UNIVERSITY OF SOUTHAMPTON

Hardware Level Countermeasures Against Differential Power Analysis

by

Karthik Baddam

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Physical and Applied Sciences

School of Electronics and Computer Science

February 2012

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL AND APPLIED SCIENCES
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

Hardware Level Countermeasures Against Differential Power Analysis

by Karthik Baddam

Hardware implementations of mathematically secure algorithms unintentionally leak side channel information, that can be used to attack the device. Such attacks, known as side channel attacks, are becoming an increasingly important aspect of designing security systems. In this thesis, power analysis attacks are discussed along with existing countermeasures. In the first part of the thesis, the theory and practice of side-channel attacks is introduced. In particular, it is shown that plain implementations of block ciphers are highly susceptible to power-analysis attacks.

Dual rail precharge (DRP) circuits have already been proposed as an effective countermeasure against power analysis attacks. DRP circuits suffer from an implementation problem; balancing the routing capacitance of differential signals. In this thesis we propose a new countermeasure, path switching, to address the routing problem in DRP circuits which has very low overheads compared to existing methods. The proposed countermeasure is tested with simulations and experimentally on an FPGA board. Results from these tests show a minimum of 75 times increase in the power traces required for a first order DPA attack.

Some of the existing countermeasures to address the routing problem in DRP circuits do not consider coupling capacitance between differential signals. In this thesis we propose a new method, divided backend duplication that effectively addresses balanced the routing problem of DRP circuits. The proposed countermeasure is tested with simulations and results show a minimum of 300 times increase in the power traces required for a first order DPA attack.

Randomisation as a DPA countermeasure is also explored. It is found that randomising the power consumption of the cryptographic device itself has little impact on DPA. Randomising the occurrence of intermediate results, on which DPA relies on, has better effect at mitigating DPA.

Contents

List of Figures

DECLARATION OF AUTHORSHIP

Page left blank for declaration of authorship.

Acknowledgements

I am thankful to my supervisor Prof. Mark Zwolinski for his invaluable advice and guidance, this work would not have been possible without his guidance and support. I am specially thankful for his patience and for not giving up on me during the writing period of this thesis. I am thankful to my fellow students; Arash, Biswajit, Noohul, Marco, Sankalp, Urban, Sawal, Amit to name a few, for the small discussion we have in ESD lab. My time at ESD would be less entertaining without them. I would like to specially thank John R. Goodwin for his discussions regarding differential power analysis. To my parents; no words to say how grateful I am for having such caring and support parents. Finally to my beloved wife, Shalini, without whom writing this thesis would have been impossible, thanks for the support and the motivation you gave me.

To my wife Shalini and daughter Eshika

Chapter 1

Introduction

1.1 Motivation

Digital communications have become a major part of modern day life. As more and more information is being transmitted electronically, ways to keep the information safe from eavesdropping are becoming more complex. The value of information is always increasing, while it is subjected to an increasing number of threats. One of the common electronic transactions in everyday life is electronic banking, which ranges from internet banking to credit/debit cards. The modern credit/debit cards employ electronic circuits to authenticate the card holder. Such cards are also referred to as smart cards. Security is becoming an important metric along with cost, performance and power consumption in embedded systems such as smart cards [77]. To prevent eavesdropping, all the systems employ some form of cryptographic algorithm.

Cryptography is the science of protecting information. A cryptographic algorithm is a function that uses a *secret key* to encrypt information and without the knowledge of the *secret key* decrypting this information would be impossible. An attack on a cryptographic algorithm is the act of decrypting the encrypted information without the knowledge of the *secret key* [1]. Sometimes this is also referred as breaking an algorithm. During the past years, there has been lot of research on cryptography and as a result there are several algorithms that provide data security and authenticity, such as RSA [79], ECC [28, 52], AES [57], DES [58], TDES [60] and DSA [59]. These cryptographic algorithms are well studied and breaking them is considered to be computationally infeasible, provided they are used with the suggested key size. Although the above cryptographic algorithms are computationally infeasible to break, providing security in practice is still a challenge. All cryptographic algorithms rely on the fact that the *secret key* is kept secret. If the *secret key* is available to a third party then all the security provided by a application using this *secret key* is lost.

Smart cards are credit card sized devices used for a variety of security applications like ID cards, credit/debit cards, Cellular telephony, pay TV. Smart cards generally contain a microprocessor, volatile memory, non-volatile memory and optionally hardware accelerators for cryptographic functions. Smart cards also contain the *secret key* used in the secure application. For example, a smart card can be used to authenticate its holder. Consequently the ability to keep what is being processed on the smart card and its memory are becoming important. For example, if an attacker were to determine the *secret key* of a credit/debit card, the attacker could then essentially print money. Thus the cryptographic scheme used in such devices comes under more and more scrutiny.

Most attacks on cryptographic systems in the past have concentrated on the mathematics of the cryptographic algorithm. It has been assumed that if the cryptographic algorithm is secure then its implementation will also be secure, until

[1] All cryptographic algorithms have a known attack called brute force attack. In this an attacker tries every possible combination of the *secret key*. However, in practice, this is not computationally feasible because the time required to try all possible keys, for example, of a 128 bit key ($3.4 * 10^{38} combinations$) on a current generation computer takes few hundred of years

Kocher published timing attacks [30], and power analysis attacks [31]. Unfortunately the device implementing the mathematically secure cryptographic algorithm may leak certain information, called side channel information (it's called side channel as the device and not the cryptographic algorithm is leaking information), which can be used to find the secret key of that device. Time taken and power consumed are a common side channel information. Timing attacks depend on the fact that a different amount of time is taken for different operations. Power analysis and electromagnetic analysis depends on the fact that the hardware running the cryptographic algorithm consumes different amounts of power or emits different electromagnetic signals depending on the data being processed or operations performed. Attacks on cryptographic systems that use side channel information are called side channel attacks.

Differential power analysis (DPA) attack is a type of power side channel attack that uses statistical analysis. DPA uses power side channel leakage from several encryptions (called number of traces) of the cryptographic device on one hand and the attackers hypothetical power consumption of the cryptographic device on the other; and uses statistical correlation to report the most likely value of the *secret key*. It is shown that one of the current state-of-the-art encryption algorithm, AES, has been successfully attacked using DPA [62] and that DPA can be used to attack implementations of otherwise mathematically secure encryption algorithms [31]. Attacks that build on DPA have also been published, these are Second Order DPA and Template attacks. Second order DPA [50] attacks use the attacker's hypotheses about the side channel leakage of a device at two moments in time. Template attacks [15] consist of a two phase strategy: first a template of side channel leakage is build based on an identical but different cryptographic device as the one in the attack. Second, the side channel leakage from the attacked device is matched against the pre-characterised templates. These attacks are normally referred to as higher order differential power analysis attacks (HODPA).

To defend against DPA attacks, several countermeasures have been proposed, and have various costs associated with them; either increase in area and/or difficulty in implementing them. These countermeasures include dual rail precharge logic styles [91, 94], algorithmic masking [1, 66], gate level masking [69, 103], and randomisation [8, 112].

1.2 Research Objectives

The main objective of this research is to investigate and find hardware level countermeasures to prevent DPA attacks on cryptographic systems. Dual rail precharge countermeasures have been shown to remove data dependent power consumption, however implementing them in a secure way is shown to be difficult. Our first objective is to investigate and extend dual rail precharge countermeasures that can be easily implemented. Existing randomisation countermeasures at hardware level are shown to have low area costs associated with them. Our second objective is to investigate randomisation countermeasures at hardware level and compare them against other existing countermeasures.

Summary of our objectives are:

- To investigate and extend dual rail precharge countermeasures that can be easily implemented.
- To investigate randomisation countermeasures at hardware level.

1.3 Scope and Assumptions

We needed a way to measure the effectiveness of countermeasures against DPA on cryptographic designs. To achieve this, we used these as our test designs:

DES Sbox, AES Sbox and AES design. We used first-order DPA attack in our evaluations to measure DPA resistance and did not consider HODPA and template attacks. We assume that any resistance gained by a countermeasure against DPA, when compared to a DPA on an unprotected implementation, will also be gained against HODPA and template attacks, when compared to HODPA and template attack on an unimplemented design. In the remainder of the thesis, we refer to first order DPA as DPA.

1.4 Research Contributions

Our research objectives have led to the following contributions:

- We have developed a simulation based power analysis setup that aids in power analysis attacks. This setup can be used to evaluate the countermeasures that prevent power analysis attacks. We have also developed an FPGA based power analysis setup for the same purpose. Both these setups are configurable, such that any design can be evaluated for power analysis attacks.
- Dual rail precharge logic styles have been already proposed as a countermeasure for power analysis attacks. We have shown that dual rail precharge circuits are vulnerable to DPA if the routing capacitance of differential signals is not properly balanced. To solve the dual rail differential routing problem we have proposed a new countermeasure, path switching, to improve DPA resistance. We have published results from this work in [3, 4].
- Exiting solutions to solve the dual rail differential routing problem do not consider coupling capacitance. To address this we have proposed a new method, called divided backend duplication. We have published results from this work in [2].

- A C++ program has been developed to aid in the transformation from a normal circuit to a dual rail precharge circuit. Further enhancements have been made to this program to incorporate path switching and divided backend duplication solutions. Scripts have also been written that automate the divided backend duplication process. These programs can be easily adopted to implement any design in dual rail precharge logic style.
- Randomisation countermeasures are also explored as a solution to prevent power analysis attacks. It is shown that randomising the power consumption itself does not prevent DPA.

1.5 Thesis Organisation

The organisation of the thesis is as follows:

- Chapter 2 introduces the basic concepts of cryptography and then discusses side channel attacks, particularly differential power analysis attacks.
- Chapter 3 classifies exiting countermeasures against DPA and gives a brief overview of some of the existing solutions to prevent DPA.
- Chapter 4 introduces DPA flow developed during the course of this research. DPA results from simulation based DPA flow and measurements on a FPGA are presented.
- In Chapter 5 DRP logic is introduced as a solution to prevent DPA. Later it is shown that routing of dual rail nets poses a problem to the security of dual rail precharge logic designs. Path switching is proposed as a solution to overcome the balanced routing problem in dual rail designs. Based on simulation and real FPGA experiments, we show that path switching increases DPA resistance of dual rail logic styles.

- In Chapter 6 discusses the shortcomings of existing methods for routing balanced dual rail nets. Another countermeasure, called divided backend duplication is proposed as a solution to overcome balanced routing in dual rail logic styles.

- Chapter 7 is about randomisation countermeasures to prevent DPA. Various randomisation methods are discussed to counteract DPA.

- Finally Chapter 8 contains our conclusions about side channel attacks and its countermeasures. Future research points are also discussed.

- Appendix A shows the research papers written as part of the work that led to the production of this thesis.

- Appendix B contains tabular listing of DPA results based on our FPGA setup.

Chapter 2

Cryptography and Side Channel Attacks

2.1 Introduction

Cryptography is the science of protecting information. Cryptography is used in most secure applications such as including smart cards. In the first part of this chapter, basic techniques and theory of cryptography are reviewed. Cryptography is a vast subject and as a result, only the most basic terms are reviewed. For further information about cryptography, we refer the reader to a standard text such as [82]. In the later part of this chapter, side channel attacks are reviewed.

2.2 Basic Review of Cryptography

A cryptographic algorithm is a function that uses a *secret key* to encrypt information. The process, where the message is encrypted, is called *encryption*. The input to *encryption* process is generally referred to as *plaintext* and the output from the *encryption* process is referred to as *ciphertext*. It is important to note

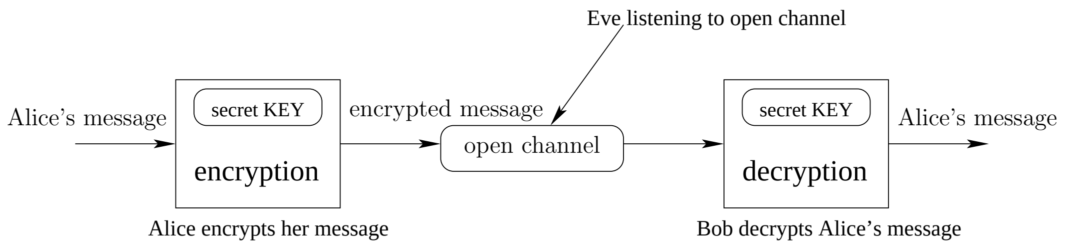


FIGURE 2.1: Process showing usage of Cryptography to securely communicate between Alice and Bob

that it is difficult, if not impossible, to obtain the *plaintext* from *ciphertext* without knowledge of the *secret key. Decryption* is the inverse of *encryption* where, the *ciphertext* and *secret key* is used to obtain the *plaintext.*

2.2.1 Why Use Cryptography?

Consider an example, where Alice has to send a secret message to Bob and does not want anyone else to see this message. Eve is interested in knowing what the message is, so she taps into the communication line that Alice and Bob use. If Alice sends her message as is (i.e, without encrypting it), then Eve or anyone else who can see the message can understand what that message is about. To prevent such eavesdropping, Alice and Bob decide to use cryptography to communicate. Alice and Bob meet beforehand and agree on a cryptographic algorithm and a *secret key* to use. Later on when Alice needs to communicate with Bob, she encrypts her message with the *secret key* and then sends it to Bob. Even though Eve can see the encrypted message, it is useless without the knowledge of the *secret key.* Bob on the other hand, can see Alice's message by decrypting the message he received, as he already has the same *secret key* that Alice used for encrypting. This process is depicted in Figure 2.1. Thus, by using cryptography Alice and Bob can securely communicate without having to worry about eavesdropping.

Although the example discussed above is simple, it shows the need for cryptography and how it can be used to securely communicate between two parties.

2.2.2 Symmetric Cryptography

In Symmetric Cryptography, the same *secret key* is used for *encryption* and *decryption*, as shown in Figure 2.9. Thus *secret key* **K** will be used in both encryption, $\mathbf{C = E(P,\ K)}$ and decryption, $\mathbf{P = D(C,\ K)}$. Decryption is the inverse operation of encryption, i.e, $\mathbf{P = D(E(P,\ K),\ K)}$.

There are two types of symmetric cryptographic algorithms, block ciphers and stream ciphers. A block cipher operates on n-bit plaintext and produces n-bit ciphertext as output, n being the block size. Current block ciphers usually operate on 128 bit blocks.

A stream cipher on the other hand, operates on smaller units of plaintext, usually bits. Stream ciphers encrypts data as it comes, i.e, it does not need fixed length data to operate on. Stream ciphers employ random number generators to encrypt their data and they work on a continuous stream of data. With a block cipher, encryption of any particular plaintext will result in the same ciphertext when the same key is used. On the other hand, ciphertext from a stream cipher will vary depending on when they are encountered during the encryption process. Block ciphers can be adapted to act as stream ciphers, although bespoke stream ciphers are generally faster and less complex than block ciphers.

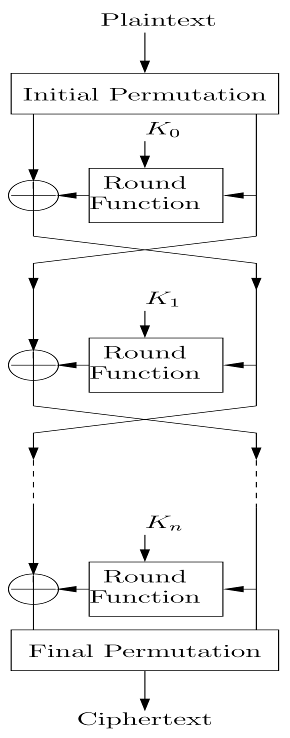
2.2.2.1 Block Ciphers

Block ciphers are so called, because they operate on n-bit blocks. A block cipher operates on n-bit plaintext and produces n-bit ciphertext as output and is dependent on the *secret key.* Block ciphers often use diffusion and confusion to encrypt data. Diffusion means the redundancy in the plaintext and secret key are dissapated in the ciphertext. Even a change in single input bit will be diffused over several ciphertext bits and hence it will be difficult for the attacker to gain knowledge about the plaintext from the ciphertext. Confusion refers to making the relationship between inputs and the ciphertext as complex as possible. Even if an attacker can figure out some ciphertext patters, he cannot use the cipher method and patterns to figure out the secret key.

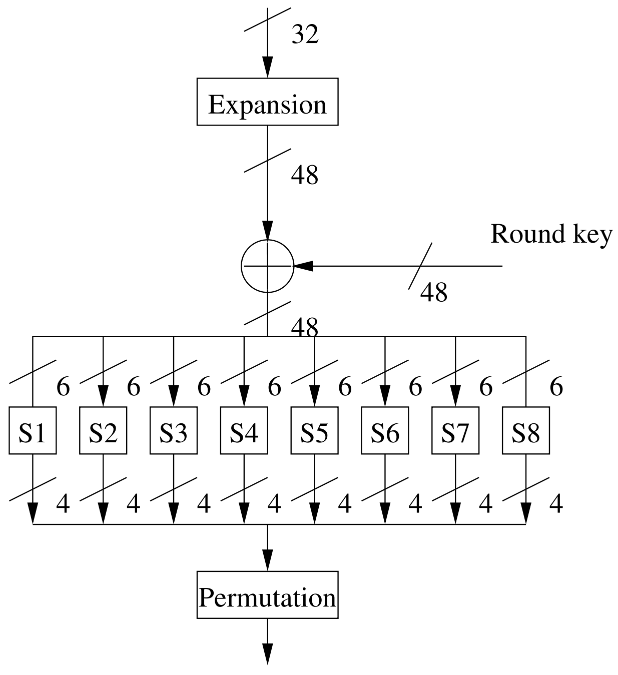
A cipher that combines two or more simple operations in a manner intending that the resulting cipher is more secure than the individual components, is called a product cipher. These simple operations usually increase either confusion or diffusion. A cipher involving the sequential repetition of an internal function is called an iterated block cipher. This internal function is also referred to as a round function. Although block ciphers represent a very complicated transformation most are composed of repeating iterations of simpler functions, also called iterated product ciphers. Two popular schemes for designing block ciphers are Substitution-Permutation (SP) networks and Feistel networks.

A SP network is a product cipher composed of a number of stages each involving substitutions and permutations. During substitution the data is separated into smaller blocks and the values in these blocks are substituted for others, typically using a lookup table called s-box, this increases the confusion. Permutation works across several blocks and mixes the data, swapping bits or combining values so the influence of data from one part of the plaintext is diffused through the whole ciphertext.

Feistel networks are a subset of SP networks, so are also made up of a number of stages each involving substitutions and permutations. The difference is that, the plaintext is split into two equal halves and the round function is applied to the right hand half. The result is then XOR-ed with the left hand side and becomes the new right hand side. The original right hand side then becomes the new left hand side. An advantage of Feistel networks is that encryption and decryption is very similar.



(a) Block diagram of DES



(b) Round function of DES

Figure 2.2: DES block diagram and round function

Some of the common symmetric block cryptographic algorithms are Data Encryption Standard (DES) [58], TDES [60] and Advanced Encryption Standard [57]. An example of an algorithm based on SP network is AES. An example of an algorithm based on a Feistel network is DES.

2.2.2.2 DES

The Data Encryption Standard (DES) was standardised in 1976 by National Bureau of Standards and was developed by IBM. DES is a block cipher that works on 64-bit blocks of data and uses 56-bit key and is based on Feistel networks [58].

Block diagram of DES is shown in Figure 2.2(a). As it can be seen, it has an Initial Permutation stage, series of round functions and finally a Final Permulation stage. The round function operates on the right half of the input block and round key. There are a total of 16 rounds in DES.

The round function is shown in Figure 2.2(b). In the first stage of round function, 32-bit input is expanded into 48-bit. The expanded data is then xor-ed with the round key and split into eight 6-bit blocks. Each 6-bit block is then put through a different s-box with 4-bit output and the resulting eight 4-bit blocks are re-arranged by a fixed permutation.

The 56-bit key is expanded into sixteen 48-bit round key. This is achieved by splitting the initial 56-bit into two halves. Each 28-bit half is then rotated left by either one or two bits depending on the round. From this 24 bits are then selected from each half by a fixed permutation. The process is repeated for each round.

Although the design details of DES are not published, the details of all the permutations, expansion and s-boxes are published and are part of the standard.

Security of DES

DES only uses a 56-bit key, this gives $7.2 * 10^{16}$ possible key combinations. In the 1970s this was adequate for brute force attacks to be infeasible. However with the current generation computers, this is no longer true. To highlight this fact RSA Security created a series of contests called the DES Challenges. The first challenge was in 1997 and was solved by the DESCHALL project in 96 days, which was a distributed computing project designed to break DES. Less than a year later in Feb 1998, a team from Distributed.net cracked DES in 41 days. In 2006 the universities of Bochum and Kiel developed *COPACOBANA*, which can retrive the correct DES key in an average of of 7.2 days and all keys can be tested in 14.4 days. *COPACOBANA* is built using off the self components and had 120 FPGAs. Cost to build *COPACOBANA* is less than $\$10,000$ [37].

DES can no longer be considered secure against brute force attacks. In order to increase the security against brute force attacks without having to change to a different algorithm a variant of DES, called Triple DES, was developed and is discussed in Section 2.2.2.3. In 1997 NIST announced the development of a new standard, to replace DES. It was published in 2002 and is called the Advanced Encryption Standard (AES) and is discussed in Section 2.2.2.4.

2.2.2.3 Double and Triple DES

To make DES secure without significantly changing the underlying algorithm, two variants of DES have been proposed, namely Double DES and Trible DES. Double DES refers to the use of two DES blocks with two seperate keys, effectively doubling the DES key length from 56-bit to 112-bit. In Triple DES (TDES), three DES blocks with three different keys are used. A common structure of TDES is linking the DES encryption blocks in series. This is commonly called as EEE as all the steps are encryptions. To make TDES backward compatible with DES an EDE structure is used, i.e, the second block is DES decryption block. If the three keys to TDES EDE are same, then its operation is same as a DES encryption.

One would expect that by using Double DES the key space would increase to 2^{56*2}. Diffie and Hellman [18] have shown that this is not true by developing *meet in the middle attack* [18]. It is a known plaintext attack where the attacker calculates one encryption of the plaintext for all possible n keys and stores the results. Then the attacker calculates one decryption of the ciphertext for each key in turn, if the result is also in the previous list of results then it is likely that the correct keys have been found. This can then be verified with another plaintext and ciphertext pair. For this reason double DES would not increase the security from 2^{2n}, but only to 2^{n+1}. Although a three key TDES has a key size of 168-bit, but due to the *meet in the middle attack* the effective security it provides is only 112-bit.

2.2.2.4 AES

In January 1997 the National Institute of Standards and Technology (NIST) body announced the initiation of the Advanced Encryption Standard (AES) development effort, to replace the ageing DES. Its aim is to create a new standard for a block cipher that would provide secure encryption.

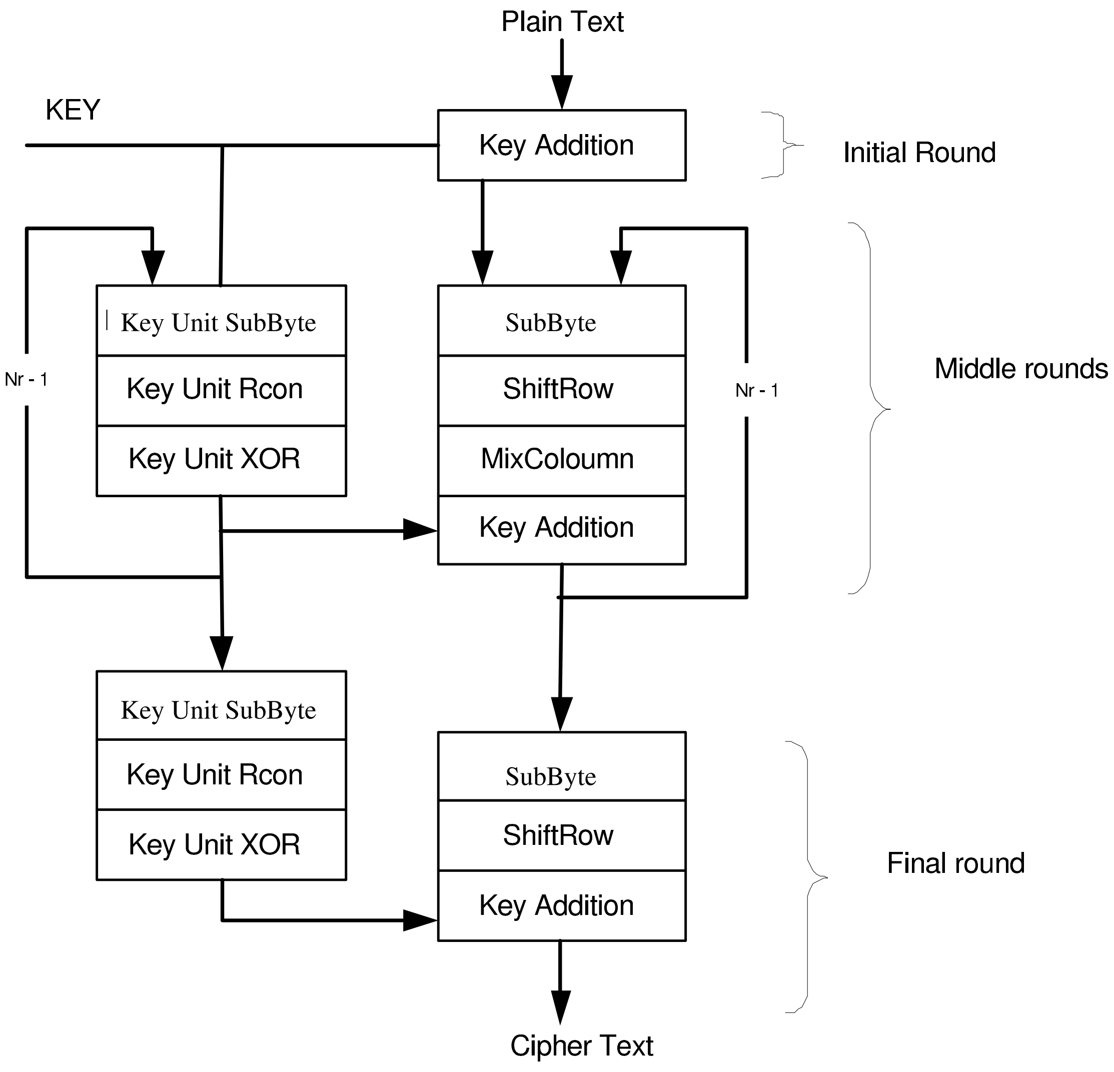


FIGURE 2.3: AES round operation

In October 2001 the algorithm Rijndael, developed by Vincent Rijmen and Joan Daemen, was selected to be AES [57]. AES is a symmetric block cipher that is based on SP networks. The structure of AES is shown in Figure 2.3. AES operates on a state that is initialized with a plaintext block, and after encryption this contains the ciphertext. The *state* can be pictured as a rectangular array of byte of four rows and four columns.

AES consists of a number of rounds, each round makes a number of transformations on a state, and uses a round key derived from the encryption key. AES supports three different key lengths and the number of rounds is dependent on the key length. The number of rounds is 10 for 128, 12 for 196 and 14 for 256 bit key.

An encryption of an input block starts with AddRoundKey transformation. This is followed by an odd number of regular rounds, and ends with a special final round. Each AES round function, except the final round, consists of four transformations: the *SubByte*, the *ShiftRows*, the *MixColumns*, and the *AddRoundKey*, while the final round does not have the *MixColumns* transformation. All AES transformations are invertible which makes decryption possible. In this section encryption transformations are breifly discussed.

SubByte Transformation

SubByte is generally called as S-box or substitution box, is a non linear byte substitution, operating on each of the state bytes independent of the round and the position of the byte.

S-box is constructed by composing two transofrmations: first by taking the multiplicative inverse in the finite field GF(2^8), then by applying the affine transformation over GF(2), given by Equation 2.1.

$$b_i^{'} = b_i \oplus b_{(i+4)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+6)\ mod\ 8} \oplus b_{(i+7)\ mod\ 8} \oplus c_i \tag{2.1}$$

S-box operates on individual bytes of the *state*, as shown in Figure 2.4. $a_{0,0}$ represents the 8 MSB's of the plaintext and $a_{3,3}$ represents the 8 LSB's of the plaintext.

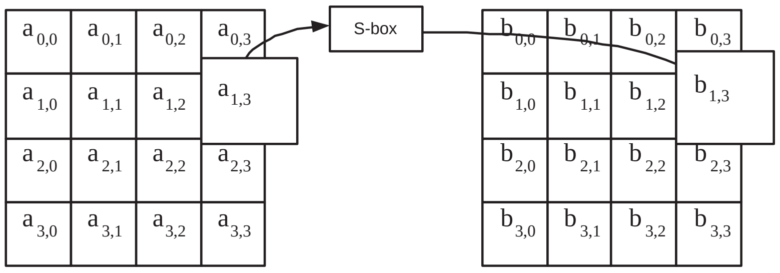

S-box
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,0}$ $a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,0}$ $a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,0}$ $b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,0}$ $b_{3,1}$ $b_{3,2}$ $b_{3,3}$

FIGURE 2.4: S-box operation on states

S-box can be implemented as look-up tables, which take up lot of memory, or in combinational logic using composite field arithmetic. The advantage of implementing S-box in combinational logic is that they take smaller area. In [115] composite field arithmetic has been briefly introduced and details are given for implementing S-box. In [57], s-box lookup table values are given.

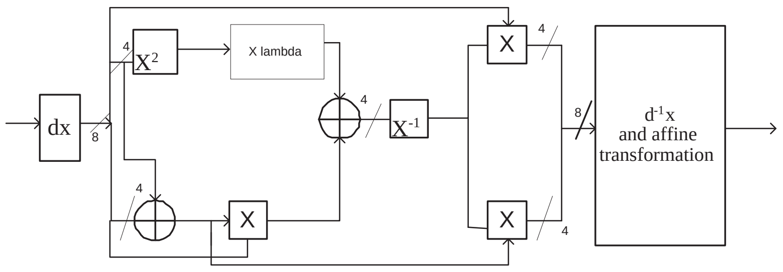


FIGURE 2.5: Composite filed implementation of S-box

ShiftRows Transformation

ShiftRow is a simple shifting transformation. The rows of the state are cyclically shifted over different offsets. Row 0 is not shifted, Row 1 is shifted over 1 byte, Row 2 is shifted over 2 bytes and Row 3 is shifted over 3 bytes.

Figure Figure 2.6 illustrates the ShiftRow transformation. In actual hardware ShiftRows does not take any area, only the output of S-box should be rotated accordingly before passing it to MixColumn.

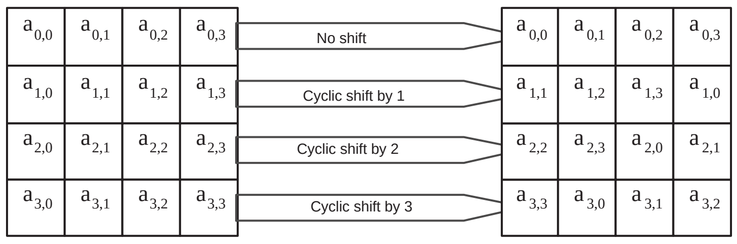


FIGURE 2.6: Shift Rows transformation.

The MixColumn Transformation

The MixColumn transformation considers the four bytes in each column of the state, as the coefficient of a polynomial over GF(2^8) and multiplied by a(x) modulo x^4+1, where

$$a(x) = (03)_{16}\,x^3 + (01)_{16}\,x^2 + (01)_{16}\,x + (02)_{16} \tag{2.2}$$

In matrix form MixColumn can be represented as

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

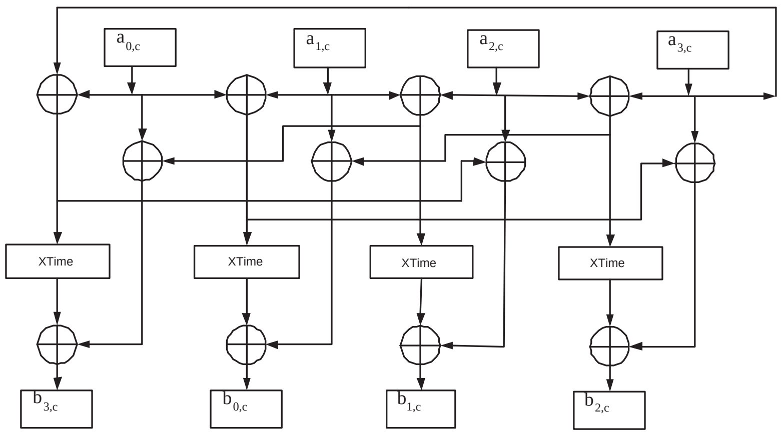


FIGURE 2.7: MixColumn transformation, adapted from [115].

Efficient implementation of MixColumn presented in [115], shown in Figure 2.7, is summarised here. XTime does the constant multiplication by $(02)_{16}$. This can be written as

$$
\begin{aligned}
(02)_{16}S &= xS \\
&= s_7x^8 + s_6x^7 + s_5x^6 + s_4x^5 + s_3x^4 + s_2x^3 + s_1x^2 + s_0x \text{mod p(x)} \\
&= s_6x^7 + s_5x^6 + s_4x^5 + (s_3 + s_7)x^4 + (s_2 + s_7)x^3 + s_1x^2 + (s_0 + s_7)x + s_7
\end{aligned} \tag{2.3}
$$

Thus XTime can be implemented only by 3 *XOR* gates, with 1 *XOR* gate in the critical path.

AddRoundKey Transformation

In this operation a round key (generated by RoundKey generation block from the secret key) is bitwise *XOR*-ed with state.

RoundKey Generation (Key Expansion)

The RoundKey process in AES generates an array of 4(Nr + 1) 4-byte words. The Key expansion process can be described by the pseudo code listed below.

```
$for\ i = 0\ to\ N_k - 1$
    $w_i = key_i$
$end$
$for\ i = N_k\ to\ 4(N_r + 1) - 1$
    $temp = w_{i-1}$
    $if\ (i\ mod\ N_k = 0)$
        $temp = SubByte(RotWord(w_{i-1}))\ XOR\ Rcon(\frac{i}{N_k})$
    $else\ if\ (N_k > 6\ and\ i\ mod\ N_k = 4)$
        $temp = SubByte(w_{i-1})$
    $end\ if$
    $w_i = w_{i-Nk}\ XOR\ temp$
$end$
```

In RoundKey Generation process, SubByte applies S-box transformation. RotWord cyclically shifts each byte in a word to the left. Rcon is a constant word array with leftmost byte in each array as nonzero. [80] implemented Rcon in combinational logic as shown in Figure 2.8.

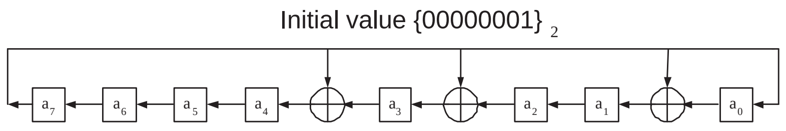


FIGURE 2.8: Rcon in combinational logic.

2.2.3 Asymmetric Cryptography

A common problem with symmetric cryptography is key distribution. For example, consider the case when Alice has to securely communicate with Bob. To use symmetric cryptography to encrypt their messages, Alice and Bob should exchange the *secret key* beforehand and ensure that it is not leaked to anyone else. Also both Alice and Bob have to ensure the secrecy of the key after the exchange. Once

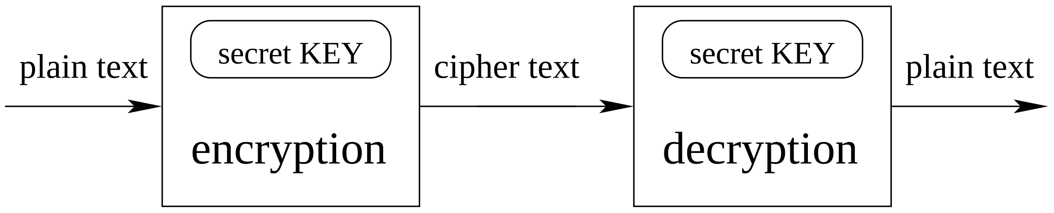


FIGURE 2.9: Symmetric Cryptography

the keys are exchanged, Alice and Bob can securely communicate and both have to maintain one secret key.

Now consider a scenario where Alice has to communicate separately with 100 of her friends. She has to exchange 100 secret keys beforehand and maintain them. Thus key distribution becomes a problem with symmetric cryptography.

In asymmetric cryptography, a different *key* is used for encryption and decryption, as shown in Figure 2.10. RSA [79] and ECC [28, 52] are the most popular asymmetric ciphers. In all asymmetric cryptographic algorithms, the *keys* come as pairs and are commonly referred to as the *public key* and *private key.* The *public key*, as the name suggests, is kept in the public domain so that anyone can access it. The *private key* on the other hand is kept secret. Although these *keys* are related, it would be difficult to deduce the *private key* given the *public key*, as these algorithms rely on solving a computationally intractable problem. For example, RSA relies on the difficulty of solving e^{th} root modulo n (where e and n are the RSA public key) [79].

Now lets say both Alice and Bob publish their public keys, key_{Apub} and key_{Bpub} respectively, while they keep their private keys key_{Apri} and key_{Bpri} in a secure location. When Alice wants to communicate with Bob, she looks up his public key, key_{Bpub}, and uses it to encrypt her message. As Bob is the only one with a private key to decrypt Alice's message, she can send it to Bob via an open channel. Bob uses his private key, key_{Bpri}, to decrypt Alice's message. Similarly, if Bob wants to send a message to Alice, he will use Alice's public key to encrypt his

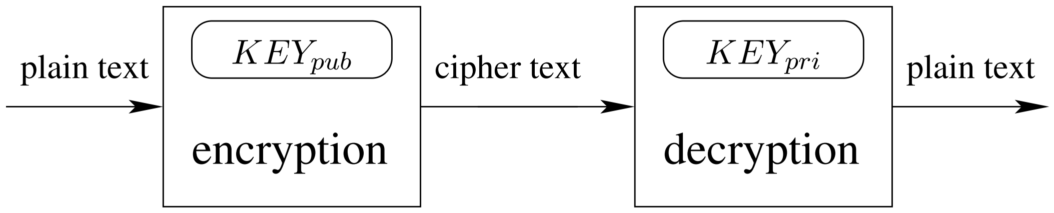


Figure 2.10: Asymmetric Cryptography

message. When received, Alice will use her private key to decrypt Bob's message. Thus using asymmetric cryptography eliminates the need to exchange the *secret key* beforehand. It also removes the burden of sharing their *secret key* with anyone else. Now even if Alice has to communicate with 100 of her friends, she only has to manage her private key.

Although asymmetric cryptography solves the key distribution problem, its performance is quite slow when compared to symmetric cryptography. As a result, a combination of both types are used. For example, Alice uses Bob's public key to encrypt the symmetric algorithm's *secret key* she will use later on. Bob acknowledges this, by first decrypting Alice's message and then encrypting the *secret key* with Alice's public key. Now both Alice and Bob can use a faster symmetric algorithm to communicate with a key they both agreed upon.

2.3 Attacks on Cryptographic Devices

Cryptanalysis is a process to attempt to circumvent the security of a cryptographic algorithm. The cryptanalyst is usually referred to as the attacker. Traditional cryptanalysis is based on the observation of inputs and outputs of the cryptographic device. The cryptanalyst would attempt to extract the *secret key* based on these observations or choice and some knowledge of the implemented algorithms. This has led to the development of mathematically more secure algorithms, such as AES, where extracting the secret keys based on the input output relation is extremely difficult. Even though the algorithm is secure from a mathematical point of view, its hardware implementation can often be used to extract the *secret key*. On devices that implement cryptographic algorithms, the attacker can mount many different types of attacks. Attacks can be classified into invasive types or non-invasive types.

Invasive attacks are those which leave a physical evidence of tampering on the device. Kömmerling and Kuhn [33] have discussed many invasive approaches to attack smart cards, such as de-packing smart card chips, memory reverse engineering, and micro probing. A secure cryptosystem is usually equipped with electromagnetic shielding, low-pass filters, and clock signal generators to protect it from most invasive attacks.

Non-invasive attacks are those that do not physically tamper with the device, instead they use information that is leaked from the device to attack. For either type of attack, it is necessary to have physical access to the cryptographic device. Non-invasive attacks are also commonly referred to as side channel attacks and are discussed in the rest of this chapter.

2.4 Side Channel Attacks

A side channel attack is an attack on cryptographic device which exploits unintentionally leaked information. Side channel attacks are a major threat to secure devices, as these attacks can be carried out with relatively inexpensive components. Without necessary countermeasures to protect secure devices, side channel attacks can allow an attacker to reveal the secret key from these secure devices.

Side channel attacks on smart cards were first discussed in 1996 by Kocher. Since then many variation of side channel attacks have been published. Although Kocher was the first to publish about side channel attacks on an electronic system, the basic concept of side channels existed before. For example many safes that used rotary combination locks were cracked by listening to sounds while trying to manipulate the combination locks . Here sound from the safe is treated as side channel leakage, which is an unintentional leakage from the implementation of the safe. Many movies often depict the process of safe-cracking by listening to noise.

2.5 Side Channel Attack Scenarios

Cryptographic algorithms are used in various applications. An electronic device that implements or executes a cryptographic algorithm is usually called a cryptographic device or cryptographic system or cryptosystem in short. One of the applications of cryptography is in smart cards. Smart cards are like electronic vaults that are used to protect the data they contain. Only after authenticating a card reader, the smart card provides some information. Some of the common applications of smart cards are bank credit/debit cards, GSM phone SIM cards, physical access control cards, pay TV cards, etc. More recently passports are also incorporating cryptographic devices. In all these applications security of the cryptographic devices is of utmost importance. This means the secret key used in these devices should not be accessible by unauthorised persons.

In many applications even the legitimate owner/user of a smart card is also not allowed to access the secret key. An example of this is the pay TV user who is not allowed to make extra money by duplicating and selling subscriptions on his pay TV card. Another example is an electronic purse, where the owner cannot increase the amount held in the electronic purse. Therefore an owner/user of a cryptographic device also has motivation to attack it.

In applications such as e-passports and physical access control cards a third person may have an interest to impersonate the owner/user of such cards. In such cases the attacker may take the cryptographic device for a short period of time to mount an attack and then return the device.

Whatever the motivation for attacking a cryptographic device, it is important to note that access to the cryptographic device is necessary to implement side channel attacks. Apart from the input/output to the cryptographic device, the attacker would be able to measure the side channel information leaked during the device operation.

2.6 Side Channel Leakage Types

As a cryptographic device processes input data, that is it either performs encryption or decryption, it emits information outside of the device that is dependent on the data being processed. This information leakage is not a result of intentional design of the device, but rather a by-product during the device operation. The name *'side channel leakage'* is used as this information can be considered as a side effect of the cryptographic device operation.

As the secret key is a part of the data processed by the cryptographic device, any side channel leaked from a cryptographic device will be related to the device's secret key. Attacks on the cryptographic device, to find the secret key, that use the side channel leakage are called side channel attacks. The most common side channel leakages are execution time, power consumption and electro-magnetic(EM) radiation. Some of the newly invented side channel attacks on PC/servers, called as cache-timing attacks do not need physical access to the target device, instead they need the access to run a program on the target device [63, 67] . These attacks exploit the cache dependency of the executing algorithm to extract the secret key.

Execution Time

Some cryptographic algorithms perform operations based on a single bit of the secret key. For example, in implementations of asymmetric ciphers such as ECC and RSA, operations such as additions and multiplications are controlled by individual key bits [28, 52, 79]. Often a single key bit decides whether addition is performed instead of multiplication. This leads to different execution times, based on the operation selected. Thus, in such algorithms, execution time will be dependent on the secret key. Kocher [30] first published attacks based on execution time. These attacks are referred to as timing side channel attacks or timing attacks. One way to counteract timing attacks is by making all operations consume the same amount of time can prevent timing attacks. We do not consider attacks based on execution time in this thesis.

Power Consumption and Electromagnetic Emanations

Power and Electromagnetic (EM) leakage are the mostly exploited side channel leakage types against cryptographic devices. The majority of cryptographic devices are implemented using CMOS logic. The power consumption of CMOS logic is data dependent. Thus, power consumption as well as the EM field that is caused by the currents flowing in a cryptographic circuit implemented in CMOS leak information about the secret key. Kocher *et al.* [31] first published attacks based on power consumption namely, simple power analysis (SPA) attacks and differential power analysis (DPA) attacks.

2.7 Power Consumption of CMOS Logic

Complementary Metal Oxide Semiconductor (CMOS) logic is the most widely used logic style in designing digital semiconductor devices. CMOS gates consist of two parts, a pull up network consisting of pmos transistors and a pull down network consisting of nmos transistors. These two parts are complementary, i.e, only one of these networks is conductive at a time. For a detailed description of CMOS logic we refer the reader to these text books [75, 108].

Power consumption in CMOS consists of three components, namely 1)dynamic power, 2)short circuit power and 3)static power. Dynamic power is the major source of total power consumption and is the result of charging and discharging of load capacitance. Dynamic power consumption is given by the following equation:

$$P_{dyn} = \alpha\ C_{load} V_{dd}^2\ f \tag{2.4}$$

In Equation 2.4, α is the switching activity factor of the circuit, C_{load} is the load capacitance including the parasitic capacitance, V_{dd} is the supply voltage and f is the circuit's operating frequency. The switching activity factor α is, in turn, dependent on the circuit's input data and is a measure of the $0 \rightarrow 1$ transitions on the output of CMOS gates. For a given circuit, if the supply voltage and frequency are constant so the resulting dynamic power consumption is directly dependent on the processed data. More importantly a $0 \rightarrow 1$ transition consumes a different amount of power than $1 \rightarrow 0$ transition, while $1 \rightarrow 1$ and $0 \rightarrow 0$ transitions (or lack of transition) consume significantly less power. This data dependent power consumption of CMOS circuits makes power analysis attacks possible.

Short circuit power is caused when CMOS gates transition either $0 \rightarrow 1$ or $1 \rightarrow 0$. When the CMOS gate's output changes state, both the pull up network and pull down network are conducting simultaneously for a short period of time. This causes a direct path between the supply voltage and ground and hence power dissipation. Short circuit power consumption is also referred to as direct path power consumption. Like dynamic power consumption, short circuit power consumption is dependent on switching activity factor, α. In some literature, short circuit power is considered as a part of the circuits total dynamic power.

The static power consumption of a circuit is given by the following equation:

$$P_{static} = I_{static} \, V_{dd} \tag{2.5}$$

where I_{static} is the current that flows between the supply rails when the circuit is stable, i.e, no operation (no switching activity). Static power consumption is also referred to as leakage power. As CMOS technology scales down, static power consumption tends to be the major portion of the total power consumption [38]. Static power consumption also depends on the inputs applied to a particular logic gate. For example, in [38], it is shown that a 2 input NAND gate has different static power consumption depending on the input pattern.

Electromagnetic signals are leaked by a cryptographic device in much the same way as its power consumption. Currents flowing though a conductor induces EM emanations. As the power consumed by a cryptographic device varies while data are being processed, so does its EM field.

As power consumption and EM radiation are data dependent, side channel attacks based on them are possible. An attack that exploits power consumption is called a power side channel attack, while an attack that exploits EM leakage is called an EM side channel attack. Power and EM side channel attacks are also referred to as power analysis and EM analysis attacks respectively. Although the source of EM and power consumption leakage is switching activity, EM side channel attacks can be prevented by metal casing around the cryptographic device, while preventing power attacks is more complicated. EM attacks have a different setup than power attacks, however both these attacks can be carried out externally to the target device.

Power and EM side channel attacks are similar in most ways. The source of the side channel leakage for both power and EM is from the switching activity of a device. As such the methods employed for power side channel attacks can also be applied to EM side channel attacks. The only difference is in the probe used and the method employed to measure the side channel leakage. A resistor or a current probe is used to measure the power consumption. Simple home-made coils placed close to the cryptographic device have been successfully used to measure EM side channel leakage in the near field [20, 39, 74]. Mangard has discussed EM side channel attacks in the far field and mentioned that these are more difficult to implement that the near field EM attacks, as there is more noise in the far field EM signals.

2.8 Power Analysis Attacks

In [31], Kocher *et al.* coined the terms simple power analysis (SPA) and differential power analysis (DPA), as a way to categorise power analysis attacks. Later Quisquater and Samyde coined similar terms for EM based side channel attacks, called simple EM analysis (SEMA) and differential EM analysis (DEMA) [74]. Although the experimental setup required for power attacks and EM attacks is different, they are both based on the fact that different transitions lead to different energy consumption. Mangard in his thesis [39] used the terms simple side channel analysis (SSCA) to refer to simple power analysis and simple EM analysis, and differential side channel analysis (DSCA) to refer to differential power analysis and differential EM analysis. In order to perform power analysis attack on a cryptographic device, the attacker should be able to measure the power consumption of the device directly by probing the supply/ground lines to the device or indirectly by measuring the EM field around the device. Another requirement for the attacker is the ability to control and/or observe the inputs and/or outputs to the cryptographic device with the same key.

In Section 2.8.1 hypothetical power model is discussed. Then in Section 2.8.2 simple power analysis is explained and in Section 2.8.3 differential power analysis is explained. Note that the principle described for power analysis attacks also apply to EM analysis attacks.

2.8.1 Hypothetical Power Model

The key components of a DPA flow are the actual power consumption of physical device and the attacker's predicted hypothetical power model of this device, as shown in Figure 2.11. The actual power consumption is obtained by measuring the cryptographic device's instantaneous power consumption while it is operating. Power consumption can be measured directly by measuring the current flowing through supply/ground wires or by measuring the EM field surrounding the device. One of the important steps in DPA attack is the attacker's prediction of the hypothetical power consumption, $H_{1\ldots 2^K,1\ldots N}$ from the intermediate result $I_{1\ldots 2^K,1\ldots N}$. The effectiveness of DPA depends on the attacker's hypothetical power model of the target device [64, 100].

In [64], it has been suggested that attackers can have detailed information regarding the target device. Depending on the level of information the attacker has about the target device, his/her ability to accurately predict the hypothetical power consumption varies. In most cases, information about the algorithm used and the architecture of the device can be sufficient to accurately predict the hypothetical power consumption. Such information can be readily available from the device's data sheet, which is usually available in the public domain. In this section we will discuss common power models used to attack cryptographic devices.

In [100], Tiri and Verbauwhede have summarised power models depending on the level of abstraction used. At the instruction level, the number of instructions used can be a measure of dynamic power consumption. In implementations of asymmetric ciphers such as ECC and RSA, operations such as additions and multiplications are controlled by individual key bits [28, 52, 79]. In such cases, based on the operation executed, the number of instructions executed varies, thus varying the power consumption [100].

At the Register Transfer Level (RTL), the toggle count is a common measure for estimating power consumption. The toggle count is a measure of the number of $0 \rightarrow 1$ transitions for a given circuit. As discussed in Section 2.7, the power consumption of CMOS logic depends on $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions. This model is also called the transition count model or the Hamming distance model, since the number of $0 \rightarrow 1$ transitions depends on the previous state of the circuit. The Hamming distance model has been successfully used to implement DPA in [62]. Another commonly used model at this level is the Hamming weight model. The Hamming weight is a count of the number of bits that are at logic 1. For example, an 8 bit register which has 3 bits at logic 1 and 5 bits at logic 0 has a Hamming weight of 3. In the first DPA publication, Kocher *et al.* [31] used a single bit Hamming weight model. They used a selection function to predict the intermediate result of 1 bit. Depending on this intermediate value, they partitioned the power traces to perform a difference of mean correlation test. This model works in practice, because the energy used to store a logic 1 is different to the energy required to store logic 0. Although single bit attacks work, multiple bit attacks (attacks where predictions of multiple bits are done) are more effective [43].

At the layout level, the power consumption model includes parasitic capacitance along with toggling activity. At this level, parasitic capacitance can be used to implement DPA against a circuit employing dual rail countermeasures [100]. Back-annotated parasitic data has been successfully used to mount DPA against a device employing masking countermeasures [42]. Although information at the layout level provides more insight into the target device, in practise we found that Register Transfer Level information sufficient to mount DPA on various circuits.

2.8.2 Simple Power Analysis

SPA attacks, as the name suggests, is a simple analysis of power consumption of the cryptographic device. These attacks were first published by Kocher *et al.* [31]. To perform SPA, the attacker first measures the cryptographic device's power consumption (directly or by measuring the EM field) while operating on a single plain/ciphertext. This measured power consumption for a single plain/ciphertext is usually referred to as a power trace. The attacker then visually analyses the power trace to determine part of the secret key or an entire key itself.

Some cryptographic algorithms perform operations based on a single bit of the secret key. For example, in implementations of asymmetric ciphers such as ECC and RSA, operations such as additions and multiplications are controlled by individual key bits [28, 52, 79]. In such cases, based on the operation executed, power consumption varies, allowing the attacker to determine the secret key. Simple power analysis has been successfully used to find the secret key of an ECC implementation in [65].

Hardware implementations of symmetric block ciphers are not as susceptible to SPA as asymmetric ciphers. The reason is that hardware implementations of symmetric block ciphers can operate in parallel on the data. Symmetric block ciphers such as AES [57] do not have operations that are dependent on part of the secret key, rather all the data is processed in parallel. However software implementation of block ciphers have been reported vulnerable to SPA [47]. In this thesis, attacks on software implementations of block ciphers are not considered. Although hardware implementation of symmetric block ciphers are less susceptible to SPA, differential power analysis (DPA) attack is quite effective against them. In this thesis we describe DPA attacks on hardware implementations of DES and AES block ciphers.

2.8.3 Differential Power Analysis

Differential power analysis (DPA) attack is a more powerful method than SPA and is a major threat to the security of cryptographic devices. DPA exploits the correlation between the data and the instantaneous power consumption of the cryptographic device [31] . As this correlation is very small, statistical methods are employed to exploit its efficiency. In a DPA, the attacker uses a hypothetical model (discussed in Section 2.8.1) of the device under attack and then statistically analyses the correlation of power consumption from the actual device to the hypothetical model in order to find the secret key. The efficiency of the hypothetical model depends on the capabilities of the attacker and how much knowledge of the implementation he/she has of the cryptosystem.

The process of implementing DPA varies according to the target device and its setup. For example, the attacker can only observe either inputs or outputs of the target device. In another case, the attacker can measure the power consumption of a cryptographic device only while performing decryption. In all DPA implementations, it is common to first acquire the power consumption of a device and later perform analysis.

DPA is used to extract/find the secret key from a cryptographic device. This secret key's bit length depends on the algorithm implemented in the device. It is important to note that the entire secret key is not extracted at once, but rather part of a secret key is extracted at a time until the entire secret key is known. In DPA jargon, this part of the secret key is called a subkey. The choice of subkey depends on the device algorithm, architecture and the attacker's knowledge of the device. If an attacker attempts to extract the entire secret key at once, then the effort to implement the hypothetical power model would be same as a brute force attack. For example, consider a device implementing AES [57] with a secret key of length 128 bits. A DPA attack on the entire secret key, on this device would take $2^{128} = 340,282,366,920,938,463,463,374,607,431,768,211,456$ key hypotheses, which requires similar effort to that of brute force attack. Even if we could process a billion billion keys (10^{18}) per second, it would still require about 10^{13} years to finish all possible key hypotheses [82] . However if a subkey of 8 bits is chosen, then DPA on this 8 bit sub key would require $2^8 = 256$ iterations and then entire secret key would require $(2^8) * 16 = 256 * 16 = 4096$ iterations, which is a small number when compared to the brute force attack.

A DPA attack on a cryptographic module performing encryption is described below. This process is also depicted in Figure 2.11. The cryptographic device's secret key length is **S** bits, while the length of the subkey chosen for DPA attack is **K** bits. The process described below assumes that the device is encrypting and that the attacker can control the inputs to this device.

1. The power consumption of the cryptographic device is recorded while it encrypts N different plaintext inputs with the same key and is denoted as a matrix $P_{1...N,1...T}$, where T is the number of points that are recorded per encryption. The number N is usually referred to as the *number of traces.*

2. The attacker chooses an intermediate result of the executed algorithm that is a function of the plaintext and the subkey. Based on the plaintexts and all possible values for the sub-key, hypothetical values for the intermediate results are calculated as a matrix $I_{1...2^K,1...N}$ (called the hypothetical intermediate results) where K is the number of subkey bits and 2^K is the number of possible values of the subkey. We also refer to this intermediate result as the *attack point.*

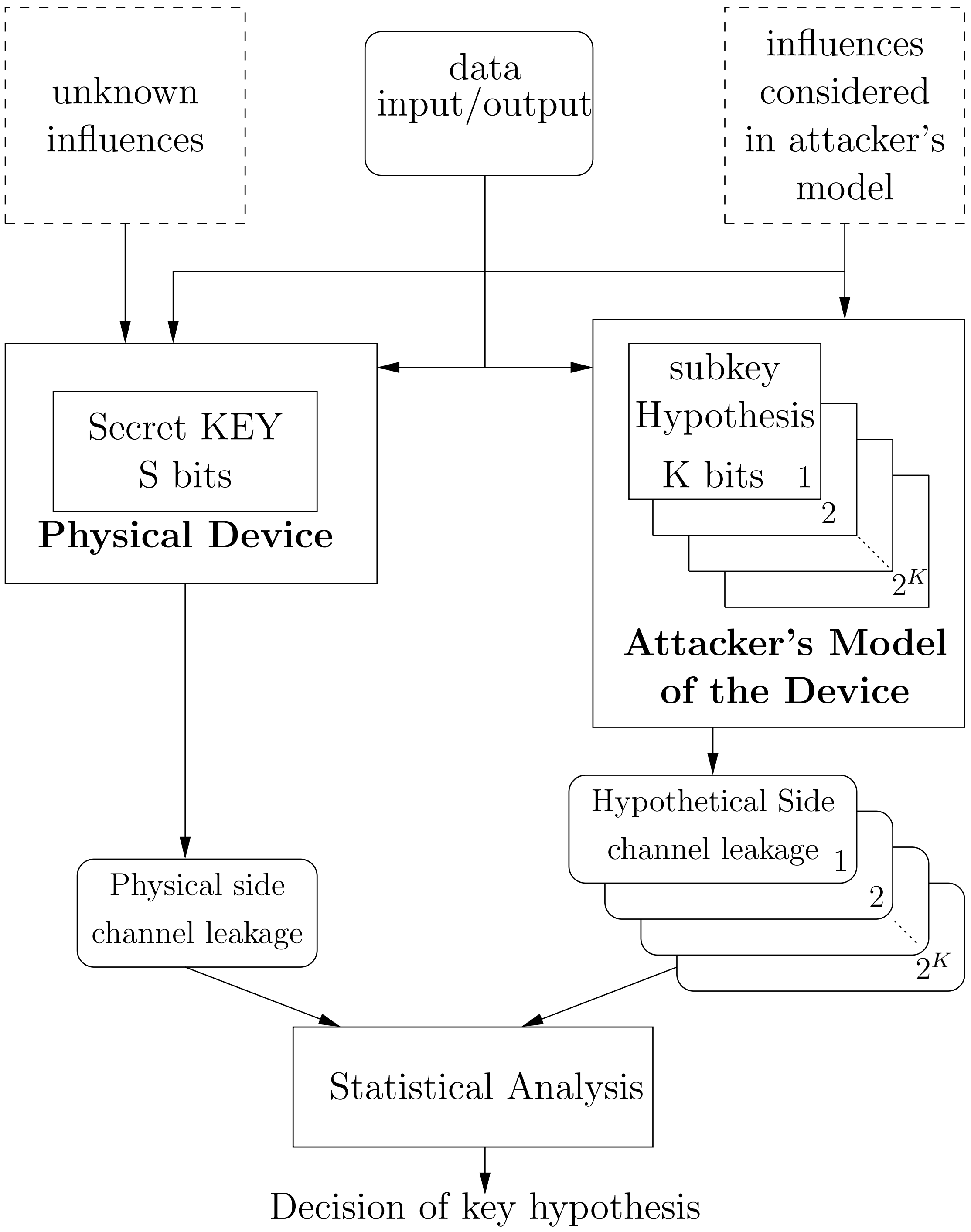

unknown influences
data input/output
influences considered in attacker's model
Secret KEY S bits
Physical Device
subkey Hypothesis K bits 1
2
2^K
Attacker's Model of the Device
Hypothetical Side channel leakage 1
2
2^K
Physical side channel leakage
Statistical Analysis
Decision of key hypothesis

FIGURE 2.11: Principle of differential side channel analysis

3. The attacker then determines a hypothetical power consumption value $H_{k,n}$ for every $I_{k,n}$. The absolute values of the $H_{1...2^K,1...N}$, are not important, only the relative distances between the values are relevant.

4. The attacker reveals the correct subkey by correlating the hypothetical power consumption $H_{1\ldots 2^K,1\ldots N}$ with the power traces $P_{1\ldots N,1\ldots T}$.

There are two correlation methods that are commonly employed in a DPA attack. They are the difference of mean method [31] and the Pearson correlation method [62].

Difference of Mean Method

The Difference of Mean (DM) is the original method proposed by Kocher *et al.* [31]. The basic idea of this method is to split the power traces into two groups for each key hypothesis based on a so-called selection function. In [31], Kocher *et al.* first partitioned the captured traces $P_{1\ldots N,1\ldots T}$ into two sets, based on a selection function. The matrix $H_{1\ldots 2^K,1\ldots N}$ corresponds to the selection function for the DM method. The means of the power traces in both sets are calculated and the means of one set are subtracted from those of the other set. In the original paper, Kocher *et al.* [31] used the possibility of a particular bit to partition the traces into two sets (if the bit is 1, add current trace to set one else the other set). Rather than a value of a bit, a threshold α can also be used to partition the traces based on $H_{1\ldots 2^K,1\ldots N}$ [39]. For example, when the matrix $H_{1\ldots 2^K,1\ldots N}$ contains the Hamming weight or the Hamming distance of an intermediate result of 8 bits, then α of 4 can be used to partition the traces. Equation 2.6 shows how a set can be partitioned. Here N_{high} is the number of samples for which the condition $H_{2^K,N} > \alpha$ is true. The differences of means, $R_{1\ldots 2^K,1\ldots T}$, are calculated for every key hypothesis according to Equation 2.7.

$$P_{Set1} = \frac{1}{N_{high}} \sum_{\forall N | H_{2^K,N} > \alpha} P_{N,T} \tag{2.6}$$

$$R_{2^K,T} = P1_{(\forall N|H_{2^K,N}>\alpha),T} - P1_{(\forall N|H_{2^K,N}\leq\alpha),T} \tag{2.7}$$

The resulting DM matrix, $R_{1\ldots 2^K,1\ldots T}$ will have a difference of mean trace for every key hypothesis. The difference of mean trace for correct key hypothesis will have significantly visible peaks when compared to the the other result traces [31].

Messerges *et al.* have shown that using a different α value for the selection function results in different number of traces required for a successful DPA attack, and called their attack, for $\alpha > 1$, a multiple bit DPA attack. Based on an attack on a smart card executing DES, they showed that a multiple bit DPA usually requires fewer traces than a single bit DPA.

Pearson Correlation Method

The Pearson correlation is a common method to determine a linear relationship between two variables. The formula for Pearson correlation takes on many forms. A commonly used formula is shown in Equation 2.8.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}} \tag{2.8}$$

where, X and Y are two arrays with equal numbers of entries and N, $\sum X$ and $\sum Y$ are the sums of all the elements in X and Y respectively.

The calculation of Pearson correlation between matrices $P_{1\ldots N,1\ldots T}$ and $H_{1\ldots 2^K,1\ldots N}$ for every fixed T and K leads to $R_{1\ldots 2^K,1\ldots T}$ of correlation coefficients. Since the measured power consumption matrix, $P_{1\ldots N,1\ldots T}$ represents a sequence of numbers for every plaintext, a representative value from this matrix, $RP_{1\ldots N}$, is used for calculating the Pearson correlation. This representative matrix can either contain an average value or a peak value from the original matrix.

For the correct subkey, 2^{Kc}, the matrices $RP_{1...N}$ and $H_{2^{Kc},1...N}$ are highly correlated. Since the values $RP_{1...N}$ and $H_{2^{K}!=2^{Kc},1...N}$ are largely uncorrelated, the correlation coefficients $R_{2^{K}!=2^{Kc}}$ are significantly lower than $R_{2^{Kc}}$. If N is sufficiently large in an attack the difference between correlations can be detected in matrix $R_{1...2^{K},1...T}$. In this case, one correlation coefficient of $R_{1...2^{K},1...T}$ is larger than others and this position leads to the correct subkey.

Örs *et al.* [62] implemented DPA on AES ASIC implementation using the Pearson correlation method. The intermediate result chosen was the 8 MSB's of the initial round of AES where key is XOR-ed with Plain text and loaded into registers. The hypothetical power consumption was built by measuring the number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions on the 8 MSB registers of the initial round of AES. Örs *et al.* [62] showed that almost 4000 encryption rounds (of power measurements) are sufficient to extract the secret key from an AES ASIC implementation.

Higher Order Differential Power Analysis and Template Attacks

Higher order DPA attacks use hypotheses about the side-channel leakage of a device at two or more moments in time. Higher order DPA has been shown to be effective against masking countermeasures in [50, 105]. However, higher order DPA attacks are significantly more difficult to mount in practise than the normal DPA attack [39].

In template attacks [15], an adversary needs to have access to an identical device as the one on which the attack will be mounted. The adversary then builds a template of side channel leakage, based on the identical device, for all possible sub-keys under attack. This step is usually referred to as a profiling step. After side channel information is measured from the device under attack, it is matched to the previously built template. The idea is to reduce the possibility to a few keys or to even find the correct key. Template attacks are difficult in practise, as the availability of an identical device and the ability to program it is slim. Nevertheless, designers of cryptographic devices need to be aware of higher order and template attacks and they need to make sure that their devices are resistant against them. In this thesis, we have not considered higher order attacks or template attacks as we found that the basic first order DPA attack to be sufficient to extract the secret key from a cryptographic device.

2.9 Summary

This chapter has introduced some basic background material on cryptography. Power consumption of CMOS circuits is also reviewed. Security is an ever changing model. As the old cryptographic algorithms are broken, new algorithms are designed. However the current standard algorithms, AES for example, have no known mathematical attack. And as AES supports either 128, 192 or 256 bit length keys, a brute force attack on it using current computational power would be infeasible. However implementation of these mathematically secure algorithms can leak side channel information and DPA is well known to exploit even the faintest of the information available.

Next chapter, Chapter 3, introduces existing countermeasures to power analysis attacks and Chapter 4 discusses power analysis attacks in practise.

Chapter 3

Differential Power Analysis Countermeasures

3.1 Introduction

Power attacks pose a severe threat to the implementations of secure devices. During the last few years a lot of countermeasures have been proposed to prevent power side channel attacks;. these countermeasures range from ad-hoc solutions to elaborate, well understood solutions. As power attacks rely on the data dependent power consumption nature of the cryptographic device, power analysis attack countermeasures either try to eliminate this data dependency on power consumption or try to minimise the data dependent power consumption beyond a point where the attacks are not feasible.

Every countermeasure carries a certain cost, such as increase in silicon area, increase in overall power consumption, or decrease in performance. Furthermore, the effort that is needed to integrate the countermeasure into the design flow also represents a cost. In order to understand how countermeasures are used, we first establish some understanding of the digital design flow. In Section 3.3 we classify countermeasures according to their approach to prevent DPA. Section 3.4 discusses algorithmic countermeasures, Section 3.5 discusses various architectural countermeasures, including constant power consumption, randomising and gate level. In Section 3.6 we compare countermeasures and finally conclude the chapter.

3.2 Digital Design Flow

Digital circuits can be implemented in one of the following two ways: 1) as application specific integrated circuits (ASIC), 2) as field programmable gate arrays (FPGA). ASICs, as the name suggests, are designed and manufactured for a specific application or group of applications. FPGAs are also integrated circuits, but they are manufactured in advance, and have the ability to be field programmable to work as intended by the designer. FPGAs contains, among others, collection of configurable logic block (CLB) and configurable interconnect fabric. CLBs are usually formed of programmable look up table (LUT) and a register element. It is the ability to programme these CLB to function as necessary and the ability to programme the interconnect fabric to connect these CLBs in the required way, that gives FPGAs their programmable nature. CLBs are also refered to as logic slice (and sometimes just slice). ASICs, once manufactured, have fixed functionality, but FPGAs can be reprogrammed to different designs. Since ASICs are designed for a specific functionality, they are usually out perform their FPGA counterparts. As FPGAs are manufactured in advance, they have fixed resources (usually measured in units of CLBs, embedded memory and optionally fixed functional blocks such as multipliers), so a designer has to choose an appropriately sized FPGA for their designs. For more information on how FPGAs work, we refer the reader to these books [44, 75, 109]. In this section, we briefly discuss ASIC and FPGA design flows.

3.2.1 ASIC Design Flow

There are two common ways of designing ASIC circuits: one is by using a full custom design flow, the other by using a standard cell design flow. Full custom refers to the design flow where layout of the design is done from grounds up. In standard cell design flow commonly used logic gates are pre-designed for a specific fabrication process. These pre-designed gates are called standard cells. All these standard cells are compiled into a technology library that can be used by various tools, called standard cell libraries or standard cell design kits. These standard cell design kits are then shared across designs to reduce their development costs. Full custom design flow is common for analog designs, while standard cell design flow is common for digital design. Digital design is a vast, expanding and evolving area. The most basic steps of the flow are shown in Figure 3.1.

The design starts from the target product's requirement: features and functionality that need to be supported, area (cost) of the design, power consumption of the design, and the performance of the design (the speed at which the design can work). These requirements then form the specification of the design to be implemented.

Behavioural description is then created to analyse the design in terms of the functionality required and to some extent the performance required. Behavioural description is usually done using high level languages such as c/c++, system c and system verilog.

Using the design specification and behavioural description, Register Transfer Level (RTL) design is created. RTL is described in hardware description languages such as verilog and VHDL. This transformation, from behavioural to RTL, can be done either manually by design engineers or automatically by using behavioural synthesis tools. RTL design is then functionally verification. At this stage, power estimation can also be done on the RTL description.

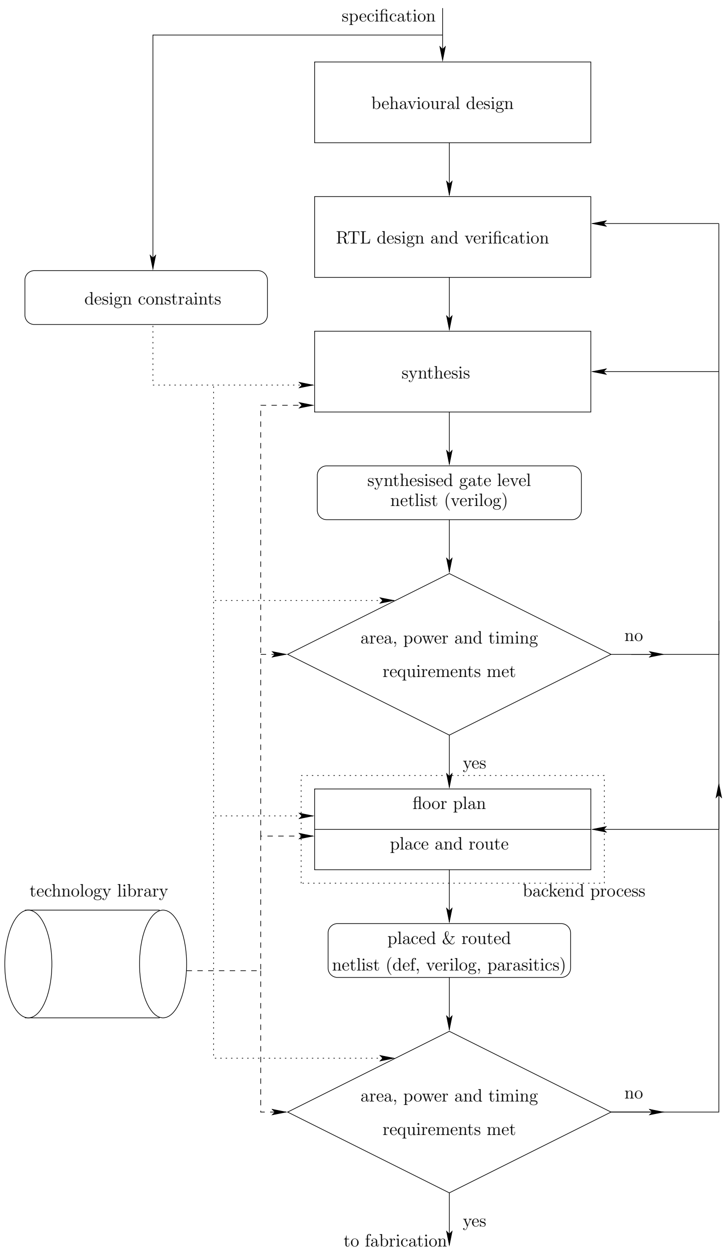
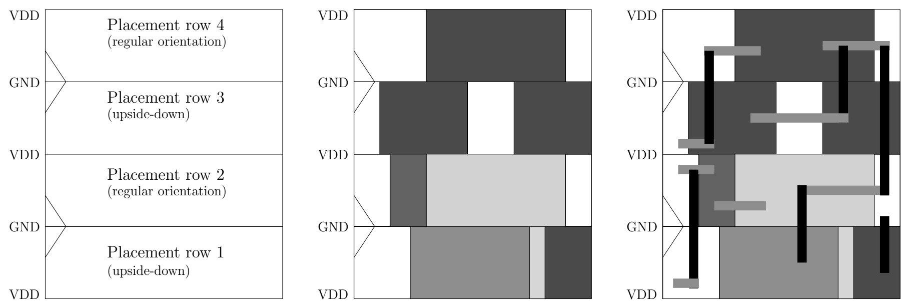

specification
behavioural design
RTL design and verification
design constraints
synthesis
synthesised gate level netlist (verilog)
area, power and timing requirements met
no
yes
floor plan
place and route
backend process
technology library
placed & routed netlist (def, verilog, parasitics)
area, power and timing requirements met
no
yes
to fabrication

Figure 3.1: Digital design flow overview

RTL is then converted into a netlist of logic gates using logic synthesis tools. Along with the RTL design, logic synthesis tools require the target technology library and design constraints. Design constraints to the logic synthesis tool is also derived from the product spec and include clock frequency details, input and output pin delays, power constraints.

Resulting netlist from logic synthesis is then verified to be functionally equivalent to the RTL design. This is accomplished by using formal equivalence checking tools and by gate level simulations. Most importantly the resulting netlist is checked against design constraints, namely area, speed and power. Area is reported directly by the synthesis tool. Power estimation is done in two steps: first the synthesised gate level netlist is used to run gate level simulations and the switching activity of this netlist is collected into a file during the simulation. In the second part, the synthesised netlist, switching activity file and technology library as passed to a power estimation tool and these tools then report average power, peak power and instantaneous power consumption. Finally the synthesised netlist is checked to see if it meets the timing constraints by using a static timing analysis tool.

The synthesised netlist is then passed to the place & route tool. The place & route flow usually involves the following steps, shown in Figure 3.2. First a floor plan is made (3.2(a)). This is where the aspect ratio (or the dimensions) of the chip is fixed. Next the standard cells are placed (3.2(b)) and finally the wires are routed(3.2(c)). Place & route step as also referred to as the backend process.



(a) Standard backend flow: create a floor plan (b) Standard backend flow: place the standard cells (c) Standard backend flow: route the wires

FIGURE 3.2: Standard backend flow overview

The result of this step is a layout netlist file in design exchange format (DEF) [12]. From this placed & routed gate level netlist and the parasitic routing capacitance can be extracted. This netlist is also checked against design constraints: area, speed and power. Parasitic routing capacitance is also used in the constraints analysis. Additionally, signal integrity (SI) and cross talk are also analysed at this step. Placed & routed netlist is also verified to be functionally equivalent to the synthesised netlist. Once all the constraints are met, the placed & routed design is sent to the fabrication facility in the form of a GDSII netlist. This final constraint checking step is called as sign-off step.

Engineering Change Order

During a design process, specification can change after synthesis or the place & route step, this could be a result of a change in product requirement or a bug in the RTL design implementation. Such late changes in specification are called as Engineering Change Orders (ECOs). If an ECO is required, then designers change the synthesised netlist, manually or with the help of commercially available ECO tools, to reflect to the new specification.

3.2.2 FPGA Design Flow

FPGA design flow shares some common steps with the ASIC design flow: product requirement, specification, behavioural description, design entry and functional verification are identical. Synthesis and place & route steps differ from the ASIC flow.

For the FPGA synthesis, the target FPGA needs to be specified instead of a standard cell library. The result of the synthesis tool is then passed to the FPGAs implementation tool. This implementation tool is normally provided by the FPGA vendor. The implementation tool does the following steps: 1) translate, 2) map, and 3) place & route. In the first step, the synthesised netlist and design constraints are translated into an internal design format. This design representation is then mapped into the target FPGA, i.e, the design is fitted into the available FPGA CLBs. The place & route then places all the CLBs and routes the connections between them. The place & route tool also provides a back-annotated delay file, that can be used in functional verification and timing analysis. FPGA vendors also provide tools which translate the placed & routed design into a bitstream, a format that is necessary to program the FPGA.

3.2.3 Evaluating DPA Resistance

One way to know if a design is resistant to DPA attack is to run power estimation simulations and analyse for DPA resistance on the simulation power consumption. More details about simulation based DPA is presented in Section 4.3. Another way to know if a design is resistant to DPA attack is to implement it on a FPGA device and mount DPA on this device, more details about this are presented in Section 4.4. In general, aim of a DPA countermeasure is to completely prevent a DPA attack, and when not possible, to make DPA attack difficult by increasing the number of traces required for an attack. So, a design flow intended to create DPA resistant designs should include DPA analysis along with area, power and timing in the sign off analysis.

3.3 Classification of Countermeasures

Countermeasures can be classified in many different ways; one of them being the level at which they are applied such as gate level or algorithmic level. In this thesis we classify them according to their approach to counteract the power analysis attacks. In general there are five approaches to counteract power analysis attacks.

The basic idea of the first approach is to change the key of the cryptographic module frequently, so that the attacker cannot capture enough traces to mount a successful attack. Keys that are used for a few cryptographic operations are called ephemeral keys. These ephemeral keys are based on a master key and are generated when necessary. Power attacks can still be mounted at the time of generation of ephemeral keys, so care should be taken to protect against DPA. These kinds of countermeasures are applied at the protocol level and are usually referred to as protocol level countermeasures. For example, it is suggested to use a hashing algorithm (SHA256) on the key to generate a new one, so that the newly generated key can replace the existing one, and to use the new key in the current transaction; this process repeats for every transactions. The idea is that the same key should not be used again [29]. Kocher suggested that cryptographic protocols should be designed to withstand certain amount of leakage and that the cryptographic devices used with these protocols should be validated to make sure that they do no leak more information than assumed in the protocol [29]. Number of transactions required to successfully implement a DPA can be used to quantify the about of leakage.

The second approach to counteract power attacks is to randomise the intermediate results of the cryptographic module. The idea is to make the power consumption of the randomised intermediate results uncorrelated to actual intermediate results. This approach is generally referred to as masking. Masking can be applied at two levels. One is at the algorithmic level and the other is at the gate level.

The third approach to counteract power attacks is to try to make the power consumption of the cryptographic module independent of the data it processes, by building a device that consumes the same amount of power for any combination of inputs. This can be achieved by using dynamic and differential logic or current mode logic circuits.

The fourth approach to counteracting power attacks is to make the power consumption of the cryptographic device constant, even when no data is processed. These countermeasures employ analogue circuitry to achieve this. Note that this approach is different to the third approach in that in this countermeasure the instantaneous power is always the same, whereas in the third approach, power consumption for different inputs is the same.

The aim of the fifth approach is to make the power consumption of the entire device random enough that the power attack is difficult to implement. This can be achieved by randomising the sequence of cryptographic operations, by randomly varying the supply voltage and frequency, by randomly precharging the input gates or by adding noise generators. Such countermeasures are generic in nature and can be applied at different levels.

The countermeasures described in the above approaches to counteract power attacks can be applied at different implementation levels in a design flow. These implementation levels are algorithmic-level, gate-level, architecture-level and protocol-level. Some of the countermeasures are suitable only at specific level, whereas the others are more generic in nature and can be applied at more than one level. In the next sections countermeasures at different implementation levels are discussed in detail.

3.4 Algorithm Level Countermeasures

While designing a secure system, designers first think of the algorithms to be implemented. Then they decide whether to implement this algorithm as dedicated hardware or software running on an embedded CPU or both, depending on the constraints and available resources. Although countermeasures can be used at different abstraction levels, such as gate level, there exist some algorithmic level countermeasures to prevent power attacks. The basic idea of algorithmic level countermeasures is to rewrite the cryptographic algorithm such that it does not leak any side channel information. As the designer of algorithms has no influence on the implementation details, the only way to prevent side channel leakage is to use masking techniques.

In recent years, algorithmic level masking has received a lot of attention. In this section we cover the basic principles of masking and discuss their overheads. Basic idea of masking is to replace every intermediate result i with a masked result r and a mask m; the mask m is assumed to be randomly generated. This makes correlating intermediate result i with power consumption difficult as it would require guessing the correct mask for each trace. The challenge of masking schemes is to modify the intermediate functions that are used in a block cipher in such a way that they work with masked inputs. While this is straightforward for linear functions like additions and permutations, it is non-trivial for non-linear functions. Additive masking is normally used for linear function, eg: $r = i \oplus m$ [39]. Special care needs to be taken for non-linear functions and is normally dependent on the algorithm. In AES, Sbox function is the non-linear function. Below articles are some of the masking scheme for AES. The method presented in [1] involves adding a mask to the plaintext, removing it before the Sbox operation, replacing it with a multiplicative mask and after the non-linear Sbox operation, this multiplicative mask is replaced with the original mask. The method presented in [102] is similar to the above one, except that the same mask is used for both linear and non-linear functions and a new mask is generated for every AES encryption round. The method presented in [66, 72, 73] uses the same mask for linear and non-linear (Sbox) functions of AES, and uses a correction term to the result of Sbox.

The modified algorithm (masked algorithm) can be implemented in software or in hardware, which is an advantage to algorithmic level masking. However the changes applied to one algorithm cannot be applied to another algorithm. Furthermore algorithmic masking cannot be automated (to date no one has published such claims), which is a drawback when compared to gate level masking.

Mangard *et al.* [42] has successfully attacked two variants of masked (algorithm) AES ASIC implementation: one based on [1] and the other based on [66]. In this article Mangard *et al.* used toggle information of the Sbox combinational logic to build the hypothetical power model. This was achieved by using a gate level simulation of a back annotated Sbox netlist. Although it is highly improbable that an attacker can get the detailed layout of the secure chip, never the less it is shown that masked algorithms can be cracked, if such information is available. Mangard and Schramm [40] have shown that the attack on masked AES was possible due to the glitches in the Sbox region.

3.5 Architectural Level Countermeasures

Architectural level countermeasures cover a very broad spectrum. Any countermeasure that cannot be classified into algorithmic level can be classified as architectural level. Generally, approaches three, four and five (discussed in Section 3.3) fall into this level. We first discuss approach four (constant power consumption), then discuss approach five (randomisation) and finally discuss gate-level countermeasures (masking and dynamic & differential logic).

Researchers in the asynchronous community say that asynchronous designs provide better security than synchronous designs, as there is no clock signal to reference power traces. Moreover any changes in power supply or temperature do not affect asynchronous designs (preventing power glitch, clock glitch and extreme operating condition attacks). We do not cover asynchronous designs in this thesis for this reason: lack of CAD tool support for fully automated design and the lack of acceptability as an industry standard. For readers interested in asynchronous DPA countermeasures we refer to the following papers [7, 17, 35].

3.5.1 Constant Power Consumption using Analog Circuits

In this section we discuss countermeasures that use analogue circuits to compress the power leakage.

3.5.1.1 On Chip Signal Compression Countermeasure

This countermeasure adds a suppression circuit to existing cryptographic hardware, without modifying the actual cryptographic system [76]. Figure 3.3 shows an architecture diagram. The instantaneous current drawn by the cryptographic hardware is sensed and an appropriate current is shunted so that the total current drawn from the supply shows less variation. This countermeasure comes at the expense of area and more power consumption. Ideally the op-amp should have infinite bandwidth and zero response time. The average power for a circuit with this countermeasure will be the peak power of the same circuit without countermeasure [76].

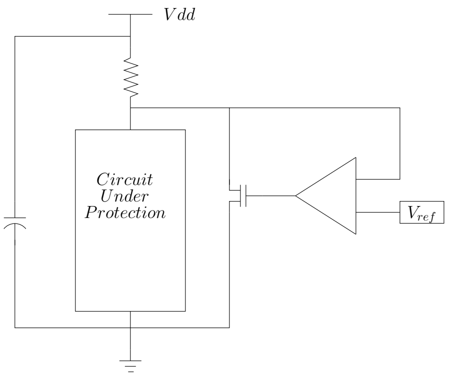


FIGURE 3.3: On chip signal compression, adapted from [76]

Similar techniques have been proposed in [49, 53]. In [53], it is shown from simulations, that the cryptographic system current variation was reduced by between 87%-70% and the extra power consumed to mask the side-channel attacks represents between 21.6%-12% of the total system power respectively. The problem with using such techniques is that they cannot completely secure the device against

EM attacks. These techniques also have very high power consumption, and often require off chip capacitance. The above articles did not report area overheads.

3.5.1.2 Real Time Current Flattening Countermeasure

Muresan and Gebotys [54] presented a real time current flattening technique called PAAR (power analysis resistant architecture), as shown in Figure 3.4. The idea of this countermeasure is to always maintain the current consumption of the processor within two programmable limits. Ideally these limits are close by, so that the processor would always draw the same amount of current.

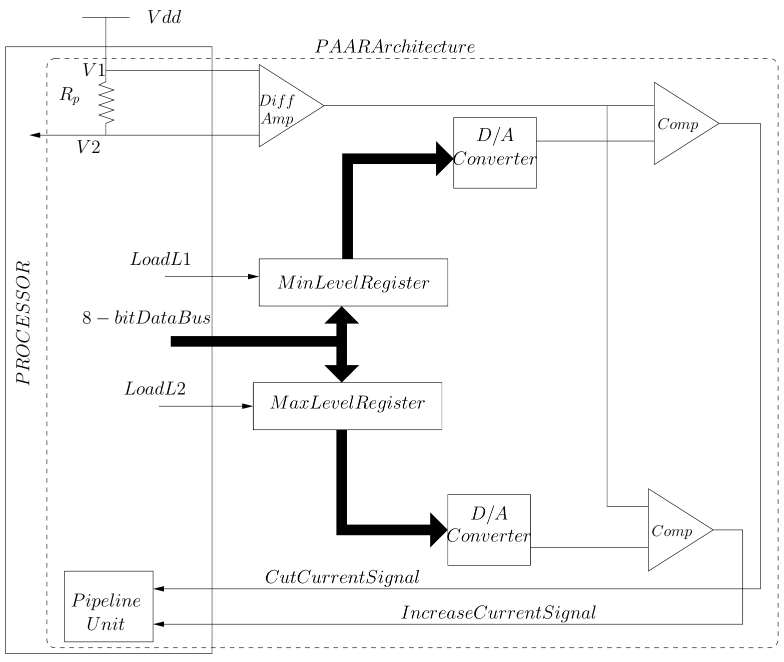


FIGURE 3.4: Real time current fattening architecture (PAAR), adapted from [54]

PAAR has two modules called Feedback Current Flattening Module (FCFM) and Pipeline Current Flattening Module (PCFM). FCFM is responsible for measuring the instantaneous current consumption at the processor's supply pin and generating two feedback signals to PCFM. PCFM is responsible for inserting non functional instructions into the pipeline in order to bring the current consumption of the processor to a value that is within two programmable limits.

3.5.2 Randomising Countermeasures

Randomising countermeasures try to randomise the execution of algorithm along with some random data. These countermeasures are quite generic in nature, and can be applied at hardware level or software level. If random delays occur during the execution of a block cipher, the power traces measured by an attacker are not located at the same position in all of the measured power traces. This random shift of traces decreases the correlation of the intermediate results and power consumption. The most common randomising techniques are applied at the software level, that is, to instructions that execute on a microprocessor.

3.5.2.1 Randomising at the Software Level

One of the randomising countermeasures is random process interrupts (RPI) [16]. The idea is to randomly interrupt the processor while it is executing the cryptographic algorithm, thus randomising the execution sequence and power consumption. Clavier *et al.* [16] have also discussed how an attacker can decrease the effects of RPI. There has been some work on how many more traces are needed if randomising techniques such as RPIs are used. In particular [16, 39, 64] have theoretical formulae for the increase in number of traces when RPI is used. Work in [26, 45, 46] discusses the possibility of embedding randomness into the processor itself.

3.5.2.2 Randomising at Hardware Level

The basic idea of randomising countermeasures at the hardware level is to randomly vary one or more variables that affect the overall power consumption. Dynamic power consumption is given by Equation 3.1. These variables are supply voltage (Vdd), frequency (f), switching activity (α) and load capacitance (C). Switching activity, is data dependent.

$$P_{dyn} = \alpha \, C_{load} V_{dd}^2 \, f \tag{3.1}$$

Yang *et al.* [112] have proposed randomly changing voltage and frequency to prevent SCA. Yang *et al.* called their idea random dynamic voltage and frequency scaling (RDVFS). Dynamic voltage and frequency scaling is a well known method to reduce power consumption [68]. However the idea of RDVFS is to randomly vary frequency and voltage to prevent SCA. The main strength of RDVFS comes from the fact that the exact time of intermediate operations is not fixed, due to random frequency scaling. Voltage is scaled as a side effect of frequency scaling [112].

Benini *et al.* [5] proposed an energy aware design technique for DPA resistance. This countermeasure is around power managed units as shown in Figure 3.5

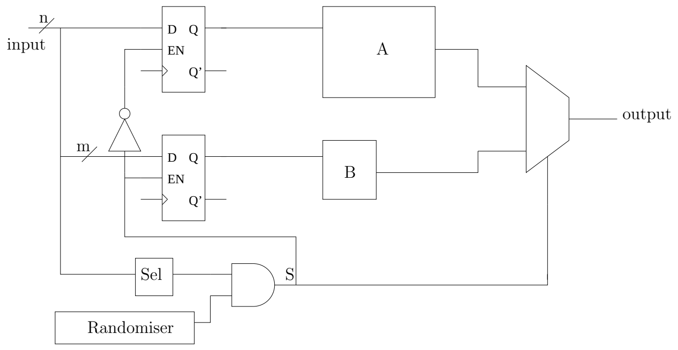


FIGURE 3.5: Power managed units, adapted from [5]

Block B is similar to block A in functionality, but with lower power cost. Block B might implement the typical behaviour of block A, but with a fewer number of inputs and outputs. *Sel* selects which of the blocks A or B should be used for computation. As a simple example, consider a 32 bit computation involving X + Y. The inputs X and Y can range from $0 to 2^{32} - 1$. Block A can be a full 32 bit datapath, whereas block B can be a 16 bit datapath. *Sel* block can then see if the upper 16 bits are zero to enable the B block.

In order to randomise the power consumption, a Linear Feedback Shift Register(LFSR) can be used with the *Sel* signal, as shown above. This introduces randomness to the power trace. If this randomness is not truly random or changes less often, the attacker will still be able to determine the secret key. Also, attacks presented in [62] target registers loading data. If the data loaded into registers was plain text and the key, then this countermeasure would be useless in defending against the DPA attack.

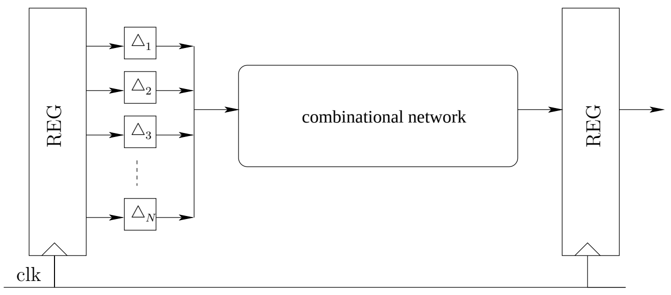


FIGURE 3.6: Random delay countermeasure overview

Another kind of hardware randomisation was proposed by Bucci *et al.*. One of their papers [9] discusses random delay insertion. The overview of this countermeasure can be seen from Figure 3.6. The idea is to randomly change delays to scramble the power consumption. However this technique has a drawback, as all the delay elements are connected to the same flip-flops, total power consumption will always be the same irrespective of the delay element chosen.

Bucci *et al.* [8] have proposed the use of random precharge logic to prevent SCA. The overview of this countermeasure can be seen from Figure 3.7. The idea is to randomly precharge all the combinational gates with a random value generated from the random number generator(RNG). As register elements cannot loose their state value, a redundant register is used to load the random value. Further, to switch between normal operation and random pre-charging a MUX is used to

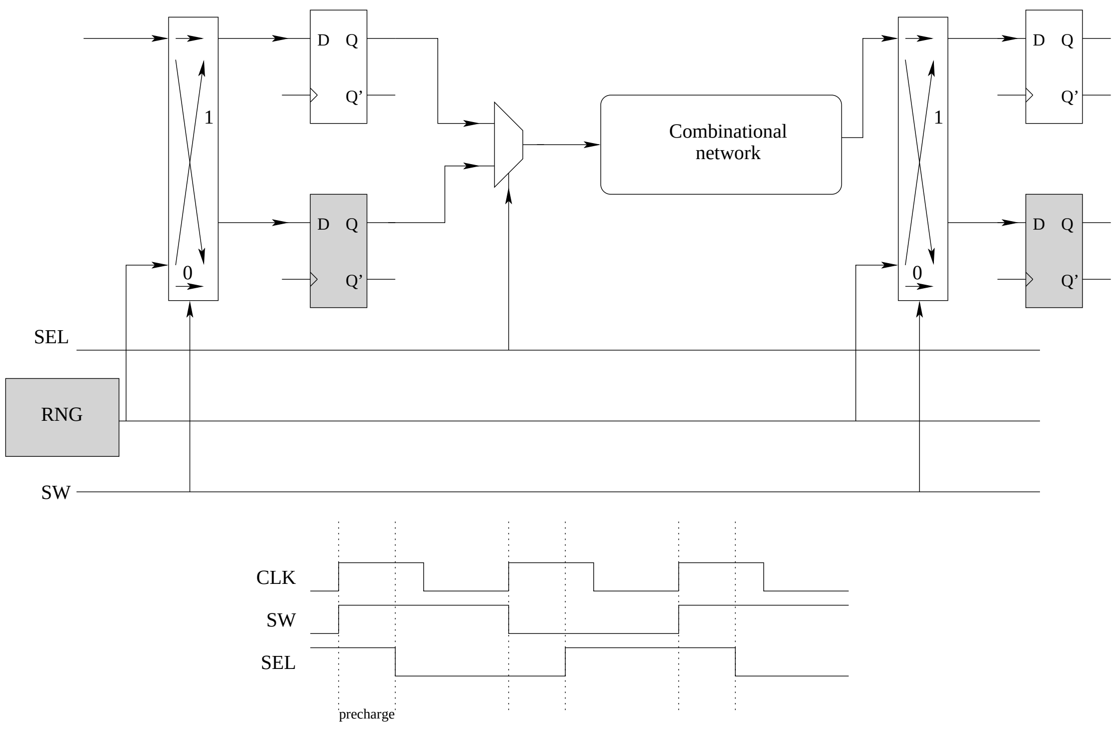


FIGURE 3.7: Random precharging countermeasure overview

control the output of the compound register. Bucci *et al.* have reported that they could reduce the DPA correlation but could not prevent an attack. However, they did not report the average number of random cycles for a single cycle of normal operation.

3.5.3 Gate-Level Countermeasures

In a standard cell design flow, RTL code is synthesised and mapped into technology specific standard cells. These standard cells are also referred to as logic gates. In CMOS technologies, power consumption of these logic gates is strongly dependent on the data processed. This is the reason for power attacks to be successful. However it is possible to design logic gates such that the power consumption of the logic gates is independent of the processed data.

Gate-level countermeasures are integrated into existing design flows after the logic synthesis phase, as shown in Figure 3.8. The dotted line shows the normal design flow. An extra step is added to translate the synthesised netlist into the appropriate gate level netlist, depending on the translation rules. These translation rules vary for different gate level countermeasures. The translation step can be a simple program to change the name of the normal logic cell into a DPA resistant logic cell. Valentini *et al.* [104] have investigated various ways to translate a normal gate-level netlist into a DPA resistant gate level netlist and suggest using an OpenAccess [83] based approach for more stable, modular and flexible solutions.

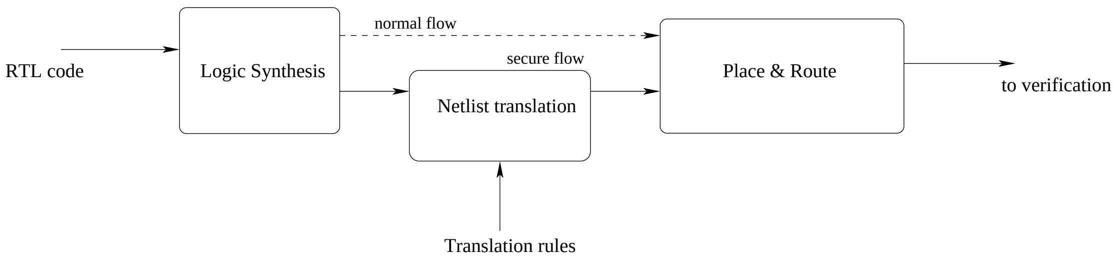


FIGURE 3.8: Integration of gate level countermeasures

In practice there are two ways to achieve data independent power consumption. The first is to use gate-level masking and the other is to use dynamic and differential logic design.

3.5.3.1 Dynamic and Differential Logic

CMOS logic is the most widely used logic style in digital design. It is a well understood fact that a CMOS circuit's power consumption is data dependent [75]. For example consider a CMOS inverter shown in Figure 3.9. It draws power from the Source Vdd only when its output changes state from low to high. During this transition stage some part of the current is stored in the load capacitor. The inverter discharges the current in the load capacitor when its output changes from high to low. For the remaining transitions, where the output does not change state (high to high and low to low), there is no current dissipated. This data dependent power consumption is the reason why power attacks are possible.

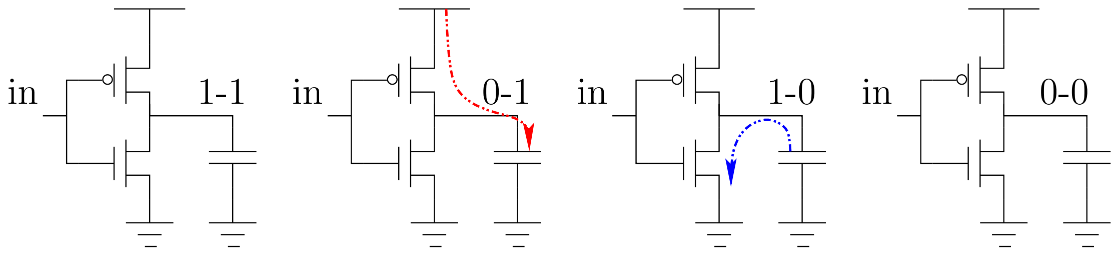
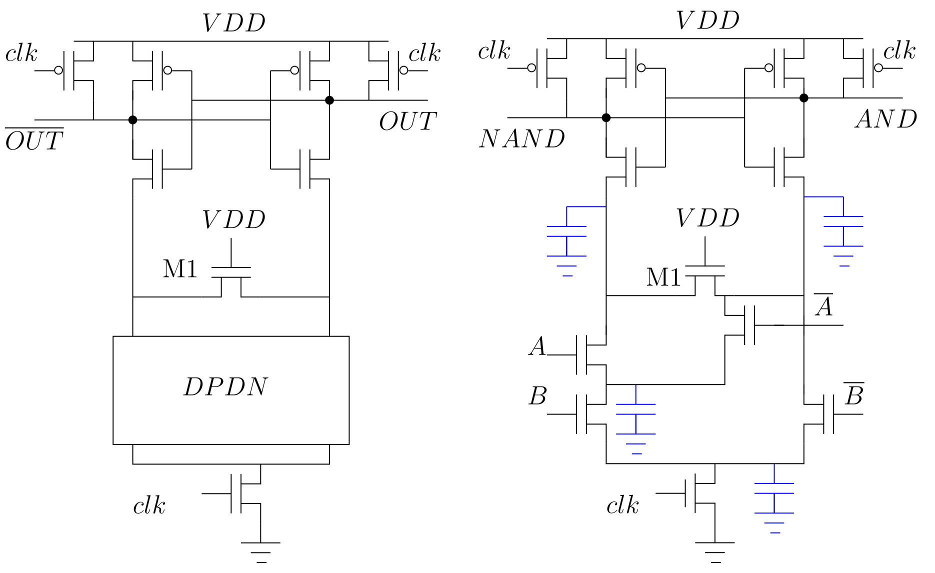


FIGURE 3.9: CMOS inverter output transitions

To overcome this limitation of CMOS gates, for security reasons, dynamic and differential logic can be used. In dynamic logic a precharge signal is used to precharge the output of a gate which is then conditionally discharged, based on the gates inputs. Usually this precharge signal is the Clock signal. Domino logic is a type of dynamic logic [75]. In differential logic a single bit is encoded as two bits, called as the true part and false part. Bit 0 is encoded as 01 and 1 is encoded as 10. The other combinations 00 and 11 are used for other purposes depending on the implementation. In dynamic and differential logic, a normal CMOS gate is replaced by a compound gate that has twice as many inputs and outputs as the CMOS equivalent. The addition inputs and outputs are because of the differential encoding. The aim is to achieve 100% switching activity independent of input combinations. By 100% switching activity, we mean that for any combination of inputs, the same number of $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions occur on the output of a gate. The combination of dynamic and differential logic results in 100% switching factor independent of input combinations.

Sense Amplifier Based Logic Style

Dynamic and differential logic is not a new concept, however its application to prevent side channel attacks was first proposed by Tiri *et al.* [91]; they have used the principles of a sense-amplifier based logic flip-flop, proposed by Nikolic *et al.* [61], and called their logic style sense amplifier based logic style (SABL).



(a) SABL generic gate example (b) SABL AND gate example, with parasitic capacitance

FIGURE 3.10: SABL example

A generic gate structure of the SABL style is shown in Figure 3.10(a) and an AND gate in Figure 3.10(b). The main idea behind SABL is that the same amount of load capacitance should be charged irrespective of the inputs. Assuming that the routing capacitance of differential nets is same, the same amount of intrinsic capacitance is charged for any combinations of input data. This can be illustrated by Figure 3.10(b). For any combination of inputs A,B = (0,0) or (1,1) or (1,0) or (0,1) all the parasitic capacitance are charged. For more detailed explanation about SABL see [91]. Tiri and Verbauwhede [92] have shown, using simulation, that SABL indeed is effective against DPA (although in this example routing capacitance were not considered). Tiri and Verbauwhede [93] have suggested a way to reduce the power consumption of SABL by charge recycling.

Wave Dynamic Differential Logic Style

A standard cell based design is the most common for large designs [75]. As SABL cells need to be custom designed and are not available to a standard cell designer, SABL has high integration cost. That is, to use SABL in a secure design flow, a

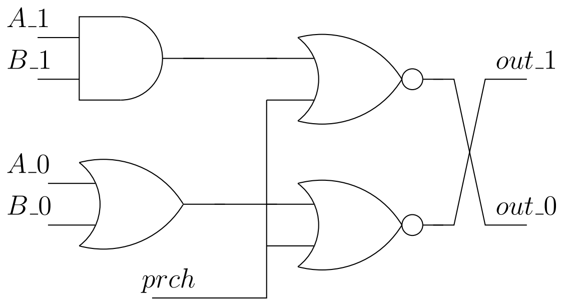


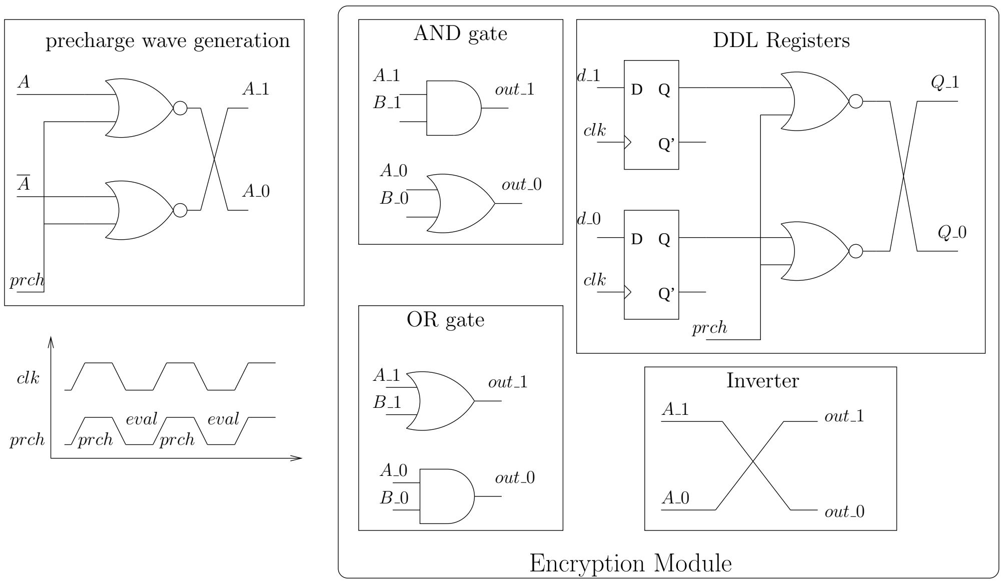
Figure 3.11: Dynamic and differential AND gate

custom library with new SABL cells has to be designed. Wave dynamic differential logic (WDDL) Tiri and Verbauwhede [94] addresses this issue by using the existing CMOS standard cells to make dynamic and differential logic.

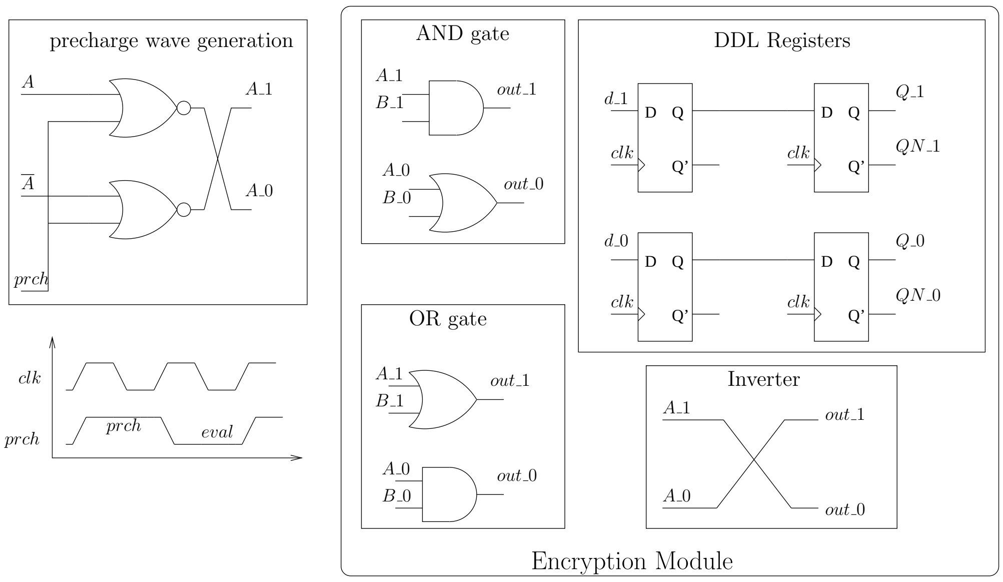
Tiri and Verbauwhede [94] have shown that dynamic and differential logic can be built from normal CMOS gates. A dynamic and differential AND gate (shown in Figure 3.11) can be built from a normal AND gate and OR gate: the true differential signals are connected to the AND gate, while the false differential signals are connected to the OR gate. Finally, the outputs of AND gate and OR gate are AND-ed with the precharge signal: the compound dynamic and differential AND gate will have exactly one $0 \rightarrow 1$ transition for any combination of inputs.

Tiri and Verbauwhede [94] have also shown that instead of generating the precharge signals for every gate, they can be generated only at the inputs to the combinational logic part, thus reducing the size of every gate. This means the precharge signal is generated only at the inputs and propagated through the circuit like a wave, hence the name wave dynamic differential logic. However a restriction for WDDL is that no inverting gates (such as NAND, NOR) can be used. Inversion can be achieved by interchanging the differential signals.

Figure 3.12 shows an overview of WDDL implementation. Tiri and Verbauwhede suggested two different implementations of register cells for WDDL: the first is to use DDL registers, where the output of every register is precharged (shown in



(a) WDDL implementation with normal DDL registers



(b) WDDL implementation with Master Slave DDL registers

FIGURE 3.12: WDDL implementation overview

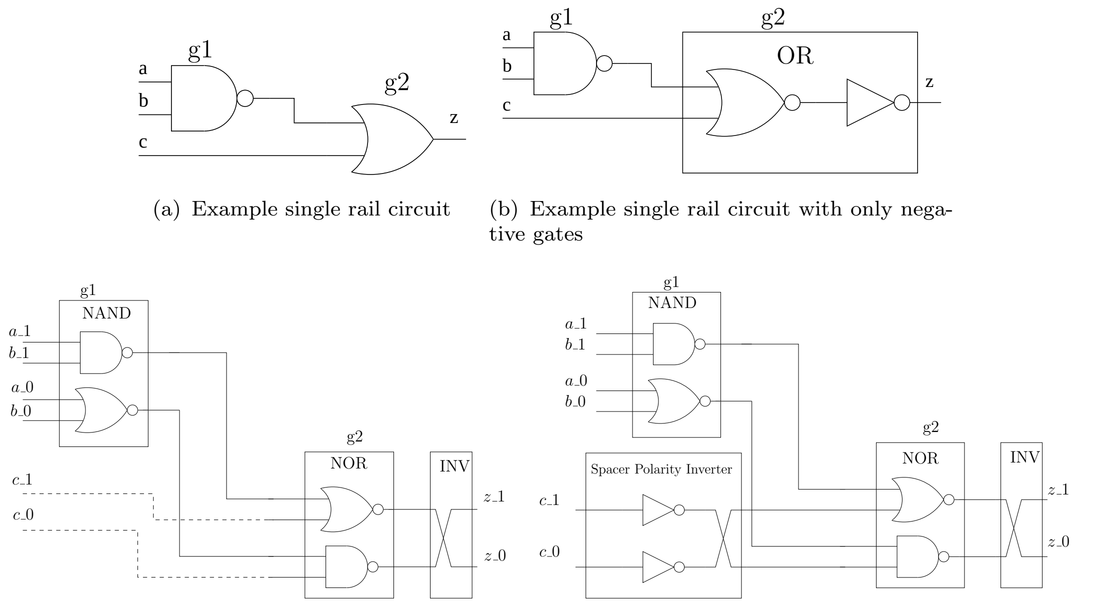
Figure 3.12(a)), and second to use DDL registers in master slave mode (shown in Figure 3.12(b)). Tiri and Verbauwhede suggest using the master slave registers despite the double clock rate to achieve the same throughput, as the compound register has 100% switching factor. Tiri and Verbauwhede [96, 97] have shown how to use WDDL for FPGAs and how WDDL can be adopted for secure design flows.

Alternating Spacer Dual Rail Logic Style

Bystrov *et al.*, suggested using an asynchronous circuit style to counter act SCA[11]. Their proposal was the dual spacer dual rail (DSDR) protocol. Their follow up papers [85, 86] also discuss some implementation details and a tool called Verimap to aid the transformation of standard CMOS netlist into secure dual spacer dual rail circuits. The easiest way to describe this countermeasure is by comparing with the WDDL style. In WDDL only one precharge event is used. That is all the gates are set to logic 0 in the precharge state and then evaluated. In asynchronous jargon, this *all zeros* precharge state is called a spacer which separates the inputs (code words) and the precharge signal. In DSDR another spacer, *all ones* is also used. Hence the name dual spacer. Another difference between WDDL and DSDR is that, WDDL uses only positive logic (positive logic means non-inverting gates like AND, OR). Whereas DSDR tries to optimise the circuit by using negative gates (such as NAND, NOR). This optimisation comes from the fact that in CMOS, positive gates are built from negative gates and an inverter. For example an AND gate is actually a NAND plus an Inverter.

However using negative gates has a problem with precharge wave propagation. In WDDL, all gates are positive, that is when all the inputs were false (bit 0) all the outputs would also be false and then precharge wave would propagate. The problem with DSDR can be understood from the example circuit shown in Figure 3.13(a). The same logic can be implemented using only negative logic, as shown in Figure 3.13(b). A dual rail implementation of the same example, shown in Figure 3.13(c), cannot propagate the spacers properly. More specifically, we want both the rails of a particular signal (z in this case) to be either at logic 0 or 1. That means, inputs to any given gate should be at same logic (1 or 0). If this condition is not satisfied then the precharge wave will not propagate. However, when at the reset state (all zero spacer) the output of gate g1 is 1/1 (both rails).

The inputs to gate g2 are not same, i.e inputs from c are at logic 0 and inputs from gate g1 are at logic 1. This disrupts the wave propagation and affects the security. To solve this issue, Bystrov *et al.* have suggested to use a spacer polarity inverter to eliminate this problem [11] (shown in Figure 3.13(d)).



(a) Example single rail circuit
(b) Example single rail circuit with only negative gates
(c) Example dual rail circuit
(d) Example dual rail circuit with spacer polarity inverters

FIGURE 3.13: DSDR combinational logic implementation overview

Sokolov *et al.* proposed to use these two spacers (all one and all zero) either in a fixed alternating way or a randomly alternating way [85]. Although DSDR was proposed for use in asynchronous designs, it can be used in synchronous circuits with small modifications. Specifically the register schemes used in the WDDL style can also be used with DSDR. DSDR is also called alternating spacer as it alternates between all ones & all zeros spacer.

Murphy and Yakovlev [55] have implemented an AES in 0.35 micron technology using the dual spacer dual rail protocol. Murphy and Yakovlev have mounted differential power attacks on a DSDR AES and a normal standard CMOS AES and reported that DSDR AES has increased the number of required traces by 40 times. Murphy and Yakovlev have also reported DSDR has an area overhead of 1.88 times, reduced throughput by 0.46 times, and increased power consumption by 2.2 times. Note that the increase in area is not twice, as expected for dual rail implementations. The reduced area requirements are due to negative gate optimisation that DSDR uses.

Problem with Dynamic and Differential Logic implementations

At first glance dynamic and differential logic (simply called dual rail from now on) provides a way to protect secure designs from SCA. However routing of differential signals poses a threat to the safety that dual rail circuits can provide. Standard CAD tools used today are designed for single rail circuits, whose aim is not to prevent SCA. Specially the place & route tools poses a big problem for adoption of dual rail circuits. Simply put, if the differential nets of a gate are not balanced, then their parasitic capacitances are different. This means that power consumption of dual rail circuits is still data dependent. Although it would take more traces to attack a dual rail circuit than a single rail circuit, dual rail circuits that do not consider routing capacitance cannot prevent SCA.

To address the routing problem in dual rail circuits, (so far) three proposals [23, 94, 95] were published. The next few paragraphs discusses them.

Divided Wave Dynamic Differential Logic Style

Tiri and Verbauwhede [94] proposed to first route the true part of the dual rail and then copy the layout (with interconnects) and replace the AND gates with OR gates and vice versa. This ensures that the differential signals see the same routing capacitance. The overview of this process can be shown in Figure 3.14. However the problem with this approach is that the designer still needs to balance the differential inputs to the entire cryptosystem and also take care of single rail to dual rail conversion. Another major problem is that if an inverter exists in the original netlist, then this process gets complicated since the inversion should be achieved by

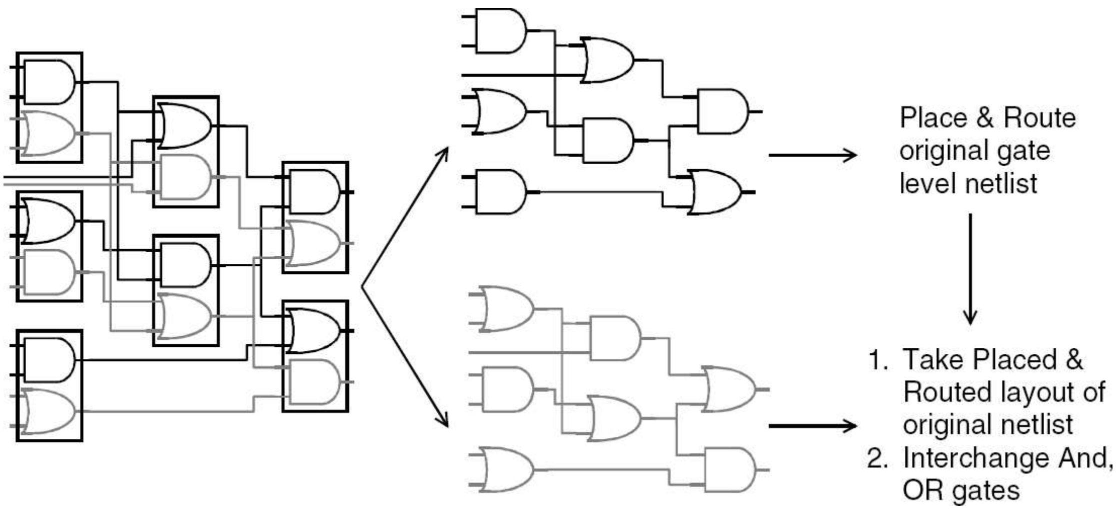

Place & Route original gate level netlist
1. Take Placed & Routed layout of original netlist
2. Interchange And, OR gates

FIGURE 3.14: DWDDL implementation from [94]

swapping the differential wires. Although this approach is a theoretically sound way to prevent power attacks, it may still not be able to prevent EM attacks because the differential netlists are physically separate.

Fat Wire Method

Tiri and Verbauwhede [95] proposed a "fat wire" method to solve the routing problem in dual rail circuits. The idea is to change the minimum routing width from W to W_f, $W_f = (P_n + 2W)/2$, where P_n is the pitch of normal wires. The new fat wire should cover both of the differential signals. Ideally its better to keep the differential signals of each cell close, so that the fat wire need not be extended any more than necessary. After the place & route with fat wire, the resulting design is transformed into final differential design. This process is illustrated in Figure 3.15. After the transformation, as the differential signals are always close to each other, crosstalk effects may exists. To avoid this, one option is to include a third wire in the fat wire, which can be a Vdd or GND line. Another option is to simply increase the space between differential wires. Note that either method results in increased area.

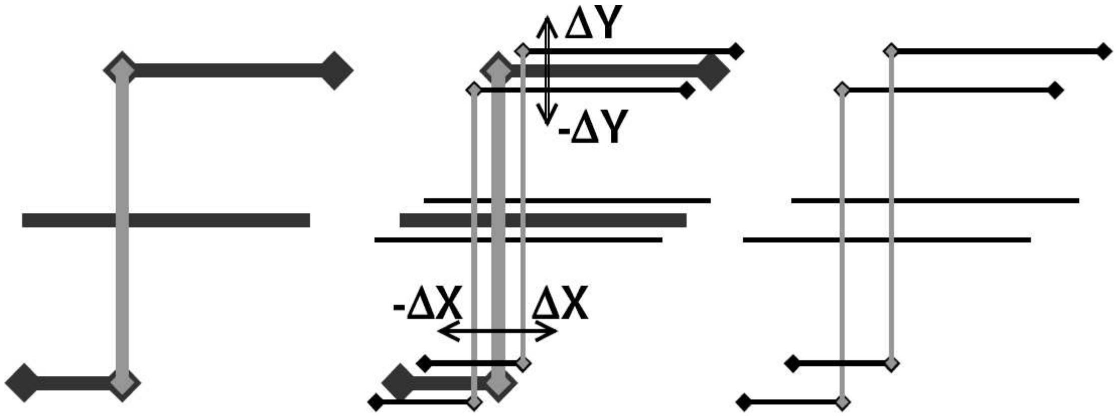


FIGURE 3.15: Fatwire transformation from [95]

Tiri and Verbauwhede [98] have given an overview of secure IC design using WDDL for dual rail and “fat wire” for dual rail routing. In [99] a prototype IC for secure implementations has been developed, by using WDDL and fat wire techniques. It has been reported that this protected AES successfully prevented DPA for 5 key bytes (out of 16) for up to 1,500,000 encryption power traces. Although other 11 key bytes were disclosed for a maximum 1,276,186 encryption power traces. WDDL secure AES overheads are reported to be: Area overhead 3 times, throughput overhead 3.8 times and power consumption overhead 3.7 times.

Backend Duplication Method

Another approach to solve the dual rail routing is proposed by Guilley *et al.* [23], called the backend duplication method. In the backend duplication method, first a single rail netlist is taken as an input to the place & route tool. Then its floor plan is altered to accommodate the dual rail gates and wires. This is done by approximately doubling the length & width of the single rail chip dimensions. Then obstructions are implemented: this step is necessary as it reserves the space for dual rail counterparts. After this second step, the placement and routing is run as in a normal design flow. Finally the duplication is done.

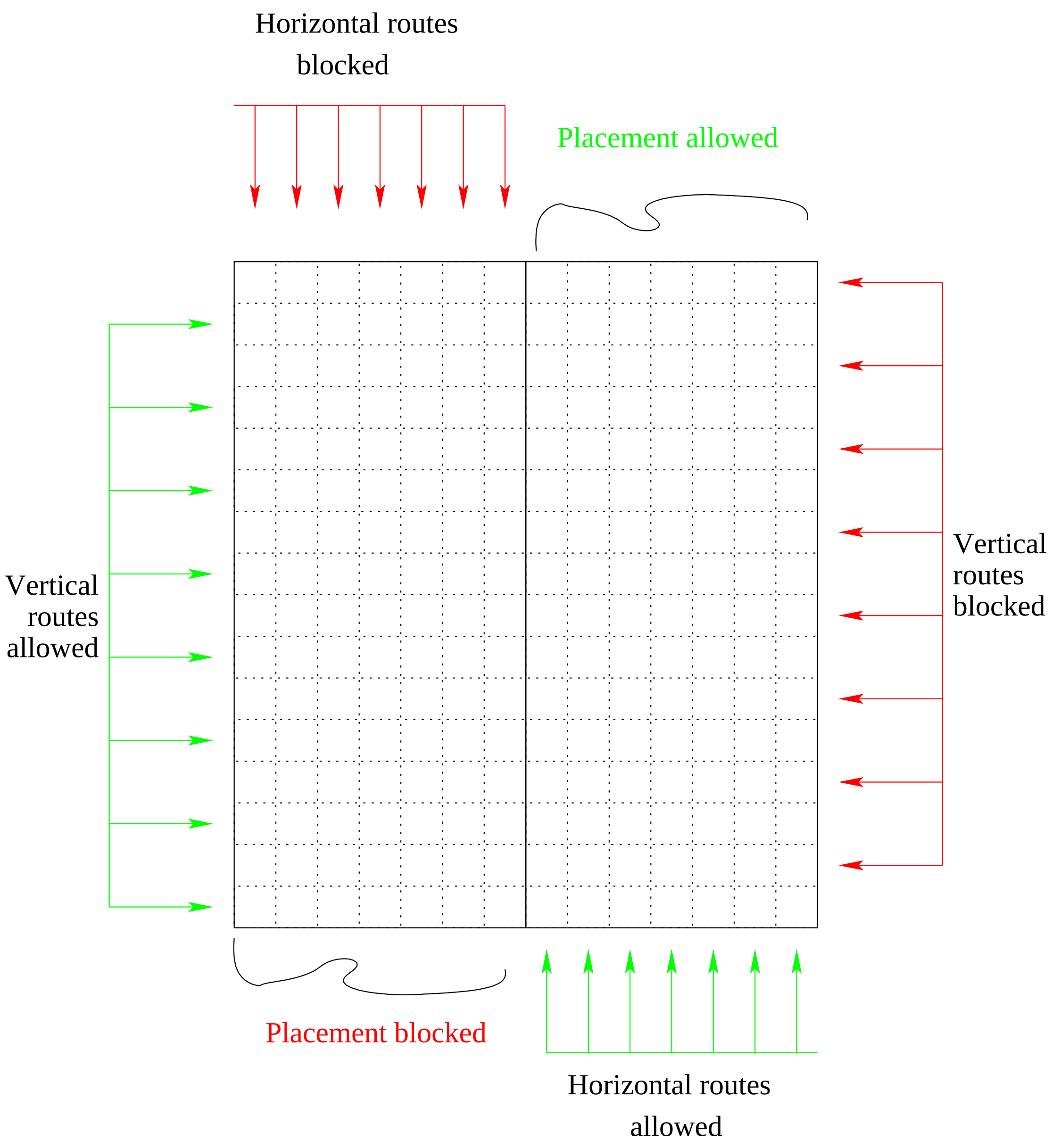


FIGURE 3.16: Secure backend flow: obstruction

Obstruction is needed for two reasons, cell placement and wire routing. For cell placement, every other row is blocked for the dual rail counterparts. For wire routing, every other vertical routing pitch is blocked. The row that is blocked for placement is also blocked for horizontal routing. This obstruction forces the place & route tool to leave enough room for the dual rail counterparts, in a way that routing capacitances are matched.

Duplication is also done for cells and wire routing. For cells, this step is a simple translation (*AND* to *OR*, *NAND* to *NOR*, etc. . .) and a horizontal flip. Note that

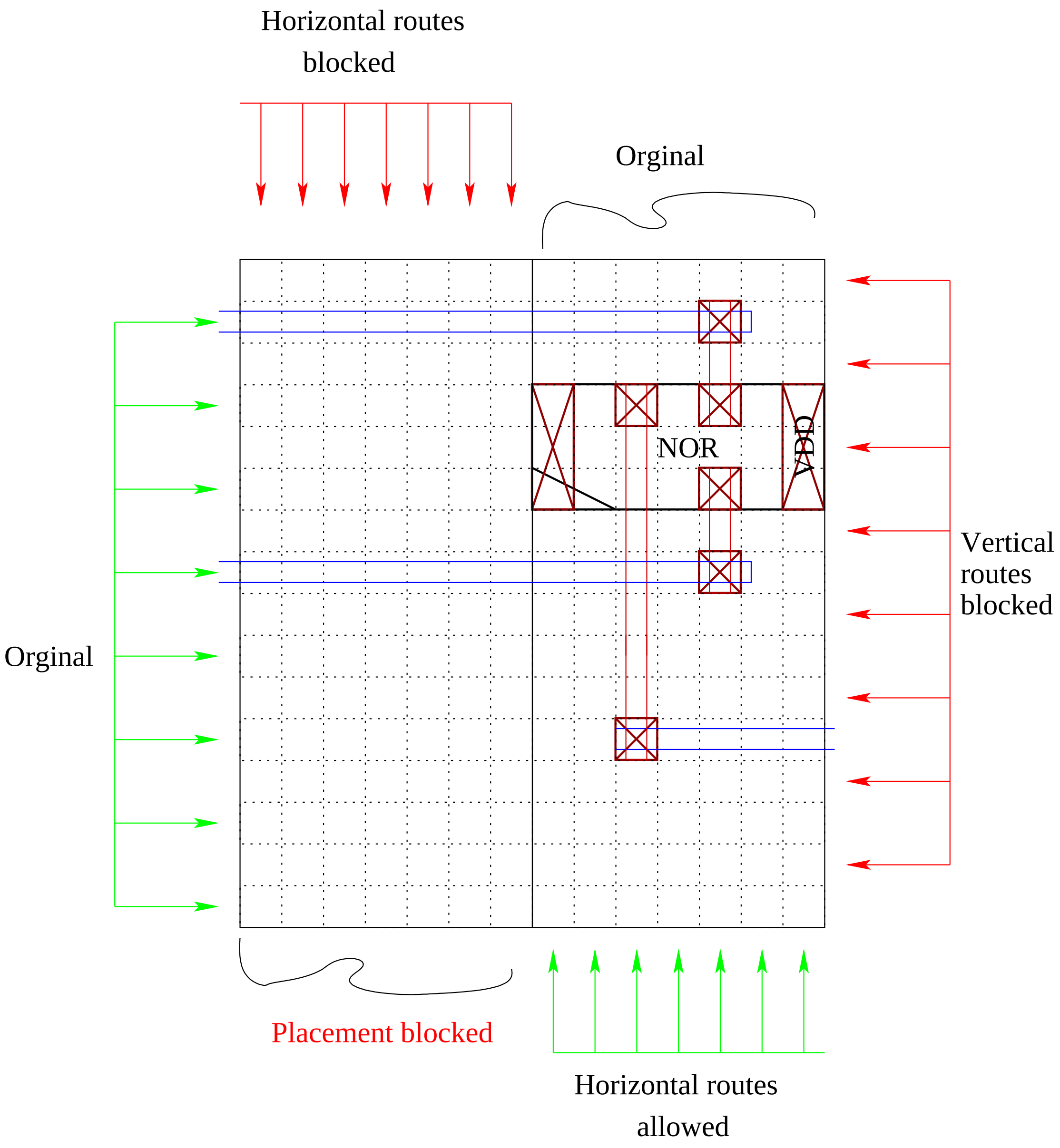


FIGURE 3.17: Secure backend flow: place & route single rail netlist

the wires within the rows also are copied. For vertical wires, this step is a simple shift in position by the routing pitch. The concepts of obstruction and duplication can be understood from the Figures 3.16 3.17 and 3.18. Note that the duplicated cells are shifted by the distance of routing pitch to accommodate the duplicate vertical wires.

However the problem with duplicate place & route is that it does not address the coupling capacitance that exists between the complementary wires, that are

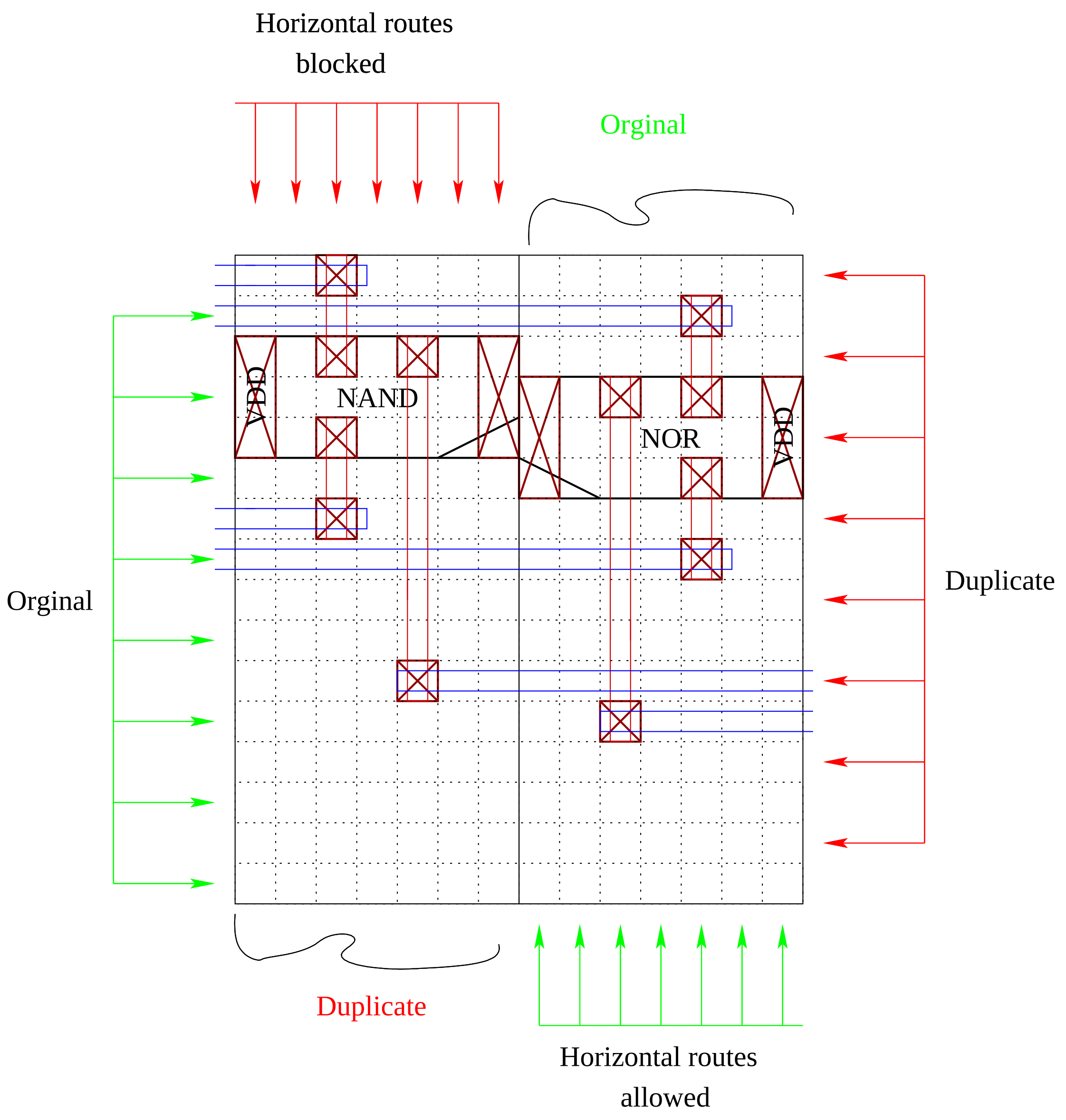


FIGURE 3.18: Secure backend flow: duplication

physically located next to each other. Moreover the coupling capacitance that affects the overall capacitance is not properly distributed.

This of balancing coupling capacitance can be clearly seen from Figure 3.19 and Figure 3.20. Wires W1_1, W2_1, W3_1 are the original wires where as W1_0, W2_0, W3_0 are the duplicated wires. The coupling capacitance as seen by W2_1 is different than that seen by W2_0. Moreover if the capacitance between W2_1 and W2_0 is high, then crosstalk problems exists. One way to solve this problem is to insert a power signal wire between the differential wires or to increase the

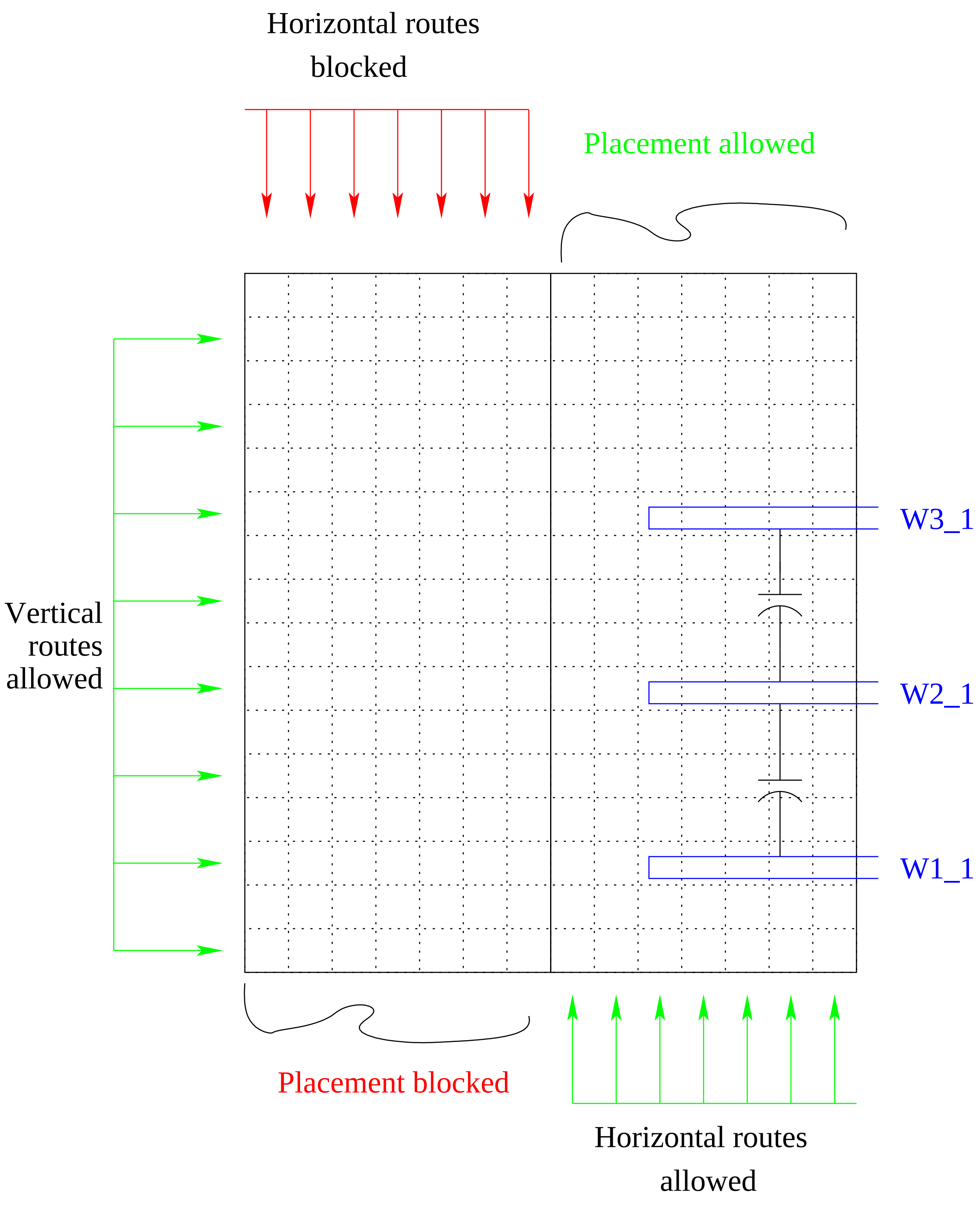


FIGURE 3.19: Secure backend flow: capacitance in single rail circuit

distance between the differential wires (as advised in the fat wire method), but both solutions increase the overall area.

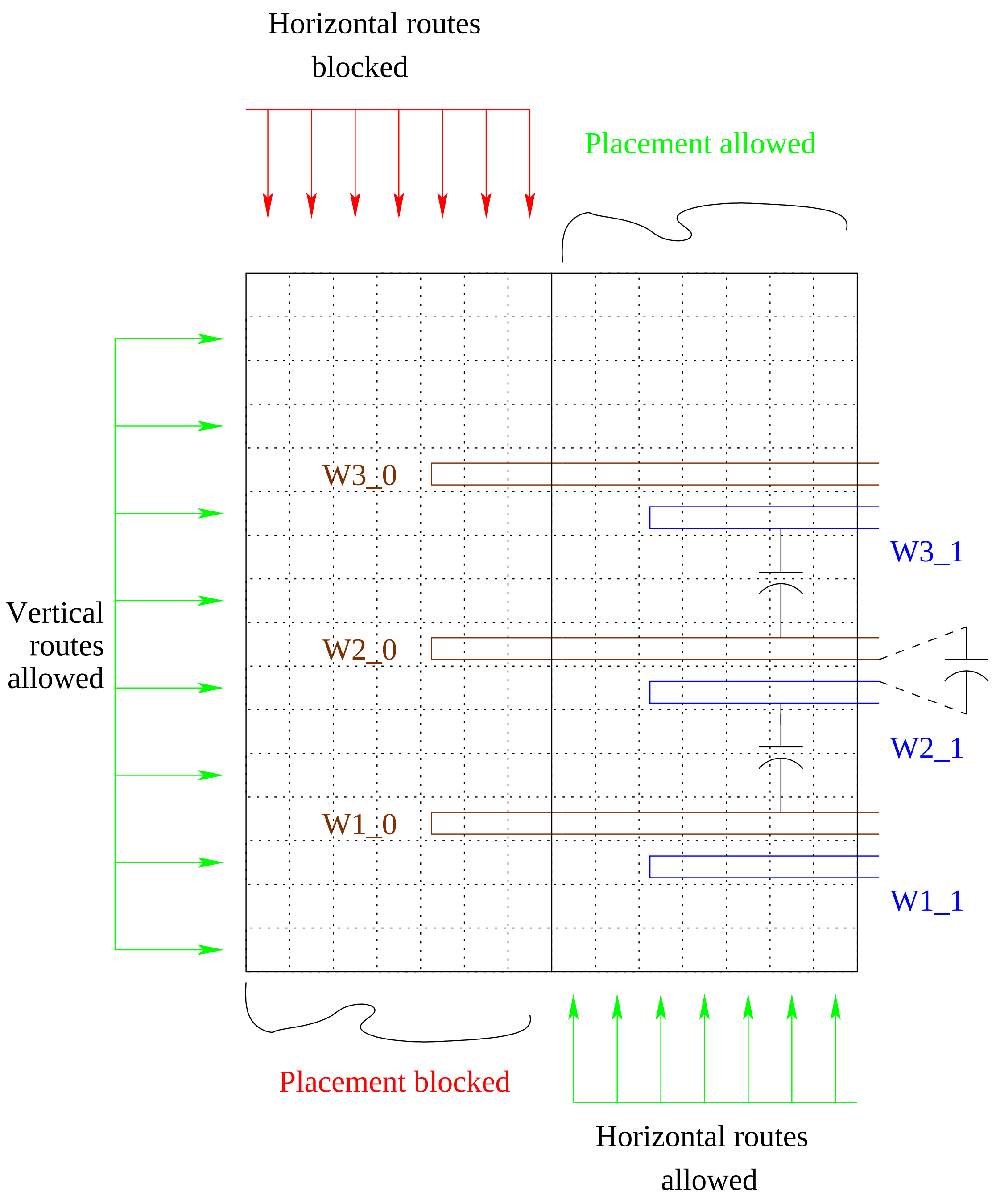


FIGURE 3.20: Secure backend flow: capacitance in dual rail circuit

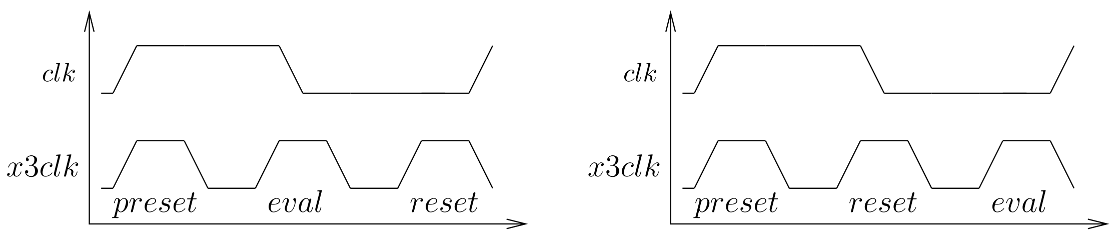


FIGURE 3.21: Three phase schemes

Three Phase Dual Rail Logic Style

More recently, Bucci *et al.* [10] proposed to use three phases instead of two phases (precharge/discharge and evaluate) in dual rail circuits to overcome the routing problem. The idea is to precharge, evaluate and then discharge all the gates, so that the power consumed by per clock cycle is always same. The phasing scheme can be implemented as shown in Figure 3.21.

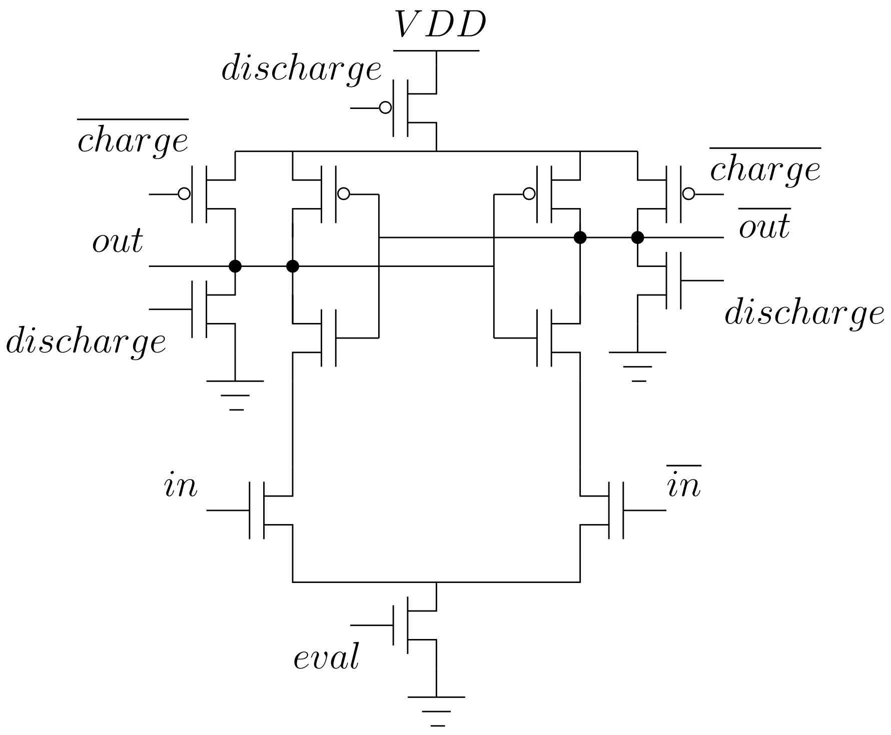


FIGURE 3.22: Three phase dual rail Inverter

A three phase dual rail (TPDR)) inverter is shown in Figure 3.22 and its signal timing shown in Figure 3.23. Although Bucci *et al.* [10] have proposed to use SABL style cells, a WDDL style implementation is also possible. TPDR circuits provide their security by charging and discharging all the gates (and their routing capacitance), thus making the energy consumed per clock cycle constant. However the energy difference between the three phases stills leak information. Bucci *et al.* [10] assume that it is difficult to extract information between phases as all the gates change states at once. This assumption hold true if logic style as shown in

Figure 3.22 is used where evaluation, precharge and discharge signals are same in all the gates.

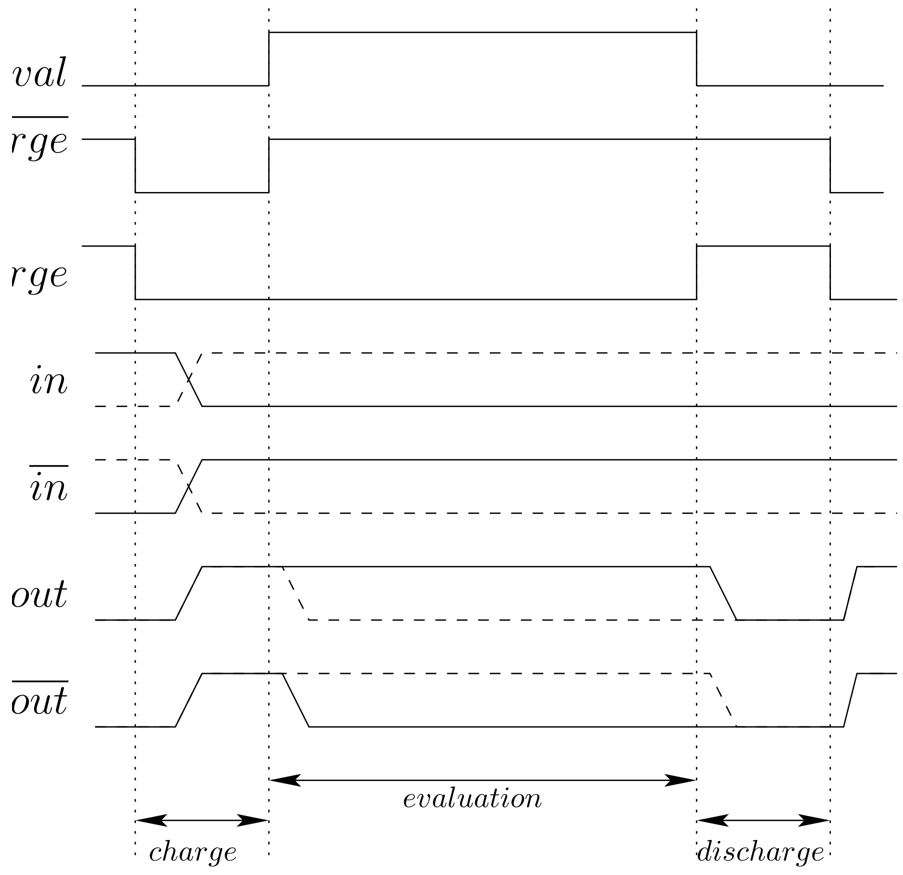
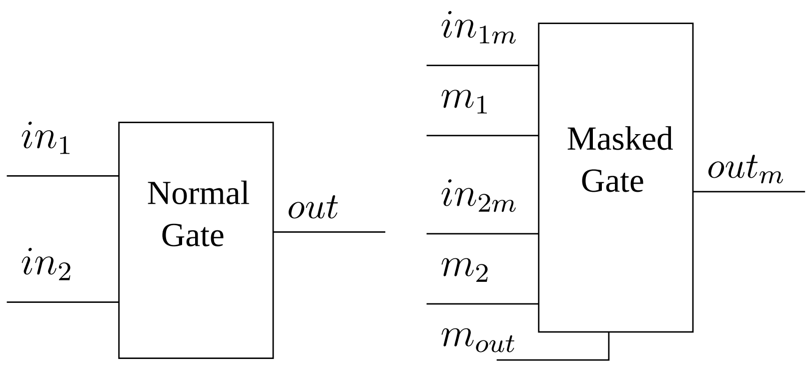


FIGURE 3.23: Three phase dual rail timing diagram

3.5.3.2 Gate Level Masking

The basic idea behind gate level masking can be seen from Figure 3.24. Every input of a normal gate is replaced by a masked input and a mask value and every output is replaced by a masked output. This means $out_m = out \oplus m_{out}$, $in_{1m} = in_1 \oplus m_1$ and $in_{2m} = in_2 \oplus m_2$. Now the output of masked gate is a function of $in_{1m}, m_1, in_{2m}, m_2, m_{out}$. These mask values are randomly generated when required. This means the power consumption of masked gates is not directly dependent on the original inputs, hence it can prevent power attacks. The number of mask bits used need not always be 1. It is shown that more than one mask can be used for a bit [27], but is not practical as the overheads are significantly higher.

Articles [101, 103] present a solution to implement masking for AES non-linear function at the gate level. [27] discusses gate level masking from a theoretical point of view.



(a) Normal gate (b) Masked gate

FIGURE 3.24: Gate level masking

Gate Level Masking Overheads

In general, masking at logic level has high area, power and performance overheads. An example masked AND gate from [103] is shown in Figure 3.25. In this case the area overhead is roughly 8 times. This increase in the number of gates also increases the power consumption and the critical path delay resulting in poor performance.

To use gate Level masking countermeasures, first the normal netlist is translated into a masked gate level netlist (as shown in Figure 3.8). Depending on the availability of resources, a library of masked gates is developed to support the backend flows. Next the translated masked gate level netlist and the library of masked gates are used to finish the design.

The cost of incorporating Masked logic cells into design flows can be very low, if the masked gates are developed from standard logic gates. In this case no extra backend libraries are required. For example this is true if a masking scheme such as

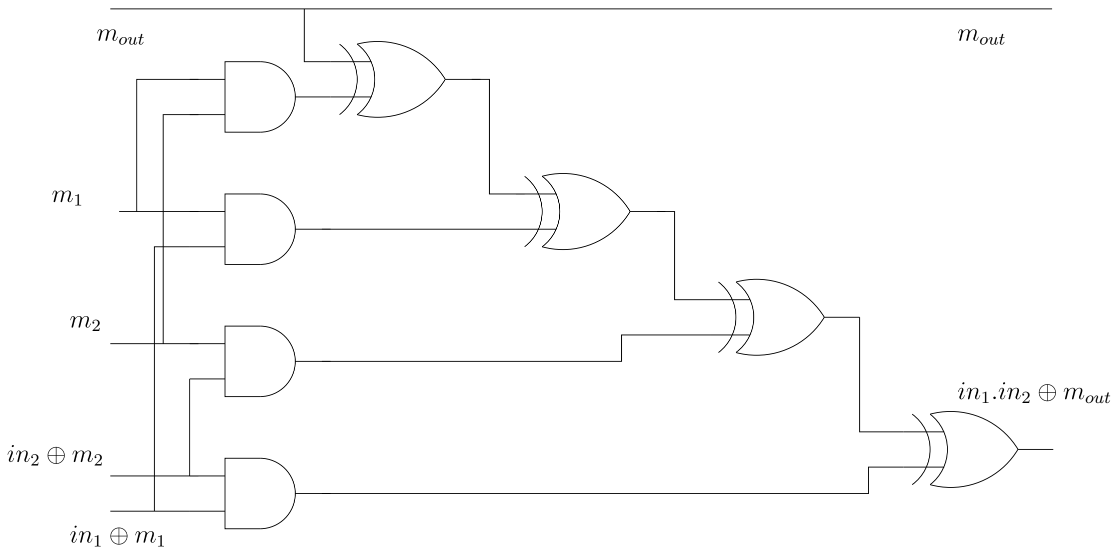


FIGURE 3.25: Masked AND gate from [103]

shown in Figure 3.25 is used. Designers only need to write a program to translate the normal netlist into masked netlist.

Glitches problem in Gate Level Masking

Although gate level masking looks like an interesting option to counteract power attacks, Mangard [39] has shown that masking techniques that do not consider glitches are still susceptible to power attacks. Mangard *et al.* [41] have clearly shown that glitches occur in masked logic gates and that they are susceptible to power attacks. To overcome the glitch problem in gate level masking, Popp and Mangard [69] introduced a new gate level countermeasure based on masking and dual rail precharge, called masked dual rail pre-charge logic style (MDPL). The basic idea of this solution is to avoid glitches in masked logic styles by using dual rail precharge techniques [69]. MDPL gates were realised by using CMOS majority gates.

However the MDPL style also has high overheads. Popp and Mangard [70, Table 1] have shown that MDPL has about 4.76 times area overhead, performs at about 0.6 times the equivalent CMOS designs speed and consumes about 17 times more power. Clearly this is an extremely high overhead and cannot be adopted where constraints are more tight.

3.6 Comparison of DPA Countermeasures

TABLE 3.1: Comparison of some of the DPA countermeasures

countermeasure and design	**area increase**	**speed reduction**	**increase in number of traces**	**notes**
Algorithmic masking				
Masking [1], AES Sbox	4.2	1.7	5.2	susceptible to glitches ([42]
Masking [66], AES Sbox	2.5	2.1	1.2	susceptible to glitches ([42]
Gate-level masking				
MDPL, AES [69]	4.76	0.59	182	key not found for 3,511,000 traces [71]
Constant Power Consumption using Analog Circuits				
On Chip Signal Compression, DES [76]	-	-	-	almost constant power consumption from the device, but needs off-chip capacitance
Randomisation				
DVFS, AES [112]	-	1.2	-	easy to detect synchronisation points from power trace and attack
Random precharging, AES [8]	-	-	-	reduction in correlation by 30%
Dual rail precharge logic				
SABL, DES Sbox [91]	1.79	-	-	needs custom logic cellsto manufacture
WDDL + fat wire, AES [99]	3.1	0.25	119.7	5 subkey bytes not found for 1,500,000 traces
WDDL + backend duplication with shielded differential wires, DES [24]	11.8	0.5	350	key not found for 6,400,000 traces [22]

Since DPA was published, it has received significant amount of attention from researchers, as such, there are numerous proposals to protect a design from DPA attack. Some of these countermeasures are tabulated in Table 3.1. The first column presents the countermeasure type and the design used in evaluating. The second and third columns present area and speed overheads respectively. The fourth column presents the increase in number of traces to find the secret key, when compared to an unprotected implementation of the same design. The final column has some comments related to the countermeasure.

These proposals vary in their approach to prevent DPA and comparing them is not trivial. For applications in smart cards. the most important factor to consider should be security, i.e, the number of traces required to find the secret key. Indeed, the protocol level countermeasures discussed in [29] need this number to design the protocol. The next factor to consider should be area cost and implementation effort, followed by speed and power consumption. Our justification for these statements is as follows; DPA attack is primarily aimed at smart cards. For smart cards, security should be foremost otherwise these cards will be useless in their function. The speed of operation or power consumption is not crucial because the smart cards are only used for few seconds at a time and most smart card readers have dedicated power sources.

Algorithmic masking countermeasures aim to randomise power consumption by using a random mask with the intermediate results. These masking schemes need to take special care of non-linear functions often found in block ciphers. It is shown in [42] that algorithmic masking countermeasures suffer from glitches and that the design netlist with back annotated delays (from layout) can be used to mount DPA.

Gate-level masking also suffer from glitches [41], unless special care is taken to avoid them [69]. MDPL [69] is currently the only known secure gate-level masking scheme, it is shown in [71] that a MDPL AES implementation did not reveal the secret key for 3,511,000 traces. When compared to an unprotected AES implementation this is a 181 times increase in the number of traces. However MDPL has an area overhead of almost 5 times when compared to an unprotected AES.

Randomisation countermeasures such as [8, 112] can only increase the number of required traces by a margin.

Constant Power Consumption countermeasures such as [49, 53, 76] use current sensing circuits to dissipate power such that the device (with countermeasure) power consumption is constant. These solutions have been shown to work in simulations, however one of the requirements of these circuits is an off-chip capacitance. To the best of our knowledge, such countermeasures have not been made in silicon, but if an off-chip capacitance is indeed required, then an attacker can use it to measure the dissipated power to find the cryptographic device actual power consumption.

Dual rail precharge logic style countermeasures such as SABL and WDDL have been shown to be fully secure, however special care needs to be taken when routing the differential nets [95]. SABL is a custom design logic style, so is expensive to manufacture. WDDL on the other hand, can be built from standard cell CMOS library. As mentioned earlier, WDDL designs needs to have fully balanced dual rail nets. In [99], a WDDL AES implementation with special care to routing, using fat wire method, has been designed. When compared to an unprotected implementation of AES on the same IC, WDDL needed 119 times increase in the number of traces and 5 of the subkey bytes could not be found for 1,500,000 traces. The WDDL implementation in [99] has 3.1 times area overhead when compared to an unprotected implementation.

In [24] a WDDL DES with special care to routing, using backend duplication method with shielding of differential nets, has been implemented. In [22], this

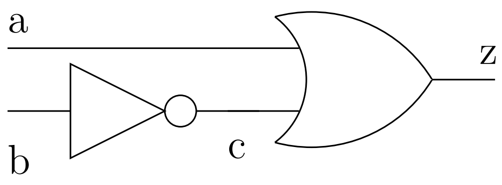


FIGURE 3.26: Circuit to demonstrate the Early propagation effect

designs DPA analysis was not successful even after 6,400,000, an improvement of 350 times when compared to an unprotected DES implementation. However the WDDL implementation in [24] is reported to have an area overhead of 11.8 times when compared to an unprotected implementation.

3.7 Recent Research

Early Propagation Effect

The early propagation effect is an effect that commonly occurs during the operation of a CMOS logic gate and is best described by an example. Consider the circuit shown in Figure 3.26. When the input a is at logic 1 (i.e. true), the output z takes the value of logic 1, whatever the value of input b. As the input a is connected to the OR gate directly, any change in its value will reach the OR gate sooner than b. Say, both the inputs a and b change at the same time. For cases where input a is changed to logic 1, the output z will take the value of logic 1 irrespective of input b. That is, the output z does not have to wait until b arrives. This phenomenon is known as the early propagation effect.

Early propagation effect is a part of the normal behaviour of digital CMOS logic, i.e, it normally goes un-noticed, as it does not alter the functional behaviour of the design. But for secure designs under threat from DPA, it results in leakage of information via power consumption which may be used by an adversary to gain knowledge of secret information. Kulikowski *et al.* [36] have demonstrated that the early propagation effect can be used to attack secure balanced dual rail pre-charge logic styles.

Directional Latch Based Logic

Directional latch based logic (DLBL) is proposed by Kulikowski *et al.* [34] as a dual rail logic style to counter act against dual rail routing imbalance, which also prevents early propagation effect. Signalling between gates (i.e, sending a logic 1 or logic 0) is done by discharging one of the logic rails (unlike other designs where the logic rail is charged). Discharging one rail causes the other to discharge through the directional latch. Directional latch can sense which rail is discharged first and hence the logic value can be determined at the receiving end. This discharging of both the rails leads to a routing capacitance insensitive power consumption [34]. Kulikowski *et al.* [34] have implemented DLBL in schematics only for evaluation and thus could not compare the actual area overheads. However they reported the number of transistors required for a DLBL AND gate to be 29 (compared to 12 for WDDL and 18 for SABL).

Countering early evaluation: an approach towards robust dual-rail precharge logic

Bhasin *et al.* [6] have proposed a dual rail precharge based logic style that is designed to prevent early propagation effect and is specifically targeted at FPGAs. The basis of this logic style, is to use the dual rail spacers [1] to prevent early propagation. Consider a two input AND gate (four inputs in dual rail implementation, composed of two input AND and two input OR); when this gate is being precharged, both the inputs (0/0,0/0 in DRP) inputs are at 0,0 and its outputs

[1]spacers are briefly discussed in Page 63

(0,0 in DRP) is at 0. The idea of this logic style is to prevent the logic gate from evaluating unless both the inputs are evaluated, i.e, both inputs to the dual rail cell should not be 00 or 11. The aim of this proposal is to program a 4 input LUT such that it outputs a spacer value (0,0) unless both its inputs are evaluated. Bhasin *et al.* have implemented an AES in WDDL style and their proposal (called DPL-noEE), without any differential routing balancing techniques in either of the implementations. Using an FPGA based DPA setup, they have demonstrated that WDDL suffers from early evaluation effect, while DPL-noEE does not.

Exploiting dual-output programmable blocks to balance secure dual-rail logics

Sauvage *et al.* [81] have proposed to use Altera FPGA's dual output logic blocks to aid in solving the dual rail routing problem. Some of Altera's Stratix II FPGAs have an adaptive logic block. The feature of this block is, that it can be used a one 6 input LUT or as two 4 input LUTs. Using the adaptive logic block in two 4 input LUT is attractive for dual rail precharge logic, so that the true part and the false part of a gate can be placed closely. Sauvage *et al.* have experimented with various alignments to securely place a WDDL DES design; namely Horizontal place & route and Vertical place & route. Based on FPGA based DPA setup, they found that WDDL implementation without specific efforts of place & route constraints increases the robustness a little; the Horizontal place & route strategy by 6.5; the Vertical place & route strategy by 5.

Balanced Cell-based Dual-rail Logic

Balanced cell-based dual-rail logic (BCDL) [56] is a dual rail precharge logic type countermeasure that uses a global precharge signal to precharge the logic gates.

On top of this global precharge signal, a special gate is used to ensure that all the inputs are precharged; this ensures that early propagation effect eliminated. The advantage of using a global precharge signal is that it reduces the precharge phase, which in WDDL logic styles in once clock cycle. Nassar *et al.* have implemented an AES using BCDL logic style on an Altera Stratix II FPGA, but without any place and route constraints. DPA attacks have been carried out on a the BCDL AES and on an unprotected AES. With the BCDL AES implementation they were unable to find the right key with 150,000 traces providing an increase in the number of traces by 20 times when compared to the unprotected implementation.

Secure Triple Track Logic

Soares *et al.* [84] have presented a dual rail type logic style to counteract against DPA. The basis of this logic style is to have three (instead of two in dual rail precharge logic) rails to represent a bit. The third extra bit is used to synchronise, so that the early propagation is avoided. When implemented on an FPGA this logic style has an area overhead of 5.6 times. This implementation, when put through DPA revealed the correct key in 4000 traces.

Novel Countermeasures and Techniques for Differential Power Analysis

Goodwin [21] has suggested ways modify block ciphers in such a way that DPA will become difficult or impossible. Although Goodwin suggested these modifications for various block ciphers, we only review the ones for AES here. One of these suggestions is to add an initial diffusion function after the first Add Key function, in the form of Mix Columns operation, before starting the round function. The aim of this countermeasure is to increase the number of possible subkeys from 2^8 (if the output of Sub Byte or Add Key is considered) to 2^{32} (for Mix Column). This means that the encryption and decryption implementations of AES has to be modified. Using simulations based on the toggle count of the registers in the design as the power consumption, Goodwin showed that the key could not be found for atleast 4096 number of traces, when compare to 742 for an unprotected one. Another suggestion is to perpetually expand the secret key, i.e, the key scheduler of AES should keep on generating new round key for every new plaintext, in addition to AES's rounds. The idea is to not use the same key for every encryption and there by eliminating any advantage an attacker would have in using the statistical nature of DPA. Decryption implementations of this modified AES need to pre-calculate and store the round keys, this is because the AES decryption requires the keys in reverse order. Goodwin reported that the area overhead of perpetually expanding key version of AES, for an encryption only implementation to be minimal compare to an unprotected one. For an encryption/decryption implementation the area overhead were between 2 to 3 times.

3.8 Summary

In this Chapter, we have discussed different types of DPA countermeasures. The effectiveness of a countermeasure is measured by the increase in number of traces required by DPA to find the *secret key*, when compared to a normal implementation. Every countermeasure has an overhead associated with it (in terms of area, performance or power consumption) and offers an increase in the number of traces required by the DPA.

Dual rail countermeasures (discussed in Section 3.5.3.1) theoretically offer the best DPA resistance, as they eliminate the data dependent side channel leakage. Dual rail countermeasure has been reported to have at-least three times the area overhead compared to a normal design while providing DPA security [24, 99].

However implementing dual rail countermeasures is a challenging task, specifically the routing of differential nets. Solutions to this problem is discussed in Chapter 5 and in Chapter 6.

Gate level masking countermeasures are discussed in Section 3.5.3.2. It is reported that most of the gate level masking solutions suffer from glitches (i.e, leak side channel information when glitches occur) [41]. MDPL combines dual rail precharge logic with gate level masking techniques, to overcome glitch issues in masked gates [69]. MDPL has been reported to have a high area overhead of 4.5 times. In [71] it is shown that MDPL circuits are secure against DPA.

Algorithmic level countermeasures are briefly discussed in Section 3.4. In [40] it is shown that, using back annotated simulations an attack on masked AES took 130,000 traces compared to 25,000 traces for normal AES.

Randomising countermeasures are presented in Section 3.5.2. The aim of these countermeasures is to increase the noise in the side channel leakage. The amount of noise induced can often be traded with area and/or performance. For example, the percentage of random process interrupt can be increased or decreased depending on the amount of noise required [16]. Overheads of randomisation countermeasures depend on the type of randomisation used. In Chapter 7 we discuss hardware level randomisation solutions.

The next chapter presents our DPA flow, based on simulations and on a FPGA based experimental setup.

Chapter 4

Power Side Channel Attacks in Practise

4.1 Introduction

Power side channel attacks, particularly differential power analysis (DPA) attacks, are a serious threat to the safety of a secure devices. This chapter demonstrates how to carry out DPA attacks. Two DPA setups are presented. One uses circuit simulations while the other uses measurements from an FPGA device. The main aim in developing such setups is to prove that DPA is a real and serious threat. These setups can also be used to evaluate various DPA countermeasures. Simulation based DPA setup is a valuable tool that can be employed by designers to evaluate a countermeasures effectiveness before using it in a real device.

The rest of the chapter is organised as follows, Section 4.2 discusses the test circuits used for DPA. Section 4.3 discusses DPA using circuit simulation, while Section 4.4 discusses DPA on a FPGA. In Section 4.5 we discuss the difference between simulation based DPA and a measured one.

4.2 Test Circuits

In this section we discuss the test circuits used for DPA in this thesis. Most of the literature on side channel attacks uses standard cryptographic algorithm implementations or part of their implementation such as DES and AES [30, 62, 91, 92, 95, 100]. Using implementations of standard algorithms for DPA analysis can aid in comparing results with other researchers. We have used AES, AES Sbox and DES Sbox as test circuits in our research.

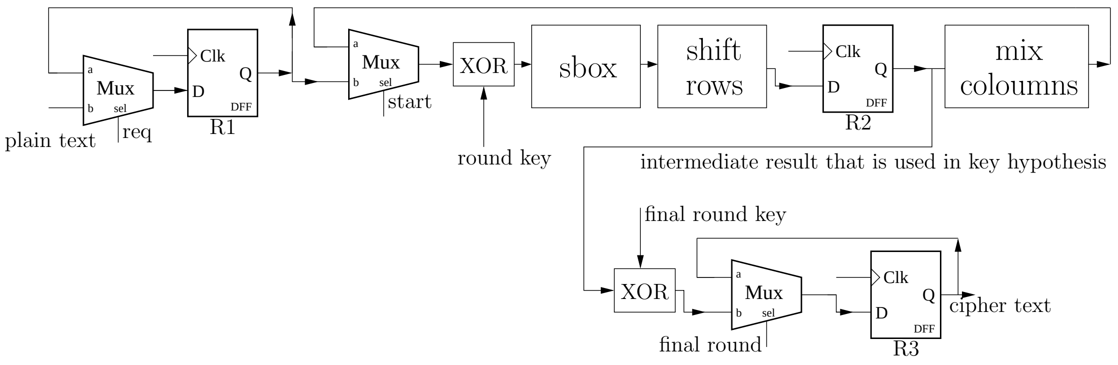


Figure 4.1: AES architecture used in DPA attack

We developed an AES encryption circuit to perform power analysis attacks. The algorithmic description of AES is presented in Section 2.2.2.4. Figure 4.1 shows the architecture of AES used in the DPA attack. Verilog HDL models of AES have been developed. These models were then used with automatic synthesis and place & route tools to implement the circuit. A 0.35μ AMS v370 design kit and 0.13μ ST12 design kit were used for implementation. The same HDL model was used for FPGA implementation as well. For FPGA implementation on a Xilinx VertexE device, Xilinx's design tools were used.

The 8 MSBs (most significant bits) of register R2 (as shown in Figure 4.1) are used for DPA attack. This register contains the intermediate outputs from the *XOR* and *Sbox* operations. The *shift rows* operation does not affect the 8 MSBs of the intermediate result. In the first round of AES, the original secret key is used to generate this intermediate result. As we also know the plain text in the initial round, the only unknown variable will be the secret key. As a result, a DPA attack on this intermediate result will reveal the correct key. Since the secret sub-key used in calculating the attacked intermediate result is 8 bits wide, we have 256 possible sub-keys. The intermediate result is given by Equation 4.1.

$$Intermediate\ result,\ I \ = \ Sbox(key \ \oplus \ plaintext) \tag{4.1}$$

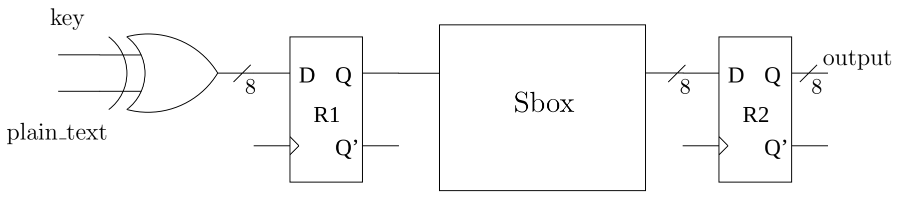


Figure 4.2: AES Sbox circuit used in DPA attack

The AES test circuit shown in Figure 4.1 resulted in 20,000+ logic gates. Because of the size of this circuit, power simulation times were quite high, often taking few days. As DPA on this circuit is focused on the 8 bits intermediate result given by Equation 4.1, other parts of the circuit can be seen as noise sources from the DPA point of view. To reduce simulation time, we also developed part of the AES shown in Figure 4.2. On this circuit, the registered output from the Sbox is used for DPA attack and the hypothesis used for the AES circuit in Figure 4.1 applies to this as well.

In addition to the AES Sbox circuit, we also used a DES Sbox, shown in Figure 4.3, as a test circuit. Since the DES Sbox was smaller than AES Sbox, its simulation time was shorter and as the DES Sbox's subkey length was 6 bits, its analysis time is also reduced. For the DES Sbox circuit in Figure 4.3, *cipher_text* is used for the DPA attack. For DPA on this circuit, inputs L and R are assumed to be known to the attacker. Parts of standard algorithms like AES and DES, similar to the ones described above, have been used for DPA attacks in [91, 92, 95] as well.

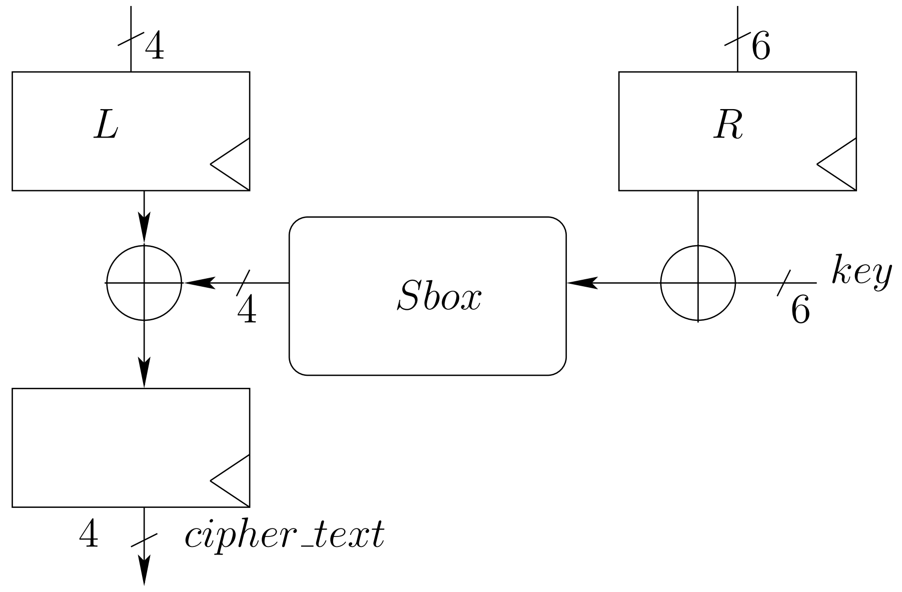


FIGURE 4.3: DES Sbox circuit used in DPA attack

4.3 DPA Setup based on Simulations

In a simulation-based DPA flow, the power consumption of the target design is obtained from simulation instead of measurements from the actual silicon device. Figure 4.4 shows our simulation-based DPA flow. The only part that is different from a hardware based DPA flow is the actual power measurement. In a simulation-based DPA flow, power consumption is obtained using power estimation tools. In a hardware based DPA, power consumption is measured from the target cryptographic device using a digital oscilloscope.

A major advantage of a simulation-based DPA flow over a DPA experiment on hardware, for example an ASIC, is that DPA vulnerability can be estimated early in the design cycle, before committing to actual silicon. For example, a new countermeasure against DPA can be evaluated before fabricating the actual chip. A DPA-based design flow is also presented in [8]. Other advantages of a simulation-based DPA flow include ease of DPA implementation and noise free power measurements from simulation tools. One major disadvantage with simulation based

DPA flow is the time taken by the simulation tools to calculate the power consumption.

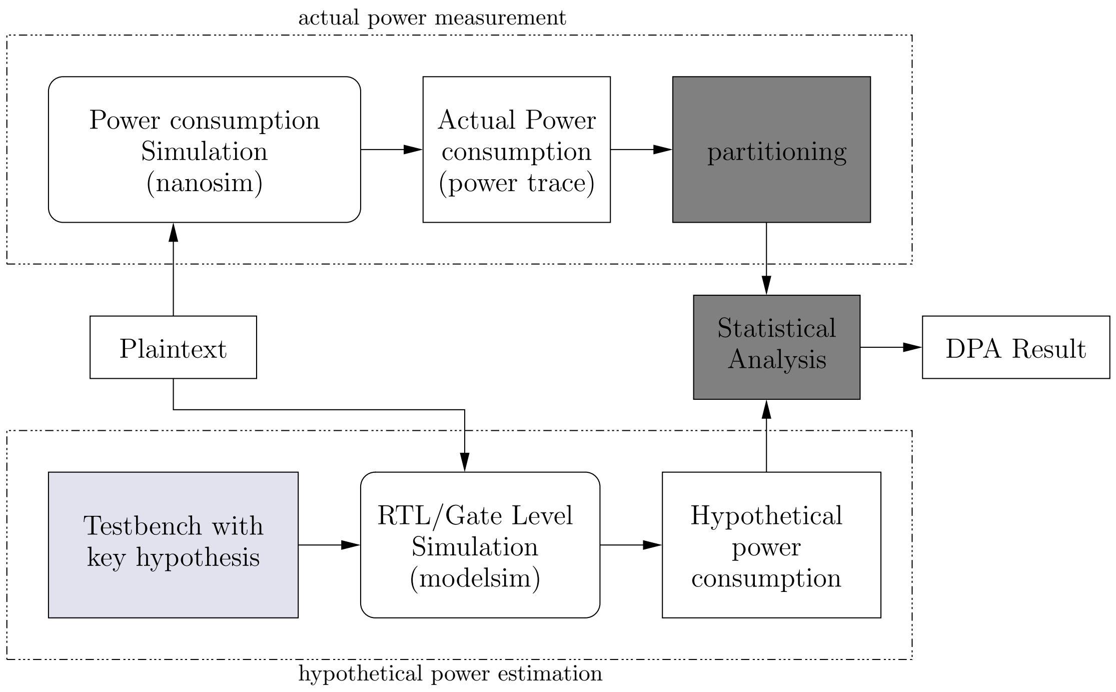

actual power measurement
Power consumption Simulation (nanosim)
Actual Power consumption (power trace)
partitioning
Plaintext
Statistical Analysis
DPA Result
Testbench with key hypothesis
RTL/Gate Level Simulation (modelsim)
Hypothetical power consumption
hypothetical power estimation

FIGURE 4.4: Simulation based DPA flow

The simulation based DPA flow is as follows. The target designs representation (either in transistor netlist or gate level netlist) is taken along with a set of plain text inputs for power simulations. A secret key is used in these simulations and it is kept constant for all plain text inputs. As simulation tools generally save the power consumption result as continuous data, it has to be partitioned with respect to its corresponding plain text input. The same set of plain text inputs are used to generate hypothetical power consumption. Finally statistical analysis is performed on the hypothetical power and simulated power and a decision about the secret key is made.

For key hypothesis generation, we used RTL/gate level simulation using Modelsim [48]. As test bench and RTL models for our target cryptographic device were already developed as part of the design implementation, using them for key hypothesis generation was easy. Partitioning of simulation data and statistical analysis was done using a C++ program developed during this research work.

This program is capable of Pearson Correlation and Difference of Mean statistical analyses.

4.3.1 Power Estimation Tools

There are many different power estimation tools available in the market. Accuracy and speed of these tools depend on the abstraction level they work at. A complete survey of power estimation tools is beyond the scope of this thesis. For further information regarding these tools, see [19]. Availability of a tools is an important factor in deciding its usability. As DPA relies on statistical analysis, the number of encryptions N (Section 2.8.3) and thus the simulation time is a determining factor in the choice of power estimation tools. We had access to three different types of power estimation tools: 1) spice level with best possible accuracy, eg: HSPICE [88] and SPECTRE [13], 2) spice level with best possible performance, eg: Nanosim [89] and Ultrasim [14], and finally 3) gate level, eg: Primepower.

HSPICE and SPECTRE are full spice level tools that provide best possible accuracy. These tools are mainly used in analog circuit design. The computation time required by these tools increases with circuit size. As a result these tools are not preferred.

Primepower is a fast gate level power estimation tool. Efficiency of this tool depends on the availability of properly characterised library. Since the standard cell design kits we used (ST12 and AMS350) did not have a library characterised for Primepower, we did not use it.

Nanosim and Ultrasim are known as fast spice simulators. They claim to trade off accuracy for speed. Nanosim for example, claims to be within 90-95% of accuracy of HSPICE [89]. As both the design kits we used supported Nanosim, it became the simulator of our choice. Nanosim has also been used in [8, 69] to evaluate DPA resistance.

4.3.2 Results from Simulation based DPA

Simulations have been done on a SPICE netlist without routing parasitics using the fast spice simulator Nanosim. To limit the time spent on simulation, encryption rounds (N) of 10,000 have been initially chosen. This simulation of the AES circuit shown in Figure 4.1, which was run on a workstation running RHEL4 on an AMD Opteron 246 with 1Gb memory, took about 25 hours of CPU time.

After obtaining the simulation results, the intermediate result matrix, discussed in Page 35, $I_{1\ldots 2^K,1\ldots N}$ can be constructed by using Equation 4.1. We have used two different hypothetical power models, namely Hamming weight (HW) model and Hamming distance (HD) model (described in Section 2.8.1) to implement DPA.

The Hamming distance hypothesis model is used to predict the $0 \rightarrow 1$ transition on the selected intermediate result. This model is also referred to as the transition count hypothesis. The second key hypothesis is to predict the Hamming weight of the selected intermediate result (for example if 3 out of n bits of an intermediate result were at logic 1 its Hamming weight is 3). These two hypothesis models differ in that the Hamming distance hypothesis relies on exact bit transitions where as the Hamming weight hypothesis is a more generic model, checking which bits are at logic 1.

Results from the Pearson Correlation (Equation 2.8) statistical analysis are discussed here. The Difference of Mean analysis also gave similar results. Statistical analysis using Hamming weight did not reveal the correct key, however the Hamming distance model revealed the correct 8 key bits, 167 in this case. The DPA result plot for two different numbers of encryption rounds, 1000 and 10,000, is

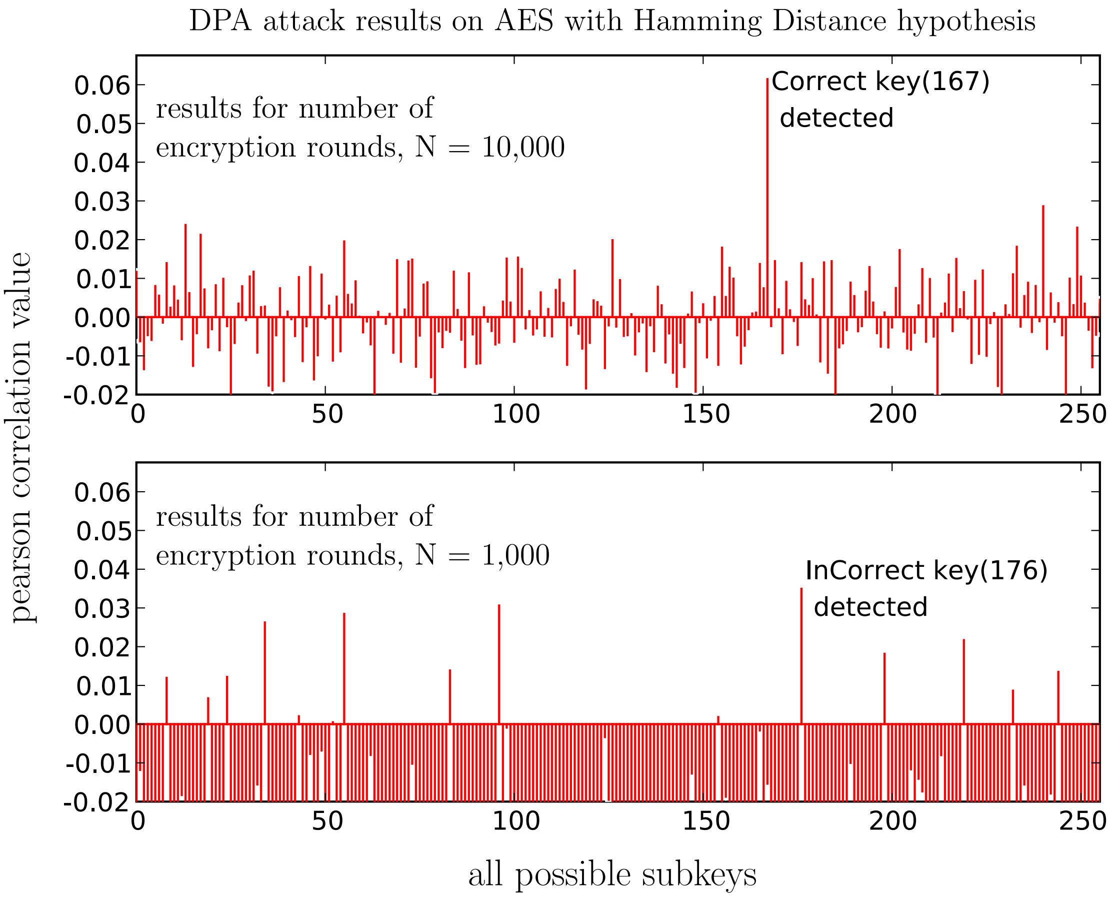


FIGURE 4.5: DPA result for 10,000 and 1000 encryption rounds on AES without any countermeasure, based on simulations

shown in Figure 4.5. The X-axis represents all possible key values (0 to 255) and the Y-axis represents the correlation of a particular key's hypothetical power consumption to the actual power consumption. The key with the highest correlation represents the correct key. It is also important to note that the absolute value of correlation for the correct key is not important, it is the relative value from other possible keys. As can be seen in Figure 4.5, the correct key value was detected for 10,000 encryption rounds but not for 1000 encryption rounds. For our AES test circuit a minimum of 2500 rounds was needed to differentiate the correct key, i.e. for the correlation of correct key to be significantly higher than the correlation of other possible keys.

Because of the circuit complexity and time to simulate the entire AES circuit we also implemented DPA on part of AES, the Sbox shown in Figure 4.2. Figure 4.6

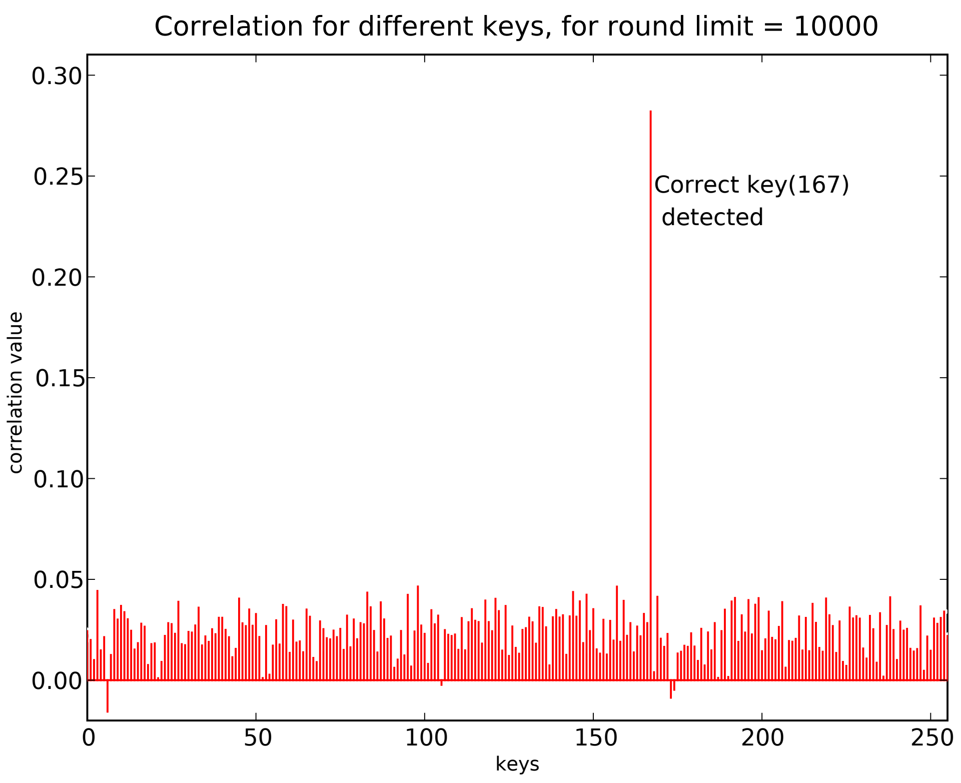


FIGURE 4.6: DPA result for 10000 rounds on AES Sbox without any countermeasure, based on simulations.

shows the DPA result on this circuit. The hypothetical power consumption is same as for the original AES circuit in Figure 4.1. For the Sbox circuit, the attack point is at the register R2. Although this circuit is trivial when compared to the complete AES system, it enables us to see the effectiveness of a countermeasure in a shorter simulation time. Moreover as the there is no other logic operating in parallel in the Sbox circuit, the power consumption observed would have less noise than in the AES circuit (as there is more logic operating at a given time in AES), thus any countermeasure proved against this circuit should work for the entire AES as well. The effect of noise can be clearly seen in Figure 4.6, the correct key detected for Sbox has much higher correlation value when compared to the DPA result on the entire AES in Figure 4.5.

4.4 DPA Setup based on FPGA

As part of this research, we also developed an FPGA based DPA flow. Figure 4.7 shows the general block diagram of our DPA setup on an FPGA. The flow of the experiment is as follows. The PC initiates the experiment, by first setting up the oscilloscope and then the FPGA (cryptographic device). Next the PC sends data to the FPGA for encryption. While the FPGA is performing cryptographic operation, it sends a trigger signal to the oscilloscope. The oscilloscope then records the side channel leakage of the FPGA and sends it to the PC. After the PC receives the side channel leakage data from the oscilloscope, it initiates a new data transfer to the FPGA.

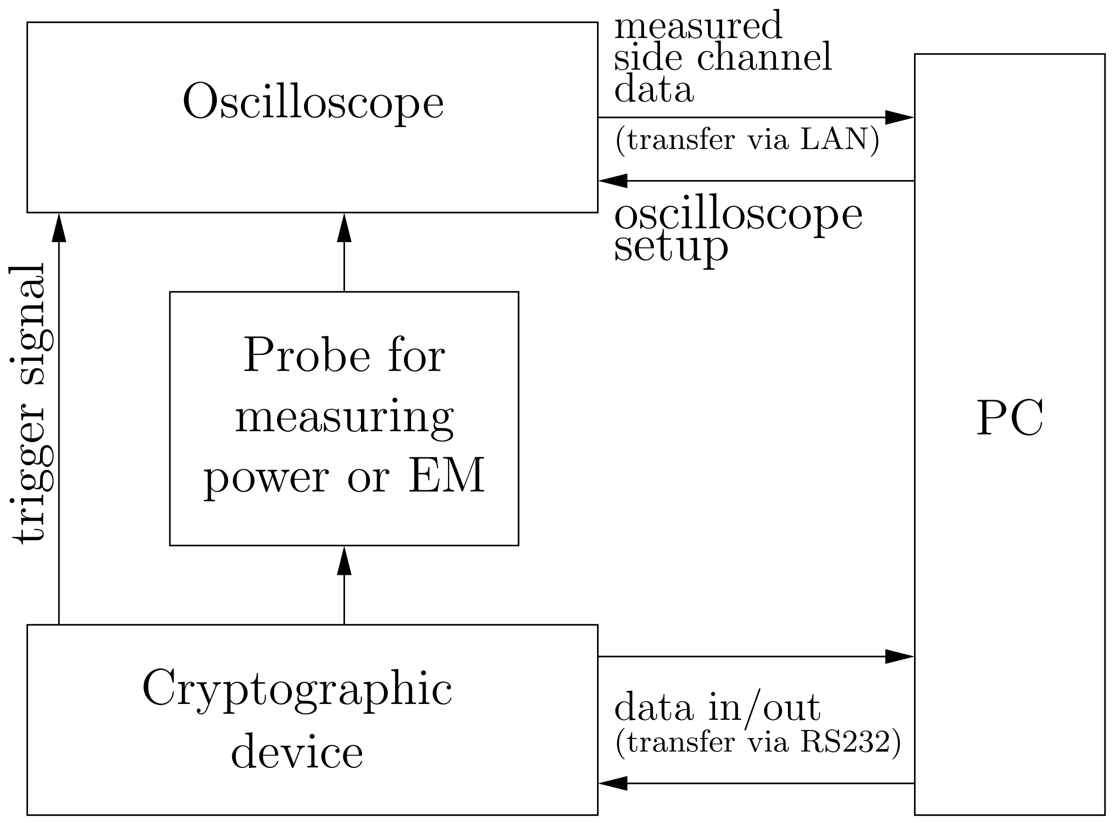


FIGURE 4.7: Block diagram of DPA setup

Oscilloscope

The Oscilloscope is used to measure the instantaneous power consumption of the device under attack, digitise it and transfer it to the PC. At the time of this research, we had access to a few different types of oscilloscopes including the Agilent DSC80204B series oscilloscope. We preferred the Agilent DSC80204B for two reasons. First the availability of an active differential probe with this scope and second the availability of an Ethernet port for data transfer. For all our experiments we used this oscilloscope, which had 2 GHz bandwidth and 40 GSa/s aquisition rate (shown in Figure 4.8) and an active differential probe from Agilent.

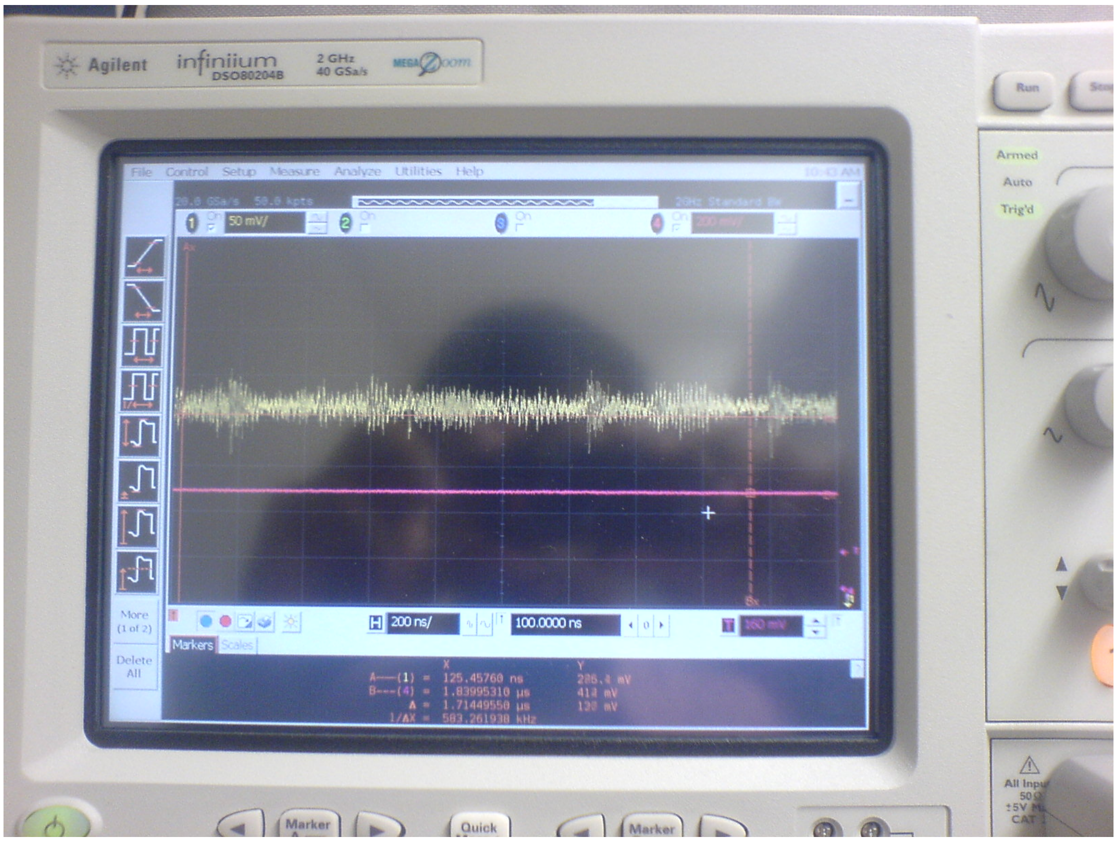


FIGURE 4.8: Picture of the Oscilloscope while measuring power consumption

Data Transfer

Communication between the oscilloscope and PC was done via Ethernet. Communication between FPGA board and PC was done via RS232 for which we developed an RS232 interface for the FPGA. To aid in these experiments, we developed a C++ program that communicates with the FPGA via RS232 and with oscilloscope via Ethernet. This program also has a behavioural model of AES, AES Sbox and DES Sbox so that any data sent back from the FPGA is verified against the expected result.

Target Device

The FPGA board that we had access to is a Xilinx BG560 Prototype board [110]. This board has some peripheral equipment along with a socket for Xilinx Vertex FPGA. The Xilinx Prototype board has three separate power supply terminals, one to the internal core of the FPGA, the other to the IO of the FPGA and a third to other peripherals on board. This enabled us to measure the power consumption of just the FPGA's internal logic elements and not the IO circuitry.

Measuring Power Consumption

There are different ways to measure the power consumption of a device. The most common way is to use a resistor across power supply terminals and to measure the voltage across it. The measured voltage is proportional to the current flowing through the resistor and hence power consumed by the device. One other way is to use a commercially available current probe. These so called current probes have an inductor through which a wire is passed. The amount of current passing through this wire is proportional to the voltage measured across the inductor.

Current probes are better to use at low voltages as they do not have the voltage drop that is found in a resistor based setup.

Since we did not have access to a current probe, we used a resistor across the voltage supply line to measure power consumption. Although our target board had separate power supply lines for core and IO, they all had coupling capacitance to ground and this resulted in noisier measurement. To overcome this, we also experimented with simple coils to measure the device's power consumption via the electro magnetic field. Figure 4.9 shows a photograph of our test setup using one such coil.

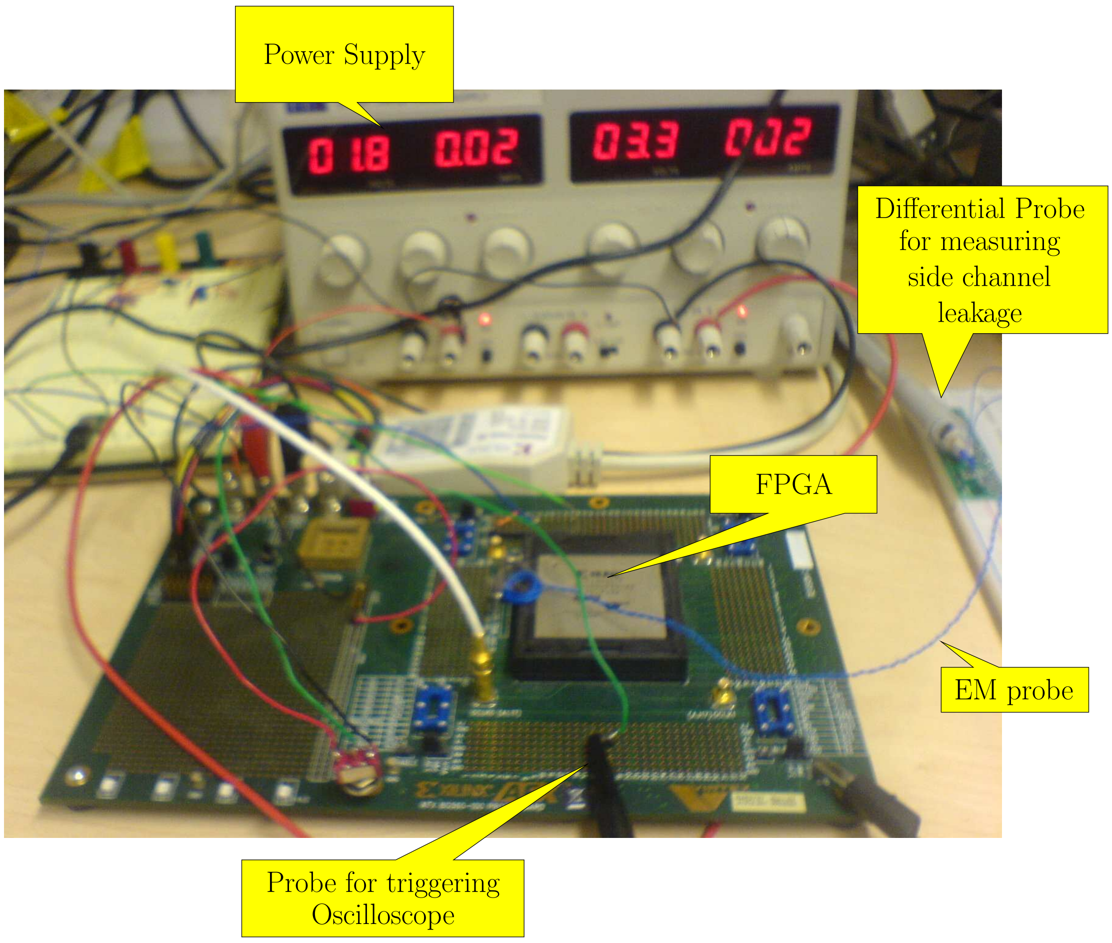

Power Supply
Differential Probe for measuring side channel leakage
FPGA
EM probe
Probe for triggering Oscilloscope

FIGURE 4.9: Picture of the Xilinx FPGA board used in the DPA experiment

In our experiments, both coil and resistor yielded similar results. Resistor based measurements did not yield any results for big circuits, where more current is consumed. The coil we used is sensitive to its position relative to the FPGA.

In all cases we had to try various positions, until we could see a clear current consumption signal on the scope. The coil is also more sensitive to other EM signals, especially from mobile phones. Results from experiments using resistor and coil based measurements are discussed next.

4.4.1 Results from DPA on FPGA

Test circuits used in the FPGA based setup are same as the ones from the simulation based DPA. The only difference is that there is extra RS232 logic to communicate with the PC. In all cases we made sure that only the encryption logic operates when the oscilloscope measures the side channel leakage.

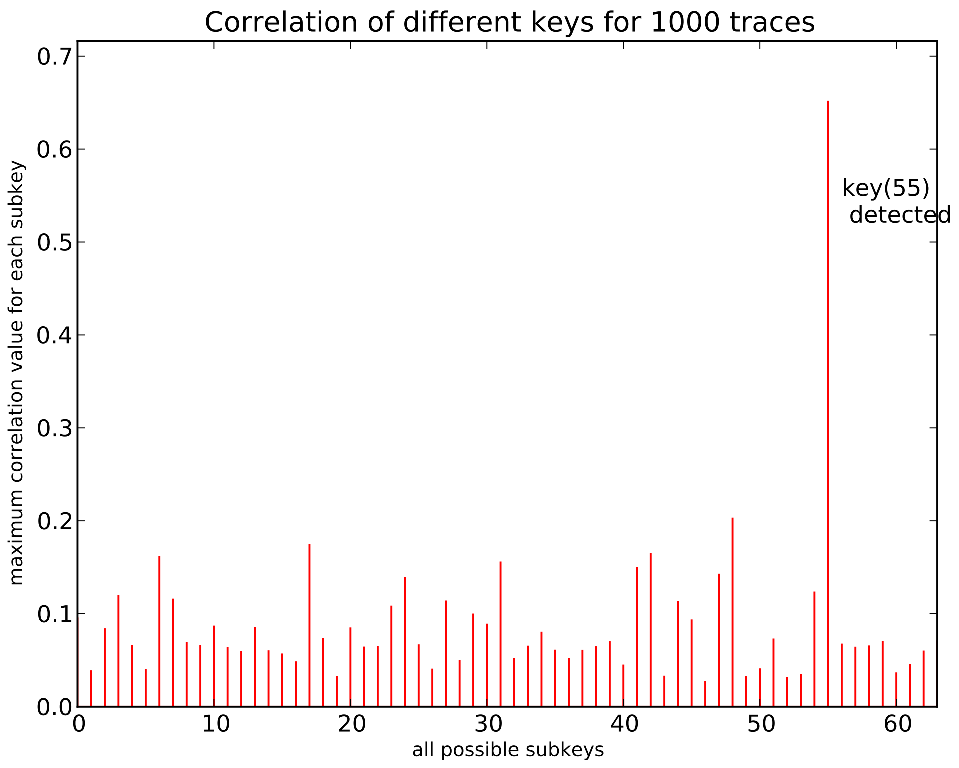


FIGURE 4.10: DPA result for 1000 rounds on FPGA implementation of DES Sbox without any countermeasure.

We first experimented on the AES Sbox circuit shown in Figure 4.2. On this circuit, power side channel leakage measurements from both the resistor and coil did not reveal the correct key. We suspect this is because of noisier power supply connections on our board due to coupling capacitance and because the AES Sbox circuit's power consumption is too small to be detected. We then changed our AES Sbox design on the FPGA in such a way that the outputs from register R2 were driven out to the FPGA IO ports. Our aim in doing this is to increase the power consumption of AES Sbox by driving the extra capacitive load. With the new design, measurements from both resistor and coil revealed the correct key. Our DES Sbox experiments were similar as well, i.e, the outputs were driven out to FPGA IO ports in order to find the correct key. Both AES Sbox and DES Sbox revealed their secret Key within 500 encryption rounds. Figure 4.10 and Figure 4.11 shows the DPA results on DES Sbox and AES Sbox respectively.

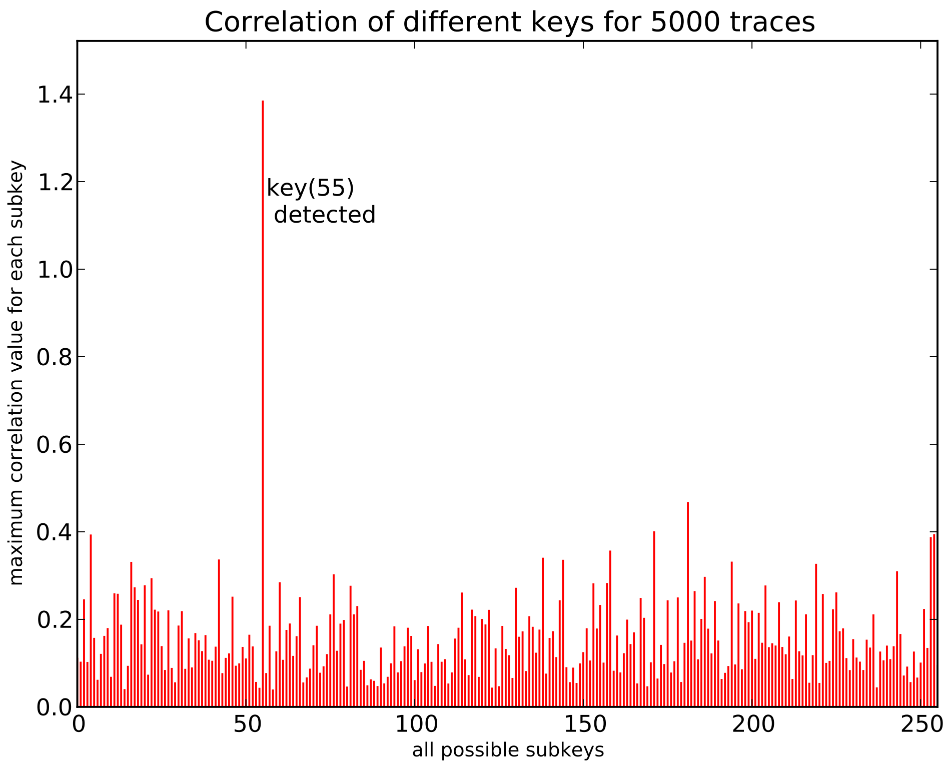


FIGURE 4.11: DPA result for 5000 rounds on FPGA implementation of AES Sbox without any countermeasure.

For our AES circuit, in Figure 4.1, we first measured power consumption via the resistor for 5000 encryption rounds. This number however did not reveal the correct key. Later we used the coil to measure EM signal around the FPGA. Results from the coil based measurements revealed the correct key. We suspect this because of noisier power measurement. Figure 4.12 shows DPA result on AES circuit. Minimum number of traces required for a successful DPA attack on our FPGA board is tabulated in Table 4.1.

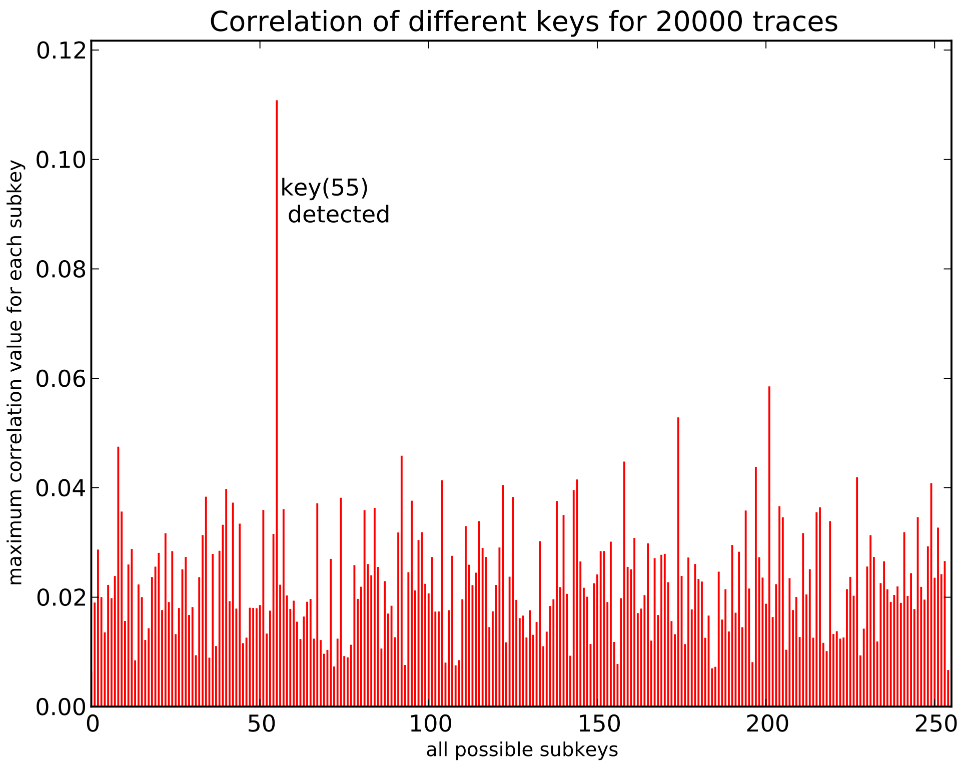


FIGURE 4.12: DPA result for 20000 rounds on FPGA implementation of AES without any countermeasure.

TABLE 4.1: Minimum number of traces required for successful DPA attack of unprotected designs implemented on our FPGA board

design	minimum number of traces
DES Sbox	43
AES Sbox	31
AES	232

4.4.1.1 Discussion about comparing DPA setup

The FPGA based DPA setup discussed above was developed as a part of this research and its main purpose was to evaluate the effectiveness of countermeasures. During the development of this board, we did not try to calibrate our setup by comparing results from other DPA publications. To do that we will have to have exactly same (or atleast similar) components, like oscilloscope, measurement probes, power supply to the FPGA, FPGA device and the PCB board containing FPGA device. Instead, we tried to tune our setup such that we would find the correct secret key in as fewer traces as possible. To measure the effectiveness of a countermeasure, we compare the number of traces required for an implementation with the countermeasure to the one without any countermeasures. This approach of evaluating the countermeasures has also been used in various publications such as these [22, 71, 99, 113].

However there is an effort to standardise an FPGA based DPA setup that various researchers can use for effective comparison of DPA results and is available via [78] ("Side-channel Attack Standard Evaluation Board"). For any one seeking to work in DPA related topic, we strongly recommend using a standard board such as [78] as it 1) reduces the DPA setup development time and 2) facilitates in comparing DPA results with other research work.

4.4.1.2 How to know if the DPA Attack is Successful ?

After a DPA attack how can one know if the attack was successful in finding the key? One option is to use the key found via DPA attack to decrypt a ciphertext which is a result of encrypting a known plaintext using the device under attack. For this to be possible, the attacker needs to be able to control the plaintext and be able to see the ciphertext. Another option is to do a DPA analysis for different number of traces, N (number of encryptions). As N is increased, the correlation

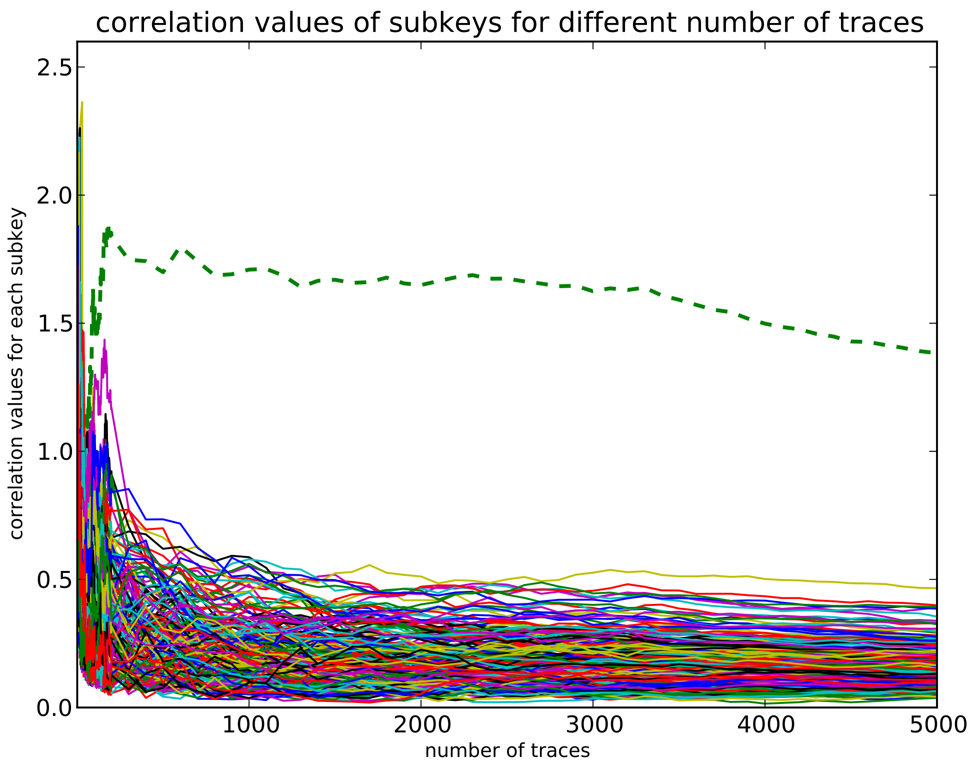


FIGURE 4.13: DPA result for all encryption rounds on FPGA Implementation of AES Sbox without any countermeasure.

of the correct key should higher than the rest. A figure depicting this is shown in Figure 4.13; as the number of encryptions is increased the correct key correlation gets separated from the rest of the other keys.

4.5 Comparison between Simulation based DPA and a Practical one

Simulation based DPA flow, despite its long simulation time, is quite useful for secure circuit designers and researchers, where DPA countermeasures can be evaluated. However there may be concerns about the similarities between simulation based DPA and a DPA on an physical device. For example, simulations provides a noise free, accurate instantaneous power consumption. Whereas, power measurements from physical devices using an oscilloscope will have measurement noise. Moreover, the accuracy of measurement will depend on the sampling rate and bandwidth of the oscilloscope used. Of course the accuracy of simulation depends on power estimation tool and the simulation model used. Here we assume that the simulation is done using a SPICE or Fast SPICE based tool to get reasonably accurate results.

Simulation-based DPA can be considered as an ideal attack environment and hence an ideal environment to evaluate countermeasures, as there is no measurement noise and the simulated power consumption is accurate without any bandwidth or sampling rate issues. If a countermeasure is successful in preventing DPA in simulations then it can be assumed that it will be successful in preventing DPA on the physical device, assuming that the simulation model used is an accurate representation of the physical device. Of course the main drawback of simulation-based DPA is the simulation time.

On the other hand, it may seem that simulation based DPA may be pessimistic in evaluating DPA countermeasures. Since simulated power consumption is free of measurement noise, an oscilloscope based measurement may have some noise in its measurements and this noise in measurement may prevent a DPA attack. However this scenario can be ignored, as DPA uses statistical methods and by simply increasing the number of traces, the affect of noise can be minimised.

Simulated power consumption also provides the maximum available sampling rate and is only limited by the simulation model and the simulation tool used. Although oscilloscopes have limited sampling rate, the current state of the art oscilloscope has about 40GSa/s (Giga Samples per second), while most of the smart cards operate at clock frequencies of a few hundred mega hertz. Moreover for most smart cards, clock frequency can also be controlled by the attacker. Since the designer of a cryptographic device cannot predict which oscilloscope an attacker might use, it is in the interest of the device's security that a worst case scenario be assumed.

A huge difference between a simulation model and a real device is that in the simulation model we did not consider the effects of chip packaging. Chip packaging introduces parasitics and affect the measurement of the power consumption signal [87]. However, in [87], Steinkogler has done a comparative analysis of DPA from simulation of a chip on one hand and measurements on the actual device on the other. Steinkogler found that considering the chip packaging in the simulation did not increase the correlation significantly and that DPA results from transistor-level simulations correlated well against DPA results from measurements on an actual chip.

4.6 Summary

In this chapter, we presented our DPA flows, one based on circuit simulations and other based on an FPGA board. Although our DPA flows are not real world experiments, they are easily adaptable and provide a way to implement DPA and evaluate countermeasures. These flows are used in countermeasures presented in the following chapters.

The FPGA board used during this research has few drawback with regards to DPA. Specifically there were coupling capacitance that made it difficult to get a noise free power consumption. To prevent this a board designed specifically for DPA can be used. For example, Research Centre For Information Security offer an FPGA based evaluation board for DPA called SASEBO [78].

Using our simulation based DPA flow and FPGA based DPA flow on an AES circuit we showed that differential power analysis attacks are a real threat to the security of cryptographic devices. It is also important to note that DPA is simple enough to implement.

Finally differences between simulation and a measurement-based DPA are discussed. Simulation based DPA is a vital tool for evaluating effectiveness of DPA countermeasures.

Chapter 5

Path Switching: A Technique to Tolerate Dual Rail Routing Imbalances

5.1 Introduction

Some of the countermeasures discussed in Chapter 3 rely on randomisation to prevent DPA. The countermeasures presented in [5, 8] (discussed in Section 3.5.2) randomise power consumption of the device thus reducing the data dependent power consumption factor. However, the solution presented in [8] is only shown to increase the number of traces needed for the DPA attack, but could not completely prevent it.

Algorithmic masking [39, 64, 73] (discussed in Section 3.4) is specific to a given implementation and cannot be exploited by automation. Gate level masking [103] (discussed in Section 3.5.3.2) on the other hand can be automated very easily, but proposals so far have high area overheads. It has been shown that both algorithmic and Gate level masking suffer from glitches, which can be exploited to find the secret key [41, 42]. To overcome the glitch problem in gate level masking, dual rail dynamic logic has been used together with masking in [69], known as masked dual rail precharge logic (MDPL). However MDPL has high area overhead, a 4.5 times increase [70].

Dynamic and differential logic (also referred to as dual rail precharge - DRP) [86, 92, 94] (discussed in Section 3.5.3.1) has been proposed to prevent DPA. The idea is to consume the same amount of power for any combinations of inputs. This is achieved by using differential logic (two signals instead of one) and by precharging both the differential nets in every clock cycle. In DRP circuits, for every logic gate, a complementary gate exists, usually referred to as *false* logic (or *false* part). DRP circuits have been proved to prevent DPA provided the routing of differential nets is balanced [99].

Secure dual rail designs suffer from one major problem: balancing the routing capacitance of differential nets[95]. To address the routing problem so far four proposals have been put forward: DWDDL [94], fat wire [95], backend duplication [23] and three phase dual rail [10]. Three of the four proposals [23, 94, 95] impose some constraints on backend implementation flows and try to solve the problem by eliminating the difference of routing capacitance of differential nets. Three phase dual rail [10] tries to avoid the routing problem by introducing a third phase, which is an additional overhead.

In this chapter we present a simple yet effective solution, called path switching, to improve dual rail circuits tolerance to routing imbalances. Our solution assumes that the difference in capacitance of differential nets exists. Instead of eliminating this routing difference, we use this difference to randomly switch the path taken by the differential net, thereby improving DPA resistance.

The rest of the chapter is organised as follows. In Section 5.2 we summarise dual rail precharge logic styles. Section 5.3 briefly introduces our test circuit, hypothesis models and simulation tools used. In Section 5.4 we show the results from successful DPA attacks on dual rail circuits. In Section 5.5 we propose our countermeasure to improve the security of dual rail circuits. In Section 5.6 we discuss path switching implementation and extensions to logic gates. Finally in Section 5.9 we conclude this chapter and discuss some limitations of path switching.

5.2 Dual Rail Precharge Circuits

Dual rail precharge circuits were discussed in detail in Section 3.5.3.1. SABL [92] is a dynamic and differential logic style specially built to consume constant power for any combination of inputs. As SABL cells are not available in the standard library, users of SABL need to develop a SABL library which adds to the total system costs. Wave dynamic and differential logic (WDDL) [94] applies the same dynamic and differential logic style techniques to standard CMOS cells, avoiding the expensive development of SABL library. WDDL has an area overhead of 2-fold for combinational logic and 2-fold or 4-fold on flip-flops ([94]).

Dual spacer dual rail (DSDR) [86] is similar to WDDL except for the following. In WDDL only one precharge[1] event is used. That is all the gates are set to logic 0 in the precharge state and then evaluated. In asynchronous jargon, this 'all zeros' precharge state is called a spacer which separates the inputs (code words) from the precharge signal. In DSDR another spacer, 'all ones' is also used. Hence the name dual spacer. Another difference between WDDL and DSDR is that WDDL uses only positive logic (positive logic means non-inverting gates like AND, OR). Whereas DSDR tries to optimise the circuit by using inverting gates (such as NAND, NOR). This optimisation comes from the fact that in CMOS, positive gates are built from inverting gates and an inverter. For example an AND gate

[1]By *precharge state* we mean precharge or pre-discharge

is actually a NAND plus an INVERTER. However DSDR, with inverting gates, might suffer from glitches and may compromise the secret key (similar to the glitch problem in masking [41, 42]), this has not yet been evaluated. DSDR has been proposed in both Synchronous and Asynchronous systems.

The single rail to dual rail converters, flip-flops and precharge wave generators used in DSDR [86] were presented from an asynchronous design point. To maintain compatibility with WDDL style designs, we used the single rail to dual rail converter and precharge generation as shown in Figure 5.1, which are compatible with both WDDL and DSDR. The precharge wave generation circuit can be build with 2 D-type flip-flops and 1 T-type flip-flop. An additional AND gate can be used to control the precharge value changes (i.e. to randomly control the dual spacer).

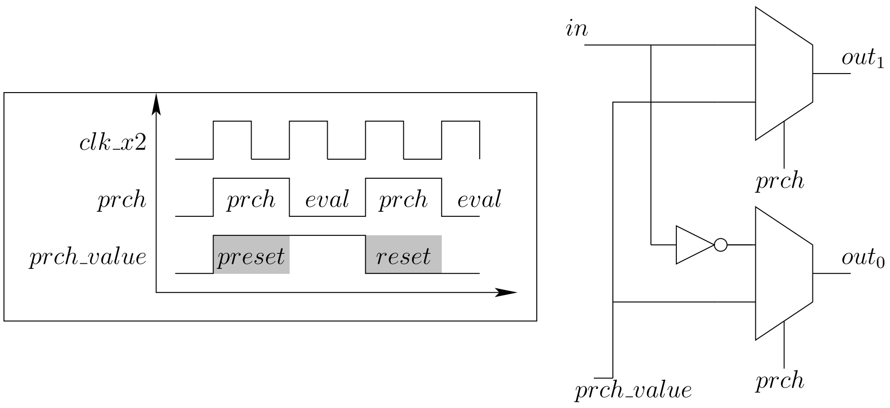


FIGURE 5.1: Precharge wave generation and single rail to dual rail converter for dual spacer protocol

5.3 Test Circuit

Our DPA test setup was presented in detail in Section 4. We used a DES Sbox, shown in Figure 5.2 as a test circuit for experiments related to this chapter. First we implemented a DPA attack on a normal single rail circuit, to serve as a comparison for the effectiveness of dual rail countermeasures. We chose a difference of mean DPA attack [31], which was discussed in Section 2.8.3. We developed two key hypothesis models, namely Hamming weight model and Hamming distance model, to attack the test circuit. These two hypothesis models differ in that the Hamming distance hypothesis relies on exact bit transitions whereas the Hamming weight hypothesis is a more generic model, checking which bits are at logic 1. If the hypothesis value was more than 2, then the corresponding power trace is added to set1 else to set0. To simplify our analysis, the only unknown term in our setup was 'key'. As we show later the results of DPA attack for both these hypothesis models differ.

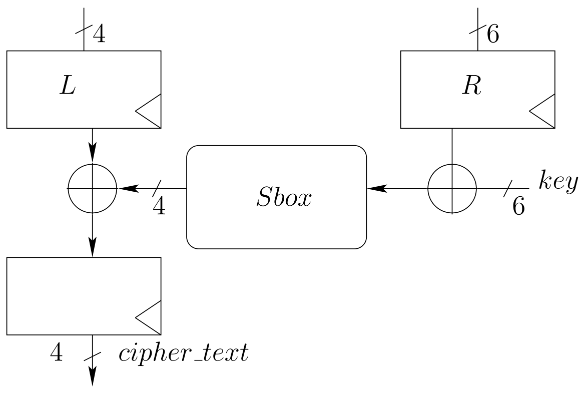


Figure 5.2: Test circuit: DES Sbox

Our test circuit used in this chapter was implemented in AMS .35μ technology. Since our AMS V370 design kit did not have positive cells such as AND and OR in the standard library, which are necessary to implement WDDL circuits, we used a NAND-INVERTER pair to implement an AND gate (similarly NOR-INVERTER for OR gate) to realise a WDDL style DES Sbox. As the WDDL style only uses positive gates, we call our WDDL style implementation *positive logic DES sbox8.* We also realised a DSDR style DES Sbox, with the converters shown in Figure 5.1 and WDDL style master slave flip-flops. As DSDR with negative gates has a wave propagation problem [86], we developed a program which inserts spacer inverters (see [86] for more details) at appropriate places without changing the functionality of the circuit. As the DSDR style proposes to only use negative gates, our DSDR Sbox will be called *negative logic DES sbox8.* By which we imply a circuit with

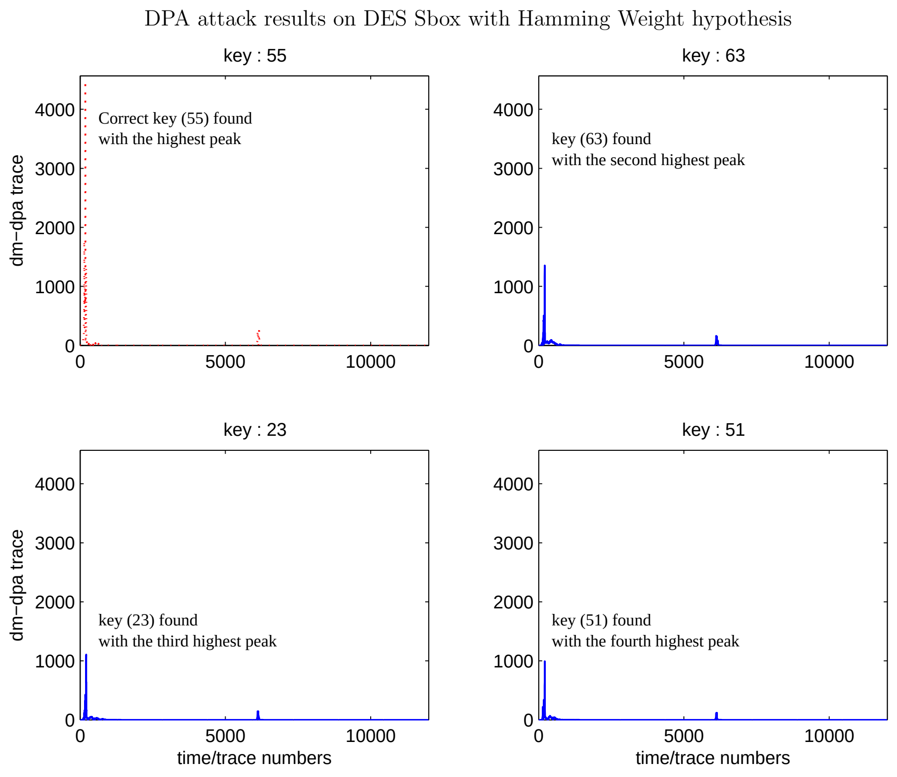


FIGURE 5.3: DPA result on single rail DES Sbox, based on simulations

negative gates only without using the dual spacer protocol. When a dual spacer is used for a particular scenario we will specify it. For all our simulations we used 55 as the secret key.

DPA on the single rail circuit was successful with just under 100 traces for both our hypothesis models. It took 60 traces with the transition count hypothesis model and 90 traces with the Hamming weight hypothesis model. From this we can say that the transition count hypothesis is more effective for this DPA attack. The DPA result on normal single rail circuit for the Hamming weight hypothesis model is shown in Figure 5.3. Figure 5.3 has four subplots which show the DPA result trace, each belonging to a different key. Four different keys results are presented for comparison. The top-left subplot belongs to the subkey with the highest peak in the results trace, which according to the DPA is the estimated-correct key. The top-right and bottom-left subplots belong to the subkeys with the second highest and third highest peaks in the results trace, respectively. If the correct key used

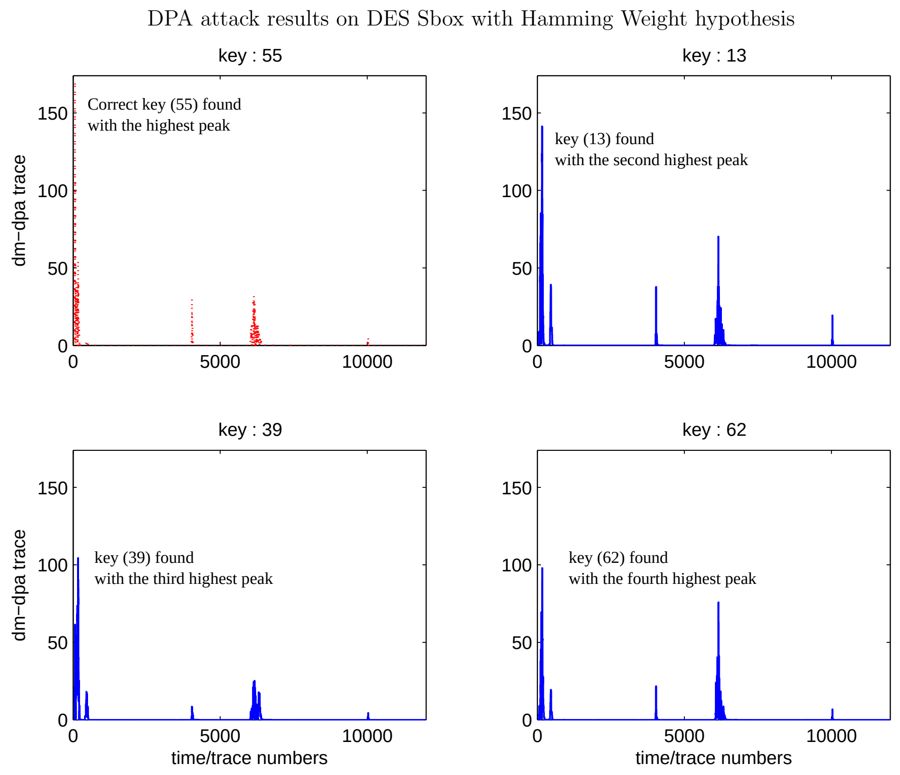


Figure 5.4: DPA result on negative logic DES Sbox without routing capacitance for 10000 traces

for simulation (55) is not in these three subplots, it is then plotted in the bottom-right subplot for comparison purposes. If the correct key (55) is in one of the three subplots then the key with the fourth highest peak is plotted in the bottom-right subplot. For the rest of this chapter, all the DPA results shown are of the same format. We choose this way of presenting DPA results as the DPA result trace's peak position can be easily visualised. As we see later, the result trace's peak position changes, especially for dual rail circuits with two phases.

5.4 DPA on Dual Rail Circuits

To check the effectiveness of DRP logic styles we first mounted DPA on a netlist without routing capacitance, since a netlist without considering routing capacitance can be considered as a DRP netlist with balanced routing capacitance for the differential nets. We used the same key hypothesis as used for the single

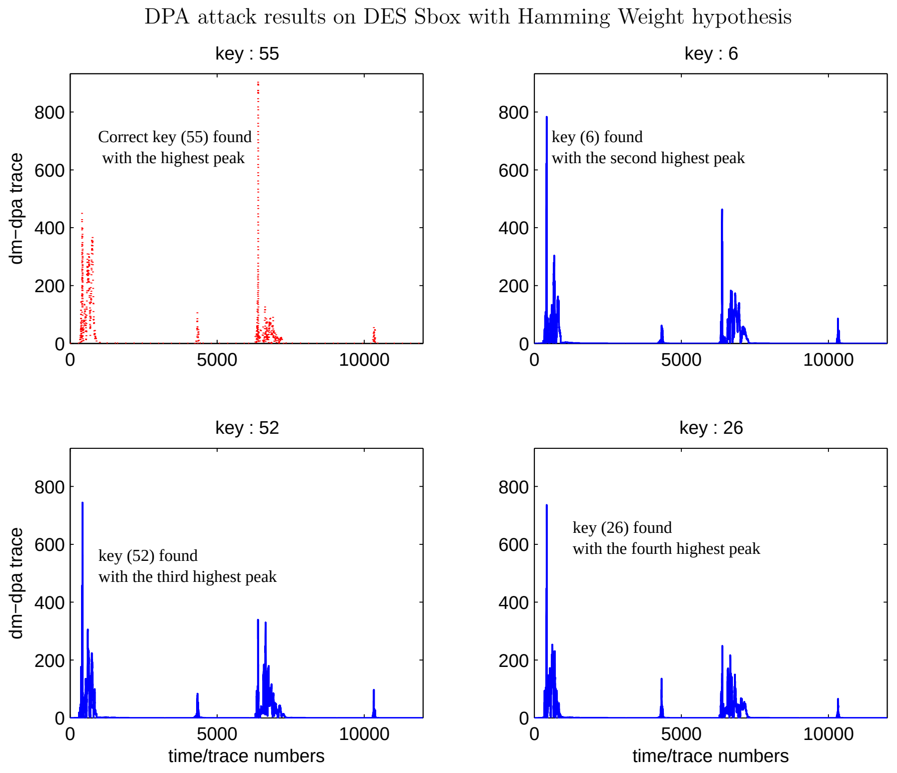


FIGURE 5.5: DPA result on negative logic DES Sbox for 2000 traces

rail circuit. DPA on *positive logic DES sbox8* without routing capacitance was successful after 2,000 traces with the transition count hypothesis and 600 traces with the Hamming weight hypothesis. This is an unexpected result, since DRP logic styles with balanced differential nets are expected to provide a high level of security. DPA on *negative logic DES sbox8* without routing capacitance with the transition count hypothesis was successful after 4000 traces and after 1000 traces with the Hamming weight hypothesis. The DPA result on the *negative logic DES sbox8* with the Hamming weight hypothesis is shown in Figure 5.4.

It is reasonable to assume that dual rail precharge circuits without routing capacitance are the same as dual rail precharge circuits with properly balanced differential wires. Dual rail precharge circuits with properly balanced differential wires are expected not to yield the correct key to the DPA attack. As WDDL and DSDR styles are built on basic CMOS gates rather than custom logic style such as SABL, the difference in power consumption between *true* logic and *false* logic leads to a successful DPA attack. Further the DPA peak was observed in the precharge/discharge phase rather than the evaluate phase (see Figure 5.4). However, when both *negative logic DES sbox8* and *positive logic DES sbox8* were used with the dual spacer protocol (i.e. alternate reset and preset in precharge phase), neither of the hypothesis models revealed the correct key. For time reasons, we limited this simulation to 30,000 traces.

Next we implemented DPA on the *positive logic DES sbox8* with routing capacitance but without the dual spacer. As expected, the correct key was found for both the hypothesis models. The results were similar with *negative logic DES sbox8*, except that this DPA took fewer traces than the *positive logic DES sbox8*. We believe this might have been the effect of glitches from using negative logic. The DPA result on *negative logic DES sbox8* with the transition count hypothesis is shown in Figure 5.5.

When the dual spacer protocol was used in both *negative logic DES sbox8* and *positive logic DES sbox8* circuits the transition count hypothesis was unsuccessful in finding the secret key, but the Hamming weight hypothesis successfully found the secret key. The only difference in the result is that *positive logic DES sbox8* took far more traces than the *negative logic DES sbox8* to find the secret key, again we suspect glitches in the negative logic circuit. From this we can confirm that by using the dual spacer protocol, the number of traces needed to perform DPA increased by 50 times for positive logic and 85 times for negative logic compared to a normal dual rail circuit. It is also interesting to note that the Hamming weight hypothesis yields results quicker than the transition count hypothesis for dual rail circuits.

5.5 Path Switching

Dual rail precharge circuits (WDDL, DSDR) have 100% switching activity, i.e, the (compound dual rail) logic gates switch the same number of times for any combination of inputs. However DPA is successful on dual rail precharge circuits because the power consumed by the *true* part and the *false* part is different. This difference is mainly caused by differences in the routing capacitance. In other words, the path taken by a *true* signal is fixed and different from that of a *false* signal. The greater the imbalance between paths, the greater its effect on DPA.

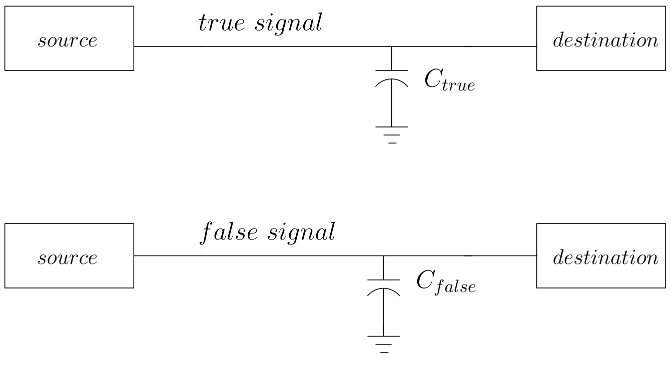


FIGURE 5.6: Normal dual rail

The output signals of registers and other signals (bus) with high load capacitance are usually used for the DPA hypothesis model (focus of attack) [62, 92]. In this section we propose a countermeasure to withstand DPA on high load capacitance signals, without having to change any routing.

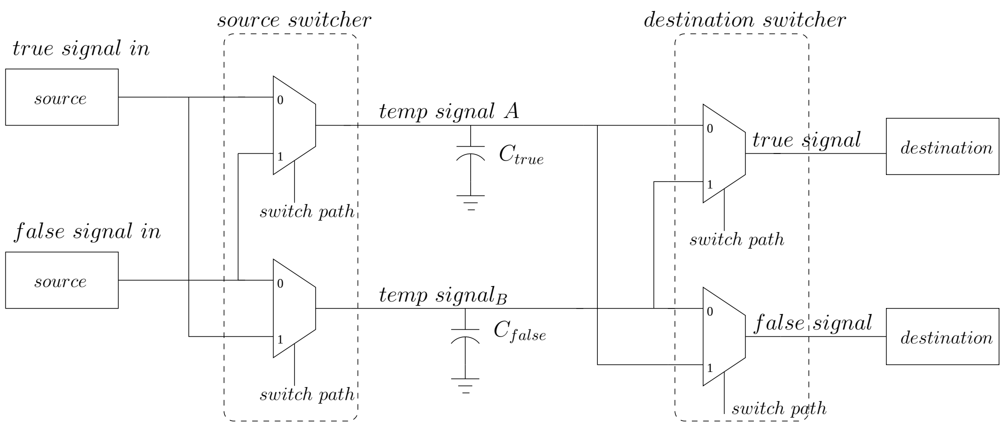


FIGURE 5.7: Path switching dual rail

To solve the imbalance between a *true* signal and a *false* signal we propose a method called path switching. The idea is to randomly swap the *true* signal path and the *false* signal path. By path we mean a signal path which includes wires and any gates. Consider the dual rail signal path in Figure 5.6. If the *true* signal and *false* signal see the same load capacitance $C_{true} = C_{false}$, then the power consumed by the *true* part and the *false* part is the same and hence DPA on this signal would be unsuccessful. However if there is a difference in the load capacitance, then DPA can be successful (shown in Section 5.4). We propose to use the modification in Figure 5.7 to randomly switch the paths. When *'switch path'* changes its value the *true* path and *false* path are swapped. However the functionality remains the same, as the *destination switcher* will switch according to the *source switcher.* The signal *'switch path'* can be tied to a randomly seeded LFSR for randomness. If the path capacitance between *sources/destinations* and the *switcher* is significantly less than the capacitance seen by the original signals, this method will introduce randomness in power consumption. This method will especially introduce randomness in the difference in power consumption between the *true* path and the *false* path. This is easy to achieve, as the *source switcher* can be placed near the *source cell* and the *destination switcher* can be placed with the *destination cell.* In Figure 5.7 multiplexers are used to construct the switcher. As the *'switch path'* signal switches between two complementary signals, an XOR gate can also be used as shown in Figure 5.8, which results in reduced area and wiring.

To see the effectiveness of path switching we modified our test circuit from Figure 5.2 into the circuit shown in Figure 5.9 (complementary logic is not shown for clarity reasons). Note that we have inserted the path switching XOR gates at the *original L input* and *cipher_text output* to implement path switching. We implemented DPA with both hypothesis models. For DPA on *negative logic DES sbox8* (with path switching) we could not find the secret key, for 30,000 traces.

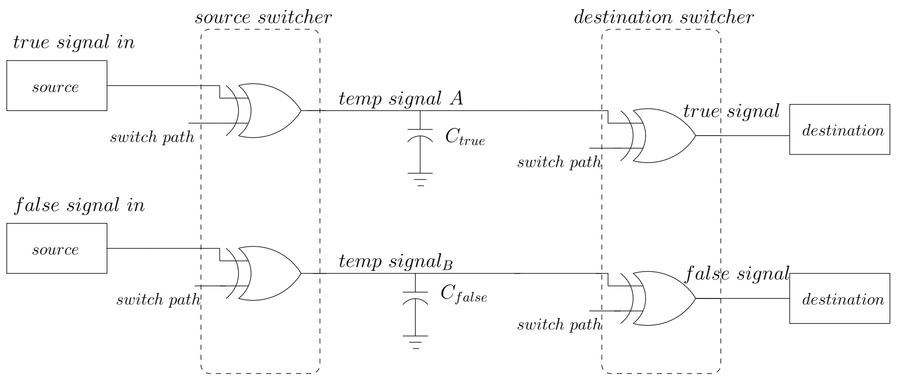


FIGURE 5.8: Path switching dual rail with XOR gate

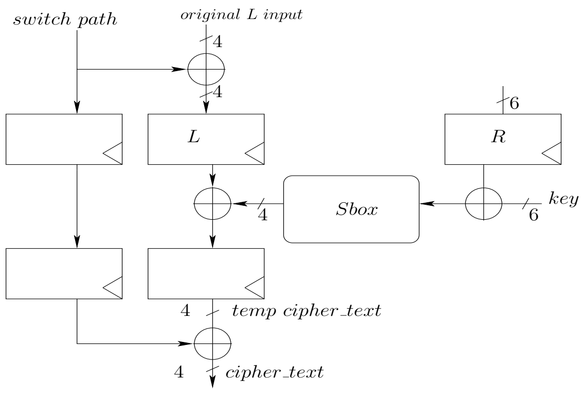


FIGURE 5.9: Test circuit with path switching (complementary logic not shown)

However for DPA on *positive logic DES sbox8* (with path switching) it took about 20,000 traces to find the secret key. The DPA result on *positive logic DES sbox8* with path switching is shown in Figure 5.10.

Next we implemented DPA on a circuit with both dual spacer and path switching and could not find the correct secret key for both the hypothesis models. For time reasons, simulations were limited to 30,000 traces. However 30,000 is not a large number to make sure this countermeasure works. So we simulated *negative logic DES sbox8* with dual spacer and path switching for 300,000 traces. Even then the correct key was not successfully found. Although this does not guarantee that path switching with dual spacer prevents DPA completely, it shows that this solution can increase the number of traces required for DPA significantly. DPA results for this simulation are shown in Figure 5.11.

Finally all the DPA results are summarised in Table 5.2. DRP with path switching

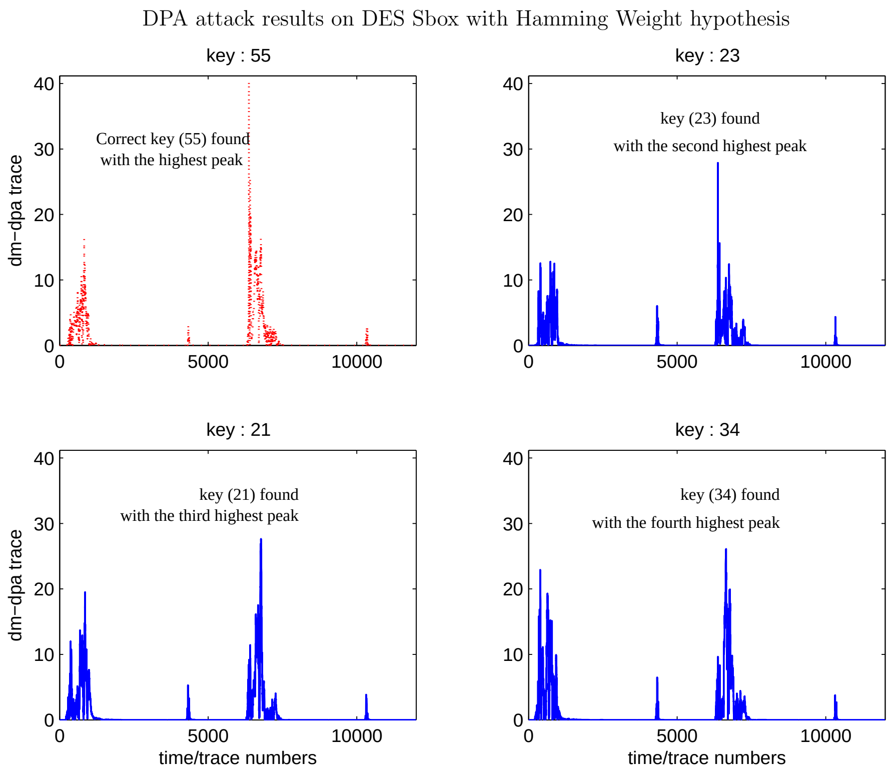


FIGURE 5.10: DPA result on positive logic DRP DES Sbox with path switching, for 30,000 traces

and dual spacer improved the number of traces required by at least 3000 times when compared to single rail circuits and by at least 75 times when compared to normal dual rail precharge circuits.

TABLE 5.1: Area for various implementations of DES Sbox in AMS 0.35μ technology

design	**area in** mm^2	**times increase**
DES Sbox single rail	20202.4	1.0
DES Sbox negative logic dual rail	53974.8	2.67
DES Sbox positive logic dual rail	64530.6	3.19
DES Sbox negative logic dual rail + path switching	61108.8	3.02
DES Sbox positive logic dual rail + path switching	71664.4	3.54

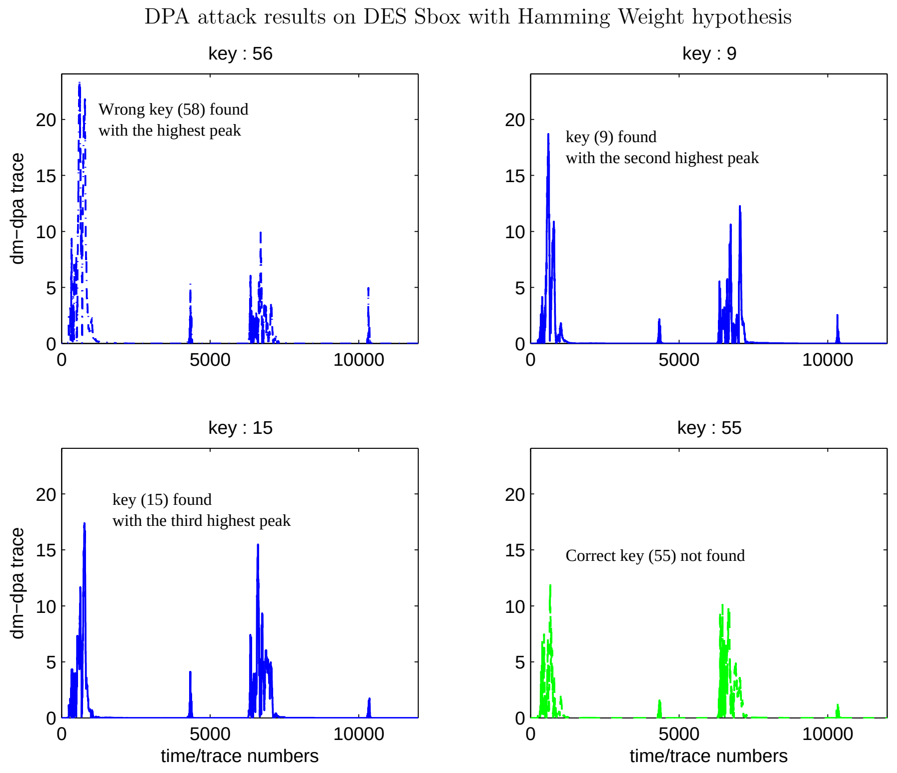


FIGURE 5.11: DPA result on negative logic DRP DES Sbox with dual spacer and path switching, for 300,000 traces

5.6 Path Switching Implementation

Path switching adds two XOR gates to a flip-flop instance or to a high capacitance wire. These XOR gates are redundant in terms of logic functionality, as they do not change the functionality of the design. Implementing path switching before the logic synthesis phase can optimise away these XOR gates. Hence it is desirable to implement path switching after the synthesis phase. It is also necessary to implement dual rail after synthesis as the currently available CAD tools are designed for single rail designs and would optimise any or all of the complementary parts of a dual rail design. Moreover extending (duplicating) a single rail design into a dual rail is not a complex problem and has been discussed in [94]. Since path switching is an extension to dual rail circuits, it also can to be applied after synthesising a single rail design. Path switching implementation, within the context of design flow is shown in Figure 5.12 (standard design flow is shown in Figure 3.1).

Table 5.2: Number of traces required for successful DPA attack on dual rail circuits based on simulations. X means key not found for upto 300,000 traces

	Transition count Hypothesis	**Hamming weight Hypothesis**
DES Sbox single rail	60	90
DES Sbox positive logic without routing cap	2000	600
DES Sbox negative logic without routing cap	4000	1000
DES Sbox positive logic no dual spacer	1100	400
DES Sbox positive logic + dual spacer	X	20,000
DES Sbox positive logic + path switching	X	20,000
DES Sbox positive logic + dual spacer + path switching	X	X
DES Sbox negative logic no dual spacer	90	70
DES Sbox negative logic + dual spacer	X	6,000
DES Sbox negative logic + path switching	X	X
DES Sbox negative logic + dual spacer + path switching	X	X

Path switching can be implemented in two steps. In the first step, path switching is applied before place & route, as shown in Figure 5.13 (solid lines). A list of flip-flop instances and high capacitance wires are selected by the designer to be processed. Such a list of high capacitance nets can be known from the floor plan of the design. A program has been developed, based on OpenAccess [25], to insert the path switching XOR gates. The resulting netlist from path switching insertion can be placed & routed. In the second step, as shown in Figure 5.13 (dashed lines), parasitic data can be extracted from place & route tools to find if any dual rail net pairs have a high ratio of parasitic capacitance. The designer can then generate a new list to apply path switching, based on the dual rail net pairs that have a high ratio of parasitic capacitance. Then path switching can be applied to the new list to re-implement the design. The second step can often be avoided by proper

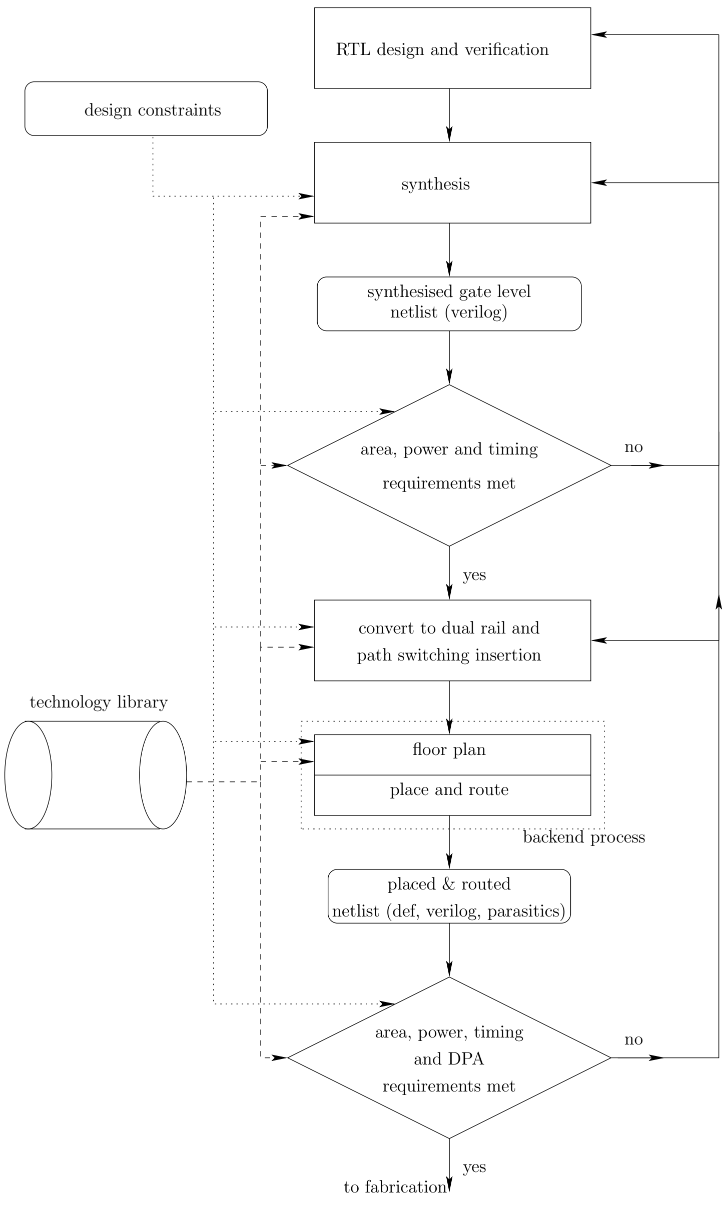

RTL design and verification
design constraints
synthesis
synthesised gate level netlist (verilog)
area, power and timing requirements met
no
yes
convert to dual rail and path switching insertion
technology library
floor plan
place and route
backend process
placed & routed netlist (def, verilog, parasitics)
area, power, timing and DPA requirements met
no
yes
to fabrication

FIGURE 5.12: Path switching implementation within the context of digital design flow

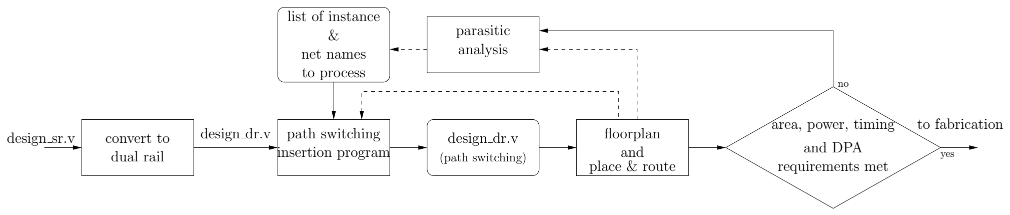

list of instance & net names to process
parasitic analysis
design_sr.v
convert to dual rail
design_dr.v
path switching insertion program
design_dr.v (path switching)
floorplan and place & route
area, power, timing and DPA requirements met
no
to fabrication
yes

FIGURE 5.13: Implementing path switching. Dashed lines show optional steps

floor planning and by inserting path switching XOR gates to all the nets that are expected to have high capacitance. Finally the resulting path switching netlist should be put through DPA analysis requirements, if they are not met, the whole path switching process should be repeated until the DPA analysis requirements are met.

5.6.1 Path Switching Implementation on FPGAs

Path switching on FPGAs is more promising than on ASICs as path switching is the first solution than can be applied to solve the dual rail routing problem on FPGAs, where routing resources are limited. Note that path switching does not fully solve the routing problem, it just increases the tolerance to routing imbalances. The path switching flow presented in Figure 5.13 can be applied to FPGAs and ASICs in general. However FPGAs have a different architecture which enables better implementation of path switching.

The basic idea of path switching is to randomly change (swap) the path taken by the *true* signal and the *false* signal. In order to achieve this we can insert XOR gates at the source and destination of a signal (Figure 5.8). However in most FPGAs the basic building block is a four input Look Up Table (LUT). Implementing a 2 input XOR in a 4 input LUT would waste resources. Instead if the source/destination gate uses less than 4 inputs, the source/destination gate and the XOR gate can be merged into a single LUT. For example consider the example in Figure 5.14(a). As the source gate is using less than the available inputs, the XOR gate can be merged with the source gate. The new LUT will have a different programming value based on the previous LUT's value. For example if a 3 input LUT implementing an AND function will have programming value of $8'HFE$, the new 4 input LUT, with XOR merged can have a programming value of $16'H5556$ (note that LUT programming value also depends on the order inputs are connected).

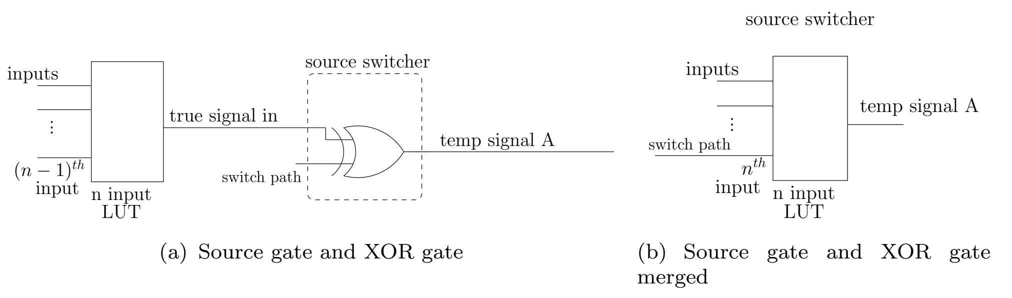


Figure 5.14: Merging XOR gate and Source gate

5.6.2 DPA Results on FPGA Implementation

In this section we present DPA results for the FPGA implementation of path switching. The DPA setup on FPGA was already discussed in Section 4.4 and is unchanged except for the design that is loaded onto the FPGA. We implemented the same DES Sbox circuits, that are used in simulation, on the FPGA. We also used the same key hypotheses that are used in the simulated DPA. Our aim is to test path switching for a large number of traces and to show that path switching can be applied to FPGA circuits. As such, we did not implement the optimisation discussed in Section 5.6.1.

First, we applied DPA to a normal single rail DES Sbox. DPA on single rail circuit with the transition count hypothesis was successful after 91 traces and with the Hamming weight hypothesis was successful after 43 traces. Next we implemented a positive logic dual rail DES Sbox, for which DPA with the transition count hypothesis was successful after 446 traces and with the Hamming weight hypothesis was successful after 262 traces. Next we implemented a positive logic dual rail DES Sbox with dual spacer protocol, for which DPA with the transition count hypothesis was successful after 1,100 traces and with the Hamming weight hypothesis was successful after 832 traces.

Finally, we implemented DPA on a dual rail DES Sbox with path switching. Neither our transition count hypothesis nor Hamming weight hypothesis revealed the

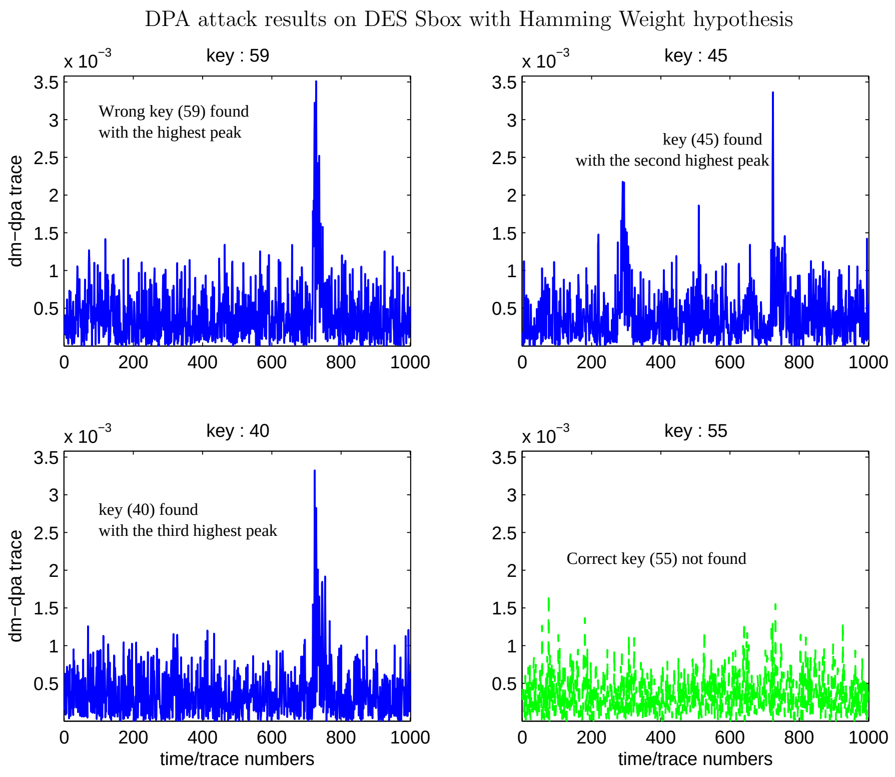


FIGURE 5.15: DPA result on FPGA implementation of positive logic DRP DES Sbox with path switching, for 300,000 traces for hypothesis partition value of 2

correct secret key. The DPA result for dual rail DES Sbox with path switching is shown in Figure 5.15. We found similar results for DPA on dual rail DES Sbox with path switching and dual spacer. The DPA result for dual rail DES Sbox with path switching and dual spacer is shown in Figure 5.16. For the DES Sbox with path switching, we collected a total of 300,000 traces and for DES Sbox with path switching and dual spacer, we collected a total of 3,000,000 traces.

Note that both our hypotheses partition traces into different sets based on the hypothesis value. If the hypothesis value is more than 2, then it is added to set 1 else to set 0 (Page 112). When we used a different number to partition the traces, we found different results. For the dual rail DES Sbox with path switching, a partition cutoff number of 3 revealed the correct key in 155,000 traces. Whereas for the dual rail DES Sbox with path switching and dual spacer, a partition cutoff number of 1 revealed the correct key in 1,330,000 traces. DPA results for different hypothesis partition values are shown in Figure 5.17 and Figure 5.18 for a DES

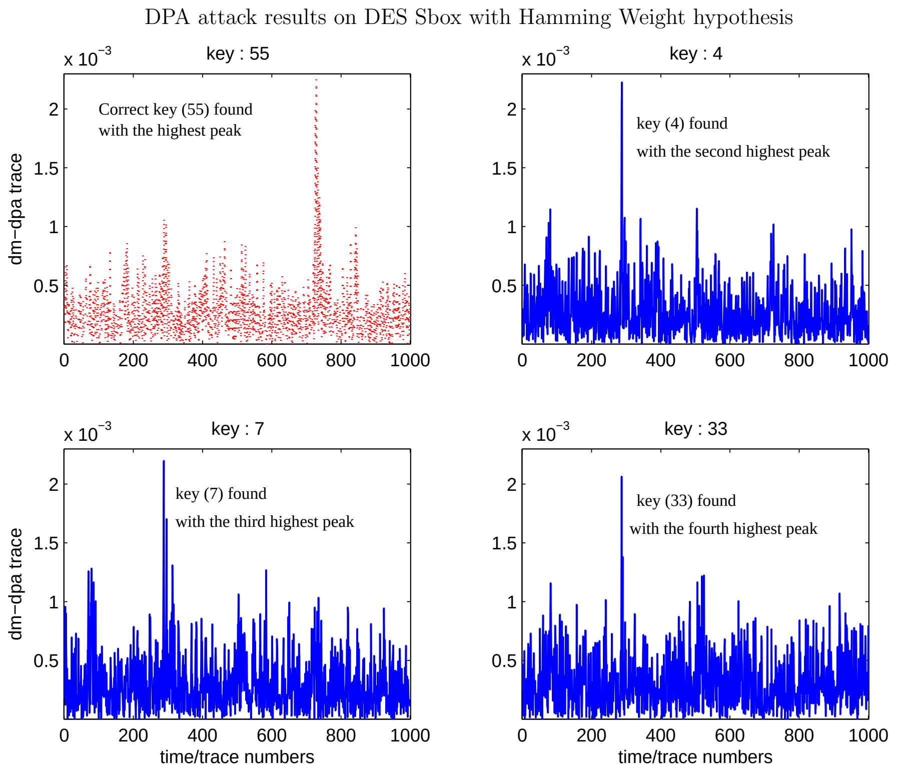


FIGURE 5.16: DPA result on FPGA implementation of positive logic DRP DES Sbox with path switching and dual spacer, for 3000,000 traces for hypothesis partition value of 2

Sbox with path switching and a DES Sbox with path switching and dual spacer respectively.

In addition to the DES Sbox test circuit, we also mounted DPA on AES Sbox, shown in Figure 4.2. We were able to find the correct key from DPA on AES Sbox in 97 traces using transition count hypothesis and 31 traces using Hamming weight hypothesis. For the dual rail AES Sbox it took 200 and 900 traces for transition count and hamming weight hypothesis respectively. Next we implemented DPA on dual rail AES Sbox with path switching and dual spacer; and it required 332,900 traces to find the correct key, using Hamming weight hypothesis and a partition function of 6.

The dual rail DES Sbox with dual spacer protocol on FPGA was easily attacked when compared to the simulation. However both dual rail with path switching and dual rail with path switching and dual spacer have similar DPA results in

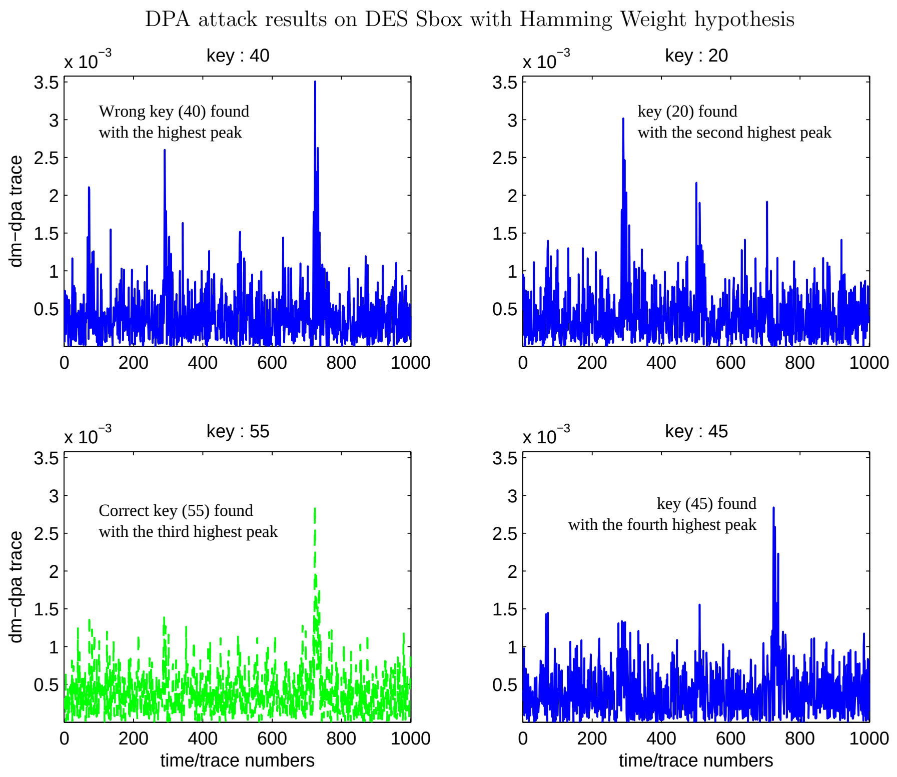


FIGURE 5.17: DPA result on FPGA implementation of positive logic DRP DES Sbox with path switching, for 300,000 traces for hypothesis partition value of 3

TABLE 5.3: Number of traces required for successful DPA attack on dual rail circuits based on FPGA setup

	Transition count hypothesis	**Hamming weight hypothesis**
DES Sbox single rail	91	43
DES Sbox positive logic no dual spacer	446	262
DES Sbox positive logic + dual spacer	1,100	832
DES Sbox positive logic + path switching	X	155,000
DES Sbox positive logic + dual spacer + path switching	X	1,330,000
AES Sbox single rail	97	31
AES Sbox positive logic + no dual spacer	200	900
AES Sbox positive logic + dual spacer + path switching	X	332,900

simulation and on the FPGA. DRP with path switching and dual spacer for DES Sbox increased the number traces required for successful DPA by 30930 when

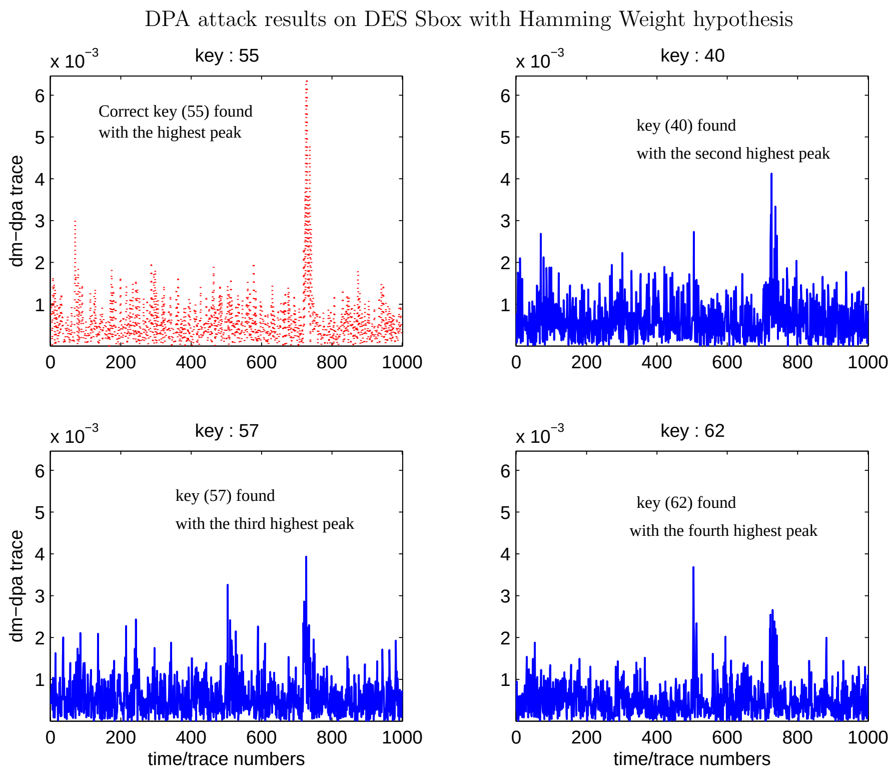


FIGURE 5.18: DPA result on FPGA implementation of positive logic DRP DES Sbox with path switching and dual spacer, for 3000,000 traces for hypothesis partition value of 1

compared with normal single rail design and by 5076 when compared with normal dual rail on our FPGA test setup. DRP with path switching and dual spacer for AES Sbox increased the number traces required for successful DPA by 10738 when compared with normal single rail design and by 1664 when compared with normal dual rail on our FPGA test setup. All the results are summarised in Table 5.3. Detailed results of DPA based on FPGA are presented in Appendix B.

5.7 Area for Implementing Path Switching

Path switching is an addition to dual rail precharge circuits, so its area increase when compared to single rail circuit is atleast two times. Path switching adds 4 multiplexers (or 4 XOR gates) per bit. The area overhead of the dual spacer protocol on top of the existing dual rail circuit is also much less. Only 2 D-type

Table 5.4: Area for various implementations on our Xilinx FPGA board

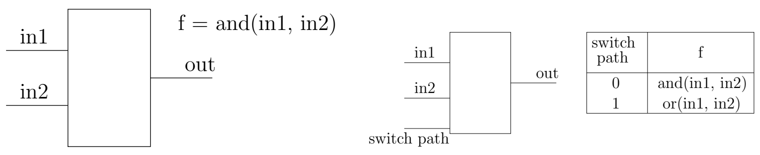
design	**area in number of slices**	**times increase**
DES Sbox single rail	16	1.0
DES Sbox single rail implemented from ASIC netlist	54	3.37
DES Sbox positive logic dual rail	123	7.68
DES Sbox positive logic dual rail + path switching	178	11.125
AES Sbox single rail	45	1.0
AES Sbox single rail implemented from ASIC netlist	74	1.64
AES Sbox positive logic dual rail	158	3.5
AES Sbox positive logic dual rail + path switching	231	5.13

flip-flops and 1 T-type flip-flop is required to make a normal dual rail circuit dual spacer ready. Overall using the dual spacer with path switching has low area overhead on top of existing dual rail precharge logic overheads, while providing a high level of security. Area figures for DES Sbox are tabulated in Table 5.1. *Positive logic DES sbox8* is bigger than its *negative logic* equivalent because of the fact that our AMS 0.35μ technology library did not contain them and we had to mimic the AND-OR braviour using NAND-NOR and INVERTER gates. Path Switching adds approximately 3 times the area overhead for DES Sbox circuit when compared to an unprotected implementation.

Area figures for path switching implementation on FPGA are tabulated in Table 5.4. The DES Sbox path switching implementation has an area overhead of 11 times, while the AES Sbox has an area overhead of 5 times. These numbers are quite different to the ASIC design area figure in Table 5.1, for the following reason; we used an ASIC netlist to implement dual rail circuits on FPGA. This is to ensure the wave propagation feature of WDDL is not disrupted by inverting logic. Unlike ASIC synthesis, we did not find a way to constraint the FPGA synthesis tool to only use certain type of logic. Tiri and Verbauwhede [96] presented a way to optimise FPGA synthesis for WDDL style, so in theory the area figure of path switching on FPGA could be better.

5.8 Extending Path Switching

Path switching's inherent limitation is that it cannot be applied to logic gates. There might be a case where the logic gates need protection against imbalanced dual rail routing. In such cases path switching can be applied to logic gates as well, but at the expense of increased area, as described below. Every n input gate is replaced by an $n + 1$ input gate. The additional input is connected to *switch path* signal. The new gate will have the following function: if *switch path* is 0, the new gate will have the same function as the previous gate (AND), if *switch path* is 1, the new gate will have the complementary function of the previous gate (OR). An example gate is shown in Figure 5.19.



(a) Original gate (b) New gate with path switching

FIGURE 5.19: Extending path switching to logic gates

With logic gates that can change functionality from AND to OR and vice verse, path switching, which was previously only applicable to flip-flops and buses, can now be applied to the entire design. Now path switching XOR gates only need to be inserted at the primary inputs and outputs of the design. The area overhead of extended path switching is quite high, as every gate has to provide dual functionality. Also the *switch path* signal has to be connected to every cell in the design. Extended path switching, although it has high overheads, can be an attractive solution for FPGAs.

5.9 Summary

In this chapter we evaluated dual rail precharge logic styles (WDDL and DSDR) and found that DRP circuits are vulnerable to DPA if the routing capacitance of differential signals is not properly balanced. We then showed that a dual spacer can increase the number of traces required to attack dual rail circuits significantly, but cannot completely prevent it. To solve the dual rail differential routing problem we propose a new countermeasure, path switching, to improve DPA resistance. Using circuit simulations and implementations on FPGA, we showed that dual rail circuits with the dual spacer and path switching significantly increase the number of traces required by DPA. DPA tests on our experiments have shown an increase of 1664 in the number of traces required to find the secret key.

Path switching can be applied to high capacitance buses and registers at an area over head of four XOR gates per bit. An ASIC implementation of path switching on DES Sbox has an area overhead of 3 times. The FPGA implementation of DES Sbox has 11 times the area overhead, while the AES Sbox one has 5 times. The path switching combined with dual spacer area overhead is insignificant when compared to an existing dual rail precharge logic style's overhead. Path switching has an inherent limitation. It cannot be applied to logic gates. It also does not address attacks on combinational logic. However extended path switching can be applied to an entire design, with an increase in area overhead.

Chapter 6

Divided Backend Duplication for Balanced Dual Rail Routing

6.1 Introduction

Of all the DPA countermeasures discussed in Chapter 3, the logic level countermeasures that fall under the dynamic and differential logic style discussed in Section 3.5.3.1 (also referred to as dual rail precharge - DRP) theoretically offer the best resistance to DPA. The basic principle behind DRP logic is to eliminate any information leakage by consuming the same amount of power in every clock cycle. DRP circuits have been proved to prevent DPA, provided the routing of differential nets is balanced [99].

Balancing differential nets (balanced dual rail routing) is not, however, a trivial task. To address the routing problem, to date the following proposals have been put forward: DWDDL [94], fat wire [95], backend duplication [23], three phase dual rail [10], path switching [4], Double WDDL [113] and an iterative correction flow [7]. Of these, three proposals [23, 94, 95] impose some constraints on backend implementation flows and try to solve the problem by eliminating the difference of routing capacitance of differential nets and in the process, these solutions introduce coupling capacitance between the differential nets. Three phase dual rail [10] tries to avoid the routing problem by introducing a third phase, which is an additional overhead. Path Switching [4] (discussed in Chapter 5) offers an improvement to dual rail circuits and can only protect registers and buses with high capacitance. Double WDDL, as the name implies, has two separate WDDL implementations thereby increasing the area overheads by four times. Double WDDL was developed mainly for use in FPGAs [113]. The first WDDL part is implemented using normal place & route flow. The second WDDL part is obtained by copying the first WDDL part, including the routing details, and reversing the original and complementary logic [113]. Backend correction flow, described in [7], is iterative and can consume a significant amount of time to implement a design. In this method the design is successively routed and analysed until every dual rail routing pair is balanced. The analysis consists of collecting routing parasitics for every differential pair nets. This method also requires a complex strategy to constrain the router and a non trivial algorithm to guide the iterative process towards convergences

In this Chapter, we concentrate on the implementation of balanced dual rail precharge logic styles rather than the alternatives. We present a simple yet effective solution to improve dual rail circuit routing capacitance while eliminating coupling capacitance between the differential nets. In Section 6.2 we discuss dual rail precharge logic styles, give a brief introduction to backend design flow, and discuss existing methods and their shortcomings. In Section 6.3 we present the inversion problem and discuss its solutions. In Section 6.4 we present our proposed methodology. In Section 6.4.1 & Section 6.4.2 we present ASIC & FPGA implementations respectively and then conclude the chapter.

6.2 Background

6.2.1 Dual Rail Precharge Logic Styles

Dynamic and differential logic (also referred to as dual rail precharge - DRP) [86, 92, 94] has been proposed to prevent DPA. Dual rail precharge circuits were discussed in detail in Section 3.5.3.1. The idea is to consume the same amount of power for any combination of inputs. This is achieved by using differential logic (two signals instead of one) and by precharging both the differential nets in every clock cycle. In DRP circuits for every logic gate, a complementary gate exists, usually referred to as *false* logic (or *false* part).

Dual rail precharge logic styles can be classified into two types based on the way precharge is applied. Sense amplifier based logic (SABL) is a DRP logic based on the principles of domino logic, where a special precharge signal is applied to every gate to force the gate to precharge. Wave dynamic dual rail (WDDL) and dual spacer dual rail (DSDR) on the other hand propagate the precharge signal from a designs primary inputs and state-elements (flip-flops). WDDL and DSDR have the following differences over SABL: 1) WDDL and DSDR can be constructed using existing CMOS standard cells and 2) that the *true* logic and *false* logic are two different cells. The second point is not true in all cases. WDDL and DSDR both need special inverters, where the *true* and *false* wires are cross connected. As differential logic has both *true* and *false* outputs, an inverter is implemented by exchanging the outputs. Moreover an inverter is an inverting gate, it will stop the precharge wave propagation. Figure 6.1 shows the basic building blocks of WDDL with master slave WDDL flip-flops. Although double the clock frequency is required to get same data rate using master slave flops, these are recommended [94]. All primary inputs are driven by a ‘precharge wave generation’ block, so that individual gates will propagate the precharge. Note that the inverter is implemented by exchanging the dual rail pairs.

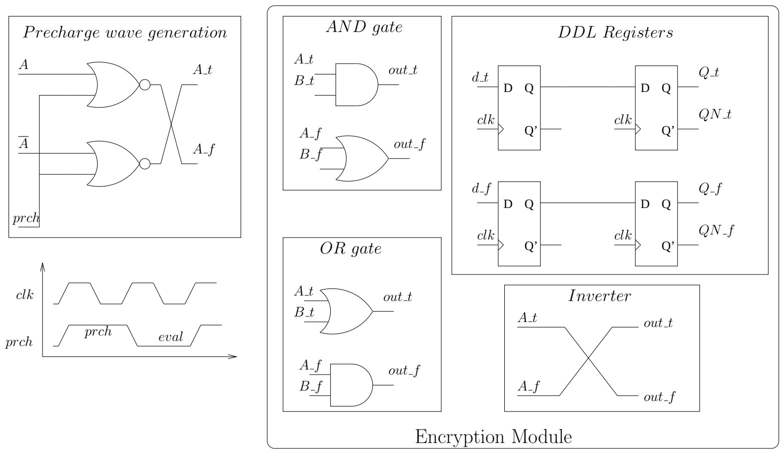
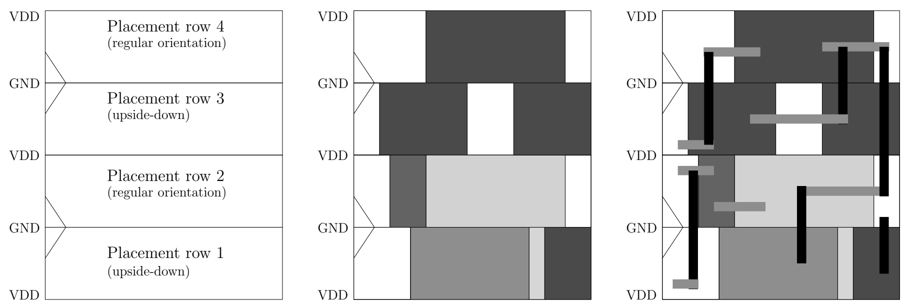


FIGURE 6.1: Building blocks of WDDL,with Master Slave WDDL flip-flops

6.2.2 Backend Design Flow

Most of the digital designs implemented today are based on a standard cell flow. A set of commonly used standard cells are designed and characterised such that CAD tools can be used to automate most of the design flow. Design entry is typically in behavioural HDL which is synthesised and mapped to the target technology's standard cells. After the synthesis, the resulting netlist is placed and routed to get the final design. Backend design usually refers to the implementation of the design after the synthesis phase and mainly involves floor planning, placement and routing. A placer partitions the available core area into rows, where the standard cells are placed. In a similar fashion, a router partitions the core area into horizontal and vertical routing grids. Each grid has a minimum size defined by the target technology's wire pitch size.

The place and route flow usually involves the following steps, shown in Figure 6.2. First a floor plan is made (Figure 6.2(a)). This is where the aspect ratio (or the dimensions) of the chip are determined. Next the standard cells are placed (Figure 6.2(b)) and finally the wires are routed (Figure 6.2(c)).



(a) Normal backend flow: create a floor plan (b) Normal backend flow: place the standard cells (c) Normal backend flow: route the wires

FIGURE 6.2: Normal backend flow overview

6.2.3 Existing Methods

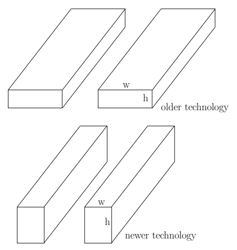
Divided wave dynamic differential logic (DWDDL) was proposed by *Tiri and Verbauwhede* [94] to address routing imbalances in DRP logic styles. DWDDL's idea is to place and route a single rail design (the *true* part), copy it and replace the complementary cells (for example 'and' with 'or' and vice verse) to get the *false* part. However, this method assumes that there is no inversion in the single rail design, as an inverting cell would stop the precharge wave propagation. However, in practice it is difficult to have logic without inversion. This is the only known limitation for DWDDL and no further work has been reported on it.

Fat wire was proposed by *Tiri and Verbauwhede* [95] to address routing imbalances in DRP logic styles. In this methodology a fat wire is constructed from two adjacent normal wires. For the fat wire method to work, first the dual rail netlist, instantiating dual rail cells, has to be placed. Then instead of routing two differential wires (for the *true* and *false* signals) a single fat wire is routed and later decomposed into two normal single wires which will have same wire length.

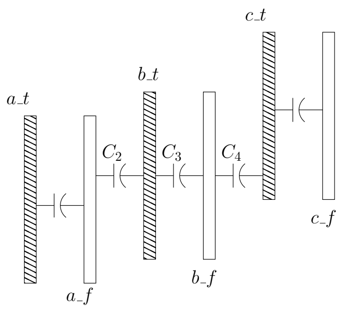
Backend duplication was proposed by Guilley *et al.* [23] [23] to address routing imbalances in DRP logic styles. The basic idea of backend duplication is based on placement and routing obstructions (constraints to the CAD tool). The first step of backend duplication is to constrain the CAD tool (1) to only use alternate rows for placing cells and routing horizontal routes (2) and to use the alternate routing pitches for routing vertical routes. Thus, when the placer has finished placing the single rail design, a dual rail design can be obtained from copying (and transforming) the single rail into the previously obstructed rows. Note that this operation is a simple shift in coordinates of the placed cells. Duplicating the routes is done in two steps. Once the design is routed, horizontal routes are duplicated in the same way as cells. Vertical routes are duplicated by a simple shift in the x-axis of the routing pitch.

6.2.4 Shortcomings of the Existing Methods

Coupling capacitance has become one of the most critical issues in deep sub micron physical designs because of 1) interconnect dominated circuit delay and 2) strong coupling effects between interconnect wires [108]. As technology scales the wire widths, their relative height is increased and coupling capacitance between wires increases [108] (Figure 6.3(a)).



(a) Effect of shrinking wire widths on coupling capacitance



(b) Coupling capacitance in dual rail circuit

FIGURE 6.3: Coupling capacitance effects

In the fat wire and backend duplication methods (vertical routes) dual rail wires end up next to each other, as shown in Figure 6.3(b). With coupling capacitance increasing, the effective capacitance seen by a *true* and *false* signal will vary. For example consider dual rail pairs b_t & b_f. The coupling capacitance seen by b_t

is C_2 & C_3 whereas the coupling capacitance seen by b_f is C_3 & C_4. Now if the capacitance C_2 & C_4 vary by a huge difference, the resulting design can have unbalanced wire capacitance and this can lead to information leakage. Note that this effect becomes more and more dominant as technology scales down. The effect of coupling between differential wires is more significant in the fat wire method than in backend duplication since the horizontal wires are also next to each other.

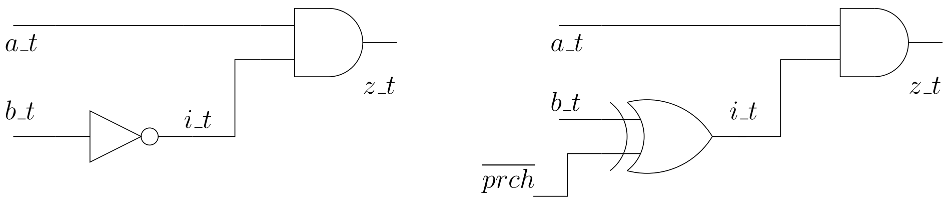
Of course spacing between dual rail wires can always be increased to reduce the coupling capacitance, however such an increase comes at the expense of increased area and reduced routing resources. Of the three methods to address routing problems, DWDDL is the simplest and most effective. However practical designs will always have inversion and hence will not be able to use the DWDDL method.

6.3 Inversion Problem in DRP Logic

Inversion in dual rail precharge logic styles is considered as a free operation, since dual rail signal pairs are complementary; inversion is simply obtained by exchanging the dual rail pairs. On the other hand, an inverter cannot exist in a WDDL or DSDR style design since it would stop the precharge wave propagation. In other words, inversion is only possible by exchanging the dual rail pair. This property of WDDL and DSDR logic styles prevents designs from using a DWDDL style of implementation. Of course dual rail pairs can be exchanged after DWDDL implementation, but there is no systematic way of doing this. Moreover, the extra wire capacitance from this exchange can add to the critical path delay of a design and can introduce unbalanced wires. This issue of exchanging wires can be worst when the number of unused inverters in a design increases. As an example a 8ns clock period, 128 bit AES had 5,762 inverters from a total gate count of 22,704, excluding buffers used for the clock tree. For this example, we increased the area and delay cost of the original inverter by 10 times so that synthesis tool will use it only when inversion is needed and not for buffering.

6.3.1 Mitigating the Inversion Problem in DRP Logic

Inverters cannot exists in WDDL and DSDR style designs as they would stop the precharge wave propagation. On the other hand, designing logic without inversion is difficult. It is possible to have a cell that behaves as an inverter and still not prevent the precharge wave propagation: this is possible by using a two input XOR gate instead of an inverter and connecting the second input of XOR to the negated precharge signal that is used in generating the precharge wave (Figure 6.1).



a_t	b_t	i_t	$prch$	z_t
0	0	1	1	0
1	1	0	1	0
0	0	1	0	0
0	1	0	0	0
1	0	1	0	1
1	1	0	0	0

a_t	b_t	i_t	$prch$	z_t
0	0	0	1	0
1	1	1	1	1
0	0	1	0	0
0	1	0	0	0
1	0	1	0	1
1	1	0	0	0

FIGURE 6.4: Using XOR instead of an Inverter (Inputs a_t & a_f are driven by precharge wave generation block shown in Figure 6.1)

Consider the example circuit on the left side of Figure 6.4, with the truth table shown. When the $prch$ signal is high all primary inputs are set to logic 1 (inputs a_t & a_f are driven by precharge wave generation block shown in Figure 6.1). However intermediate signal i_t (output of the inverter) will not propagate the precharge wave and the output signal z_t will not be precharged. Now consider the circuit on the right of Figure 6.4. A two input XOR gate is used instead of an inverter. The original input and output of the inverter are connected as before to the XOR. The second input of the XOR is connected to the $\overline{prch}$ signal, which is used in precharge wave propagation. When *prch* is high the XOR will act as a buffer allowing the precharge wave to propagate and when *prch* is low XOR will act as an inverter as intended in the original circuit.

It is also possible to use a Domino-style inverter (similar to the one presented in [10]) instead of an XOR gate. As in the case of the XOR, $\overline{prch}$ is used to precharge the domino-inverter. In the case of a domino style inverter, the timing of $\overline{prch}$ is important for the circuit to work. Because of this, we prefer to use an XOR gate and in the rest of this chapter we use XOR gates to replace inverters. Note that inverters that are used in clock tree synthesis need not be replaced, as the clock signal is not precharged like normal inputs. Based on this, we now present a method to implement a fully balanced dual rail design.

6.4 Proposed Method: Divided Backend Duplication

With XOR gates replacing inverters, a dual rail circuit can be implemented as a physically separate (without any connections) *true* part (original single rail part) and *false* part (complementary part). The primary inputs and outputs will still remain common for both the *true* and *false* parts. With this advantage the divided WDDL implementation, [94], can now be implemented provided that 1) the pins of complementary standard cells should be same, i.e, at same location and same metal layer and 2) the size of the complementary standard cells are the same. Divided backend duplication implementation, within the context of design flow is shown in Figure 6.5 (a standard design flow is shown in Figure 3.1).

Figure 6.6 shows the overview of our proposed method for balanced dual rail routing. This method is similar to the backend duplication method, [23]. A single rail design is used for the initial place and route process and then duplicated to

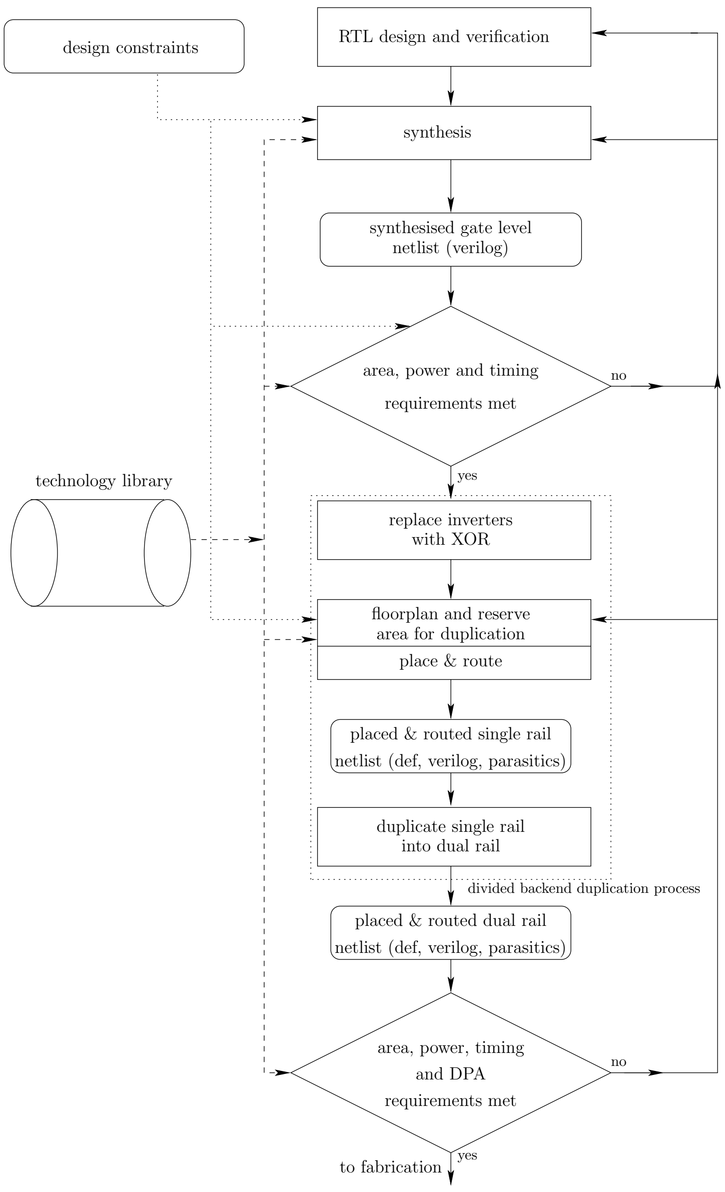
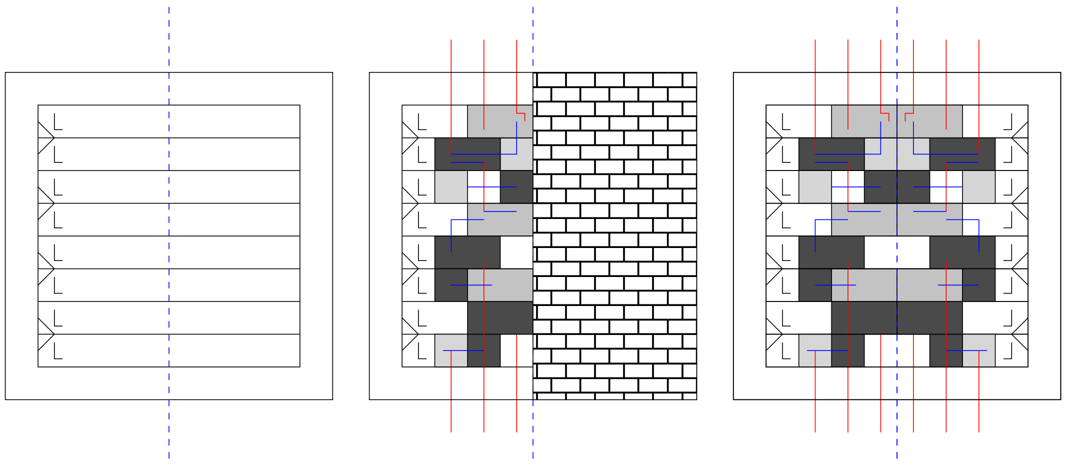

design constraints
RTL design and verification
synthesis
synthesised gate level netlist (verilog)
area, power and timing requirements met
no
yes
technology library
replace inverters with XOR
floorplan and reserve area for duplication
place & route
placed & routed single rail netlist (def, verilog, parasitics)
duplicate single rail into dual rail
divided backend duplication process
placed & routed dual rail netlist (def, verilog, parasitics)
area, power, timing and DPA requirements met
no
yes
to fabrication

FIGURE 6.5: Divided backend duplication implementation within the context of digital design flow

get the final dual rail design. The process can be divided into the following steps (shown in Figure 6.7).

1. A WDDL-compliant single rail design is processed to replace the inverter cells with XOR cells (Figure 6.4). A program has been written for this conversion, based on OpenAccess [83]. At this stage the design is still single rail.



(a) Initial Floor Plan (b) Reserve space for duplicating complimentary logic (c) Flip every object (cells & routes) to right

FIGURE 6.6: Proposed method overview

2. A floor plan is made for the processed single rail design, with utilisation of half the required final utilisation. This ensures that there is enough space for duplicating the complementary part (Figure 6.6(a)).

3. Half of the floor plan area is reserved (obstructed) for the complementary part (Figure 6.6(b)).

4. The single rail design is implemented in the usual way, i.e, place and route, timing analysis, SI analysis, ECO fixes, etc.

5. After the single rail design is finalised, the complementary part can be obtained by flipping every object in the single rail design to the right and by replacing the complementary cells, *AND* with *OR* and vice verse, as shown in Figure 6.6(c). This step can be done by processing the DEF file and is similar to the process used in fat wire [95] and backend duplication [23].

As our proposed method is derived from DWDDL and backend duplication, we call it divided backend duplication (DBD). A small variation to the duplication process can be made: 1) Instead of flipping the design objects to right, they can

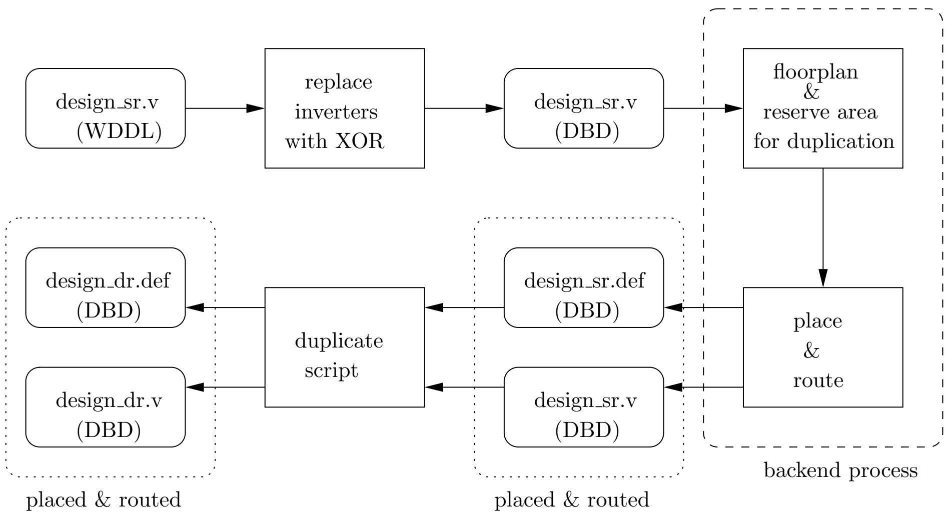
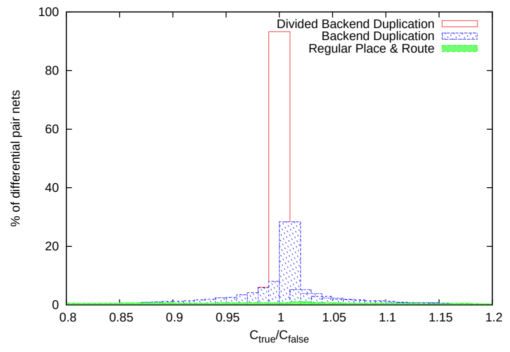


FIGURE 6.7: Divided backend duplication implementation

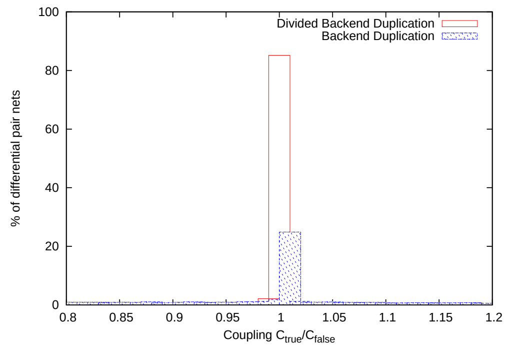
be shifted by half of the core width. 2) Instead of flipping the design objects along the x-axis, this can be done on the y-axis too (flipping to top or bottom).

6.4.1 ASIC Implementation

To show the effectiveness of divided backend duplication, we implemented an AES test circuit with 100k+ gates in a 130nm process. Three different designs are implemented. All designs have the same constraints and netlist. The difference is in implementation. The first implementation, which we call "regular place & route design", is implemented without any special techniques. The second implementation, which we call "backend duplicated design", is implemented as suggested in [23] and is based on the WDDL logic style [94]. The third design, which we call "divided backend duplicated design", is implemented as suggested in Section 6.4 and is also based on WDDL logic style [94]. All the designs aspect ratios are set to 1. The row utilisation of "regular place & route design" is set to 0.70 while for "backend duplicated design" and "divided backend duplicated design" it is set to 0.35 (half the required utilisation, so that enough room is available for duplication). We used Cadence Encounter tools [90] to perform the backend implementation. For parasitic extractions we used Encounter's native extractor and set the "detailed" and "coupling" switches to true. After the parasitic extraction, all the parasitic information was exported into a Standard Parasitic Exchange Format (SPEF) file containing the ground capacitance, coupling capacitance and resistance of every wire.



(a) Ratio of total capacitance of differential pair nets



(b) Ratio of coupling capacitance of differential pair nets

FIGURE 6.8: Ratio of capacitance of differential pair nets

Figure 6.8 shows histograms in which the internal interconnect capacitance of the regular place and route design, the backend duplicated design and divided backend duplicated (DBD) design are compared. We have not implemented fat wire [95] as the effect of coupling on dual rail signal pairs from fat wire should be similar to that of the backend duplication method [23]. The capacitance per net was extracted from the SPEF file, which in turn was reported from Encounter. Figure 6.8(a) shows the distribution of the ratio between the capacitance at the *true* signal net and the capacitance at the corresponding *false* signal net (C_{true}/C_{false}). The ratio C_{true}/C_{false} for regular place & route method is between 0.01 & 10 and for the backend duplication method it is between 0.70 & 1.5. On the other hand, for the divided backend duplication method this ratio is only between 0.90 & 1.1. The percentage of nets that have a ratio of 1 for divided backend duplication is 93.25% when compared to 28.34% for backend duplication.

Figure 6.8(b) is similar as Figure 6.8(a) except that coupling capacitance is only considered instead of total capacitance. The cumulative coupling capacitance per net was extracted from SPEF file, which in turn was reported from Encounter. Coupling capacitance ratio, $Coupling\ C_{true}/C_{false}$ for regular place & route method are not shown as the ratio for some nets was as high as 70. For the backend duplication method, the ratio $Coupling\ C_{true}/C_{false}$ is between 0.22 & 3.52 while for divided backend duplication is 0.60 & 1.9. The percentage of nets that have a ratio of 1 for divided backend duplication is 85.15% when compared to 24.86% for backend duplication. As discussed in Section 6.2.4, this increase in capacitance ratio for backend duplication method is due to unevenly distributed coupling capacitance, whereas the divided backend duplication method shows much less variation.

6.4.2 FPGA Implementation

Differential routing on FPGAs is more difficult than on ASICs as the routing resources are limited. *Tiri and Verbauwhede* [96] have discussed a WDDL implementation on FPGAs and proposed a synthesis flow. However, the differential routing problem in FPGAs has not been addressed to the best of our knowledge. In this section we discuss how the divided backend duplication method can be applied to get balanced differential routing in FPGAs.

Before implementing a design in FPGA, it has to be synthesised to the target FPGA. Synthesising for a secure dual rail implementation has been discussed in detail in [96]. We adopt the flow presented in [96] to synthesise for divided backend duplication implementation with the modifications shown in Figure 6.9. After replacing the inverters with XORs, FPGA synthesis can be done with a commercial CAD tool or "Clustering" technique described in [96]. Care needs to be taken if Commercial CAD tools are used, to preserve the wave dynamic nature of the design. Note that the structural *true* and *false* part are identical for FPGAs, the only difference being the LUT programming value.

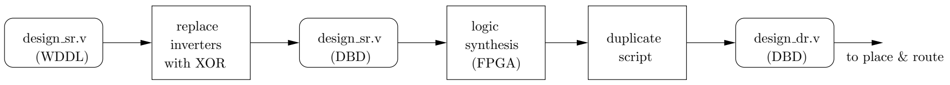
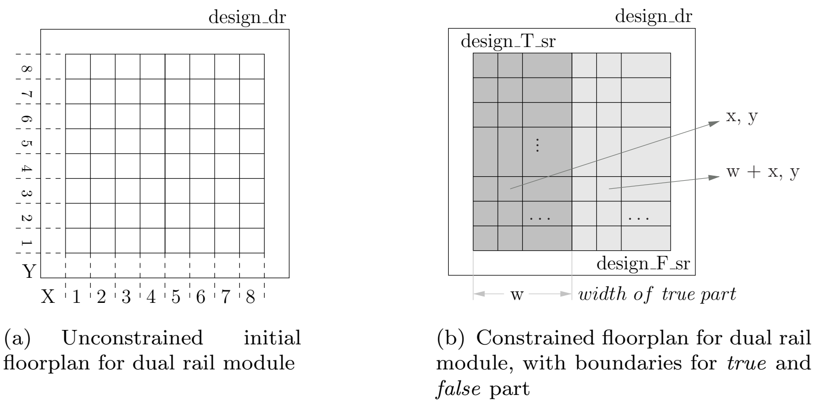


FIGURE 6.9: Divided backend duplication synthesis for FPGAs



(a) Unconstrained initial floorplan for dual rail module

(b) Constrained floorplan for dual rail module, with boundaries for *true* and *false* part

FIGURE 6.10: Floorplanning to implement divided backend duplication dual rail design on FPGAs

FPGAs have highly regular structure as shown in Figure 6.10(a). Each box in Figure 6.10(a) corresponds to a configurable logic Block (CLB) and its associated routing resources. Unlike ASICs, the place & route process of FPGAs is not standardised. This makes it difficult to duplicate the placement and routing information for complementary parts of a dual rail design. Although each FPGA vendor has a specific implementation tool, most of the tools offer procedures to 1) floor plan and 2) constrain a designs instance to a specific location. However, constraining a net to a specific routing resource is not supported. Based on this, the process to implement a balanced dual rail design in FPGAs can be divided into the following steps.

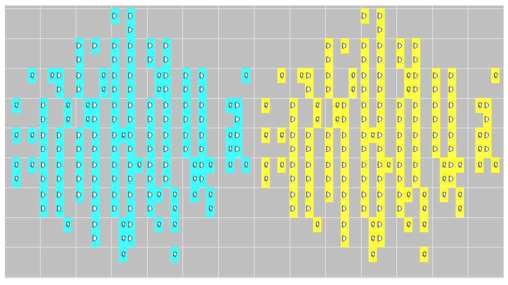
1. The WDDL-compliant single rail design is processed to replace the inverter cells with XOR cells and to transform the netlist into a FPGA-specific netlist (Figure 6.9).

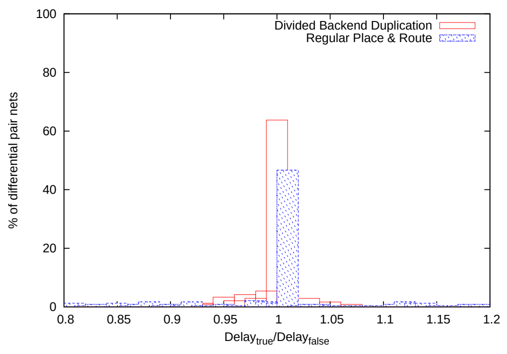
2. The floorplan area is divided into two equal parts (for the *true* and *false* parts), comprising equal number of CLBs, local routing resources and global routing resources (Figure 6.10(b)).

3. The top-level dual-rail design is implemented in the usual way, without violating the boundary constraints set above. The implementation steps usually are place & route, timing analysis, ECO fixes, etc.

4. After the top-level dual-rail design is successfully implemented, locations of all the instances of *true* part are saved to a file. Based on the location of a *true* part's instance, the corresponding *false* part's instance is calculated and written to a constraint file.

5. Based on the new constraints, the *false* part is re-implemented.


(a) Floorplan view of duplicated design on a Xilinx FPGA



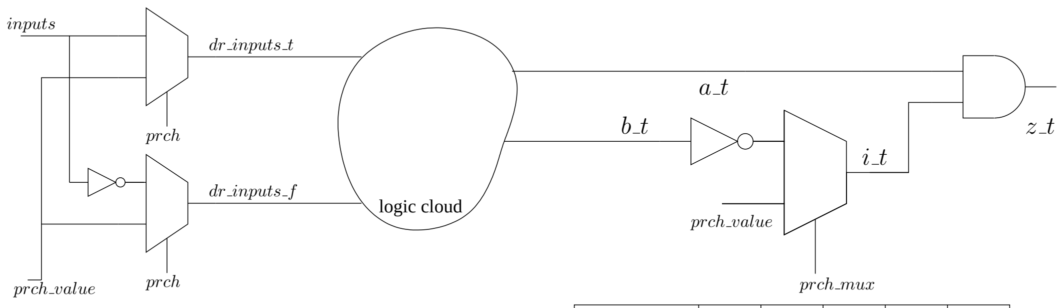
(b) Ratio of delay of differential pair nets

FIGURE 6.11: Divided backend duplication implementation results on a Xilinx FPGA

To see the effectiveness of backend duplication, we implemented a DES sbox on a Xilinx FPGA [111]. Xilinx's XST tool was used for synthesis and ISE was used for implementation. The Xilinx Floorplan editor was used to constrain the floor plan. After the initial place & route Xilinx's Floorplan editor was used to save all the instance locations. The final place & route process was constrained by using Xilinx's UCF file. Figure 6.11(a) shows a floor plan view of such a duplicated design. Although FPGA implementation tools do not report detailed parasitic information, they report delays associated with an instance and interconnect in a Standard Delay File (SDF). This SDF file was analysed and the resulting distribution of the ratio between the delay at the *true* signal net and the delay at the corresponding *false* signal net ($Delay_{true}/Delay_{false}$) is shown in Figure 6.11(b). The delay ratio $Delay_{true}/Delay_{false}$ for the regular place & route method is between 0.40 & 2.7 and for the divided backend duplication method it is between 0.8 & 1.2. The percentage of nets that have a ratio of 1 for divided backend duplication is 64.25% compared to 46.34% for regular place & route. Although we have constrained an instance to be at a specific location, the implementation tool is free to connect the wires and may be the reason for only 64.25% of nets to have a ratio of 1. Note that we are not constraining the FPGA tool to duplicate the routes, as we could not find a way to achieve this. *Yu and Schaumont* have implemented a duplication method for Double WDDL style on Xilinx FPGAs [113] that can be used to completely balance the routing of differential nets on FPGAs. a way to constrain the routing, there are tool specific ways to achieve this, Xilinx for example has Guided PAR & SmartGuide.

6.4.3 Backend Duplication by using Mux

The XOR gate, discussed above, can introduce glitches, especially when both its input change at the same time. As the same *prch* signal is used to drive the the primary inputs to precharge and the XOR gate, both the inputs to XOR gate can change at the same time resulting in glitches. Mangard *et al.* [41] demonstrated on how glitches at gate level can be used in DPA. Hence it is desirable to avoid glitches.



$prch\_value$	a_t	b_t	i_t	$prch$	z_t
0	0	0	0	1	0
1	1	1	1	1	1
X	0	0	1	0	0
X	0	1	0	0	0
X	1	0	1	0	1
X	1	1	0	0	0

FIGURE 6.12: Using MUX instead of an Inverter

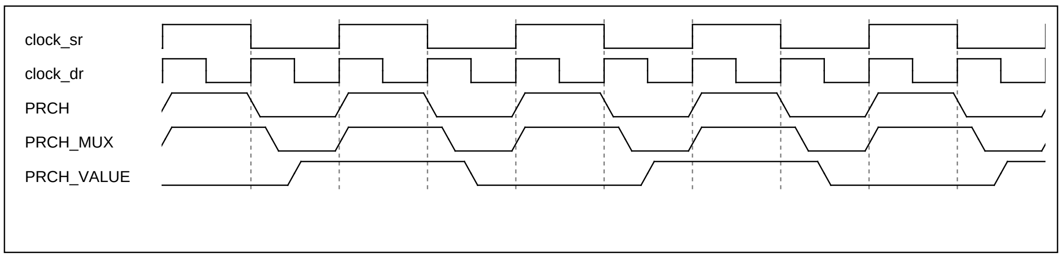


FIGURE 6.13: Timing of PRCH and PRCH_MUX signals to avoid glitches in divided backend duplication using MUX

To avoid any possible glitches at the XOR gate, a MUX can be used as shown in Figure 6.12. First input of MUX is the original signal, second input is the *PRCH_VALUE* signal used in precharge circuits at primary inputs, and the select signal of MUX is *PRCH_MUX*. Figure 6.13 shows the timing relationship of *PRCH*, *PRCH_MUX* and *PRCH_VALUE* signals. Aim is to avoid unnecessary signal transitions. *PRCH_MUX* and *PRCH* go high at the same time, but *PRCH_MUX* goes low after *PRCH*. This delay in *PRCH_MUX* ensures that first input to MUX is settled before selecting it. Similarly, *PRCH_VALUE* should not change its value at the same time as *PRCH* and *PRCH_VALUE* signals. The control circuit to generate the *PRCH*, *PRCH_MUX* and *PRCH_VALUE* signals is, obviously, more complicated to implement than a normal precharge wave generation circuit.

6.4.4 Area for Implementing Divided Backend Duplication

Table 6.1: area for for various implementations in ST 0.12μ technology

design	**area in mm^2**	**times increase**
DES Sbox single rail	3282.2	1.0
DES Sbox normal dual rail	7936	2.4
DES Sbox DBD dual rail	9226.7	2.8
AES Sbox single rail	6001.2	1.0
AES Sbox normal dual rail	13099.6	2.2
AES Sbox DBD dual rail	21289.7	3.54
AES single rail	209945.4	1.0
AES normal dual rail	464742.6	2.2
AES DBD dual rail	697204.8	3.3

As DBD is based on dual rail logic, its area overhead is twice as much of the single rail design. In Table 6.1 area for various designs in ST 0.12μ technology are tabulated. The DES Sbox DBD implementation has 2.8 times area overhead compared to the unprotected single rail, while the AES SBox has 3.5 times and the AES has 3.3 times.

6.4.5 Advantages of Divided Backend Duplication

The main advantage of divided backend duplication is that both the *true* and *false* parts see the same environment. The coupling capacitance problem discussed in Section 6.2.4 is now eliminated. As divided backend duplication is based on standard cells implementation styles such as WDDL and DSDR, it can be adapted to both ASICs and FPGAs.

Divided backend duplication will not have a problem with diagonal routing, an upcoming interconnect technology (already available in Xilinx FPGAs and supported by the Cadence X architecture router), whereas backend duplication currently cannot handle it. Implementing divided backend duplication process is a straightforward process. Neither specific design rules need to be changed nor specific routing blocks have to be imposed on the design. In our example implementation for ASIC, the run time was 3 times less when compared to backend duplication. As the *true* and *false* parts are not interleaved, implementing any engineering change orders (ECOs) is also simple and straightforward.

The only requirements to implement divided backend duplication are that 1) the pins of complementary standard cells should be same, i.e, at same location and same metal layer and 2) the size of complementary standard cells are the same. This is an advantage when compared to the requirements imposed by fat wire [95] and backend duplication [23].

As divided backend duplication separates the *true* and *false* part, a by-product is that two separate data sets can be processed at the same time, instead of one. Divided backend duplication designs can have a *random mode* where one part can process the required data and the other can process random data. Further the entire design can be configured such that the design can randomly switch from *dual rail mode* to *random mode* and back. Divided backend duplication designs can even be configured to operate either the *true* or *false* part at a given time to reduce power consumption, when DPA countermeasure is not required. The only requirement to achieve this is to change the input/output interface to the dual rail design.

6.4.6 Disadvantages of Divided Backend Duplication

The main disadvantage of the divided duplication method is the additional area and delay overhead introduced by replacing inverters with XOR gates. The number of XOR cells used depends on the design and cannot be generalised. For our AES test circuit about 25% of cells were XORs. This increased the critical path delay by 1.2 times. The delay and area overhead introduced by XOR can be minimised by using a domino style inverter instead of XOR. Also the *prch* signal needs to be buffered as it drives all the extra XOR cells.

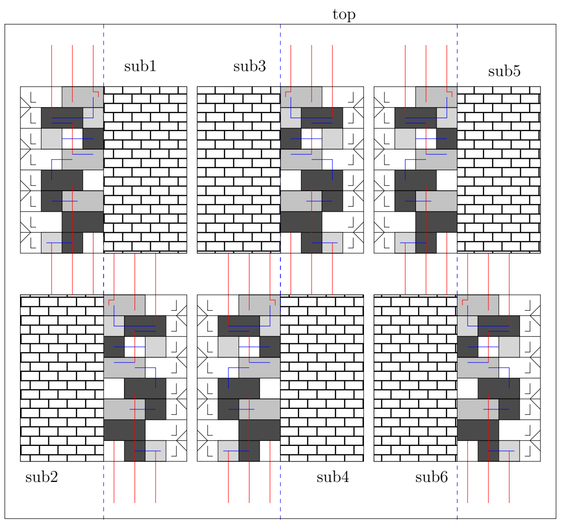


FIGURE 6.14: Hierarchical divided backend duplication

As the *true* and *false* part of the design are physically separated, there may be a concern that EM analysis attacks [20] may be successful, by only observing the *true* or *false* part. Although this may seem unlikely, one may minimise the extent of this concern by taking a hierarchical approach to implementing divided backend duplication compared with that shown in Figure 6.7. An example floor plan for a hierarchical divided backend duplication is shown in Figure 6.14. Another approach would be to use the backend duplication method [23], but with the following difference for duplication: instead of shifting to the right, every object can be flipped to the right.

6.5 DPA Results

In this section we present results from simulation based DPA on a divided backend duplication circuit. In previous chapters AMS 0.35μ technology design kit was used to simulate our circuit for experimenting with DPA. AMS 0.35μ technology design kit, however, did not have all the necessary files to allow us to duplicate the design as described above. Hence ST 0.12μ technology was used. We used same DES sbox circuit as in Chapter 5. The major difference between simulating a ST design kit and AMS is that, in AMS we could get a spice netlist including all parasitics, where as in ST the paracitics were used from a SPEF file. In both the cases Nanosim fast SPICE simulator was used.

Table 6.2: Number of traces required for successful DPA attack on different implementations of DES sbox, based on simulations. X means key not found in 300,000 traces

	Hamming weight Hypothesis
single rail	1,000
dual rail	3,500
dual rail + dual spacer	50,000
dual rail without RC	X
dual rail without RC + dual spacer	X
dual rail + XOR based DBD	700
dual rail + XOR based DBD + dual spacer	6,000
dual rail + MUX based DBD	X
dual rail + MUX based DBD + dual spacer	X

We first implemented DPA on normal single rail circuit. For this circuit the correct key was revealed in 1000 traces. Dual rail circuit revealed correct key in 3,500 traces and dual rail circuit with dual spacer revealed correct key in 50,000 traces. Although these actual numbers are not exactly the same in comparison to results on AMS 0.35μ technology the percentage of increase in comparison to a normal single rail circuit in both the design kits is similar.

XOR based DBD circuit revealed the correct key in 700 hundred traces where as XOR based DBD circuit with dual spacer revealed 3,500 traces. As we suspected, glitches caused by the XOR gate lead to data dependent power consumption and eventually led the XOR based DBD circuit to reveal the correct key. The DPA result on XOR based DBD circuit is shown in Figure 6.15.

As discussed in Section 6.4.3, a MUX based DBD minimises the glitches introduced by the DBD methodology. And as expected, MUX based DBD circuit did not

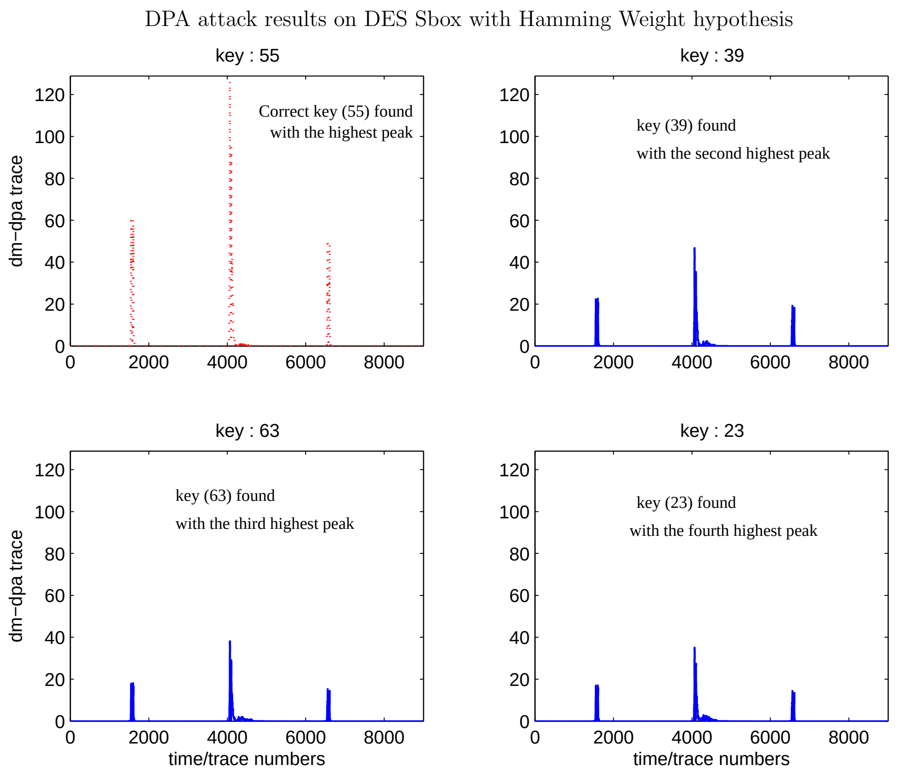


Figure 6.15: DPA result on XOR based DBD circuit

reveal the correct key in 300,000 traces. This is at least 300 times improvement on single rail circuit. The DPA result on MUX based DBD circuit is shown in Figure 6.16.

FPGA based DPA Results

As previously discussed, our FPGA flow had a few limitations. In particular, we had to drive IO ports to see any difference in power consumption. To achieve a fully balanced design, these IO routes has to be duplicated as well. Evaluating backend duplication on FPGAs including balanced IO routes can be an extension of this work. Yu and Schaumont [113] reported a successful implementation of backend duplication on FPGA. Although the design they duplicated is different, they none the less duplicated a design in much the same way as our proposal. As

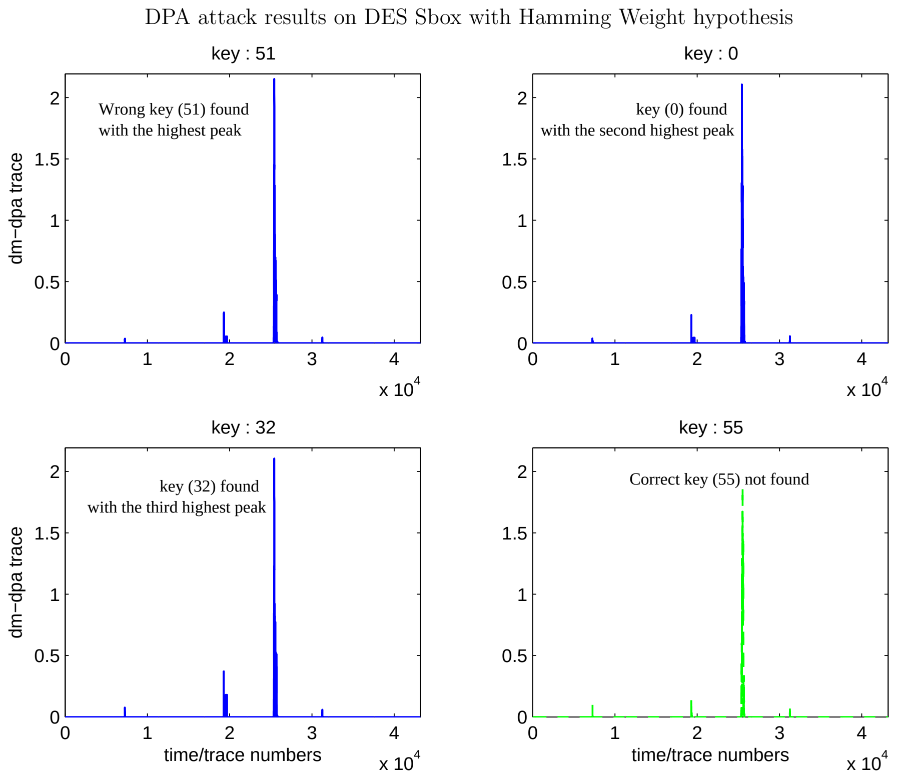


FIGURE 6.16: DPA result on MUX based DBD circuit

our method is similar, once a design is properly routed, its DPA resistance will increase.

6.6 Summary

We have shown that coupling capacitance between dual rail nets can cause routing imbalances. To address this, we have proposed a new method, called divided backend duplication. We have shown that the divided backend duplication method can be applied to get a balanced dual rail design in both ASICs and FPGAs and that it offers a significant improvement in balancing routing capacitance compared to previous methods. Divided backend duplication has an area overhead of around 3.3 times compared to an unprotected single rail design. XOR based divided backend duplication method is show to have issues with glitches and can lead to a successful DPA attack. MUX based divided backend duplication is introduced as a solution and it is shown that the MUX based divided backend duplication will offer at least a 300 times improvement in the number of traces required for a successful DPA attack.

Chapter 7

Randomisation Countermeasures

7.1 Introduction

Randomisation countermeasures in the context of power analysis attacks mean randomly varying one or more factors that affects the power consumption of the cryptographic device under attack, there-by preventing the power analysis attack. In randomisation countermeasures, the side channel information is still leaked. But as a result of randomisation, this information is embedded in unwanted power consumption (noise). Suppose randomisation occurs during the execution of a block cipher, then all the power traces measured by the attacker are not correlated to the intermediate results and this decreases the correlation of the intermediate results and the power consumption. As a result randomisation countermeasures increases the noise in the measured power consumption of the cryptographic device.

However, adding noise in the power consumption to a point where DPA will be unsuccessful is not trivial. For example, adding extra logic to run in parallel will not prevent DPA. Consider the attack in Chapter 4 on AES circuit shown in Figure 4.1. The DPA attack was on the output of Sbox, where there were 7 other identical Sboxes running in parallel to the one chosen in the attack. Even then the attack was successful. This indicates that adding similar logic to run in parallel, although increases noise in the power consumption, doesn't prevent DPA.

$$P_{dyn} = \alpha\ C_{load} V_{dd}^2\ f \tag{7.1}$$

Dynamic power consumption is given by the Equation 3.1. These variables are supply voltage (V_{dd}), frequency (f), switching activity (α) and load capacitance (C). Switching activity, α is data dependent. In this chapter we explore solutions to DPA that vary one or more of these variables. In Section 7.2 we discuss random dynamic voltage and frequency scaling as a DPA countermeasure. This countermeasure varies f and V_{dd} to achieve randomisation. In Section 7.2.1 we evaluate RDVFS and present its shortcomings. In Section 7.3 we present a different randomising countermeasure, random charging. This countermeasure varies N and C to achieve randomisation. In Section 7.3.1 we propose enchantments to random charging countermeasure and present DPA results from simulations. Finally we conclude the chapter.

7.2 Random Dynamic Voltage and Frequency Scaling as DPA Countermeasure

Yang *et al.* [112] proposed dynamic voltage and frequency scaling as a countermeasure to prevent DPA by altering voltage and frequency randomly, thus reducing the correlation of input data to the power consumed. We call the technique used in [112] random dynamic voltage scaling (RDVFS) to avoid confusion with normal dynamic voltage and frequency scaling (DVFS) technique [114]. The difference between RDVFS and DVFS is that RDVFS randomly scales voltage and frequency to randomise the power consumption, whereas DVFS scales voltage and frequency to save power consumption. In both DVFS and RDVFS the voltage and frequency $\{V_{dd}, f\}$ pairs are same, i.e, if f is changed V_{dd} is changed accordingly. The main strength of RDVFS as a DPA countermeasure is the difficulty in predicting key hypothesis due to variation of frequency, as it is difficult to predict when internal signals change and thus making it difficult to mount DPA.

The main advantage of RDVFS as a DPA countermeasure is that it does not need the cryptographic algorithm to be altered nor its implementation design flow. As RDVFS does not process random data (like masking countermeasure) nor includes complementary logic styles (like transistor-level countermeasures) to prevent DPA, its area and power overhead are less. However Yang *et al.* [112] did not verify the effectiveness of RDVFS by implementing a DPA attack.

7.2.1 Evaluation of RDVFS

Dynamic voltage and frequency scaling (DVFS) is an effective approach to reduce energy [114]. DVFS utilises the fact that the power P is directly proportional to the clock frequency f and to the square of the supply voltage V_{dd}. $P \propto f \cdot V_{dd}^2$. In DVFS, scaling of supply voltage and clock frequency takes place dynamically to adjust to performance demand. Each such power-performance mode has a $\{V_{dd}, f\}$-pair, which are predetermined. It is important to note that frequency and voltage are both changed together as a pair and both these values are related. Yang *et al.* [112] proposed to randomly vary voltage V_{dd} and frequency f to prevent DPA (RDVFS). This technique is similar to DVFS in that both vary voltage and frequency dynamically. Except that the RDVFS aim is to prevent DPA whereas the DVFS aim is to reduce energy.

DVFS has been generally used with microprocessors to reduce overall power consumption, in the DPA context, RDVFS as a DPA countermeasure is of interest to both microprocessors and ASICs. In order to check the effect of RDVFS on DPA, we implemented RDVFS on an AES Sbox. Although our circuit is trivial when compared to a microprocessor, it enables us to check the effectiveness of RDVFS as a DPA countermeasure. The $\{V_{dd}, f\}$-pairs have been arbitrarily chosen to bring in randomness. We assumed frequency and voltage change values instantly and are modelled as piecewise linear source in our SPICE simulations. We implemented a DPA attack as discussed in Chapter 4 and found that we could not find the key.

Most of the circuits used today are sequential, employing flip-flops and latches, i.e, circuit operation is synchronised to a single clock pulse. Thus at the rising (or falling) edge of a clock pulse, there will be a burst of operation (transistors switching) which will settle down toward the end of the clock period. Current consumed at the rising edge of clock pulse will thus be higher and goes down along with switching activity. This can be clearly observed in the Figure 7.1. From this it is clear that a change in frequency can be easily detected by observing the power consumption trace. Based on this, we observed the power consumption trace of Sbox employing RDVFS and recorded the circuit frequency for each input applied. By knowing the frequency, we found the voltage by looking at the voltage frequency $\{V_{dd}, f\}$-pairs. We changed our hypothetical power consumption to reflect the changes in frequency and voltage. With this new hypothetical power consumption we performed DPA attack on the same circuit, this time our attack was successful. Although we could not find an automated way to determine frequency from power consumption, this experiment shows that it is possible to implement DPA attack on systems employing the RDVFS countermeasure, where frequency and voltage values are related.

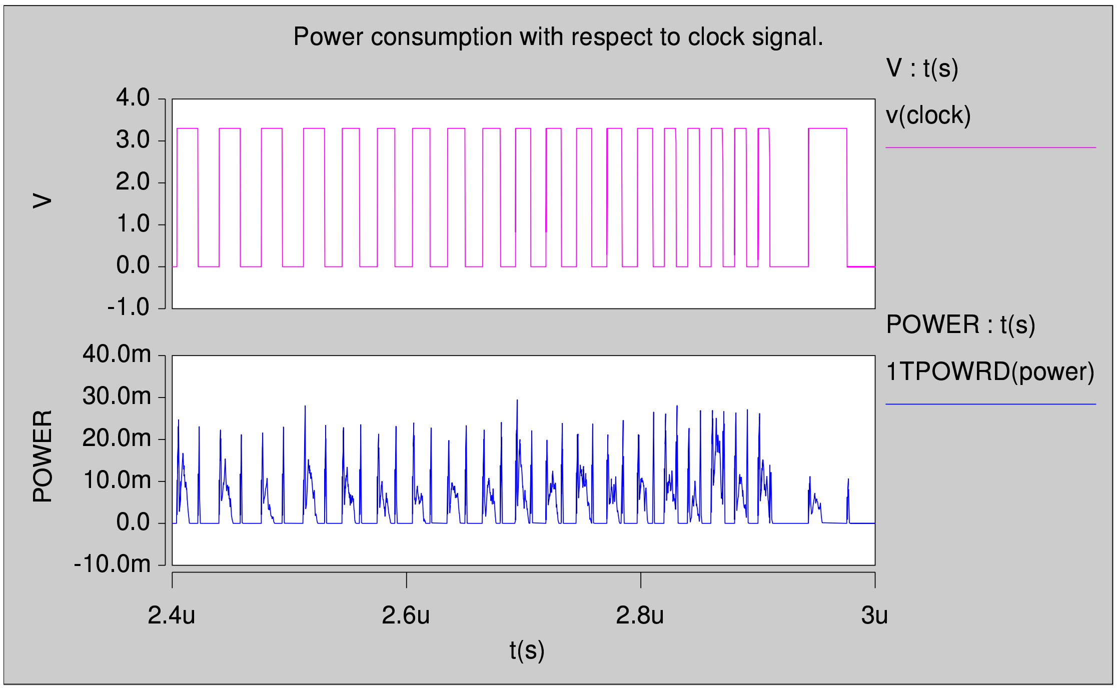
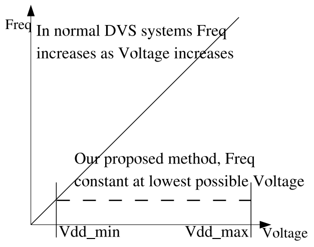


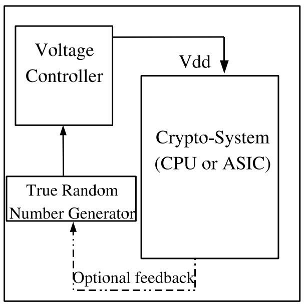
Figure 7.1: Power consumption with respect to clock pulse

7.2.2 Random Supply Voltage Variation as DPA countermeasure

As discussed in the previous section, countermeasures that depend on varying the frequency are susceptible to DPA. Since frequency is easily detectable, one approach to overcome this would be to randomly change the supply voltage while keeping the frequency constant, such that the circuit is operational under all possible supply voltages as shown in 7.2(a). A simplified block diagram of the proposed method is shown in Figure 7.2(b). The additional blocks required are a true random number generator (RNG) and a voltage controller. RNGs already exists for secure smart card applications and are not additional overhead. A voltage controller is the only additional block required for this countermeasure. Benefit in this type of countermeasure is that it can be applied to a custom ASIC or a general micro controller, without modification to the algorithm or its design flow (unlike masking countermeasures or gate level countermeasures). The main restriction of our proposed method is that the attacker should not have access to any of these blocks directly, i.e, if the connection between RNG and voltage controller is cut off, then there would be no randomness in the power consumption.



(a) Voltage and frequency relationship.



(b) Block diagram of our proposed method.

FIGURE 7.2: Our proposed method.

As we propose to vary the voltage, the maximum limit of V_{dd} (V_{dd_max}), the minimum limit of V_{dd} (V_{dd_min}) and the frequency of change of V_{dd} (dvs_rate) affect the DPA result. This section discusses these parameters and their effect on DPA. For our countermeasure to work effectively, the change in power consumed ($\delta_{voltage}$) due to a change in voltage (V_{dd}) should be close to change in power consumed ($\delta_{switching}$) due to a change in input (or switching activity). This is explained below.

Let [In_1, In_2, In_3, In_4, In_5, In_6] be a set of input vectors and [$P1_1$, $P1_2$, $P1_3$, $P1_4$, $P1_5$, $P1_6$] the power consumed per input at voltage V_{dd1} and [$P2_1$, $P2_2$, $P2_3$, $P2_4$, $P2_5$, $P2_6$] be the power consumed for the same inputs at voltage V_{dd2} and [$P3_1$, $P3_2$, $P3_3$, $P3_4$, $P3_5$, $P3_6$] for V_{dd3}. For a constant V_{dd}, the attacker can easily correlate the hypothetical power model and the actual power consumption to determine the key. But if the voltage was varied after input In_2 and In_4, then the resultant power consumption would be [$P1_1$, $P1_2$, $P2_3$, $P2_4$, $P3_5$, $P3_6$] (assuming a change in supply voltage would take much less time when compared to the time to process each input). This would significantly reduce the correlation between input data and power consumption, as the difference in $P1_2 - P1_3$ is not same as $P1_2 - P2_3$. This can be seen as introducing randomness in power consumption. For this countermeasure to be effective, the difference in $P1_2 - P2_3$ should be equal to $P1_1 - P1_2$ or $P1_3 - P1_4$ or $P1_4 - P2_5$ or $P1_5 - P2_6$, i.e, a change in V_{dd} should manifest itself as a change in input. But finding V_{dd1}, V_{dd2} and V_{dd3} values to satisfy the above condition would be difficult as the inputs to the system can be any value (and are usually unknown). Moreover these values cannot be generalised, as the inputs (switching activity) and the voltage range vary from system to system.

The rate of change of voltage (*dvs_rate*) should be much less than the time to process the minimum number of inputs to successfully mount a DPA attack. For our test circuit AES, the minimum number of encryption rounds required to successfully implement a DPA attack was 2500. If the *dvs_rate* is close to the above number, then the attacker could simply implement a DPA attack, before the randomness is introduced.

The amount of randomness in power consumption can be increased by increasing the available voltage range, i.e, if V_{dd_min} and V_{dd_max} are close to each other then the amount of randomness is less and if this range is more, randomness is more. Zhai *et al.* [114] [114] discussed the limits of voltage scaling and showed that digital circuits can work even in the sub-threshold region. Since we propose to keep the frequency to the lowest possible, selecting the V_{dd_min} will be constrained by the expected circuit speed. However to overcome such a limitation, one could increase V_{dd_max} to increase the overall range at the expense of higher power consumption.

To test the effect of voltage variation on DPA we simulated AES Sbox with V_{dd_max} of 3.3v and V_{dd_min} of 3.0v and found that the correlation strength was lowered when compared to the Sbox without any countermeasure. When we increased the range, V_{dd_max} of 3.7v and V_{dd_min} of 1.6v, the correlation strength was lowered by at least 5 times. We assumed that change in supply voltage is done instantly without any delay. While this assumption is not realistic, it quickly enables us to check the effectiveness of a countermeasure, without affecting the quality of the experiment.

Figure 7.3 shows the effect of the available V_{dd} range for *dvs_step* of 0.1v on DPA. It can be clearly seen that as this range (the number of available V_{dd} to vary) increases, the correlation of the signal decreases. However for the set of experiments we conducted, the correlation of signal was never below a point where the secret key was undetectable.

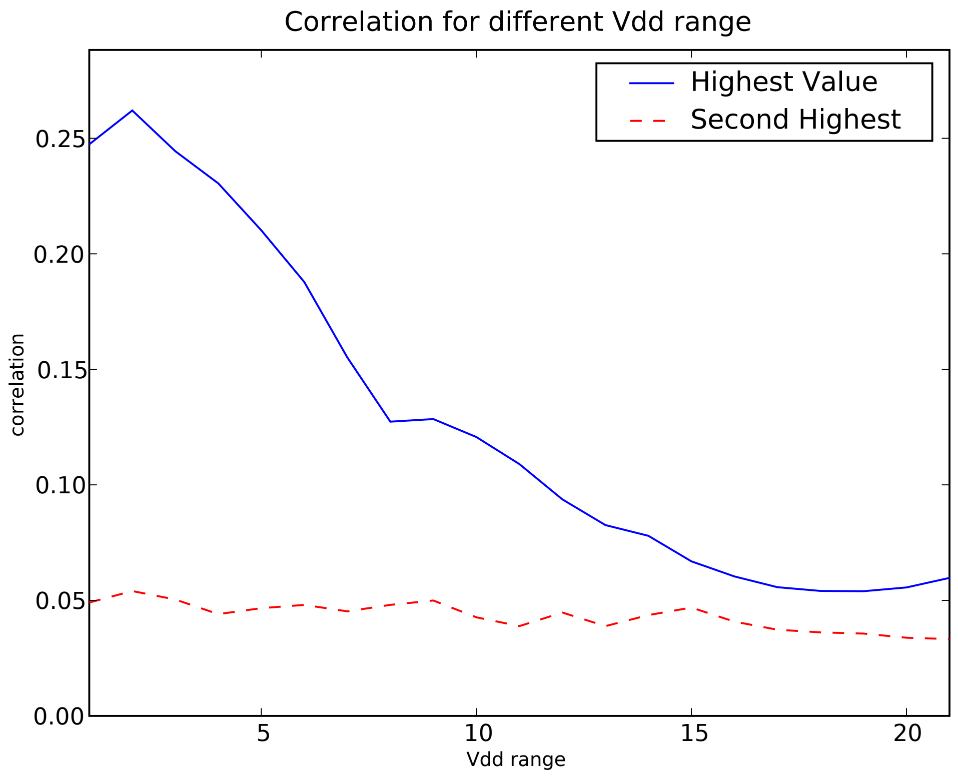


FIGURE 7.3: DPA result for different V_{dd} ranges. As the V_{dd} range widens correlation value of the highest key and the next highest key reduces

7.2.3 Key Strength of RDVFS

The key strength of RDVFS was in randomising the frequency, f, which in turn randomised the occurrence of intermediate results. For sequential designs, it is easy to extract any changes in frequency and thereby overcoming this countermeasure.

Randomising the supply voltage alone did not introduce enough randomness in the power consumption to prevent DPA. From this experiment we can conclude that, randomising one or more of the variables of the power consumption equation, Equation 7.1, will increase the noise in the power consumption and thereby reduce the correlation of the intermediate results and the power consumption. Designers using such countermeasures should ensure that no information regarding the randomisation itself, leaks. Randomising the occurrence time of calculation of the intermediate result has a better effect on DPA.

7.3 Random Pre-charging

Bucci *et al.* [8] have proposed to use random precharge logic to prevent SCA. This hardware based countermeasure is similar to the software based random interrupts countermeasure [16]. An overview of this countermeasure can be seen from Figure 7.4. The idea is to randomly precharge all the combinational gates with a random value generated from the random number generator(RNG). As register elements cannot loose their state value, a redundant register is used to load the random value. To switch between normal operation and random pre-charging a MUX is used to control the output of the compound register.

The intention of this countermeasure is to randomly precharge all the gates. In every cycle, all the gates are randomly precharged before the actual data is processed. Bucci *et al.* have reported that they could reduce the DPA correlation but could not prevent an attack.

7.3.1 Multi Cycle Random Pre-charging

The feature of the RDVFS countermeasure that prevented the DPA attack is in randomising the occurrence time of the intermediate result of the cryptographic

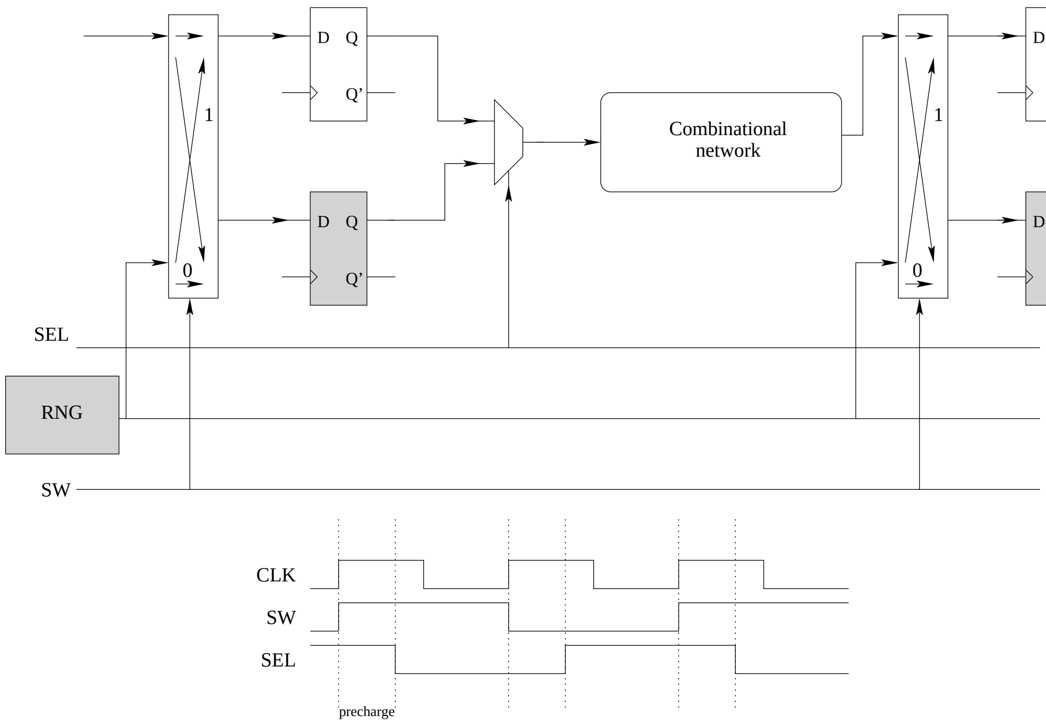


FIGURE 7.4: Random precharging countermeasure overview

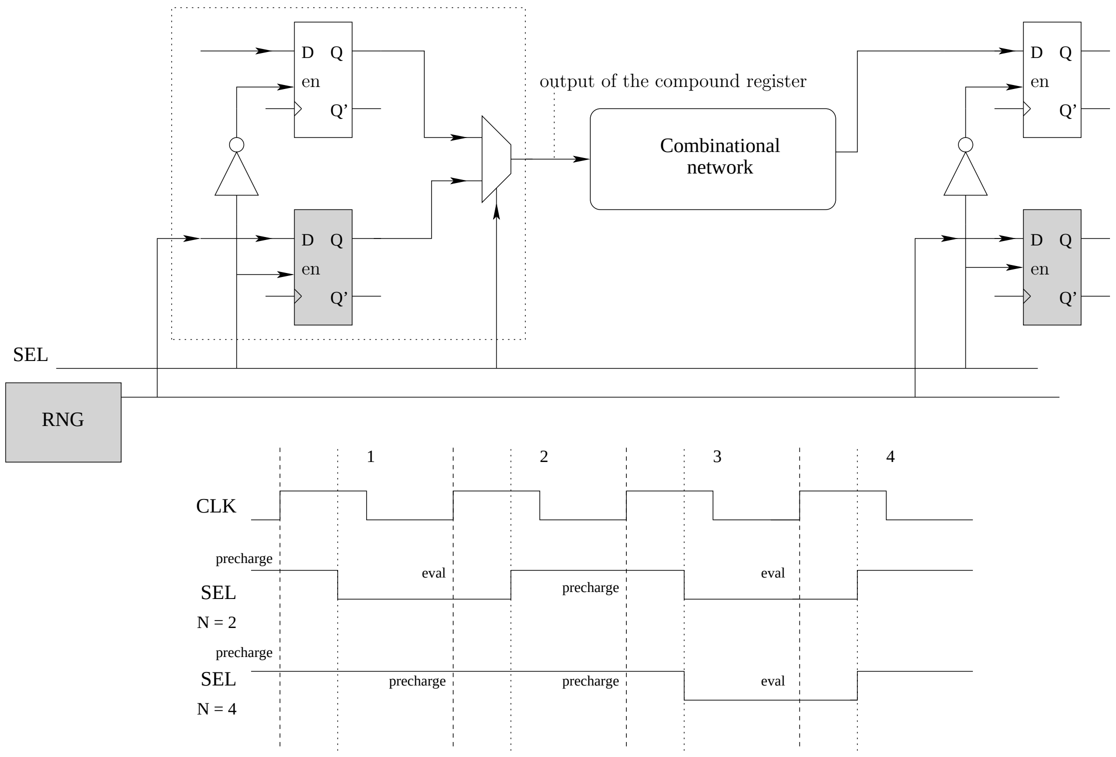


FIGURE 7.5: Multi cycle random precharging countermeasure overview

operation. As all the power traces measured by the attacker are not correlated, calculating the correlation of intermediate results and power consumption will lead to an unsuccessful attack. However using frequency to randomise the occurrence of the intermediate result is not secure as any change in frequency of the cryptographic device can be detected.

Although the random precharging countermeasure discussed in [8] only uses one *random precharge* before processing the *actual data*, it can be adapted so that the number of *random precharges* per *actual data* can be increased. The modified circuit diagram is shown in Figure 7.5. In increasing the number of *random precharges*, we increase the noise in the power consumption, there by making DPA difficult.

In the context of this countermeasure we define the following terms: every cycle of operation where the actual data is processed is referred to as an *evaluation cycle* and when random data is processed it is referred to as a *random precharge cycle.* The total number of clock cycles for processing data is the sum of *evaluation cycles* plus *random precharge cycles.* Note that the analogy of *evaluation cycle* and *random precharge cycle* is similar to the *evaluation phase* and *precharge phase* used in dual rail precharge countermeasures.

Along with the increase in the number of *random precharge cycles* per *evaluation cycle*, we can also move the occurrence of the *evaluation cycle.* Consider an example where the number of *random precharge cycles* per *evaluation cycle* is 3, so the total number of cycles to process data will be 4. Now the *evaluation cycle* can occur in any of the available 4 cycles. This behaviour can be achieved by changing the control logic to control the *sel* signal in Figure 7.4. This feature of randomising the occurrence of the *evaluation cycle* imitates the random behaviour of the RDVFS countermeasure, discussed in Section 7.2, but without its drawback.

7.3.1.1 Test Circuit and DPA Results

We wanted to see how DPA resistance changes when the number of random data cycles increases for a given cycle of normal data. For this experiment, we have used the DES sbox circuit shown in Figure 4.3 and used the ST12 design kit. The secret key used for simulation is 55. In Chapter 6, we have shown that a single rail DES sbox took 1,000 traces for a successful DPA attack (Page 156). We will use this as a reference to see the effect of the random precharge countermeasure.

The DES sbox circuit was modified in such a way that, for every clock cycle of normal operation, we will have $N-1$ of random operations. We have experimented with values of 2, 4 and 8 for N. Based on this design, we have two scenarios, one where the actual operation occurs at the same time and the other where the occurrence of actual operation is randomised based on an LFSR.

When the normal operation was fixed to the same cycle, DPA was successful and without any significant increase in the number of required traces. This is expected, as the additional *random precharge cycle* only adds noise to the power consumption, and does not actually affect the correlation between power consumption and intermediate result. results based on the case where the occurance of normal operation is randomised.

For the cases where the occurrence of normal operation is randomised, DPA was successful without any significant increase in the number of required traces as well. Upon investigating we found that there was a significant number of glitches before the MUX is switched from *actual data* to *random data* (through SEL signal in Figure 7.3.1). DPA was successful as these glitches were dependent on the current value and the previous value of the *actual data*, thereby leaking information. Consider the circuit in Figure 7.5 when $N = 2$. After the second *precharge* (3rd clock), when the SEL signal goes low, the output of the compound register is *actual data*. In the next *eval* phase (4th clock cycle), the output of the compound register

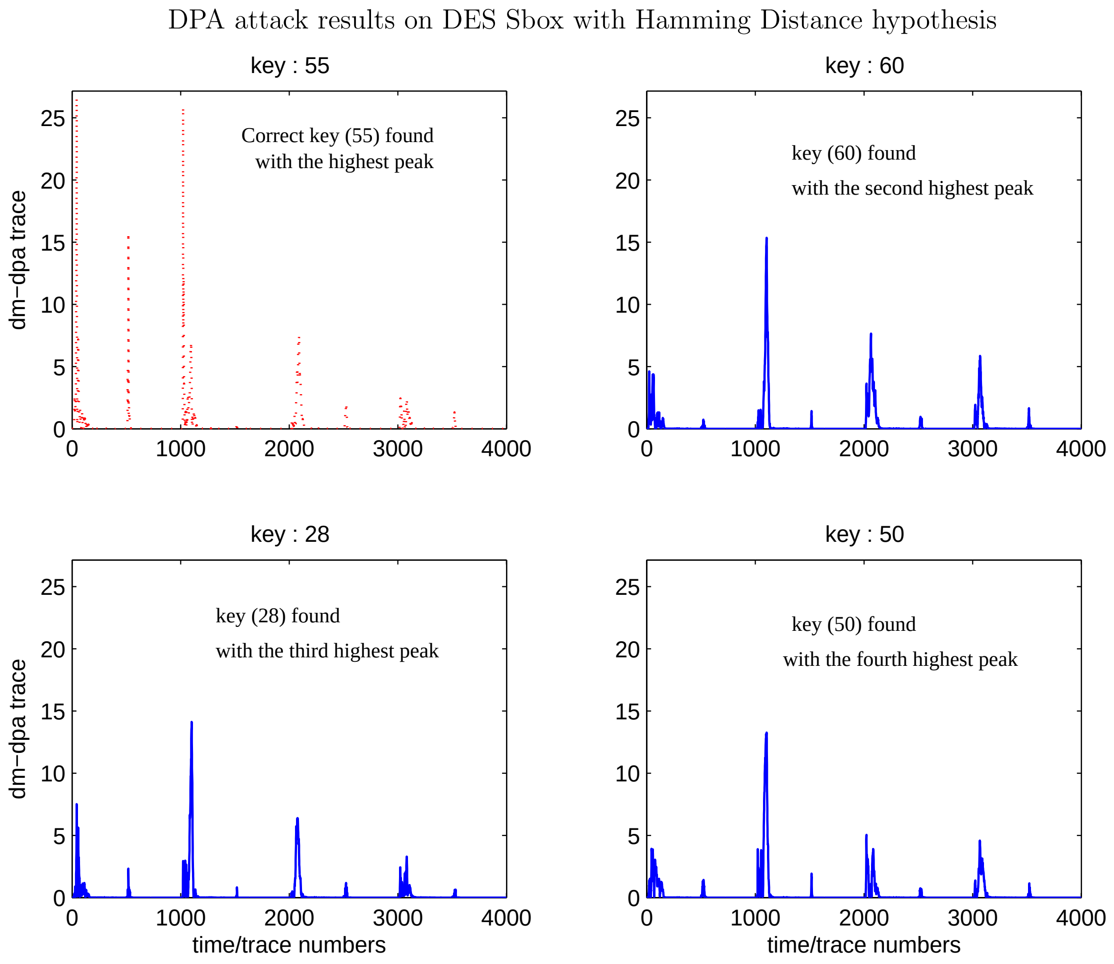


FIGURE 7.6: DPA result on a DES sbox with multi cycle random pre-charging countermeasure, where the normal operation occured in the same cycle. No of traces are 10,000.

changes from the previous value of *actual data* to the current value of *actual data*. As these transitions are dependent on *actual data*, they leak information that leads to the successful DPA attack.

7.3.1.2 Improving Multi Cycle Random Pre-charging

Since the multi cycle random precharging countermeasure, introduced above, suffers from glitches, we can prevent glitches by using two phase circuit techniques used in dual rail precharge circuits (introduced in [94] and shown in Figure 3.12).

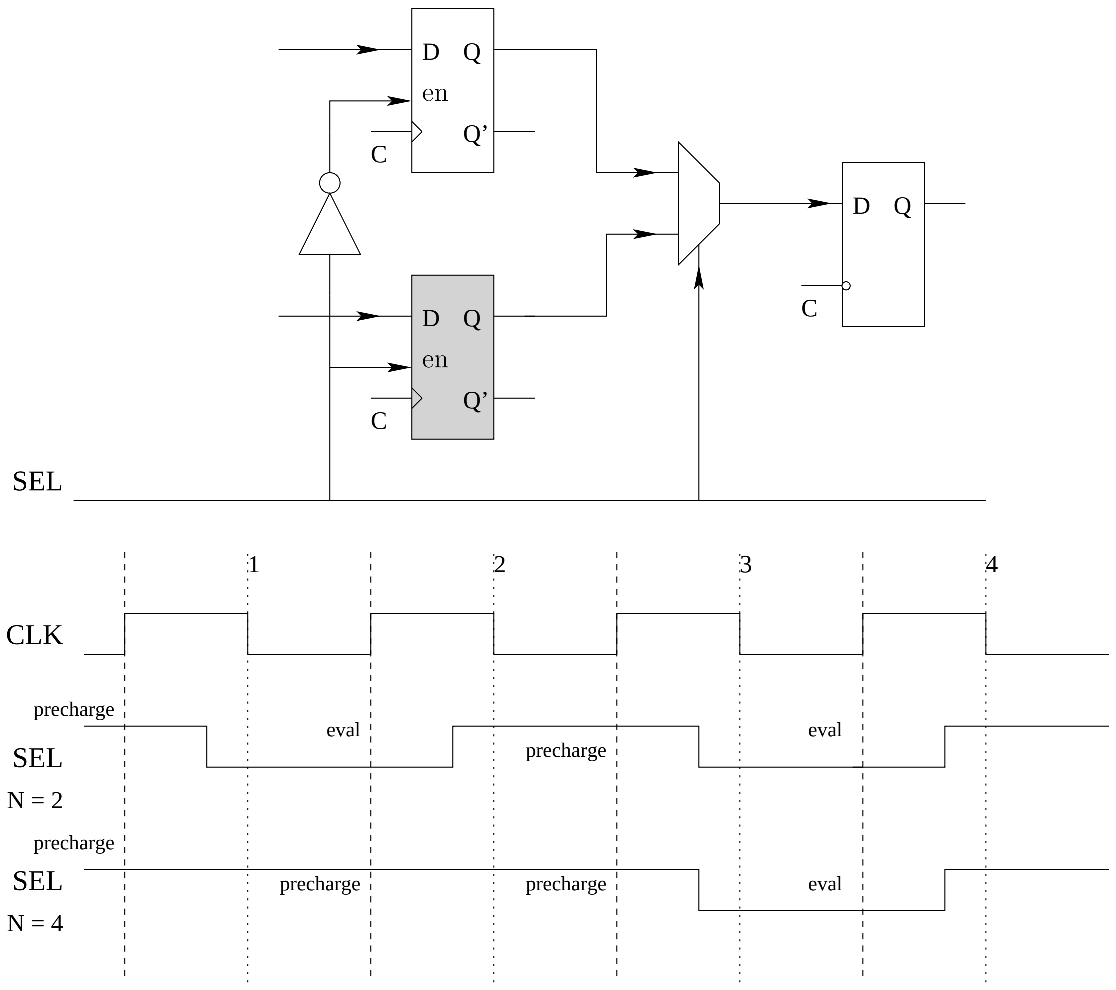


FIGURE 7.7: Multi cycle random precharging countermeasure, using latch to reduce glitches

The purpose of this is to eliminate the glitches that occur when the output of the compound register is changing from the previous value to the current value of *actual data*. To prevent these glitches, a latch can be added to the compound register in Figure 7.5, as shown in Figure 7.7. This latch is active low-enabled and is connected to the clock. Now, when the MUX is switched from *actual data* to *random data*, the output of the compound register will not change until the falling edge of the clock. This prevents the glitches from occurring at the output of the compound register.

After adding the additional latch to the test circuit, the DPA results improved. DPA results for different N are shown in Table 7.1. As N is increased, the number of traces required to find the correct key also increased.

Table 7.1: Number of traces required for successful DPA attack on DES Sbox employing multi cycle random precharge countermeasure, based on simulations.

Total number of cycles	No Of traces
1	1,000
2	20,000
4	80,000
8	250,000

7.3.1.3 Changing the DPA peak selection criteria

As discussed in Section 2.8.3, correct key is predicted by the attacker based on the correlation of the measured power consumption and hypothetical power. The key with the highest correlation is the secret key. In the multi cycle random pre-charging countermeasure, the data could be processed in any of the available clock cycles. Instead of measuring correlation for the entire operation, we used different clock cycles within the same operation. For example, when N = 4, we will have four different correlations with four possible secret keys. Then based on this, we used the maximum occurrence of a key to predict the secret key.

Table 7.2: Number of traces required for successful DPA attack on DES Sbox employing multi cycle random precharge countermeasure, using improved DPA selection.

Total number of cycles	No Of traces
1	1,000
2	20,000
4	80,000
8	150,000

Using the new DPA peak selection criteria, the improvement offered by the multi cycle random pre-charging countermeasure reduced significantly. The new results are tabulated in Table 7.2. Notice that the number of required traces when $N = 8$ reduced significantly. DPA trace plot, when $N = 8$ is shown in Figure 7.8. Notice that although the highest peak trace belongs to key 53, key 55 has more peaks per clock cycle.

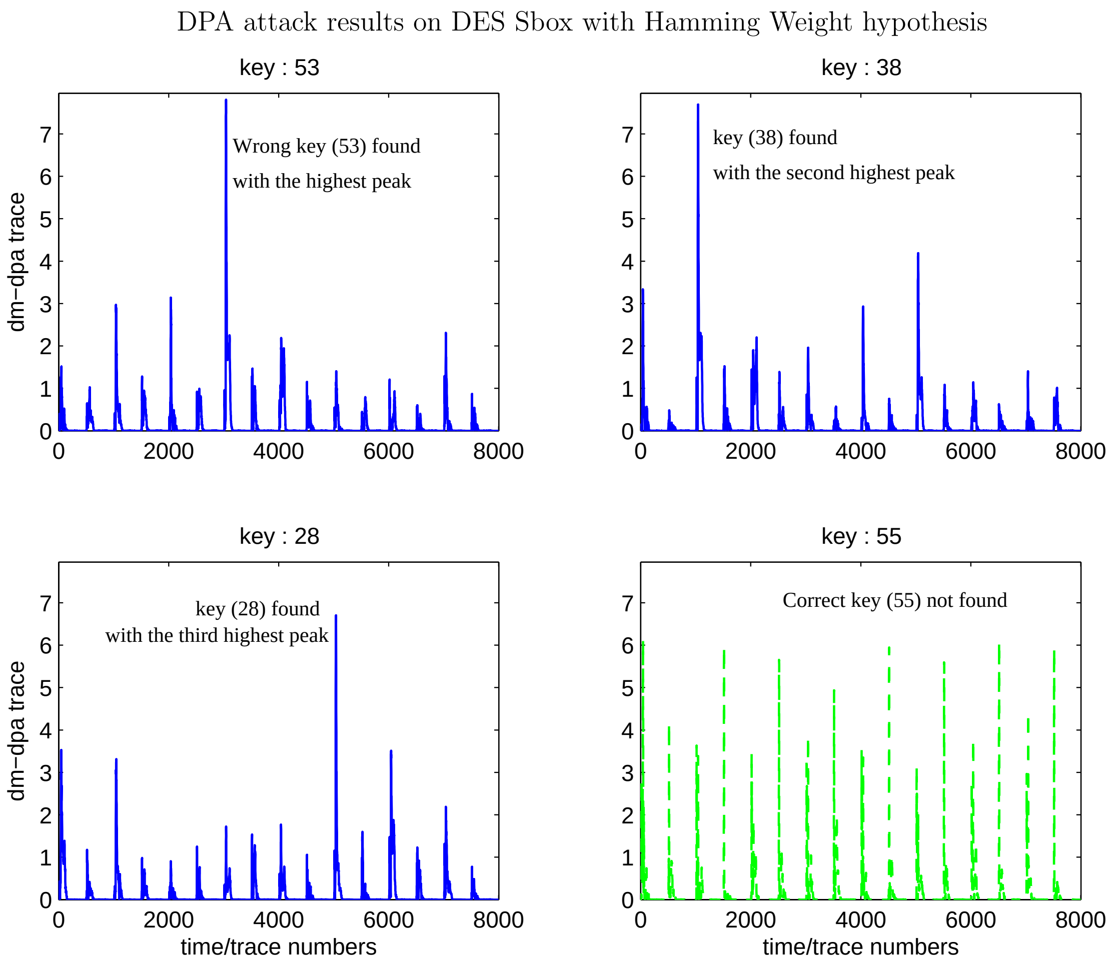


FIGURE 7.8: DPA result on a DES sbox with multi cycle random pre-charging countermeasure, when N = 8. No of traces are 300,000.

7.3.2 Area for Implementing Multi Cycle Random Pre-charging

TABLE 7.3: area for for various implementations in ST 0.12μ technology

design	area in mm^2	times increase
DES Sbox single rail	3282.2	1.0
DES Sbox Multi Cycle Random Pre-charging	5476.7	1.66

Multi cycle random pre-charging replaces a flip-flop with two flip-flops, a MUX and a latch, overheads for implementing a multi cycle random pre-charging can be easily estimated. For our DES Sbox design it increased the area by 1.66 times when compared to an unprotected single rail circuit.

7.3.3 Advantages of Multi cycle random precharging countermeasure

The main advantage of multi cycle random precharging countermeasure is that the number of random precharge cycles can be decided by the designer. This could also be done at run time. Since this is a randomisation based countermeasure, it could be combined with other DPA countermeasures.

7.3.4 Disadvantages of Multi cycle random precharging countermeasure

The main disadvantage of the multi cycle random precharging countermeasure is the additional area and performance overhead. Every flip-flop in the unmodified design needs two registers, a MUX and a latch. The decrease in performance is dependent on the number of random precharge cycles used. Another drawback of this countermeasure is in implementing it. Although the data path part of the encryption algorithm is straightforward, the control part of the algorithm needs special attention and is more difficult to automate.

7.3.5 Using Multi cycle random precharging with other countermeasure

One may think that using multi cycle random precharging with other countermeasures may lead to more secure designs. For example, an interesting study would be to combine the multi cycle random precharge countermeasure with dual rail precharge logic styles. However, area overheads for combining multi cycle random precharge and dual rail will be significantly more; both the register in multi cycle random precharge need to be converted to dual rail, which will be four times each register, a total of eight times the total registers plus twice the area for combinational logic. This overhead of the combined multi cycle random precharge and dual rail will be more when compared to other DPA countermeasures, for example MDPL, WDDL + fat wire, path switching or divided backend duplication.

7.4 Summary

In this chapter we evaluated randomisation based solutions against DPA. From Chapter 4 we have seen that similar hardware blocks running in parallel will only increase the number of required traces and not prevent DPA.

We have discussed the limitations of RDVFS as a countermeasure for DPA. The operating frequency is detectable by monitoring glitches in the power consumption. Our experiments indicate that this information can be successfully exploited by a DPA attacker and it severely compromises the effectiveness of the proposed RDVFS countermeasure. The feature of RDVFS countermeasure that prevented DPA attack is in randomising the occurrence time of the intermediate result of the cryptographic operation. As all the power traces measured by the attacker are not correlated, calculating the correlation of intermediate results and power consumption will lead to an unsuccessful attack.

We proposed a multi cycle random precharge countermeasure which imitates the behaviour of RDVFS. Using circuit simulations we showed that our proposed countermeasure offers some resistance towards DPA, but at the expense of performance. This performance and DPA resistance trade off can be decided by the designer. For a DES Sbox circuit, implementing this countermeasure has an area overhead of 1.6 times. However this countermeasure has a weakness, in that, if the attacker knows the type of countermeasure used, then he/she can look for peaks per clock cycle rather than a single correlation peak for the entire trace, bypassing the countermeasure.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this thesis we have reviewed topics related to side channel analysis in general and DPA attacks in particular along with DPA countermeasures. DPA is quite effective at breaking an implementation of the cryptographic algorithm. DPA is successful at breaking cryptographic algorithm because it relies on the data dependent power consumption of the implementation of the cryptographic algorithm. We have demonstrated the ability of DPA on unprotected implementations of DES Sbox, AES Sbox and AES designs using our simulation based and FPGA based DPA setup.

Several countermeasures have been proposed to prevent DPA. Of these, dual rail precharge countermeasures try to eliminate the data dependent power consumption. Algorithmic masking countermeasures aim to randomise power consumption by using a random mask with the intermediate results. These masking schemes need to take special care of non-linear functions often found in block ciphers. However it is shown that algorithmic masking implementations leak data dependent in power side channel via glitches at the output of combinational logic gates

[42]. Randomisation countermeasures such as [8, 112] can only increase the number of required traces by a margin. Gate-level masking works in a similar way to algorithmic masking, i.e, each logic gate now has twice as many inputs and outputs as before, the extra ports are for the mask. Gate-level masking also suffer from glitches [41]. A masking scheme which prevent glitches, called MDPL has been proposed [69] and is currently the only known secure gate-level masking scheme against DPA, however it has an area overhead of almost 5 times. Dual rail precharge logic style countermeasures such as WDDL have been shown to be fully secure, however special care needs to be taken when routing the differential nets [95]. Fat wire [95] and backend duplication [23] have been proposed to solve the routing problem. While these have been shown to be secure against DPA, they have 3.1 and 11.8 times area overhead respectively. Moreover, these methods do not consider the coupling capacitance effect on the differential nets.

We have studied dual rail precharge logic styles and using our DPA setup confirm that without balanced routing dual rail logic styles do not offer significant protection against DPA. To work around this balanced routing issue, we proposed two countermeasure called path switching and divided backend duplication. The aim of path switching is to randomly swap the path taken by differential nets. Using circuit simulations and implementations on FPGA, we showed that dual rail circuits combined with path switching increase the number of traces required by DPA by 1664 times, at an area increase of 3 times for ASICs and 11 times for FPGAs. Because of the way path switching works, it cannot be applied to logic gates and thus cannot address attacks on combinational logic.

The aim of divided backend duplication is to split the dual rail design into two logical parts (the *true* and *false* parts); place and route one part; and duplicate the placement and routing for the second part. The benefit of this method is that both the *true* and *false* parts of the design see the same environment, including coupling capacitances. Using an ASIC implementation of DES Sbox we found that a dual rail with divided backend duplication did not disclose the correct key for up to 300,000 traces. On average divided backend duplication method has an area overhead of 3.2 times.

Randomisation countermeasures were also investigated. Although the RDVFS randomising countermeasures has low area overheads and seem like a viable option to prevent DPA, investigation using circuit simulations showed that they are vulnerable to DPA. This is because of the fact that any change in frequency and hence voltage can be easily detected by looking at spikes in current consumption of the device. As part of this investigation we found that randomising the occurrence of cryptographic operations are more effective to prevent DPA. We extended random precharging countermeasure proposed im [8] to include the previous observation, i.e, randomising the occurrence of cryptographic operation, and called it multi cycle random precharging. The aim of this countermeasure is to process N-1 random sets of data for a given set of actual data. We found that special care needs to be taken to prevent glitches when switching from random data to actual data. The number of random cycles can be decided to trade off against performance. Using circuit simulations we showed that our proposed countermeasure offers some resistance towards DPA. However this countermeasure has a weakness, in that, if the attacker knows the type of countermeasure used, then he/she can look for peaks per clock cycle rather than a single correlation peak for the entire trace.

8.1.1 Summary of Contributions

Contributions of this research are summarised below:

- Developed a simulation based an FPGA based DPA setups that can be configured for other designs.

- Developed a countermeasure, called path switching, to solve the dual rail differential routing problem.
- Developed a countermeasure, called divided backend duplication, to solve the dual rail differential routing problem, which also considers coupling capacitance.
- Developed a C++ program to aid in the transformation from a normal circuit to a dual rail precharge circuit, for path switching or divided backend duplication implementation. Scripts have also been written that automate the divided backend duplication process.
- Show that randomising the power consumption itself does not prevent DPA and that randomising countermeasures should be careful not to leak information about the randomisation method employed.

8.2 Future Work

The present thesis could be elaborated in several directions. The following paragraphs introduce some relevant areas of future research.

The path switching countermeasure described in Chapter 5 is validated using simulations and experimentally on a FPGA. Divided backend duplication described in Chapter 6 is validated using simulation. Implementing path switching (Chapter 5) and divided backend duplication (Chapter 6) countermeasures on a test chip would be helpful in further validating these countermeasures. Due to time constraints we could not implement divided backend duplication on an FPGA. However divided backend duplication can be implemented on an FPGA. As an extension of this research work, implementing divided backend duplication on an FPGA will help in validating it.

Generating noise in power consumption is the first thought that one would get to prevent DPA. As DPA is based on statistical analysis, small noise introduced does not have any effect on DPA result. The worst it can do is increase the number of power traces needed. Moreover there is significant amount of measurement noise when capturing the device power consumption, which does not effect the DPA result. This has been demonstrated in Chapter 7. The only way noise generation could prevent DPA is if the added noise is significantly greater in magnitude than the original power consumption and varies more frequently than the original power consumption. Clearly doing this will increase the overall power consumption. However if the devices original power consumption is very small, then adding noise to satisfy the above constrain would become easy. Lower power design is a well researched area. Specifically, operating a device at sub threshold voltages is not a new issue [106]. Designing an ultra low power AES circuit and analysing the effect of noise generation on DPA is an interesting research area.

Self powered smart cards is a concept we propose to prevent DPA on smart cards. Having a power source on the smart card itself eliminates the possibility of probing to measure power consumption, there by eliminating the possibility of DPA. For example, flicking a smart card to activate it could be possible. EM attacks could still be possible. However smart cards can employ a Faraday cage to prevent invasive attacks such as probing [32, 107]. A Faraday cage also prevents leakage of any EM signal thus eliminating its possibility. Micro power generators is a new area of research were power is salvaged from vibrations. Studying the feasibility of a smart card device, where power is supplied from a micro power generator is interesting. Mainly looking into details such as maximum power, average power that current micro power generators can provide and design a low power crypto-processor to meet these constrains. Ultra low power AES from the above section can be used here as well.

Power analysis attacks rely on the fact that the power consumption is data dependent. To be specific dynamic power consumption is data dependent and for process technologies up to 130 nm, Dynamic power consumption is the dominant factor in over-all power consumption. However as the transistor sizes are shrunk (process lower than 90 nm) leakage current begins to dominate. Leakage power is also data dependent, but in a different way to dynamic power. Analyses of the effect of leakage current on the DPA countermeasures discussed in this thesis would be relevant extension of work in this thesis.

Appendix A

List Of Papers

The research work in this thesis were presented and published in official proceedings of rigorously refereed conferences and a journal through the following research papers:

- Karthik Baddam and Mark Zwolinski. Path switching: a technique to tolerate dual rail routing imbalances. Design Automation for Embedded Systems, Volume 12: 207–220, 09 2008.
- Karthik Baddam and Mark Zwolinski. Divided Backend Duplication Methodology for Balanced Dual Rail Routing. In Elisabeth Oswald and Pankaj Rohatgi, editors, CHES 2008: Proceedings of the 10th international workshop on Cryptographic Hardware and Embedded Systems, pages 396–410, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-85052-6.
- Karthik Baddam and Mark Zwolinski. A Dual Rail Circuit Technique to Tolerate Routing Imbalances. In Proc. of Second International Workshop on Embedded Systems Security in conjunction with 7th Annual ACM International Conference on Embedded Software (EMSOFT 2007), Salzburg, Austria, October 2007.

- Karthik Baddam and Mark Zwolinski. Evaluation of Dynamic Voltage and Frequency Scaling as a Differential Power Analysis Countermeasure. In 20th International Conference on VLSI Design (VLSI Design 2007), Sixth International Conference on Embedded Systems (ICES 2007), Bangalore, India, pages 854–862. IEEE Computer Society, January 2007. ISBN 0-7695-2502-4.

Appendix B

DPA Results on our FPGA Setup

This chapter contains the DPA results by using difference of mean correlation analysis on our FPGA setup.

B.1 DES Sbox

DPA results from various DES Sbox implementation are presented here.

B.1.1 DES SBox with Hamming Weight Hypothesis and Partition Function of 2

TABLE B.1: Different number of traces and the subkey with highest correlation value for unprotected DES Sbox using Hamming weight hypothesis and partition function of 2

number of traces	**subkey with highest correlation**	**correlation value**
1	63	0.8292082
2	63	0.7691006
3	63	0.6985500

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
4	63	0.7313769
5	63	0.7490733
6	55	0.7462826
7	48	0.7329142
8	61	0.7508598
9	48	0.7283354
10	61	0.7489878
11	28	0.7305681
12	25	0.7325509
13	61	0.6500452
14	61	0.6204740
15	61	0.6880193
16	24	0.5866135
17	0	0.5772402
18	0	0.5920610
19	0	0.6072298
20	0	0.5861466
21	0	0.5769913
22	24	0.5878594
23	41	0.5269809
24	41	0.5075252
25	58	0.4534925
26	58	0.4787796
27	58	0.4731103
28	24	0.4290293

Continued on next page. . .

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
29	24	0.4449944
30	24	0.4236937
31	24	0.4518262
32	58	0.5266732
33	58	0.5139535
34	58	0.5070974
35	58	0.5179328
36	58	0.5007089
37	58	0.4798657
38	58	0.4905086
39	58	0.4523649
40	58	0.4396411
41	40	0.3395542
42	24	0.3512571
43	58	0.4338010
44	58	0.4603610
45	58	0.4542913
46	58	0.4668437
47	58	0.4407149
48	58	0.4294990
49	58	0.4245436
50	58	0.4286150
51	58	0.4063285
52	58	0.4001933
53	58	0.3857235

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
54	37	0.3919194
55	37	0.3998409
56	37	0.4243392
57	37	0.4227895
58	37	0.4074173
59	37	0.4186861
60	37	0.4046105
61	37	0.4040150
62	37	0.3870760
63	2	0.3602313
64	2	0.3535247
65	2	0.3590095
66	2	0.3734985
67	2	0.3749662
68	2	0.3683322
69	2	0.3847299
70	2	0.3947381
71	2	0.4121739
72	2	0.4280438
73	2	0.4128306
74	2	0.4062371
75	2	0.3902922
76	2	0.4051390
77	2	0.3882840
78	2	0.3823746

Continued on next page. . .

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
79	2	0.3683599
80	2	0.3733133
81	2	0.3677845
82	2	0.3666974
83	2	0.3608207
84	2	0.3520156
85	2	0.3479559
86	2	0.3610630
87	2	0.3696622
88	2	0.3683901
89	2	0.3169535
90	2	0.3117577
91	2	0.2971890
92	2	0.2959409
93	2	0.2827994
94	42	0.2907343
95	42	0.3090416
96	42	0.3150326
97	42	0.3169208
98	42	0.3208259
99	42	0.3166802
100	42	0.3097727
101	42	0.2948093
102	17	0.2850874
103	17	0.2817122

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
104	17	0.2691006
105	17	0.2828432
106	17	0.2729809
107	17	0.2684930
108	17	0.2660278
109	42	0.2741175
110	42	0.2648972
111	42	0.2585568
112	42	0.2567575
113	63	0.2558955
114	45	0.2662445
115	63	0.2680096
116	63	0.2659681
117	45	0.2628239
118	17	0.2654250
119	63	0.2724655
120	63	0.2766803
121	55	0.2754703
122	55	0.2688088
123	55	0.2723651
124	55	0.2799802
125	55	0.2912384
126	55	0.2853073
127	55	0.2842662
128	55	0.2817384

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
129	55	0.2922097
130	55	0.2907432
131	55	0.2971829
132	55	0.3041807
133	55	0.3022913
134	55	0.3117246
135	55	0.3087922
136	55	0.3089722
137	55	0.3096783
138	55	0.3199515
139	55	0.3274661
140	55	0.3318089
141	55	0.3186324
142	55	0.3173617
143	55	0.3162827
144	55	0.3221733
145	55	0.3261251
146	55	0.3186804
147	55	0.3177949
148	55	0.3090269
149	55	0.3064626
150	55	0.3109862
151	55	0.3201605
152	55	0.3228353
153	55	0.3323382

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
154	55	0.3185710
155	55	0.3196457
156	55	0.3116354
157	55	0.3177347
158	55	0.3118588
159	55	0.3154340
160	55	0.3105639
161	55	0.3049207
162	55	0.3082675
163	55	0.2997649
164	55	0.3038635
165	55	0.3071916
166	55	0.3083748
167	55	0.3049924
168	55	0.3161179
169	55	0.3151687
170	55	0.3244238
171	55	0.3344271
172	55	0.3227794
173	55	0.3293190
174	55	0.3229689
175	55	0.3168854
176	55	0.3155612
177	55	0.3082075
178	55	0.3019426

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
179	55	0.2994902
180	55	0.3059745
181	55	0.3075860
182	55	0.3182279
183	55	0.3199289
184	55	0.3142439
185	55	0.3047627
186	55	0.3053366
187	55	0.3078247
188	55	0.3002016
189	55	0.2991713
190	55	0.2923999
191	55	0.2868119
192	55	0.2875825
193	55	0.2824718
194	55	0.2768689
195	55	0.2714655
196	55	0.2772256
197	55	0.2711063
198	55	0.2789289
199	55	0.2779502
200	55	0.2824231
300	55	0.2278576
400	33	0.2049206
500	55	0.1875466

Continued on next page...

Table B.1 – continued from previous page

number of traces	subkey with highest correlation	correlation value
600	55	0.2201948
700	55	0.2115890
800	55	0.2022365
900	55	0.1968179
1000	55	0.2027333

B.1.2 Dual Rail DES SBox with Hamming Weight Hypothesis and Partition Function of 2

Table B.2: Different number of traces and the subkey with highest correlation value for dual rail DES Sbox using Hamming weight hypothesis and partition function of 2

number of traces	subkey with highest correlation	correlation value
1	63	1.0996550
2	62	1.1134000
3	62	1.1808575
4	62	1.2269280
5	62	1.2331317
6	61	1.2370471
7	61	1.2690600
8	61	1.2816600
9	61	1.3024010
10	61	1.3234555
11	61	1.3044658

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
12	58	1.2971662
13	58	1.3174314
14	58	1.3200613
15	61	0.8273571
16	21	0.7738066
17	21	0.7446711
18	58	0.6550473
19	58	0.6334789
20	58	0.6351333
21	6	0.5773066
22	58	0.5636440
23	58	0.5478825
24	58	0.5259296
25	33	0.4471030
26	10	0.4462181
27	10	0.4413380
28	21	0.4424980
29	58	0.4426489
30	58	0.4468327
31	10	0.4455617
32	10	0.4390971
33	10	0.4325060
34	10	0.4398874
35	10	0.4315207
36	10	0.4288170

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
37	10	0.4272609
38	10	0.4404577
39	10	0.4439545
40	10	0.4391842
41	58	0.3658384
42	58	0.3836109
43	10	0.3590090
44	10	0.3573073
45	58	0.3492804
46	2	0.3493464
47	2	0.3531632
48	58	0.3330213
49	58	0.3353233
50	58	0.3400278
51	58	0.3371864
52	58	0.3418779
53	58	0.3362966
54	58	0.3380066
55	58	0.3403017
56	58	0.3364773
57	58	0.3380782
58	2	0.3354842
59	2	0.3302099
60	2	0.3384920
61	58	0.3249585

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
62	2	0.3231711
63	2	0.3207500
64	2	0.3240276
65	58	0.3227433
66	58	0.3340148
67	58	0.3227603
68	58	0.3260910
69	58	0.3269573
70	58	0.3382455
71	58	0.3480255
72	58	0.3461678
73	58	0.3537665
74	58	0.3462475
75	58	0.3499349
76	58	0.3509914
77	58	0.3211006
78	58	0.3222614
79	58	0.3238817
80	58	0.3191816
81	58	0.3103211
82	10	0.3048258
83	10	0.3058306
84	10	0.3002149
85	10	0.3033637
86	10	0.3009436

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
87	58	0.2959439
88	10	0.2871476
89	10	0.2838136
90	58	0.2720290
91	58	0.2712510
92	58	0.2783271
93	58	0.2794020
94	58	0.2744490
95	58	0.2817708
96	58	0.2860841
97	58	0.2739285
98	58	0.2742464
99	58	0.2515998
100	58	0.2471742
101	58	0.2484123
102	58	0.2402844
103	58	0.2421044
104	58	0.2398702
105	58	0.2368598
106	57	0.2315591
107	58	0.2296323
108	58	0.2297843
109	58	0.2328989
110	58	0.2351773
111	58	0.2366447

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
112	58	0.2370878
113	58	0.2215936
114	58	0.2158777
115	58	0.2185005
116	58	0.2212781
117	58	0.2223946
118	58	0.2157784
119	58	0.2067332
120	58	0.2079482
121	58	0.2059622
122	58	0.2082384
123	58	0.2072124
124	27	0.2040852
125	58	0.2077976
126	58	0.2035863
127	58	0.2028607
128	58	0.2035780
129	58	0.2046966
130	10	0.2102354
131	10	0.2034356
132	10	0.2041123
133	10	0.2076704
134	10	0.2061765
135	10	0.2084115
136	10	0.2075114

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
137	10	0.2095076
138	10	0.1983820
139	52	0.2006485
140	52	0.2007623
141	52	0.1995115
142	52	0.1983431
143	52	0.1991589
144	52	0.1938385
145	52	0.1949769
146	52	0.1905804
147	52	0.1878787
148	52	0.1814923
149	52	0.1770016
150	38	0.1776262
151	44	0.1786649
152	38	0.1806049
153	52	0.1808559
154	38	0.1791485
155	38	0.1803924
156	38	0.1803165
157	38	0.1787059
158	38	0.1787600
159	38	0.1751800
160	38	0.1772391
161	44	0.1776960

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
162	38	0.1745706
163	38	0.1734031
164	61	0.1694919
165	61	0.1751295
166	61	0.1762530
167	61	0.1753698
168	61	0.1733539
169	61	0.1721847
170	61	0.1745377
171	61	0.1721983
172	61	0.1684038
173	38	0.1740647
174	38	0.1722004
175	38	0.1741360
176	61	0.1745140
177	61	0.1737525
178	61	0.1728716
179	61	0.1755634
180	61	0.1758721
181	61	0.1730145
182	61	0.1704748
183	38	0.1767282
184	38	0.1770154
185	38	0.1757883
186	38	0.1705510

Continued on next page. . .

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
187	61	0.1731293
188	61	0.1747503
189	61	0.1711764
190	61	0.1737293
191	61	0.1719528
192	61	0.1692074
193	61	0.1668613
194	61	0.1663988
195	61	0.1634031
196	61	0.1577890
197	61	0.1562021
198	61	0.1587199
199	61	0.1577525
200	61	0.1561727
201	61	0.1546501
202	61	0.1517943
203	61	0.1508899
204	37	0.1512772
205	37	0.1488454
206	38	0.1510153
207	38	0.1510507
208	38	0.1492602
209	38	0.1491916
210	38	0.1488466
211	38	0.1529564

Continued on next page. . .

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
212	38	0.1526853
213	38	0.1498246
214	38	0.1484952
215	38	0.1494744
216	38	0.1476185
217	38	0.1457808
218	61	0.1444195
219	38	0.1452075
220	38	0.1446139
221	38	0.1453367
222	38	0.1453655
223	55	0.1460800
224	55	0.1465194
225	38	0.1474651
226	38	0.1476652
227	38	0.1507334
228	38	0.1460743
229	15	0.1428934
230	15	0.1431820
231	15	0.1436468
232	15	0.1434539
233	0	0.1430526
234	0	0.1433206
235	38	0.1441868
236	0	0.1431007

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
237	55	0.1439243
238	55	0.1437912
239	15	0.1415835
240	15	0.1433444
241	38	0.1398519
242	38	0.1419969
243	38	0.1417880
244	38	0.1421146
245	38	0.1409726
246	38	0.1424090
247	38	0.1424010
248	38	0.1416582
249	38	0.1436905
250	38	0.1417497
251	38	0.1398612
252	38	0.1389282
253	38	0.1381458
254	15	0.1367075
255	15	0.1346717
256	15	0.1349070
257	37	0.1351089
258	37	0.1349176
259	37	0.1353920
260	37	0.1363465
261	37	0.1368203

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
262	55	0.1419772
263	55	0.1481859
264	55	0.1500990
265	55	0.1481533
266	55	0.1479101
267	55	0.1460811
268	55	0.1506325
269	55	0.1504068
270	55	0.1538203
271	55	0.1518938
272	55	0.1516083
273	55	0.1521128
274	55	0.1526189
275	55	0.1498188
276	55	0.1508800
277	55	0.1521995
278	55	0.1518457
279	55	0.1540432
280	55	0.1544860
281	55	0.1568385
282	55	0.1548118
283	55	0.1562953
284	55	0.1559741
285	55	0.1563889
286	55	0.1534449

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
287	55	0.1504891
288	55	0.1505402
289	55	0.1503928
290	55	0.1512638
291	55	0.1509939
292	55	0.1492252
293	55	0.1498384
294	55	0.1499670
295	55	0.1494702
296	55	0.1503224
297	55	0.1516498
298	55	0.1528798
299	55	0.1571700
300	55	0.1555019
301	55	0.1555249
302	55	0.1549690
303	55	0.1544300
304	55	0.1557553
305	55	0.1566801
306	55	0.1573852
307	55	0.1537053
308	55	0.1515119
309	55	0.1501329
310	55	0.1497264
311	55	0.1495497

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
312	55	0.1456342
313	55	0.1457615
314	55	0.1444890
315	55	0.1457997
316	55	0.1470931
317	55	0.1474820
318	55	0.1485807
319	55	0.1485524
320	55	0.1481110
321	55	0.1483184
322	55	0.1491054
323	55	0.1486985
324	55	0.1467230
325	55	0.1496658
326	55	0.1489510
327	55	0.1479699
328	55	0.1470072
329	55	0.1471628
330	55	0.1453987
331	55	0.1423396
332	55	0.1463835
333	55	0.1472212
334	55	0.1462750
335	55	0.1453037
336	55	0.1449787

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
337	55	0.1450995
338	55	0.1452927
339	55	0.1459005
340	55	0.1452947
341	55	0.1451657
342	55	0.1477512
343	55	0.1477506
344	55	0.1455993
345	55	0.1420560
346	55	0.1449242
347	55	0.1445476
348	55	0.1457562
349	55	0.1457904
350	55	0.1471237
351	55	0.1464495
352	55	0.1451547
353	55	0.1448445
354	55	0.1416047
355	55	0.1421123
356	55	0.1430852
357	55	0.1425840
358	55	0.1406678
359	55	0.1375957
360	55	0.1373789
361	55	0.1349199

Continued on next page. . .

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
362	55	0.1369257
363	55	0.1354486
364	55	0.1349177
365	55	0.1329634
366	55	0.1339634
367	55	0.1339730
368	55	0.1331395
369	55	0.1341446
370	55	0.1324318
371	55	0.1310219
372	55	0.1299621
373	55	0.1328486
374	55	0.1321852
375	55	0.1308678
376	55	0.1304502
377	55	0.1322605
378	55	0.1308172
379	55	0.1266916
380	55	0.1252851
381	55	0.1252147
382	55	0.1236888
383	55	0.1230609
384	55	0.1229974
385	55	0.1270965
386	55	0.1267258

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
387	55	0.1274721
388	55	0.1257382
389	55	0.1283159
390	55	0.1303993
391	55	0.1307480
392	55	0.1305898
393	55	0.1300707
394	55	0.1295993
395	55	0.1303035
396	55	0.1295035
397	55	0.1288091
398	55	0.1275641
399	55	0.1282679
400	55	0.1300116
401	55	0.1297331
402	55	0.1300844
403	55	0.1290046
404	55	0.1293118
405	55	0.1289551
406	55	0.1289634
407	55	0.1291461
408	55	0.1305205
409	55	0.1288290
410	55	0.1277860
411	55	0.1286714

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
412	55	0.1284225
413	55	0.1261054
414	55	0.1262038
415	55	0.1267023
416	55	0.1282393
417	55	0.1298280
418	55	0.1305640
419	55	0.1309953
420	55	0.1301646
421	55	0.1286253
422	55	0.1283781
423	55	0.1283186
424	55	0.1279716
425	55	0.1276392
426	55	0.1280225
427	55	0.1268301
428	55	0.1281303
429	55	0.1271449
430	55	0.1280950
431	55	0.1293371
432	55	0.1284008
433	55	0.1282904
434	55	0.1299416
435	55	0.1332432
436	55	0.1339561

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
437	55	0.1327022
438	55	0.1317440
439	55	0.1311138
440	55	0.1300043
441	55	0.1276054
442	55	0.1280232
443	55	0.1274192
444	55	0.1269631
445	55	0.1275720
446	55	0.1273531
447	55	0.1271970
448	55	0.1274729
449	55	0.1261572
450	55	0.1253971
451	55	0.1256688
452	55	0.1249336
453	55	0.1253212
454	55	0.1268202
455	55	0.1277675
456	55	0.1277369
457	55	0.1282542
458	55	0.1281003
459	55	0.1252742
460	55	0.1255331
461	55	0.1238343

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
462	55	0.1212329
463	55	0.1218324
464	55	0.1220213
465	55	0.1210990
466	55	0.1205266
467	55	0.1200449
468	55	0.1203562
469	55	0.1201673
470	55	0.1217500
471	55	0.1224826
472	55	0.1263315
473	55	0.1272585
474	55	0.1257991
475	55	0.1260778
476	55	0.1260937
477	55	0.1281346
478	55	0.1271719
479	55	0.1263909
480	55	0.1262450
481	55	0.1255806
482	55	0.1252319
483	55	0.1258398
484	55	0.1250345
485	55	0.1220276
486	55	0.1217035

Continued on next page. . .

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
487	55	0.1222320
488	55	0.1210998
489	55	0.1224492
490	55	0.1219056
491	55	0.1213827
492	55	0.1207231
493	55	0.1202557
494	55	0.1206149
495	55	0.1205005
496	55	0.1205463
497	55	0.1197712
498	55	0.1203851
499	55	0.1188802
500	55	0.1197846
600	55	0.1202966
700	55	0.1098941
800	55	0.1171481
900	55	0.1215631
1000	55	0.1228230
1100	55	0.1159752
1200	55	0.1252646
1300	55	0.1256369
1400	55	0.1237615
1500	55	0.1186465
1600	55	0.1166875

Continued on next page...

Table B.2 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1700	55	0.1151947
1800	55	0.1119309
1900	55	0.1118905
2000	55	0.1114829
2100	55	0.1065890
2200	55	0.1098200
2300	55	0.1122863
2400	55	0.1157294
2500	55	0.1182918
2600	55	0.1186988
2700	55	0.1220862
2800	55	0.1235627
2900	55	0.1226064
3000	55	0.1256955

B.1.3 Dual Rail Alternating Spacer DES SBox with Hamming Weight Hypothesis and Partition Function of 2

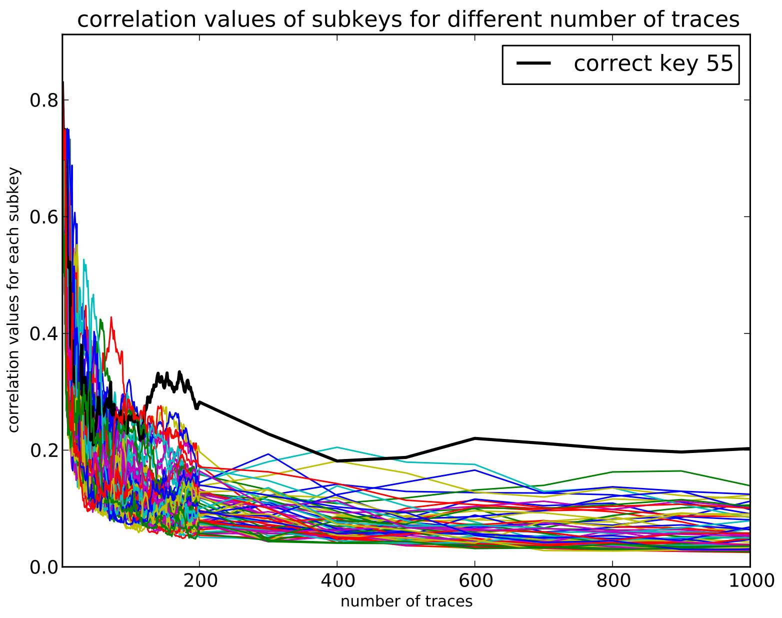


FIGURE B.1: DPA result for all encryption rounds on FPGA Implementation of DES Sbox without any countermeasure for Hamming weight hypothesis and partition function of 2

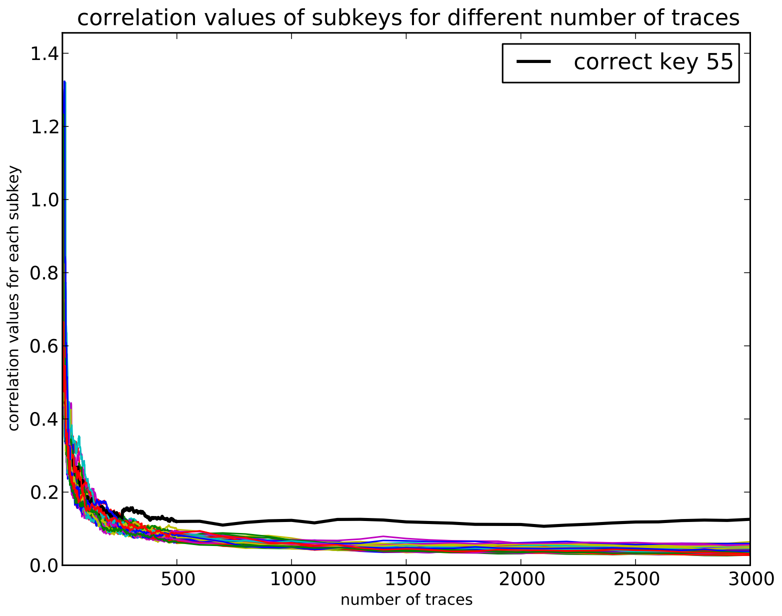


FIGURE B.2: DPA result for all encryption rounds on FPGA Implementation of dual rail DES Sbox for Hamming weight hypothesis and partition function of 2

TABLE B.3: Different number of traces and the subkey with highest correlation value for dual rail alternating spacer DES Sbox using Hamming weight hypothesis and partition function of 2

number of traces	subkey with highest correlation	correlation value
1	63	0.9628500

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
2	62	1.0604700
3	62	1.2041225
4	62	1.1392632
5	62	1.0889002
6	61	1.0362080
7	61	1.0685308
8	61	1.0547751
9	61	1.0320346
10	61	1.0457079
11	61	1.0449193
12	58	1.0765135
13	58	1.0789076
14	58	1.1061031
15	58	0.8928546
16	58	0.9024238
17	58	0.8726353
18	58	0.8795834
19	58	0.9017522
20	58	0.9143887
21	30	0.7447051
22	30	0.7075082
23	30	0.7237754
24	30	0.7295673
25	30	0.7062031
26	30	0.7228330

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
27	30	0.7464529
28	30	0.7299688
29	30	0.6969801
30	30	0.6504684
31	30	0.6124712
32	30	0.6204839
33	30	0.5940668
34	30	0.5839104
35	30	0.5812711
36	30	0.5966761
37	30	0.5731604
38	30	0.5690841
39	30	0.5853552
40	30	0.5596671
41	30	0.5347302
42	30	0.5193378
43	30	0.5356045
44	10	0.4885304
45	10	0.4358543
46	30	0.4521296
47	30	0.4658780
48	30	0.4658764
49	30	0.4529360
50	10	0.4308862
51	10	0.4336871

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
52	10	0.4377022
53	30	0.4243955
54	30	0.4383238
55	30	0.4307328
56	30	0.4246198
57	30	0.4211958
58	61	0.3873221
59	61	0.3889485
60	39	0.3816157
61	39	0.3907143
62	39	0.3918521
63	39	0.3630283
64	39	0.3559780
65	39	0.3624772
66	39	0.3747092
67	39	0.3477996
68	39	0.3550044
69	39	0.3429248
70	39	0.3351730
71	39	0.3389471
72	39	0.3428742
73	39	0.3477659
74	39	0.3442647
75	39	0.3508983
76	39	0.3534594

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
77	39	0.3468936
78	39	0.3389924
79	39	0.3175666
80	39	0.3159308
81	39	0.3125393
82	39	0.2930952
83	39	0.2876867
84	39	0.2948221
85	39	0.2994224
86	39	0.3058096
87	39	0.3008829
88	39	0.2787941
89	39	0.2802942
90	40	0.2672891
91	40	0.2686134
92	40	0.2603080
93	24	0.2526315
94	40	0.2553958
95	24	0.2521536
96	40	0.2519468
97	40	0.2512948
98	40	0.2510953
99	40	0.2537110
100	48	0.2552815
101	48	0.2676500

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
102	30	0.2740271
103	48	0.2831607
104	30	0.2783922
105	30	0.2867722
106	30	0.3028152
107	30	0.2949587
108	30	0.2913263
109	30	0.2869844
110	30	0.2845517
111	48	0.2649731
112	48	0.2760452
113	30	0.2637161
114	48	0.2622053
115	48	0.2623100
116	30	0.2457067
117	48	0.2333978
118	30	0.2352631
119	30	0.2398095
120	30	0.2295933
121	30	0.2347109
122	13	0.2309045
123	30	0.2284542
124	30	0.2224012
125	30	0.2265641
126	30	0.2290568

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
127	30	0.2288860
128	30	0.2228145
129	30	0.2258477
130	61	0.2129662
131	61	0.2130950
132	61	0.2252345
133	30	0.2136993
134	30	0.2150404
135	30	0.2163874
136	13	0.2217126
137	18	0.2228124
138	18	0.2263788
139	30	0.2177582
140	18	0.2215502
141	18	0.2300317
142	18	0.2356066
143	18	0.2397542
144	18	0.2398031
145	18	0.2359133
146	18	0.2361064
147	18	0.2415185
148	18	0.2493793
149	18	0.2552745
150	18	0.2598056
151	18	0.2639828

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
152	18	0.2632952
153	18	0.2482980
154	18	0.2408453
155	18	0.2423312
156	18	0.2412254
157	18	0.2409376
158	18	0.2299884
159	18	0.2337411
160	18	0.2302731
161	18	0.2272401
162	30	0.2088903
163	30	0.2115380
164	37	0.2101203
165	37	0.2060345
166	37	0.2003638
167	49	0.2021025
168	49	0.2032330
169	49	0.2029327
170	49	0.2005343
171	49	0.2051795
172	49	0.2096263
173	61	0.2078165
174	61	0.2102609
175	61	0.2073217
176	61	0.2036933

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
177	49	0.2052163
178	49	0.1983597
179	37	0.1996499
180	37	0.1972875
181	37	0.2016233
182	37	0.1901831
183	49	0.1882833
184	48	0.1872959
185	18	0.1883315
186	18	0.1906364
187	18	0.1886688
188	18	0.1906016
189	18	0.1931537
190	18	0.1904929
191	18	0.1864216
192	18	0.1822161
193	48	0.1839115
194	18	0.1795913
195	49	0.1790812
196	49	0.1804996
197	49	0.1753848
198	49	0.1816944
199	49	0.1832250
200	49	0.1849448
201	49	0.1889619

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
202	49	0.1914305
203	49	0.1911783
204	49	0.2000731
205	49	0.2023150
206	49	0.2051221
207	49	0.2078982
208	49	0.2097734
209	49	0.2056108
210	49	0.1996608
211	49	0.1953877
212	49	0.1980386
213	49	0.2001207
214	49	0.1952921
215	49	0.2028607
216	49	0.2058597
217	49	0.2109348
218	49	0.2037267
219	49	0.2009103
220	49	0.1958382
221	49	0.1963090
222	49	0.1922682
223	49	0.1957802
224	49	0.1955788
225	49	0.1965796
226	49	0.1892982

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
227	49	0.1870862
228	18	0.1850452
229	18	0.1874118
230	18	0.1895888
231	18	0.1899791
232	18	0.1938205
233	18	0.1925661
234	18	0.1940584
235	18	0.1944974
236	18	0.1858882
237	18	0.1818452
238	18	0.1781285
239	49	0.1781282
240	48	0.1783495
241	48	0.1818247
242	48	0.1780757
243	18	0.1766021
244	48	0.1784520
245	18	0.1788530
246	48	0.1784892
247	48	0.1805757
248	48	0.1816234
249	48	0.1842358
250	48	0.1810228
251	48	0.1809208

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
252	18	0.1806700
253	48	0.1805794
254	48	0.1801528
255	48	0.1777200
256	18	0.1767658
257	48	0.1782816
258	48	0.1818351
259	48	0.1804692
260	48	0.1816956
261	48	0.1752966
262	48	0.1790766
263	48	0.1759571
264	28	0.1761344
265	28	0.1762389
266	28	0.1736759
267	28	0.1721950
268	28	0.1743188
269	28	0.1718262
270	28	0.1743581
271	28	0.1747565
272	28	0.1796540
273	28	0.1762312
274	28	0.1748061
275	28	0.1721287
276	28	0.1701546

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
277	28	0.1727514
278	28	0.1720477
279	28	0.1732448
280	28	0.1758428
281	28	0.1682401
282	28	0.1683186
283	28	0.1659199
284	28	0.1675814
285	28	0.1660710
286	18	0.1631370
287	18	0.1697149
288	18	0.1731569
289	18	0.1771124
290	18	0.1698254
291	18	0.1715394
292	18	0.1729372
293	18	0.1697916
294	39	0.1686159
295	39	0.1684477
296	39	0.1714649
297	39	0.1670796
298	39	0.1726334
299	39	0.1739006
300	39	0.1758110
301	39	0.1734214

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
302	39	0.1763947
303	39	0.1729072
304	39	0.1687088
305	39	0.1659225
306	39	0.1656563
307	39	0.1677531
308	39	0.1619477
309	39	0.1637070
310	39	0.1656315
311	39	0.1663646
312	39	0.1665841
313	39	0.1695825
314	39	0.1644295
315	39	0.1657524
316	39	0.1643746
317	39	0.1640507
318	18	0.1619539
319	18	0.1668759
320	18	0.1654850
321	18	0.1665512
322	18	0.1691000
323	18	0.1677949
324	18	0.1627473
325	37	0.1660061
326	37	0.1669059

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
327	18	0.1667373
328	18	0.1684358
329	18	0.1724843
330	18	0.1763466
331	18	0.1745491
332	18	0.1732219
333	18	0.1698799
334	18	0.1682814
335	18	0.1702150
336	18	0.1678593
337	18	0.1664225
338	18	0.1643217
339	18	0.1636082
340	18	0.1607493
341	18	0.1583373
342	18	0.1628203
343	18	0.1607597
344	18	0.1591150
345	18	0.1610664
346	18	0.1591004
347	18	0.1611168
348	18	0.1636455
349	18	0.1622055
350	18	0.1642243
351	18	0.1652817

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
352	18	0.1652387
353	18	0.1641312
354	18	0.1646098
355	18	0.1631209
356	18	0.1615546
357	18	0.1590540
358	18	0.1575298
359	18	0.1558147
360	18	0.1581650
361	18	0.1593369
362	18	0.1594416
363	18	0.1640000
364	18	0.1624099
365	18	0.1603306
366	18	0.1591406
367	18	0.1600065
368	18	0.1609791
369	18	0.1556436
370	18	0.1558724
371	18	0.1543505
372	18	0.1523219
373	18	0.1518324
374	18	0.1530937
375	18	0.1536322
376	18	0.1516273

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
377	18	0.1535635
378	18	0.1556598
379	18	0.1574052
380	18	0.1552525
381	18	0.1503801
382	18	0.1522202
383	18	0.1555785
384	18	0.1544578
385	18	0.1564090
386	18	0.1576005
387	18	0.1547482
388	18	0.1556558
389	18	0.1540293
390	18	0.1521604
391	18	0.1531869
392	18	0.1532714
393	18	0.1531761
394	18	0.1517153
395	18	0.1492051
396	18	0.1483224
397	18	0.1505389
398	18	0.1506538
399	18	0.1519879
400	18	0.1538488
401	18	0.1562288

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
402	18	0.1587763
403	18	0.1555915
404	18	0.1538778
405	18	0.1524153
406	18	0.1498376
407	18	0.1505064
408	18	0.1501850
409	18	0.1510380
410	18	0.1495875
411	18	0.1478649
412	18	0.1464674
413	18	0.1478491
414	18	0.1483548
415	18	0.1489668
416	18	0.1471737
417	18	0.1458150
418	18	0.1468477
419	18	0.1461521
420	18	0.1466923
421	18	0.1486364
422	18	0.1435017
423	18	0.1406183
424	18	0.1410003
425	18	0.1394430
426	18	0.1380898

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
427	18	0.1394885
428	18	0.1425241
429	18	0.1444001
430	18	0.1439961
431	18	0.1479813
432	18	0.1466357
433	18	0.1448503
434	18	0.1456851
435	18	0.1427061
436	18	0.1445740
437	18	0.1466048
438	18	0.1475823
439	18	0.1481150
440	18	0.1477143
441	18	0.1432429
442	18	0.1451692
443	18	0.1421864
444	18	0.1423977
445	18	0.1451276
446	18	0.1441335
447	18	0.1427758
448	18	0.1420583
449	18	0.1422582
450	37	0.1414181
451	37	0.1425920

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
452	18	0.1411395
453	39	0.1394379
454	39	0.1420514
455	39	0.1402389
456	39	0.1413976
457	39	0.1436064
458	39	0.1445891
459	39	0.1425817
460	39	0.1404693
461	39	0.1367813
462	39	0.1352374
463	18	0.1335108
464	18	0.1337180
465	18	0.1318395
466	18	0.1309887
467	18	0.1326791
468	18	0.1288184
469	18	0.1293254
470	49	0.1260147
471	49	0.1290260
472	18	0.1282197
473	18	0.1297751
474	18	0.1344412
475	18	0.1334255
476	18	0.1311705

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
477	49	0.1304229
478	49	0.1285789
479	18	0.1280867
480	18	0.1296682
481	18	0.1304155
482	18	0.1307206
483	18	0.1317029
484	18	0.1329752
485	18	0.1326202
486	18	0.1342505
487	18	0.1357879
488	18	0.1357984
489	18	0.1366893
490	18	0.1370095
491	18	0.1341777
492	18	0.1363420
493	18	0.1369103
494	18	0.1375913
495	18	0.1354819
496	18	0.1363129
497	18	0.1338921
498	18	0.1341287
499	18	0.1348891
500	18	0.1292840
501	18	0.1295645

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
502	18	0.1312788
503	18	0.1314806
504	18	0.1293374
505	18	0.1283792
506	18	0.1271955
507	18	0.1255588
508	18	0.1239390
509	18	0.1239968
510	18	0.1243274
511	18	0.1254098
512	18	0.1261656
513	18	0.1257740
514	39	0.1256766
515	39	0.1243338
516	39	0.1229026
517	18	0.1236459
518	49	0.1250817
519	49	0.1260376
520	49	0.1238059
521	49	0.1232605
522	49	0.1218932
523	49	0.1234715
524	49	0.1239852
525	49	0.1248634
526	49	0.1235143

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
527	49	0.1213004
528	49	0.1221156
529	49	0.1207974
530	49	0.1209080
531	49	0.1211851
532	49	0.1234721
533	49	0.1221218
534	49	0.1233733
535	49	0.1237086
536	49	0.1218787
537	49	0.1215031
538	49	0.1240225
539	49	0.1199410
540	39	0.1195988
541	39	0.1194021
542	49	0.1197649
543	49	0.1191516
544	45	0.1194277
545	49	0.1203139
546	49	0.1173420
547	49	0.1187374
548	49	0.1193399
549	49	0.1209557
550	49	0.1218194
551	49	0.1224721

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
552	49	0.1204453
553	49	0.1190238
554	49	0.1192047
555	49	0.1177390
556	49	0.1166905
557	39	0.1177233
558	39	0.1170948
559	39	0.1187485
560	39	0.1186708
561	39	0.1204108
562	39	0.1180676
563	39	0.1178552
564	39	0.1169827
565	39	0.1171240
566	39	0.1184590
567	39	0.1180581
568	39	0.1167063
569	39	0.1198615
570	39	0.1207194
571	39	0.1223095
572	39	0.1231237
573	39	0.1197775
574	39	0.1178294
575	39	0.1179339
576	39	0.1173940

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
577	39	0.1169757
578	39	0.1197161
579	39	0.1201913
580	39	0.1180706
581	49	0.1162395
582	49	0.1157247
583	55	0.1154037
584	55	0.1163758
585	49	0.1149995
586	49	0.1128041
587	55	0.1140458
588	55	0.1136099
589	55	0.1137199
590	55	0.1141768
591	55	0.1133121
592	55	0.1131199
593	55	0.1094684
594	39	0.1083393
595	14	0.1074586
596	49	0.1076488
597	49	0.1061224
598	49	0.1076224
599	49	0.1077360
600	49	0.1084938
601	49	0.1095750

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
602	55	0.1064229
603	55	0.1067695
604	55	0.1067525
605	55	0.1084464
606	55	0.1097182
607	55	0.1097392
608	55	0.1050589
609	49	0.1067534
610	49	0.1060889
611	49	0.1083980
612	49	0.1083555
613	49	0.1069420
614	49	0.1059582
615	49	0.1063252
616	49	0.1052257
617	49	0.1062611
618	49	0.1064111
619	49	0.1071214
620	49	0.1057828
621	49	0.1066986
622	49	0.1082211
623	49	0.1075969
624	49	0.1045545
625	49	0.1032140
626	49	0.1018017

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
627	49	0.1010281
628	49	0.1020342
629	49	0.0996894
630	49	0.0995977
631	49	0.0990331
632	49	0.0982770
633	14	0.0978618
634	49	0.0968251
635	55	0.0981613
636	55	0.0985784
637	14	0.0988998
638	55	0.0991768
639	55	0.0988777
640	55	0.0989425
641	55	0.0996000
642	14	0.0992395
643	14	0.0999982
644	55	0.0990678
645	55	0.1002666
646	55	0.0993017
647	55	0.1000512
648	55	0.1002881
649	55	0.0996570
650	55	0.0997963
651	55	0.1030620

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
652	55	0.1031921
653	55	0.1029411
654	55	0.1015666
655	55	0.1017696
656	55	0.1039830
657	55	0.1045796
658	55	0.1042838
659	55	0.1051968
660	55	0.1048713
661	55	0.1066924
662	55	0.1063299
663	55	0.1057296
664	55	0.1066080
665	55	0.1058089
666	55	0.1070742
667	55	0.1068519
668	55	0.1064489
669	55	0.1080524
670	55	0.1069648
671	55	0.1065559
672	55	0.1071839
673	55	0.1046295
674	55	0.1044543
675	55	0.1040199
676	55	0.1009969

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
677	55	0.1021250
678	55	0.1027856
679	55	0.1022817
680	55	0.1029390
681	55	0.1022654
682	55	0.1020496
683	55	0.1023012
684	55	0.1009532
685	55	0.1018987
686	55	0.1005915
687	55	0.0992748
688	55	0.1016186
689	55	0.1013163
690	55	0.0996928
691	55	0.0995013
692	49	0.1007736
693	49	0.0996137
694	49	0.1002489
695	49	0.1004881
696	49	0.1020854
697	49	0.1026888
698	49	0.1019159
699	49	0.1033104
700	49	0.1041163
701	49	0.1031803

Continued on next page. . .

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
702	49	0.1037608
703	49	0.1041044
704	49	0.1030363
705	49	0.1013827
706	49	0.1006497
707	49	0.0981518
708	49	0.0955409
709	55	0.0933841
710	49	0.0921843
711	55	0.0916460
712	55	0.0908314
713	55	0.0909533
714	55	0.0906236
715	55	0.0905431
716	55	0.0903893
717	55	0.0909031
718	55	0.0918530
719	55	0.0915886
720	55	0.0928344
721	55	0.0929415
722	55	0.0923451
723	55	0.0961234
724	55	0.0950473
725	55	0.0946160
726	55	0.0949606

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
727	55	0.0962001
728	55	0.0955587
729	55	0.0961317
730	55	0.0968861
731	55	0.0978185
732	55	0.0943292
733	55	0.0948866
734	55	0.0953457
735	55	0.0946788
736	55	0.0933441
737	55	0.0926212
738	55	0.0930544
739	55	0.0918708
740	55	0.0926779
741	55	0.0919959
742	55	0.0927638
743	55	0.0920838
744	55	0.0905848
745	55	0.0915521
746	55	0.0919022
747	55	0.0911126
748	55	0.0907697
749	55	0.0909209
750	55	0.0903230
751	4	0.0890581

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
752	4	0.0888933
753	49	0.0875617
754	4	0.0878989
755	4	0.0900351
756	4	0.0896846
757	4	0.0901866
758	4	0.0909353
759	4	0.0911943
760	4	0.0909901
761	4	0.0916005
762	4	0.0911186
763	4	0.0903651
764	4	0.0895175
765	4	0.0882749
766	55	0.0870990
767	55	0.0878423
768	55	0.0871923
769	55	0.0871487
770	55	0.0860260
771	13	0.0860864
772	13	0.0862226
773	13	0.0856429
774	13	0.0867188
775	13	0.0869435
776	13	0.0867114

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
777	13	0.0867184
778	13	0.0878863
779	13	0.0888550
780	13	0.0895537
781	13	0.0905561
782	13	0.0900785
783	13	0.0899887
784	13	0.0899358
785	13	0.0890704
786	49	0.0895328
787	49	0.0910152
788	49	0.0891937
789	49	0.0897551
790	49	0.0907220
791	49	0.0910111
792	49	0.0921488
793	49	0.0930149
794	49	0.0903739
795	49	0.0899434
796	49	0.0888389
797	49	0.0860980
798	49	0.0874996
799	49	0.0882701
800	49	0.0870577
801	49	0.0871849

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
802	49	0.0881486
803	49	0.0873684
804	49	0.0879287
805	49	0.0869492
806	49	0.0866824
807	49	0.0867352
808	49	0.0863614
809	49	0.0858475
810	49	0.0870789
811	49	0.0888005
812	49	0.0884198
813	49	0.0876120
814	49	0.0883852
815	49	0.0875165
816	49	0.0861759
817	55	0.0865913
818	49	0.0881848
819	49	0.0873942
820	55	0.0875900
821	55	0.0897614
822	55	0.0899131
823	55	0.0895936
824	55	0.0882686
825	49	0.0896300
826	49	0.0886353

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
827	49	0.0882864
828	49	0.0883791
829	49	0.0869607
830	49	0.0877766
831	49	0.0882844
832	55	0.0880652
833	55	0.0894067
834	55	0.0871492
835	55	0.0875038
836	55	0.0884654
837	55	0.0880998
838	55	0.0874045
839	55	0.0884925
840	55	0.0888081
841	55	0.0888056
842	55	0.0887486
843	55	0.0889774
844	55	0.0892100
845	55	0.0906139
846	55	0.0930832
847	55	0.0940336
848	55	0.0946389
849	55	0.0954573
850	55	0.0953053
851	55	0.0943355

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
852	55	0.0954708
853	55	0.0927160
854	55	0.0929324
855	55	0.0944501
856	55	0.0962862
857	55	0.0971369
858	55	0.0967049
859	55	0.0963681
860	55	0.0965303
861	55	0.0967337
862	55	0.0968336
863	55	0.0990499
864	55	0.0972141
865	55	0.0986663
866	55	0.0974706
867	55	0.0958905
868	55	0.0937199
869	55	0.0938506
870	55	0.0955982
871	55	0.0948846
872	55	0.0954380
873	55	0.0945044
874	55	0.0954321
875	55	0.0955732
876	55	0.0943632

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
877	55	0.0940160
878	55	0.0944340
879	55	0.0926690
880	55	0.0928925
881	55	0.0919919
882	55	0.0940479
883	55	0.0938717
884	55	0.0929550
885	55	0.0938559
886	55	0.0937409
887	55	0.0949950
888	55	0.0923506
889	55	0.0932639
890	55	0.0938137
891	55	0.0939190
892	55	0.0947786
893	55	0.0954386
894	55	0.0949424
895	55	0.0945879
896	55	0.0948134
897	55	0.0954230
898	55	0.0959859
899	55	0.0953008
900	55	0.0951610
1000	55	0.0910483

Continued on next page...

Table B.3 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1100	55	0.0866581
1200	55	0.0744983
1300	18	0.0804488
1400	55	0.0724334
1500	55	0.0784814
1600	55	0.0756255
1700	55	0.0779620
1800	55	0.0804208
1900	55	0.0843767
2000	55	0.0824246
2100	55	0.0775019
2200	55	0.0787086
2300	55	0.0774521
2400	55	0.0787248
2500	55	0.0787341
2600	55	0.0786563
2700	55	0.0799934
2800	55	0.0816448
2900	55	0.0778432
3000	55	0.0766251

B.1.4 Dual Rail Path Switching DES SBox with Hamming Weight Hypothesis and Partition Function of 3

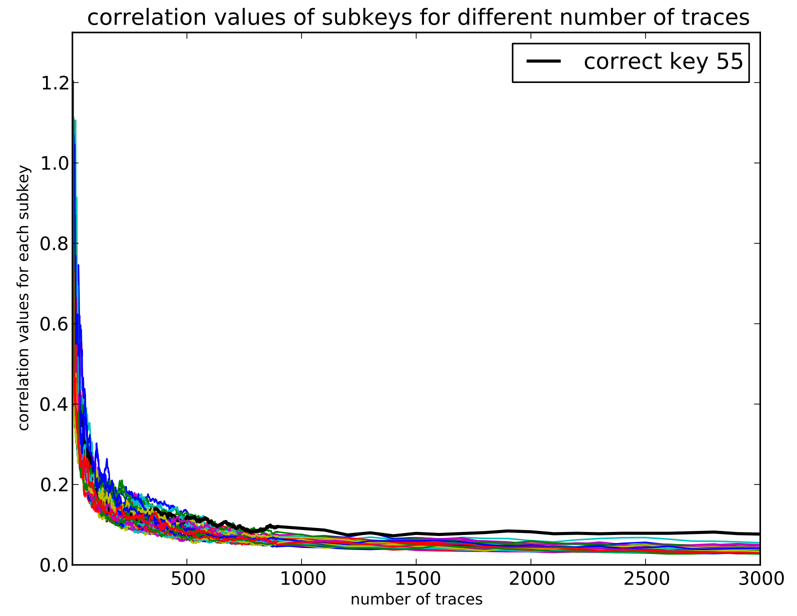


FIGURE B.3: DPA result for all encryption rounds on FPGA Implementation of dual rail alternating spacer DES Sbox for Hamming weight hypothesis and partition function of 2

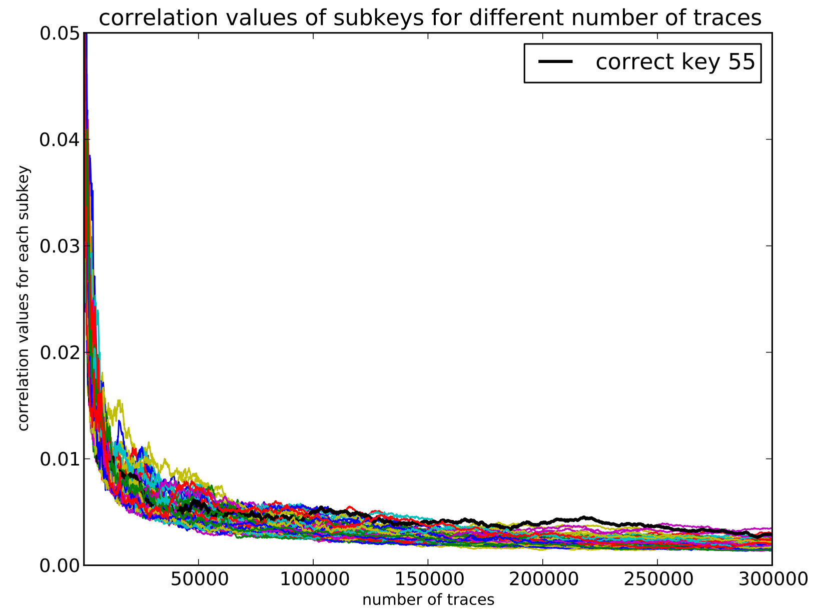


FIGURE B.4: DPA result for all encryption rounds on FPGA Implementation of dual rail path switching DES Sbox for Hamming weight hypothesis and partition function of 3

TABLE B.4: Different number of traces and the subkey with highest correlation value for dual rail path switching DES Sbox using Hamming weight hypothesis and partition function of 3

number of traces	subkey with highest correlation	correlation value
100	2	0.1864230

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1000	13	0.0571700
2000	38	0.0395304
3000	30	0.0369098
4000	30	0.0322767
5000	39	0.0253517
6000	39	0.0228228
7000	39	0.0186005
8000	30	0.0167514
9000	30	0.0153658
10000	19	0.0140555
11000	11	0.0146432
12000	11	0.0145491
13000	11	0.0142485
14000	11	0.0139054
15000	11	0.0155173
16000	11	0.0143426
17000	11	0.0145552
18000	11	0.0127238
19000	11	0.0119311
20000	11	0.0121111
21000	11	0.0111994
22000	11	0.0113200
23000	11	0.0102036
24000	30	0.0104529
25000	30	0.0108667

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
26000	21	0.0108181
27000	11	0.0107061
28000	11	0.0111509
29000	11	0.0106669
30000	35	0.0100548
31000	35	0.0095394
32000	35	0.0094404
33000	35	0.0086670
34000	35	0.0090721
35000	35	0.0095915
36000	35	0.0096214
37000	35	0.0094423
38000	35	0.0083414
39000	35	0.0086830
40000	35	0.0087643
41000	35	0.0085365
42000	35	0.0085661
43000	35	0.0082714
44000	35	0.0087147
45000	35	0.0084287
46000	35	0.0089015
47000	11	0.0082525
48000	11	0.0084689
49000	11	0.0085251
50000	35	0.0082216

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
51000	35	0.0080078
52000	35	0.0076579
53000	11	0.0076615
54000	35	0.0075904
55000	35	0.0072969
56000	49	0.0072853
57000	11	0.0071251
58000	11	0.0073078
59000	11	0.0072875
60000	11	0.0074691
61000	11	0.0067014
62000	11	0.0063585
63000	35	0.0063174
64000	35	0.0059372
65000	4	0.0059724
66000	52	0.0057689
67000	19	0.0061662
68000	35	0.0057848
69000	30	0.0059224
70000	4	0.0056294
71000	4	0.0058068
72000	4	0.0058432
73000	4	0.0058744
74000	4	0.0059235
75000	4	0.0055344

Continued on next page. . .

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
76000	4	0.0057432
77000	4	0.0055065
78000	27	0.0056586
79000	0	0.0057694
80000	9	0.0055703
81000	8	0.0056989
82000	8	0.0055870
83000	8	0.0057978
84000	8	0.0058555
85000	8	0.0058818
86000	8	0.0057786
87000	8	0.0054693
88000	9	0.0055625
89000	9	0.0055769
90000	9	0.0057077
91000	4	0.0054866
92000	9	0.0056443
93000	9	0.0057376
94000	9	0.0056233
95000	9	0.0056760
96000	30	0.0055087
97000	8	0.0054830
98000	30	0.0053676
99000	30	0.0052376
100000	8	0.0052288

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
101000	30	0.0054092
102000	30	0.0053012
103000	30	0.0053579
104000	55	0.0053197
105000	30	0.0053938
106000	30	0.0054175
107000	30	0.0052218
108000	55	0.0051043
109000	55	0.0050359
110000	55	0.0049911
111000	8	0.0050790
112000	55	0.0050549
113000	8	0.0051048
114000	8	0.0051858
115000	8	0.0053583
116000	8	0.0055252
117000	8	0.0053803
118000	8	0.0051494
119000	8	0.0050019
120000	8	0.0049568
121000	8	0.0049257
122000	8	0.0047920
123000	55	0.0047446
124000	8	0.0046972
125000	8	0.0048873

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
126000	8	0.0050926
127000	8	0.0051280
128000	8	0.0050630
129000	8	0.0050891
130000	8	0.0049499
131000	27	0.0050037
132000	27	0.0047330
133000	27	0.0048803
134000	27	0.0049152
135000	27	0.0047058
136000	27	0.0046844
137000	27	0.0045961
138000	27	0.0046757
139000	9	0.0045648
140000	9	0.0046518
141000	9	0.0046424
142000	9	0.0044978
143000	9	0.0045772
144000	9	0.0045676
145000	9	0.0045794
146000	9	0.0043532
147000	9	0.0043971
148000	9	0.0044097
149000	9	0.0043729
150000	9	0.0043418

Continued on next page. . .

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
151000	9	0.0043250
152000	9	0.0042529
153000	55	0.0040829
154000	9	0.0040796
155000	55	0.0040543
156000	55	0.0041429
157000	55	0.0042096
158000	55	0.0041349
159000	55	0.0041784
160000	55	0.0041035
161000	55	0.0040417
162000	55	0.0040020
163000	55	0.0039734
164000	55	0.0040880
165000	55	0.0042057
166000	55	0.0043470
167000	55	0.0042089
168000	55	0.0041050
169000	55	0.0042293
170000	55	0.0042147
171000	55	0.0040342
172000	55	0.0038762
173000	55	0.0040316
174000	55	0.0039696
175000	55	0.0037897

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
176000	55	0.0036552
177000	29	0.0037744
178000	29	0.0037713
179000	29	0.0038361
180000	29	0.0039491
181000	29	0.0040198
182000	29	0.0039052
183000	29	0.0038448
184000	29	0.0037289
185000	29	0.0037061
186000	29	0.0037826
187000	29	0.0039236
188000	29	0.0038037
189000	29	0.0038420
190000	29	0.0039137
191000	55	0.0039176
192000	55	0.0039366
193000	55	0.0040412
194000	55	0.0040350
195000	55	0.0038481
196000	55	0.0037853
197000	55	0.0038053
198000	55	0.0037659
199000	55	0.0039166
200000	55	0.0039746

Continued on next page. . .

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
201000	55	0.0040610
202000	55	0.0039310
203000	55	0.0040197
204000	55	0.0041089
205000	55	0.0041645
206000	55	0.0043203
207000	55	0.0043185
208000	55	0.0043026
209000	55	0.0042295
210000	55	0.0041645
211000	55	0.0041330
212000	55	0.0042378
213000	55	0.0042294
214000	55	0.0042991
215000	55	0.0042335
216000	55	0.0043252
217000	55	0.0044414
218000	55	0.0045097
219000	55	0.0044128
220000	55	0.0043861
221000	55	0.0044492
222000	55	0.0043496
223000	55	0.0041958
224000	55	0.0041501
225000	55	0.0040774

Continued on next page. . .

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
226000	55	0.0039557
227000	55	0.0039465
228000	55	0.0039548
229000	55	0.0039128
230000	55	0.0038581
231000	55	0.0038739
232000	55	0.0037486
233000	55	0.0037329
234000	55	0.0038066
235000	55	0.0037969
236000	55	0.0038390
237000	55	0.0039049
238000	55	0.0038894
239000	55	0.0037870
240000	55	0.0038600
241000	55	0.0037560
242000	55	0.0038716
243000	55	0.0038441
244000	55	0.0037063
245000	55	0.0036830
246000	55	0.0037319
247000	55	0.0037048
248000	55	0.0036445
249000	55	0.0036656
250000	40	0.0038047

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
251000	40	0.0037881
252000	40	0.0038942
253000	40	0.0038852
254000	40	0.0039223
255000	40	0.0038078
256000	40	0.0038195
257000	40	0.0037277
258000	40	0.0037581
259000	40	0.0037664
260000	40	0.0037538
261000	40	0.0036617
262000	40	0.0037087
263000	40	0.0037063
264000	40	0.0036396
265000	40	0.0035687
266000	40	0.0036479
267000	40	0.0035209
268000	40	0.0035795
269000	40	0.0036515
270000	40	0.0035722
271000	40	0.0036029
272000	40	0.0036422
273000	40	0.0035021
274000	40	0.0034101
275000	40	0.0033921

Continued on next page...

Table B.4 – continued from previous page

number of traces	subkey with highest correlation	correlation value
276000	40	0.0034313
277000	40	0.0033595
278000	40	0.0033533
279000	40	0.0033677
280000	40	0.0033121
281000	40	0.0032578
282000	40	0.0031910
283000	40	0.0032872
284000	40	0.0031690
285000	40	0.0032337
286000	40	0.0032292
287000	40	0.0033336
288000	40	0.0034192
289000	40	0.0033518
290000	40	0.0033895
291000	40	0.0034493
292000	40	0.0034332
293000	40	0.0034439
294000	40	0.0034178
295000	40	0.0033915
296000	40	0.0034610
297000	40	0.0034470
298000	40	0.0034446
299000	40	0.0034420
300000	40	0.0035082

B.1.5 Dual Rail Path Switching and Alternating Spacer DES SBox with Hamming Weight Hypothesis and Partition Function of 1

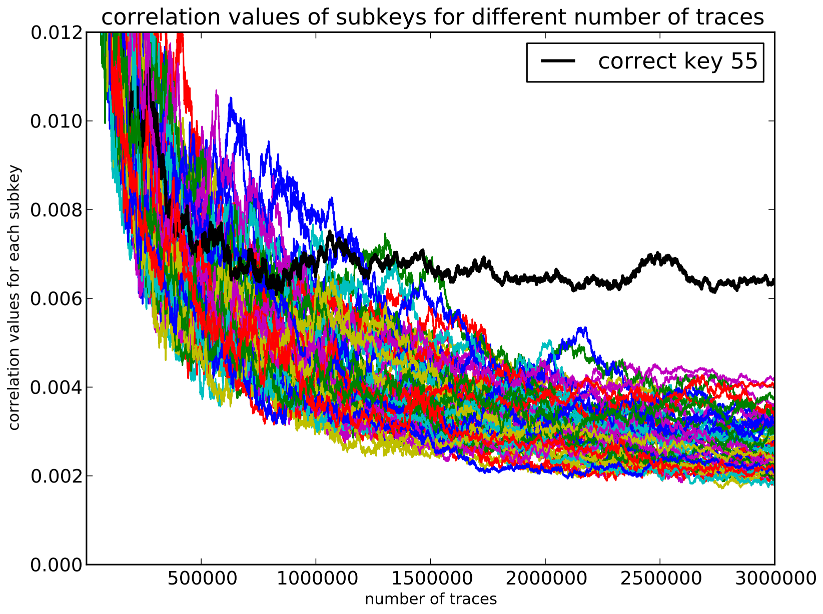


FIGURE B.5: DPA result for all encryption rounds on FPGA Implementation of dual rail path switching and alternating spacer DES Sbox for Hamming weight hypothesis and partition function of 1

TABLE B.5: Different number of traces and the subkey with highest correlation value for dual rail path switching and alternating spacer using Hamming weight hypothesis and partition function of 1

number of traces	**subkey with highest correlation**	**correlation value**
100	6	5.3739707
3000	4	0.2100494
6000	4	0.1350354
9000	56	0.1093362

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
12000	63	0.0844690
15000	36	0.0624367
18000	36	0.0613229
21000	11	0.0552916
24000	19	0.0518406
27000	19	0.0522575
30000	8	0.0495707
33000	27	0.0530693
36000	27	0.0462193
39000	27	0.0409506
42000	8	0.0421104
45000	8	0.0426450
48000	8	0.0414585
51000	27	0.0373147
54000	27	0.0345453
57000	31	0.0321307
60000	31	0.0314572
63000	27	0.0265172
66000	46	0.0277116
69000	46	0.0279779
72000	46	0.0249346
75000	31	0.0261547
78000	58	0.0254682
81000	58	0.0246726
84000	26	0.0252818

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
87000	27	0.0262779
90000	26	0.0259857
93000	26	0.0254840
96000	26	0.0243916
99000	26	0.0246518
102000	26	0.0250454
105000	26	0.0244660
108000	26	0.0228652
111000	31	0.0229301
114000	31	0.0221332
117000	29	0.0208623
120000	8	0.0196844
123000	8	0.0199263
126000	8	0.0210342
129000	8	0.0207446
132000	8	0.0221198
135000	8	0.0209493
138000	8	0.0206186
141000	8	0.0198728
144000	26	0.0195697
147000	26	0.0197015
150000	26	0.0195837
153000	26	0.0194169
156000	26	0.0186968
159000	27	0.0195825

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
162000	27	0.0208858
165000	27	0.0201589
168000	27	0.0183216
171000	27	0.0165416
174000	27	0.0169392
177000	27	0.0176339
180000	27	0.0164896
183000	26	0.0164601
186000	19	0.0165769
189000	27	0.0168050
192000	26	0.0166662
195000	26	0.0167873
198000	26	0.0167928
201000	26	0.0161910
204000	46	0.0158292
207000	46	0.0172919
210000	46	0.0177745
213000	46	0.0174733
216000	46	0.0174399
219000	46	0.0167660
222000	26	0.0165160
225000	26	0.0162490
228000	26	0.0161804
231000	26	0.0161690
234000	26	0.0158373

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
237000	26	0.0157588
240000	26	0.0152679
243000	26	0.0154761
246000	32	0.0153332
249000	32	0.0152866
252000	26	0.0150850
255000	26	0.0150248
258000	32	0.0151464
261000	32	0.0152050
264000	32	0.0152117
267000	32	0.0154610
270000	32	0.0159063
273000	32	0.0156025
276000	32	0.0148691
279000	32	0.0140731
282000	32	0.0136029
285000	32	0.0143472
288000	32	0.0138630
291000	32	0.0136291
294000	32	0.0137277
297000	32	0.0134650
300000	32	0.0134665
303000	32	0.0139631
306000	32	0.0129805
309000	32	0.0132760

Continued on next page. . .

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
312000	32	0.0128513
315000	32	0.0135526
318000	32	0.0126694
321000	32	0.0122913
324000	32	0.0118667
327000	32	0.0126394
330000	32	0.0128460
333000	32	0.0130144
336000	32	0.0131921
339000	32	0.0125825
342000	32	0.0120754
345000	32	0.0121127
348000	32	0.0117571
351000	32	0.0113184
354000	32	0.0110913
357000	46	0.0109471
360000	32	0.0114608
363000	32	0.0112190
366000	46	0.0115805
369000	46	0.0113435
372000	46	0.0112666
375000	32	0.0108436
378000	32	0.0113357
381000	43	0.0108254
384000	46	0.0103036

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
387000	32	0.0104243
390000	32	0.0108768
393000	32	0.0114462
396000	32	0.0118735
399000	32	0.0119687
402000	32	0.0115217
405000	32	0.0116417
408000	32	0.0116889
411000	32	0.0119021
414000	32	0.0120916
417000	32	0.0121092
420000	32	0.0117748
423000	32	0.0114698
426000	32	0.0105435
429000	32	0.0109783
432000	32	0.0103924
435000	32	0.0103363
438000	32	0.0109192
441000	32	0.0104560
444000	32	0.0106811
447000	32	0.0104026
450000	32	0.0104922
453000	32	0.0102907
456000	32	0.0102824
459000	32	0.0103129

Continued on next page. . .

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
462000	32	0.0096562
465000	32	0.0095937
468000	36	0.0097050
471000	32	0.0096270
474000	32	0.0094090
477000	32	0.0093824
480000	32	0.0098964
483000	32	0.0096303
486000	26	0.0094629
489000	26	0.0095103
492000	26	0.0094072
495000	26	0.0094045
498000	43	0.0093136
501000	34	0.0094973
504000	34	0.0096856
507000	34	0.0096067
510000	34	0.0094838
513000	29	0.0092175
516000	43	0.0090579
519000	30	0.0091453
522000	30	0.0094254
525000	30	0.0093472
528000	34	0.0089803
531000	30	0.0089736
534000	34	0.0093532

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
537000	30	0.0093654
540000	46	0.0093746
543000	46	0.0097470
546000	46	0.0097896
549000	46	0.0100516
552000	46	0.0097007
555000	46	0.0097260
558000	46	0.0095389
561000	46	0.0098776
564000	46	0.0103790
567000	46	0.0105724
570000	46	0.0104257
573000	46	0.0105334
576000	46	0.0101494
579000	46	0.0095663
582000	46	0.0094398
585000	46	0.0091849
588000	30	0.0088067
591000	42	0.0088557
594000	42	0.0089596
597000	42	0.0089601
600000	46	0.0089620
603000	46	0.0090544
606000	42	0.0090809
609000	42	0.0096147

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
612000	42	0.0094984
615000	42	0.0098073
618000	42	0.0099008
621000	42	0.0101128
624000	42	0.0101779
627000	42	0.0100255
630000	42	0.0103537
633000	42	0.0102597
636000	42	0.0102035
639000	42	0.0101802
642000	42	0.0101697
645000	42	0.0099863
648000	42	0.0100775
651000	42	0.0100642
654000	42	0.0097487
657000	42	0.0098822
660000	24	0.0096640
663000	24	0.0098802
666000	24	0.0098189
669000	24	0.0097702
672000	24	0.0099454
675000	24	0.0097271
678000	24	0.0097118
681000	24	0.0095264
684000	24	0.0098145

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
687000	24	0.0099466
690000	24	0.0097517
693000	24	0.0095786
696000	24	0.0092863
699000	24	0.0092896
702000	24	0.0091363
705000	24	0.0091020
708000	24	0.0091345
711000	24	0.0087885
714000	30	0.0086176
717000	24	0.0088787
720000	30	0.0085914
723000	24	0.0085690
726000	24	0.0087205
729000	46	0.0085209
732000	46	0.0085135
735000	46	0.0084449
738000	46	0.0084377
741000	46	0.0084766
744000	46	0.0083779
747000	24	0.0083246
750000	24	0.0082457
753000	24	0.0083991
756000	24	0.0087487
759000	24	0.0089386

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
762000	24	0.0090976
765000	24	0.0090966
768000	24	0.0089219
771000	24	0.0089208
774000	24	0.0090094
777000	24	0.0089521
780000	24	0.0090104
783000	24	0.0091998
786000	24	0.0092621
789000	24	0.0093209
792000	24	0.0093855
795000	24	0.0094170
798000	24	0.0093175
801000	24	0.0089980
804000	24	0.0089144
807000	24	0.0087205
810000	24	0.0086377
813000	24	0.0086460
816000	24	0.0089100
819000	24	0.0088302
822000	24	0.0087974
825000	24	0.0087704
828000	24	0.0088905
831000	24	0.0090200
834000	24	0.0090356

Continued on next page. . .

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
837000	24	0.0090039
840000	24	0.0087975
843000	24	0.0089089
846000	42	0.0089031
849000	42	0.0086979
852000	42	0.0085105
855000	42	0.0086222
858000	42	0.0084939
861000	24	0.0084462
864000	42	0.0084298
867000	42	0.0082688
870000	42	0.0085504
873000	42	0.0086918
876000	42	0.0088277
879000	42	0.0084793
882000	42	0.0085471
885000	42	0.0086087
888000	42	0.0085781
891000	42	0.0083225
894000	42	0.0086783
897000	42	0.0086582
900000	42	0.0086524
903000	42	0.0085046
906000	42	0.0085974
909000	42	0.0085506

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
912000	42	0.0084726
915000	42	0.0085897
918000	42	0.0084701
921000	42	0.0083585
924000	42	0.0079758
927000	42	0.0081404
930000	42	0.0079439
933000	42	0.0080602
936000	42	0.0079516
939000	42	0.0081909
942000	42	0.0082215
945000	42	0.0082754
948000	42	0.0083094
951000	42	0.0080805
954000	42	0.0080784
957000	42	0.0078767
960000	42	0.0079119
963000	42	0.0078990
966000	42	0.0078798
969000	42	0.0079191
972000	42	0.0078203
975000	42	0.0077489
978000	42	0.0078601
981000	42	0.0080549
984000	42	0.0079371

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
987000	42	0.0079539
990000	42	0.0079580
993000	42	0.0079425
996000	42	0.0078473
999000	42	0.0078922
1002000	42	0.0077435
1005000	42	0.0077763
1008000	42	0.0079061
1011000	42	0.0077792
1014000	42	0.0077761
1017000	42	0.0077692
1020000	42	0.0076013
1023000	42	0.0077813
1026000	42	0.0078925
1029000	42	0.0078967
1032000	42	0.0079520
1035000	42	0.0078437
1038000	42	0.0077193
1041000	42	0.0076571
1044000	42	0.0076677
1047000	42	0.0077746
1050000	42	0.0079288
1053000	42	0.0078583
1056000	42	0.0080997
1059000	42	0.0080249

Continued on next page. . .

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1062000	42	0.0082066
1065000	42	0.0082083
1068000	42	0.0081917
1071000	42	0.0082355
1074000	42	0.0080954
1077000	42	0.0079169
1080000	42	0.0077165
1083000	42	0.0076093
1086000	42	0.0076472
1089000	42	0.0074675
1092000	42	0.0073341
1095000	42	0.0073591
1098000	30	0.0074981
1101000	42	0.0073324
1104000	42	0.0073928
1107000	42	0.0075132
1110000	42	0.0074058
1113000	42	0.0075387
1116000	42	0.0075772
1119000	42	0.0073070
1122000	42	0.0071948
1125000	55	0.0071384
1128000	55	0.0072318
1131000	55	0.0072480
1134000	55	0.0071333

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1137000	55	0.0071245
1140000	30	0.0073030
1143000	30	0.0073777
1146000	30	0.0073828
1149000	42	0.0074158
1152000	42	0.0074470
1155000	42	0.0073998
1158000	42	0.0072859
1161000	42	0.0072334
1164000	42	0.0073224
1167000	30	0.0071799
1170000	30	0.0072428
1173000	30	0.0072828
1176000	30	0.0073105
1179000	30	0.0072648
1182000	30	0.0072367
1185000	42	0.0070653
1188000	42	0.0069950
1191000	42	0.0069715
1194000	42	0.0070185
1197000	42	0.0069411
1200000	42	0.0069179
1203000	42	0.0069512
1206000	25	0.0070618
1209000	25	0.0070588

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1212000	25	0.0069158
1215000	25	0.0069763
1218000	25	0.0070091
1221000	25	0.0069915
1224000	25	0.0068595
1227000	25	0.0067928
1230000	25	0.0068715
1233000	25	0.0068995
1236000	25	0.0069465
1239000	55	0.0067475
1242000	25	0.0070003
1245000	25	0.0070302
1248000	25	0.0069183
1251000	25	0.0070566
1254000	25	0.0071042
1257000	25	0.0070704
1260000	25	0.0071172
1263000	25	0.0071345
1266000	25	0.0070995
1269000	25	0.0070036
1272000	25	0.0070387
1275000	25	0.0071129
1278000	25	0.0070959
1281000	25	0.0070463
1284000	25	0.0069021

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1287000	25	0.0069887
1290000	25	0.0071178
1293000	25	0.0070253
1296000	25	0.0069401
1299000	25	0.0071972
1302000	25	0.0074150
1305000	25	0.0072691
1308000	25	0.0072573
1311000	25	0.0072861
1314000	25	0.0072987
1317000	25	0.0072689
1320000	25	0.0071290
1323000	25	0.0070973
1326000	25	0.0070790
1329000	25	0.0069734
1332000	55	0.0068854
1335000	55	0.0068001
1338000	55	0.0068746
1341000	55	0.0069675
1344000	55	0.0069952
1347000	55	0.0069820
1350000	55	0.0069931
1353000	55	0.0070146
1356000	55	0.0068316
1359000	55	0.0067698

Continued on next page...

Table B.5 – continued from previous page

number of traces	subkey with highest correlation	correlation value
1362000	55	0.0067524
1365000	55	0.0067995
1368000	55	0.0067683
1371000	55	0.0067521
1374000	55	0.0067472
1377000	55	0.0067550
1380000	55	0.0068868
1383000	55	0.0068435
1386000	55	0.0067894
1389000	55	0.0067658
1392000	55	0.0067847
2997000	55	0.0063799
3000000	55	0.0064045

B.2 AES Sbox

DPA results from various AES Sbox implementation are presented here.

B.2.1 AES SBox with Hamming Weight Hypothesis and Partition Function of 5

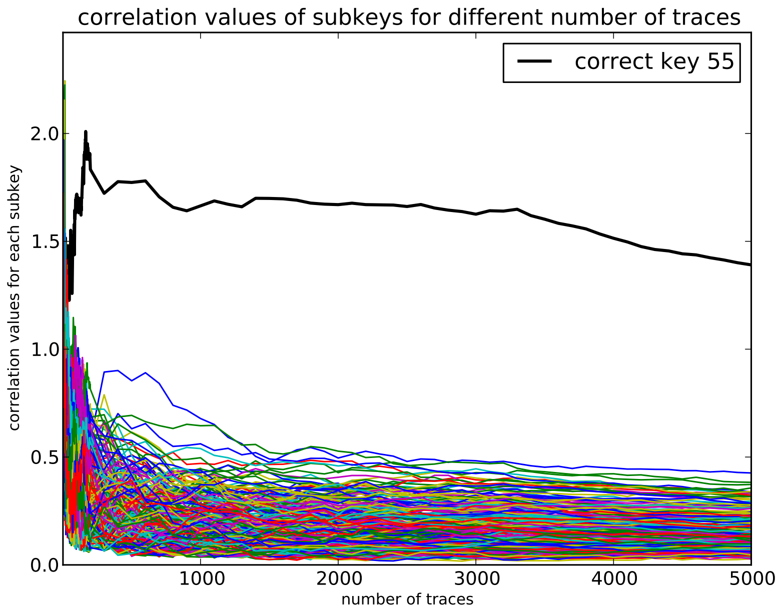


FIGURE B.6: DPA result for all encryption rounds on FPGA Implementation of AES Sbox without any countermeasure for Hamming weight hypothesis and partition function of 5

TABLE B.6: Different number of traces and the subkey with highest correlation value for unprotected AES Sbox using Hamming weight hypothesis and partition function of 5

number of traces	subkey with highest correlation	correlation value
1	255	1.2630413
2	253	1.7985434
3	253	1.6788640
4	253	1.7901968
5	253	1.8804406
6	253	1.9701186
7	246	2.0406005
8	246	2.0738040
9	246	2.1159032
10	246	2.1550969

Continued on next page...

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
11	242	2.1664025
12	242	2.1886611
13	242	2.1979131
14	242	2.2248992
15	198	2.2437524
16	15	1.4320802
17	15	1.5637997
18	66	1.5130662
19	55	1.5154489
20	55	1.4499997
21	56	1.5000871
22	102	1.3386762
23	243	1.4167817
24	243	1.3734384
25	243	1.3717232
26	102	1.3575847
27	102	1.3919015
28	102	1.3543731
29	243	1.3679693
30	102	1.3441177
31	55	1.3709769
32	55	1.3948820
33	55	1.3748670
34	55	1.4119024
35	55	1.4243068

Continued on next page...

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
36	55	1.4637633
37	55	1.4125204
38	55	1.4799811
39	55	1.4093935
40	55	1.3567415
41	55	1.3885749
42	55	1.3768263
43	55	1.4226672
44	55	1.4054749
45	55	1.3848384
46	55	1.2263570
47	55	1.3010535
48	55	1.2802982
49	55	1.2591252
50	55	1.3112822
51	55	1.2773553
52	55	1.3350066
53	55	1.3761053
54	55	1.3841436
55	55	1.3467669
56	55	1.4138026
57	55	1.5514006
58	55	1.4509537
59	55	1.4892469
60	55	1.4121466

Continued on next page...

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
61	55	1.4381564
62	55	1.4353020
63	55	1.3863895
64	55	1.3731556
65	55	1.4164625
66	55	1.3257131
67	55	1.2583099
68	55	1.2862437
69	55	1.3769566
70	55	1.4342134
71	55	1.4501447
72	55	1.4638102
73	55	1.4972211
74	55	1.4657037
75	55	1.4742830
76	55	1.5279856
77	55	1.5134902
78	55	1.4959699
79	55	1.4941519
80	55	1.4861487
81	55	1.5053766
82	55	1.5781465
83	55	1.5112580
84	55	1.4375484
85	55	1.5058158

Continued on next page...

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
86	55	1.6055982
87	55	1.6448240
88	55	1.6036326
89	55	1.6272550
90	55	1.6380619
91	55	1.6289738
92	55	1.6527666
93	55	1.6944706
94	55	1.6730881
95	55	1.6499684
96	55	1.6478046
97	55	1.7019706
98	55	1.6898489
99	55	1.7150950
100	55	1.6888085
101	55	1.7187922
102	55	1.6850029
103	55	1.6748693
104	55	1.7042422
105	55	1.7038727
106	55	1.6965097
107	55	1.7017183
108	55	1.7024576
109	55	1.7043235
110	55	1.6940711

Continued on next page. . .

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
111	55	1.6693175
112	55	1.6806518
113	55	1.6767794
114	55	1.6585076
115	55	1.6322155
116	55	1.6573778
117	55	1.6475488
118	55	1.6549817
119	55	1.6544064
120	55	1.6418997
121	55	1.6657254
122	55	1.6986194
123	55	1.6537970
124	55	1.6635044
125	55	1.6642765
126	55	1.6663192
127	55	1.6973641
128	55	1.6871108
129	55	1.6720154
130	55	1.6798688
131	55	1.6384639
132	55	1.6214708
133	55	1.6620521
134	55	1.6424062
135	55	1.6641516

Continued on next page...

Table B.6 – continued from previous page

number of traces	subkey with highest correlation	correlation value
136	55	1.7023159
137	55	1.6990461
138	55	1.6997472
139	55	1.7369018
140	55	1.7420421
141	55	1.7510590
142	55	1.7777076
143	55	1.8181909
144	55	1.8348008
145	55	1.8423445
146	55	1.8388566
147	55	1.8354966
148	55	1.8296975
149	55	1.8368166
150	55	1.7943260
4900	55	1.4009086
5000	55	1.3908096

B.2.2 Dual Rail AES SBox with Hamming Distance Hypothesis and Partition Function of 2

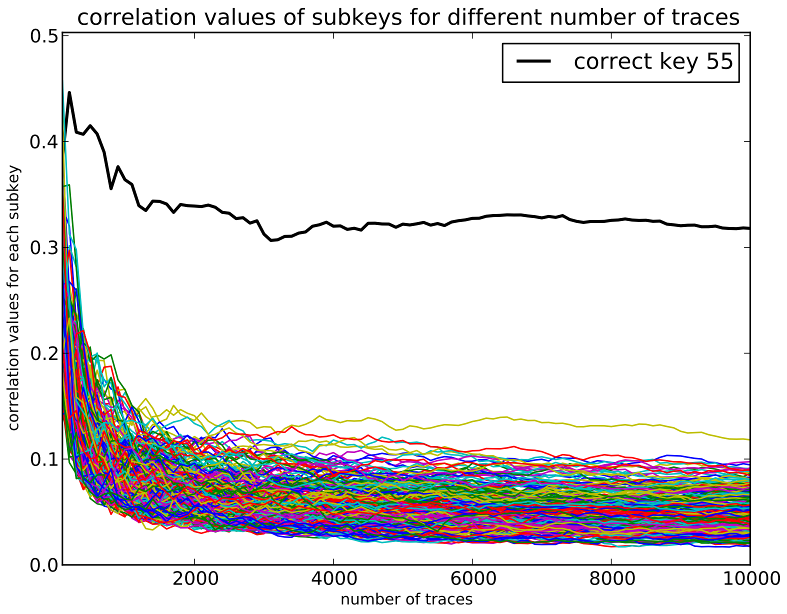


FIGURE B.7: DPA result for all encryption rounds on FPGA Implementation of dual rail AES Sbox for Hamming distance hypothesis and partition function of 2

TABLE B.7: Different number of traces and the subkey with highest correlation value for dual rail AES Sbox using Hamming distance hypothesis and partition function of 2

number of traces	subkey with highest correlation	correlation value
100	214	0.4573025
200	55	0.4463399
300	55	0.4088132
400	55	0.4068426
500	55	0.4149826
600	55	0.4072080
700	55	0.3899829
800	55	0.3554054
900	55	0.3762647
1000	55	0.3639463

Continued on next page...

Table B.7 – continued from previous page

number of traces	subkey with highest correlation	correlation value
2000	55	0.3390435
3000	55	0.3125784
4000	55	0.3199967
5000	55	0.3219624
6000	55	0.3274026
7000	55	0.3278641
8000	55	0.3255407
9000	55	0.3204393
10000	55	0.3179254

B.2.3 Dual Rail Path Switching and Alternating Spacer AES SBox with Hamming Weight Hypothesis and Partition Function of 6

Table B.8: Different number of traces and the subkey with highest correlation value for dual rail path switching and alternating spacer AES Sbox using Hamming weight hypothesis and partition function of 6

number of traces	subkey with highest correlation	correlation value
100	184	3.7244470
200	120	1.6268552
300	7	0.9280507
400	7	0.8022216
500	7	0.6770780

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
600	115	0.5919207
700	231	0.5377365
800	192	0.5271584
900	231	0.5150363
1000	231	0.4396287
2000	14	0.2930141
3000	136	0.1995848
4000	242	0.1760428
5000	231	0.1435665
6000	242	0.1431486
7000	242	0.1285860
8000	190	0.1305335
9000	229	0.1246925
10000	229	0.1156641
11000	47	0.1020992
12000	205	0.1077231
13000	4	0.1019187
14000	205	0.1042027
15000	122	0.1004655
16000	122	0.1091084
17000	223	0.1010120
18000	201	0.0977701
19000	201	0.0959144
20000	201	0.0820969
21000	201	0.0802810

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
22000	201	0.0757032
23000	122	0.0740441
24000	7	0.0741756
25000	201	0.0700609
26000	201	0.0692413
27000	201	0.0684092
28000	201	0.0661208
29000	7	0.0633062
30000	201	0.0607386
31000	201	0.0624711
32000	201	0.0661944
33000	201	0.0641500
34000	201	0.0598044
35000	201	0.0605517
36000	201	0.0538536
37000	122	0.0507238
38000	7	0.0510412
39000	7	0.0511577
40000	7	0.0493656
41000	225	0.0500583
42000	68	0.0519712
43000	68	0.0517961
44000	68	0.0526788
45000	68	0.0527977
46000	68	0.0489943

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
47000	68	0.0468788
48000	251	0.0494824
49000	251	0.0495595
50000	7	0.0470209
51000	7	0.0477863
52000	7	0.0468247
53000	7	0.0431205
54000	7	0.0461674
55000	7	0.0472631
56000	7	0.0465017
57000	137	0.0460838
58000	7	0.0454875
59000	7	0.0458326
60000	7	0.0450992
61000	7	0.0431937
62000	137	0.0427780
63000	137	0.0446129
64000	137	0.0437369
65000	137	0.0426653
66000	137	0.0415927
67000	137	0.0430407
68000	137	0.0404119
69000	249	0.0395140
70000	249	0.0406183
71000	249	0.0385215

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
72000	232	0.0392126
73000	249	0.0385371
74000	232	0.0381966
75000	232	0.0397256
76000	232	0.0405868
77000	232	0.0390645
78000	243	0.0390102
79000	243	0.0386934
80000	243	0.0399839
81000	243	0.0403155
82000	249	0.0410672
83000	249	0.0425562
84000	249	0.0439586
85000	249	0.0428372
86000	249	0.0424620
87000	249	0.0426041
88000	249	0.0413076
89000	249	0.0419462
90000	249	0.0399539
91000	249	0.0375386
92000	24	0.0382264
93000	24	0.0383415
94000	24	0.0374184
95000	243	0.0374418
96000	243	0.0376459

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
97000	243	0.0370115
98000	125	0.0374536
99000	125	0.0364398
100000	125	0.0371804
101000	125	0.0373255
102000	125	0.0368776
103000	125	0.0364705
104000	125	0.0355352
105000	24	0.0354862
106000	24	0.0353939
107000	125	0.0355266
108000	125	0.0367945
109000	24	0.0359103
110000	125	0.0369796
111000	24	0.0363374
112000	24	0.0369675
113000	24	0.0372868
114000	24	0.0364387
115000	24	0.0350208
116000	24	0.0350044
117000	24	0.0353315
118000	24	0.0346486
119000	24	0.0336473
120000	24	0.0331480
121000	243	0.0346872

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
122000	243	0.0338257
123000	243	0.0343575
124000	125	0.0334394
125000	243	0.0331163
126000	243	0.0334237
127000	243	0.0328164
128000	243	0.0322323
129000	243	0.0315885
130000	7	0.0312468
131000	7	0.0305260
132000	243	0.0306095
133000	7	0.0306353
134000	125	0.0313761
135000	125	0.0311161
136000	125	0.0323157
137000	125	0.0320256
138000	7	0.0317896
139000	7	0.0315214
140000	7	0.0311156
141000	7	0.0305633
142000	7	0.0309297
143000	7	0.0319914
144000	7	0.0323856
145000	7	0.0330045
146000	7	0.0332507

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
147000	7	0.0328859
148000	7	0.0324081
149000	7	0.0321522
150000	125	0.0320092
151000	125	0.0333342
152000	125	0.0334710
153000	125	0.0326095
154000	125	0.0329126
155000	125	0.0328310
156000	125	0.0322136
157000	125	0.0311235
158000	125	0.0311713
159000	125	0.0298715
160000	7	0.0304348
161000	125	0.0312657
162000	125	0.0316613
163000	125	0.0307573
164000	125	0.0303555
165000	125	0.0307447
166000	125	0.0302331
167000	125	0.0292737
168000	125	0.0283970
169000	7	0.0287190
170000	125	0.0281221
171000	125	0.0277521

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
172000	125	0.0275677
173000	125	0.0276677
174000	125	0.0291437
175000	125	0.0292298
176000	125	0.0295571
177000	125	0.0297752
178000	125	0.0291237
179000	125	0.0287089
180000	125	0.0288552
181000	125	0.0281346
182000	125	0.0278074
183000	125	0.0274920
184000	125	0.0272845
185000	125	0.0276268
186000	125	0.0278489
187000	125	0.0283912
188000	125	0.0286614
189000	125	0.0283097
190000	125	0.0280221
191000	125	0.0271545
192000	125	0.0270877
193000	125	0.0269277
194000	125	0.0263055
195000	125	0.0258951
196000	125	0.0263374

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
197000	125	0.0254375
198000	125	0.0257136
199000	125	0.0260499
200000	125	0.0266048
201000	125	0.0260801
202000	125	0.0262457
203000	125	0.0263546
204000	125	0.0258974
205000	125	0.0251279
206000	24	0.0245835
207000	24	0.0247764
208000	125	0.0247668
209000	125	0.0252387
210000	125	0.0254998
211000	125	0.0252915
212000	125	0.0246360
213000	125	0.0244368
214000	24	0.0242236
215000	24	0.0237116
216000	24	0.0237973
217000	24	0.0237411
218000	24	0.0243256
219000	24	0.0240737
220000	24	0.0239775
221000	24	0.0239636

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
222000	24	0.0247847
223000	24	0.0251253
224000	24	0.0249496
225000	24	0.0241849
226000	24	0.0237708
227000	24	0.0241817
228000	24	0.0248917
229000	24	0.0255664
230000	24	0.0250131
231000	24	0.0254449
232000	24	0.0256372
233000	24	0.0250285
234000	24	0.0250901
235000	24	0.0251891
236000	24	0.0254255
237000	24	0.0251076
238000	24	0.0256274
239000	24	0.0256094
240000	24	0.0250828
241000	24	0.0246391
242000	24	0.0244572
243000	24	0.0231556
244000	24	0.0228956
245000	24	0.0221309
246000	24	0.0218709

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
247000	125	0.0216468
248000	198	0.0218616
249000	24	0.0215871
250000	125	0.0216885
251000	24	0.0223530
252000	24	0.0225171
253000	24	0.0228232
254000	24	0.0223507
255000	24	0.0221046
256000	24	0.0222071
257000	24	0.0219795
258000	24	0.0220725
259000	24	0.0222751
260000	24	0.0224321
261000	24	0.0231369
262000	24	0.0229846
263000	24	0.0230508
264000	24	0.0217960
265000	24	0.0214536
266000	24	0.0213509
267000	24	0.0213913
268000	113	0.0213593
269000	113	0.0218494
270000	24	0.0215720
271000	24	0.0212413

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
272000	24	0.0214930
273000	24	0.0217560
274000	113	0.0216302
275000	113	0.0213705
276000	24	0.0209745
277000	24	0.0209677
278000	24	0.0208001
279000	24	0.0204962
280000	24	0.0200265
281000	24	0.0203590
282000	24	0.0208510
283000	24	0.0208611
284000	24	0.0210604
285000	24	0.0206407
286000	24	0.0198153
287000	24	0.0197201
288000	24	0.0197090
289000	24	0.0199253
290000	24	0.0199620
291000	24	0.0197397
292000	229	0.0196939
293000	24	0.0196472
294000	229	0.0194854
295000	229	0.0196892
296000	229	0.0200776

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
297000	229	0.0199288
298000	229	0.0198157
299000	24	0.0199736
300000	24	0.0202403
301000	24	0.0199633
302000	229	0.0196468
303000	229	0.0198210
304000	229	0.0197098
305000	229	0.0198487
306000	229	0.0200089
307000	229	0.0199732
308000	229	0.0204115
309000	229	0.0208150
310000	229	0.0206295
311000	229	0.0202104
312000	229	0.0206910
313000	229	0.0205220
314000	229	0.0208565
315000	229	0.0208949
316000	229	0.0211505
317000	229	0.0210075
318000	229	0.0208002
319000	229	0.0206268
320000	229	0.0209402
321000	229	0.0205891

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
322000	55	0.0207690
323000	55	0.0208771
324000	55	0.0206159
325000	55	0.0208833
326000	55	0.0207281
327000	55	0.0207306
328000	229	0.0206982
329000	229	0.0212321
330000	229	0.0214438
331000	229	0.0218306
332000	55	0.0218117
333000	55	0.0217869
334000	55	0.0216798
335000	55	0.0217749
336000	55	0.0218825
337000	55	0.0223706
338000	55	0.0220895
339000	55	0.0222558
340000	55	0.0220959
341000	55	0.0222145
342000	55	0.0221357
343000	55	0.0220782
344000	55	0.0220295
345000	55	0.0217248
346000	55	0.0215112

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
347000	55	0.0214448
348000	55	0.0216312
349000	55	0.0216876
350000	55	0.0218882
351000	55	0.0223945
352000	55	0.0221346
353000	55	0.0220795
354000	55	0.0220083
355000	55	0.0221257
356000	55	0.0222797
357000	55	0.0220011
358000	55	0.0218257
359000	55	0.0220595
360000	55	0.0218837
361000	55	0.0221139
362000	55	0.0221623
363000	55	0.0221749
364000	55	0.0224645
365000	55	0.0223872
366000	55	0.0222405
367000	55	0.0221726
368000	55	0.0221020
369000	55	0.0223520
370000	55	0.0224420
371000	55	0.0220249

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
372000	55	0.0216910
373000	55	0.0215215
374000	55	0.0216578
375000	55	0.0216308
376000	55	0.0218414
377000	55	0.0219150
378000	55	0.0219368
379000	55	0.0219999
380000	55	0.0218310
381000	55	0.0221663
382000	55	0.0223279
383000	55	0.0226135
384000	55	0.0223150
385000	55	0.0224520
386000	55	0.0224363
387000	55	0.0228021
388000	55	0.0229445
389000	55	0.0232717
390000	55	0.0233513
391000	55	0.0239379
392000	55	0.0241630
393000	55	0.0241718
394000	55	0.0242756
395000	55	0.0243793
396000	55	0.0240023

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
397000	55	0.0239537
398000	55	0.0236963
399000	55	0.0233214
400000	55	0.0234706
401000	55	0.0233352
402000	55	0.0230254
403000	55	0.0230326
404000	55	0.0231034
405000	55	0.0232212
406000	55	0.0230188
407000	55	0.0230195
408000	55	0.0231459
409000	55	0.0231887
410000	55	0.0230927
411000	55	0.0231073
412000	55	0.0233436
413000	55	0.0233997
414000	55	0.0237388
415000	55	0.0237687
416000	55	0.0239607
417000	55	0.0238959
418000	55	0.0234179
419000	55	0.0237562
420000	55	0.0238315
421000	55	0.0237402

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
422000	55	0.0235916
423000	55	0.0236014
424000	55	0.0233059
425000	55	0.0230491
426000	55	0.0230564
427000	55	0.0230940
428000	55	0.0230110
429000	55	0.0230557
430000	55	0.0229402
431000	55	0.0229618
432000	55	0.0230116
433000	55	0.0230239
434000	55	0.0230180
435000	55	0.0231708
436000	55	0.0230037
437000	55	0.0229208
438000	55	0.0228704
439000	55	0.0231334
440000	55	0.0232334
441000	55	0.0232192
442000	55	0.0234457
443000	55	0.0234220
444000	55	0.0234055
445000	55	0.0235141
446000	55	0.0234993

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
447000	55	0.0233849
448000	55	0.0233191
449000	55	0.0233027
450000	55	0.0232371
451000	55	0.0229592
452000	55	0.0226528
453000	55	0.0231291
454000	55	0.0229982
455000	55	0.0226765
456000	55	0.0227310
457000	55	0.0226068
458000	55	0.0224857
459000	55	0.0226412
460000	55	0.0225684
461000	55	0.0222587
462000	55	0.0223807
463000	55	0.0230071
464000	55	0.0228478
465000	55	0.0226159
466000	55	0.0227177
467000	55	0.0228161
468000	55	0.0223049
469000	55	0.0223177
470000	55	0.0225707
471000	55	0.0223031

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
472000	55	0.0223659
473000	55	0.0220573
474000	55	0.0219455
475000	55	0.0215685
476000	55	0.0218031
477000	55	0.0218389
478000	55	0.0218350
479000	55	0.0218537
480000	55	0.0215342
481000	55	0.0216015
482000	55	0.0216218
483000	55	0.0216055
484000	55	0.0220969
485000	55	0.0223460
486000	55	0.0224792
487000	55	0.0226021
488000	55	0.0225726
489000	55	0.0224884
490000	55	0.0227462
491000	55	0.0228085
492000	55	0.0227225
493000	55	0.0228811
494000	55	0.0228604
495000	55	0.0229069
496000	55	0.0228947

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
497000	55	0.0228236
498000	55	0.0228959
499000	55	0.0228987
500000	55	0.0227826
501000	55	0.0230496
502000	55	0.0227087
503000	55	0.0226896
504000	55	0.0226136
505000	55	0.0226715
506000	55	0.0224303
507000	55	0.0224457
508000	55	0.0224758
509000	55	0.0225994
510000	55	0.0225171
511000	55	0.0225612
512000	55	0.0226682
513000	55	0.0227632
514000	55	0.0228389
515000	55	0.0228103
516000	55	0.0229536
517000	55	0.0226491
518000	55	0.0225654
519000	55	0.0226624
520000	55	0.0225522
521000	55	0.0223820

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
522000	55	0.0223008
523000	55	0.0222721
524000	55	0.0221730
525000	55	0.0221412
526000	55	0.0217941
527000	55	0.0220037
528000	55	0.0221141
529000	55	0.0219386
530000	55	0.0220872
531000	55	0.0221661
532000	55	0.0220153
533000	55	0.0219078
534000	55	0.0218308
535000	55	0.0219328
536000	55	0.0217187
537000	55	0.0215033
538000	55	0.0211548
539000	55	0.0211142
540000	55	0.0210562
541000	55	0.0212899
542000	55	0.0214787
543000	55	0.0215879
544000	55	0.0213474
545000	55	0.0211755
546000	55	0.0211922

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
547000	55	0.0213414
548000	55	0.0215083
549000	55	0.0217715
550000	55	0.0217808
551000	55	0.0219963
552000	55	0.0219093
553000	55	0.0219472
554000	55	0.0220414
555000	55	0.0219047
556000	55	0.0218647
557000	55	0.0217703
558000	55	0.0216374
559000	55	0.0214333
560000	55	0.0215560
561000	55	0.0214722
562000	55	0.0216013
563000	55	0.0219330
564000	55	0.0223188
565000	55	0.0223341
566000	55	0.0226094
567000	55	0.0226072
568000	55	0.0226010
569000	55	0.0227757
570000	55	0.0228269
571000	55	0.0227803

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
572000	55	0.0226850
573000	55	0.0224587
574000	55	0.0225482
575000	55	0.0221905
576000	55	0.0222250
577000	55	0.0218607
578000	55	0.0218837
579000	55	0.0217405
580000	55	0.0218165
581000	55	0.0219392
582000	55	0.0219871
583000	55	0.0219425
584000	55	0.0219708
585000	55	0.0220695
586000	55	0.0218843
587000	55	0.0220913
588000	55	0.0222594
589000	55	0.0223136
590000	55	0.0225650
591000	55	0.0227302
592000	55	0.0225318
593000	55	0.0227376
594000	55	0.0228337
595000	55	0.0229801
596000	55	0.0228682

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
597000	55	0.0227811
598000	55	0.0227969
599000	55	0.0228563
600000	55	0.0230683
601000	55	0.0231087
602000	55	0.0231731
603000	55	0.0233245
604000	55	0.0234199
605000	55	0.0232844
606000	55	0.0231278
607000	55	0.0231720
608000	55	0.0229948
609000	55	0.0228187
610000	55	0.0229781
611000	55	0.0231629
612000	55	0.0231795
613000	55	0.0232121
614000	55	0.0230083
615000	55	0.0230498
616000	55	0.0230516
617000	55	0.0226402
618000	55	0.0226550
619000	55	0.0228322
620000	55	0.0227171
621000	55	0.0227997

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
622000	55	0.0228952
623000	55	0.0229424
624000	55	0.0231936
625000	55	0.0231594
626000	55	0.0232569
627000	55	0.0231145
628000	55	0.0231023
629000	55	0.0229381
630000	55	0.0227285
631000	55	0.0228452
632000	55	0.0229047
633000	55	0.0229856
634000	55	0.0233396
635000	55	0.0234676
636000	55	0.0235362
637000	55	0.0235755
638000	55	0.0234514
639000	55	0.0234729
640000	55	0.0235233
641000	55	0.0234755
642000	55	0.0236967
643000	55	0.0238389
644000	55	0.0238730
645000	55	0.0241038
646000	55	0.0241206

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
647000	55	0.0239100
648000	55	0.0237463
649000	55	0.0239280
650000	55	0.0239636
651000	55	0.0239914
652000	55	0.0240062
653000	55	0.0240729
654000	55	0.0241318
655000	55	0.0240166
656000	55	0.0239436
657000	55	0.0238965
658000	55	0.0240353
659000	55	0.0240496
660000	55	0.0240956
661000	55	0.0243200
662000	55	0.0244447
663000	55	0.0243435
664000	55	0.0244033
665000	55	0.0241688
666000	55	0.0240757
667000	55	0.0239031
668000	55	0.0237516
669000	55	0.0238108
670000	55	0.0238812
671000	55	0.0239931

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
672000	55	0.0240523
673000	55	0.0240444
674000	55	0.0239572
675000	55	0.0239704
676000	55	0.0238484
677000	55	0.0235509
678000	55	0.0234045
679000	55	0.0234024
680000	55	0.0234343
681000	55	0.0232026
682000	55	0.0234332
683000	55	0.0233939
684000	55	0.0234660
685000	55	0.0234866
686000	55	0.0231471
687000	55	0.0231934
688000	55	0.0232617
689000	55	0.0232258
690000	55	0.0231364
691000	55	0.0232283
692000	55	0.0231072
693000	55	0.0232464
694000	55	0.0232144
695000	55	0.0232073
696000	55	0.0233805

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
697000	55	0.0233881
698000	55	0.0232461
699000	55	0.0230926
700000	55	0.0232006
701000	55	0.0231249
702000	55	0.0232489
703000	55	0.0233417
704000	55	0.0234226
705000	55	0.0235472
706000	55	0.0235346
707000	55	0.0233754
708000	55	0.0231544
709000	55	0.0233531
710000	55	0.0234396
711000	55	0.0232269
712000	55	0.0234352
713000	55	0.0234541
714000	55	0.0235668
715000	55	0.0235591
716000	55	0.0235458
717000	55	0.0236515
718000	55	0.0237732
719000	55	0.0238728
720000	55	0.0237836
721000	55	0.0239790

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
722000	55	0.0240992
723000	55	0.0238232
724000	55	0.0237860
725000	55	0.0240105
726000	55	0.0240203
727000	55	0.0240424
728000	55	0.0240608
729000	55	0.0240656
730000	55	0.0243310
731000	55	0.0244402
732000	55	0.0245026
733000	55	0.0244404
734000	55	0.0245753
735000	55	0.0244084
736000	55	0.0245404
737000	55	0.0243670
738000	55	0.0244719
739000	55	0.0244948
740000	55	0.0244064
741000	55	0.0242937
742000	55	0.0242620
743000	55	0.0242333
744000	55	0.0242874
745000	55	0.0242240
746000	55	0.0240976

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
747000	55	0.0240453
748000	55	0.0243117
749000	55	0.0242743
750000	55	0.0241840
751000	55	0.0241468
752000	55	0.0241667
753000	55	0.0240898
754000	55	0.0239337
755000	55	0.0238944
756000	55	0.0239291
757000	55	0.0237617
758000	55	0.0236625
759000	55	0.0236655
760000	55	0.0239088
761000	55	0.0239018
762000	55	0.0237480
763000	55	0.0238418
764000	55	0.0239636
765000	55	0.0239694
766000	55	0.0240804
767000	55	0.0238987
768000	55	0.0237583
769000	55	0.0237099
770000	55	0.0236951
771000	55	0.0237075

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
772000	55	0.0236823
773000	55	0.0235986
774000	55	0.0235174
775000	55	0.0235360
776000	55	0.0236109
777000	55	0.0234234
778000	55	0.0235185
779000	55	0.0233714
780000	55	0.0234807
781000	55	0.0234866
782000	55	0.0234880
783000	55	0.0234812
784000	55	0.0235509
785000	55	0.0236515
786000	55	0.0236503
787000	55	0.0237584
788000	55	0.0236717
789000	55	0.0237603
790000	55	0.0236194
791000	55	0.0238536
792000	55	0.0238561
793000	55	0.0238031
794000	55	0.0239408
795000	55	0.0238575
796000	55	0.0238053

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
797000	55	0.0239255
798000	55	0.0238762
799000	55	0.0237332
800000	55	0.0237255
801000	55	0.0236927
802000	55	0.0236133
803000	55	0.0236426
804000	55	0.0236700
805000	55	0.0235764
806000	55	0.0237045
807000	55	0.0236769
808000	55	0.0234635
809000	55	0.0235421
810000	55	0.0234485
811000	55	0.0233473
812000	55	0.0234858
813000	55	0.0233426
814000	55	0.0231738
815000	55	0.0231832
816000	55	0.0232449
817000	55	0.0232205
818000	55	0.0232821
819000	55	0.0231954
820000	55	0.0231902
821000	55	0.0232414

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
822000	55	0.0233720
823000	55	0.0235225
824000	55	0.0236230
825000	55	0.0235111
826000	55	0.0234112
827000	55	0.0233830
828000	55	0.0232918
829000	55	0.0231659
830000	55	0.0232540
831000	55	0.0232876
832000	55	0.0232665
833000	55	0.0233810
834000	55	0.0233733
835000	55	0.0235672
836000	55	0.0235497
837000	55	0.0236442
838000	55	0.0236199
839000	55	0.0235480
840000	55	0.0236317
841000	55	0.0234359
842000	55	0.0235311
843000	55	0.0235977
844000	55	0.0235520
845000	55	0.0236138
846000	55	0.0235512

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
847000	55	0.0235428
848000	55	0.0235835
849000	55	0.0234947
850000	55	0.0235558
851000	55	0.0234952
852000	55	0.0236776
853000	55	0.0236028
854000	55	0.0235475
855000	55	0.0234832
856000	55	0.0233600
857000	55	0.0233232
858000	55	0.0232326
859000	55	0.0232118
860000	55	0.0234756
861000	55	0.0235324
862000	55	0.0236168
863000	55	0.0236407
864000	55	0.0236369
865000	55	0.0236786
866000	55	0.0235739
867000	55	0.0233832
868000	55	0.0231632
869000	55	0.0232710
870000	55	0.0233687
871000	55	0.0233418

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
872000	55	0.0233605
873000	55	0.0235022
874000	55	0.0235241
875000	55	0.0233553
876000	55	0.0233502
877000	55	0.0233903
878000	55	0.0234680
879000	55	0.0235384
880000	55	0.0236377
881000	55	0.0235877
882000	55	0.0236188
883000	55	0.0235467
884000	55	0.0235441
885000	55	0.0234065
886000	55	0.0233904
887000	55	0.0235787
888000	55	0.0235087
889000	55	0.0235353
890000	55	0.0235128
891000	55	0.0235904
892000	55	0.0235699
893000	55	0.0234802
894000	55	0.0235128
895000	55	0.0235203
896000	55	0.0233965

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
897000	55	0.0231740
898000	55	0.0231272
899000	55	0.0231249
900000	55	0.0231913
901000	55	0.0230778
902000	55	0.0230895
903000	55	0.0231124
904000	55	0.0230406
905000	55	0.0229261
906000	55	0.0230405
907000	55	0.0229102
908000	55	0.0227871
909000	55	0.0228716
910000	55	0.0229370
911000	55	0.0230202
912000	55	0.0230052
913000	55	0.0229561
914000	55	0.0228839
915000	55	0.0229717
916000	55	0.0229945
917000	55	0.0228082
918000	55	0.0227637
919000	55	0.0227433
920000	55	0.0227323
921000	55	0.0226642

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
922000	55	0.0227366
923000	55	0.0226116
924000	55	0.0226218
925000	55	0.0225490
926000	55	0.0227553
927000	55	0.0225934
928000	55	0.0225345
929000	55	0.0226190
930000	55	0.0226316
931000	55	0.0225449
932000	55	0.0225108
933000	55	0.0224592
934000	55	0.0224098
935000	55	0.0224335
936000	55	0.0224549
937000	55	0.0224357
938000	55	0.0223296
939000	55	0.0222871
940000	55	0.0223180
941000	55	0.0222505
942000	55	0.0223439
943000	55	0.0222692
944000	55	0.0222578
945000	55	0.0221808
946000	55	0.0221179

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
947000	55	0.0222285
948000	55	0.0222113
949000	55	0.0222151
950000	55	0.0221704
951000	55	0.0221381
952000	55	0.0220697
953000	55	0.0220132
954000	55	0.0219369
955000	55	0.0219256
956000	55	0.0218211
957000	55	0.0218643
958000	55	0.0218604
959000	55	0.0220530
960000	55	0.0219090
961000	55	0.0218173
962000	55	0.0217987
963000	55	0.0219435
964000	55	0.0219593
965000	55	0.0218910
966000	55	0.0219860
967000	55	0.0219453
968000	55	0.0219114
969000	55	0.0219574
970000	55	0.0219392
971000	55	0.0219915

Continued on next page. . .

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
972000	55	0.0219212
973000	55	0.0217792
974000	55	0.0219482
975000	55	0.0221158
976000	55	0.0221622
977000	55	0.0220344
978000	55	0.0219768
979000	55	0.0219030
980000	55	0.0219602
981000	55	0.0219804
982000	55	0.0220225
983000	55	0.0219750
984000	55	0.0220859
985000	55	0.0221643
986000	55	0.0222136
987000	55	0.0223513
988000	55	0.0223944
989000	55	0.0223372
990000	55	0.0224221
991000	55	0.0225230
992000	55	0.0225839
993000	55	0.0224223
994000	55	0.0225553
995000	55	0.0225747
996000	55	0.0226124

Continued on next page...

Table B.8 – continued from previous page

number of traces	subkey with highest correlation	correlation value
997000	55	0.0227537
998000	55	0.0227757
999000	55	0.0227587
1000000	55	0.0227692

B.3 AES

DPA results from an unprotected AES implementation are presented here.

B.3.1 AES with Hamming Weight Hypothesis and Partition Function of 4

Table B.9: Different number of traces and the subkey with highest correlation value for unprotected AES using Hamming weight hypothesis and partition function of 4

number of traces	subkey with highest correlation	correlation value
1	255	0.6758750
2	253	1.0351033
3	253	1.1191625
4	253	1.1593580
5	253	1.2122250
6	251	1.2562100
7	251	1.2538525

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
8	251	1.2536889
9	251	1.2665570
10	251	1.3114655
11	251	1.3304925
12	235	1.3579862
13	235	1.3539579
14	235	1.3481587
15	203	1.3637250
16	63	1.3654182
17	63	1.3936950
18	12	1.3969879
19	12	1.3950910
20	12	1.3943971
21	12	1.4160432
22	12	1.4184161
23	12	1.4309512
24	12	1.4292568
25	12	1.4254550
26	12	1.4234270
27	12	1.4176029
28	12	1.4264476
29	12	1.4360847
30	12	1.4451339
31	12	0.7429189
32	12	0.7431372

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
33	7	0.4394990
34	7	0.4336728
35	7	0.4372876
36	7	0.4321726
37	7	0.4326957
38	7	0.4307614
39	7	0.4400395
40	12	0.3637262
41	12	0.3631307
42	12	0.3235807
43	12	0.3261257
44	12	0.2503319
45	202	0.2487427
46	143	0.2459191
47	7	0.2423425
48	118	0.2386479
49	143	0.2420172
50	143	0.2377949
51	143	0.2357187
52	143	0.2457941
53	143	0.2503964
54	118	0.2452437
55	187	0.2311450
56	187	0.2509266
57	187	0.2346607

Continued on next page. . .

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
58	187	0.2358395
59	187	0.2386027
60	143	0.2275443
61	143	0.2331801
62	143	0.2322681
63	187	0.2425794
64	187	0.2405627
65	187	0.2415246
66	187	0.2358835
67	187	0.2141718
68	143	0.2007716
69	143	0.2101496
70	143	0.2031910
71	143	0.1947658
72	143	0.1908288
73	143	0.1915637
74	71	0.1898652
75	219	0.1767728
76	219	0.1771314
77	219	0.1815394
78	219	0.1755060
79	127	0.1768228
80	127	0.1762851
81	127	0.1728932
82	127	0.1768010

Continued on next page. . .

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
83	219	0.1690609
84	127	0.1722271
85	127	0.1739762
86	127	0.1840075
87	219	0.1773703
88	127	0.1775180
89	127	0.1749925
90	127	0.1799524
91	127	0.1807868
92	127	0.1801107
93	127	0.1725874
94	127	0.1732246
95	127	0.1815166
96	127	0.1778331
97	127	0.1664960
98	127	0.1724545
99	127	0.1666192
100	74	0.1802946
101	74	0.1841296
102	74	0.1778614
103	74	0.1758622
104	74	0.1790654
105	74	0.1927517
106	74	0.1933392
107	74	0.1929603

Continued on next page. . .

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
108	74	0.1936024
109	74	0.2016625
110	74	0.2048862
111	74	0.2023200
112	74	0.2013189
113	74	0.1985370
114	74	0.1999572
115	74	0.1999185
116	74	0.2005384
117	74	0.2064412
118	74	0.1999775
119	74	0.1995589
120	74	0.2039623
121	74	0.2044964
122	74	0.1965770
123	74	0.1965466
124	74	0.1962712
125	74	0.1974360
126	74	0.1903870
127	74	0.1904489
128	74	0.1850362
129	74	0.1849990
130	74	0.1736224
131	74	0.1728044
132	74	0.1757530

Continued on next page. . .

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
133	74	0.1773746
134	74	0.1727366
135	74	0.1731759
136	74	0.1752731
137	74	0.1764341
138	74	0.1833671
139	74	0.1831146
140	74	0.1829893
141	74	0.1841158
142	74	0.1840747
143	74	0.1794033
144	74	0.1753723
145	74	0.1746202
146	74	0.1653067
147	74	0.1655105
148	74	0.1674302
149	74	0.1626814
150	74	0.1628953
151	74	0.1608651
152	74	0.1608929
153	74	0.1593175
154	130	0.1575456
155	74	0.1550866
156	130	0.1546735
157	130	0.1570533

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
158	130	0.1549847
159	130	0.1502595
160	207	0.1502523
161	207	0.1501145
162	207	0.1506643
163	207	0.1507927
164	207	0.1506798
165	207	0.1501027
166	207	0.1537547
167	207	0.1506503
168	207	0.1519700
169	207	0.1526288
170	207	0.1542613
171	207	0.1548295
172	207	0.1527021
173	207	0.1541045
174	207	0.1540347
175	207	0.1520896
176	207	0.1504668
177	207	0.1503730
178	130	0.1510642
179	130	0.1533268
180	130	0.1541372
181	130	0.1552816
182	130	0.1547044

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
183	130	0.1528105
184	130	0.1480359
185	55	0.1482863
186	207	0.1488339
187	207	0.1477876
188	207	0.1484644
189	55	0.1502245
190	55	0.1483035
191	55	0.1474328
192	130	0.1461076
193	130	0.1435286
194	130	0.1412316
195	55	0.1398395
196	130	0.1415853
197	130	0.1420224
198	130	0.1461644
199	130	0.1457538
200	130	0.1509892
201	130	0.1503682
202	130	0.1556449
203	130	0.1529034
204	130	0.1510262
205	130	0.1486238
206	130	0.1498581
207	130	0.1482608

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
208	130	0.1461196
209	130	0.1454608
210	130	0.1472858
211	130	0.1485989
212	130	0.1498534
213	130	0.1486492
214	130	0.1475874
215	130	0.1450107
216	130	0.1435337
217	130	0.1428901
218	130	0.1405639
219	130	0.1411633
220	130	0.1380882
221	55	0.1364736
222	55	0.1366161
223	55	0.1438833
224	55	0.1425170
225	130	0.1422681
226	130	0.1417496
227	130	0.1414196
228	130	0.1411823
229	130	0.1421313
230	130	0.1423911
231	130	0.1390693
232	55	0.1356278

Continued on next page...

Table B.9 – continued from previous page

number of traces	subkey with highest correlation	correlation value
233	55	0.1360938
234	55	0.1383433
235	55	0.1375430
236	55	0.1376478
237	55	0.1386513
238	55	0.1400772
239	55	0.1390974
240	55	0.1391016
241	55	0.1369315
242	55	0.1397030
243	55	0.1371697
244	55	0.1356038
245	55	0.1379631
246	55	0.1363991
247	55	0.1383068
248	55	0.1376121
249	55	0.1369554
250	55	0.1402948
251	55	0.1420217
252	55	0.1398381
253	55	0.1385114
254	55	0.1384862
255	55	0.1405477
256	55	0.1385977
257	55	0.1405170

Continued on next page...

Table B.9 – continued from previous page

number of traces	**subkey with highest correlation**	**correlation value**
258	55	0.1405461
259	55	0.1413527
260	55	0.1408701
261	55	0.1440162
262	55	0.1439708
263	55	0.1463417
264	55	0.1461975
265	55	0.1483996
266	55	0.1462930
19800	55	0.1140298
19900	55	0.1143453
20000	55	0.1144660

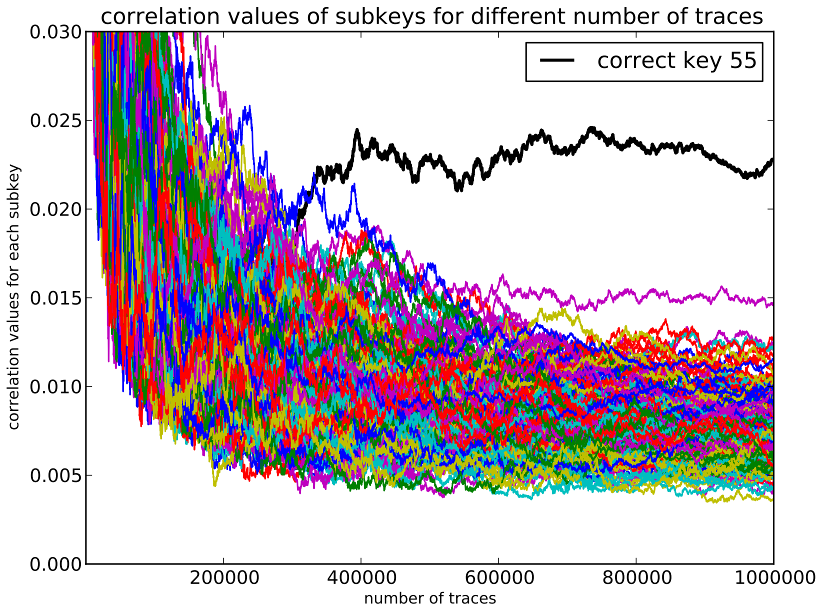


FIGURE B.8: DPA result for all encryption rounds on FPGA Implementation of dual rail path switching and alternating spacer AES Sbox for Hamming weight hypothesis and partition function of 6

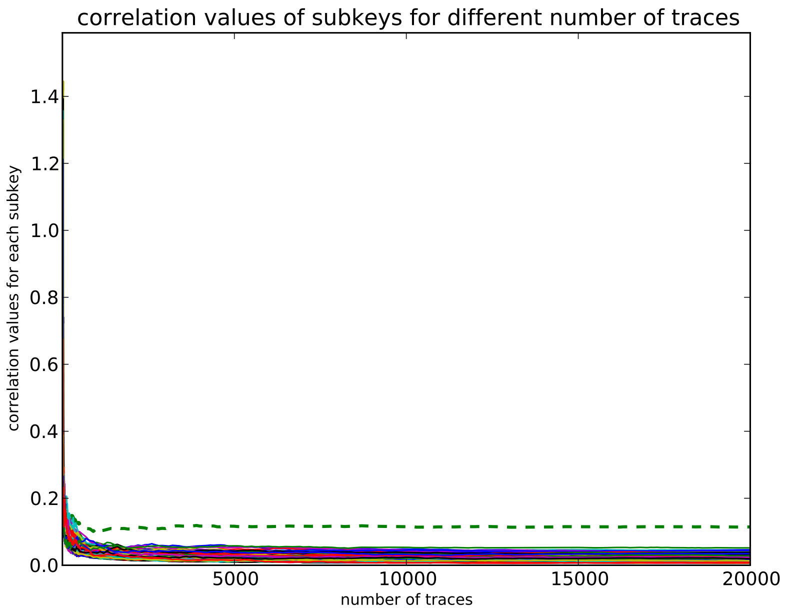


FIGURE B.9: DPA result for all encryption rounds on FPGA Implementation of AES without any countermeasure for Hamming weight hypothesis and partition function of 4

References

[1] M.-L. Akkar and C. Giraud, “An implementation of des and aes, secure against some attacks,” in *Cryptographic Hardware and Embedded Systems CHES 2001*, ser. Lecture Notes in Computer Science, e. Ko, D. Naccache, and C. Paar, Eds. Springer Berlin / Heidelberg, 2001, vol. 2162, pp. 309–318.

[2] K. Baddam and M. Zwolinski, “Divided Backend Duplication Methodology for Balanced Dual Rail Routing,” in *CHES ’08: Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, Elisabeth Oswald and Pankaj Rohatgi, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 396–410.

[3] K. Baddam and M. Zwolinski, “A Dual Rail Circuit Technique to Tolerate Routing Imbalances,” in *Proc. of Second International Workshop on Embedded Systems Security in conjunction with 7th Annual ACM International Conference on Embedded Software (EMSOFT 2007)*, Salzburg, Austria, Oct 2007.

[4] K. Baddam and M. Zwolinski, “Path switching: a technique to tolerate dual rail routing imbalances,” *Design Automation for Embedded Systems*, vol. Volume 12, pp. 207–220, 09 2008. [Online]. Available: http://www.springerlink.com/content/32181g28411w2121

[5] L. Benini, A. Macii, E. Macii, E. Omerbegovic, F. Pro, and M. Poncino, "Energy-aware design techniques for differential power analysis protection," in *DAC '03: Proceedings of the 40th conference on Design automation.* New York, NY, USA: ACM Press, 2003, pp. 36–41.

[6] S. Bhasin, S. Guilley, F. Flament, N. Selmane, and J.-L. Danger, "Countering early evaluation: an approach towards robust dual-rail precharge logic," in *Proceedings of the 5th Workshop on Embedded Systems Security*, ser. WESS '10. New York, NY, USA: ACM, 2010, pp. 6:1–6:8.

[7] G. F. Bouesse, M. Renaudin, S. Dumont, and F. Germain, "DPA on Quasi Delay Insensitive Asynchronous Circuits: Formalization and Improvement," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 424–429.

[8] M. Bucci, M. Guglielmo, R. Luzzi, and A. Trifiletti, "A power consumption randomization countermeasure for DPA-resistant cryptographic processors," *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation. 14th International Workshop, PATMOS 2004. Proceedings (Lecture Notes in Comput. Sci. Vol.3254)*, pp. 481–90, 2004.

[9] M. Bucci, R. Luzzi, M. Guglielmo, and A. Trifiletti, "A countermeasure against differential power analysis based on random delay insertion," in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on, 23-26 May 2005*, 2005, pp. 3547–3550.

[10] M. Bucci, L. Giancane, R. Luzzi, and A. Trifiletti, "Three-phase dual-rail pre-charge logic," in *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop*, L. Goubin and M. Matsui, Eds., vol. 4249. Springer, 2006, pp. 232–241. [Online]. Available: http://www.iacr.org/cryptodb/archive/2006/CHES/19/19.pdf

[11] A. Bystrov, D. Sokolov, A. Yakovlev, and A. Koelmans, “Balancing Power Signature in Secure Systems,” in *Proc. 14th UK Asynchronous Forum, 2003*, 2003.

[12] *Design Exchange Format (DEF)*, Cadence, Inc, April 2006, http://www.cadence.com.

[13] *Spectre*, Cadence, Inc, April 2006, http://www.cadence.com.

[14] *Ultrasim*, Cadence, Inc, April 2006, http://www.Cadence.com.

[15] S. Chari, J. R. Rao, and P. Rohatgi, “Template Attacks,” in *CHES ’02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2003, pp. 13–28.

[16] C. Clavier, J.-S. Coron, and N. Dabbous, “Differential Power Analysis in the Presence of Hardware Countermeasures,” in *CHES ’00: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2000, pp. 252–263.

[17] P. Cunningham, R. Anderson, R. Mullins, G. Taylor, and S. Moore, “Improving Smart Card Security Using Self-Timed Circuits,” in *ASYNC ’02: Proceedings of the 8th International Symposium on Asynchronus Circuits and Systems.* Washington, DC, USA: IEEE Computer Society, 2002, p. 211.

[18] W. Diffie and M. E. Hellman, “Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard,” *Computer*, vol. 10, pp. 74–84, June 1977. [Online]. Available: http://portal.acm.org/citation.cfm?id=1300749.1301665

[19] S. Eike and S. Frank, “Towards Activity Based System Level Power Estimation,” April 2006, http://www.us.design-reuse.com/articles/article12728.html Visited on 10 April 2006.

[20] K. Gandolfi, C. Mourtel, and F. Olivier, “Electromagnetic Analysis: Concrete Results,” in *CHES ’01: Proceedings of the Third International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2001, pp. 251–261.

[21] J. Goodwin, “Novel countermeasures and techniques for differential power analysis,” Ph.D. dissertation, ECS, University of Southampton, UK, 2009.

[22] S. Guilley, L. Sauvage, P. Hoogvorst, R. Pacalet, G. Bertoni, and S. Chaudhuri, “Security evaluation of wddl and seclib countermeasures against power attacks,” *Computers, IEEE Transactions on*, vol. 57, no. 11, pp. 1482 –1497, nov. 2008.

[23] S. Guilley, P. Hoogvorst, Y. Mathieu, and R. Pacalet, “The backend duplication method,” in *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, August 29 - September 1, 2005, Proceedings*, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 383–397. [Online]. Available: http://www.iacr.org/cryptodb/archive/2005/CHES/603/603.pdf

[24] S. Guilley, F. Flament, P. Hoogvorst, R. Pacalet, and Y. Mathieu, “Secured cad back-end flow for power-analysis-resistant cryptoprocessors,” *IEEE Design and Test of Computers*, vol. 24, pp. 546–555, 2007.

[25] M. Guiney and E. Leavitt, “An introduction to OpenAccess: an open source data model and API for IC design,” in *ASP-DAC ’06: Proceedings of the 2006 conference on Asia South Pacific design automation.* New York, NY, USA: ACM Press, 2006, pp. 434–436.

[26] J. Irwin, D. Page, and N. P. Smart, “Instruction Stream Mutation for Non-Deterministic Processors,” in *ASAP ’02: Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors.* Washington, DC, USA: IEEE Computer Society, 2002, pp. 286–295.

[27] Y. Ishai, A. Sahai, and D. Wagner, “Private circuits: Securing hardware against probing attacks,” in *Advances in Cryptology - CRYPTO 2003, Proceedings*, vol. 2729. Springer-Verlag, 2003, pp. 463–481.

[28] N. Koblitz, “Elliptic curve cryptosystems,” in *Mathematics of Computation*, vol. 48, 1987, pp. 203 – 209.

[29] P. Kocher, “Design and validation strategies for obtaining assurance in countermeasures to power analysis and related,” in *Attacks, in the proceedings of the NIST Physical Security Workshop*, 2005. [Online]. Available: http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper09.pdf

[30] P. C. Kocher, “Timing Attacks on Implementations of Diffie-Hellman RSA DSS and Other Systems,” in *CRYPTO ’96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1996, pp. 104–113.

[31] P. C. Kocher, J. Jaffe, and B. Jun, “Differential Power Analysis,” in *CRYPTO ’99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology.* London, UK: Springer-Verlag, 1999, pp. 388–397.

[32] O. Kömmerling and M. G. Kuhn, “Design Principles for Tamper-Resistant Smartcard Processors,” in *Proceedings of the USENIX Workshop on Smartcard Technology, Chicago, 10–11 May, 1999.*, 1999, pp. 9–20.

[33] O. Kömmerling and M. G. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," in *Proceedings of the USENIX Workshop on Smartcard Technology, Chicago, 10–11 May, 1999.*, 1999, pp. 9–20. [Online]. Available: citeseer.ist.psu.edu/kommerling99design.html

[34] K. Kulikowski, V. Venkataraman, Z. Wang, and A. Taubin, "Power balanced gates insensitive to routing capacitance mismatch," in *Design, Automation and Test in Europe, 2008. DATE '08*, march 2008, pp. 1280 –1285.

[35] K. J. Kulikowski, M. Su, A. Smirnov, A. Taubin, M. G. Karpovsky, and D. MacDonald, "Delay Insensitive Encoding and Power Analysis: A Balancing Act," in *11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05).*, vol. 00, 2005, pp. 116–125.

[36] K. J. Kulikowski, M. G. Karpovsky, and A. Taubin, "Power Attacks on Secure Hardware Based on Early Propagation of Data," in *IOLTS '06: Proceedings of the 12th IEEE International Symposium on On-Line Testing.* Washington, DC, USA: IEEE Computer Society, 2006, pp. 131–138.

[37] S. Kumar, C. Paar, J. Pelzl, G. Pfeiffer, and M. Schimmler, "Breaking Ciphers with COPACOBANA – A Cost-Optimized Parallel Code Breaker," in *IN WORKSHOP ON CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS, CHES 2006,YOKOHAMA.* Springer Verlag, 2006, pp. 101–118.

[38] L. Lin and B. Wayne, "Leakage-based differential power analysis (LDPA) on sub-90nm CMOS cryptosystems," in *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, Seattle, WA, USA,, May 2008, pp. 252–255.

[39] S. Mangard, “Securing Implementations of Block Ciphers against Side-Channel Attacks,” Ph.D. dissertation, IAIK, University of Technology Graz, Austria, 2004, http://www.iaik.tu-graz.ac.at/research/sca-lab/publications/abstracts/index.php.

[40] S. Mangard and K. Schramm, “Pinpointing the Side-Channel Leakage of Masked AES Hardware Implementations,” in *Cryptographic Hardware and Embedded Systems – CHES 2006, 8th International Workshop,Yokohama, Japan, October 10-13, 2006, Proceedings*, ser. Lecture Notes in Computer Science, Louis Goubin and Mitsuru Matsui, Eds., vol. 4249. Springer, 2006, pp. 76–90.

[41] S. Mangard, T. Popp, and B. M. Gammel, “Side-Channel Leakage of Masked CMOS Gates,” in *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, Alfred Menezes, Ed., vol. 3376. Springer, 2005, pp. 351–365.

[42] S. Mangard, N. Pramstaller, and E. Oswald, “Lecture Notes in Computer Science,” in *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, Scotland, August 29 - September 1, 2005, Proceedings*, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 157–171.

[43] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Reavealing the Secrets of Smart Cards.* Springer, 2007.

[44] C. Maxfield, *The Design Warrior's Guide to FPGAs.* Newnes, 2004.

[45] D. May, H. L. Muller, and N. P. Smart, “Random Register Renaming to Foil DPA,” in *Cryptographic Hardware and Embedded Systems CHES 2001, LNCS 2162*, C. K. Koc, D. Naccache, and C. Paar, Eds. Springer Verlag, May 2001, pp. 28–38.

[46] D. May, H. L. Muller, and N. P. Smart, "Non-deterministic Processors," in *ACISP '01: Proceedings of the 6th Australasian Conference on Information Security and Privacy.* London, UK: Springer-Verlag, 2001, pp. 115–129.

[47] R. Mayer Sommer, "Smartly Analyzing the Simplicity and the Power of Simple Power Analysis on Smartcards," in *CHES '00: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2000, pp. 78–92.

[48] Mentor Graphics, "ModelSim Simulator," 2006, http://www.model.com/products/default.asp.

[49] D. Mesquita, J.-D. Techer, L. Torres, G. Sassatelli, G. Cambon, M. Robert, and F. Moraes, "Current mask generation: a transistor level security against DPA attacks," in *SBCCI '05: Proceedings of the 18th annual symposium on Integrated circuits and system design.* New York, NY, USA: ACM Press, 2005, pp. 115–120.

[50] T. S. Messerges, "Using Second-Order Power Analysis to Attack DPA Resistant Software," in *CHES '00: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2000, pp. 238–251.

[51] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Examining Smart-Card Security under the Threat of Power Analysis Attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, 2002.

[52] V. S. Miller, "Use of elliptic curves in cryptography," in *CRYPTO 85*, 1985.

[53] R. Muresan, H. Vahedi, Y. Zhanrong, and S. Gregori, "Power-smart system-on-chip architecture for embedded cryptosystems," in *CODES+ISSS '05: Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2005, pp. 184–189.

[54] R. Muresan and C. Gebotys, “Current flattening in software and hardware for security applications,” in *CODES ISSS ’04: Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis.* New York, NY, USA: ACM Press, 2004, pp. 218–223.

[55] J. Murphy and A. Yakovlev, “An Alternating Spacer AES Crypto-processor,” in *Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedings of the 32nd European, Vol., Iss., Sept. 2006.*, 2006, pp. 126–129.

[56] M. Nassar, S. Bhasin, J.-L. Danger, G. Duc, and S. Guilley, “Bcdl: a high speed balanced dpl for fpga with global precharge and no early evaluation,” in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE ’10. 3001 Leuven, Belgium, Belgium: European Design and Automation Association, 2010, pp. 849–854.

[57] *Announcing the ADVANCED ENCRYPTION STANDARD (AES): FIPS Publication 197*, National Institute of Standards and Technology, 2001, http://csrc.nist.gov/CryptoToolkit/aes/. [Online]. Available: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[58] *Federal Information Processing Standards Publication 46-2*, National Institute of Standards and Technology, December 1993.

[59] *Announcing the Standard for DIGITAL SIGNATURE STANDARD (DSS): FIPS Publication 186*, National Institute of Standards and Technology, 1994, http://csrc.nist.gov/CryptoToolkit/aes/. [Online]. Available: http://www.itl.nist.gov/fipspubs/fip186.htm

[60] *Federal Information Processing Standards Publication 46-3*, National Institute of Standards and Technology, October 1999. [Online]. Available: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[61] B. Nikolic, V. G. Oklobdzija, V. Stojanovic, W. Jia, J. K.-S. Chiu, and M. M.-T. Leung, "Improved sense-amplifier-based flip-flop: design and measurements," *Solid-State Circuits, IEEE Journal of, Vol.35, Iss.6, Jun 2000 Pages:876-884*, 2000.

[62] S. B. Örs, F. K. Gürkaynak, E. Oswald, and B. Preneel, "Power-Analysis Attack on an ASIC AES implementation," in *ITCC '04: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04) Volume 2.* Washington, DC, USA: IEEE Computer Society, 2004, p. 546.

[63] D. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of aes," in *Topics in Cryptology CT-RSA 2006*, ser. Lecture Notes in Computer Science, D. Pointcheval, Ed. Springer Berlin / Heidelberg, 2006, vol. 3860, pp. 1–20.

[64] E. Oswald, "On Side-Channel Attacks and the Application of Algorithmic Countermeasures," Ph.D. dissertation, IAIK, University of Technology Graz, Austria, 2003, http://www.iaik.tu-graz.ac.at/research/sca-lab/publications/abstracts/index.php.

[65] E. Oswald, "Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems," in *Cryptographic Hardware and Embedded Systems – CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. Lecture Notes in Computer Science, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, Eds., vol. 2535. Springer, 2003, pp. 82–97.

[66] E. Oswald, S. Mangard, and N. Pramstaller, "Rijmen: A side-channel analysis resistant description of the aes s-box," in *Fast Software Encryption 2005, LNCS 3557.* Springer, 2005, pp. 413–423.

[67] D. Page, “Theoretical use of cache memory as a cryptanalytic side-channel,” Department of Computer Science, University of Bristol, Tech. Rep. CSTR-02-003, June 2002. [Online]. Available: http://www.cs.bris.ac.uk/Publications/Papers/1000625.pdf

[68] P. Pillai and K. G. Shin, “Real-time dynamic voltage scaling for low-power embedded operating systems,” in *SOSP ’01: Proceedings of the eighteenth ACM symposium on Operating systems principles.* New York, NY, USA: ACM Press, 2001, pp. 89–102.

[69] T. Popp and S. Mangard, “Masked Dual-Rail Pre-Charge Logic: DPA-Resistance without Routing Constraints,” in *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, Edinburgh, Scotland, August 29 - September 1, 2005, Proceedings*, ser. Lecture Notes in Computer Science, Josyula R. Rao and Berk Sunar, Eds., vol. 3659. Springer, 2005, pp. 172–186.

[70] T. Popp and S. Mangard, “Implementation Aspects of the DPA-Resistant Logic Style MDPL,” in *International Symposium on Circuits and Systems (ISCAS 2006), Island of Kos, Greece, May 21 - 24, 2006, Proceedings.* IEEE Computer Society, May 2006, pp. 2913–2916, iSBN 0-7803-9390-2.

[71] T. Popp, M. Kirschbaum, and S. Mangard, “Practical attacks on masked hardware,” in *Topics in Cryptology CT-RSA 2009*, ser. Lecture Notes in Computer Science, M. Fischlin, Ed. Springer Berlin / Heidelberg, 2009, vol. 5473, pp. 211–225.

[72] N. Pramstaller, “An AES ASIC-Implementation Resistant to Differential Power Analysis,” Master’s thesis, IAIK, University of Technology Graz, Austria., 2004, http://www.iaik.tu-graz.ac.at/research/sca-lab/publications/abstracts/index.php Downloaded on 16 July 2005.

[73] N. Pramstaller, E. Oswald, S. Mangard, F. K. Gürkaynak, and S. Haene, "A Masked AES ASIC Implementation," in *Austrochip 2004, Villach, Austria, Proceedings*, E. Ofner and M. Ley, Eds., October 2004, pp. 77–82.

[74] J.-J. Quisquater and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," in *E-SMART '01: Proceedings of the International Conference on Research in Smart Cards.* London, UK: Springer-Verlag, 2001, pp. 200–210.

[75] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits Second Edition.* Prentice Hall, 2003.

[76] G. B. Ratanpal, R. D. Williams, and T. N. Blalock, "An On-Chip Signal Suppression Countermeasure to Power Analysis Attacks," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 3, pp. 179–189, 2004.

[77] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in embedded systems: Design challenges," *Trans. on Embedded Computing Sys.*, vol. 3, no. 3, pp. 461–491, 2004.

[78] *Side-channel Attack Standard Evaluation Board*, Research Center for Information Security at National Institute of Advanced Industrial Science and Technology, Feb 2010, http://www.rcis.aist.go.jp/special/SASEBO/index-en.html.

[79] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," in *Communications of the ACM*, vol. 21 (2), 1978, p. 120126.

[80] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," in *ASIACRYPT '01:*

Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security. London, UK: Springer-Verlag, 2001, pp. 239–254.

[81] L. Sauvage, M. Nassar, S. Guilley, F. Flament, J.-L. Danger, and Y. Mathieu, "Exploiting dual-output programmable blocks to balance secure dual-rail logics," *Int. J. Reconfig. Comput.*, vol. 2010, pp. 5:1–5:35, February 2010.

[82] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C Second Edition.* John Wiley & Sons, 1996.

[83] *OpenAccess Coalition*, Si2, Org, April 2007, http://openeda.si2.org/.

[84] R. Soares, N. Calazans, V. Lomné, P. Maurine, L. Torres, and M. Robert, "Evaluating the robustness of secure triple track logic through prototyping," in *Proceedings of the 21st annual symposium on Integrated circuits and system design*, ser. SBCCI '08. New York, NY, USA: ACM, 2008, pp. 193–198.

[85] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Lecture Notes in Computer Science," in *Cryptographic Hardware and Embedded Systems – CHES 2004: 6th International Workshop, August 11-13, 2004. Proceedings*, M. Joye and J.-J. Quisquater, Eds., vol. 3156. Springer, 2004, pp. 282–297.

[86] D. Sokolov, J. Murphy, A. Bystrov, and A. Yakovlev, "Design and Analysis of Dual-Rail Circuits for Security Applications," *IEEE Transactions on Computers*, vol. 54, no. 4, pp. 449–460, 2005.

[87] M. Steinkogler, "The Power Consumption of Integrated Circuits in Simulation and Reality," Master's thesis, IAIK, University of Technology Graz, Austria., 2006, http://www.iaik.tu-graz.ac.at/research/sca-lab/publications/abstracts/index.php Downloaded on 16 July 2007.

[88] *Hspice reference manual*, Synopsys, Inc, April 2006, http://www.synopsys.com.

[89] *Nanosim user guide*, Synopsys, Inc, April 2006, http://www.synopsys.com.

[90] C. D. Systems, “ENCOUNTER DIGITAL IC DESIGN PLATFORM,” April 2007, http://www.cadence.com/products/digital_ic/index.aspx?lid=dic.

[91] K. Tiri, M. Akmal, and I. Verbauwhede, “A dynamic and differential cmos logic with signal independent power consumption to withstand differential power analysis on smart cards,” pp. 403 –406, sept. 2002.

[92] K. Tiri and I. Verbauwhede, “Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology,” in *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, 2003, pp. 125–136.

[93] K. Tiri and I. Verbauwhede, “Charge Recycling Sense Amplifier Based Logic: Securing Low Power Security ICs against Differential Power Analysis,” 2004, http://eprint.iacr.org/. [Online]. Available: CryptologyePrintArchive, Report2004/067

[94] K. Tiri and I. Verbauwhede, “A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation,” in *DATE ’04: Proceedings of the conference on Design, automation and test in Europe.* Washington, DC, USA: IEEE Computer Society, 2004, pp. 246–251.

[95] K. Tiri and I. Verbauwhede, “Place and Route for Secure Standard Cell Design,” in *6th International Conference on Smart Card Research and Advanced Applications (CARDIS 2004)*, August 2004, pp. 143–158.

[96] K. Tiri and I. Verbauwhede, “Synthesis of Secure FPGA Implementations,” in *International Workshop on Logic and Synthesis (IWLS 2004)*, June 2004, pp. 224–231.

[97] K. Tiri and I. Verbauwhede, “Secure logic synthesis,” in *Field Programmable Logic and Application.* Springler, 2004, pp. 1052–1056.

[98] K. Tiri and I. Verbauwhede, “A VLSI Design Flow for Secure Side-Channel Attack Resistant ICs,” in *DATE ’05: Proceedings of the conference on Design, Automation and Test in Europe.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 58–63.

[99] K. Tiri and I. Verbauwhede, “Prototype IC with WDDL and Differential Routing DPA Resistance Assessment,” in *Cryptographic Hardware and Embedded Systems – CHES 2005, 7th International Workshop, August 29 - September 1, 2005, Proceedings*, J. R. Rao and B. Sunar, Eds., vol. 3659. Springer, 2005, pp. 354–365.

[100] K. Tiri and I. Verbauwhede, “Simulation models for side-channel information leaks,” in *DAC ’05: Proceedings of the 42nd annual conference on Design automation.* New York, NY, USA: ACM, 2005, pp. 228–233.

[101] E. Trichina, “Combinational Logic Design for AES SubByte Transformation on Masked Data,” 2003, http://eprint.iacr.org/. [Online]. Available: CryptologyePrintArchive,Report2003/236

[102] E. Trichina, D. De Seta, and L. Germani, “Simplified adaptive multiplicative masking for aes,” in *Cryptographic Hardware and Embedded Systems - CHES 2002*, ser. Lecture Notes in Computer Science, B. Kaliski, e. Ko, and C. Paar, Eds. Springer Berlin / Heidelberg, 2003, vol. 2523, pp. 71–85.

[103] E. Trichina, T. Korkishko, and K. Lee, “Small size, low power, side channel-immune aes coprocessor: Design and synthesis results,” in *Advanced Encryption Standard AES*, ser. Lecture Notes in Computer Science, H. Dobbertin, V. Rijmen, and A. Sowa, Eds., vol. 3373. Springer, 2005, pp. 572–572.

[104] E. Valentini, E. Haselwanter, R. Ulmer, and T. Popp, “Configurable Logic Style Translation Based on an OpenAccess Engine,” in *12th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2005), December 11-14th 2005, Gammarth, Tunisia, Proceedings*, vol. 1. IEEE Computer Society, 2005, pp. 389–392.

[105] J. Waddle and D. Wagner, “Towards Efficient Second-Order Power Analysis,” in *CHES 04: Proceedings of the Sixth International Workshop on Cryptographic Hardware and Embedded Systems*, 2004, pp. 1–15.

[106] A. Wang and A. Chandrakasan, “A 180mV FFT processor using subthreshold circuit techniques,” *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, vol. 1, pp. 292–529, Feb 2004.

[107] S. H. Weingart, “Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defences,” in *CHES ’00: Proceedings of the Second International Workshop on Cryptographic Hardware and Embedded Systems.* London, UK: Springer-Verlag, 2000, pp. 302–317.

[108] N. Weste and D. Harris, *CMOS VLSI Design A Circuits and Systems Perspective (3rd Edition).* Addison Wesley, 2004.

[109] P. R. Wilson, *Design Recipes for FPGAs: Using Verilog and VHDL.* Newnes, 2007.

[110] Xilinx, “Xilinx Virtex AFX-BG560 Prototype Board,” 2007. [Online]. Available: http://www.xilinx.com/products/devkits/HW-AFX-BG560-100.htm

[111] *Xilinx FPGA*, Xilinx, Inc, April 2007, http://www.xilinx.com/.

[112] S. Yang, W. Wolf, N. Vijaykrishnan, D. N. Serpanos, and Y. Xie, “Power Attack Resistant Cryptosystem Design: A Dynamic Voltage and Frequency

Switching Approach," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe.* Washington, DC, USA: IEEE Computer Society, 2005, pp. 64–69.

[113] P. Yu and P. Schaumont, "Secure FPGA circuits using controlled placement and routing," in *CODES+ISSS '07: Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis.* New York, NY, USA: ACM, 2007, pp. 45–50.

[114] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and practical limits of dynamic voltage scaling," in *DAC '04: Proceedings of the 41st annual conference on Design automation.* New York, NY, USA: ACM Press, 2004, pp. 868–873.

[115] X. Zhang and K. K. Parhi, "High-speed VLSI architectures for the AES algorithm," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 9, pp. 957–967, 2004.