1. Circuit Area: The randomized clock has a negligible effect on the area of the circuit. The system-level bit-balancing increases circuit area by just over 3x,

but is still comparable and competitive against similar dual rail precharge logic designs. It is also important to note that the AES algorithm itself is relatively small, and added circuit size does not create as many challenges as it may have in the past with limited logic blocks.

2. Circuit Power: If the average frequency is matched to an unprotected circuit, the added power drawn by the randomized clock is again negligible. Similar to circuit area, the system-level balancing draws over 3x the amount of power than the original circuit. This added power is expected given the increase in circuit size, and is still competitive against power increases by dual rail systems and common masking implementations.

3. Circuit Delay: The randomized clock only delays the circuit by 2x when the original frequency can be increased and optimized. The system-level bit-balancing countermeasure increases delay by 1.8x. Given that the algorithm completes in a matter of nanoseconds, the small increase in runtime delay has little effect compared to the execution time of the entire system (data input and output from a user/external system). Also, the delay increases compare favorably versus dual rail logic and masking techniques.

The two countermeasures meet their main objective while keeping the circuit small, efficient, and fast, making them viable choices as a means to protect AES from side-channel attacks.

5.1.3 Derived Objective: Perform DPA Attack. In order to develop and test countermeasures, it was important to first gain the ability to perform DPA on the circuit. Several common power attacks, including correlation using the bit model, HW model, HD model, and zero-value model were all successfully performed in this research. The most effective attack (HD attack) for the AES module chosen was used for comparisons of the two countermeasure's effectiveness.

5.2 Conclusions

Through this research, two hiding countermeasures were developed and implemented on an FPGA with the goal of protecting hardware AES from side-channel attacks. The two countermeasures are modified versions of an iterative 128-bit hardware AES implemented on a Virtex-5 FPGA.

The first countermeasure, randomizing the clock, was an implementation of an idea presented in [17]. However, the randomized clock module design differs in structure than originally presented. The randomized clock prevents an attacker from easily aligning traces, a step necessary to perform DPA. Therefore, the randomized clock module is effective in obfuscating the intermediate values of AES from an attacker performing side-channel analysis.

The second countermeasure, system-level bit-balancing, greatly improves upon the design given in [1]. The bit-balancing architecture is unique and simplified, while additional features are added to defend against HD attacks as well (system-level precharging). Also, the design architecture is implemented on only one chip, greatly increasing the difficulty of an attacker to isolate just one half of the design. The system-level bit-balancing countermeasure effectively protects the AES circuit by balancing the HW of the intermediate values, therefore minimizing the power consumption differences between varying values of data.

Both countermeasures show a minimum of 600% increase in side-channel protection against common DPA attacks. Also, because clock frequency variation and bit-balancing are two relatively independent hiding techniques, the combination of the two countermeasures only increases the protection of the circuit. The costs to the user for both countermeasures combined include just under a 3x increase in circuit area, 3.5x increase in power drawn, and an estimated average 2.7x increase in circuit delay. In the end, the two countermeasures were successful in obfuscating the secret key from a side-channel attack without significant cost to the user.

5.3 Contributions

The contributions provided by this research are summarized below:

- Developed a straightforward and effective system-level bit-balancing countermeasure for use on FPGAs.
- Added support for HD protection on the system level, effectively creating a system-level "precharging" scheme.
- Implemented and tested the system-level bit-balancing countermeasure on an FPGA.
- Implemented and tested the randomized clock countermeasure on an FPGA.
- Developed and performed a testbed to run and monitor hardware AES. The data collected from the setup was then used to complete several forms of side-channel analysis on the Virtex-5 system.

5.4 Future Work

While this research succeeds in meeting its specified goals as outlined in this paper, there are still areas of growth that this research could take. Most of the ideas listed below work to further solidify the success of the countermeasures.

5.4.1 Pipelined vs. Iterative AES. The AES version used in this research is an iterative 128-bit hardware version. The system-level bit-balancing design is somewhat specific to iterative AES, but can easily be modified for pipelined AES given a few modifications. This flexibility is important since most commercial hardware AES implementations use some form of pipelining. The bit-balancing countermeasure would remain the same in structure, but the added HD defense features would need to be modified. First, the 10 extra cycles can be removed since the intermediate data of the next input block is what each round sees next, not intermediate data from the same input block. In fact, pipelined AES contains significantly more natural

protection to HD attacks than iterative AES. The degree of protection and whether or not any additional randomization is needed could be determined in future work.

5.4.2 Expanding the Scope of Side-Channel Attacks. Only a few of the most common attacks were used in this research. The focus of this thesis was the development and implementation of countermeasures, not research into all the different types and strengths of the various side-channel attacks. Therefore, it made logical sense to pick the most common attacks and use those for countermeasure evaluation. However, a plethora of attacks exist to be used on hardware cryptographic circuits. Further research can be made to attempt more attacks on the countermeasures. Two ideas are listed below:

1. Perform an attack on the randomized clock countermeasure by pinpointing the clock cycle in question and performing correlation analysis on the integration of the EM signal.
2. Attempt template and/or fault attacks on the system-level bit-balancing countermeasure to determine the countermeasure's strength against stronger types of attacks.

5.4.3 Evaluate More Traces. The research conducted for this thesis was limited to the specifications of a standard computer workstation. The main limiting factor was system memory during data processing. By increasing the RAM to values greater than 16 GB, ensuring there is several hundred GBs of storage space free, and allowing more time for data collection, a user will be able to evaluate traces numbering greater than three million with ease.

5.4.4 Work with Multiple FPGA Platforms. The research was conducted on a Virtex-5 FPGA evaluation platform. However, there are several newer chips available, and their differences could have an impact on the effectiveness of the countermeasures. Many newer chips feature lower power usage and smaller gate widths.

While the countermeasure strengths are not expected to vary significantly depending on which FPGA the design is implemented on, it would be best to test this theory.

5.5 Summary

This research provides contributions for the protection of information processed using hardware AES. The results and analysis indicate successful achievement of the goals of this research, mainly to obfuscate sensitive data against side-channel attacks. Not only was a design developed and presented, but implementation and real-world testing were performed on an FPGA system.

Appendix A. Simulated AES Algorithms

The following figures include simulation results of the AES algorithm with and without several countermeasure features. The simulation is performed using the ModelSim software.

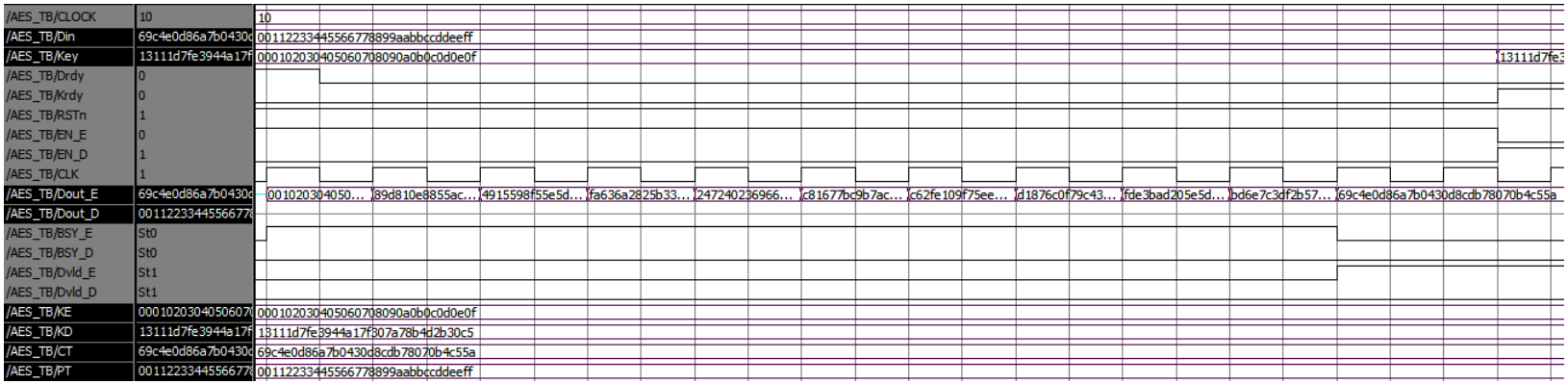

Figure A.1: Simulation of AES Algorithm Using ModelSim

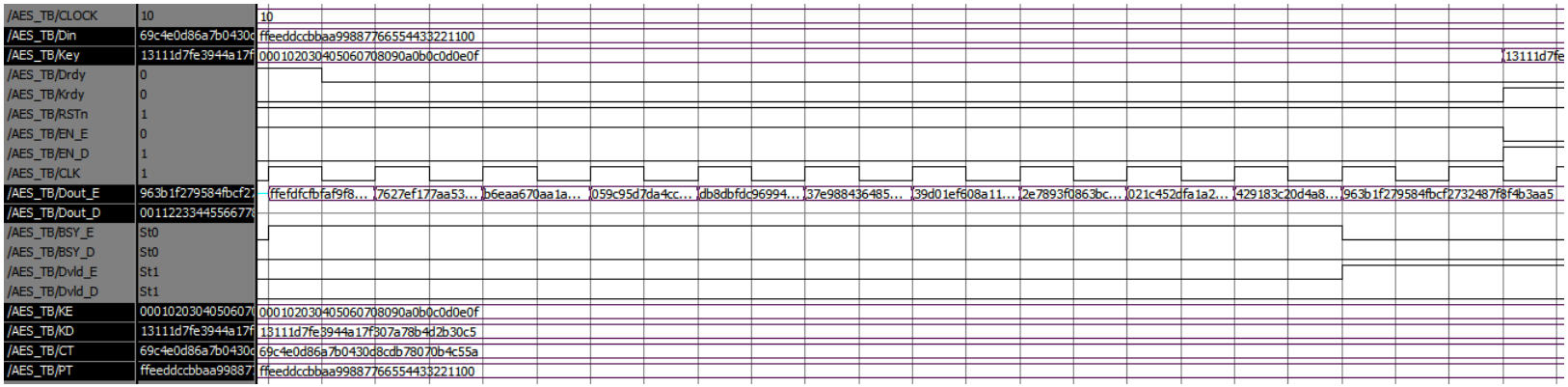

Figure A.2: Simulation of Inverted AES Algorithm Using ModelSim

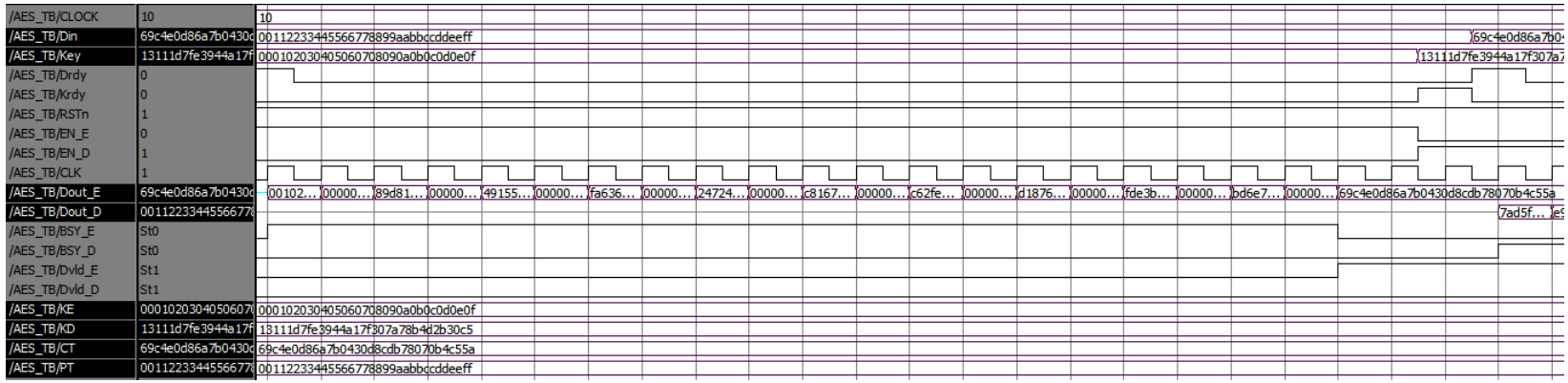

Figure A.3: Simulation of AES Algorithm with HD Resistance Using ModelSim

Bibliography

1. Ambrose, J. A., Parameswaran, S., and Ignjatovic, A., "MUTE-AES: A multiprocessor architecture to prevent power analysis based side channel attack of the AES algorithm," *Proc. IEEE/ACM Int. Conf. Computer-Aided Design ICCAD 2008*, 2008, pp. 678–684.

2. NIST, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)," November 2001, Federal Information Processing Standards Publication 197.

3. Kocher, P., Jaffe, J., and Jun, B., "Introduction to Differential Power Analysis and Related Attacks," 607 Market Street, 5th Floor, San Francisco CA, 94102, 1998, p. 5.

4. Kocher, P., Jaffe, J., and Jun, B., "Differential Power Analysis," Springer-Verlag, 1999, pp. 388–397.

5. Mangard, S., Oswald, E., and Popp, T., *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

6. Kamoun, N., Bossuet, L., and Ghazel, A., "Experimental implementation of DPA attacks on AES design with Flash-based FPGA technology," *Proc. 6th Int. Multi-Conf. Systems, Signals and Devices SSD '09*, 2009, pp. 1–4.

7. Soydan, S., "Analyzing the DPA Leakage of the Masked S-box via Digital Simulation and Reducing the Leakage by Inserting Delay Cells," *Proc. Fourth Int Emerging Security Information Systems and Technologies (SECURWARE) Conf*, 2010, pp. 221–227.

8. Alberts, D. S., *Information Age Transformation: Getting to a 21st Century Military*, Information Age Transformation Series, Command and Control Research Program, 2002.

9. "Federal Information Security Management Act of 2002." *Title III of the E-Government Act - Information Security, 44 U.S.C. Sec. 3541,*, December 2002.

10. Trappe, W. and Washington, L. C., *Introduction to Cryptography with Coding Theory*, Pearson Prentice Hall, Upper Saddle River, New Jersey 07458, 2nd ed., 2006.

11. Mount, M. and Quijano, E., "Iraqi insurgents hacked Predator drone feeds, U.S. official indicates," *CNN*, 2009.

12. Haulman, D. L., "USAF Manned Aircraft Combat Losses 1990-2002," Air Force Historical Research Agency, 2002.

13. Blomer, J., Merchan, J. G., and Krummel, V., "Provably Secure Masking of AES," *In SAC*, Springer, 2004, pp. 69–83.

14. Canright, D. and Batina, L., "A Very Compact "Perfectly Masked" S-Box for AES (corrected)," Tech. rep., Naval Postgraduate School, Monterey CA, 2008.

15. Liu, P.-C., Chang, H.-C., and Lee, C.-Y., "A low overhead DPA countermeasure circuit based on ring oscillators," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, Vol. 57, July 2010, pp. 546–550.

16. Rivain, M. and Prouff, E., "Provably Secure Higher-Order Masking of AES," *Cryptographic Hardware and Embedded Systems, CHES 2010*, edited by S. Mangard and F.-X. Standaert, Vol. 6225 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2010, pp. 413–427.

17. Zafar, Y., Park, J., and Har, D., "Random clocking induced DPA attack immunity in FPGAs," *Proc. IEEE Int Industrial Technology (ICIT) Conf*, 2010, pp. 1068–1070.

18. Oswald, E., Mangard, S., and Pramstaller, N., "A Side-Channel Analysis Resistant Description of the AES S-box," *Fast Software Encryption 2005, LNCS 3557*, Springer, 2005, pp. 413–423.

19. Alam, M., Ghosh, S., Mohan, M. J., Mukhopadhyay, D., Chowdhury, D. R., and Gupta, I. S., "Effect of glitches against masked AES S-box implementation and countermeasure," *IET Information Security*, Vol. 3, No. 1, 2009, pp. 34–44.

20. Riscure, "Inspector Series Product User Training," Online, 2011.

21. Aigner, M. and Oswald, E., "Power Analysis Tutorial," Tech. rep., Institute for Applied Information Processing and Communication, University of Technology Graz - Seminar.

22. Carlier, V., Chabanne, H., Dottax, E., Pelletier, H., and Sa, S., "Electromagnetic Side Channels of an FPGA Implementation of AES," *Cryptology ePrint Archive, Report 2004/145*, 2004.

23. Humphries, J. W. and Pooch, U. W., "All Random Numbers Are Not Created Equal: Differences Between Generators For Simulation And Cryptograpy," *Proc. Advanced Simulation Technology Conference: Military, Government, and Aerospace Simulation, Society for Computer Simulation*, Washington DC, April 16-20 2000.

24. Kafi, M., Guilley, S., Marcello, S., and Naccache, D., "Deconvolving Protected Signals," *Availability, Reliability and Security, 2009. ARES '09. International Conference on*, march 2009, pp. 687 –694.

25. Wu, J., Kim, Y.-B., and Choi, M., "Low-power side-channel attack-resistant asynchronous S-box design for AES cryptosystems," *Proceedings of the 20th symposium on Great lakes symposium on VLSI*, GLSVLSI '10, ACM, New York, NY, USA, 2010, pp. 459–464.

26. Knuth, D. E., *The art of computer programming, volume 1 (3rd ed.): fundamental algorithms*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1997.

27. Satoh, A., "AES Encryption/Decryption Macro," Graduate School of Information Sciences, Tohoku University, http://www.aoki.ecei.tohoku.ac.jp/crypto/, 2007.

28. Rodríguez-Henríquez, F., Saqib, N. A., Díaz-Pèrez, A., and Koc, C. K., *Cryptographic Algorithms on Reconfigurable Hardware (Signals and Communication Technology)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

29. Popp, T. and Mangard, S., "Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints," *Cryptographic Hardware and Embedded Systems CHES 2005*, edited by J. Rao and B. Sunar, Vol. 3659 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2005, pp. 172–186.

30. Chen, Z. and Zhou, Y., "Dual-Rail Random Switching Logic: A Countermeasure to Reduce Side Channel Leakage," *Cryptographic Hardware and Embedded Systems - CHES 2006*, edited by L. Goubin and M. Matsui, Vol. 4249 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2006, pp. 242–254.

31. Mangard, S., "A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion," *Information Security and Cryptology ICISC 2002*, edited by P. Lee and C. Lim, Vol. 2587 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, 2003, pp. 343–358.

32. Czapski, P. P. and Sluzek, A., "Power Optimization Techniques in FPGA Devices: A Combination of System- And Low- Levels," World Academy of Science, Engineering and Technology, 2007.

33. Becker, J., Huebner, M., and Ullmann, M., "Power estimation and power measurement of Xilinx Virtex FPGAs: trade-offs and limitations," *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, sept. 2003, pp. 283 – 288.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
22–03–2012	Master's Thesis	Sept 2012 — Mar 2012

4. TITLE AND SUBTITLE

Obfuscating Against Side-Channel Power Analysis Using Hiding Techniques for AES

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Fritzke, Austin W., 2d Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GE/ENG/12-15

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Dr. Robert L. Herklotz
Program Manager - Information Operations and Security Air Force Office of Scientific Research (AFOSR/RSL)
875 N. Randolph Street, Suite 325, Room 3113
Arlington, VA 22203-1768
(703) 696-6565 robert.herklotz@afosr.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR/RSL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approval for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The transfer of information has always been an integral part of military and civilian operations, and remains so today. Because not all information we share is public, it is important to secure our data from unwanted parties. Message encryption serves to prevent all but the sender and recipient from viewing any encrypted information as long as the key stays hidden. The Advanced Encryption Standard (AES) is the current industry and military standard for symmetric-key encryption. While AES remains computationally infeasible to break the encrypted message stream, it is susceptible to side-channel attacks if an adversary has access to the appropriate hardware. The most common and effective side-channel attack on AES is Differential Power Analysis (DPA). Thus, countermeasures to DPA are crucial to data security. This research attempts to evaluate and combine two hiding DPA countermeasures in an attempt to further hinder side-channel analysis of AES encryption. Analysis of DPA attack success before and after the countermeasures is used to determine effectiveness of the protection techniques. The results are measured by evaluating the number of traces required to attack the circuit and by measuring the signal-to-noise ratios.

15. SUBJECT TERMS

Advanced Encryption Standard, Countermeasures, Differential Power Analysis, Encryption, FPGA, Side-Channel Analysis

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj Todd Andel, USAF (ENG)
U	U	U	UU	100	19b. TELEPHONE NUMBER *(include area code)* (937) 255–3636, ext 4901; todd.andel afit.edu