An Investigation into the Security of Self-timed Circuits

A thesis submitted to the University of Manchester

for the degree of Doctor of Philosophy in the

Faculty of Science & Engineering

2003

ZHONGCHUAN YU

Department of Computer Science

Contents

List of Figures

List of Tables

Abstract

The Differential Power Analysis (DPA) attack poses a great threat to the security of cryptosystems such as smartcards. Asynchronous design has the potential to improve the resistance to such attacks, and to other non-invasive attacks. The self-timed ARM-compatible SPA processor designed at Manchester aims to demonstrate these advantages for secure applications. The security evaluation of self-timed circuits becomes a crucial design task in demonstrating that the security has, indeed, been enhanced by these techniques.

This thesis introduces a methodology suitable for asynchronous power analysis which copes with the lack of a timing reference and other difficulties that do not affect the analysis for synchronous systems. Investigations into the security properties of asynchronous logic using this methodology are then presented.

As the Linear Feedback Shift Register (LFSR) is commonly used in cryptosystems, the LFSR was taken as a small example to demonstrate how self-timed circuits and dual-rail encoding can improve security. Six LFSRs were implemented using different design technologies with the objective to evaluate the impact on security of those technologies. This experiment offers an opportunity to understand the fundamental difference in power consumption characteristics between synchronous circuits and asynchronous circuits.

An investigation is also presented into the security of the SPA processor using both extracted layout simulation and silicon chip measurement. The power consumption of a single-rail and a dual-rail self-timed processor are analyzed and the security impact of using a novel, secure latch is also investigated. Finally, side-channel leakage from a DES encryption algorithm running on the SPA chip is analyzed to expand the investigation.

Together, the research demonstrates how to perform differential power analysis attacks on asynchronous circuits. The results reveal that the security of single-rail self-timed circuits is inadequate and that dual-rail self-timed circuits have high resistance to differential power analysis attacks. In particular, the work shows that secure latches make a vital contribution to the security of self-timed circuits.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

The Author

Zhongchuan Yu graduated from the Beijing University of Chemical Technology, China, in 1998, majoring in Automation Control. Upon completing this undergraduate study he worked at the R&D centre of SUPCON Ltd., a part of the National Engineering Research Centre for Industrial Automation, China, participating in the design of industrial processing distributed control systems (known as DCS).

Since 2000, he has been conducting his Ph.D. research in the APT group (formerly known as the AMULET group) at the Department of Computer Science at the University of Manchester. His work focuses on the investigation of the potential of asynchronous design to enhance the security of digital systems.

Acknowledgements

First, I would particularly like to thank my supervisor Professor Steve Furber without whose invaluable insights, guidance and support this work would not have been possible. Thanks also go to my advisor, Dr. Doug Edwards, and to Dr. Luis A. Plana for many insightful comments on many issues.

I would also like to thank Dr. Aris Efthymiou who endured many questions and gave many valuable comments on my thesis. Special thanks to my proof readers: Dr. John Bainbridge, Dr. Steve Temple, Peter Riocreux, Dr. Andrew Bardsley and Professor David Bree for proof reading and commenting on the draft of this thesis. Steve also provided assistance with the design tools.

I am grateful to Dr. Jim Garside for many valuable comments on my work, and to Dr. Viv Woods for his kind arrangement of presentations. I would also like to acknowledge other members in the group who shared their striking thoughts with me.

I gratefully acknowledge the *G3Card* project for offering me the opportunity to analyze the security of the SPA processor. Thanks also go to Dr. Ross Anderson for introducing me to background material on cryptography, especially stream ciphers.

My heartfelt thanks go to my fiancee Yi, for her love, encouragement and support which helped me through difficult times; and to my parents for their timeless support. Last, but not least, I would like to thank other friends here for sharing a good time.

To: *My Parents*

Yi

Chapter 1: Introduction

Today, more and more digital applications have become part of our daily life, such as on-line shopping, e-banking, on-line music and e-business etc. [1]. Many of these applications involve the transfer or use of information of high financial value or sensitivity that must be protected. Smartcards have seen a rapid deployment in some areas, such as banking and cable-television reception, as part of a secured platform using cryptography.

A smartcard chip usually consists of three principal components: a microprocessor capable of running software applications, a memory to store programs and data, and an I/O interface for communication with a host machine or card-reader. In some applications, smartcards may also contain a secure co-processor for handling the encryption algorithms. By implementing a whole encryption-protected system on a single CMOS die, with suitable tamper-protection circuits built-in, a smartcard provides a low-cost reliable approach to security. This is why they have been so successfully deployed in many applications, such as banking, access control, pre-paid telephone cards and health care. In China, the Chinese central government decided to replace the paper-based identity card with smartcards [2]. Another reason for their success is that multi-applications smartcards will be possible for use in a variety of areas. For instance, a personal identity card could be also used as a health record card and pre-paid transportation card.

The popularity of smartcards is essentially based on their security and convenience. However, one may ask “is the smartcard really secure against malicious attacks?”. Is the tamper-resistant hardware really secure and are the embedded cryptographic algorithms really unbreakable? Thus, cryptanalysis and the evaluation of security devices have become as important as the design of security devices themselves.

During World War II, German and Japanese cryptosystems were broken by attackers with knowledge of the underlying algorithms. From the 1960s, the use of extra information leaked from the side-channels of cryptosystems became another interesting source cryptanalysts use to mount attacks. In 1960, at the request of the British prime minister, the British intelligence agency, MI5, listened to the French embassy on the negotiations about joining the EEC. Attackers found an extra faint signal on the encrypted signal from the embassy. The leaked signal turned out to be the plain-text of the enciphered message [3].

Figure 1.1 shows a cryptosystem which leaks side channel information through hardware faults, power consumption fluctuation, timing information and electromagnetic emissions. Attacks have been successfully mounted on smartcards using such side-channel information to retrieve supposedly secret data: In 1996, Anderson and Kuhn indicated that many smartcards could be broken by inserting *glitches* in their power or clock lines [4, 5]. Kocher showed how secret information could be extracted by precisely measuring the different amount of time taken to process different data on common cryptosystems [6]. In 1998, Kocher again demonstrated that most smartcards on the market were vulnerable to differential power analysis attacks [7,8]. In 2000, a similar attack was demonstrated using electromagnetic leakage from the security device running an encryption algorithm [9, 10]. Unlike mathematical cryptanalysis attacks, these kinds of attacks exploit leaked information from the weakness of the physical implementation of the cryptosystem [11, 12] rather than algorithmic aspects, and are more difficult to

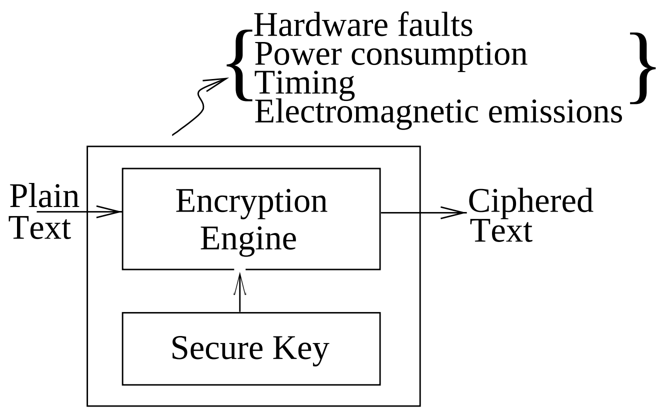


Figure 1.1 Information leakage in cryptosystems

defeat [24]. Among the attack techniques so far described, the differential power analysis (DPA) may be the most effective on smartcards and the most difficult to defeat.

The basic theory behind DPA is that the power consumption is related to the data being processed and, for a programmable machine, the instruction being executed. This data-dependency enables an attacker to obtain secret information by observing small fluctuations in the power consumption waveform and then using statistical analysis to amplify these small differences. The provision of a global clock signal in conventional clocked circuits provides a timing reference and allows statistical analysis to be carried out easily.

It has been suggested that smartcard security can be improved by using self-timed design technology [13, 14] where the absence of a global clock signal makes power analysis more difficult as there is no obvious timing reference, and the power consumption is less predictable because activity is no longer regulated by the clock. Other security advantages derived from not using a clock include immunity to clock and power glitches [14]. Furthermore dual-rail encoding, which is very often used in self-timed circuits, uses two wires to encode binary data. Transitions on one wire indicate a logic one and on the other wire a logic zero. With the return-to-zero signalling protocol described in chapter 3, the transmission of one bit of data always consumes two transitions, one rising and one falling, regardless of the data value, making the processing data-independent. Thus the power consumption could, in principle, be balanced so as to defeat differential power analysis attacks.

1.1 Research goal and contributions

The goal of this research is to investigate how and to what extent asynchronous design technologies improve a CMOS circuit’s resistance to differential power analysis attacks, and to investigate the potential of self-timed design in security applications. The main contributions of this work are highlighted as follows:

- The lack of a global clock signal in self-timed circuits creates several difficulties for power analysis making that conventional power analysis methods as used to analyze synchronous waveforms are not readily applicable to asynchronous waveforms. A suitable methodology for asynchronous power analysis is introduced to overcome these difficulties. The work described in this thesis shows how differential power analysis can be carried out on asynchronous circuits.

- The results show that single-rail self-timed circuits do not provide adequate security; their security is comparable to that of clocked circuits.

- The results confirm that dual-rail encoding is the key technology to defend against differential power analysis attacks and self-timed technologies offer extra security by making analysis much more difficult due to the lack of a timing reference.

- The security investigation also indicates that security in storage has a significant impact on security and are vital for security improvement.

- The research provides basic guidelines for applying asynchronous technologies to design secure applications in the future.

The following papers describing aspects of the work in this thesis have been published:

- *SPA - A Secure Amulet Core for Smartcard Applications* [15]; appeared in the Microprocessors and Microsystems Journal, 27/9; Co-author.

- *An investigation into the security of self-timed circuits* [16]; appeared in the proceedings of the 9th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC'03); Author.

- *Defeating power analysis attacks* [17]; appeared in the proceedings of the 9th UK Asynchronous Forum; Author.

- *Power analysis of linear feedback shift registers* [18]; appeared in the proceedings of the 10th UK Asynchronous Forum; Author.

1.2 Related work

This research has a close connection to the *G3Card* project, a joint project involving the University of Manchester, the University of Cambridge and other industrial and academic partners. Significant effort was focused on the security investigation of the SPA, a self-timed ARM-compatible processor designed within the *G3Card* project group in Manchester [19]. The XPS test chip designed by the Cambridge group consists of five 16-bit processors designed with different technologies [14]. Both groups aim to improve smartcard security using self-timed design technologies.

1.3 Thesis organization

This thesis is divided into nine chapters. The remainder of the thesis is organized as follows:

Chapter 2 provides background information on smartcards and cryptography, and smartcard security and attacks are discussed. The conventional differential power analysis attack and the theory of CMOS VLSI power consumption are also reviewed.

Chapter 3 introduces asynchronous VLSI design techniques. The fundamental differences between asynchronous and synchronous design and the impact of the absence of a global clock signal are addressed.

Chapter 4 discusses the difficulties of performing asynchronous power analysis and presents a suitable methodology. It handles those difficulties and applies correlation analysis to characterize asynchronous power waveforms.

Chapter 5 describes the design of six linear feedback shift registers implemented with different technologies to allow an evaluation of the security impacts of those technologies.

Chapter 6 analyses the power consumption of the LFSRs described in chapter 5 using the methodology discussed in chapter 4 and evaluates the impact of the different technologies.

Chapter 7 investigates the security of the SPA processor core. Several instructions are studied using extracted layout simulation to determine the security improvement of dual-rail encoding and secure latches.

Chapter 8 extends the investigation undertaken in chapter 7 by analyzing measurements taken from the silicon chip. A popular encryption algorithm, DES, is run on the SPA processor and the power consumption of DES is examined.

Finally, chapter 9 draws conclusions from the research presented in this thesis and makes suggestions for further work.

Chapter 2: Smartcard Security and DPA Attacks

2.1 Introduction

Smartcards have evolved since their introduction in the 1970s from simple memory cards to microprocessor-based devices with advanced cryptographic co-processors embedded on the card. Unlike magnetic strip cards, where only a few hundred bytes of information can be kept on the card, smartcards have sufficient memory to allow sophisticated encryption algorithms to be used on the card. The design of a microprocessor for use in smartcards, however, has to meet special limitations set in the industrial standard, ISO/IEC 7816 [20, 21]. The silicon area of the smartcard chip must not exceed 25 mm^2 (5 mm in length and 5 mm in width). At present, most smartcards use an 8-bit microprocessor, which is significantly less advanced than current general-purpose microprocessors. This is largely because of the physical die area limitation of a smartcard chip. However, the industry is starting to plan for the use of 32-bit RISC microprocessors in smartcards [22].

The principal reason for using smartcards is to increase security. However, many successful attacks have been made on smartcards, some of which have seriously damaged confidence in their security. A security system fails because of many reasons [23]. The availability of side-channel information leakage allows a wide range of attacks on smartcards [24 - 26]. Differential power analysis, one of the most successful attack methodologies on smartcards using leaked information, has become a real threat to security.

This chapter will first review basic techniques and the theory of cryptography and then review some common attacks on smartcards with an emphasis on differential power analysis (DPA).

2.2 Cryptography background

With today's increasing interest in electronic life, cryptography has been widely used in many applications, including smartcards. This section will briefly review some necessary aspects of cryptography. Further information about cryptography and security can be found in the literature [27-29].

2.2.1 Symmetric cryptography

A cryptosystem implemented with symmetric cryptography has one key used by both communicating parties, commonly referred to as Alice and Bob, to encrypt and decrypt data. For example, Alice encrypts message M with key K to yield a ciphered text $C=E(M, K)$. When Bob receives the ciphered text C, he uses the same key to decrypt the text to recover the original message, $M=D(C,K)$. One of the most famous and widely used symmetric cryptographic algorithms is the Data Encryption Standard (DES) algorithm [30]. The Advanced Encryption Standard (AES) [31] and other AES candidates [32-35] are also symmetric cryptographs.

One major problem with symmetric cryptography is the key distribution. As the same key is used for both encryption and decryption, it must be assured that the key has not been leaked to or stolen by any parties other than those authorized to know it. Thus, Alice has to give the key to Bob via a secure channel, avoiding leaking the key information, otherwise, the communication between Alice and Bob will not be secure. Therefore both Alice and Bob share the responsibility to keep the secret key safe and, furthermore, if Alice has ten friends to communicate with she perhaps needs to manage ten different keys. What if she has 100 people to communicate with? The problem can clearly grow to an unmanageable size. Thus key management is also an important issue in symmetric cryptography.

2.2.2 Asymmetric cryptography

Asymmetric cryptography, on the other hand, uses different keys for encryption and decryption. Bob generates a *private key*, which he keeps to himself in a secure place, and a *public key*, which he gives to anyone who wants to have secure communications with him. For instance, when Alice needs to send a ciphered message to Bob, she encrypts the message M with Bob's public key K_e to produce the ciphered text $C=E(M, K_e)$ and sends it to Bob. When Bob gets the ciphered message he uses his private key, K_d, to recover the original message $M=D(C, K_d)$. Although K_e and K_d are related, it is impossible to derive K_d from K_e.

The first and most famous public-key cryptography algorithm was the Diffie-Hellman key exchange protocol created by Diffie and Hellman in 1977, which was a milestone in the history of cryptography [36]. One of the main advantages of public-key cryptography is the ease of key management. Bob publishes his public key, on a web-site for example, so that anyone who wants to send secure messages to him can use this public key and Bob only needs to manage his one private key, no matter how many friends he needs to communicate with securely. Moreover, he does not necessarily need to keep his friends' public keys in a secure place which is much easier to maintain. Compared with symmetric cryptography, it makes life much easier for Bob to manage only the one private key as well as for Alice who is not particularly required to keep Bob's public key in secure place.

There are many applications for asymmetric cryptography, such as authorization and digital signatures, as well as its use in the distribution of keys for symmetric cryptography. For example, if Alice wants to send a symmetric cryptography key K to Bob, she first encrypts key K with Bob's public key K_e and sends it to Bob. Because the ciphered text containing the key K can only be decrypted by Bob with his private key, K_d, Alice can send the ciphered K through an open communication channel without worrying about leaking key information.

However, the security of both symmetric and asymmetric cryptography depends largely on the security of the secret key. In the former case, Alice and Bob have to make the key safe and, for the latter case, Bob has to keep his private key secure. Given the fact that modern cryptographic algorithms often use a large key (1024 bits for instance), it is almost impossible or impractical to memorize keys. Therefore, key management becomes an important issue in cryptography. Smartcards bring a solution because the secret keys can be stored and maintained on the same smartcards that perform the encryption algorithms. Currently, both symmetric and asymmetric algorithms can be embedded on smartcards.

2.2.3 Block cipher

If an encryption algorithm takes a fixed length of plain-text input and produces the same length of ciphered-text output, the algorithm is called a *block cipher* that is the most common and important method used in modern cryptography. The main actions in a block cipher are *permutation* and *substitution.* Permutation changes the position of each bit while substitution replaces one data pattern with another using a hardware or software mechanism (often known as an SBOX). Most cryptographic algorithms are block ciphers, such as DES, AES and other candidates for AES [30 - 35].

2.2.4 Stream cipher

Stream cipher is another kind of encryption algorithm which encrypts one bit (or byte or word) in each step. Usually, the central part of the stream cipher system is a random number generator. The security of a stream cipher depends largely on the random number generator. If the generator generates only a repeated number stream (for instance repeating every 16 bits), then this stream cipher is not going to be secure. Ideally, the random number generator should produce random numbers that never repeat. Practically, a generator should be able to produce a pseudo-random number stream which is long enough to be computably secure. In many stream cipher systems, the design of the pseudo-random number generator centres on linear feedback shift registers (LFSR) which will be looked at in more detail in chapter 5.

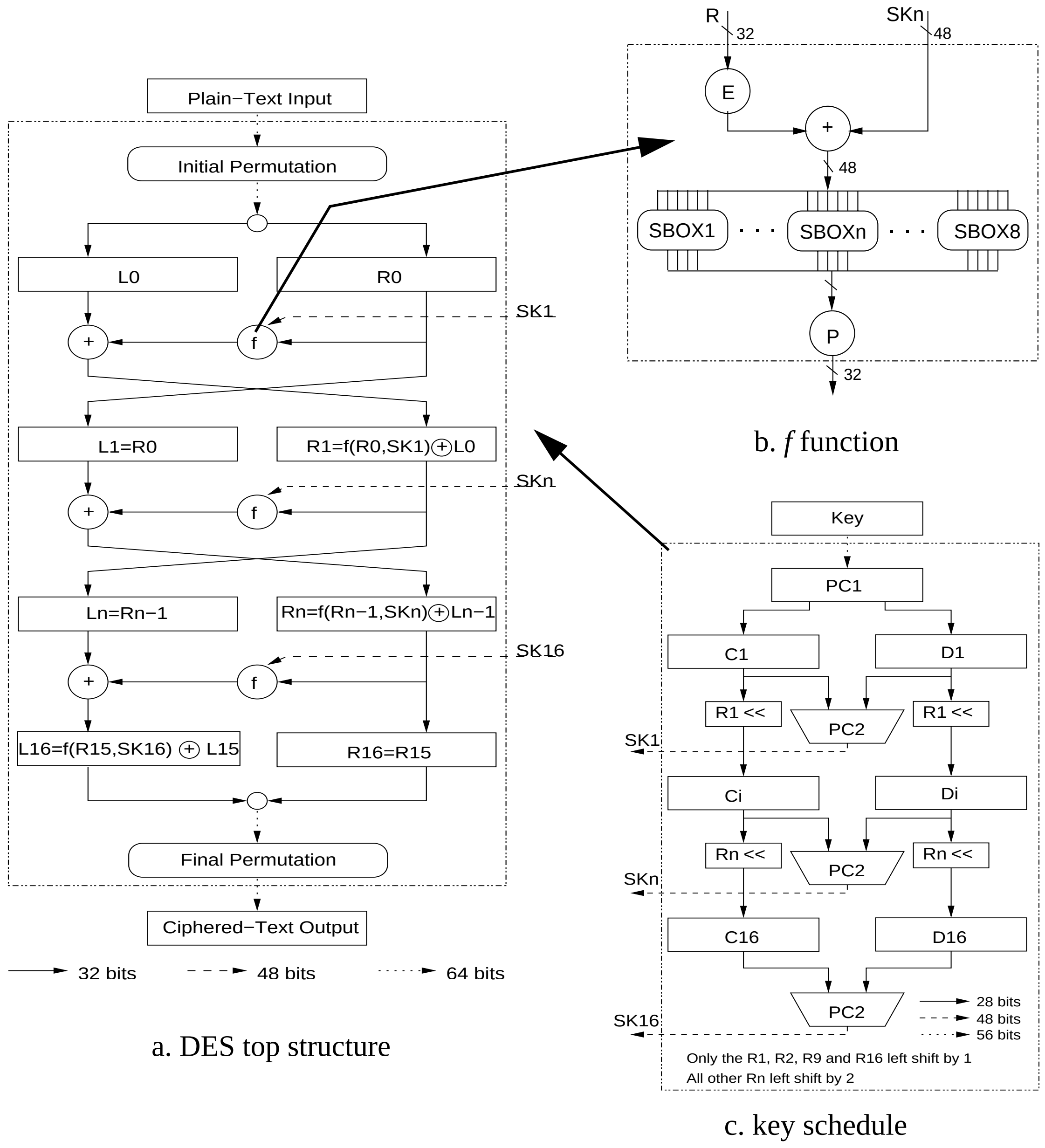
Compared to the block cipher, a stream cipher can be implemented with a smaller number of logic gates while still being able to offer good security. In the early VLSI era, the stream cipher was widely used in cryptosystems because of the prohibitive cost of the large VLSI circuits required for block ciphers.

2.2.5 Data Encryption Standard (DES)

The DES encryption algorithm [30] was originally developed by IBM in the 1970s and has been widely used in many secure applications. The following section describes the encryption process.

The DES algorithm, as shown in Figure 2.1, is a block cipher and was designed to encrypt and decrypt blocks of 64-bit data under the control of a 64-bit secure key. The algorithm contains three processes: an initial permutation (IP), 16 rounds of key-dependent computation, and a final permutation (FP) which is a reverse of IP. The two permutations are defined by their permutation tables, and the key-dependent 16-rounds of computation can be summarised as equation (1):

$$L_n = R_{n-1}, R_n = L_{n-1} \oplus f(R_{n-1}, SK_n) \qquad (1)$$



a. DES top structure

b. *f* function

c. key schedule

Figure 2.1 The DES encryption algorithm

where n is from 1 to 16, the function *f* is the main body of the DES algorithm as shown in Figure 2.1b, and the 48-bit subkey, *SKn*, is computed by the key schedule scheme shown in Figure 2.1c.

The key schedule function, shown in Figure 2.1c, is used to generate 48-bit subkeys for the 16 encryption rounds from the 56-bit key (64 bits minus 8 bits which are only used for parity check). The 56-bit key is first permuted according to the permutation choice #1, PC1, and the 56-bit output of PC1 is then divided into two sets for the 28-bit C and D shift registers respectively. The left most 28 bits are assigned to C register while the right most 28 bits are assigned to D register. These two shift registers shift their data values accordingly and the shifted value is reunited and then permuted according to the permutation choice #2, PC2, to generate one 48-bit subkey. This operation continues until all sixteen subkeys are generated. Table 2.1d. shows the PC1 and PC2 permutation tables.

The function *f*, shown in Figure 2.1b, contains eight SBOXes (Substitution BOXes). First, using an expansion table E shown in Table 2.1c., the 32-bit input from the register R is expanded to a 48-bit intermediate output which is XORed with the 48-bit subkey from the key schedule unit. The result of the XOR is divided into 8 groups of 6 bits to form the inputs to the 8 distinct SBOXes from which 8 groups of 4 bits of output are produced according to their respective substitution tables. Each substitution table has 4 rows and 16 columns. The eight 4-bit output of the SBOXes form a 32-bit output which is then permuted according to the P permutation table in Table 2.1c. to yield the 32-bit output of the *f* function.

To describe how the SBOX works, we take SBOX1 as an example. The first and the last bit of the 6-bit input (bit #1 and bit #6) represent the row number, denoted as i, in the range of 0 to 3, and the middle 4 bits (bit #2 to bit #5) represent the column number, denoted as j, in the range of 0 to 15. For each 6-bit input, a number in the range of 0 to 15, that is, a 4-bit binary data value can be found in the i^{th} row and the j^{th} column in the table. For instance, if the 6-bit input is 101100, then the row number is 10 (the 2nd row) and the column number is 0110 (the 6th column), therefore the output is 2, as highlighted in Table 2.1a, giving a 4-bit output of 0010. The other SBOXes have the same operation but with different tables.

The DES decryption process is the reverse of the encryption using the same key. The last round in encryption is the first round in decryption and the last round subkey in encryption is the first round subkey in decryption

S1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S5	2	12	4	1	7	10	11	6	8	4	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	4	11	2	14	15	0	8	13	1	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	12	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

a.

IP								FP							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	6	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	8	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	12	52	20	60	28
59	51	43	35	27	19	11	3	35	3	43	11	51	29	59	27
61	53	45	37	29	21	13	5	34	2	42	10	50	28	58	26
63	55	47	39	31	23	15	7	33	1	41	9	49	27	57	25

b.

E						P			
32	1	2	3	4	5	16	7	20	21
4	5	6	7	8	9	29	12	28	17
8	9	10	11	12	13	1	15	23	26
12	13	14	15	16	17	5	18	31	10
16	17	18	19	20	21	2	8	24	14
20	21	22	23	24	25	32	27	3	9
24	25	26	27	28	29	19	13	30	6
28	29	30	31	32	1	22	11	4	25

c.

PC1							PC2					
57	49	41	33	25	17	9	14	17	11	24	1	5
1	58	50	42	34	26	18	3	28	15	6	21	10
10	2	59	51	43	35	27	23	19	12	4	26	8
19	11	3	60	52	44	36	16	7	27	20	13	2
63	55	47	39	31	23	15	41	52	31	37	47	55
7	62	54	46	38	30	22	30	40	51	45	33	48
14	6	61	53	45	37	29	44	49	39	56	34	53
21	13	5	28	20	12	4	46	42	50	36	29	32

d.

Table 2.1: The DES expansion and permutation tables

2.3 Attacks on smartcards

The smartcard is believed to be an effective solution for many secure applications. However, many attacks have been successfully carried out on smartcards. Attacks on smartcards can be classified into invasive attacks and non-invasive attacks. In some hostile environments such as distributed and point-of-sale systems, attackers have access to the smartcard over an extended period of time. This can make smartcards very vulnerable, and those without physical access protection can be exposed to invasive attacks. Kommerling and Kuhn have discussed many invasive approaches to attack smartcard chips, such as de-packing smartcard chips, memory reverse engineering, memory modification and rewriting and microprobing [5]. Anderson and Kuhn also showed a number of invasive attacks that can be carried out by opponents with much shallower pockets [37]. A good cryptosystem is usually equipped with mechanisms, such as electromagnetic shielding, low-pass filters and autonomous clock signal generators, to protect it from those invasive attacks.

However, smartcards with physical access protection may still be vulnerable to non-invasive attacks. Manufacturers always overestimate the quality of their products' security features. Many cryptosystem algorithms can be broken by non-invasive attacks such as timing attacks, electromagnetic emission attacks, hardware fault attacks [28] and, more recently, power analysis attacks [8]. Among these attacks, power analysis attacks are probably the most powerful techniques for breaking into today's cryptographic implementations on smartcards. Many RSA and DES algorithm implementations on smartcards are vulnerable to power analysis attacks. Differential power consumption is the key source of information for attacks on RSA and DES as side-channel information may be leaked when RSA is executing modular multiplication or during the 16 rounds of encryption in DES. The attack measures variations in power consumption and uses techniques such as statistical analysis to amplify the difference and obtain the secret keys from the cryptosystem. This approach has been very successful and has received widespread attention.

These non-invasive attacks exploit information leaked from the microchip, which succeeds because the designer of a cryptosystem usually focuses on the encryption algorithm but pays less attention to the possibility of side-channel information leakage. In fact, this side-channel information leakage is difficult to avoid in practice even when the designer does consider the problem carefully. A digital system produces many analog side-effects which can be used by non-invasive attacks.

Obviously invasive and non-invasive attacks are likely going to be used together to attack a security device. This gives attackers much more power to break into the security device. For example, an attacker can learn the architecture of a device by using invasive methods and carry out attacking the devices using non-invasive methods.

2.3.1 Non-invasive attacks

An invasive attack accesses the physical circuits and the location of memory patterns on the chip. Non-invasive attack is where the attacker does not necessarily know the details of the inside of the security implementations and makes no attempt to access the devices. Non-invasive attacks include power consumption analysis, timing attack, clock and power glitching, fault analysis attack, etc. Non-invasive attacks are the main focus of this thesis.

Timing attack

A timing analysis attack [6] has a similar basis to power analysis attacks. Instead of using differential power information, timing analysis uses the leakage of information through data- and instruction-dependent timing variations. If a secure implementation takes a different amount of time to execute different instructions or to process different data, then the secure information can be extracted by analysing the timing differences. For instance, many multiply architectures employ clever algorithms, such as Booth's algorithm [75], to speed up their operation; however the timing differences may leak information that can be used to determine the Hamming weight of the multiplier's operands.

Clock and power glitch attacks

The idea behind clock and power glitch attacks [5, 14] is that increasing the clock frequency or decreasing the power supply for a very short time induces a malfunction on the circuit, so the attack can redirect the execution of the algorithm. In general, smartcards execute a routine that detects unauthorized access attempts through erroneous input. However, an attacker can create a malfunction that manages to bypass the sophisticated cryptographic barrier by preventing the execution of this code.

Clock glitches increase the clock frequency for a time so short that it can not be detected even by the high-frequency detector, which is used to determine high frequency signals, on the smartcard. Such temporary increase in the clock frequency causes some instructions, for example conditional jumps, to have insufficient time to execute correctly: the branch condition does not have enough time to evaluate, as the current cycle is artificially shortened, and instead of taking the branch, the execution falls through to the next instruction.

Such a trick can be used to override a system’s response to an attack. Power supply fluctuations can affect the charge and the discharge time of transistors. Decreasing the power supply voltage can decrease the speed of the circuit, and therefore some instructions may not have enough time to complete, leading to the same effect as a clock glitch.

2.4 Power consumption and attacks

Technological constraints in CMOS VLSI circuits result in a different power consumption when manipulating a logic “one” compared to manipulating a logic “zero”, see below in Section 2.4.1. Consequently, cryptographic algorithms often consume a slightly different amount of power when performing different encryption operations. The power analysis attack is based on observations of the power or current consumed by the microcontroller during the execution of decryption or encryption on a cryptosystem. The basic philosophy behind this observation is that there is a correlation between the power consumption and the data being processed when certain operations are being executed. If the variations in the power consumption can be recorded somehow by precisely monitoring the power current, those operations and data can possibly be identified.

Power analysis attacks have put smartcards at high risk because of the leakage of this side channel power consumption information. Techniques developed by Kocher [7] show how the attack exploits the small amount of side-channel information to break widely-used cryptographic algorithms. The following section starts with looking at power consumption in CMOS circuits, then power analysis attack techniques are discussed.

2.4.1 Power consumption in CMOS circuits

A CMOS VLSI circuit is built out of individual transistors which act as a switches. The motion of electrical charge consumes power and yields electromagnetic radiation, both of which are externally observable. If individual transistors produce externally readable electrical information, it is possible to identify aspects of the whole circuit behaviour by measuring the power consumption.

The power consumption of a CMOS VLSI circuit consists of three parts: static consumption, dynamic consumption and short-circuit consumption [40, 41]. The dynamic consumption, P_d, is normally the dominant component and is primarily responsible for information leakage. It is the result of the charging and discharging of the load capacitance, and is given by:

$$P_d = \frac{1}{2} \cdot f \cdot C_l \cdot V_{dd}^2 \cdot A_c \qquad (2)$$

where A_c is the circuit activity (the proportion of the total capacitance switching in each clock cycle on average), f is the frequency of switching, C_l is the total circuit capacitance and V_{dd} is the power supply voltage.

From this equation it can be seen that, for a given circuit where the load capacitance and supply voltage are constant, the dynamic power consumption depends on the circuit activity, A_c. In other words, the more capacitance that is switched, the more power is dissipated. Figure 2.2 shows how the power consumption varies with the number switching transistors. It also illustrates that switching transistors from “low” to “high” (the left side) consumes more power than switching from “high” to “low” (the right side).

For the attacker of a secure system, this is the characteristic of CMOS VLSI circuits that inspires power analysis attacks. For example, a multiplier adds an appropriately shifted multiplicand to a running sum when the multiplier operand bit is one whilst it does nothing when zero. Hence the multiplier dissipates more power processing a multiplier bit which is a one than when it is a zero. If the multiplier operand is a secret key, then power analysis on the multiplication operation yields vital information about the key.

2.4.2 Simple power analysis attack

A simple power analysis attack observes the power consumption waveform directly and measures the obvious peaks which are known (through prior investigation) to relate to security information. If a smart attacker has a detailed knowledge of the insides of a smartcard and knows where the security information is, then the attacker will use power information efficiently to extract that security information. Individual operations can be identified if power consumption varies significantly when performing different operations. For example, conditional branches typically need to test the value of the conditional bit. The power consumption and the execution time may differ significantly depending on the value, “one” or “zero”, of this bit. Therefore, conditional branches are vulnerable since the value of this bit can be determined by observing the power consumption at the clock cycle of the instruction that computes the bit. Kocher claimed that 16 rounds and the final permutation of DES can be determined by simple power analysis attacks [7].

By measuring power information attackers may be able to identify the instructions being executed. Figure 2.3 shows the power information revealed when executing XOR and

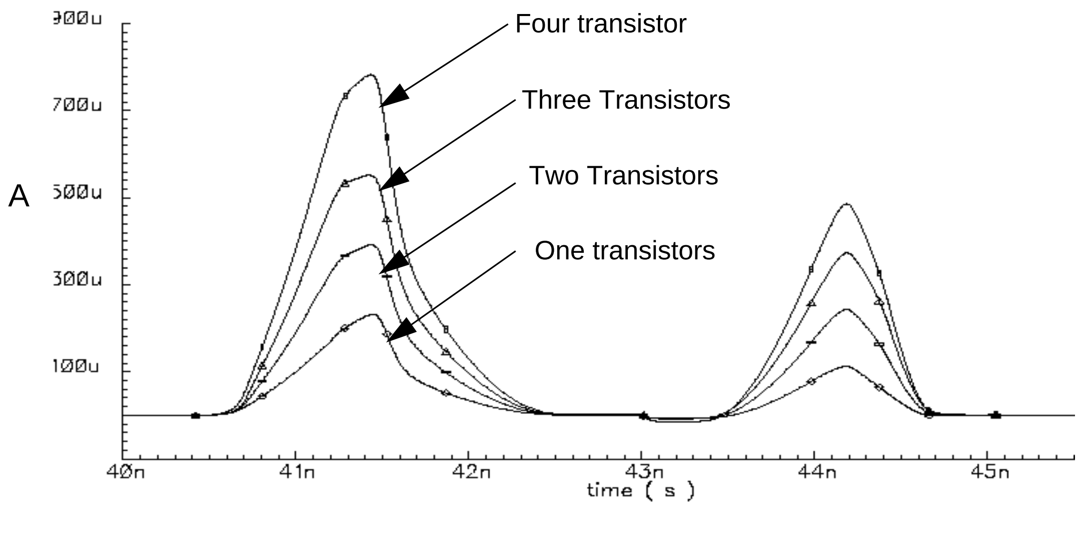


Figure 2.2 Current variation with the number of switching transistors

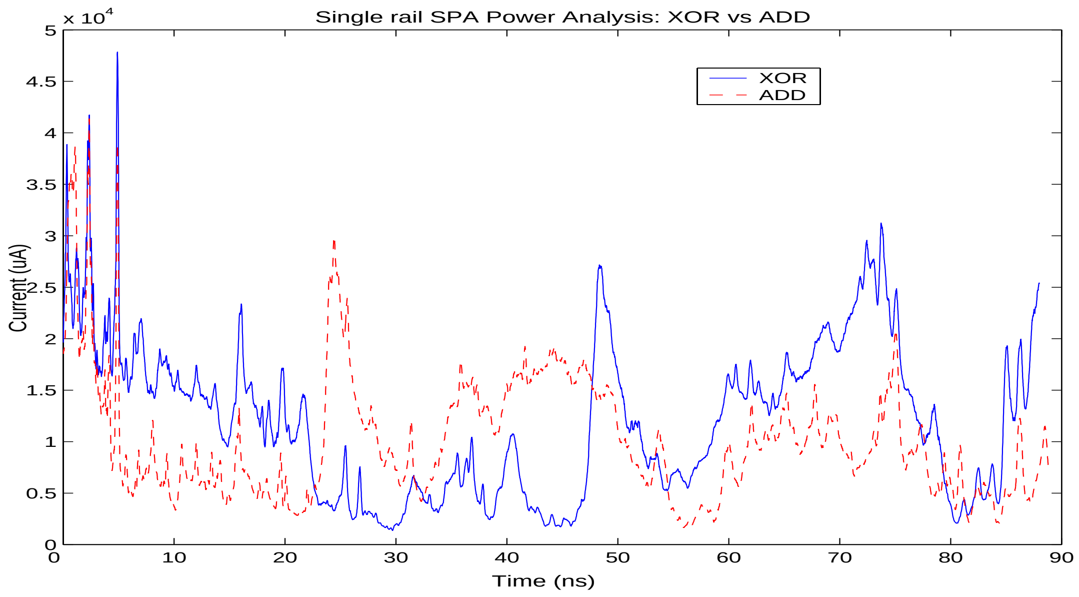


Figure 2.3 Power consumption of XOR and ADD instructions

ADD instructions on the single-rail SPA processor [15]. The solid curve is the power waveform for the XOR and the dashed curve the waveform for the ADD instruction. It is clear that the two waveforms are readily distinguishable. Such difference is likely due to processing different instructions rather than different data, which normally results in much smaller difference. This simple observation indicates the power consumption of single-rail SPA is instruction-dependent. Similarly, attackers can extract information on the data being processed if there is data-dependent side-channel information leakage.

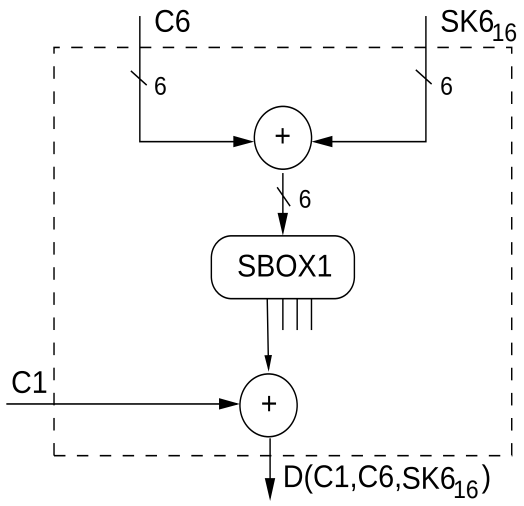
2.4.3 Differential power analysis attack (DPA)

DPA is much more difficult to defend against than simple power analysis [8, 13]. Attackers do not necessarily need to know the implementation details of the cryptosystem on which the DPA attack is mounted. DPA uses statistical analysis techniques to maximize the information extracted from power variations by subtracting the average power (which is not correlated to the data) and other extraneous signals from the waveforms. DPA aims to eliminate the power consumption and other measurement noise that is not relevant to security information, and to amplify the power consumption that is correlated with the secure information.

A DPA attack involves two steps: data collection and data analysis. Similar to the simple power analysis attack, data is collected by monitoring the power consumption of the device when running algorithms. Using statistical data analysis helps to maximize leaked side-channel information and to identify a single trace from the observations. An attack on DES as an example of a DPA process is briefly reviewed below.

Firstly, the DES algorithm is executed for N iterations and one power trace is obtained for each iteration with M sampling points for each trace, denoted as *P(i, j)* where i=1...N, indicates the number of the iteration and j=1...M is the sampling point of each iteration. Meanwhile, the ciphered-text output and the plain-text input for each operation is assumed to be available to the attacker, denoted as CTO_i and PTI_i respectively. Then a partitioning function is chosen to split the extracted waveforms into two sets in order to see if there is a difference in power consumption when different data (plain-text input) is being processed. Because, at some point during the DES operation, the value of the bit obtained from equation (3) needs to be computed, this equation is chosen as a partitioning function for the differential power analysis attack on DES encryption.

$$D(C_1, C_6, SK6_{16}) = C_1 \oplus SBOX1(C_6 \oplus SK6_{16}) \qquad (3)$$



$SK6_{16}$: the first six bits from the 16th sub−key, SK_{16}

C6: six bits from ciphered text output, CTO, that XOR with $SK6_{16}$

C1: one bit from CTO that XORs with the first bit of 4−bit SBOX1 output

Figure 2.4 The diagram of the partitioning function for differential power analysis on DES

This partitioning function is also illustrated in Figure 2.4, where C_6 is six corresponding bits of the ciphered output, $CTO_{i,}$ that XORs with $SK6_{16,}$ the first six bits of a guessed 16th round subkey, SK_{16}, to form a six-bit input to SBOX1, the first of the eight SBOXes in DES. The one bit output of the D function results from XORing C_1, which is one corresponding bit of the ciphered output, with the first bit of the 4-bit SBOX1 output.

The power waveforms, *P(i,j)*, are partitioned into two groups, group zero, *P0*, and group one, *P1*. according to the partitioning function as follows:

$$P(i, j) \in P_0 \big| D(C_1 \cdot C_6 \cdot SK6_{16}) = 0 \qquad (4)$$

$$P(i, j) \in P_1 \big| D(C_1 \cdot C_6 \cdot SK6_{16}) = 1 \qquad (5)$$

The partitioned two groups of waveforms are then averaged separately to obtain their typical characteristic waveforms, denoted by $P_0[j]$ and $P_1[j]$ respectively, following equations (4) and (5). Then, the difference between these two characteristic waveforms, denoted as dP[j], can be computed as follows:

$$dP[j] = P_1[j] - P_0[j] \qquad (6)$$

If information leakage exists on the target device through the difference in power consumption, then whenever the partitioning bit is manipulated, the result of equation (6) will display a small difference, denoted as *e*, in the power consumption corresponding to the value of this bit, ‘one’ or ‘zero’. At any other time, because the operation is not correlated to the D function, it will theoretically be equal to zero. Note that, in reality, it may not be zero because of measurement noise.

Since the ciphered-text output, CTO_i, is known, the only thing the attacker does not know is the six bit portion of subkey, $SK6_{16}$. The attacker could guess all of the 2^6 possibilities using a brute force search. For each guess, a new partition is computed and a new difference trace, *dP[j]*, is formed. One of the 2^6 difference traces will show bias whenever the partitioning bit is computed. All other difference traces will not show any significant bias. The guessed portion of the subkey for the partitioning which generates bias would likely be the correct guess. By applying this approach to the other seven SBOXes, the remaining portion of the subkey could be discovered [13].

2.5 Summary of chapter

This chapter has introduced some basic background material on both smartcards and cryptography, including describing the Data Encryption Standard (DES) algorithm which will be used in chapter 8. Smartcard security and some common attacks on smartcards, both invasive and non-invasive attacks, were discussed with a focus on power analysis attacks. This chapter also reviewed how CMOS VLSI consumes power in addition to introducing the Differential Power Analysis (DPA) attack in detail based on the DES algorithm. The following chapter introduces asynchronous design technologies and hardware security.

Chapter 3: Asynchronous Design and Hardware Security

This chapter presents an introduction to asynchronous design technology, including delay models, handshake signalling protocols, data encoding methods as well as synthesis tools for asynchronous design, and the advantages and disadvantages of asynchronous design. This chapter also discusses issues relating to hardware design for security.

3.1 Introduction

The design of digital systems can be classified into two methods, synchronous (clocked) and asynchronous (also widely known as self-timed or clockless) design. Although the two design methodologies were both developed in the early days of digital design before the introduction of VLSI (Very Large Scale Integration) circuit technology, synchronous design has dominated digital circuit design for several decades because it offers a simple design style by employing one or more globally distributed sources of timing information, the clock(s), to sequence systems. The clock signal controls the change of state within the system and all activities regulated by the clock in the system have to be completed within the clock cycle time in order to ensure that correct data is latched into storage. Figure 3.1a shows a typical synchronous communication system where both the sender and the receiver are regulated by the clock signal, clk. At the active clock edge (rising in this case), data is sent from the sender to the receiver. Data has to be ready at the sender side before that edge and must be held until received by the receiver as in Figure 3.1b.

In contrast, asynchronous circuit design does not use a global clock but employs a handshaking protocol to control communication between independent units in the system via *channels* which usually are unidirectional. Local handshake signals are used to indicate the availability of data at the sender and the acceptance of data at the receiver.

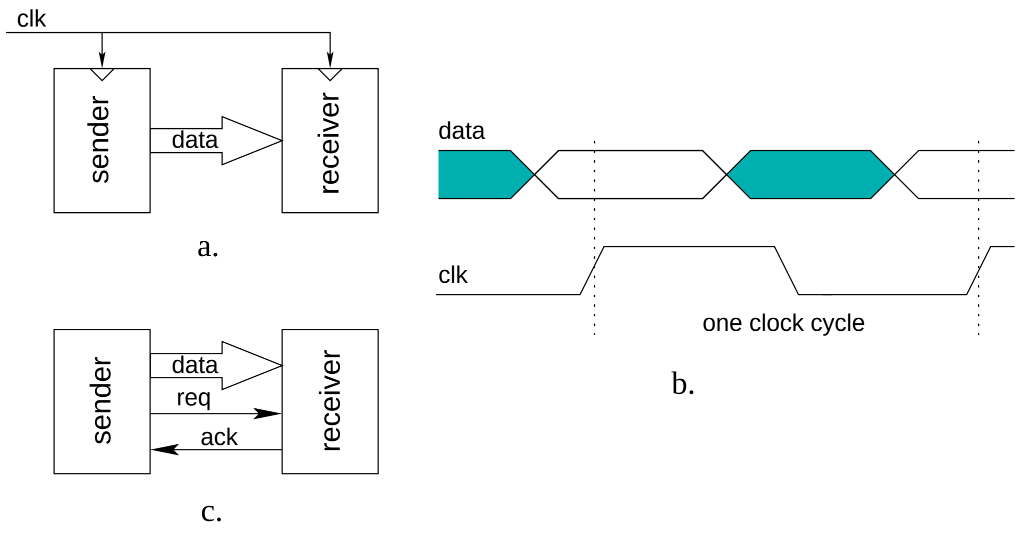


Figure 3.1 Synchronous design vs. asynchronous design

Figure 3.1c shows a self-timed communication system where data is transmitted through *a push channel* where data and the request signal travel in the same direction. (The alternative, a *pull channel,* is where the request signal travels in the reverse direction to the data.) A *request* signal travels on the req wire to indicate when the data is ready for transfer, and the *acknowledge* signal travels back to the sender on the ack wire to indicate the acceptance of data and the readiness for further data. Regulated by these two handshake control signals, data is safely transmitted through the channel. The sequences for these signals using *2-phase* and *4-phase* handshake protocols (described in the following section) will be shown in Figure 3.2.

Given such examples for comparison, the fundamental differences between synchronous and asynchronous VLSI design methodologies are:

- Activities on a synchronous circuit are dominated by the globally distributed clock signal and triggered simultaneously and completed within the clock cycle.

- Events on self-timed circuits, on the other hand, are managed by locally distributed handshake control signals, and activated at their own pace and completed accordingly.

3.2 Asynchronous design

The following sections describe some fundamental aspects of asynchronous logic design technology; many others can be found in the literature [42, 43].

3.2.1 Asynchronous circuit models

An asynchronous circuit can be characterised by the delay model as follows:

- *Bundled-data* circuits assume a pre-determined timing relationship between data and control signals, which must take account of gate and wire delays. The data must be valid before the request signal. That is, the delay in the control signal must be no less than the delay in the data path. For example, the data value in Figure 3.1c must be valid at the receiver end before the arrival of the request signal, req, otherwise incorrect data may be latched.

- *Speed-independent* (SI) circuits operate correctly regardless of gate delays, but the wire delays are assumed to be zero or so small that they can be neglected. This assumption normally holds only in small circuits since, in a large circuit where the interconnect inevitably causes long delays, the wire delays cannot be assumed to be zero.

- *Delay-insensitive (DI)* circuits are those which do not depend on gate delays or wire delays, which means that the circuit works properly despite delays on wires and gates. For a large and complex asynchronous circuit, this model is usually applied to the higher levels of the design.

- *Quasi-delay-insensitive* (QDI) circuits are similar to delay-insensitive circuits but they assume that branches in a forked wire have the same delay (the isochronic fork assumption), or at least the differences between delays in forks are negligible compared with the delays in circuit elements.

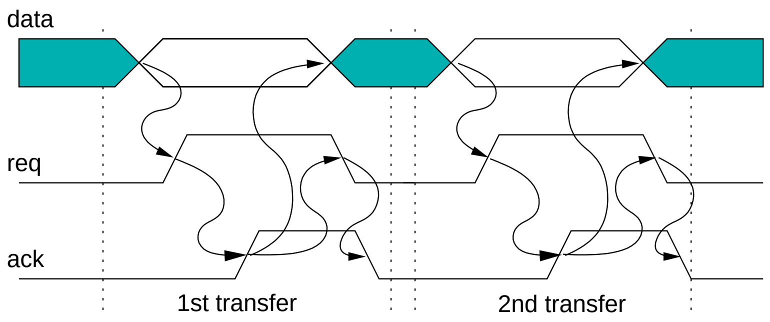
3.2.2 Handshake signalling protocol

The transfer of data between sender and receiver in asynchronous circuits is handled by the handshake signals: a request signal, req, which the sender uses to start the transfer, and an acknowledge signal, ack, by which the receiver responds to the transfer. A req indicates data validity and readiness for transmission while the acknowledge from the receiver indicates the acceptance of the transmitted data and the receiver's readiness to accept further data.

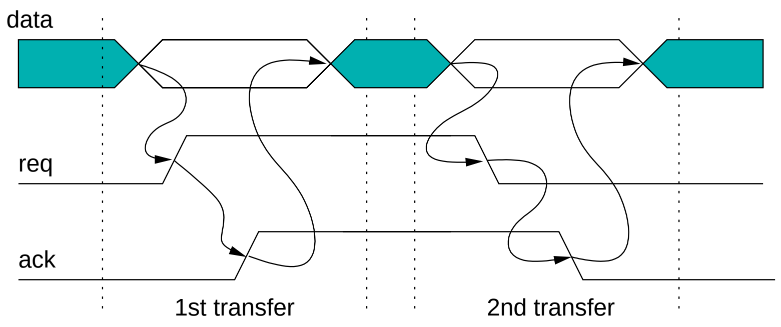
A handshake signal can be sent either on a separate wire or encoded into the data. Two different signalling protocols are commonly used to pass these signals: a 4-phase (level signalling) protocol or a 2-phase (transition signalling) protocol, as shown in Figure 3.2.

4-phase level signalling protocol

4-phase level signalling is a return-to-zero signalling scheme, shown in Figure 3.2(a). It uses the level of a signal to indicate handshake information. The request and acknowledge



(a) 4-phase level signalling



(b) 2-phase transition signalling

Figure 3.2 Handshake protocols (push channel)

events have to be passed in sequence. That is, a second request must not occur until after an acknowledgement has been received. A return-to-zero phase is needed to reset the state before the next data transfer.

2-phase transition signalling protocol

In the 2-phase protocol scheme shown in Figure 3.2(b), the logic level of the signals is no longer important. Instead, events are signalled by the transitions (both rising and falling) of the signals. Similar to the 4-phase protocol, the request and the acknowledge signals have to be in sequence. However, the 2-phase protocol does not need a return-to-zero phase. In fact, because of the lack of a return-to-zero phase, the 2-phase protocol has the potential advantages of high performance and lower power as this scheme uses only half as many transitions per transfer as the 4-phase protocol.

It is obvious that the 2-phase protocol has some advantages if it is used in communication systems (rather than computation) where the power dissipation is determined by the number of transitions required to send an amount of data. Because fewer transitions are need to transmit the same amount of data, 2-phase signalling will consume less energy. However, 4-phase control circuits are often simpler than 2-phase systems because the level sensitive signals in 4-phase circuits can be easily used to control latches.

3.2.3 Data encoding

This section describes the general categories of data encodings commonly used in asynchronous circuits, including single-rail, dual-rail, N-of-M and one-hot encodings. Among them, dual-rail encoding will be extensively used in this research and its ability to balance power consumption will be discussed throughout the thesis.

Single-rail encoding

Similar to conventional synchronous circuits, single-rail encoding uses only one wire to represent two logic values, “one” and “zero”. In asynchronous circuits, the encoding scheme will require additional wire(s) to indicate the request and acknowledge signals.

This encoding is often referred to as “bundled-data” because the data is “bundled” with the request signal.

Dual-rail encoding

Dual-rail encoding, as shown in Figure 3.3, requires two wires to express one bit of data, each line representing one of the two binary values. To transfer n-bit information in parallel needs 2n wires. An additional wire to transmit the acknowledge signal to complete the handshake protocol may also be required to build dual-rail channels. Transitions on one wire indicate a logic “one” and similarly transitions on the other mean a logic “zero”. Thus, no matter what data value is transmitted, the same number of transitions is needed for each transfer. Therefore, the same amount of power is dissipated. Differences in power consumption when processing different data could be reduced significantly. This is the motivation for using dual-rail encoding to implement a secure device.

A dual-rail circuit does not necessarily require a separate wire for the request signal because it is possible to determine the validity of the data by detecting valid data on one of the two wires for each bit of the data. If an active level (in 4-phase signalling) or an event (in 2-phase signalling) is detected new data is ready to be transferred. In other words, a completion detection tree, a mechanism to determine the validity of the data, is required.

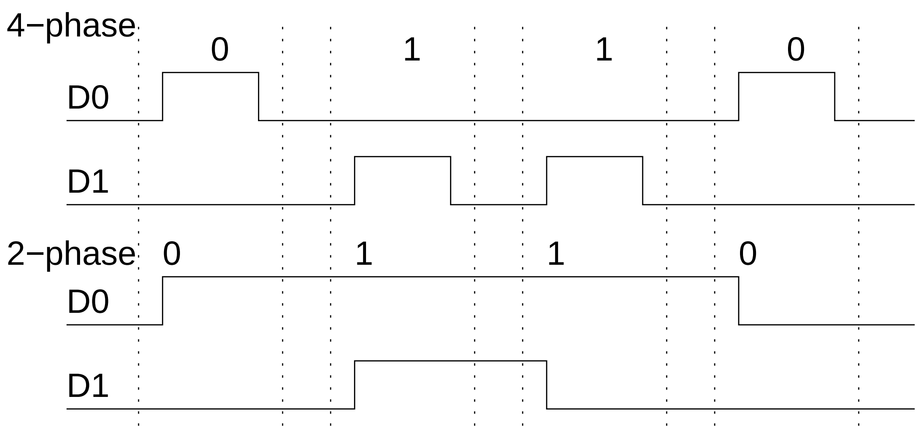


Figure 3.3 Dual-rail encoding

N-of-M encoding

An N-of-M code communicates by signalling an event on exactly N of the M wires of a code group. Data encoded using N-of-M encoding will be transmitted correctly regardless of the delay in the wires and the gates, and thus this scheme is delay-insensitive. It requires a completion detection tree to determine the validity of the data [44]. An N-of-M code can encode $\frac{M!}{N! \cdot (M-N)!}$ possible symbols representing data values. For example, a 3-of-6 encoding has twenty possible symbols which could be used to indicate a 4-bit binary value, and also End, Start and Ack events. Similarly, a 2-of-7 code can produce 21 possible symbols. Comparing these two N-of-M codes the former uses only 6 wires instead of the 7 wires used by latter whilst the 3-of-6 code uses 3 transitions instead of the 2 transitions of the 2-of-7 code to send a symbol (or 6 and 4 transitions in the level signalling protocol respectively). Dual-rail encoding can be defined as a special case of N-of-M encoding, 1-of-2 encoding. As with dual-rail encoding, N-of-M encoding consumes the same number of transitions regardless of what the data value is. It therefore has the same potential as dual-rail encoding to balance the power consumption used when processing different data values.

One-hot encoding

One-hot encoding can be recognized as a special N-of-M encoding where N is equal to one, each of the M wires representing one symbol which represents one data value. Thus it requires 2^m wires to encode an m-bit binary data value. Similarly, ten wires can directly express decimal values. Like N-of-M encoding, a completeness detection tree is required to determine the validity of the data presented.

Null Conventional Logic or NCL technology developed at Theseus logic Inc. [45] combines one-hot delay-insensitive asynchronous logic with a return-to-zero protocol. NCL defines digital circuits having three logic values, “one”, “zero” and “NULL”. Since a single wire is limited only to two values and one of them must be assigned to NULL, there is only one value left to express data, so at least two wires are required to address a one-bit binary value: one wire represents “one” and the other “zero”, while both wires return to zero to assert NULL.

In NCL, threshold gates [46] are used to implement circuits. A threshold gate yields a high output if a specified number of inputs become high and the output will remain high until all inputs fall to zero. For instance, a 2/3 threshold gate requires two of the three inputs be high to yield a high on the output, and the output remains high until these two inputs return to zero.

3.2.4 Muller C-elements

The Muller C-element is the most common gate encountered in asynchronous circuits. The basic functions of a C-element are event synchronization and state holding. Figure 3.4 shows three different C-elements and their state graphs, which describes all possible states combinations of output ports and input ports by changes in inputs and outputs: one symmetric 2-input C-element (a.) and two asymmetric 2-input C-elements (b. and c.). An input controls both edges when it connects to main body of the gate. It controls only the rising edge when connected to the '+' extension and controls only the falling edge if it is connected to the '-' extension.

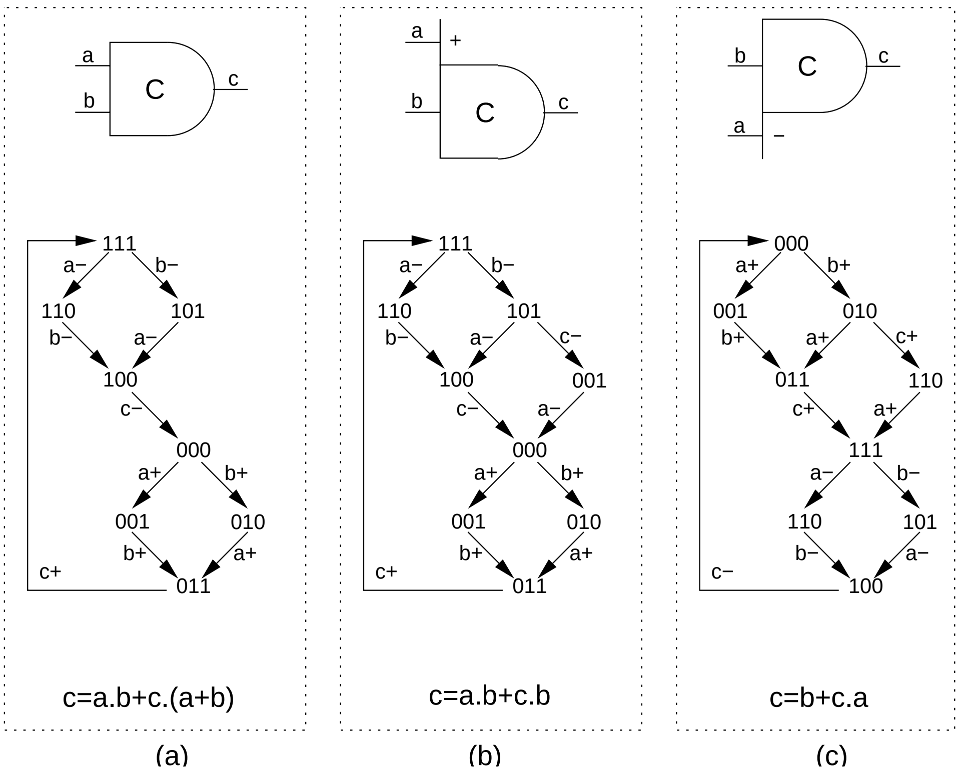


Figure 3.4 C-elements and their state graphs

Figure 3.4(a) shows a symmetric C-element with two inputs connected to the main body of the gate. Both inputs control both rising and falling edges, that is, the change of output c is dependent on both inputs, a and b. For example, the state graph below the symbol shows the c+ can only happens when both a+ and b+ have happened allowing states changes from 000 to 011 and finally to 111.

Figure 3.4(b) is an asymmetric C-element that has one input connected to the '+' extension, which means that this input controls only the rising edge of the output. So setting output c to high needs both inputs a and b to be set high. However, the return to low of input b will lower the output regardless of input a. The state graph below the symbols shows the full behaviour of the asymmetric C-element.

Figure 3.4(c) is another 2-input asymmetric C-element that has one input connected to the '-' extension. The output c is raised if an event arrives on input b without waiting for an event on input a. The output c, however, becomes low only when both inputs have become low. The behaviour of this 2-input asymmetric C-element is illustrated in the state graph shown below its symbol.

3.3 Asynchronous design for hardware security

Increasing interest in asynchronous design is mainly driven by its advantages compared to synchronous design. The claimed benefits include power efficiency, clock skew avoidance, modularity and electromagnetic compatibility. The Advanced Processor Technologies Group or APT Group (formerly known as the Amulet Group) in Manchester, U.K., has developed a series of Amulet processors [47-50], compatible with the ARM processor, in order to demonstrate these benefits. More recently, additional advantages has been proposed suggesting that asynchronous design can improve security against non-invasive attacks [14]. The research presented in this thesis investigates such potential benefits for the use of asynchronous design in security applications, in particular, there is interest in resistance to differential power analysis attacks.

The security of cryptosystems depends on both software and hardware. Whilst a rigorous encryption algorithm provides the fundamental security against cryptanalysis attacks on the algorithm's mathematical basis, the implementation of an encryption algorithm also has a great impact on the security of a cryptosystems. Software security involves techniques such as dummy code insertion, code rearrangement, data masking etc. [51-53]. On the other hand, hardware security uses hardware design technologies, such as random noise insertion, to minimize or mask information leakage from a hardware implementation to protect the implementation from both invasive and non-invasive attacks [54]. Hardware security is emphasized in this thesis.

The threats posed by the differential power analysis attack are mainly based on the fact that a physical circuit often leaks side-channel information, for instance, in power consumption. Dynamic power consumption is proportional to the number of switched transistors in CMOS VLSI circuits (as discussed in Section 2.4.1) so the data processed in the circuit and operations on that data can be determined by examining the power consumed. Implementing encryption algorithms in hardware is expensive and much more complex and time consuming than in software. However, most software encryption is implemented on general-purpose processors that are likely to leak information and are vulnerable to attack.

3.3.1 Power consumption randomization

As differential power analysis is based on the difference in power dissipation extracted from the implementation, security can be improved if there is a way that the extracted difference can be randomized and minimized. This can be achieved by adding *random noise.* Injecting random power noise could decrease the signal-to-noise ratio, where the leaked information is the signal, and hide real information in the noise, thereby protecting the device from attack [52]. However, using technologies such as Focused Iron Beam, a random noise generator may be disabled when an invasive attack is undertaken.

Self-timed design may be another approach to randomize power consumption because it produces random noise which is hard to filter out. The absence of a globally distributed clock signal makes power no longer to be dissipated synchronously among different units in the circuits. In fact, one individual sub-circuit consumes power independently from the others at its own pace, according to its handshake control signals. Despite that, the cyclic nature of these control signals still results in a periodic power consumption. However, the asynchronous behaviour smooths the power consumption and reduces the power consumption peaks. In addition, because the arrival of the handshake control signals may vary from cycle to cycle, it results in power consumption being shifted from one cycle to another causing random analysis noise during the power waveform averaging phase.

3.3.2 Balancing power consumption

Whilst power consumption randomization hides power information by inserting random power noise, balancing power consumption aims to reduce the difference in power consumption for different data and operations. Data balancing or masking is used in software security to hide sensitive information [54]. For instance, one byte of unbalanced data can be split into two bytes of balanced data by combining it with its complemented binary code, and thus the Hamming weight of the data can be balanced.

This scheme of data masking is similar to the *dual-rail encoding* which is commonly used in delay-insensitive or speed-insensitive self-timed circuits. As mentioned previously, dual-rail encoding uses two wires to represent one bit of binary data. The same number of transitions occurs to transmit data regardless of its value. Therefore, differences in power consumption between processing different data will be significantly reduced. Consequently, dual-rail encoding will improve the resistance to differential power analysis as the attacker would not learn anything useful by watching the power consumption.

3.3.3 Defeating clock and power supply glitches

In synchronous circuits, all the activities must be completed within each clock cycle for correct operation. Thus, by increasing the clock frequency or decreasing the power supply voltage (the lower the power supply voltage the slower the circuit works) malfunctions may be induced deliberately, resulting in some registers adopting an incorrect value or directing the code execution to the wrong part of the program, for instance replacing a conditional jump to bypass authorization check. Such glitch attacks are currently the simplest and most practical attacks [5].

However, self-timed circuits prevent these glitch attacks because of the absence of the global clock signal. Individual sub-circuits are allowed to communicate with each other at their own pace and operate correctly whatever the circuit speed. Reducing the power supply voltage results in a slowing down of the pace of activities in the circuits, but they can still operate without problems. This feature makes self-timed circuits inherently resistant to clock and power supply glitch attacks.

3.4 Summary of chapter

This chapter has briefly reviewed self-timed design technologies. They include three basic models of self-timed circuits, basic handshake signalling protocols for self-timed circuits and data encoding techniques which will be heavily used in the work presented in this thesis. In addition, the Muller C-elements is also discussed in particular as they are important in any self-timed circuit design.

The chapter further discussed some self-timed techniques and proposed advantages for hardware security such as power consumption randomization, balancing power consumption and defeating clock and power supply glitches.

The following chapter is going to investigate what are the fundamental differences of power consumption characteristic between self-timed and synchronous circuits; and to identify what are the difficulties for self-timed power analysis compared to conventional synchronous power analysis; and to implement a suitable methodology for self-timed power analysis.

Chapter 4: Asynchronous Power Analysis Methodology

Chapter 2 introduced the security of smartcards and their vulnerability to non-invasive attacks. In particular, differential power analysis techniques on clocked power waveforms were reviewed. However, although these techniques have been reported to be very successful in attacking secure devices such as smartcards, where a global clock signal is available to attackers as a timing reference, they are not readily applicable to asynchronous circuits, that were discussed in chapter 3 as potential countermeasures in particular to differential power analysis. The removal of any timing reference and the randomising effects of the handshake protocol on the asynchronous power consumption waveforms make differential power analysis more difficult. This chapter discusses these difficulties and how they may be overcome. This chapter details an extension to differential power analysis for a synchronous power waveforms.

4.1 Challenges for asynchronous power analysis

Differential power analysis, reviewed in chapter 2, has been well-studied and successfully applied to attack security implementations. All the analyses are based upon conventional synchronous circuits where a global clock signal is available to attackers as a time reference, enabling sophisticated statistical analysis to be carried out. However, the global clock signal is absent in self-timed design. This absence introduces new problems for differential power analysis, including the lack of a timing reference on which to base the differential power analysis, non-periodicity, fade-out of power consumption peaks and power consumption overlap. These problems make power analysis of asynchronous circuits much more difficult and require adjustment and extension to the differential power analysis for conventional synchronous power waveforms.

4.1.1 No timing reference

In synchronous circuits the global clock signal synchronizes all the activities on the circuit. Most electronic events start from the clock trigger edge and complete before the next clock trigger edge. Processing data or executing instructions on a synchronous circuit is dominated by the global clock, and event-driven power consumption follows the clock edges. Not only does this help identify events from power consumption waveforms, but also the provision of such a clock signal creates a time domain synchronized power consumption. Using the clock as a time reference, the attack is greatly simplified as it is easy to line up waveforms and apply statistical analysis techniques to the power waveforms.

Self-timed circuits, on the other hand, have many local handshake signals instead of one global signal. Sub-circuits are only activated as a result of local handshake signals, and sub-circuits work independently from each other. Thus, the power consumption waveform does not have a global time reference. The absence of a timing reference increases the difficulty of power analysis because it is more difficult to partition and align the power waveform.

Figure 4.1 illustrates two power consumption waveforms extracted from two LFSRs performing the same operation, one synchronous and one asynchronous. The upper power trace, A:i(vdd!), is the clocked waveform. Clearly, this trace can easily be divided cycle by cycle according to the clock signal on the top. In contrast, such a clock signal does not exist on the asynchronous trace, B:i(vdd!), which makes it difficult to align different cycles.

4.1.2 No periodicity

Because the power consumption in synchronous circuits follows the global clock signal, the power consumption has a characteristic periodicity at the same frequency as the clock signal.

In an asynchronous circuit, on the other hand, although the power consumption does display some characteristic of periodicity due to the cyclic nature of the handshake controls, the power consumption does not have exactly the same interval. The handshake behaviour may be affected by the preceding stage and possibly many other stages. The interval of handshake signals may vary in their timing, and it is not necessary for the handshake controls to be of constant periodicity. As a result, a power waveform partitioned according to handshake signals must be stretched or compressed to make it periodic. In addition, any such attempt to stretch a waveform may result in information loss due to the resampling of the power waveform data that must be carried out.

4.1.3 Fadeout of obvious power consumption peaks

In synchronous circuits, all activity is triggered by the clock, consuming a large mount of power. Thus, there is always a large power consumption peak at a certain time after each clock trigger edge because registers load new data values and combinational circuits act on them. These power consumption peaks in synchronous circuits usually contain information that reveals what operation is being processed in each clock cycle. If a security algorithm is being executed, then the leaked power consumption peaks contain information about the secure operations. Moreover, such obvious peaks help attackers

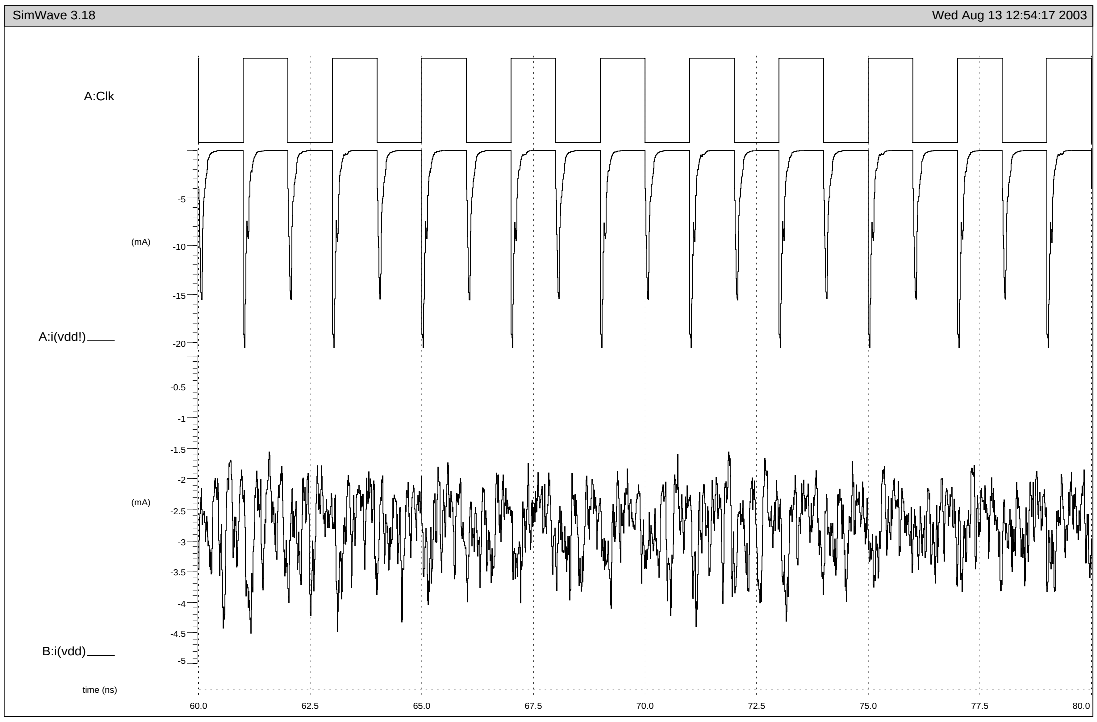


Figure 4.1 Typical power waveforms: synchronous (middle) and asynchronous (bottom)

find positions on the waveform where information is carried, making power analysis easier.

In self-timed circuits, however, each subcircuit performs its own function only when needed and is activated independently from the others. The power consumption of each subcircuit does not contribute towards the total power consumption coherently as it does in synchronous circuits. This decorrelation of activity means that any large peaks on the total power consumption waveform are fuzzy. Indeed, the power consumption of self-timed circuits is much smoother since it is the aggregation of many small peaks appearing at arbitrary times according to their handshake signals. To some extent, the power consumption from one module would be like noise to the other. Therefore, power dissipation peaks are hidden in the waveform.

Figure 4.1 shows that there is always a power dissipation pulse after the clock trigger edges in the upper trace. On the other hand, the lower trace shows that the waveform is much more like a noise signal. Periodic power dissipation peaks are masked out.

4.1.4 Overlapped and shifted waveforms

Power waveform shift and overlap is another problem raised by self-timed circuits for statistical analysis. Handshake signals dominate the behaviour of the circuit. The behaviour of one stage may be affected by the previous stage, and possibly by many other stages. For the same instruction, the cycle time can vary in length. It is not necessary for waveforms to have exactly constant periodicity. The statistical analysis of such power waveforms is hard as it is difficult to line up different cycles in order to compute the average value and to subtract this from each waveform to yield the difference.

Figure 4.2 shows a comparison of synchronous and asynchronous power consumption waveforms and the problems of waveform shift. The upper window illustrates two power traces for operations on a synchronous LFSR. Clearly, the two traces are perfectly aligned and the difference is shown. The lower window shows two traces for the same operations on an asynchronous LFSR. From the figure it can be seen that the two traces are temporally shifted and produce a significant difference. However, the visible difference does not correspond to a real difference between the two operations, but is mainly a result

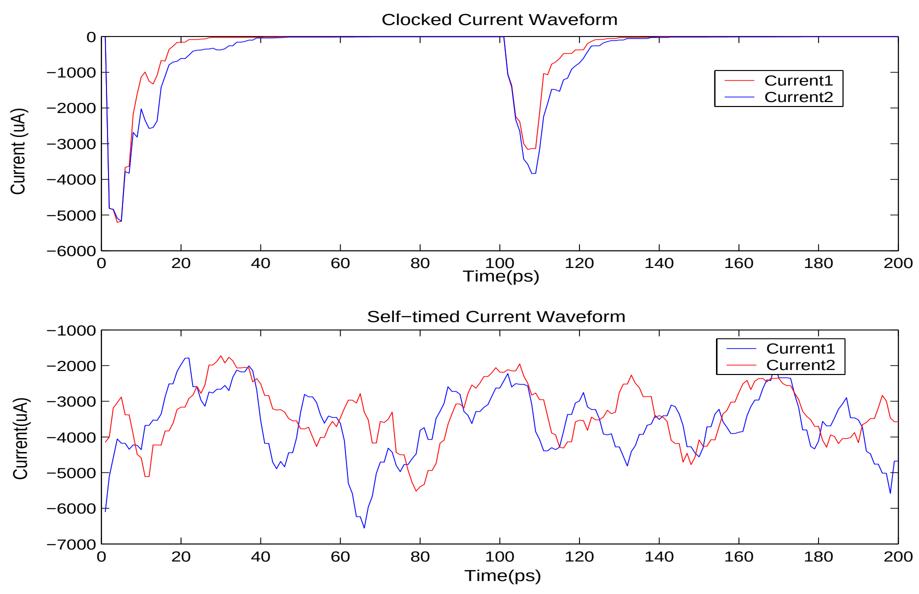


Figure 4.2 The problem of shifted waveforms

of the temporal shift problem. This means that the difference between asynchronous waveforms may contain not only the real difference, which attackers want to extract, but also the effects of temporal waveform shifting which can be considered as noise. In other words, attackers require extra time to realign those shifted waveforms.

In conclusion, differential power analysis, as applied to conventional synchronous circuits, cannot be applied directly to self-timed circuits. A new approach is required. The rest of the chapter describes a power analysis methodology which is suitable for asynchronous power waveforms.

4.2 Asynchronous differential power analysis

The power analysis of asynchronous waveforms is divided into three steps: data sampling, data preparation and data analysis. Data sampling collects power information from power simulation or measurements from the device under measurement. The data preparation step involves data partitioning and data correction, whilst the data analysis consists of waveform averaging, correlation analysis and information deduction. In addition, a signal-to-noise ratio analysis is undertaken to estimate to what extent the information-bearing signal can be extracted. Figure 4.3 shows a diagram of the asynchronous differential power analysis methodology, which was implemented in Matlab [55]. The method adjusts the conventional power analysis as used in synchronous circuits [8, 13] handling those difficulties pointed out in the previous sections.

First, power waveforms, extracted from either simulation or device measurement (see Section 4.2.1), are taken and partitioned into several sets of waveforms that correspond to the data set processed (Section 4.2.2). Next, a waveform shift correction function is performed to realign the waveforms (Section 4.2.3).

After that, each set of power waveforms is averaged to obtain its *typical characteristic waveform (TCW)* and a *common effect waveform (CEW)* is also computed by averaging the overall power waveform. In order to remove the common effect and to emphasize differences in the power waveforms, the common effect waveform is subtracted from each individual waveform and from their typical characteristic waveform. This forms *individual difference waveforms (IDW)* and their *typical difference waveforms (TDW)* (Section 4.2.4).

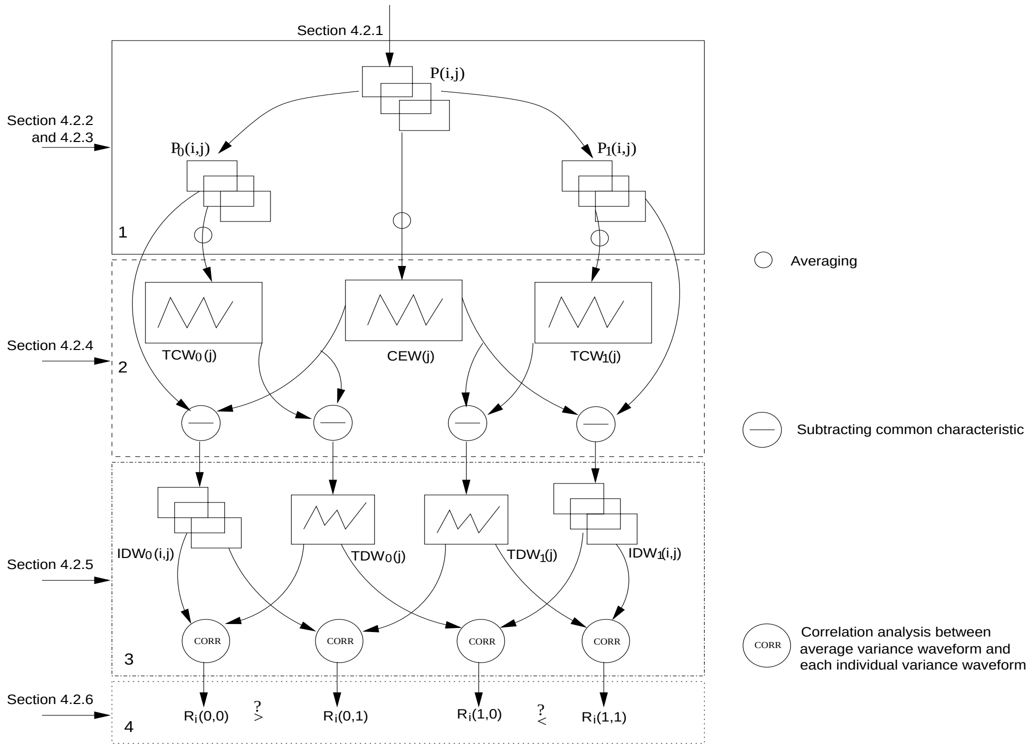


Figure 4.3 The new power analysis methodology

Finally, a correlation function is used to estimate how an individual difference waveform correlates with its typical difference waveform and with other typical difference waveforms that were not generated from the same set of waveforms (Section 4.2.5). A correct deduction is yielded if it has a stronger correlation with its own typical difference waveform than with others; otherwise an incorrect deduction results (Section 4.2.6). In addition, a signal-to-noise function, where the signal is the difference waveform and the noise is the average waveform, is performed to estimate how difficult the information is to extract (Section 4.2.7). The following sections describe the detail of the methodology.

4.2.1 Data sampling

To perform a power analysis attack, power information must first be sampled. Information is probably leaked when the circuit is processing data or executing instructions. In this thesis, two ways are used to sample data. When an analysis is performed using data from simulation, power information is recorded during the simulation using simulators such as PowerMill/Nanosim. When performing the analysis on a silicon chip, a LeCroy 9344CM oscilloscope [56] is used to measure power consumption across a series resistor in the power supply line. The potential difference is converted to digital format and transferred to a computer for analysis. For simulation, the sampling time resolution is normally set at 10 ps whilst the sampling frequency for measurement is set at 500 MHz or 250 MHz, which corresponds to 2 or 4 ns per sample respectively. For both synchronous and asynchronous circuits, this step is the same. When data is sampled, it forms a power waveform, denoted as $P(i,j)$, where the first index indicates the number of the cycle that was sampled for each iteration of the same operation, and the second index is the sample point in each cycle.

4.2.2 Waveform partitioning

Since each simulation or measurement involves processing different data for several hundred cycles, one power waveform, $P(i,j)$, extracted from data sampling consists of power fragments for several different data sets, $DT(k)$, that are being processed. To split the power waveform, a partitioning function is used to separate the waveform into several sets of waveforms, $P_k(i,j)$, that correspond to data sets. Each contains power waveforms for processing that particular data set. For instance, if the waveform contains power information for processing two sets of data, “1” and “0”, the power waveform is split into two sets of waveforms, $P_1(i,j)$ and $P_0(i,j)$ for data “1” and data “0” respectively. This step is illustrated as part 1 of Figure 4.3.

When partitioning a synchronous power waveform, the global clock signal is often used as a timing reference, because the clock signal controls when data is loaded and processed. Thus, a synchronous waveform is split according to the timing of the clock signal. On the other hand, in an asynchronous circuit with the lack of such a global signal, activities occur according to local handshake signals which are independent of each other. As a result, power may be consumed by a sub-circuit at any time. The power consumed by individual sub-circuits might be overlapped or shifted. In fact, the lack of this global synchronization signal makes the waveform partitioning one of the hardest problems in asynchronous power analysis.

However, as mentioned previously, an asynchronous circuit does display some characteristic periodicity because of the cyclic nature of the handshake controls and the algorithms being executed. Although it is approximate, this periodicity is still the inspiration for partitioning asynchronous waveforms that are approximately repetitive. Therefore, the handshake signals, such as the acknowledge signal, may be a possible timing reference for asynchronous waveforms. Here, an acknowledge signal is used as a timing reference to partition asynchronous waveforms. Note that those handshake signals are made available in this work for evaluation purposes only. It will be much more difficult for attackers because these signals may be not accessible from outside the cryptosystem.

4.2.3 Waveform correction

The data samples that are thus collected are, however, still not ready for analysis. Firstly, because an asynchronous waveform is not exactly periodic, the partitioned power waveform fragments may not cover the same interval. Secondly, as discussed in Section 4.1.4, asynchronous power waveforms may be shifted or overlapped. Thus, further actions are required to correct those partitioned waveform fragments. This correction consists of two steps: waveform stretch and waveform shift. The former is to make waveform fragments have the same interval and the latter is to correct the waveform shift problem. The step of waveform correction follows waveform partitioning in the step 1 and before waveform averaging.

Waveform stretch

After the waveforms have been partitioned, corresponding waveform fragments must be dilated or clipped to map them all onto the same basic periodic behaviour. There are several ways to stretch a waveform, such as shrinking, dilating or simple cropping. However, shrinking or dilating a waveform may damage the data, while cropping a waveform loses information that may reside in the cropped portion.

Figure 4.4 shows the different effects of shrinking and cropping the same waveform, which may be helpful when choosing a method for waveform stretching. The dark solid curve shows the cropped waveform which is formed by simply cutting the last ten sample points off the original waveform. The lighter solid curve is formed by shrinking the original waveform by ten samples. The upper dashed waveform illustrates the difference between these two waveforms. Obviously, because of shrinking, the original waveform is shifted to the left, causing the difference shown in the graph. The more the waveform is shrunk, the more it is shifted. On the other hand, the cropped waveform keeps most of the

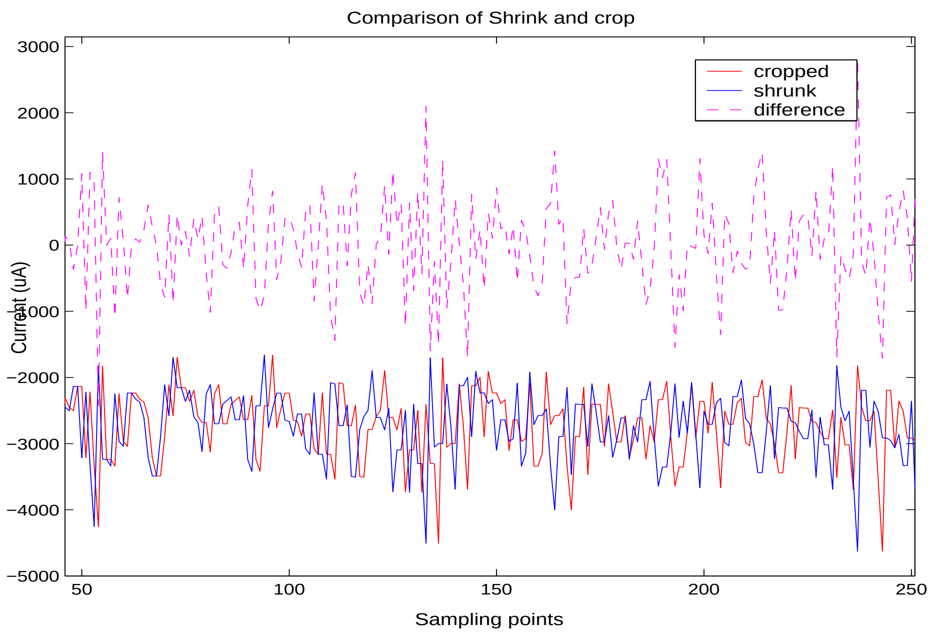


Figure 4.4 Shrunk and cropped waveforms

original power information, but loses the power information which is cut off. In most cases, however, the end of the power waveform has less impact on power analysis because the end of a waveform is normally occupied by the reset phase of the circuit. Therefore, a cropping technique is normally used to map waveforms onto the same period.

Waveform shift correction

Section 4.1.4 discussed the fact that an asynchronous power waveform may have the problem of the waveform being temporally shifted. This misalignment introduces a large amount of noise into the signal, obscuring the information-bearing component. Thus, a shift correction function may also be applied to those temporally shifted waveforms before other analysis is carried out. Because the misalignment of a power waveform varies in time, it is difficult to realign the whole waveform at once. Instead, this realignment must be a piece-wise process.

First, one waveform is taken as the base waveform and the others as the inputs to be adjusted. They are split into a number of small fragments, each containing a small group of peaks. The length of each fragment varies depending on the degree of fluctuation of the waveform. Generally speaking, the smaller the fragment, the better the correction quality can be, as a large fragment may contain different shift lags. The drawback is that using more fragments requires a longer computation time. Next, the shifting lag between each fragment and its corresponding fragment in the base waveform is computed, and the input waveform is shifted according to the lag calculated, such that the temporally shifted waveform can be realigned. Note that, because of the lack of a suitable formalized algorithm for the waveform separation, the results of waveform shift correction presented in this thesis were made according to my best effort.

Cross-correlation [57, 58] is used for this waveform shift correction. Cross-correlation estimates the correlation between two sequences over a range of lags. The correlation coefficient detects the extent that the two series are correlated. It ranges from -1 to 1: -1 indicates the strongest negative correlation, 1 indicates the strongest positive correlation, and 0 means there is no correlation between the two series. Equation (7) is an implementation of cross-correlation between two N-sample series, x and y, over a range of lags, d, from 0 to N on the y series:

$$r_{x,y}(d) = \frac{\sum_{i=1}^{N}[(x(i)-\hat{x})(y(i-d)-\hat{y})]}{\sqrt{\sum_{i=1}^{N}(x(i)-\hat{x})^2}\sqrt{\sum_{i=1}^{N}(y(i-d)-\hat{y})^2}}, d = 0, 1, 2, \ldots, N \quad (7)$$

where r(d) is the array of correlation coefficients between x and y when y is shifted by d. The highest correlation coefficient indicates the best match of the two waveform fragments and the lag of the best correlation extracted from this cross-correlation analysis is the shift time between the two fragments being compared. Thus, the fragments can be shift-corrected according this correlation lag. The whole waveform can then be corrected piece by piece.

We take the ARM multiply instruction, MUL, as a practical example to demonstrate the shift correction. The multiplier unit in SPA [19], an asynchronous ARM compatible processor, performs 32 additions internally, regardless of the values of its multiplier or multiplicand. Therefore, the strategy of using the waveform shift correction function is first to split the power waveform into 32 pieces, then the shift lags of each piece of power waveform are calculated and they are corrected in turn.

Figure 4.5 illustrates the power waveform for three of the 32 additions separated by the vertical lines. The top window in the figure shows a large amount of difference indicated by the dashed magenta curve. However, by looking at the two power waveforms it is not difficult to see that the difference is largely due to the two waveforms being shifted. Applying the cross-correlation function on the whole waveform, the shift lag is found to be about 0.62 ns. By shifting the second waveform left 0.62 ns the waveforms, as shown in the middle window in the figure, are formed. Now, the two waveform match much better and the difference between them is reduced.

However, small power fluctuations can be still found on the difference trace even though the waveform has been shift corrected. This is because the shifting lags for those three power fragments are slightly different. Applying a cross-correlation function to the whole

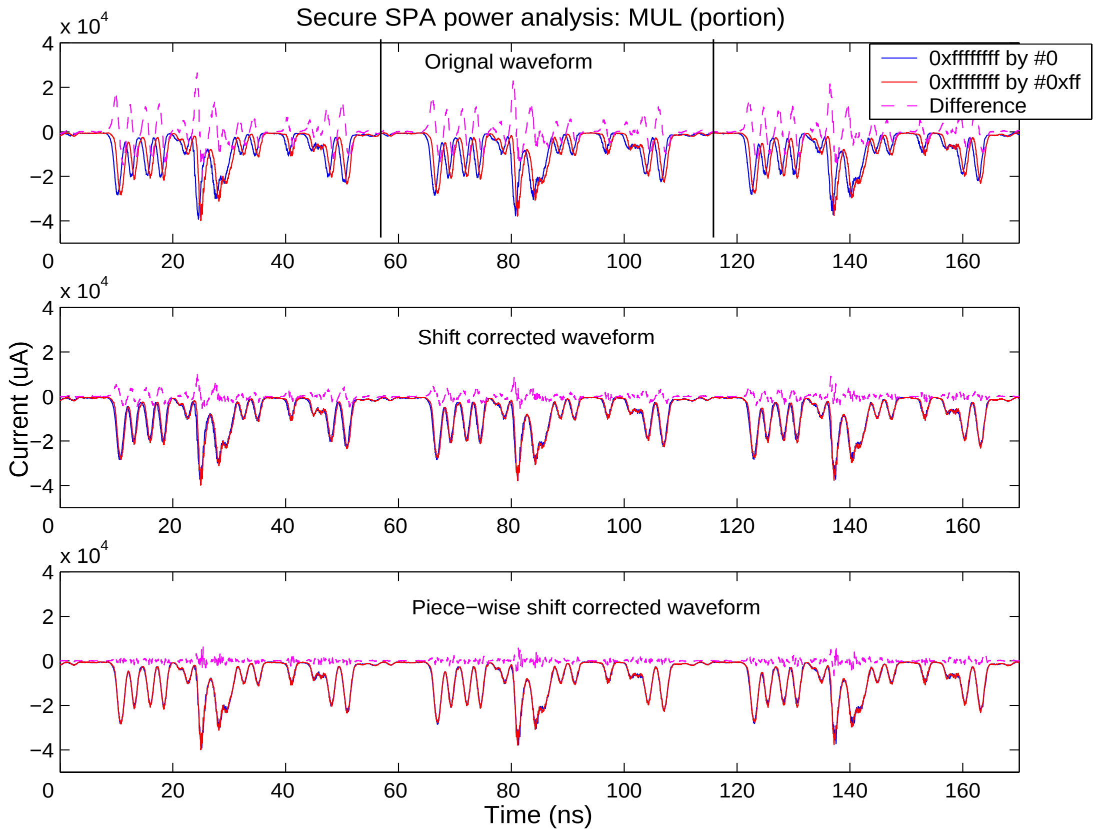


Figure 4.5 An example of waveform shift correction

waveform at once would correct only the major temporal shift. The traces in the bottom window are formed by applying the function to each addition fragment, finding the individual shifting lags and correcting piece by piece. The difference curve (the dashed magenta curve in the bottom window) is now much smoother than it was. In other words, the waveform shifting noise is significantly reduced.

While using the cross-correlation function we attempt to make the best shift correction of two waveforms and to reduce analysis noise caused by misalignment. It is, however, extremely difficult to eliminate such noise completely for a variety of reasons, such as the choice of where to split the waveforms. Because of those difficulties, the corrected asynchronous power waveform may still contain a small portion of waveform shift noise. Nevertheless, if an attacker knows exactly where the power difference partition is, for example where the target operations are performed, by focusing on this small partition rather than the whole waveform a more precise shift correction may be achieved.

4.2.4 Waveform averaging

This step is to average several waveform samples in order to extract the typical characteristic difference waveform for each data set and individual difference waveforms. This step is shown as part 2 of Figure 4.3.

After waveform partitioning and correction, the overall power waveform has been split into a set of partitioned waveforms $P_k(i, j)$, for example $P_1(i, j)$ and $P_0(i, j)$, where $i = 0, \ldots, I-1$, $j = 0, \ldots, J-1$ and $k = 0, \ldots, K-1$. This is because a whole power waveform contains $K*I*J$ sample points. For instance, j is the j^{th} sample point in the i^{th} cycle of the k^{th} data set. Typical characteristic waveforms, $TCW_k(j)$, for each of these partitioned waveform sets can be computed using equation (8);

$$TCW_k(j) = \frac{1}{I}\sum P_k(i, j), k = 0, \ldots, K-1 \tag{8}$$

where I is the number of all cycles of each set, for example $TCW_1(j)$ and $TCW_0(j)$. By averaging over all of the power waveforms, a common effect waveform, $CEW(j)$, is also obtained.

In order to remove the common effect from the typical characteristic waveforms to emphasis differences in the power waveforms, the common effect waveform, $CEW(j)$, is subtracted from each typical characteristic waveform, $TCW_k(j)$, forming typical difference waveforms, $TDW_k(j)$, following equation (9).

$$TDW_k(j) = TCW_k(j) - CEW(j) \tag{9}$$

In a power waveform where the consumption for one data pattern is distinguishable from that for the other data pattern, one typical difference waveform should be distinguished from the others. For instance, $TDW_0(j)$ should be distinguished from $TDW_1(j)$.

$$IDW_k(i, j) = P_k(i, j) - CEW(j), i = 0, \ldots, I-1 \tag{10}$$

Similarly, following equation (10), subtracting the common effect waveform from each individual waveform $P_k(i, j)$, the individual difference waveforms $IDW_k(i, j)$ are computed. One set of such waveforms (for instance $IDW_1(i, j)$ generated from processing the data 1 pattern) should show a strong correlation with the typical difference waveform which represents such a set of data patterns (for example *$TCW_1(j)$*) and a weaker correlation with others (for example *$TCW_0(j)$*).

4.2.5 Waveform correlation analysis

After suitably partitioned waveforms have been obtained and shift corrected, and enough control tests performed to obtain typical characteristic power waveforms for different data cases, the data is ready for correlation analysis. The analysis scheme discussed in this section is applied to evaluate the extent of information leakage from a circuit - to what extent does the power waveform reveal the data value being processed? It assumes, for simplicity, that only two cases must be distinguished, the data 1 case and the data 2case. This waveform correlation analysis process is shown as part 3 of Figure 4.3.

The use of a correlation analysis is necessary because an asynchronous power waveform does not display the very large peaks that characterize the synchronous circuit’s power waveform and reveal the data being processed. Instead, many small peaks can be found on the waveform. Subtracting the average makes the differences more relevant, but the waveforms are noisy.

Correlation analysis determines to what extent two variables are related; a mathematical comparison of how closely related two variables are. The correlation coefficient can range between ± 1. The higher coefficient means a stronger correlation between the two variables. A positive coefficient indicates a change in one variable results in change in the other in the same direction and negative coefficient means a change in the opposite direction. The correlation analysis employs a mathematical model shown in equation (11) that is similar to the one used for the cross-correlation analysis discussed previously (equation (7)); r is the correlation coefficient.

$$r_{x,y} = \frac{\sum_{j=1}^{N} [(x(j) - \hat{x})(y(j) - \hat{y})]}{\sqrt{\sum_{j=1}^{N} (x(j) - \hat{x})^2} \sqrt{\sum_{j=1}^{N} (y(j) - \hat{y})^2}} \qquad (11)$$

As shown in part 3 of Figure 4.3, which assumes only two sets of waveforms need to be distinguished, it analyses each individual difference waveform (*$IDW_1(i, j)$* and *$IDW_0(i, j)$* where $i = 0, \ldots, I-1$) with the two typical difference waveforms, *$TDW_1(j)$* and TDW_0(j). Thus, the analysis results in a set of 2×2 matrix of coefficients, denoted as *$R_i(x, y)$*, where $i = 0, \ldots, I-1$. For example, *$R_1(0, 0)$* is the correlation coefficient of *$IDW_0(1, j)$* and *$TDW_0(j)$* where *j=0,..., J-1*. *$R_1(0, 1)$* is the coefficient of *$IDW_0(1, j)$* and *$TDW_1(j)$*, where *j=0,..., J-1*. For *K* sets of waveforms, each with *I* cycles, there is a correlation coefficient matrix of *$R_i(x, y)$*, where *x=0,..., K-1*, *y=0,..., K-1* and *i=0,..., I-1*. This coefficient determines the degree of correlation between individual difference waveforms and typical difference waveforms.

4.2.6 Percentage of Correct Deduction (PCD)

After the correlation analysis, it is anticipated that if an individual power waveform fragment *P(i, j)* belongs to the data 1 set, then its individual difference waveform should display a strong correlation with *$TDW_1(j)$* and a weaker correlation with *$TDW_0(j)$*. If it belongs to the data 0 set the opposite should be true. To measure what percentage of these individual waveforms follow the above expectation, a simple method called “Percentage of Correct Deduction” or PCD is employed, calculating the percentage of correct deductions over all deductions. Part 4 of Figure 4.3 illustrates this computation.

The analysis yields a correct deduction if the above expectation is true; otherwise it yields a incorrect deduction. Thus, the PCD evaluates the degree of success of the correlation analysis. For an analysis of a situation as outlined where there are two alternative data values, a PCD of 100% would indicate perfect information extraction whereas a PCD of 50% would indicate that the analysis was extracting no information as the results are essentially random.

4.2.7 Signal-to-noise ratio

One of the reasons that makes differential power analysis such a successful non-invasive attack is the ability to extract information from the power signal. This information is extracted from side-channels by averaging noise out of the power signal. Therefore, when the DPA attack was developed, it was suggested that injecting random noise masks or hides the information-bearing signal so that even though the power signal can be measured by attackers, sensitive information would be still safe [8].

Noise on the power signal has many different forms, including external noise, intrinsic noise, sampling noise, quantization noise, algorithm noise [54] and also clocking noise. The activity of the global clock signal contributes towards the total dissipation, so the global clock signal may obscure the useful information in the power signature. In an asynchronous power signal, there is also a new form of noise introduced by the waveform shift and also by the intrinsic self-timing characteristic. Reducing and removing those noise sources to emphasize informative signals requires that the attacker be able to separate signal from noise.

The power analysis scheme described in this chapter illustrates how much information attackers may be able to extract from power supply characteristics. However, it does not explain how difficult the analysis would be. A ‘signal-to-noise ratio’ shows how large the information-bearing part of a signal is compared to the non-information noise that is also part of the signal, and from which it must be extracted. It gives some indication of the dynamic range required of the attacker’s equipment and the likelihood that extraneous electrical noise might obscure the information-bearing part of the signal.

Signal-to-noise ratio (SNR) is well studied in digital communication [59]. It is used to measure the magnitude of noise and signal, and to determine to what extent the meaningful signal may be extracted. Here, the signal is the individual difference waveform $IDW_k(i,\ j)$ and the noise is the common characteristic average waveform $TDW_k(j)$, Equation (12) is a simple Decibel function to compute the SNR of the power signal from experiments:

$$SNR\ =\ 20log_{10}\left(\frac{S}{N}\right) \qquad (dB) \tag{12}$$

where S and N are the root-mean-square of the information-bearing signal and that of the non-information noise respectively. Clearly, from the equation it is understood that the lower the SNR is the more difficult the analysis is, and thus lower is better.

4.3 Summary of chapter

The chapter has discussed some challenges for asynchronous power analysis, including the difficulties for waveform partitioning and waveform shift and overlap problems. A differential power analysis methodology for asynchronous power signals that copes with those difficulties has been introduced. This methodology realigns shifted power waveforms and uses a correlation function to derive information from power waveforms. It will be used in the following chapters to evaluate the power consumption characteristics of asynchronous systems and their potential abilities to enhance security against differential power analysis attacks.

Chapter 5: LFSR Design and Implementation

Self-timed logic is believed to have advantages for security-sensitive applications, such as smartcards. The absence of a clock as a reliable timing reference makes conventional differential power analysis attacks more difficult. However, the variability of the timing of self-timed circuits is a weakness that could be exploited by alternative attack techniques. In this chapter, several Linear Feedback Shift Registers (LFSRs) are designed using different technologies, including synchronous and asynchronous design styles and different data-encoding techniques such as single-rail and dual-rail. The design serves as a small and simple example to compare the security features of asynchronous and synchronous design styles and of single-rail and dual-rail encoding. To explore the benefits of various asynchronous design style in security applications, differential power analysis for asynchronous circuits, as discussed in the previous chapter, is used to measure the potential information leakage of each type of LFSR. The analysis also suggests hardware design techniques that help to hide information and minimize leakage.

5.1 Introduction

A Linear Feedback Shift Register or LFSR is the simplest kind of feedback shift register. The LFSR has been widely used to implement stream ciphers. Many military cryptographic systems, for example, centre around the LFSR. This is not only because LFSRs can easily be implemented in hardware, but also because LFSRs can offer stream ciphers a lot of security with only a few logic gates [27].

For instance, the A5 cipher system [60] used in GSM mobile phone system employs only three LFSRs and provides a secure solution to protect voice conversations on air. Governmental agencies once argued about whether the A5 cipher should be exported to particular countries [60]. However, the weakness of the A5 cipher is that its registers are short enough to make exhaustive search feasible. Anderson indicated that the divide-and-conquer attack actually takes only 2^{45} attempts as you only have to guess about the half of the bits in the third register [61] and Golic showed how to reduce the effort to 2^{40} by solving a linear equation [62]. Furthermore, Biryukov and Shamir reported that a successful real time attack on A5/1 only required a few seconds of the conversation and a few seconds to compute the key [63]. Nevertheless, the A5 cipher is efficient and passes all known statistical tests [64] and is currently used in the GSM mobile system.

Due to the simplicity of the feedback of LFSRs, cryptologists have tried to analyse their sequences. To predict the whole output sequence of a single n-stage LFSR, it is necessary to know only 2n bits of the output sequences [65]. Mathematical attacks, such as the Belekamp-Massey algorithm and divide-and-conquer attack, analyse sequences from LFSRs and are able to discover the output of a cipher system which uses LFSRs without performing a brute-force search [27]. Once the LFSR form is known, the output sequence can be determined and the stream cipher which employs the LFSR can be broken. On the other hand, as the designer increases the linear complexity of the sequence to ensure the LFSR is random enough to be secure, such mathematical attacks become more difficult. Furthermore, the more complex feedback the function uses, the more difficult the attacks are. Even though a single LFSR is not secure enough to defend against such attacks, LFSRs are still very common basic blocks used to build cipher systems. Many real world stream cipher systems, such as A5, XPD/KPD, employ LFSRs.

As cryptographers have used LFSRs widely in algorithms, and DPA has been successful on many smartcard applications that use encryption, it is useful to take the LFSR as a small point example to evaluate the different power consumption characteristics of synchronous and asynchronous circuits and different data encodings, and to demonstrate how asynchronous circuits may improve resistance to differential power analysis attacks. This chapter, therefore, addresses this goal in order to investigate the potential benefits of asynchronous circuits in security applications.

This chapter will first look at the basic theory of the LFSR and its use in cryptography. Then the details of the design and implementation of different styles of LFSR are described, including a single-rail clocked LFSR, a conventional dual-rail clocked LFSR a secure dual-rail clocked LFSR, a single-rail asynchronous LFSR (also known as bundled data LFSR), a conventional dual-rail asynchronous LFSR and a secure dual-rail asynchronous LFSR. Note that the dual-rail designs discussed in this chapter always use a return-to-zero or 4-phase protocol unless otherwise stated.

5.2 Feedback shift registers

A feedback shift register consists of two main parts: a shift register and a feedback unit. A shift register is a device whose function is to shift its contents into adjacent places, or stages, within the register or out of the register if the position is at the end. The stage at the other end becomes empty if there is no input, or filled by new contents shifted into the register. The length of the shift register is the number of bits or stages of the shift register. An n-bit register is known as an n-stage register. Each stage is a binary storage element built from flip-flops or other memory devices.

In modern digital systems, registers are widely used as data storage and state-holding devices. A shift register not only has a data storage capability but also has the ability to move data. Figure 5.1 shows several types of shift register data movement commonly used in digital systems: serial in-serial out, serial in-parallel out, parallel in-serial out and a feedback shift register. From the figure we can see that a shift register can be constructed from FIFO or buffer elements (a in Figure 5.1). It can also be used to convert from serial data to parallel data and vice versa (b, c). Moreover, another function of a shift register is to delay a serial data stream. By simply shifting the bits from one end of the shift register

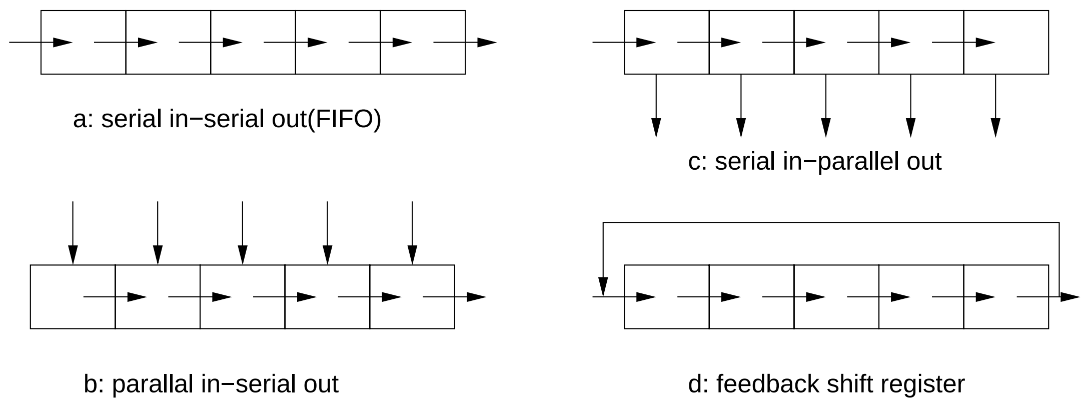

a: serial in−serial out(FIFO)
c: serial in−parallel out
b: parallal in−serial out
d: feedback shift register

Figure 5.1 Shift register data movements

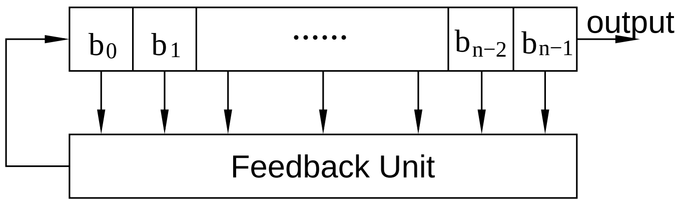


Figure 5.2 A feedback shift register

to the other end, it provides a delay equal to the length of the shift register (d). A feedback shift register is formed by taking some bits of the shift register as input to the feedback block whose output is a single bit which goes into the position that becomes empty after shifting; normally it is the left-most bit. Because feedback shift registers can generate pseudo-random sequences, they are often used to build random number pattern generators (RNPG), which are widely used in applications such as Built-In Self-Test (BIST) circuits [66, 67], error correction circuits [68, 69] and cipher systems. In this thesis, their use in cryptographic systems is emphasised. Before going into more detail, some nomenclature should be introduced:

- *Clock*: A shift action occurs only when the clock changes either from “low” to “high” or from “high” to “low” depending on the implementation. So a change on the clock input means bits in the register are shifted by one position. In stop-go style stream cipher systems, the clock could be formed from control input signals.
- *Shift direction*: A shift register can shift in either direction, right or left. However, in most cipher systems, they usually shift right. In this chapter, shift registers are considered always to shift to the right.
- *Input*: the input bit is put into the left-most bit of the register, which otherwise becomes empty after a shift.
- *Output sequence*: the right-most bit of the register is shifted out during a shift action. This bit is known as the output sequence.
- *Tap sequence*: the list of register bits which are used to construct the feedback block is referred to as the tap sequence.
- *Feedback*: a function of the tap sequence bits of the shift register, denoted as $f(b_0,b_1,...,b_{n-1})$, yields a single bit which is the input to the shift register.

5.3 Theories of LFSR

If the feedback of a shift register $f(b_0,b_1,...,b_{n-1})$ can be expressed in the form of equation (13), where c_i, the so-called feedback coefficients, are either 0 or 1, and all addition is over the Galois Field of order 2, GF(2), then the shift register is a linear feedback shift register or LFSR [65].

$$f(b_0, b_1, \ldots, b_{n-1}) = c_0 \cdot b_0 \oplus c_1 \cdot b_1 \oplus \ldots \oplus c_{n-1} \cdot b_{n-1} \tag{13}$$

Basically, the feedback function is an XOR of the tap sequence, which means the function adds all the selected bits and gives a “1” output if the result of the addition is odd, otherwise the output is “0”. Figure 5.3 shows an n-stage LFSR. Theoretically, an n-stage LFSR can generate a pseudo random-sequence of length 2^n-1. Because this feature offers good security for a small hardware price it has been much used by cryptography designers.

5.3.1 LSFR, a state machine

The contents of a shift register can be recognized as state. Every movement in the shift register can be treated as moving from one state to another. Thus, an LFSR is a class of state machine [65]. For any given state, there is a unique following state and a unique preceding state with one exception that when the content is zero the following and preceding states are also zero.

The content of the register, feedback function and the tap sequence together describe the state of the LFSR. It is easy to see that there are 2^n different states for an n-stage register. Since a state can have only one preceding and only one succeeding state, an LFSR with a

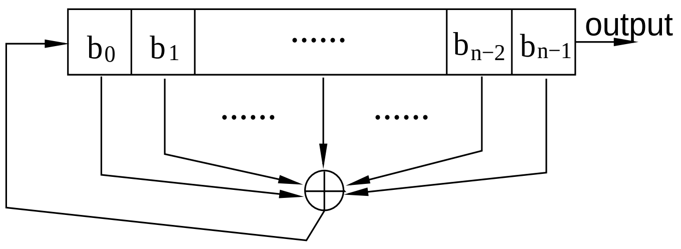


Figure 5.3 A linear feedback shift register

particular tap sequence will pass through every non-zero state once and only once before it repeats. Thus, the largest possible state space for an LFSR is $2^n - 1$ states, that is, all possible states apart from the zero state, which is a sequence with one and only one state, itself.

The output sequence is a corollary of this behaviour. Apart from the output sequence generating from the initial state of zero, which is the null sequence, for any feedback network LFSRs with different initial states produces different output sequences with periods up to $2^n - 1$. An LFSR with a given feedback network producing an output sequence with a period of $2^n - 1$ gives the maximal output sequence. If this exists then clearly every non-zero state will appear in the cycle and the length of this LFSR sequence does not depend on the initial state.

5.3.2 Maximal periodicity of an LFSR

The output sequences of a linear feedback shift register with feedback coefficient c_0=1 is periodic with period of $p \le 2^n - 1$. An n-stage LFSR can generate a output sequence with a maximal length of $2^n - 1$. However a LFSR is not always guaranteed to have the maximal periodicity. What makes an LFSR have a maximal period?

Since the maximal period of an n-stage LFSR is $2^n - 1$ and every state can have one and only one preceding and following state, every non-zero state will appear in the sequence once. Therefore, the choice of initial state may effect the length of the output sequence of an LFSR when it does not have maximal periodicity, but this is not true if the LFSR has maximal output period. In fact, whether an LFSR has the maximal output sequence or not only depends its feedback, or tap sequence. In order to understand how an LFSR may be implemented to obtain a maximal output sequence, a characteristic polynomial is formed from the tap sequence, shown in equation (14):

$$f(x) = c_0 + c_1 x + \dots + c_{n-1} x^{n-1} + x^n \tag{14}$$

Where c_i is the tap sequence and the degree of the polynomial is the length of the shift register. Note that the polynomial here is over GF(2), unless otherwise mentioned.

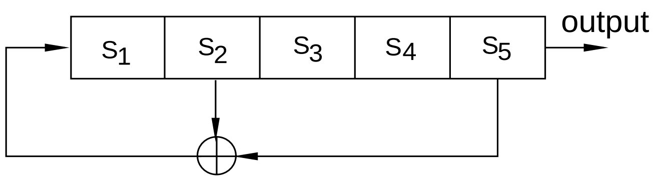
Because a polynomial over GF(2) is primitive if and only if it has order $2^n - 1$, to obtain the maximal output sequence period an LFSR must have its characteristic polynomial be primitive over GF(2) [27]. A *primitive polynomial* is a polynomial which generates all elements of an extensive field from a base field. A primitive polynomial is also an *irreducible polynomial*, that is a polynomial which cannot be factored into non-trivial polynomials in the same field [70].

The number of primitive polynomials of order n over GF(q) is determined by $\frac{\phi(q^n - 1)}{n}$, where $\phi(p)$ is Euler's *totient function*, the number of positive integers less than or equal to and *relatively prime* (no common factor) to *p*, where 1 is counted as being relatively prime to all numbers [70]. Since a number less than or equal to and relatively prime to a given number is called *totative*, $\phi(p)$ can be expressed as the number of totatives of *p*. For example, $\phi(7) = 6$ because it has six totatives (1, 2, 3, 4, 5, 6). The first ten $\phi(p)$ for p=1,2,...,10 is 1, 1, 2, 2, 4, 2, 6, 4, 6, 4. From the above description, it can be seen that the number of primitive polynomials over GF(2), $\frac{\phi(2^n - 1)}{n}$, for degree n=1,2,... is 1, 1, 2, 2, 6, 6, 18, 16, 48.... [71].

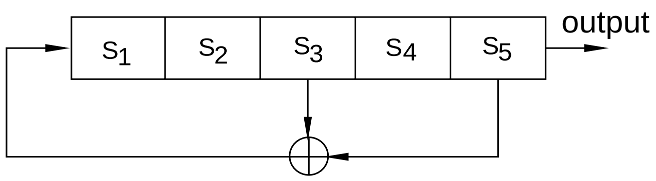
n	2^n-1	$\phi(2^n - 1)$	$\frac{\phi(2^n - 1)}{n}$	Primitive Polynomials
1	1	1	1	x
2	3	2	1	$1+x+x^2$
3	7	6	2	$1+x+x^3$, $1+x^2+x^3$
4	15	8	2	$1+x+x^4$, $1+x^3+x^4$
5	31	30	6	$1+x^2+x^5$, $1+x^3+x^5$, $1+x+x^2+x^4+x^5$, $1+x+x^2+x^3+x^5$, $1+x+x^3+x^4+x^5$, $1+x^2+x^3+x^4+x^5$
6	63	36	6	$1+x+x^6$, $1+x^5+x^6$, $1+x+x^2+x^5+x^6$, $1+x+x^4+x^5+x^6$, $1+x^2+x^3+x^5+x^6$, $1+x+x^3+x^4+x^6$

Table 5.1: Primitive polynomials

This shows that for a large number n, there is a wide range of primitive polynomials available to construct maximum period sequences. However it is not necessarily true that a larger number n will guarantee a larger choice of primitive numbers. Table 5.1 shows



(a)



(b)

Figure 5.4 5-stage maximal period linear feedback shift registers

primitive polynomials for the first few n from 1 to 6. From the table we can see that, for example, there are six primitive polynomials for n=5 and n=6.

To generate a maximum output sequence LFSR using a primitive polynomial, for example $1 + x^2 + x^5$, the LFSR should be a 5-stage register with feedback XORing the S_2 and S_5 taps as shown in Figure 5.4(a). Alternatively, an LFSR with a maximal output sequence can be also constructed by using another primitive polynomial, $1 + x^3 + x^5$. In this case, the feedback function of the LFSR XORs the S_3 and S_5 tap as shown in Figure 5.4(b).

Both designs generate maximum period output sequences, but the order of the output sequence varies because of the different taps, and the starting point of the output sequence is determined by the initial state which is usually the security key of the cryptographic implementation.

Table 5.2 compares the state sequences of two 5-stage LFSRs having maximum length output sequences and with initial state of 00001. The unshaded states column is the state sequence of the LFSR $1 + x^2 + x^5$, which means the feedback is XOR taps S_2 and S_5. The shaded states column is the state sequence of the LFSR $1 + x^3 + x^5$, whose feedback is from taps S_3 and S_5. The table clearly shows that although both LFSRs have the same initial state and generate maximum output sequences, the order of the sequence varies due to the different tap sequences used. For example, at the fourth state, one LFSR changes state from 00100 to 10010 whilst the other changes from 10100 to 01010.

N	States		N	States		N	States		N	States	
0	00001	00001	8	01101	01110	16	00111	11100	24	01110	10110
1	10000	10000	9	00110	10111	17	00011	11110	25	10111	01011
2	01000	01000	10	10011	11011	18	10001	11111	26	01011	00101
3	00100	10100	11	11001	01101	19	11000	01111	27	10101	10010
4	10010	01010	12	11100	00110	20	01100	00111	28	01010	01001
5	01001	10101	13	11110	00011	21	10110	10011	29	00101	00100
6	10100	11010	14	11111	10001	22	11011	11001	30	00010	00010
7	11010	11101	15	01111	11000	23	11101	01100	31	00001	00001

Table 5.2: Comparison of sequences of two 5-stage LFSRs

Obviously, designing an LFSR, there are several different choices of tap sequences that guarantee the maximum periodicity. However, the inner changes of state will vary from one to another. The different changes of state result in different changes of Hamming weight distances between states. Since different Hamming distances could induce different power consumption, concern arises about the choice of an appropriate primitive polynomial to implement an LFSR so that the change of Hamming weight between two serial states can be kept constant. It is therefore necessary not only to check for randomness theoretically but also to examine other issues such as Hamming weight distance in order to optimize security and minimize potential information leakage.

5.3.3 The LFSR in real cryptographic applications

Earlier in this chapter it was mentioned that LFSRs have been used in many applications. In Built-In Self-Test (BIST) circuits, LFSRs are employed to generate a random test pattern. In digital communication applications LFSRs are also used by error-correction circuits to encode data into a format such that if an error occurs during communication it can easily be detected and corrected. Moreover, cipher system designers use LFSRs as basic blocks from which to build stream ciphers. Particularly in the early VLSI era, when the scale of an integrated circuit could not accommodate a large amount of hardware, LFSRs could be implemented cheaply while offering good security. LFSRs are still widely employed in modern cryptographic applications. The following section introduces the A5/1 stream cipher which has been widely used in GSM mobile phone systems.

A5/1

A5/1 [60] is the cipher algorithm used in the GSM mobile phone system to encrypt and protect voice conversation during transmission. It consists of three LFSRs (with lengths of 19-, 22- and 23-bits) denoted by R1, R2 and R3 respectively. The taps on R1 are bits 13, 16, 17 and 18; the taps on R2 are bits 20 and 21; the taps on R3 are bits 7, 20, 21 and 22. This gives all LFSRs maximal period with lengths of 2^{19}-1, 2^{21}-1 and 2^{23}-1 for R1, R2 and R3 respectively. The three LFSRs are clocked by a stop-and-go control signal using a majority rule. Those LFSRs whose clocking taps, C1, C2 and C3, agree with the majority bits (two or three of the clocking taps), which are calculated every cycle, are actually clocked. That is, every cycle at least two LFSRs are 'go' and zero or one LFSR will 'stop'.

Figure 5.5 shows the architecture of the A5/1 stream cipher system used in GSM mobile phones. The GSM conversation is framed every 4.6 milliseconds and digitized into 228 bits, that is, 114 bits representing A to B communication and 114 bits representing B to A communication. For every frame, the A5/1 generates a 228-bit pseudo-random number to

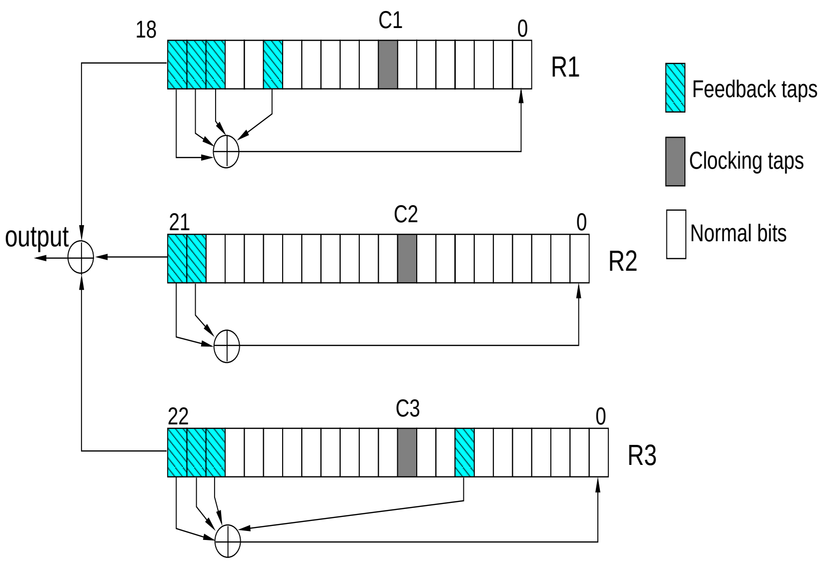


Figure 5.5 A5/1 cipher system architecture in GSM

XOR with the 228-bits of data to yield 228-bit ciphered text. This pseudo-random number is generated according to a 64-bit session key K and 22-bit frame number counter Fn that is publicly known. During the key initialization period, the stop/go clocking strategy is disabled and session key K is first XORed into the three LFSRs in parallel in 64 clock cycles and then the 22-bit frame count Fn in the next 22 clock cycles. The resulting state of A5/1 is called the initial state. Then, the cipher is clocked for another 100 cycles with the stop/go operation on but without outputs. During the next 228 clock cycles after that, the cipher generates 228 bits of output by XORing the three output bits of the LFSRs. The 228 bits are used to XOR with the voice signal captured each frame.

5.4 Synchronous LFSR design

The previous sections discussed the basic theory behind the LFSR. LFSRs are not only an indispensable component in modern digital system design, but also an important element in cipher system design, especially designing a stream cipher system. Most cryptanalyses on stream ciphers are based on mathematical analysis which is more concerned with the weakness of the encryption algorithm than the physical weakness of the hardware design. Since the development of the differential power analysis attack, many cryptosystems have been broken including smartcards which are supposed to be one of the most solid security solutions.

Investigations into differential power analysis and its countermeasures have been carried out on clocked circuits such as synchronous smartcard microprocessors [8, 13]. The task for attackers is to maximize side-channel information leakage and therefore to make an attack more successful and spend less effort on such an attack. In the reverse direction, the goal for cryptosystem designers is to minimize side-channel information leakage to protect the devices from such attacks. However, minimizing such leakage from side-channels is not always easy, and it is extremely difficult to eliminate the side-channel information entirely. If side-channel information leakage exists, there may be a way to extract and analyze information from it.

In clocked systems, the global clock signal is very helpful to such analysis by providing a time reference. Asynchronous logic, however, has no such global clock signal. Instead, it has many local synchronization signals to control communication in different sub-circuits asynchronously. Thus, the power consumption of one sub-circuit is independent of the others. The overall power consumption of the entire circuit will be unlikely to have the obvious peaks and predictable cycle that it has in synchronous circuits. The time reference which is available to attackers of synchronous circuits no longer exists in asynchronous circuits. Consequently asynchronous logic design has been assumed to improve resistance to non-invasive attacks. Furthermore, dual-rail encoding uses two wires to present one logic bit, “1” or “0”. A transition on one wire indicates logic “1” and on the other means “0”. Hence, it will always require one transition to send either “1” or “0”, so the power consumption of sending “1” and “0” would be the same and therefore side-channel information leakage is minimized.

However, the extent to which security is improved by removing the time reference and the extent to which other techniques can easily be adapted to provide further improvement has yet to be determined. These questions must be answered if people are to be convinced that asynchronous design has the potential to protect against non-invasive attacks in security applications. This chapter and those following present an evaluation and comparison of the level of information leakage of synchronous and asynchronous logic and the security they provide in terms of preventing non-invasive attacks such as power and timing analysis attacks.

As the LFSR plays an important role in cryptosystems and is simple, using it as a example to investigate the advantages and/or disadvantages of asynchronous logic is a good starting point. The architecture shown in Figure 5.6 was implemented using several different technologies to evaluate the security imparted by those different techniques. Because the longer period of an LFSR can make power analysis more difficult, for simplicity, the chosen LFSR has five stages with tap sequences of c_4 and c_5, with the result that the polynomial for this LFSR is not a primitive polynomial. Therefore this LFSR does not have a period of maximal length. In fact, it has only 21 different states with

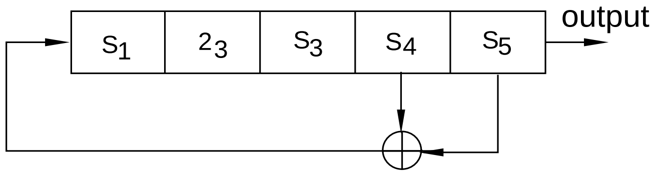


Figure 5.6 A linear feedback shift register

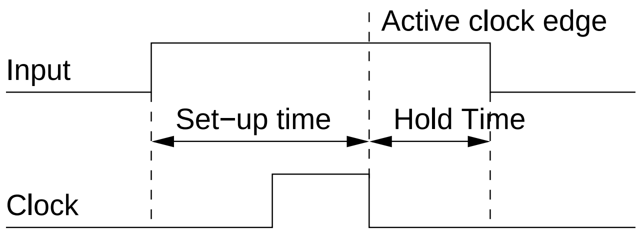


Figure 5.8 Flip-flop set-up and hold times

the initial state of 00110. However, since the aim of this experiment is to evaluate the security impact of different technologies, this LFSR should be suitable for the task.

5.4.1 Single-rail clocked LFSR

The first LFSR implementation shown in Figure 5.7 is a single-rail clocked LFSR. It uses edge triggered D-type flip-flops (DFF) controlled by a clock signal, and a simple XOR is used to generate the feedback sequence. It is initialized to have initial state 00110. In normal operation the circuit generates an output bit continuously as long as the clock signal is provided. In each clock cycle one output bit is shifted out. Note that the input data to a DFF must be valid before the clock edge and the output becomes stable after the clock triggering edge. These constraints are called the flip-flop set-up and hold times, as shown in Figure 5.8.

Clearly, the shifting of data in this LFSR is dominated by the global clock signal. All the shift and store operations will be activated on the clock rising or falling edges. Consequently, power consumption will arise within a short time after the clock edges and produce an obvious power consumption pulse.

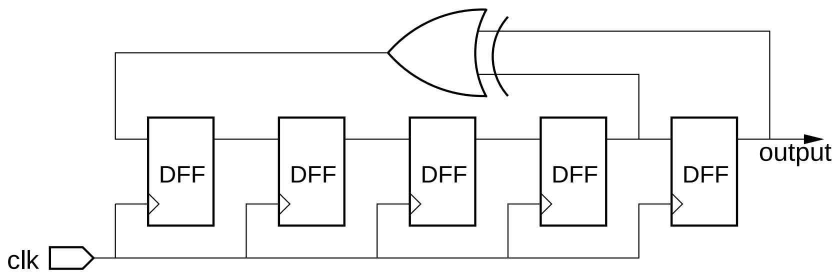


Figure 5.7 Single-rail clocked LFSR

5.4.2 Conventional dual-rail clocked LFSR

Dual-rail encoding uses two wires to transmit binary data. A transition on one wire transmits one binary value. A transition on the other wire indicates the other value. With Return-To-Zero or RTZ, dual-rail encoding adds a “null state” after every valid data value. Therefore, two transitions will always be required to transmit one bit regardless of its value. The details of dual-rail encoding can be found in chapter 3.

The basic idea behind this signalling protocol is that it always requires the same number of transitions whatever data is presented, in this case, two transitions. The advantage of this protocol is that as the number of transitions needed to transmit different data is balanced, the power consumption will be also. Other N-of-M encodings, such as 1-of-4 encoding, would achieve similar security benefits.

The implementation of a dual-rail clocked LFSR is similar to the single-rail one. However, instead of using DFFs, it uses edge-triggered RS latches as shown in Figure 5.9. The paired data value from the previous stage is presented on W0 and W1 and stored in the first RS latch (master) when Clk is low. At the same time the second RS latch (slave), in which the current data is stored, sends its data to the next stage. When Clk returns to high, the data in the master is propagated to, and stored, in the slave latch. Meanwhile the output (R0 and R1) return to zero before the slave outputs the next valid data. Note that two XORs are also required to generate the dual-railed encoded feedback signal. The Reset and the Set pins are for initializing the circuit. The circuit is set when “1” is assigned to the Set pin while the Reset pin is held at “0”. Similarly, assigning “1” to the Reset pin will reset the circuit, which means the output is set to 01, which is ‘0’ in dual-rail code.

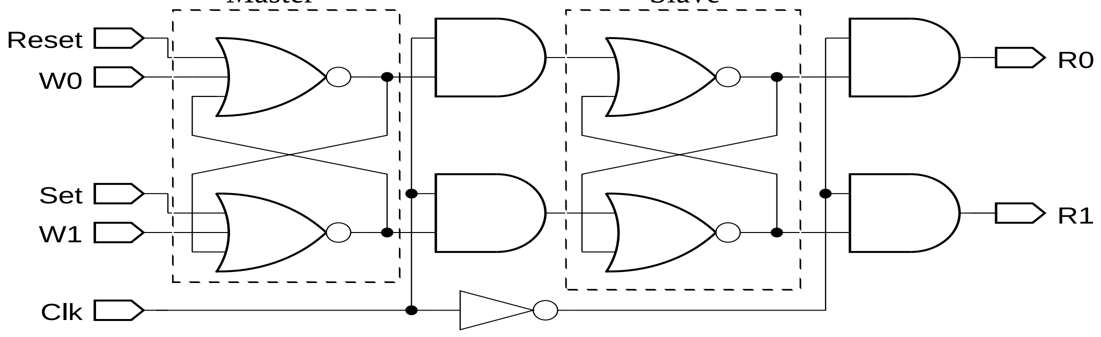


Figure 5.9 One stage of the conventional dual-rail clocked LFSR

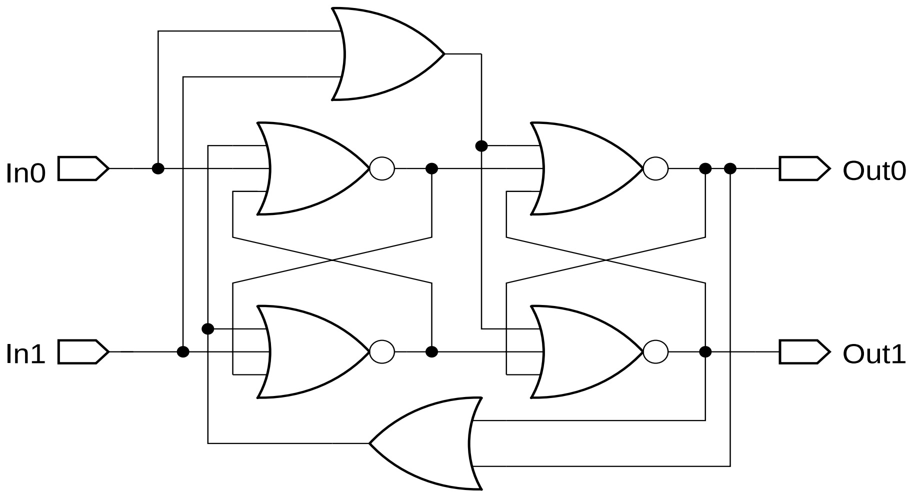


Figure 5.10 Secure dual-rail clocked latch

5.4.3 Secure dual-rail clocked LFSR

Whilst conventional dual-rail encoding provides balanced power consumption, this balanced encoding is still not totally symmetrical. It is still possible to determine the Hamming weight distance between the data value presented to a latch and the data previously stored in the place. Looking at Figure 5.9, it can be seen that if the data value the latch is loading and the data value which already occupies the latch are the same, the latch will dissipate less power than if the two data values are different. This occurs because the RS latch remains unchanged until a different data value is presented due to the fact that the cross-coupled connection of the RS latch will not consume power if its inputs remains the same. Thus, if the data value stored previously in the latches can be reset before a new data value is loaded, then this difference in power consumption will be reduced since there will always be two transitions needed for signalling every data value (one rising transition to indicate the data value, and one falling transition for the return of the wire pair to the null state).

The secure dual-rail clocked LFSR is implemented by replacing the master and slave conventional RS latches as shown in Figure 5.9 with the specially designed secure dual-rail latches as shown in Figure 5.10. The secure latch, also used in SPA [15], employs two pairs of cross-coupled NOR gates that operate in a master-slave configuration. Data presented on the In0 and In1 wires is stored in the master latch and causes the slave to go into the spacer (null) state to erase the previous data information stored there. Returning In0 and In1 to zero causes the data to be transferred to, and stored in, the slave using dual-rail encoding and it is then erased from the master which goes into the spacer state. In this way, the two RS latches operate in the normal and spacer state alternately and erase the stored information between two successive values.

5.5 Asynchronous LFSR design

Having described three different types of clocked LFSR which use different signalling encodings for the purpose of evaluating the impact of the encoding, we now look at the asynchronous or clockless LFSRs to see how they differ from their clocked counterparts, and how the different signalling encoding protocol will affect their asynchronous power signals. Three asynchronous LFSRs using the same signalling encodings as used in the clocked LFSRs are described. The first of these is the single-rail or bundled-data asynchronous LFSR.

Note that, in a clocked circuit, all data movement is controlled by the global clock signal. In asynchronous circuits, however, the data movement is controlled through the handshaking communication. To be able to communicate, a five stage LFSR has to have five empty stages and at least one extra stage as the bubble stage. The more bubble stages that are added, the faster the data will be transmitted because there are more tokens. However the speed will be reduced if too many bubble stages are inserted because the number of tokens stays the same, but they have to travel through more stages [43]. The number of states that are needed scales linearly with the number of stages holding valid data. In these asynchronous LFSRs, a total of fifteen stages is used, including five stages for valid data, five empty and five bubble stages to speed up the transmission in the latch control circuit. Unlike clocked LFSRs where the circuit is controlled by the clock, the communication between stages in an asynchronous LFSR is controlled by handshake signals between the two communicating parts.

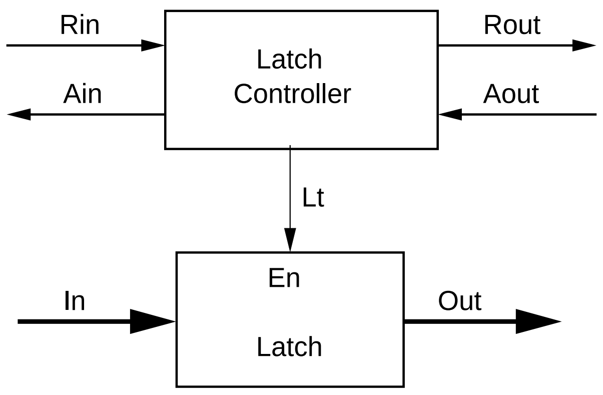


Figure 5.11 Asynchronous bundled-data LFSR stage structure

5.5.1 Single-rail (bundled-data) asynchronous LFSR

The single-rail asynchronous LFSR contains two parts: the data path built from latches which store the data, and the latch control circuit handling the handshake signals. Figure 5.11 shows one stage of an asynchronous LFSR. The data path here is formed simply from transparent D latches which are opened and closed by the control signal Lt, which is essentially the Rout signal, from the latch control circuit. The latch control circuit can be as simple as a two-input C-element, which sets its output high when both inputs are high and returns to low when both inputs have gone low.

Figure 5.12 shows a three-stage latch controller using a 4-phase Muller pipeline [43]. When the request from the previous stage rises and the acknowledge from the next stage returns to zero, the control circuit fires the latch control signal Lt to close the latch. At the same time, it sends an acknowledge back to the previous stage, indicating that the data has

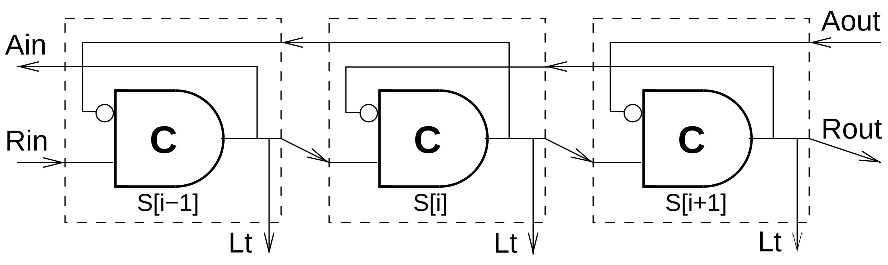
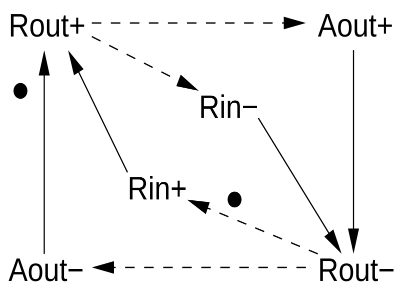


Figure 5.12 An asynchronous latch controller using a 4-phase Muller pipeline



Solid arrows indicate transitions managed by the latch controller itself

Dashed arrows indicate transitions ensured by the latch controller environment

Figure 5 .13 STG of an asynchronous latch controller

been received, and a request to the next stage to indicate that there is a new data ready for transmission. When the previous stage returns its request signal to zero and an acknowledge from the next stage has arrived, the controller returns the Lt signal to zero to open the latch again, and also returns the acknowledge for the previous stage and request for the next stage to zero. Thus, one full four-phase handshaking communication is completed. The signal transition graph (STG) [72] of the latch controller is illustrated in Figure 5 .13.

Note that the delay in the data path and the latch controller must be handled with extreme care. Lt must be raised to close the latch after the input data is ready. The STG reflects the handshake sequence on the input and output sides. Rin+ indicates the readiness of input data and is followed by Rout+ which indicates the availability of output data. The latch is opened and closed by signal Lt, and the input data is acknowledged by Ain. This is a simple latch controller for bundled-data self-timed circuits. More complex and faster latch controllers are possible and can be found in the literature [73].

5.5.2 Conventional dual-rail asynchronous LFSR

Figure 5.14 shows one stage of the conventional dual-rail asynchronous LFSR using conventional latches. One conventional RS latch is used in every stage for the data path. Unlike bundled-data circuits where data must be valid before the request signal arrives, this conventional dual-rail asynchronous LFSR is a speed-independent circuit which is insensitive to gate delays. It has a completion detection tree to detect the arrival of data.

The output of the completion detection is similar to the request signal used in a bundled-data asynchronous LFSR. In this case, it uses two AND gates to detect whether the values before the RS latch and after the RS latch are the same. When they are the same then the data has been stored in the latch and the detection circuit sends a “Write Done” signal, WD, to the handshake component, a C-element, indicating the arrival of a new data value; otherwise the data value presented is the null state, 00, and the detection circuit returns the WD signal to zero.

When the WD signal is high and the Read acknowledge signal, Rack, from the next stage has returned to zero, indicating that the next stage is ready to accept another data value, the handshake circuit generates a Read Request signal, RR, which enables the two AND gates to send the data value to the next stage. Meanwhile a Write acknowledge signal, Wack, is sent back to the previous stage indicating the acceptance of the data. At some point, the previous stage returns the data value to null, and the completion detection circuit returns the WD signal to zero. On the other side, the next stage will also return the acknowledge to high, and the signal, RR, is then returned to zero to turn off the AND gates. It also sends Wack low to the previous stage indicating readiness to accept the next data. Then, the 4-phase handshake protocol is completed.

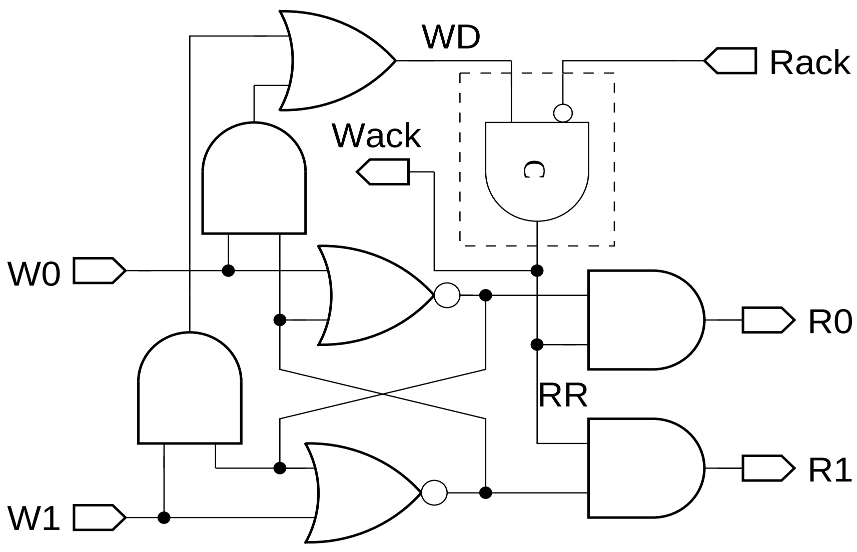


Figure 5.14 Latch for conventional dual-rail asynchronous LFSR

Like the conventional dual-rail clocked LFSR, this LFSR uses conventional RS latches which are not secure because of asymmetry, that is, changes only happen when two successive data values are different. This dual-rail coding circuit is likely only to benefit from its asynchronous operation compared to the convention dual-rail clocked LFSR.

5.5.3 Secure dual-rail asynchronous LFSR

The Null Convention Logic (NCL) technology [45] inherently and conveniently expresses asynchronous digital circuits which are delay-insensitive. For an NCL circuit to express a one-bit logic value two wires are required. This is identical to the dual-rail encoding traditionally used as a return-to-zero protocol between speed-independent circuits. Similar to return-to-zero dual-rail encoding circuits, a Null stage must be assigned after each Data stage, and at least one spacer or bubble stage is needed to move data from one stage to the next stage in the register. The register cycles between the Null state and the Data state. If it determines that the circuit is in the Null state, 00, then it indicates a readiness for a new set of data. Otherwise, it indicates the reception of the Data value. The passage of data or Null through a register occurs strictly after the request for the Data or Null values.

Figure 5.15 shows a three-stage NCL shift register with a feedback signal which determines the complete Data and Null state. The gates with label 2 are 2/2 threshold gates

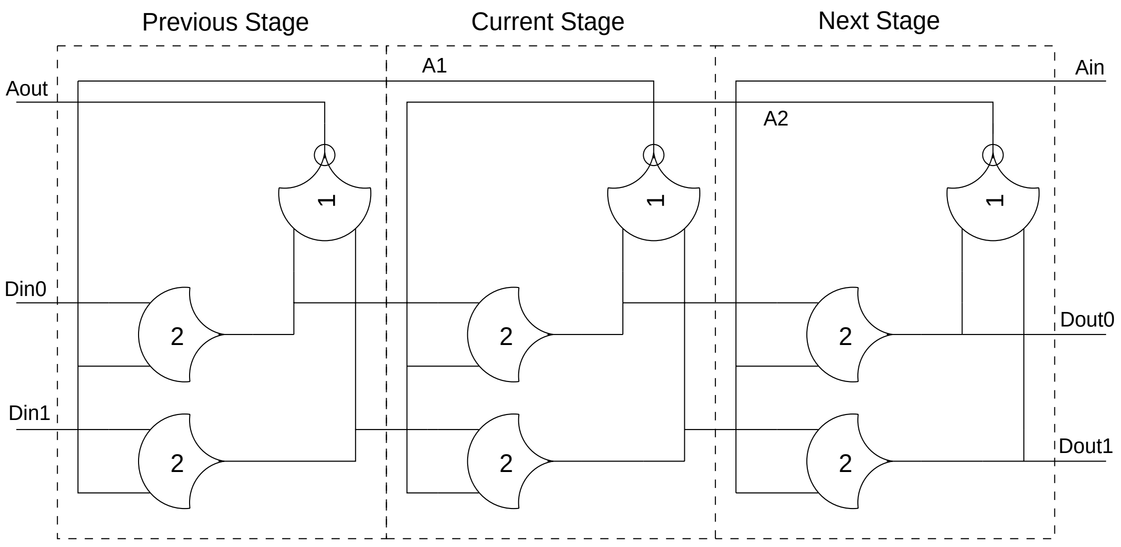


Figure 5.15 Three-stage NCL shift register

(C-elements) and with label 1 are 1/2 threshold gates (OR gates). The paired 2/2 threshold gates are the gates which assert Data or Null when they determine a complete set of Data or Null inputs. When a rising transition is detected on one of the two data wires they will assert Data, indicating that a set of data has been received and stored. They remain in the Data state until all inputs are Null. In the figure, three stages are presented: the Previous stage, the Current stage and the Next stage. If the Current stage and the Next stage are initialised to Null then their 1/2 threshold gates are set to accept new data. When the Previous stage presents a full set of data to the Current stage, the data set will propagate through the Current stage to the Next stage. When the data is detected by the Current stage, it resets its acknowledge signal, A_1, to indicate the reception of the data. At some point, a Null state arrives at the Current stage but it cannot propagate until the acknowledge signal, A_2, is reset to Null, which means that the data has been received and stored by the Next stage. The Next stage detects the complete set of data and return its acknowledge signal, A_2, to Null to indicate the arrival of the data set and to allow the Null pass through to the Next stage.

From the design, it is clear that one and only one of the two 2/2 threshold gates and the 1/2 threshold gate is active regardless of the data value presented. Furthermore, the active 2/2 threshold gate is returned to Null before it can receive another valid data value. Therefore, the active circuit (or transition) required to move different data in the LFSR will be symmetrical, resulting in balanced power consumption. Moreover, Null Convention Logic is inherently delay-insensitive. With careful design to guarantee the symmetry of the circuits, NCL circuits are expected to reduce the differences in power consumption and consequently resist differential power analysis attacks.

5.6 Summary of chapter

This chapter reviewed the basic theory of the Linear Feedback Shift Register (LFSR) and its use in cryptographic applications. Six LFSRs have been designed using different design technologies and different data-encoding techniques in order to investigate the different security impact of those technologies. They include a single-rail clocked LFSR, a conventional dual-rail clocked LFSR, a secure dual-rail clocked LFSR, a single-rail (bundled-data) asynchronous LFSR, a conventional dual-rail asynchronous LFSR and a secure dual-rail asynchronous LFSR.

The conventional dual-rail latch only consumes power when the data presenting to the latch is different from the data which has already stored in the latch, leading to the problem of leaking Hamming weight distance between two successive data whenever the latch is updated. Therefore a special secure dual-rail latch is designed using two cross-coupled RS latches two ensure the content of the latch is cleared before a new data is stored.

The six LFSRs were implemented with the objective of understanding the fundamental differences in the characteristics of power consumption between synchronous and asynchronous circuits, and to evaluate the impact on security of those technologies. The evaluation of these LFSRs is carried out in the next chapter.

Chapter 6: Differential Power Analysis of the LFSRs

This chapter applies the asynchronous power analysis methodology introduced in chapter 4 to the LFSRs described in chapter 5, with the aims of evaluating whether the asynchronous LFSRs are better than their synchronous counterparts in terms of security against power analysis attacks, and evaluating whether balanced dual-rail design actually reduces information leakage and improves security.

The analysis and evaluation of the LFSRs are based on power waveforms extracted from power simulations. The characteristic power waveforms for each LFSR are first measured to provide an insight into the power characteristics of asynchronous and dual-rail circuits and therefore to understand better how security can be improved. In addition, the chapter looks into sub-circuits to offer a deep analysis of the power consumption characteristics of synchronous and asynchronous LFSRs. After the power waveforms are measured and reviewed qualitatively, the waveforms are then statistically analyzed following the methodology discussed in the chapter 4. The results of the analysis show that the secure dual-rail asynchronous LFSR significantly improves resistance to the differential power analysis attack.

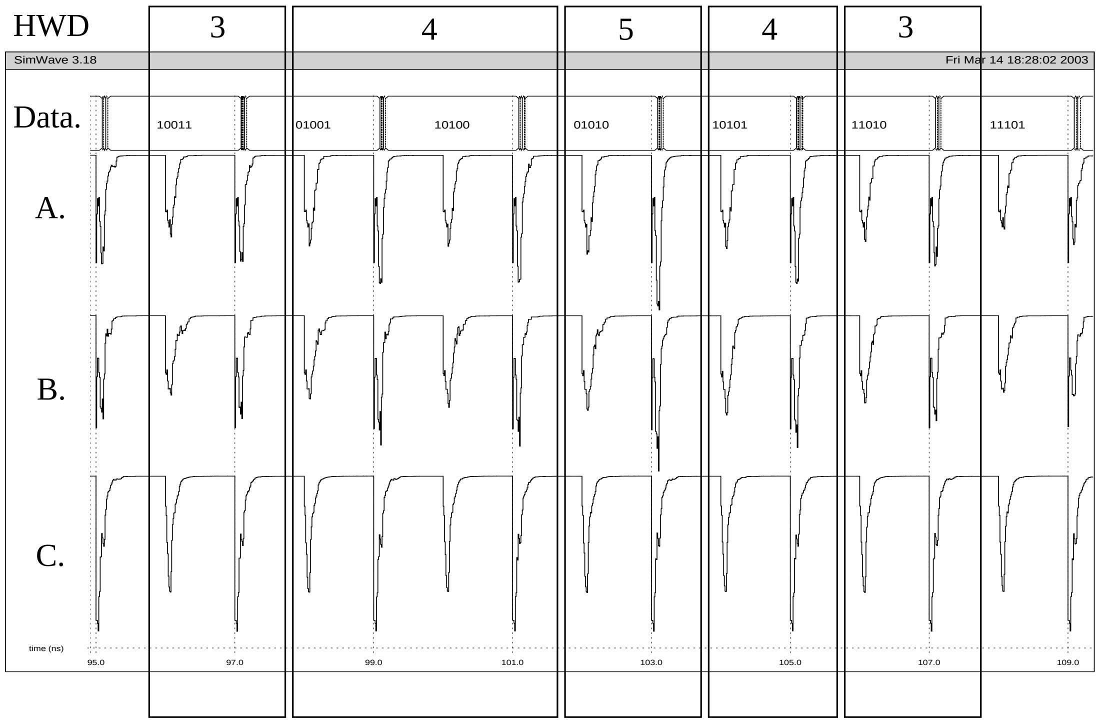
The power waveforms were obtained from simulations carried out with Nanosim using 0.18 ST CMOS technology. The power waveforms are recorded with a time resolution of 10 ps and a current resolution of 1uA.

6.1 Clocked LFSR power waveforms

Figure 6.1 shows the clocked LFSRs’ power traces. The binary logic value shown on the bus, at the top of the trace, indicates the data being processed in the LFSRs. The traces, A, B, and C, are power waveforms for the single-rail clocked, conventional dual-rail clocked, and secure dual-rail clocked LFSR respectively. The clock period in simulation was set to 2 ns for all clocked power simulations. The partitioned fragments in the square boxes labelled with 5, 4 and 3 correspond to Hamming weight distances (HWD) of 5, 4 and 3 between two successive data values respectively. It is expected that power consumption has positive correlation with the HWD.

6.1.1 Single-rail clocked LFSR power waveform

From the figure, the three marked groups on the single-rail clocked power waveform can easily be distinguished by observation. The power consumption peak for HWD=5 is clearly larger than the other two groups, whilst the power consumption peaks for HWD=4 take second place and the peaks for HWD=3 are the smallest. The larger the HWD of two successive data values, the larger the peak. In other words, there is a correlation between power consumption and Hamming weight distance. This shows the weakness of the single-rail clocked LFSR which leaks information about the Hamming weight distance of



A. single-rail B. conventional dual-rail C. secure dual-rail

HWD: Hamming weight distance Data: binary value processed

Figure 6.1 Power waveforms of clocked LFSRs

the data being processed through its power consumption. Attackers who can access such leaked information may exploit it to extract secret information processed in cryptosystems that use this single-rail clocked design style.

6.1.2 Conventional dual-rail clocked LFSR power waveform

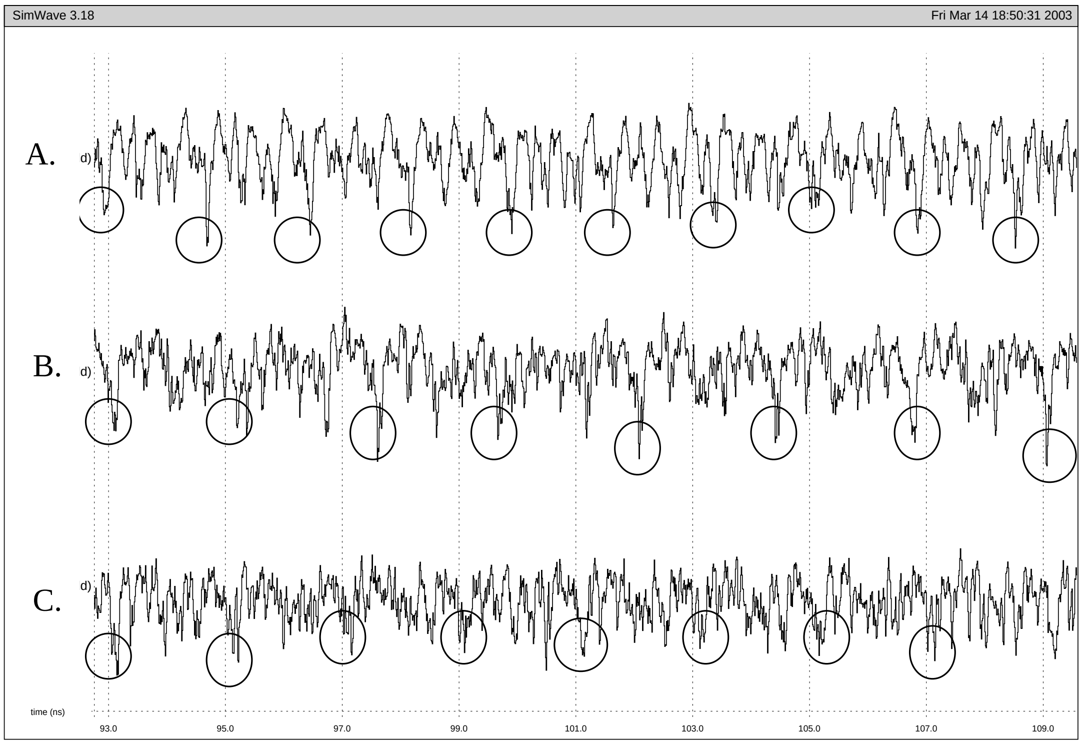
From Figure 6.1, it can be seen that the power consumption of the conventional dual-rail clocked LFSR (trace B) is also related to the Hamming weight distance. In fact, it is almost identical to the single-rail clocked design. The expected benefits of balanced dual-rail encoding are not apparent. Upon closer inspection of the implemented circuits, it is understandable why they display such similarity: although the conventional dual-rail clocked LFSR uses dual-rail encoding to balance power consumption involving data processing, the conventional RS latches used for data storage are not balanced. In an LFSR circuit there is very little data processing, thus storage power dominates. Therefore as the particular latches only consume power when the arriving data is different from the data previously kept in the latches, the Hamming weight distance is revealed in the power waveforms. Consequently, the conventional dual-rail clocked LFSR is still vulnerable to differential power analysis attacks.

6.1.3 Secure dual-rail LFSR power waveform

In Figure 6.1, there is no obvious evidence that the power waveform of the secure dual-rail clocked LFSR (trace C) discloses information about the Hamming weight distances through differences in power consumption. The partitioned power waveform groups for data with different Hamming weight distances consume almost the same power and are identical to each other. This indicates at least qualitatively that the secure dual-rail LFSR improves resistance to simple power analysis significantly.

6.2 Asynchronous LFSR power waveforms

Having analyzed clocked LFSRs, it is clear that secure dual-rail encoding balances the power consumption and provides a significant improvement in defeating power analysis attacks. This section presents the power consumption characteristics of asynchronous LFSRs. Measuring asynchronous power consumption is not as simple as measuring the



A. single-rail B. conventional dual-rail C. secure dual-rail

Figure 6.2 Power waveforms of asynchronous LFSRs

power consumption of synchronous circuits. Because of their asynchronous behaviour, self-timed circuits naturally provide a way to mask differences in power consumption due to the absence of the global clock signal. Even though handshake control signals have a cyclic nature, the power consumption peaks do not appear as obviously periodic as those in synchronous waveforms. In fact, the independent behaviour of each individual sub-circuit distributes the power consumption over a period of time rather than concentrating it at a certain time. This creates difficulties for asynchronous power analysis. Therefore, it is necessary to evaluate what improvements it offers and how many difficulties it imposes on power analysis. The following section addresses these questions.

Figure 6.2 illustrates the power consumption waveforms for single-rail, conventional dual-rail and secure dual-rail asynchronous LFSRs. As for the clocked counterparts, they are from power simulations processing the same data values. These waveforms are all the original waveforms without any analysis. The waveforms were circled according to their handshake signal. To make a successful power analysis attack, the expectation of positive correlation between power consumption and Hamming weight distance is required.

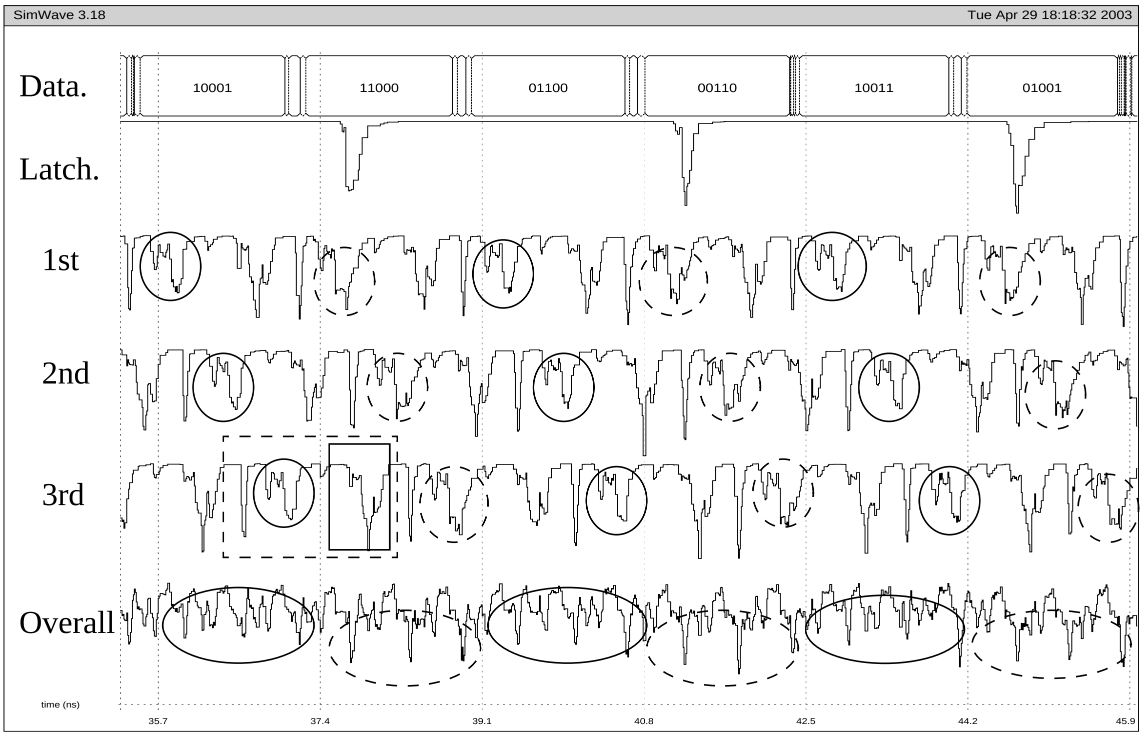
6.2.1 Single-rail asynchronous LFSR waveform

Carefully scrutinizing trace A in Figure 6.2 reveals power consumption peaks, as circled, which divide the trace into cyclic segments with some characteristic periodicity. This is because of the cyclic nature of the 4-phase non-decoupled handshake protocol control signals. However, as it is not necessary for handshake signals to arrive within a fixed interval, the periods of the cycles vary slightly from cycle to cycle. In the clocked counterpart, the power waveform shows that the power consumption peaks are proportional to the Hamming weight distance between two successive data values, and it is anticipated that the single-rail asynchronous waveform would display similar behaviour. Nevertheless, although those power peaks show differences from cycle to cycle, they do not always correlate with the Hamming weight distances. This phenomenon indicates that the anticipated difference is hidden.

Figure 6.3 shows the detail of the power consumption for one bit (the third bit of the five bit LFSR) of the single-rail asynchronous LFSR in order to understand the reasons for this behaviour. The middle bit of the 5-bit binary values at the top of the figure show the data value being processed. The power consumption for one RS latch (marked ‘Latch’) clearly shows that power is only consumed when the data value (the middle bit in this case) is changed.

Recall that, in these self-timed LFSRs, there are three stages in every bit of the LFSR: the data, the null and the spacer stage. The middle three traces illustrate the power consumption for these three stages respectively. On the ‘1st’ trace, the areas marked with a circle are where latches dissipate power. By careful measurement, it can be seen that the power consumption in the dashed circles, where different data was latched, is larger than in the solid circles where the same data was presented to the RS latch. It is quite clear that the three traces display similar characteristics, and are shifted by a small amount of time resulting from passing the latch control token between them.

By summing the power dissipation for those three stages, the bottom trace shows the overall power consumption for this bit. Visual analysis of this composite waveform is much harder than on the individual components. The dashed and solid circles show where data does and does not change respectively. Furthermore, there is also a problem of



Data: binary value processed Latch: power dissipation for one RS latch

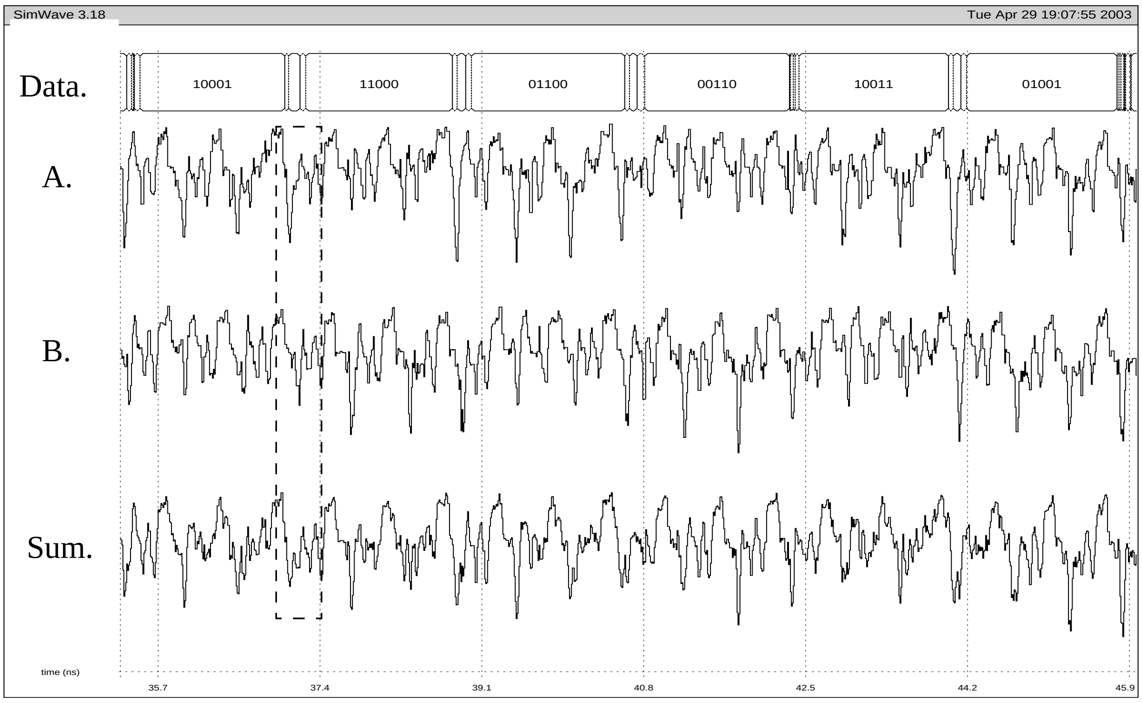
1st, 2nd and 3rd: power for the data, the null and the spacer stage respectively

Overall: power for all three stages

Figure 6.3 Power analysis of the single-rail asynchronous LFSR: One bit power consumption

overlap. The dashed rectangular-highlighted part of the power consumption on the ‘3rd’ trace spans two successive cycles. Part of the rectangle, shown as small solid rectangle, interferes with the following cycle.

Figure 6.4 shows the results of a further investigation of the single-rail asynchronous LFSR power consumption for two bits, the second bit and the third bit (the A trace and the B trace respectively). The sum of these two individual waveforms is also shown on the ‘Sum’ trace. Clearly, it is even more difficult to extract the differences from this ‘Sum’ power trace. On this trace, another difficulty, power waveform shifting, is highlighted by the dashed rectangle. Because of shifting, peaks for each individual bit do not entirely overlap to form a large peak on the ‘Sum’ trace. Such power waveform shifting noise is introduced because of the different arrival time of the individual handshake signals. This inherent shift problems in self-timed circuits cannot be corrected because sub-circuits are not accessible to attackers.



A: power for the 2nd bit B: power for the 3rd bit Sum: the sum of A and B

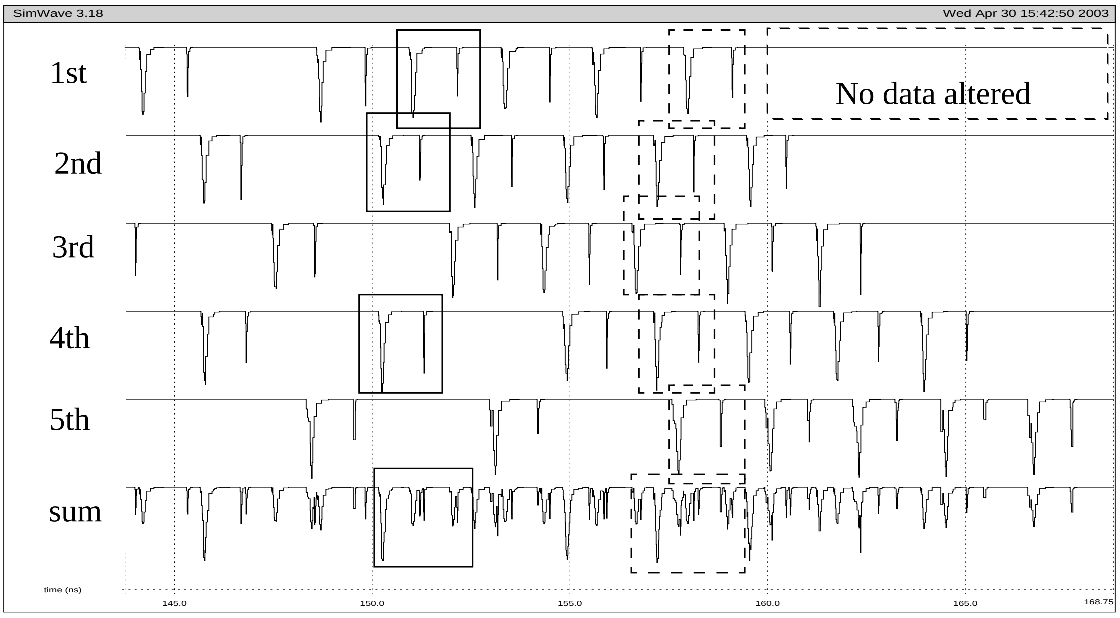
Figure 6.4 Power analysis of the single-rail asynchronous LFSR: Two bits power consumption

6.2.2 Conventional dual-rail asynchronous LFSR waveform

The power waveform of the conventional dual-rail asynchronous LFSR, shown as trace ‘B’ in Figure 6.2, can be identified as a cyclic waveform and is similar to the single-rail one. Waveform shift also occurs in this LFSR design.

Figure 6.5 illustrates the power consumption for five RS latches, one from each bit of this dual-rail LFSR. It clearly shows the difference on each individual trace. For example, the dashed rectangle with “No data altered” mark on the 1st trace is where data remains the same showing that no power was dissipated. The difference in power consumption on each individual trace, however, is not reflected in the ‘sum’ trace, the overall power consumption. For example, there are two marked groups: one group (the dashed rectangle) where the data values in all five latches change, and the other (the solid rectangle) where only three of them change. Looking at the ‘sum’ trace, there are no large peaks to reflect the differences between these two groups. In fact, as illustrated, there is a difference shown on the trace: the area in the solid square on the ‘sum’ trace has less energy consumed (the area of the portion) than the dash-marked area. Indeed, those differences are shown on the ‘sum’ trace with slight time shifting. Such evidence indicates that, although the difference in data activity is not shown by the current peaks, it may be revealed by measuring the energy consumption. Thus, the design of the LFSR may still leak information through its energy use. The energy consumption analysis is carried out in Section 6.4.2.

From the above, it is apparent that, because of the natural cyclic handshake protocol, the power consumption traces for self-timed circuits display some characteristics of periodicity. In addition, because of the non-simultaneous arrival of handshake controls, the differences in power consumption do not align perfectly and their contributions to the overall power consumption appear shifted. Thus, differential power analysis is made harder. However, differences still exist and it is possible to identify the differences by closely measuring the overall energy consumption.



1st, 2nd, 3rd, 4th and 5th: power consumption for five RS latches respectively

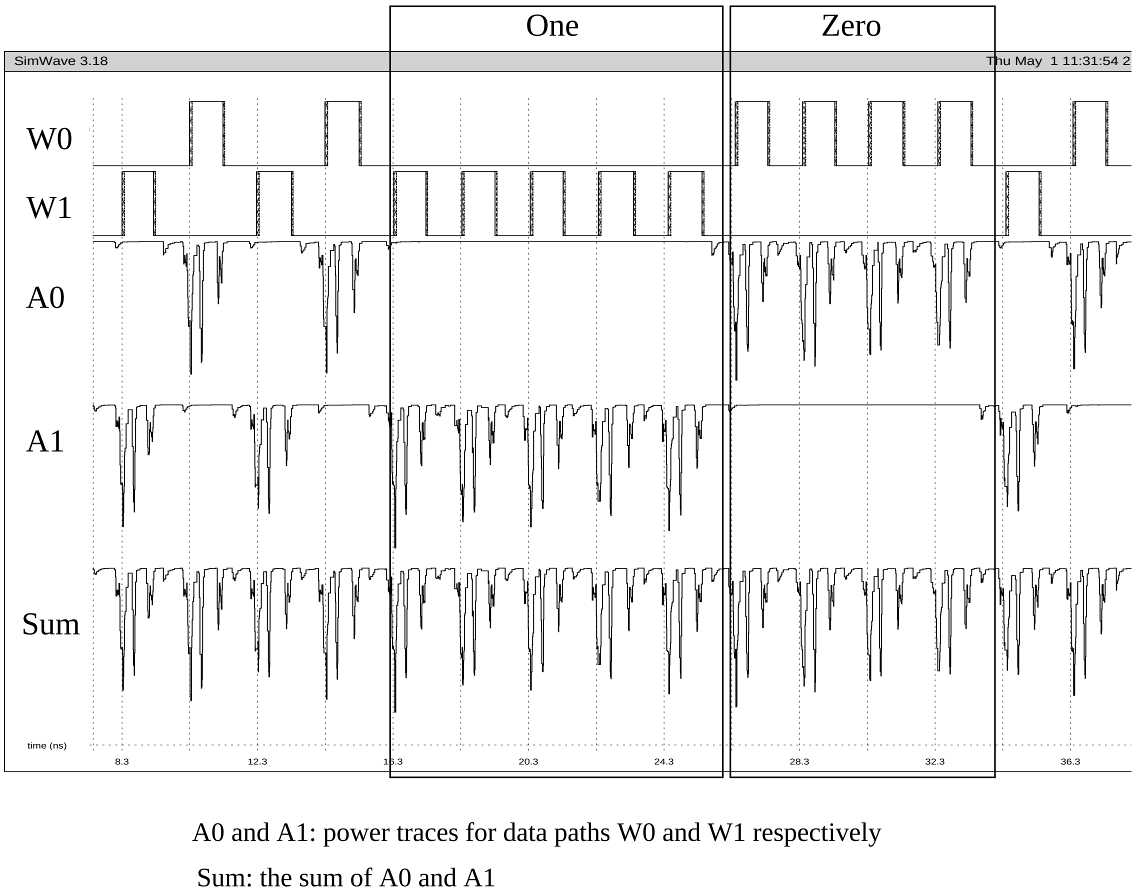
sum: the overall power consumption of the five latches

Figure 6.5 Conventional dual-rail asynchronous power analysis: RS latch power consumption

6.2.3 Secure dual-rail asynchronous LFSR waveform

The last trace (trace C) in Figure 6.2 shows the power waveform for the secure dual-rail asynchronous LFSR. A cyclic characteristic can also be found on this dual-rail power waveform as highlighted and differences can be found between cycles. However, those differences do not correlate with the data being processed. Compared with the above two designs, this power waveform displays much more randomness. The resulting waveform indicates a significant improvement, masking differences in power consumption.

The above analyses of the clocked, single-rail and dual-rail LFSRs indicates that the differences in power consumption can be reduced through balancing circuits, and that removing the clock is an effective way to make the attacker's job more difficult. However, this also makes it harder to evaluate, especially the secure dual-rail LFSR where the differences are extremely small.



A0 and A1: power traces for data paths W0 and W1 respectively

Sum: the sum of A0 and A1

Figure 6.6 Power analysis of the secure dual-rail asynchronous LFSR: One stage power waveform

To determine if there is a difference in power consumption, the power consumption analysis of one stage of the secure dual-rail asynchronous LFSR is shown in Figure 6.6. The top two logic signals indicate the logic value changing on the data0 and data1 wires, W0 and W1 (see Figure 5.10). Traces A0 and A1 are the power traces for the W0 and W1 data path activity respectively. Clearly, when the data value changes on the W0 wire, there is power consumption on the A0 trace and no activity on the other trace, and vice versa. By observing these two traces the data being changed is revealed. However, the power consumptions on the W0 and W1 paths are almost identical to each other. Their overall power consumption, shown as the ‘Sum’ trace, displays a very regular power consumption characteristic, both in terms of power consumption peaks and in terms of energy. Thus, it is confirmed that the power consumption is further balanced by this implementation of the LFSR.

6.3 Comparison of clocked and self-timed waveforms

Having observed power consumption waveforms for clocked and self-timed circuits and sub-circuits, the different power dissipation characteristics between clocked and self-timed circuits become clear and are summarized in this section.

- Time reference: the global clock signal in clocked circuits provides a time reference to the attacker by synchronizing the logic behaviour of the circuit, allowing easy sampling and partitioning of the power waveform. Asynchronous design, by discarding the global clock such that the individual parts of the circuit operate independently and therefore consume power independently, removes this timing reference. This gives the waveform a peusdo-random style with no time reference. Thus, inherently self-timed circuits have a potential advantage against power analysis attacks because of this lack of a timing reference.

- Periodicity: because of the clock, a synchronous waveform is a periodic waveform. Events start from the clock trigger edge and have to be completed before the next edge. Asynchronous logic does display some characteristics of periodicity because of the cyclic nature of handshake controls. However, such periodicity may vary from cycle to cycle because of small variations in the arrival time of those handshake control signals.

- Fade-out of power consumption peaks: because in synchronous circuits most events are triggered by the global clock signal, large power consumption peaks occur after the clock edges. However, in self-timed circuits, because events are controlled by their local handshake signals, the sub-circuits work independently and consume power independently. Therefore, power is consumed throughout the handshaking period.

- Overlap and shift: in asynchronous circuits, as handshake signals may be effected by their previous stages and even many other stages, so the arrival time of handshake signals may vary randomly. Thus, power consumption may occur at different times, causing the power waveforms to shift and overlap relative to each other.

6.4 Analysis of LFSR power waveforms

Not only did the previous section give details of power consumption waveforms for the six LFSRs designed using different technologies, but it also showed observations of the different power consumption characteristics of these LFSRs. Nevertheless, while these observations help understand the different power consumption characteristics of the LFSRs, and give a qualitative analysis of the power waveforms, they do not provide quantitative analysis data for these power waveforms. This section uses the method introduced in chapter 4 to investigate quantitatively how much improvement the asynchronous design style and the dual-rail signal encoding bring to the resistance to differential power analysis.

The use of a correlation analysis is necessary because an asynchronous power waveform does not display the very large peaks that characterize the synchronous circuit’s power waveform and reveal the data being processed. Instead, many small peaks can be found on the waveform. Subtracting the average makes the difference more relevant, but the waveforms are noisy.

6.4.1 Correlation analysis of LFSR

This quantitative analysis follows the asynchronous power analysis methodology illustrated in Figure 4.3 in page 53. However, because the LFSR analysed here has twenty-one states, the power waveform is classified into *k* sets, *k*=0,..., 20, each set corresponding to one data pattern (one state) and containing power waveform fragments for two hundred cycles (*i*=0,..., 199), as the power simulation runs continuously for two hundred rounds. The partitioned power waveform fragment is denoted as $P_k(i, j)$, where *k*=0,..., 20, *i*=0,..., 199 and *j* is the sample point for each cycle.

Referring to Section 4.2.4 on page 61, typical characteristic waveforms, $TCW_k(j)$, for each type of processed data can be derived by averaging each set of power waveforms over 200 cycles; a common effect waveform, *CEW(j)*, is obtained by averaging all sets of a power waveform. By subtracting this common effect waveform from each typical characteristic waveform, $TCW_k(j)$, typical difference waveforms, $TDW_k(j)$, are obtained. Similarly, individual difference waveforms, $IDW_k(i, j)$ can be obtained.

According to Section 4.2.5 on page 62, a 21*21 correlation coefficient matrix, $R_i(x, y)$, is computed by correlating each $IDW_k(i, j)$ with each $TWD_{k'}(j)$, where $x = 0, \ldots, 20$, $y = 0, \ldots, 20$ and $i = 0, \ldots, 199$. That is each $IDW_k(i, j)$ is correlated with all $TWD_{k'}(j)$, for *k*=0,...,20, $k' = 0, \ldots, 20$ and $i = 0, \ldots, 199$. The correlation coefficient is used to indicate the intensity of correlation. The stronger the two variables correlate, the higher the positive coefficient generated. The expected results are that each individual difference waveform, $IDW_k(i, j)$ has a stronger correlation with the typical difference waveform $TWD_{k'}(j)$ when $k = k'$ than when $k \neq k'$. Consequently, it is expected that the strongest correlations happen on the diagonal of the coefficient matrix, $R_i(k, k)$. If this is not the case, it means that the trace is not correlated with its typical difference waveform. Therefore, it is assumed to lead to a wrong deduction in an attack.

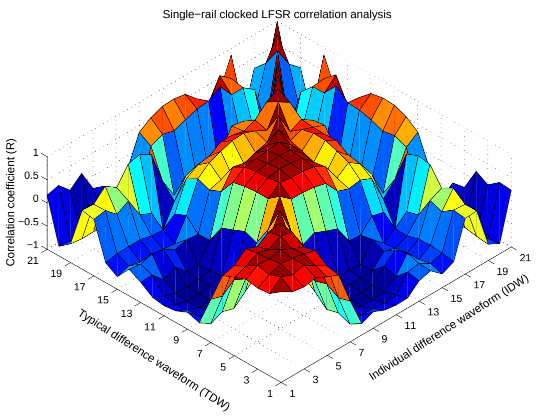
Figure 6.7 shows the correlation analysis of the power waveforms for the six LFSRs. Note that the results shown in these figures only take one of the 200 cycles as an example, in other words, there are 200 similar results. The *x* axis indicates the 21 typical difference waveforms, $TWD_{k'}(j)$ where $k' = 0, \ldots, 20$, *j*=0, 1,...; the *y* axis represents the 21 individual difference waveforms $IDW_k(i, j)$ *k*=0,..., 20 and *j*=0, 1,...; and the *z* axis are the computed correlation coefficients between $IDW_k(i, j)$ and $TWD_{k'}(j)$. As mentioned before, the typical difference waveform and the individual difference waveform are expected to have the strongest correlation if $k = k'$. Therefore, it is anticipated that all the strong correlations will appear on the diagonal from coordinates (0,0) to (20,20).

The three graphs on the left side of Figure 6.7 (a, b and c) present the correlation analysis of the clocked LFSRs and those on right side (d, e and f), those of the asynchronous LFSRs. The two graphs on the top row are for single-rail LFSRs (a, d), the middle row for conventional dual-rail LFSRs (b, e) and the bottom row for the secure LFSRs (c, f).

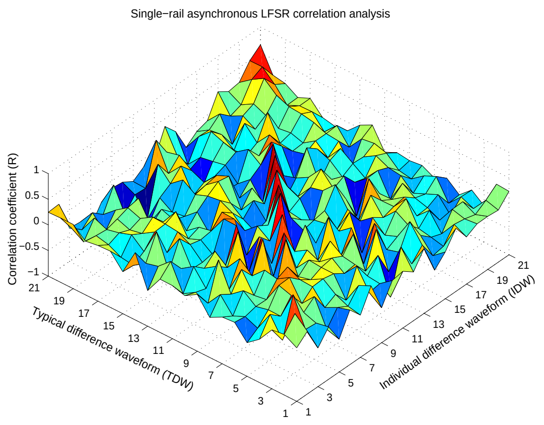
The correlation coefficients for the clocked single-rail and conventional dual-rail LFSRs are similar (Figure 6.7a and Figure 6.7b). Different colours indicate different values of the coefficients. The deep red converging on the diagonal means that most coefficient peaks are centralized on the diagonal, which is what is expected. The figure also illustrates that the closer the Hamming Weight Distance, HWD, the data patterns possess, the stronger the correlation appears. Note that the HWD is calculated between two successive data. For instance, the deeper red of correlation coefficients are generated from a pair of power curves both of which represent data patterns having Hamming weight distances of 3,4,5, and the light red of correlation coefficients are generated from a pair of power curves both of which represent data patterns having with HWDs of 1, 2. These two groups show strong correlations. On the other hand, coefficients in blue colour are the correlations between waveforms having HWDs of 1,2 and waveforms having HWD of 3,4,5. It appears small and has negative coefficients. In short, when the HWDs are both high or low will generate high coefficient, otherwise low coefficient.

In addition, the graph is symmetric across the diagonal since if one typical difference waveform has a strong correlation with another individual difference waveform, the individual difference waveform of the former would also have a strong correlation with the typical difference waveform of the latter. The correlation analysis confirms that the power consumption of these two LFSRs are data-dependent.

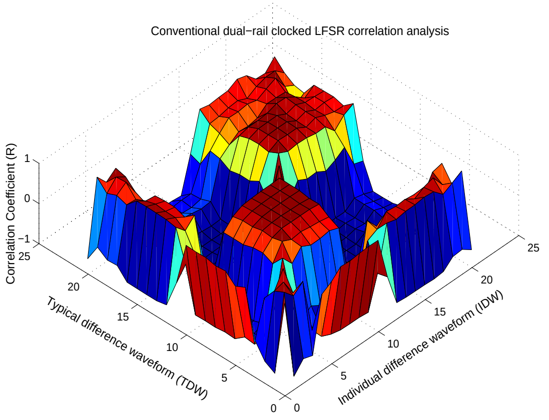
The secure dual-rail clocked LFSR correlation analysis shown in Figure 6.7(c) demonstrates a significant improvement over the above two LFSRs. It is hard to find coefficient peaks on the diagonal. This suggests that the individual difference waveforms do not correspond to their typical difference waveforms. Therefore, the information deduced from power analysis will be incorrect. In addition, the regularity of symmetry of the coefficients is not in evidence in the secure dual-rail clock LFSR. Hence, no correlation between power consumption and Hamming weight distance exists.



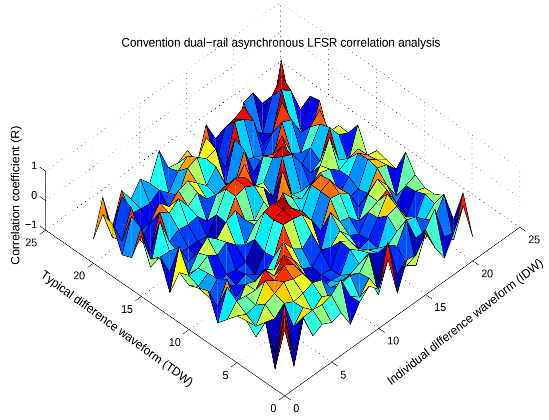
(a)



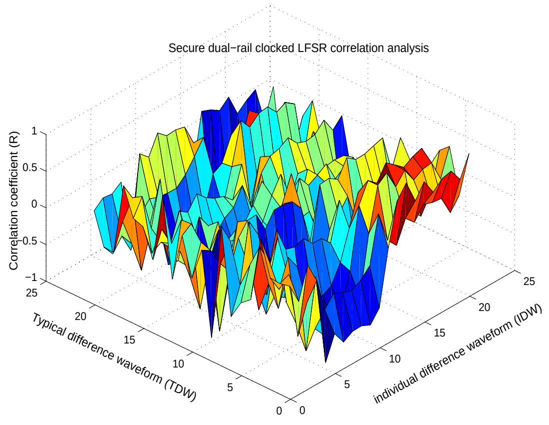
(d)



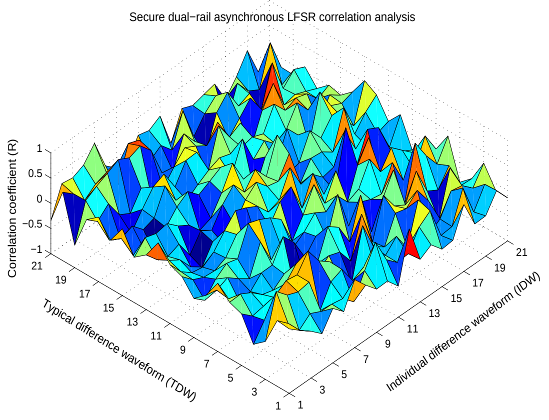
(b)



(e)



(c)



(f)

Figure 6.7 Correlation analysis of power waveforms for LFSRs

In contrast, the two graphs on the top right side of Figure 6.7 (d and e) show the correlation coefficient matrices for the two asynchronous LFSRs. There are still a small number of coefficient peaks along the diagonal, meaning that one typical difference waveform $TDW_{k'}(j)$ and its individual difference waveform $IDW_k(i, j)$ have a strong correlation when $k = k'$. However, the more colourful coefficient matrix indicates an increased variation in the coefficient values, indicating that correlation between different data may be irrelevant even through they have a closer Hamming weight distance. Such irregularity of the correlation coefficients implies an improvement of resistance to power analysis attacks gained by using the asynchronous design style.

Figure 6.7(f) represents the secure dual-rail asynchronous LFSR correlation coefficient matrix. The same observation can be draw from this matrix, that is, coefficient peaks rarely happen on the diagonal. In fact, there are no strong correlations as the coefficients vary between 0.5 and -0.5. The matrix mesh is much flatter than the matrices for the other LFSRs with no obvious coefficient peaks or valleys. Moreover, the differences between the coefficients are much smaller, indicating that the degree of correlation between different curves is very similar, which makes it difficult to determine the correct deduction when carrying out attacks.

From those correlation analyses, it is clear that the asynchronous logic design style and secure dual-rail encoding provide significant improvements in the defence against differential power analysis attacks. To estimate how much side-channel information can be obtained from the single-rail clocked LFSR and how much information the secure dual-rail asynchronous LFSR can hide, the last step of the correlation analysis methodology uses the simple percentage of correct deduction or PCD, defined in Section 4.2.6 on page 63, to quantify the set of deductions yielded by the analysis. For an analysis of a situation as outlined, where there are twenty-one alternative data values, a PCD of 100% would indicate perfect information extraction whereas a PCD of about 5% would indicate that the analysis was extracting no information as the results are essentially random.

Table 6.1 shows the PCD for the correlation analysis of the six LFSRs. The 100% PCD yielded by the single-rail and conventional dual-rail synchronous LFSRs discloses poor resistance to differential power analysis attacks. On the other hand, the single-rail and conventional dual-rail asynchronous LFSRs yield about 40% PCD, confirming the slight improvement. Furthermore, only 18% and 13% of correct deductions can be made with secure dual-rail synchronous and asynchronous LFSRs respectively, showing the significant improvement achieved by employing secure dual-rail encoding with secure storage.

PCD(%)	Synchronous LFSR	Asynchronous LFSR
Single-rail	100	45
Conventional dual-rail	100	41
Secure dual-rail	18	13

Table 6.1: Percentage of correct deduction of LFSR correlation analysis

6.4.2 Differential energy analysis of LFSRs

The PCD results for the correlation analysis indicate that single-rail and conventional dual-rail asynchronous LFSRs have better DPA resistance than their synchronous counterparts. This improvement may be achieved because there are difficulties in analysing asynchronous power waveforms, including the lack of a time reference and waveform shifting. From the detailed measurement of power waveforms discussed previously, it is known that a difference in power consumption actually exists and is present in the waveform, but it is difficult to distinguish. It is also understood that the difference is spread over the waveform rather than being exposed by large peaks. Thus, calculating the energy over a period where one data value is being processed may reveal the difference because the dispersed difference is put together. On the other hand, it must be considered that computing energy might lose the difference since small information-bearing fluctuations would be filtered out.

In general, energy (E) is the product of the voltage, current and time shown in equation (15), where *V* is the power supply voltage, *I* the average current and *t* the amount of time taken to process one data value.

$$E(Joule) = V \times I \times t \qquad (Joule) \tag{15}$$

Alternatively, this energy can be calculated as the integral of current over the required period of time, *t*, multiplied by the power supply, as shown in equation (16).

$$E = V \cdot \int_{t1}^{t2} I dt = V \cdot \sum I_k (T_{k+1} - T_k) \tag{16}$$

Following equation (16), all ‘individual’ energies are computed from partitioned waveform fragments and divided into five sets according to their Hamming weight distance. A typical average energy magnitude is computed for each of the ‘individual’ energy sets. In a similar way to the PCD in correlation analysis, it yields a correct deduction if an ‘individual’ energy is closer to the typical average energy for the set from which it is generated than that for others. Otherwise, it yields a wrong deduction. Note that, in the section on correlation analysis, the power waveform is partitioned into 21 sets according to the different states. However, here the waveform is partitioned into only 5 sets according to the Hamming weight distance, because it is not worth determining the difference between two data patterns having the same number of bits change. For example, data changes from 01111 to 00111 and for 00111 to 00011 will likely yield the same energy result. Each case has a Hamming weight distance of one, but the changed bit happens at the second bit for the former case and at the third bit for the latter case. The difference, which may be detectable on the current waveforms, would be invisible in the energy consumption.

PCD(%)	Synchronous LFSR	Asynchronous LFSR
Single-rail	100	98
Conventional dual-rail	100	67
Secure dual-rail	27	24

Table 6.2: PCD of differential energy analysis of LFSR

Table 6.2 shows the results for the differential energy analysis of the LFSRs. 100% PCD can be extracted from the synchronous LFSRs whilst 98% PCD can still be deduced for the single-rail asynchronous LFSR. However, the PCD for the conventional dual-rail LFSR is reduced to 67% confirming the slight enhancement in energy balance. Compared to the random probability of 20%, 67% means that this LFSR still leaks information about the Hamming weight distance. Thus, difference information masked on the power waveform because of the asynchronous power consumption characteristics is unfortunately leaked by the energy consumption. This indicates that both single-rail and conventional dual-rail asynchronous LFSRs are not as secure as might have been expected.

The differences in average energy between different Hamming weight distances for the secure dual-rail LFSRs were found to be very small, and do not correlate with their Hamming weight distances. Correlation between energy and Hamming weight distance is hardly to be found in secure dual-rail LFSRs. Therefore, both the synchronous and asynchronous secure dual-rail LFSRs are secure against energy analysis.

6.4.3 Signal-to-noise analysis of LFSRs

By applying correlation analysis and differential energy analysis to these power consumption waveforms, the improved resistance to differential power analysis attacks that result from asynchronous design and secure dual-rail encoding becomes apparent. However, one may ask how difficult it is to extract such information from the power waveforms of these LFSRs. Signal-to-noise ratio (SNR) analysis provides the answer indirectly by indicating how much the signal (the difference of the waveforms) can be retrieved from the power waveforms compared to the noise (the average of the waveforms).

SNR(dB)	Synchronous LFSR	Asynchronous LFSR
Single-rail	-11.3	-14.3
Convention dual-rail	-12.4	-16.7
Secure dual-rail	-21.0	-19.8

Table 6.3: Signal-to-noise ratio of the LFSR power waveforms

Table 6.3 shows the signal-to-noise ratio for the power waveforms of the six LFSRs. From the table it can be seen that, for synchronous LFSRs, the SNR for the secure dual-rail LFSR is much lower than that for the single-rail and conventional dual-rail LFSRs, indicating that the ratio of the variance of power consumption relative to the average power consumption for the secure dual-rail LFSR is much smaller than that for the other two LFSRs. This means that the secure dual-rail clocked LFSR is much harder to analyse.

Signal-to-noise analysis of the asynchronous LFSRs shows them to be slightly better than the synchronous LFSRs. However, the SNR of the secure dual-rail asynchronous LFSR is slightly worse than its synchronous counterpart. This could be because the power waveform of the secure dual-rail asynchronous LFSR is extremely difficult to realign, resulting in the variance trace containing noise produced during the power waveform partitioning and stretching. In general, because of the inherent asynchronism and randomness features of the power consumption, partitioning asynchronous power waveforms is much more difficult than synchronous power waveforms. Therefore, the difference trace of an asynchronous power waveform is larger than it should be simply because of noise.

6.5 Analysis results

Six power waveforms have been measured and analyzed to evaluate the security effects of asynchronous design, dual-rail encoding and secure storage, to identify the different power consumption characteristics between synchronous and asynchronous circuits. From this analysis, the following conclusions can be drawn:

- Single-rail and conventional dual-rail asynchronous LFSRs offer a reasonable improvement over their synchronous counterparts. However, the energy analysis indicates these two asynchronous LFSRs still leak information in their usage.

- The analysis confirms that a significant improvement is achieved by using the secure dual-rail design style, which is not only suitable for asynchronous implementations, but can also be beneficial to clocked design. This study indicates the profound importance of using secure storage to balance power consumption.

- Even though the cyclic nature of handshake controls results in power consumption with some characteristics of periodicity, the self-timed behaviour and the lack of a time reference make power analysis of asynchronous waveforms much more difficult than that for synchronous power waveforms, especially when trying to partition the asynchronous waveforms. Thus, it adds an important improvement against power analysis attacks by using self-timed circuits

The power consumption in the secure dual-rail designs is balanced by explicitly returning storage to zero, erasing information carried by the preceding data value. However, because it is difficult to manage the power consumption and other emissions, a physical implementation of a well-balanced design of a robust cryptographic algorithm may still leak more or less information through its side channels for unexpected reasons. For instance, a difference may occur during the layout processes of a physical device, such as a small difference of output capacitance load and different lengths of wires which create diverse wire capacitance. Asynchronous design has an advantages over synchronous design against power analysis attacks since it is much more difficult to analyze the asynchronous power waveforms than the synchronous ones.

6.6 Summary of chapter

This chapter has analyzed power waveforms for LFSRs implemented in a variety of styles to demonstrate their distinguishing power consumption characteristics. The chapter also illustrated how and why the asynchronous power waveforms are difficult to analyze. Furthermore, it confirms the importance of using secure storage in dual-rail encoding design.

This chapter serves as a starting point for the investigation and evaluation of asynchronous power consumption characteristics and the potential improvement of resistance to power analysis attacks. In the following chapters, the investigation and evaluation will look at an asynchronous ARM-compatible processor which is specially designed for security applications such as smartcards.

Chapter 7: Security Investigation on SPA: At Layout Level

It is believed that self-timed logic has advantages for security-sensitive applications. The absence of a clock, as a reliable timing reference, makes conventional power analysis attacks more difficult. However, the variability of the timing of self-timed circuits may be a weakness that could be exploited by alternative attack techniques.

In the last chapter, the power analysis of different LFSRs demonstrated the difference in power consumption characteristics between asynchronous and synchronous circuits and also between single-rail and dual-rail encoding. It showed that the secure dual-rail encoding has an significant impact on the resistance to power analysis attacks and the self-timed technology have extra improvement against power analysis attacks in addition to making the attacks much harder because of the lack of a global clock. In this chapter, differential power analysis is applied to a complex processor: evaluating the security features of a self-timed, ARM-compatible processor, SPA, which is specially designed to explore the benefits of self-timed design in secure applications. In addition to power analysis, timing analysis is also applied to the same design. The analysis in this chapter was based on SPA layout simulation. The results from the investigation are presented and confirm that self-timed logic with dual-rail encoding significantly improves resistance to non-invasive attacks.

7.1 Introduction

SPA [19] is a synthesized self-timed ARM-compatible processor designed specifically for smartcard applications, and designed to investigate the ability of self-timed circuits to resist differential power analysis attacks. The processor was designed using Balsa [39], an asynchronous hardware description language and synthesis system. With a specially-developed secure, dual-rail back-end, the Balsa family of tools was used to produce a gate-level netlist of a secure dual-rail processor, which was then laid out as a semi-custom VLSI device using commercial CAD tools.

Designing circuits to minimize side-channel information leakage and devising attacks to exploit the weaknesses in those designs are complementary activities. This chapter investigates the power signature leaked by SPA layout simulation. The chapter first reviews the security features embedded in the SPA processor. In Section 7.3, the analysis environment is introduced. Power and timing analysis results from SPA simulation are presented in Section 7.4 and Section 7.5 respectively. The differential power analysis of SPA is summarized at the end of the chapter.

7.2 SPA security features

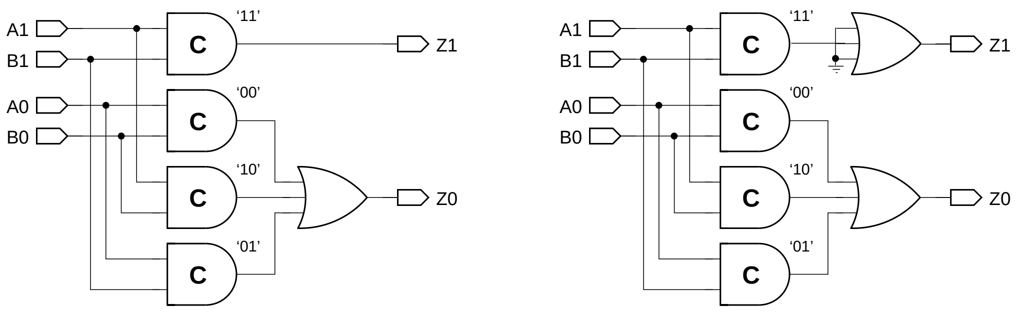
SPA was developed specifically for smartcard applications. It was designed with a specific focus on delivering the ability of self-timed circuits to offer improved security through their resistance to non-invasive attacks, such as differential power analysis and timing analysis attacks. Although the removal of the clock makes statistical analysis more difficult in asynchronous designs, this alone is not sufficient. To achieve such a goal, SPA employs several additional design techniques to minimize side-channel information leakage. The following briefly reviews these main security concerns related to the work in this thesis and more details can be found elsewhere [19].

7.2.1 Security in data communication

As discussed in the previous chapter, the single-rail encoding used in synchronous and bundled-data circuits leaks information since power consumption is proportional to the number of bits that change state when data changes. Although using one wire to deliver each bit of data is area-efficient, it is not secure. Dual-rail encoding is used in SPA to reduce the power consumption signature due to differing data Hamming weights.

7.2.2 Security in data processing

While dual-rail encoded data values have the same Hamming weight, making communication balanced, the balanced communication is not necessarily symmetric



(a) DIMS implementation (b) Balanced DIMS implementation

Figure 7.1 Dual-rail AND gate implementations

because of the possibility of asymmetric logic from which SPA was built. Figure 7.1a shows a dual-rail DIMS (Delay Insensitive Minterm Synthesis [74]) AND gate for dual-rail signalling. Note that the logic for processing data values '0' and '1' are not exactly the same. In order to assert the '0' output (Z0) the three intermediate signals must be ORed whilst the '1' output (Z1) is asserted directly by a single minterm. Therefore, there is a slightly different power consumption and gate delay for the two data values.

This asymmetry problem is solved by placing a dummy load on the chosen signal, as shown in Figure 7.1(b). This pseudo-balanced implementation not only balances the number of signal transitions but also equalizes the gate delay. Such a technique can be employed by any other implementation of dual-rail logic. It is particularly beneficial to implement data-dependent control logic using this pseudo-balanced technique. Interestingly, the DIMS implementations of a number of operations, such as XOR, are naturally symmetric and do not require a dummy load.

7.2.3 Security in data storage

Balancing in processing and communication is achievable using the above techniques. However, it may not be sufficient to ensure security. As discussed in the previous chapter, balancing storage is also crucial. Without such balance the above endeavour to improve security can be futile. Balanced storage can easily be constructed by storing dual-rail coded data. As conventional latches and flip-flops dissipate energy in proportion to the

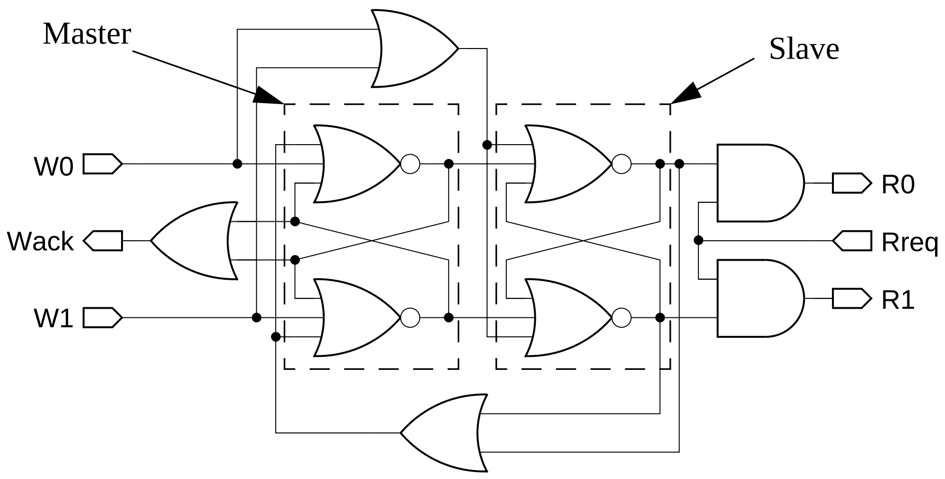


Figure 7.2 Secure dual-rail latch

number of bits changed during a load, to eliminate the difference in Hamming weight between a word loaded into storage and the word previously occupying that place, each storage element is explicitly reset before loading a new data value. The differences between successive values can be hidden, albeit at the cost of a slower and as much as twice as expensive latch than conventional dual-rail latches [15].

The latch shown in Figure 7.2 consists of two stages of cross-coupled NOR gates that operate in a master-slave configuration, similar to the clocked latch shown in Figure 5.10 on page 80. Data is stored in the slave using a dual-rail coding. The master is normally left in its spacer state. New data is stored in the master stage, causing the slave to go into the spacer state, 'erasing' the previous value. After the write operation is acknowledged, data is transferred from the master to the slave and is erased from the master by returning it to the spacer state. In this way, all operations are balanced.

7.2.4 Security in timing

Self-timed circuits are often built to exhibit average-case performance. However, this advantage may leak information. Even using dual-rail encoding, the timing variance may still be exposed to attackers. For instance, different multiplier operands can cause not only different power consumptions, but also different timing because of the different Hamming weights. This timing problem is well-known in the design of cryptographic software for security applications [6]. The cycle time of software loops is the software analogue of the hardware timing problem. Making each loop take the same amount of time to execute is the software solution to the problem. To avoid such a timing problem in hardware, the software solution is reflected in the hardware solution. Functional units in SPA, such as the adder, incrementer, comparators and multiplier, are designed with a constant response time which makes execution time data-independent. This is achieved by making these functions have constant propagation delays. Take the adder as a example. By constructing full-adders which wait for their full complement of inputs to arrive before asserting their outputs, a constant adder propagation delay can be made by forcing the carry to propagate all the way up the carry chain. Each stage of the adder will wait until the previous stage has generated its carry output before responding. However, this solution will make the adder always operate at the worst-case performance. More over this may make execution cycles more fixed giving less ‘randomization’ in the power waveform. This is a dilemma and a price to pay for the security.

7.2.5 Security in behaviour

Whilst the previous sections describe aspects of the design which can be applied automatically during the synthesis process, higher-level asymmetry must be addressed by the designer. This is not as significant an issue as those described above because, at least in a general-purpose microprocessor, the higher-level asymmetry will be due largely to software, which cannot be assumed to be secure.

However, where practical, steps were taken to try to disguise macro-level behaviours. For example, by exercising all parts of the datapath in the execution unit regardless of the nature of the instruction. This requires the components of the datapath to be able to appear to be computing an output while in reality just passing data through. As an example of this, the barrel-shifter, when asked to perform a shift of zero places, instead performs a rotation which returns all the data bits to their original locations.

The applicability of these methods is inherently limited by the need to produce different behaviours for different instructions. While in some cases the efforts have been successful, results are mixed, and some aspects of behaviour have, as yet, not received attention to disguise their nature; for example, whether a conditional branch instruction is taken or not. This therefore requires the software engineer to mask the conditional branch instruction; for instance, always to perform the same task regardless of whether or not the branch is taken.

7.3 SPA security analysis

To evaluate those security properties, the experimental chip designed to improve the resistance of asynchronous designs to non-invasive attacks contains both a single-rail SPA and a secure dual-rail SPA and also contains configurable latches which can switch between secure mode and conventional mode at the chip level.

An optimal experimental environment was set up for the analysis in order to compute the maximum potential side-channel information leakage:

- A simple program that continuously executes the targeted instruction was run in each experiment and only two different data values were used, minimizing the difficulty of the attack. For example, to analyze the ADD instruction the program can execute:

  ```
  ADD r3, r2, r0
  ADD r3, r1, r0
  ```

  where r3 is the destination register, r1 contains 0xff, r2 contains 0xffffffff and r0 contains 0x1. To eliminate interference between the two instructions, several no-op instructions are inserted before, between and after the two ADD instructions. The difference between the two ADD instructions is in the different carry propagation distance. One has 8-bit and the other 32-bit carry propagation. This simple test sequence is used to measure whether there is information-bearing side-channel leakage resulting from the difference in carry propagation. Several hundreds iterations of this test code were run in simulation for differential power analysis.

- The knowledge of which instruction is being executed and which data values are used was made readily available to the attacker, so that an optimal partitioning function can be applied.

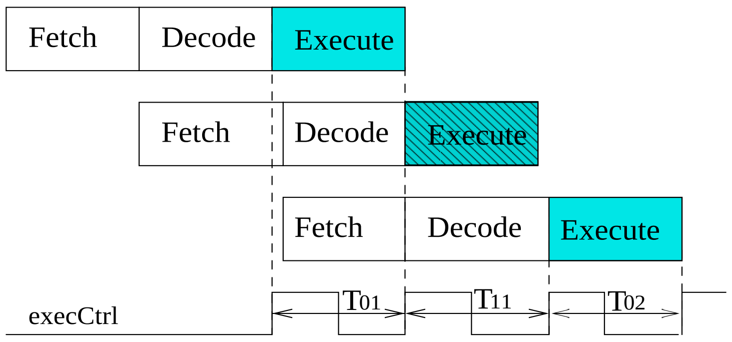


Figure 7.3 The SPA 3-stage pipeline

- A signal (execCtrl) is artificially made available to the attacker to allow optimal synchronization of the power traces. The rising edge of the signal indicates the beginning of the execution stage of an instruction.

Figure 7.3 shows the SPA 3-stage pipeline along with the timing of execCtrl. In a self-timed processor every stage of the pipeline is no longer required to take exactly the same time; it can take different times to execute different instructions. There may be fine-grain timing noise [14], but this can be a source of side-channel information to attackers. The following sections discuss this further.

In a real attack, the environment available to the attacker would obviously not be so helpful, therefore much more effort would be required to undertake such an analysis. The absence of the timing reference signal would particularly complicate the attacker's task. Consequently the results obtained here can be considered pessimistic in terms of what they say about the intrinsic security of the SPA system. Nevertheless, they provide a baseline measure of the relative security of the system.

7.4 Power analysis of SPA

In this section the asynchronous power analysis methodology introduced in Chapter 4 is used to evaluate the effectiveness of the measures taken in the design of SPA to protect it against attacks.

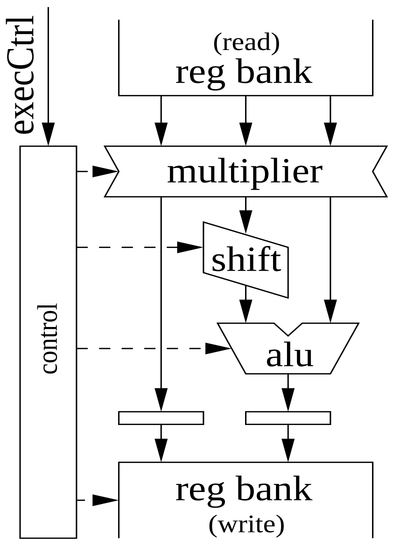
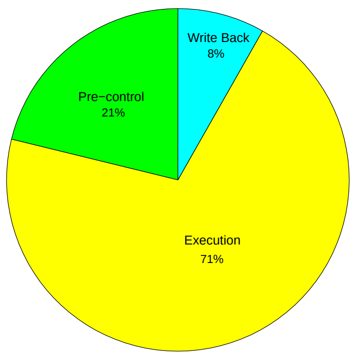


Figure 7.4 The SPA execution stage

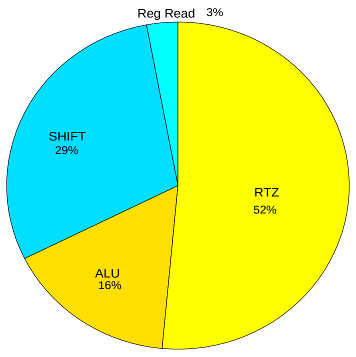
The analysis of SPA is based upon power waveforms taken from layout simulation. The traces were partitioned using the execCtrl signal introduced earlier. In order to understand these waveforms, the SPA execution unit is described in detail. It is illustrated in Figure 7.4 and includes three stages: pre-control, execution and write-back.

The pre-control stage determines whether an instruction is executed. Some instructions, for example conditional branches, might not need to be executed. Only if an instruction is executed is the execCtrl signal triggered, otherwise it remains inactive.

The second stage, the execution stage, can be broken down into sub-stages: register read, multiply, shift, ALU and return-to-zero. As it is designed for ARM compatibility, the register bank in SPA consists of 16 32-bit registers. The multiplier is transparent if the instruction does not need a multiplication operation. The shifter in the SPA execution unit contains five shift stages with different shift distances: 1, 2, 4, 8 and 16 bits. The ALU consists of XOR, AND, ADD and OR basic functions. To maximise security an instruction passes through the ALU even if it does not require any ALU operation - not to do so would reveal information in the power and timing signatures. The last stage, write-back, writes results back into the register bank.



(a)



(b)

Figure 7.5 SPA execution time breakdown

For further partitioning of the power waveform, timing information for each of the stages is presented in Figure 7.5. As shown in Figure 7.5(a), on average about 21% of the time is used in the pre-control stage and about 8% of the time is for register write-back. The majority of the time, about 71%, is used during the execution stage of the instruction. Figure 7.5(b) shows a further breakdown of the execution stage timing. About 3% of the execution time is spent in register read, 29% in the shift operation and 16% in the ALU, while over half of the time is used for the execution unit to return to zero. The multiply operation is not included in this case and the execution unit takes a longer overall time if a multiplication is needed.

The power analysis of SPA considers six instructions: a shift with varying distance (distance-related shift), a shift with varying data (data-related shift), a branch, an xor, an add and a multiply. These were selected as they are the most likely to generate power consumption differences. The data-related shift instruction is explained in detail below while the results from the other instructions follow.

The data-related shift instructions use the same shift distance of 16 bits, but shift different data values, 0x0 and 0xffff. We observe the differential power consumption caused by processing these two data values. To isolate the effects of the previous and following instructions, three no-op instructions are inserted between the two shift instructions.

Figure 7.6 shows power consumption traces of SPA executing these data-related shift instructions. The upper window shows the single-rail SPA and the lower window the dual-rail secure SPA. There are three traces in each window. The dashed power trace is executing a right shift of 0x0000 by 16 bits and the solid trace a right shift of 0xffff by 16 bits. The two curves are the average (mean) waveform of hundreds of simulation loops, one for each data set. The difference in power consumption is shown as the lighter solid trace. The secure SPA waveform is split into sections according to the timing breakdown described previously. Likewise, the single-rail SPA power waveform can also be broken into several sections.

The first part of the waveform is the register read operation which accesses the register operands in the register bank. Clearly, an obvious difference appears on the single-rail SPA waveform. On the other hand, there is almost no evidence of a power difference in the secure SPA, confirming the safety of the dual-rail register operations. This is also true for the register write-back period.

The shift operation on the single-rail SPA generates different power spikes. The positive spikes indicate that shifting 0xffff consumes more power than shifting 0x0000. A small

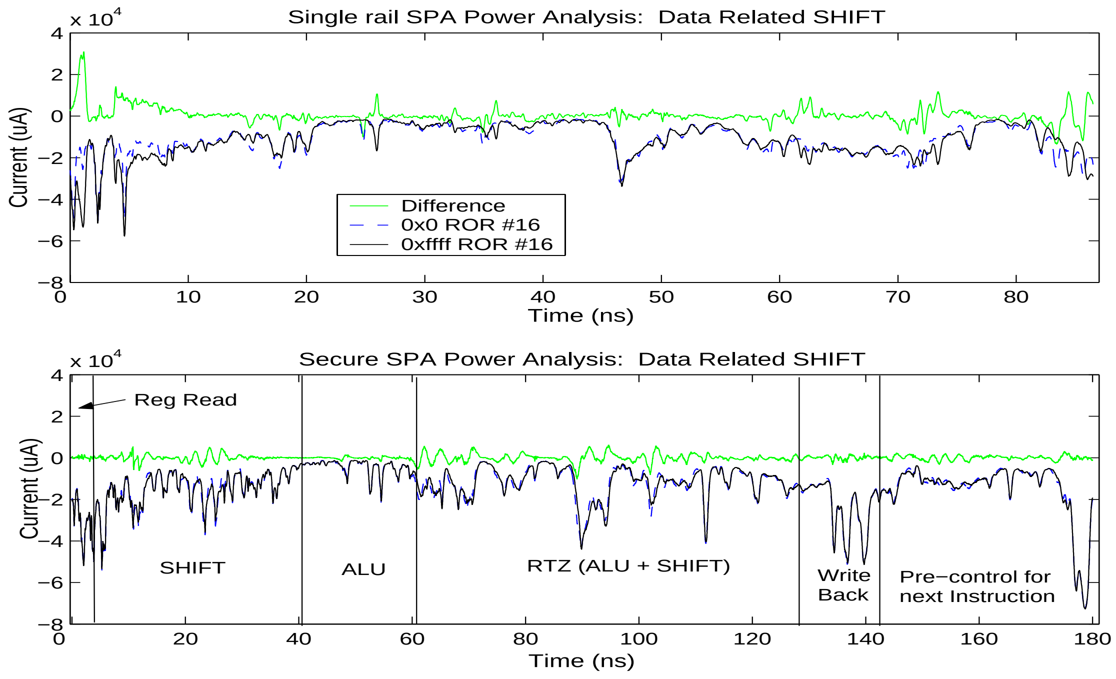


Figure 7.6 Differential power waveform of the data-related SHIFT instruction

power fluctuation is also seen on the secure dual-rail SPA. However, such power consumption differences are likely to be due to power waveform shift problems rather than due to data because the fluctuation is small and oscillates around zero with a peak-valley-peak style. Thus, the security of the shift operation is good on the secure dual-rail SPA whilst the single-rail SPA still leaks information.

During the ALU phase the shift instruction does not require any operation in the ALU. The ALU is still active, however, and there is clearly power consumed during this period. The secure SPA does not exhibit any significant differential power signature, but again there is a very clear spike on the single-rail SPA.

The large region between the ALU and register write-back phases is for the execution unit to return to zero. In this period, the differential power curves exhibit small spikes on both the single-rail and secure SPAs. Although they are small compared to the average waveform they may leak information. Fortunately, during this period the power is less correlated with the data.

The last region on the waveform is the pre-control unit for the next instruction. Because the execCtrl signal is generated after the pre-control stage, the power waveforms

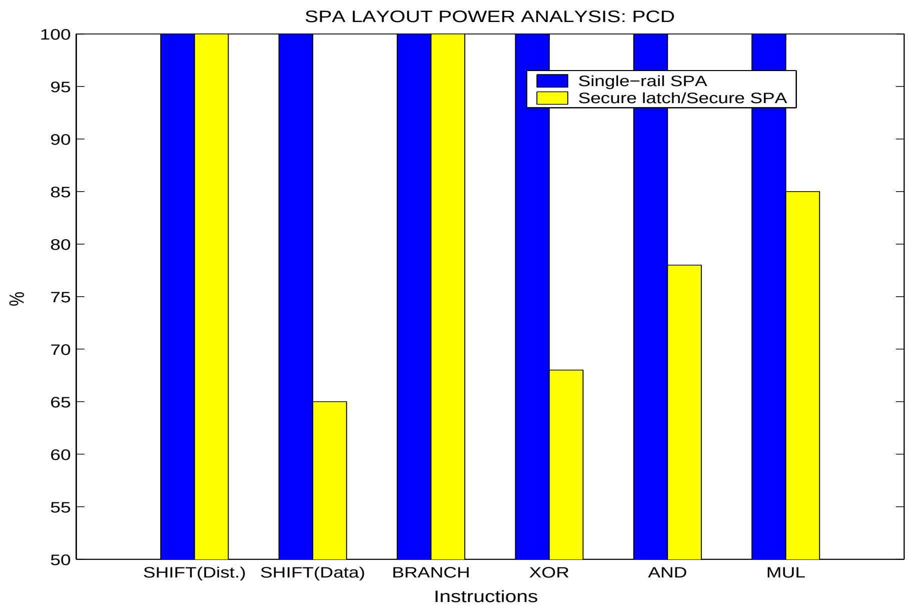


Figure 7.7 SPA power analysis:
percentage of correct deduction (PCD)

partitioned using this signal begin with the execution stage and end with the pre-control stage for the next stage. In this experiment, the next instruction is a no-op which moves data from one register to the same register. Even then a spike is still shown on the single-rail SPA.

Figure 7.7 shows PCD results indicating the extent of information leakage, derived from the application of the power analysis scheme discussed in Chapter 4 to simulated power traces obtained from SPA. The left bar (blue) of each group represents the single-rail SPA. In every experiment, single-rail SPA measurements yield a 100% correct deduction, disclosing poor resistance to differential power analysis. By contrast, the right bar (yellow) represents the secure SPA which produces a much lower PCD in most cases, apart from the BRANCH and distance-related SHIFT instructions which have (nearly) 100% PCD. This means that such instructions are still vulnerable on the secure SPA.

The signal-to-noise ratio of the data-related shift instruction is the group of bars named SHIFT(Data) in Figure 7.8, where the SNR of the single-rail SPA is the blue bar to the left of each pair and that of the secure SPA is the yellow bar to the right. The SNR of the single-rail SPA executing the data-related shift is about -10dB whilst that of the secure SPA is about -18dB. In an environment where external noise was a factor in determining

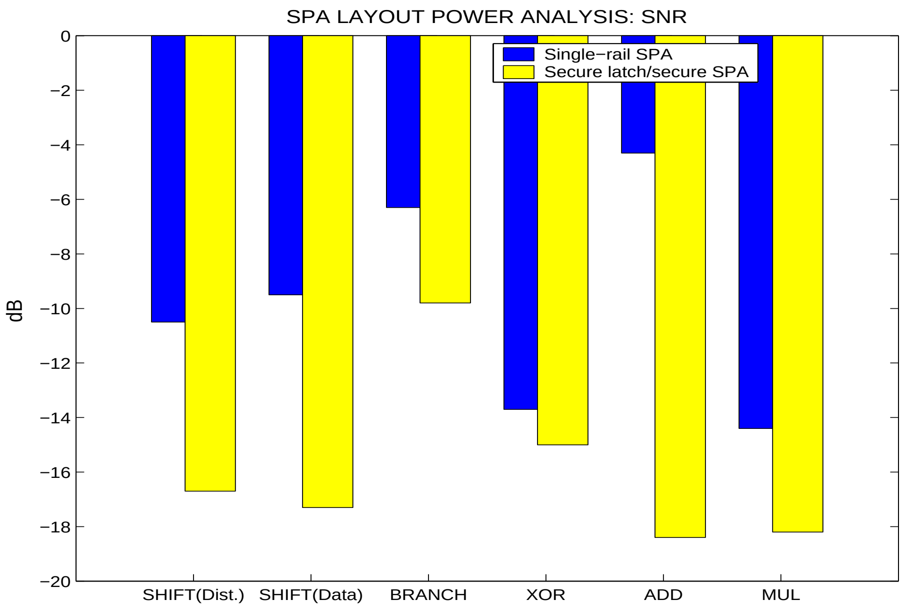


Figure 7.8 SPA power analysis: signal-to-noise ratio

the difficulty of extracting the information, the secure SPA could clearly be much harder to analyze than the single-rail SPA.

The other groups of bars show the SNRs of the remaining instructions being measured. Apart from the branch instruction, where the SNR is greater than -10dB for both the single-rail and the secure SPA, the secure SPA SNRs are around -18dB. In contrast, the SNRs are about -10 dB for the single-rail SPA. This means the power differences in the secure SPA are three times smaller than the power differences in the single-rail SPA.

7.5 Timing analysis of SPA

The theory for timing analysis is similar to that for power analysis. Power analysis uses information leakage though data- or instruction-dependent power consumption [6]. Timing analysis measures data- or instruction-dependent timing differences.

The timing analysis is based upon running one instruction with two different data values alternately. The timing information is obtained using the execCtrl signal, measuring the time from one rising edge to the next (that is, from the beginning of the execution of one instruction to the beginning of the execution the next instruction as shown in Figure 7.3). If the times for the first and second executions are T_{01} and T_{11} respectively, the timing information T_{ij} extracted from the signal can be split into two sets using a partitioning function:

$$T_0 = \{T_{ij}\}, i = 0 \qquad \textbf{(17)}$$

$$T_1 = \{T_{ij}\}, i = 1 \qquad \textbf{(18)}$$

Then, the average time for each set is computed:

$$TA_0 = \frac{1}{|T_0|} \sum_{T_{ij} \in T_0} T_{ij} \qquad \textbf{(19)}$$

$$TA_1 = \frac{1}{|T_1|} \sum_{T_{ij} \in T_1} T_{ij} \qquad \textbf{(20)}$$

Where $|T_0| + |T_1| = N$, the number of iterations. Subtracting the two average times, a timing difference is obtained:

$$\Delta T = |TA_0 - TA_1| \tag{21}$$

For a chosen partitioning function *f(I,D$_i$)* where *I* represents instructs and *D* represents data, corresponding to time consumption T_i, an ideal secure implementation would require the variation, ΔT, to be zero. However, if the time consumption is data-dependent, the chosen partitioning function will yield a non-zero variation. An attacker who can extract this variation will probably be able to extract the data value. The larger the variation, the easier the data is extracted.

The timing analysis results of an ADD instruction executed by both SPAs are shown in Figure 7.9. The left hand chart is the result for the secure SPA and the right hand chart is for the single-rail SPA. The x axis displays the range of timings which each SPA takes to execute the instruction and the y axis indicates the number of cycles which fall into the respective bins. The red bar in each chart is for executing an ADD instruction which causes an 8-bit carry propagation and the green bar a 32-bit carry propagation. From the figure, it is obvious that the two sets of bars are almost fully overlapped for the secure

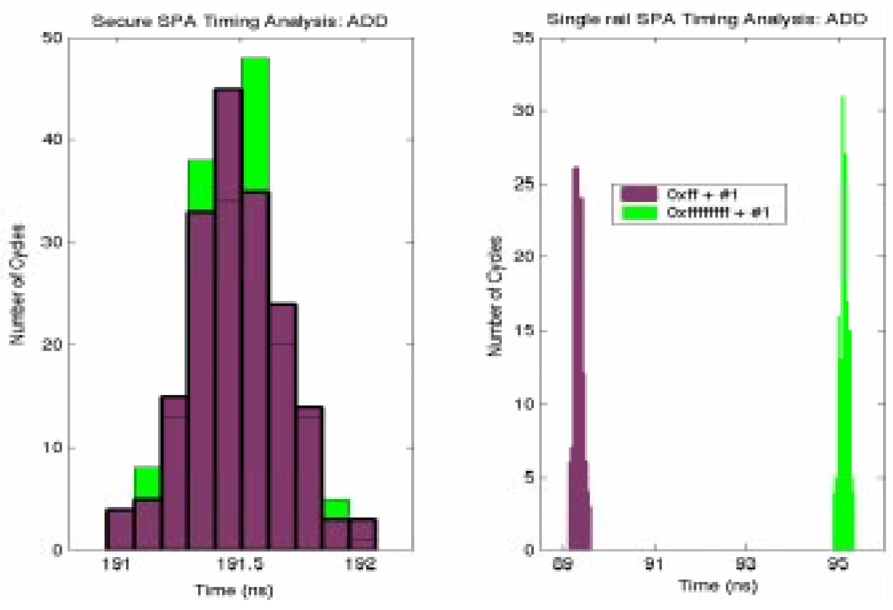


Figure 7.9 SPA ADD instruction timing analysis

SPA. This means that using an ADD instruction to process two different data values consumes almost exactly the same amount of time. For the secure SPA, the average time to execute an ADD instruction for both data causing 8- and 32-bit carry propagation is about 191.5 ns. In contrast, the two sets of bars are totally separated on the other chart for the single-rail SPA. The times fall into two groups, one around 89.3 ns and the other around 95.1 ns. The difference, 6 ns, between the two average times is obvious and is about 6% of the average time. The larger variation clearly indicates that the single-rail SPA leaks information through its timing much more readily than does the secure SPA.

The other conclusion that can be drawn from the figure is that processing the same data with the same instruction does not always take the same amount of time. For instance, the time to execute the ADD instruction with an 8-bit carry propagation on the secure SPA varies from 191ns to 192ns while most of the cycles reside at the centre at about 191.5ns. In synchronous circuits, timing variations are usually the result of noise. However, in asynchronous circuits they result not only from noise but also from the effects of handshake protocols. A handshake stage is affected by its neighbouring stages. Even though the same instruction and data are processed, there is still the possibility to take a slightly different amount of time when the instruction is taken at different stage. This feature is also visible on the power consumption waveform. This confirms that the self-timing behaviour of asynchronous circuits contributes to making attacks more difficult.

Instructions	Average time (ns) (Single-rail SPA)			Average time (ns) (Secure SPA)		
	TA_0	TA_1	ΔT	TA_0	TA_1	ΔT
SHIFT (Dist.)	81.64	82.49	0.85	175.22	180.02	4.80
SHIFT(Data)	86.50	87.66	1.16	180.12	180.11	0.01
BRANCH	48.49	226.68	178.19	92.09	462.63	370.54
XOR	88.11	89.37	1.26	174.11	174.19	0.08
ADD	89.35	95.11	5.76	191.49	191.49	0
MUL	244.93	245.13	0.20	2192.49	2193.38	0.89

Table 7.1: Average instruction execution times

Average times for the observed instructions are shown in Table 7.1. SHIFT (Dist.) is the right rotate shift with different shift distances and the same data, while SHIFT(Data) is the data-related shift instruction with the same shift distance. The BRANCH instruction measures the different timing information when the processor does or does not jump. XOR is executed with one constant operand (0xff) and one of two other values, 0x00 and 0xff, to produce outputs of either 0xff or 0x00. This measures whether the timing information is correlated with the Hamming weight of the result. The ADD instruction measures the relationship between carry propagation and timing as discussed previously. The last instruction measured is MUL, the multiply instruction. The multiply instruction used in SPA always performs 32 iterations regardless of the operand values.

In the single-rail SPA results it is obvious that the timing varies when different instructions are executed. For example, MUL takes three times as long as a shift instruction. For each individual instruction, the time varies when different data is processed. For example, ADD reveals a 6% time variation when processing the two different data values. In the secure SPA results, it is clear that timing differences still exist with some instructions, particularly for SHIFT (Dist.) and BRANCH. For the other instructions the timing variation is almost eliminated. The BRANCH instruction displays very different timing as it is not executed if the branch condition is false. Clearly it will take much longer to execute when it does result in a jump operation. Thus, the BRANCH instruction will always reveal whether its condition is true or false.

Instructions	Standard Deviation of average time (Single-rail SPA)			Standard Deviation (Secure SPA)		
	$STDA_0$	$STDA_1$	STDA	$STDA_0$	$STDA_1$	STDA
SHIFT (Dist.)	0.11	0.11	0.44	0.21	0.16	2.41
SHIFT (Data)	0.11	0.11	0.59	0.18	0.19	0.18
BRANCH	0.11	0.16	89.50	0.20	0.40	186.00
XOR	0.11	0.12	0.64	0.17	0.24	0.21
ADD	0.88	0.96	2.89	0.18	0.19	0.18
MUL	0.15	0.18	0.18	0.56	0.52	0.70

Table 7.2: Standard deviation of average instruction execution times

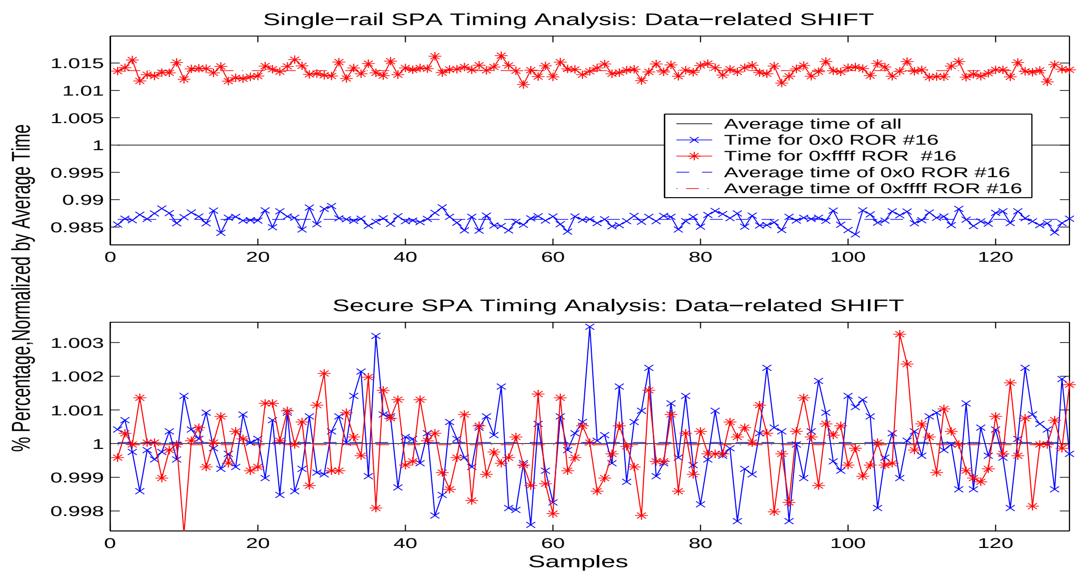


Figure 7.10 SPA timing analysis: Data-related SHIFT

Furthermore, Table 7.2 shows the standard deviation of the average instruction execution time. $STDA_0$ is the standard deviation for TA_0, $STDA_1$ for TA_1 and STDA for the overall instruction average time. From the table we can see that, for the single-rail SPA, the STDAs are much larger than their $STDA_0$ and $STDA_1$, indicating that the execution time for both operation groups has a larger difference. In contrast, apart from SHIFT (Dist.) and BRANCH operations, STDAs are very close to their $STDA_0$ and $STDA_1$ in the secure SPA case, confirming that the execution times for both cases are very close in the secure SPA.

Figure 7.10 illustrates the normalized time for executing data-related instructions. It clearly shows that the execution time in the upper window, for the single-rail SPA, is split into two groups: 0x0 ROR #16 (in the blue colour) and 0xffff ROR #16 (in the red colour). On the other hand, in the lower window, showing the same results for the secure SPA, the execution times for both groups are mixed up and cannot be easily distinguished. This indicates that, whilst the single-rail SPA is still vulnerable to timing attacks, it is more difficult to apply the timing attack since the timing in the secure SPA is largely data-independent.

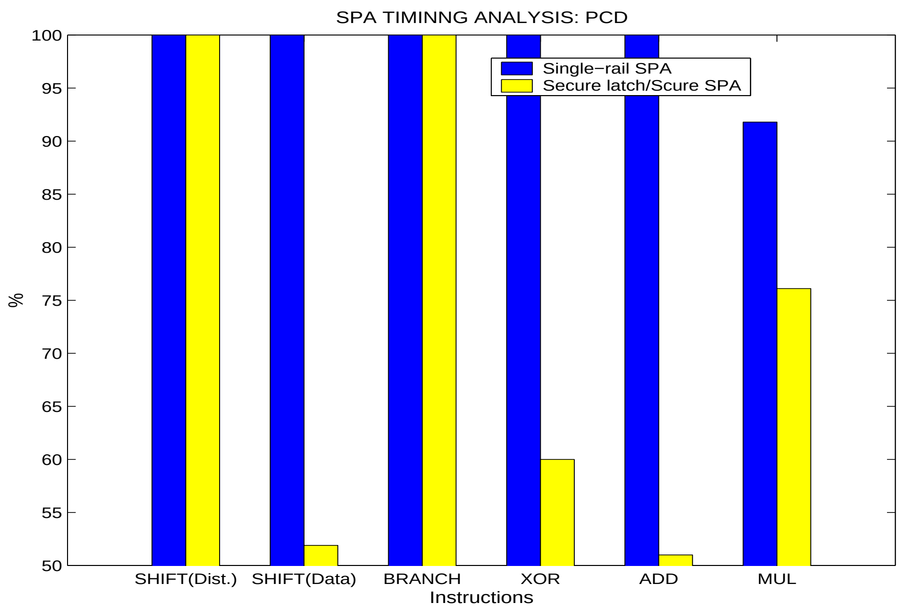


Figure 7.11 SPA timing analysis:
percentage of correct deduction (PCD)

Just as for a power analysis, PCD is used to determine how much information an attacker can extract from the different timing information using a partitioning function. In this experiment, as there are only two possible choices, the worst deduction would be 50% correct, which can be obtained by random choice. For the ADD instruction, the results show that the data can be extracted correctly (100%) by measuring the time difference information when the single-rail SPA is analysed, which confirms its poor resistance to a timing attack. However, the same analysis of the secure SPA yields only 51% correct deduction, barely better than random guessing.

Figure 7.11 shows the timing analysis results. Apart from the MUL instruction, which gives 91.8% correct deduction, the 100% correct data can be extracted from the single-rail SPA. However, 100% correct data can be extracted from the secure SPA only when executing distance-related shifts and branch instructions. The data-related shift, ADD and XOR instructions give data only slightly better than random guesses. The MUL instruction still leaks about 76.1% accurate information.

7.6 Summary of chapter

The chapter reviewed the security features of SPA, an ARM-compatible processor specially designed for security application such as smartcards. Differential power analysis and timing analysis attacks were used to investigate the security improvements incorporated in SPA. The results indicate that the secure dual-rail SPA has significantly improved resistance to non-invasive attacks compared to the single-rail SPA. This confirms that self-timed logic has the potential to improve the resistance of CMOS VLSI devices to such attacks. However, the single-rail SPA security is inadequate and comparable to that of clocked logic.

The removal of the clock creates difficulties for would-be attackers using statistical analysis techniques. These difficulties include the lack of a time reference, misaligned waveforms and random timing noise. Dual-rail encoding and balanced logic with secure latches dramatically reduces the differential power signature, though great care is still required to maintain balance and to avoid revealing information through unintentional timing variations. During layout, for example, great attention must be paid to the balanced routing of dual-rail signals and matching transistors in all inputs to the standard cells used in the design.

From these experiments we have also learnt that some instructions, such as BRANCH and distance-related SHIFT, leak significant information. This knowledge is useful in guiding the programmer of the present SPA and the designer of an enhanced future variant. In a secure application of the current design, conditional instructions should avoid depending on secure information such as a key as they reveal the condition in their power and timing signatures. The processor could be redesigned in the future to ensure that it always does similar work whether or not a condition is passed, so that it has similar timing and power characteristics and leaks less information. For example, a conditional BRANCH instruction could always branch, either to the target or to the next sequential instruction, rather than skipping the branch execution as in the present SPA.

It is clearly possible to design a self-timed processor that is highly resistant to timing and power-analysis attacks. The present secure SPA design demonstrates both strengths and weaknesses in this respect, but it clearly points a way forward for this technology.

Chapter 8: Security Investigation of the G3Card Chip

In Chapter 7, power and timing analysis were used to assess the success of the security features incorporated into the SPA design. The investigation was based on layout simulation. The precision of power information extracted from simulation, however, is limited by the models of transistors that are used in simulation and is often not the same as in the real chip. This chapter explores the security of SPA based on measurement of the *G3Card* silicon chip. The results of investigations on the silicon chip are shown for comparison with those from simulation. Furthermore, because running code on the silicon chip takes much less time than simulation, in addition to analysing individual ARM instructions, an investigation of more sophisticated algorithms is possible on the chip. This chapter, therefore, takes a further step in the analysis of power consumption by processing the Digital Encryption Standard (DES) algorithm on the chip. The results confirm the significant improvement in resistance to differential power analysis attacks suggested by the results based on simulation that were presented in the previous chapter.

8.1 Introduction

G3Card was a collaborative project between academic research groups and industrial partners. It was expected at the commencement that the Manchester group would adapt its existing Amulet self-timed processor subsystem for use in smartcard chips to exploit the advantages of self-timed circuits, due to the absence of a global clock signal, against attempted attacks to compromise security. However it turned out that this solution was not as promising as expected because an analysis of Amulet2e showed that, despite the lack of a global synchronization clock signal, the self-timed processor still had a significant leakage that could be exploited by DPA, in a manner comparable to that of a clocked processor [76].

In consequence, the Manchester group decided to design a new self-timed ARM-compatible processor core specially for security applications such as smartcards using asynchronous design technology and other design techniques, as discussed in the previous chapters, to minimize information leakage. The SPA self-timed processor core was designed and synthesized using Balsa, an asynchronous design and synthesis system[39], and memories and other peripheral devices connected by a CHAIN network [79] were integrated into SPA to form the *G3Card* chip, which was fabricated by TSMC in October 2002. After a fault, which required an on-chip modification, was fixed by using a Focused Ion Beam (FIB) in Israel, the *G3Card* test chip provided the ability to mount functional tests as well as security analyses. Figure 8.1 shows a plot of the G3Card test chip.

In order to test the chip and evaluate the security assumptions which guided the design of this self-timed processor, several essential testing components were implemented on the *G3card* chip.

First, for the purposes of assessing the security improvement against power analysis attacks from dual-rail encoding, in addition to designing a dual-rail encoded SPA core (at the bottom left on the chip plot), a single-rail SPA core was also implemented on the test

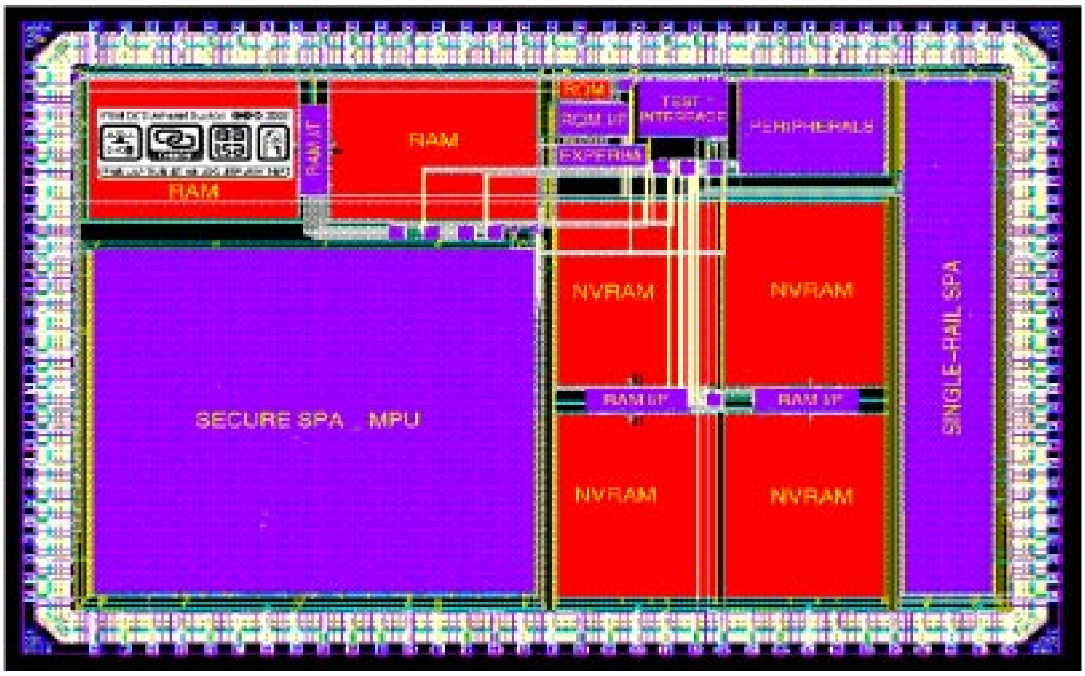


Figure 8.1 *The G3Card* test chip

chip (on the right). The single-rail SPA core was synthesised from the same Balsa source, allowing the most direct comparison possible of the two implementation styles.

Second, to investigate the theoretical understanding that conventional dual-rail storage would leak information about Hamming weight distances because the storage state changes only when a different data value is presented, new dual-rail latches were designed, described in the section “Security in data storage” on page 110, for data storage and state-holding registers. The configurable latches can be switched between conventional and secure mode on a chip-wide basis. This latch design allows an investigation into the impact of secure storage on the resistance to differential power analysis attacks.

In addition, the test chip contains a synchronization coprocessor that provides an external signal, triggered by software. Such a signal can be used as an alternative synchronization signal to a global clock signal. Of course, the provision of the external synchronization signal would not normally be available to attackers.

This chapter will firstly review the *G3Card* chip security investigation environment, and the test PCB. Then, individual instructions, discussed in the previous chapter, will be

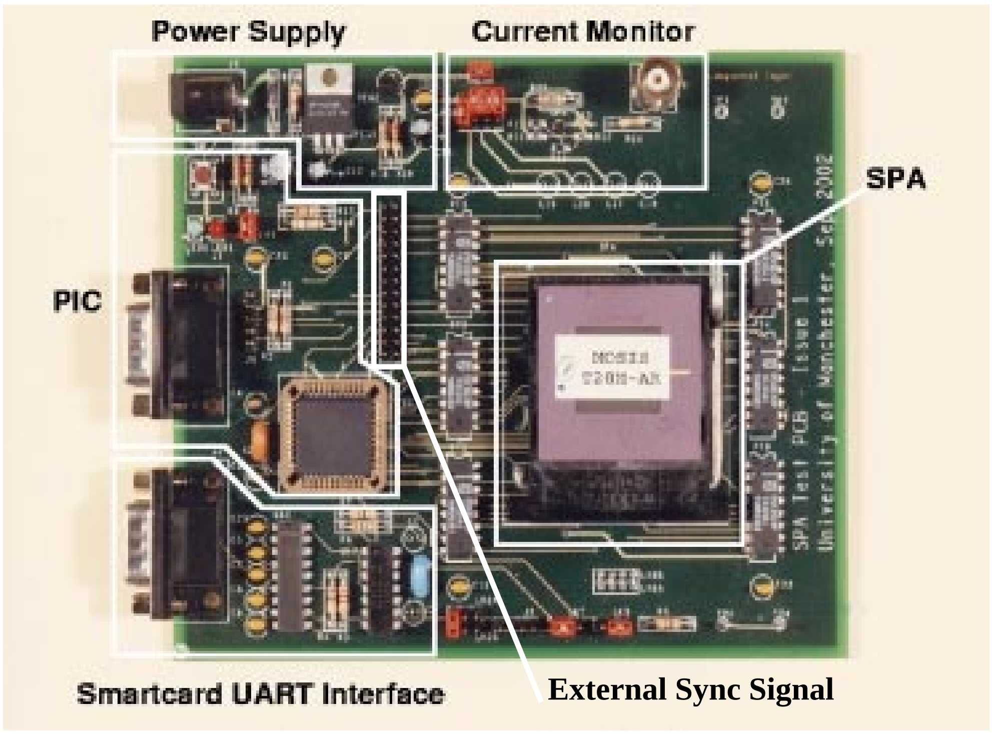


Figure 8.2 The PCB hosting the *G3Card* test chip

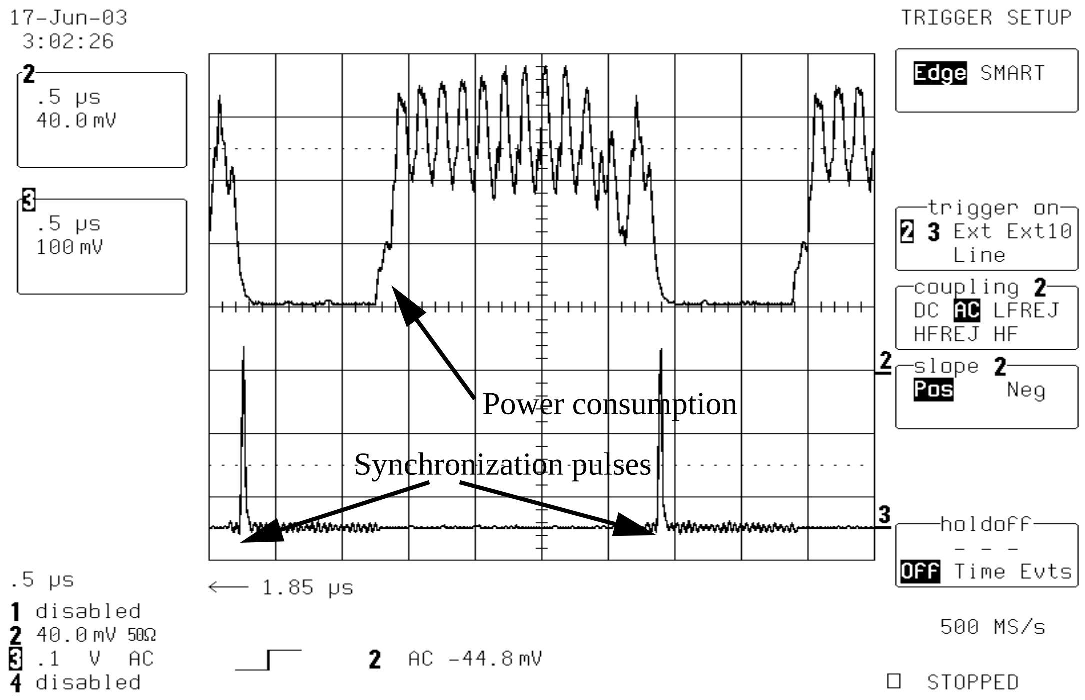


Figure 8.3 *G3Card* power consumption trace on an oscilloscope

investigated on the test chip and the results compared to the layout simulation analysis. Finally, an analysis of the DES algorithm is described at the end of the chapter.

8.2 The investigation environment

In order to undertake functional tests and the security investigation, a PCB [77] was designed to host the *G3Card* test chip, as shown in Figure 8.2. The power supply unit provides four separate power supplies for different sections of the chip. The area marked SPA is the *G3Card* test chip, and there are also two communication ports: one serial port in the PIC area and another serial port in the UART area. Both ports connect the test chip to a workstation. The Current Monitor area is where the external measurement devices attach to measure the power supply current.

8.2.1 Measuring power supply current

To measure the current going to the SPA, a simple scheme was implemented on the PCB using a 10 Ohm resistor in series with the power supply, a high-bandwidth (~300MHz) op-amp and a BNC connector. Jumpers are used to switch the current being measured on different parts of the chip, such as the dual-rail SPA or the single-rail SPA. The 10 Ohm resistor senses the current, which is then amplified by the op-amp, a MAX4221 [78]. The sensed current is captured by an external measuring device attached to the BNC connector which links to the output of the op-amp. In this experiment, a LeCroy 9344CM oscilloscope was used to measure the current. The oscilloscope can sample signals at frequencies of up to 500MHz with 8-bit resolution.

Figure 8.3 shows a screen dump of a power waveform measurement on the oscilloscope. On the screen there are two traces: one is the SPA power consumption trace (the upper trace) and the other the synchronization signal generated by the synchronization co-processor as an artificial timing reference signal. The waveform information is digitized and stored, and then transferred to a workstation for analysis using Matlab.

8.3 ARM instruction power analysis

The power analysis of the chip was undertaken on the same instructions that were investigated at the simulation level. The test code used to generate power waveforms for analysis and evaluation is as follows:

```
loop0   LDR    r0, [r8]        ; Make trigger pulse
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        XOR    r4, r2,    r1   ; Instruction under test
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        B      loop1           ; jump to loop1

loop1   LDR    r0, [r8]        ; Make trigger pulse
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        XOR    r4, r3,    r1   ; Instruction under test
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        MOV    r0, r0          ; nop
        B      loop0           ; jump back to loop0
```

The code is used to determine the power consumption of the XOR instruction processing different data. For instance, when r1 is 0xffffffff, r2 is 0xffffffff and r3 is 0x000000000, the first XOR instruction (line 6) would result in 0x00000000 whilst the second one (line 18) would yield 0xffffffff. Thus, by measuring the power consumption of these two XOR instructions, the experiment aims to examine whether there are differences in power consumption when the XOR instruction is processing data with different Hamming weights. Note that in order to eliminate the interference from instructions before and after the instruction being examined, several no-op instructions were inserted before and after the test instruction. Thus power for one instruction could be affected by one or more previous instructions and could also affect following instructions.

The two branch instructions (line 11 and 23) cause the two loops to be executed alternately. In addition, at the beginning of each loop (line 1 and 13), an extra load instruction, LDR, was inserted to generate the synchronization signal. Note that the single-rail SPA processor could not generate such a synchronization signal due to the design of the *G3Card* chip. The test code was run on both the single-rail SPA and the secure dual-rail SPA separately to explore the security improvement of the dual-rail encoded implementation. The same experiments were repeated with the configurable latches set as conventional (insecure) ones to evaluate the impact of different latch implementations.

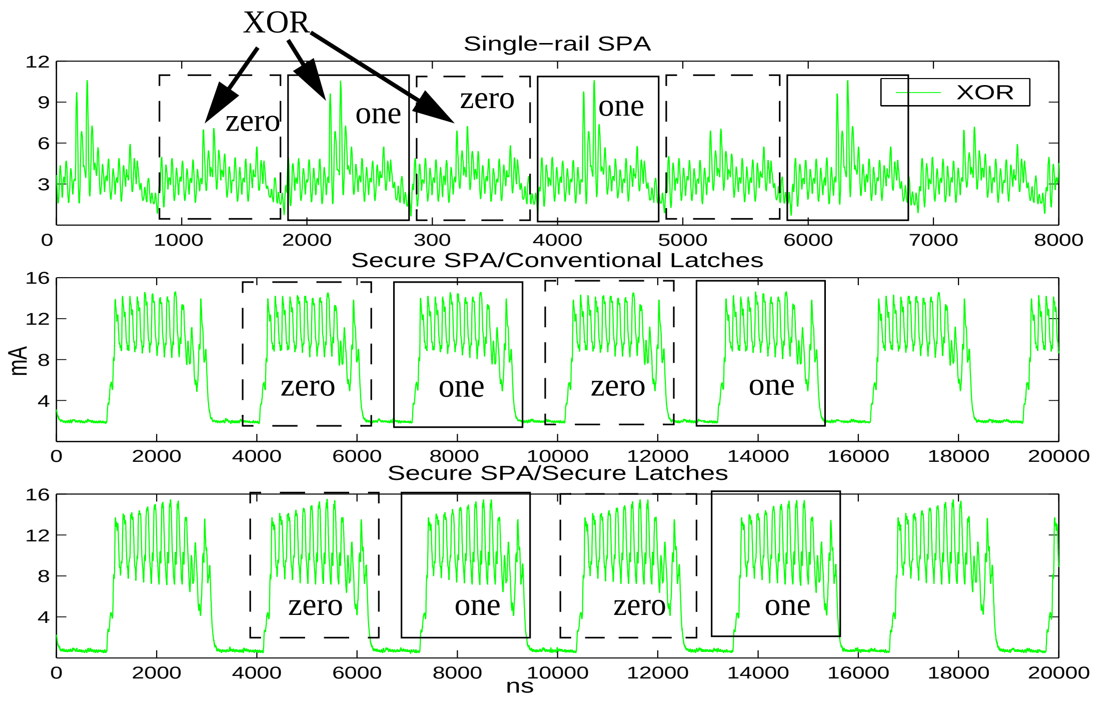


Figure 8.4 Power consumption of the XOR instruction on *G3Card* chip

Figure 8.4 shows the power consumption extracted from SPA when the test code was executed. The three traces show the power consumption processing the code on the single-rail SPA, the secure dual-rail SPA with conventional latches and the secure dual-rail SPA with secure latches respectively. The power trace can be broken down into two groups, loop0 marked with a dashed square and loop1 marked with a solid square corresponding to the two loops in the test code. The fifth power consumption peak in the middle of each square is where the XOR instruction is being executed. From the figure, the difference on the top trace can easily be distinguished (as highlighted). The loop0 group consumes much less power than the loop1 group does. It is, however, hard to see a difference on the bottom two traces by such observation.

This simple observation, therefore, suggests that the single-rail SPA does not really provide much security as the difference in power consumption is clearly visible, comparable to that of clocked circuits. Secondly, it indicates that the secure SPA balances the difference in power consumption and therefore improves the resistance to power analysis attacks. In addition, it also shows that partitioning the dual-rail secure SPA power waveform is much easier than partitioning the single-rail waveform because the synchronization coprocessor takes a long time to communicate with the CHAIN network [79], causing SPA to reduce its power to a low level for a long period as a large part of the circuitry is idle.

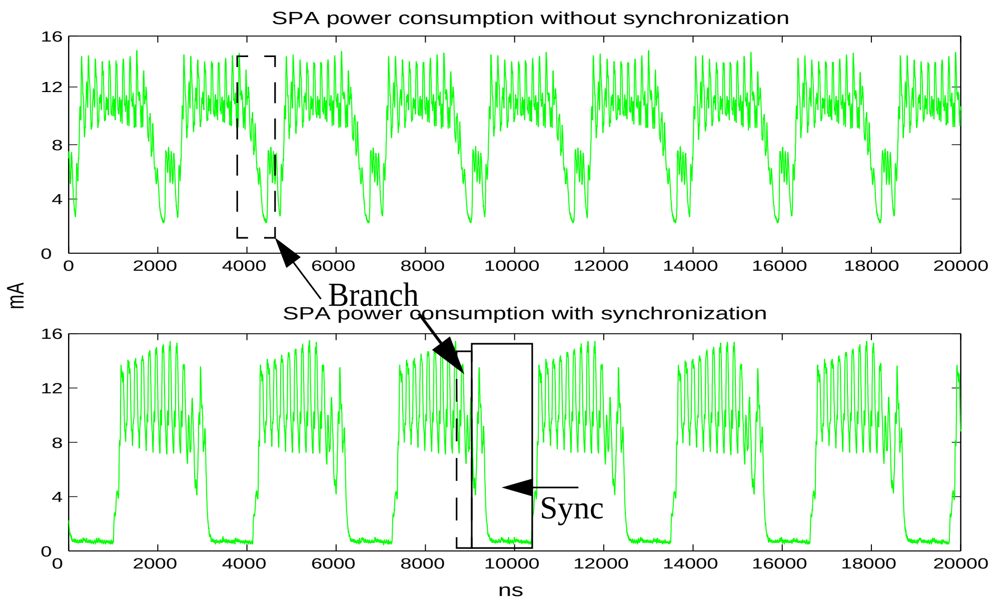


Figure 8.5 Power consumption of SPA with and without the synchronization signal

Figure 8.5 shows a comparison of power waveforms with and without the synchronization signal extracted from secure SPA chip measurement. Obviously, the waveform with the synchronization signal can easily be partitioned because the power is low and stable for a long period of time. However, partitioning the waveform without the synchronization is still possible. In fact, the branch instruction executed alternately between the two loops helps to create a guideline for partitioning the waveform. Hence, the branch instruction can provide similar assistance to the synchronization signal, and it may be used as a timing reference if the synchronization signal is not available.

8.3.1 Differential power analysis of SPA instructions

From simple observation, it is clear that the single-rail SPA does not really provide any security improvement and its power consumption characteristic is comparable to that of clocked circuits. The observation also indicates that the dual-rail secure SPA reduces the power signature significantly. However, to investigate how much the secure SPA improves security and what impact the secure latches offer, it is necessary to perform differential power analysis on the chip power waveforms.

The analysis of the SPA chip measurements follows the same procedure as for simulation. ARM assembly code was run for several hundreds of cycles. However, there are some difficulties in the chip-level power analysis which did not occur in the simulation level.

- In the simulation, an extra signal, execCtrl, which indicates the start of execution of each instruction, was very helpful in the power waveform partitioning. This signal is not available at the chip level. The absence of this signal means that extra effort is required to partition the waveform.

- Secondly, because of the low sampling frequency of 500MHz, many details of the power consumption, which appear in the simulation (where the time resolution was set at 10 ps), do not appear on the chip measurement. Because of this, some useful power information may be lost (which is good for security but difficult for analysis!).

Figure 8.6 shows the typical characteristic waveform for the SPA executing the XOR test code. Looking first at the dual-rail SPA with conventional latches, the loop0 trace (green/ light) shows the power consumption when the XOR results in 0x00000000, whilst the loop1 trace (red/dark) illustrates the power consumption for the XOR yielding 0xffffffff. From the figure, it can be seen that the power consumption difference is very small. However, by careful scrutiny, a small variance could be found at the fifth peak (on the upper window, shown circled) where the XOR instruction takes place. It indicates that the power consumed by yielding the result of 0xffffffff is slightly more than that for

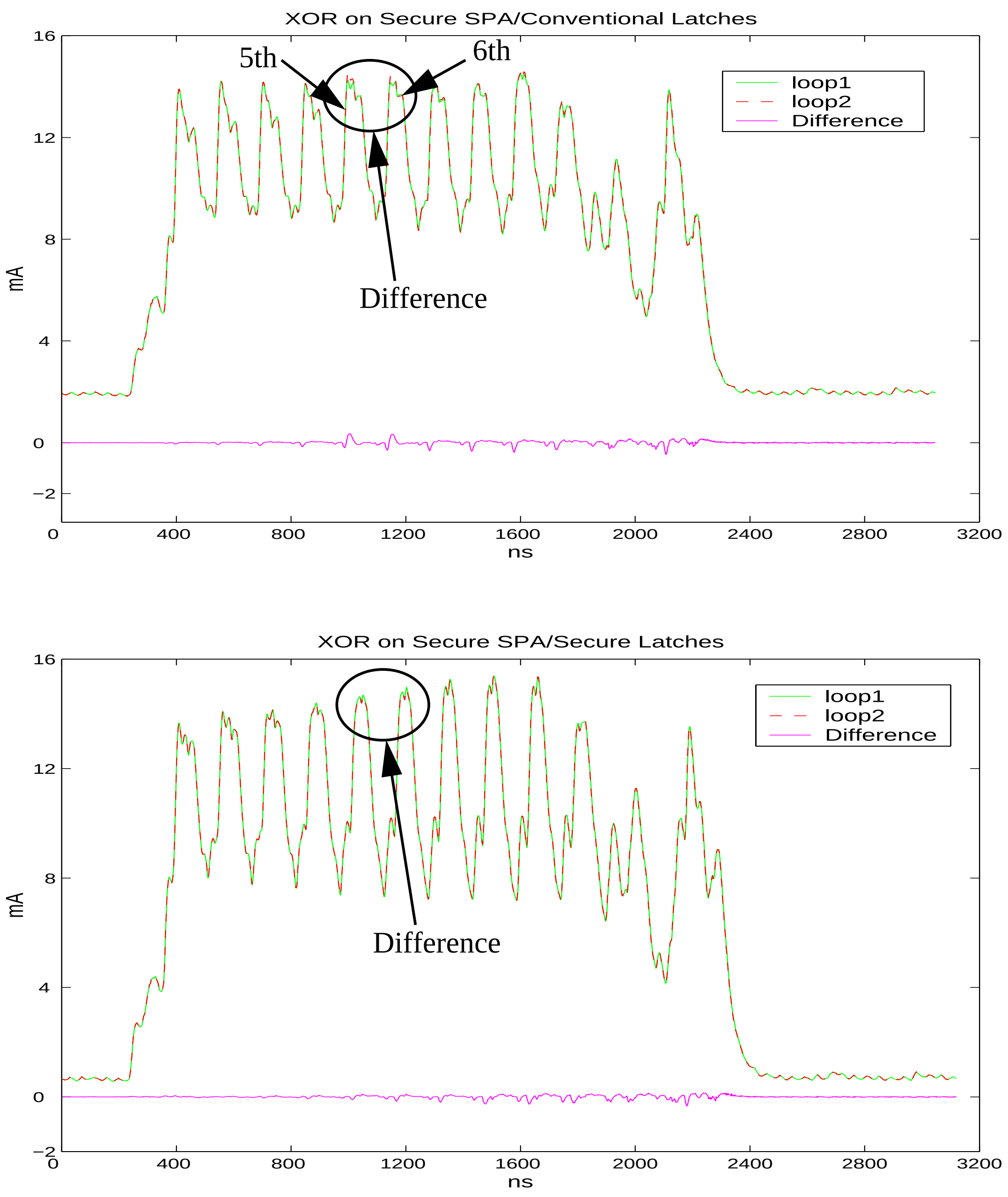


Figure 8.6 DPA on SPA: XOR

0x00000000. In addition, this also affects following instructions and a difference is found at the sixth peak where a no-op instruction was executed and where no difference is normally expected, such as at the fourth peak.

In contrast, with the secure latch this difference on the lower trace is imperceptible. The power consumed processing both data values on the dual-rail SPA with secure latches is almost perfectly matched, confirming the significant reduction in power consumption difference due to the use of secure latches. Figure 8.7 shows a close up view of those differences. It can be seen that the differences on the secure SPA with conventional latches are much larger than with secure latches.

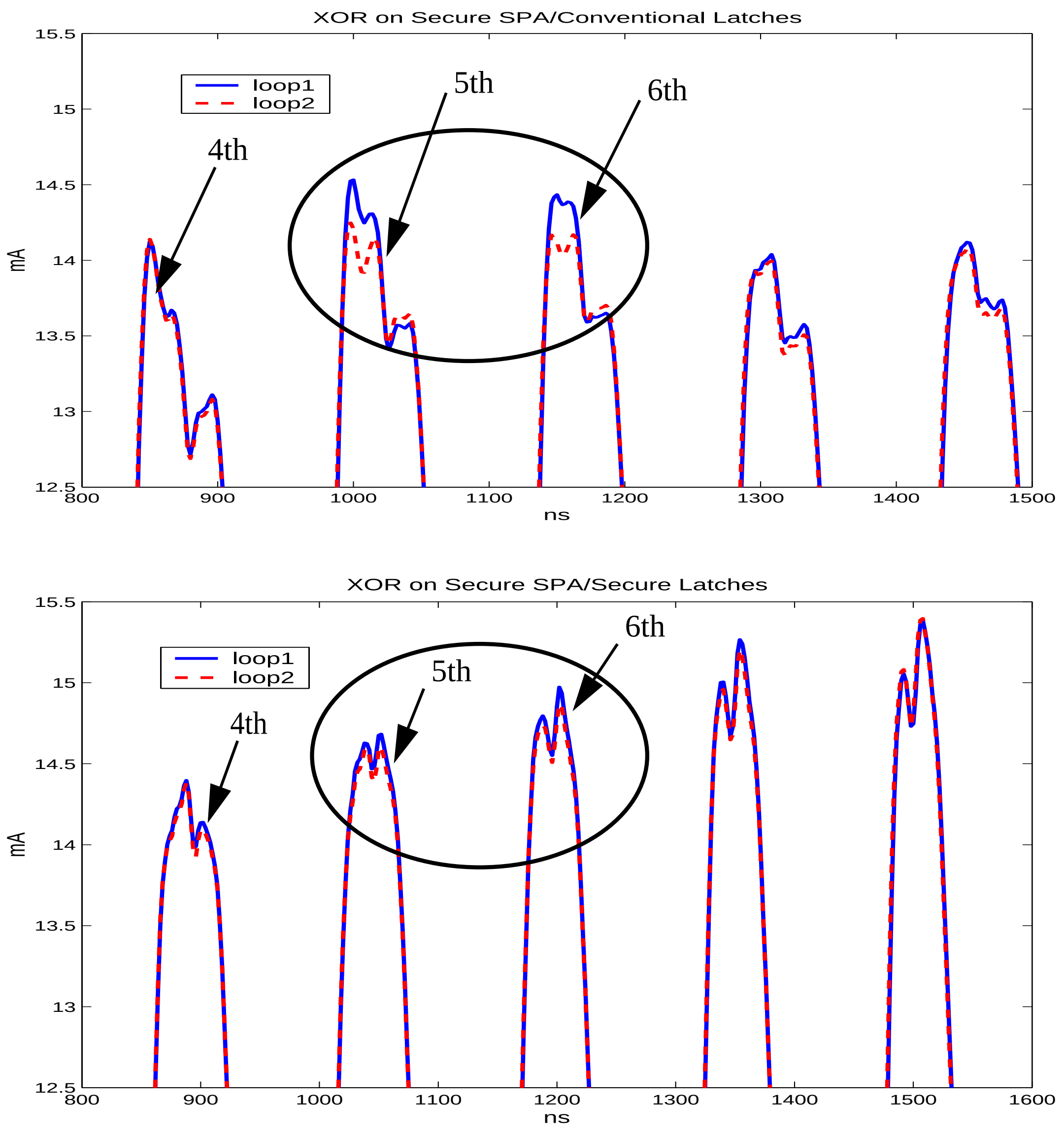


Figure 8.7 DPA on SPA: close up view of XOR

The differential power analysis of the XOR instruction using the asynchronous power analysis methodology showed that 100% correct deduction could be obtained on the single-rail SPA, about 93% correct deduction on the secure SPA with conventional latches, and only about 61% correct deduction on the secure SPA with secure latches. Note that a 50% correct deduction means random guessing, and perfect security. This means that the secure latches make a vital contribution to the security of the *G3Card* chip.

The same analysis was also carried out on the other instructions. Figure 8.8 shows the PCD results indicating the extent of information leakage, derived from the application of the power analysis scheme discussed in Chapter 4 to actual power traces obtained from measurements on the *G3Card* chip. The left most bar of each group represents the single-rail SPA. In every experiment, the single-rail SPA measurements yield a 100% correct deduction, disclosing poor resistance to differential power analysis. The right most bar, representing the secure SPA, shows a much lower PCD in most cases indicating the enhanced security of the secure SPA against DPA. However, the BRANCH instruction still shows a 100% PCD and distance-related SHIFT and ADD instructions show about 80% PCD. This means that these instructions are still vulnerable to differential power analysis attacks.

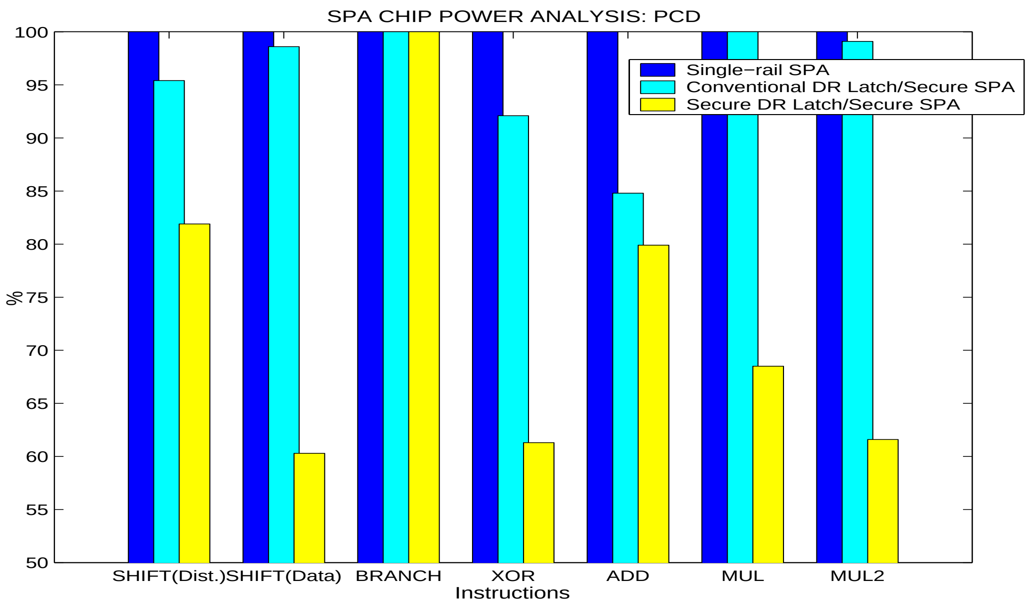


Figure 8.8 SPA chip power analysis: PCD

The middle bars in Figure 8.8 show the PCDs for the secure SPA with conventional dual-rail latches. These results indicate that the secure SPA with the conventional latches improves security only slightly over the single-rail SPA, suggesting that the use of dual-rail encoding and balanced logic without the secure latches is not likely to deliver adequate security. The results clearly show that the secure latches have a large impact on the PCD, indicating that their use results in significantly enhanced security.

8.4 Analysis of the DES algorithm on SPA

The Digital Encryption Standard algorithm, known as DES, has been widely used in many cryptographic applications, both software and hardware, to encrypt and decrypt digital data. Although the AES algorithm [31] will replace DES in the future, DES is still widely used in cryptographic implementations. Meanwhile, many attacks, both invasive and non-invasive, have focused on the DES algorithm. Among them the differential power analysis attack has been very successful and many security devices, such as smartcards, which employ the DES algorithm have been the victims of DPA [8]. This has eroded confidence in the security capabilities of smartcards. Thus, an investigation was undertaken into whether the SPA would defeat DPA when running the DES algorithm.

8.4.1 Running DES on SPA

A DES algorithm, discussed in the section “Data Encryption Standard (DES)” on page 24, was implemented in ARM assembly code, and synchronization code was inserted at the beginning of each of the sixteen rounds to generate a synchronization signal. The code was down-loaded onto the SPA on-chip memory via the UART interface port and executed on the secure SPA with the configurable latches set to secure mode. The power consumption of the SPA was measured and extracted using the digital oscilloscope. The DES algorithm was run several hundred times with random plain-text inputs and a fixed secret key, and power waveforms for each run were recorded.

Figure 8.9 shows the power consumed by running DES on the SPA with a synchronization signal inserted for one full encryption. As illustrated on the figure, the sixteen rounds of the encryption (marked by the dashed rectangle) have been clearly separated by executing the synchronization instruction. The initial permutation (at the left end of the Figure 8.9)

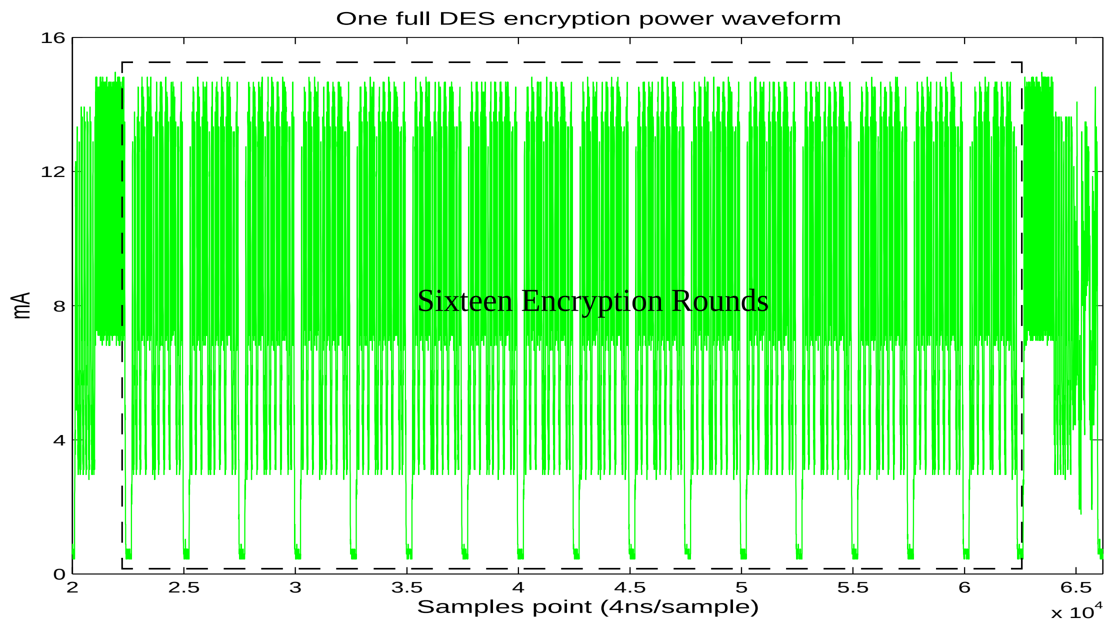


Figure 8.9 Power consumption of SPA: one full DES encryption

and final permutation (at the right end of Figure 8.9) can also be distinguished. In contrast, Figure 8.10 shows the power waveform running DES without a synchronization signal on the SPA for one full encryption. Obvious, it is more difficult to recognize the encryption round on this waveform than it is with the synchronization signal.

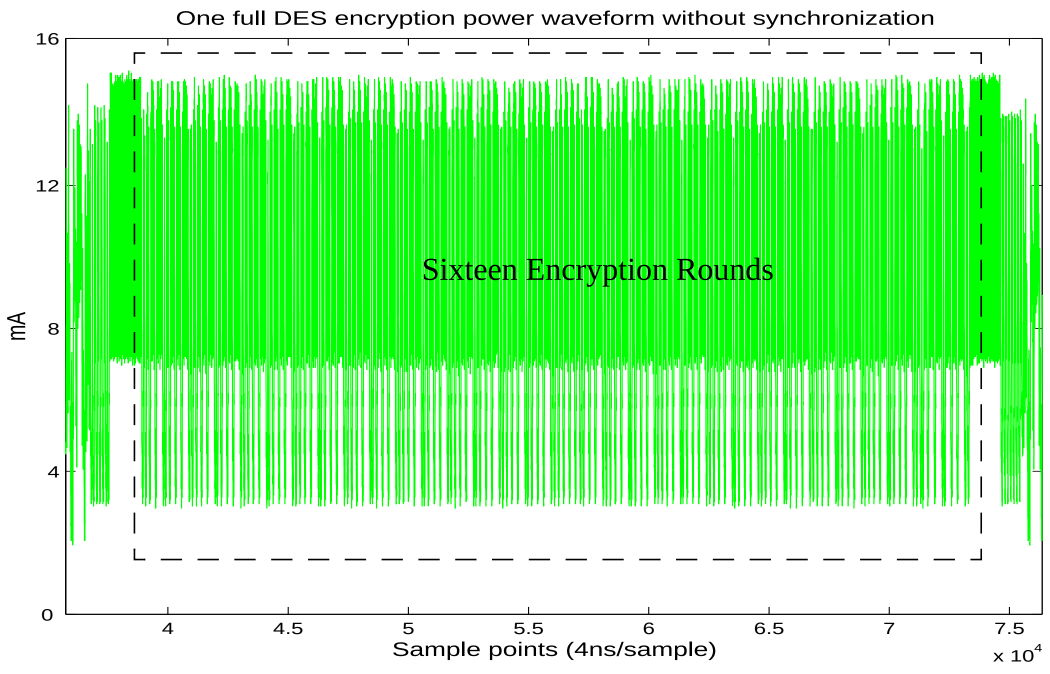


Figure 8.10 Power consumption of SPA without synchronization: one full DES encryption

Figure 8.11 illustrates the last (16th) round of the DES encryption. It shows that there are 40 instructions being executed during the 16th round. It is possible to determine whether the instruction is a data processing instruction, such as ADD and XOR, or a data transfer instruction, such as LDR and STR, by measuring the execution time. The data transfer instructions take much longer to execute than the data processing instructions. Note that this timing information may be used by attackers to determine which kind of instruction is being executed. However, here we only focus on whether there are differences in power consumption when processing different plain-text inputs.

8.4.2 DES waveform partitioning

In attacks on smartcards it is not necessary to analyze all of the executing code. In fact, attackers would always like to focus on as small a range of waveforms as possible, preferably on one clock cycle if they know where the difference happens. The smaller the range of waveforms, the less data that is needed. In this section, the DPA analysis of DES is focused on the 16th round (the last round) of the encryption.

From the description of DPA reviewed in chapter 2, it is known that DPA attacks usually obtain information about the plain-text input, PTI_i, the ciphered-text output, CTO_i and the power waveform, $P_i[j]$, measurement. These waveforms are split into two sets using a

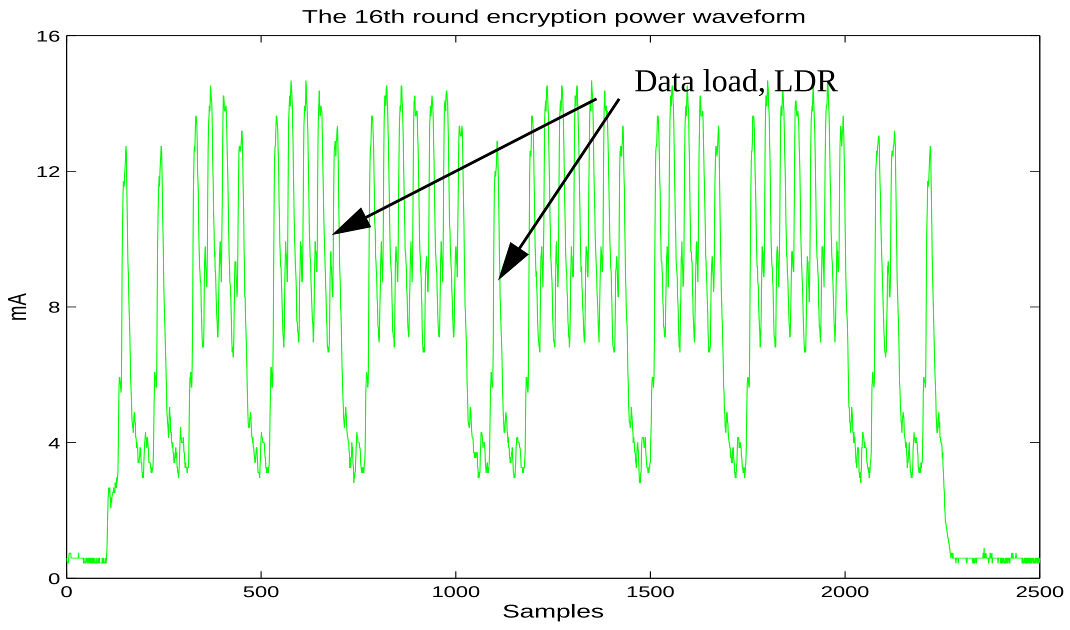


Figure 8.11 Power consumption of SPA: the 16th round of DES encryption

partitioning function, denoted as D. Choosing an appropriate partitioning function will result in a signal bias, which is used to verify guessed portions of the secret key if there is a difference in power consumption when different plain-text inputs are processed. As the SBOX table lookup is usually vulnerable to DPA attacks, one possible partitioning function is based on the SBOX table lookup [[13]], shown as equation (3) on page 34. As discussed in chapter 2, this partitioning function is chosen because at some point during the DES operation, the value of the bit from the function needs to be computed.

Thus, the power waveforms are divided into two groups: group zero, P_0, and group one, P_1 according to equations (4) and (5) on page 35. Then, they are averaged to compute their typical characteristic waveforms. However, as mentioned in previous chapters, averaging self-timed power waveforms is not as simple as clocked power waveforms because of the problems of waveform shift. In this chip level measurement, waveform shift was also one of the difficulties in the analysis, even though the synchronization signal is provided. The provision of the synchronization signal synchronizes the power waveforms and greatly helps partition the waveforms as it provides a visible indication. However, because of the asynchronous behaviour, some partitioned waveforms still have a slight shift problem between synchronization pulses. Thus, refinement of the synchronization was required before averaging those waveforms.

Figure 8.12 illustrates the problem of waveform shift on the SPA chip measurement. The figure shows three waveforms: the reference trace, the shifted trace and the difference trace. Because of the shift problem, a significant difference was present. However, the difference was largely caused by noise from the waveform shift. Furthermore, the degree of shift varies in the time domain. For instance, in this case, the shift lag at around 1000 ns is less than that at around 5000 ns. Thus, a piece-wise shift correction process is needed to realign the waveforms. The lower window of the figure shows the shift corrected waveforms. The difference between the reference trace and corrected waveform is now much smaller than the original one, confirming that the waveform shift injects a large amount of noise into the difference.

However, it is worthy of mention that it is arguable that the shift problems may be vulnerable to timing analysis attack as this waveform shift may be data dependent. This

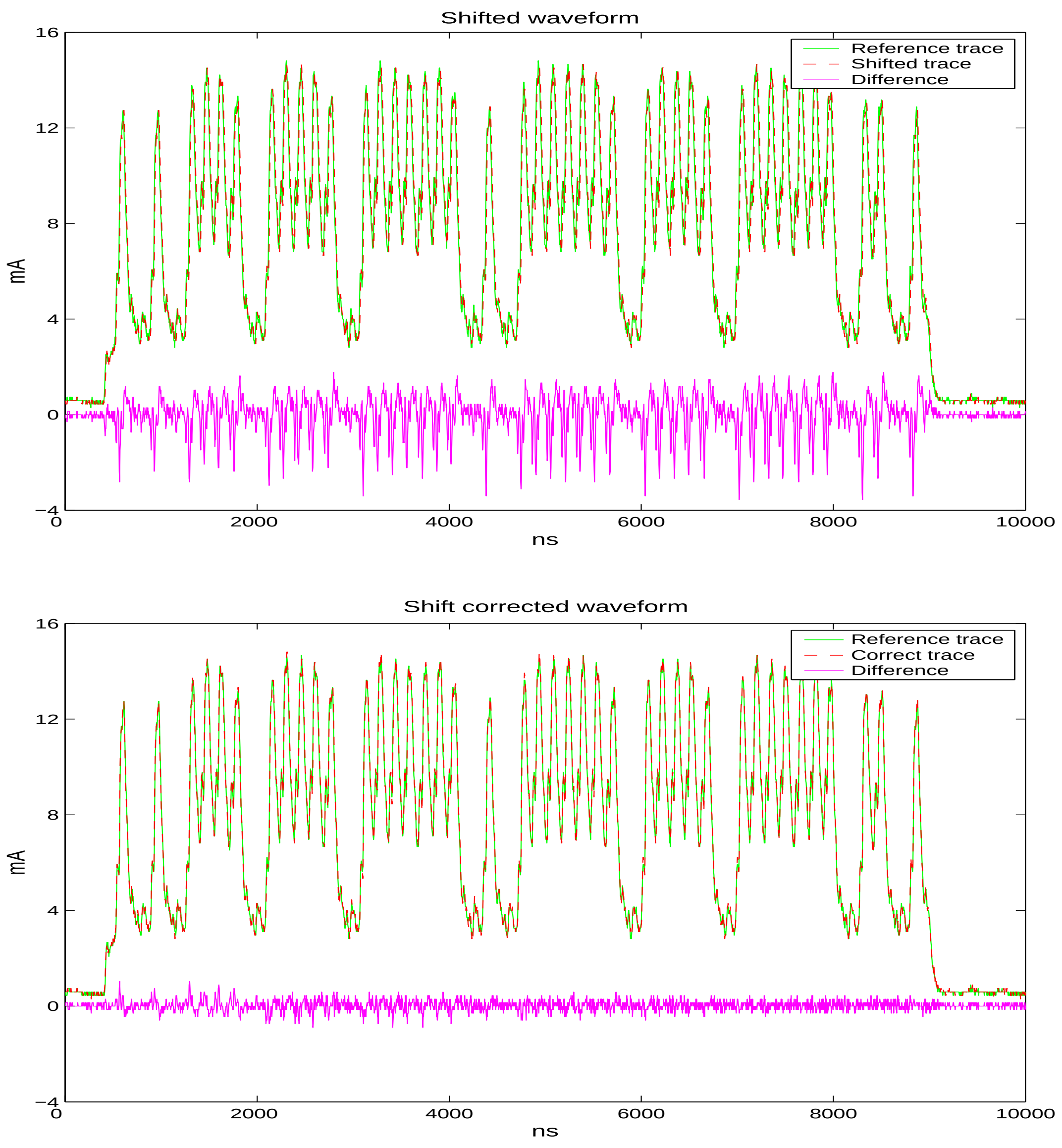


Figure 8.12 Waveform shift problem and correction

could be the basis for further research work. However, in this chapter, the waveform shift is regarded as a noise on power waveform and it creates difficulties for the analysis.

8.4.3 DES analysis results

When the waveforms are corrected, each of the two groups of waveforms is then averaged separately to obtain their typical characteristic waveforms, denoted by $TCW_0(j)$ and $TCW_1(j)$ respectively and the difference waveform $TDW(j)$ between these two characteristic waveforms can also be computed by subtracting one from the other.

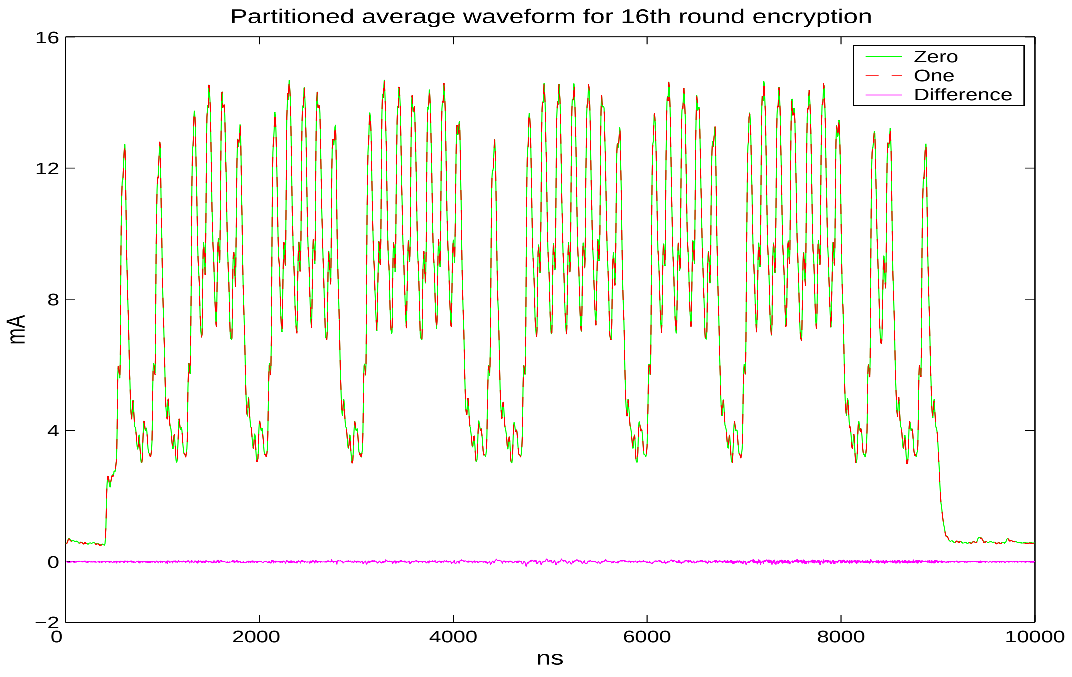


Figure 8.13 Partitioned average power waveform for the 16th round encryption

If the SPA leaks information through power consumption, then whenever this partitioning bit is manipulated, the result will display a small difference in power consumption waveform, denoted as *e*, in the power consumption corresponding to the value of this bit, ‘one’ or ‘zero’. At any other time, it will theoretically be equal to zero. Note that, in reality, it may not be zero because of noise.

For SPA to be secure against DPA, the difference trace, *TDW(j)*, for the correct key guess should be similar to that for all other guesses, as there is no difference in power consumption regardless of the value of the partitioning bit. Thus, the DPA attacker who attempts to exploit the subkey by measuring bias on the power waveform would not be able to deduce the key. Consequently the DES implementation would be safe to run on the SPA processor.

Figure 8.13 shows the two typical characteristic power waveforms for the 16th round of encryption in DES. The ‘zero’ trace (green/bright) is the typical characteristic waveform from averaging waveforms in group zero, P_0, whose partitioning function D(.,.,.,) is equal to ‘zero’, while the ‘one’ trace (red/dark) comes from averaging waveforms in group one, P_1. From the figure, it is extremely difficult to find any difference between them. Indeed, both traces are almost entirely overlapped. The difference trace at the bottom is very close to zero. A signal-to-noise ratio (SNR) is performed to indicate how much signal can be extracted from the power waveforms. The SNRs for both cases are about -60dB which means that the signal is about 1000 times less than the noise. This result indicates that processing the DES algorithm on the secure SPA is data independent and secure from DPA attacks.

Finally, the power analysis method described in Chapter 4 was used to obtain the percentage of correct deduction (PCD) and signal-to-noise ratio (SNR) for the DES power analysis. Two PCDs are calculated: one for waveforms where the key bit is correctly guessed and the other for waveforms where the key bit is incorrectly guessed. It is expected that the former case would have a much higher PCD than the latter case because, in the former case, power waveforms are well separated accordingly to the partitioning function where the key is correctly guessed, while in the latter case the power waveforms are randomly separated because the key bit is incorrectly guessed which leads to the result of the partitioning function being as random as the random plain-text input. The resulting PCDs are that 53.6% correct deduction is obtained for the former case and 52.9% for the latter case, which indicates guessing correct or incorrect key will almost result in the same. In other words, by analyzing the difference in power consumption can hardly reveal whether a key is guessed correctly. Besides, both cases are near to random deduction (50%) as the waveforms are split into two groups for both cases. The result indicates that even though the correct key bit is guessed, the partitioned waveforms are not correlated to the data by which they were partitioned. Therefore, using the power waveforms, an attacker is unlikely to be able to deduce a key bit correctly.

This is a very encouraging result, confirming that the secure SPA with secure storage significantly improves security against differential power analysis and the DES encryption should be protected from DPA when it is implemented and run on SPA.

8.5 Summary of chapter

This chapter investigated the security of the SPA test chip against differential power analysis attacks. The investigation looked into several individual instructions and also examined the DES algorithm. The investigation on the real silicon chip drew similar conclusions to the simulation analysis. The single-rail SPA is vulnerable to differential power analysis, and comparable to clocked circuits. The secure dual-rail SPA improves the resistance to differential power analysis attacks significantly. The use of secure latches has an important impact on enhancing security.

While the provision of the synchronization signal provides an artificial timing reference and provides a great help in waveform partitioning, the nature of the self-timed behaviour still makes the asynchronous power analysis much more difficult than that for synchronous circuits.

Chapter 9: Conclusions

Confidence in the security of smartcards has been weakened since the invention of the differential power analysis attack. Self-timed circuit design technologies have been suggested as a novel solution for implementing secure cryptosystems in order to improve resistance to the differential power analysis attack and potentially other non-invasive attacks. The work described in this thesis identifies difficulties for performing differential power analysis on asynchronous power consumption waveforms, presents a suitable methodology for asynchronous differential power analysis, and confirms the security enhancement provided by the asynchronous circuit design technologies.

9.1 Difficulties and methodology

Because of the lack of a globally distributed clock signal, the events on asynchronous circuits are controlled by local handshake signals. Therefore each sub-circuit regulated by its local handshake signals is independent from other sub-circuits, resulting in the power consumption characteristics being different from synchronous circuits and producing difficulties for power analysis:

- The lack of a clock as a timing reference
- Non-periodicity of the power waveform
- Power waveform shift
- Fade-out of power consumption peaks due to the lack of a timing information

Due to such difficulties, a new differential power analysis methodology was developed, adjusting and extending the conventional techniques for synchronous power analysis, was developed. The methodology includes issues that were not of concern to synchronous power analysis, as follows:

- The methodology involves using artificial timing information, which is not usually available to attackers, to partition asynchronous power waveforms. The provision of this artificial timing reference is very convenient for the investigation of asynchronous power analysis. Without such a timing reference, the analyses would be much more difficult.

- The methodology also includes a method to realign shifted waveforms and make them periodic.

- Because, in asynchronous circuits, power consumption is spread over a period of time and obvious power consumption peaks can be smoothed and hidden, the methodology employs a correlation analysis technique rather than simple averaging to deduce the best correlation between the data and the power information.

9.2 Improved security

Two experimental examples were taken in this work as a basis upon which to evaluate the security improvement offered by asynchronous design techniques: differential power analysis of LFSRs, and a security investigation into the SPA processor. The evaluation first looked at six simple LFSRs to understand the power consumption characteristics of asynchronous circuits and the difference between asynchronous and synchronous power consumption. The evaluation then moved to investigate the large and complex SPA processor at both simulation level and chip measurement level. Suggestions drawn from the investigations and evaluations in this work provide a basic guideline for using self-timed logic in security applications in the future. They are summarized below:

- The work shows how to perform differential power analysis attacks on asynchronous circuits where a global clock signal is absent.

- The work reveals that single-rail asynchronous circuits do not provide adequate security and are still vulnerable to both power and timing analysis attacks.

- The work confirms that dual-rail is the key technology against differential power analysis attacks. Dual-rail code can be beneficial not only to asynchronous designs but also to synchronous designs.

- The work proves that asynchronous design can increase dual-rail security against differential power analysis by making the analysis more difficult because of the lack of a timing reference.

- The work especially demonstrates that security in storage has a significant impact on the overall security. Without a secure storage strategy, an asynchronous circuit enhanced by dual-rail encoding and balanced logic is still not likely to deliver adequate security.

- The work indicates that inserting extra instructions, such as conditional branches which take a long time to execute, may create an artificial timing reference in asynchronous circuits.

9.2.1 Suggestions for improving the security of SPA

The investigation into the security of the prototype SPA processor showed that several instructions were clearly at risk of leaking information about their operation. Those instructions include conditional branches and distance-related shift operations. Therefore, this should be taken into account when using those instructions and in future improvements to the design of the SPA processor.

For software engineers who want to implement an encryption algorithm running on SPA, the obvious suggestion is to avoid using those instructions when computing with secret information. For instance, secure key information should not be used as the decision-making condition in conditional branch instructions.

For hardware engineers who want to design further versions of the SPA processor, the conditional branch on SPA should be designed such that no matter what the condition, the branch is always taken. For instance, when the conditional branch is false the branch is still taken and performs a dummy task that is equivalent to the task performed when the condition is true.

9.3 Further work

The goal of the work presented in this thesis was to evaluate the potential of asynchronous design for use in security applications. The research provides a basic evaluation and includes suggestions for designing security applications using asynchronous technologies in the future. However, there are several ways to extend the work.

Improving the methodology

The partitioning function in the asynchronous power analysis methodology discussed in Chapter 4 employs an artificial timing reference on which power waveforms were partitioned. In the layout-level simulations of SPA, a signal was generated to indicate the commencement of execution of an instruction, and a synchronization signal was also generated during the SPA chip measurements. Those artificial timing reference are a great help for waveform realignment and partitioning. However, in a real attack environment, such artificial timing information is not usually available to attackers.

From the evaluation experiments discussed in this thesis, instructions, such as conditional branch instructions which the processor normally takes extra time to execute, may be used as a synchronization signal. Therefore, those instructions may be used as alternative timing references for power analysis. Further investigations into SPA could explore this possibility.

Further investigation of SPA

The investigation of SPA discussed in this work only looked at certain instructions which were mostly ALU instructions and conditional branches. However, although those instructions provided representative and reliable results and conclusions, the investigation can be expanded to cover other instructions, such as data load and store instructions.

The investigation examined the security of running the DES algorithm on SPA, with the aim of giving a reasonable judgement on how the SPA would protect against power analysis attacks when running a real encryption algorithm. However, analyzing the security of other encryption algorithms, such as AES, running on SPA may provide more useful and solid results. However, another research carried out in Manchester shows a secure implementation of the AES algorithm using asynchronous technologies and dual-rail encoding [58].

Timing analysis

The variability of timing of self-timed circuits is a weakness that could be exploited by alternative attack techniques such as timing analysis. In chapter 7, timing analysis was undertaken based on the SPA layout simulation. The success of this analysis benefited from the provision of the *execCtrl* signal, the synchronization signal indicating the start of execution of each instruction, as the timing information from this signal enabled the analysis.

However, on the test chip this signal was not available. Although another synchronization signal (more precisely a pulse) was successfully generated by communicating to a co-processor, the timing information of this signal was not very accurate. One reason was the low sampling frequency (4 ns per sample) compared to the small timing variation (about 1ns to 5ns for the single-rail SPA and less than 1ns for the secure SPA). Thus, timing analysis was not performed on the SPA chip-level measurement in this work. Further work is encouraged to look into alternative ways, both hardware and software, for obtaining timing information.

Other ways of attacking SPA

Any encryption chip is in principle vulnerable. In this thesis, attention has been given mostly to differential power analysis attacks and to some extent to timing analysis. However, further investigation may be carried out on other ways of attacking SPA to scrutinize whether SPA leaks information through other alternative side-channels, for instance, electromagnetic emissions. It could also be worth exploring other possible measuring instruments that can detect information leakage in other side-channels. For example, the basic security idea behind dual-rail encoding is the balanced power consumption using paired wires. However, if a measuring instrument that can observe signal events on one single wire can be applied to SPA, then the security advantage of dual-rail encoding may be eliminated.

Co-analysis with third parties

Security is an issue that is difficult to prove. A cryptographic device that is proved to be secure against one attack may be still vulnerable to other attacks and sometimes even different people or groups may deduce different results. Therefore, third party evaluation would provide a valuable independent reference point.

9.4 The future

The SPA processor was designed for the *G3Card* project aiming to explore the suitability of self-timed logic for a highly attack-resistant smartcard processor. In Philips the Smart*MX* family of microcontrollers for smartcard applications, based on the 8051 architecture, was developed using asynchronous technologies and has been used in many commercial secure applications [80]. The improved security features evaluated in this work suggest that there is a potential possibility for using asynchronous logic in security applications such as smartcards in the future.

However, there are also drawbacks for asynchronous design. The high cost of dual-rail encoding (as much as twice the cost of single-rail and as much as twice as slow than single-rail) conflicts with the silicon die area limitation of smartcard processors. The complexity of asynchronous design and the lack of CAD support tools would also be an obstacle to the development of asynchronous design. Nevertheless, with increasing interests in asynchronous design, it is possible to overcome such problems by further research.

9.4.1 Alternative technologies

As shown in this thesis, dual-rail asynchronous circuits improve the resistance to differential power analysis attacks significantly. Dual-rail encoding plays a key role against the attack and asynchronous design provides extra security by making the attack much more difficult. However, the security was achieved at the price of much more cost (in terms of silicon area) and incurred difficulties of asynchronous design. The questions may be asked: if the same effort had been put into improving synchronous versions of the dual-rail circuit, could it have been made as good as the asynchronous version? What if it is designed as a contactless smartcard; does a contactless smartcard has less power signature leakage?

Since dual-rail encoding is also beneficial to synchronous circuits, dual-rail versions of synchronous circuits should be able to reduce the difference in power consumption. This is indeed demonstrated by the secure dual-rail LFSR discussed in Chapter 5 and Chapter 6. However, because it is difficult to balance a physical circuit entirely, for instance, it is difficult to manage the load capacitance and wire length exactly during the circuit layout stage, physical circuits can leak more or less side-channel information. Having asynchronous technology to randomize the leakage and make analysis difficult is very important. Nevertheless, it may be interesting to investigate the power consumption and timing information in a large dual-rail synchronous system.

Reference

[1] Maloney, D.L., “Card Technology in Healthcare”, *Proceedings of CardTech/ SecurTech*, pp. 333-351, 1999

[2] ChineseNews, “Chinese government is planning to issue new IC personal identity card”, URL: http://news.sina.com.cn/c/178972.html, 2001

[3] Wright, P., “*Spycatcher--The Candid Autobiography of a Senior Intelligence Officer*”, William Heinemann, Australia, ISBN 0-85561-098-0, 1987

[4] Anderson, R. and Kuhn, M., “Tamper resistance-a cautionary notice”, *Proceedings of Second USENIX Workshop on Electronic Commerce*, pp. 1-11, California, 1996

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

[6] Kocher, P., “Timing attacks on implementations of Diffie-Hellman, RSA,DSS and other system”, *Proceedings of CRYPTO’96*, pp. 104-113, Santa Barbara, 1996

[7] Kocher, P., Jaffe, J. and Jun, B., “Introduction to Differential Power Analysis and Related Attacks”, http:/www.cryptography.com/dpa/, 1998

[8] Kocher, P., Jaffe, J. and Jun, B., “Differential Power Analysis”, *Proceedings of CRYPTO’99*, pp. 388-397, Santa Barbara, 1999

[9] Gandolfi, K., Mourtel, C. and Olivier, F., “Electromagnetic analysis: concrete results”, *Proceedings of CHES 2001*, LNCS Vol. 2162, pp. 251-261, Springer-Verlag, 2001

[10] Quisquater, J.J. and Samyde, D., “ElectroMagnetic Analysis: (EMA) Measures and Counter-Measures for Smart Cards”, *Proceedings of E-Smart Smartcards Programming and Security*, LNCS Vol. 2140, pp. 200-210, Springer-Verlag, 2001

[11] Weingart, S., “Physical Security Devices for Computer Subsystems: A Survey of Attacks and Defences”, *Proceedings of CHES 2000*, LNCS 1965, pp. 302-317, Worcester, MA, 2000

[12] Weingart, S.H., White, S.R. and et al., “An Evaluation System for the Physical Security of computer Systems”, *Proceedings of Sixth Annual computer Security Applications Conference*, Tucson Az. pp. 232-243, 1990

[13] Messerges, T.S., Dabbish, E.A. and Sloan, R.H., “Investigation of Power Analysis Attacks on Smartcards”, *Proceedings of USENIX Workshop on Smartcard Technology*, pp. 151-162, Chicago, 1999

[14] Moore, S., Anderson, R. and et al., “Improving Smart Card Security using Self-timed Circuits”, *Proceedings of Async2002*, IEEE Computer Society Press, pp. 211-218, Manchester, 2002

[15] Plana., L. A. and et al., “SPA - A Secure Amulet Core for Smartcard Applications”, *Journal of Microprocessors and Microsystems*, 27/9, pp. 431-446, 2003

[16] Yu, Z., Furber, S. B. and Plana, L. A., “An Investigation into the Security of Self-timed Circuits”, *Proceedings of ASYNC2003*, IEEE Computer Society Press, pp. 206-215, Vancouver, 2003,

[17] Yu, Z.C. and Furber, S.B., “Defeating Power Analysis Attacks”, *Proceedings of 9th UK Asynchronous Forum*, Cambridge, 2000.

[18] Yu, Z. and Furber, S. B., “Power Analysis of Linear Feedback Shift Registers” *Proceedings of 10th UK Asynchronous Forum*, Cambridge, 2002

[19] Plana, L.A., Riocreux, P.A. and et al. “SPA - A Synthesizable Amulet Core for Smartcard Applications”, *Proceedings of Async2002*, IEEE Computer Society Press, pp. 201-210, Manchester, 2002

[20] “ISO/IEC 7816-2 Integrated Circuit(s) Cards with Contacts - Part 2: Dimensions and Location of the Contacts”, International Organization for Standardization

[21] "ISO/IEC 7816-3 Integrated Circuit(s) Cards with Contacts - Part 3: Electronic Signals and Transmission Protocols", International Organization for Standardization

[22] Li, Z.J. and Zhou, R.D., "*ULSI Devices, Circuits and Systems*", ISBN 0-0300-7242-1, Science Press, China, 2000

[23] Anderson, R., "Why Cryptosystems Fail", *Communications of ACM, 37(11),* pp. 32-40, 1994

[24] Hess, E., Janssen, N., Meyer, B. and Shutze, T., "Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures: A Survey", *Proceedings of EUROSMART Security Conference*, pp. 55-64, Marseille, June 2000

[25] Ward, R., "Survey of Cryptographic Smart Card Capabilities and Vulnerabilities", Technique Report, George Mason University, URL: http://ece.gmu.edu/courses/ECE636/project/reports/RWard.pdf, 2001

[26] Skorobogatov, S. and Anderson, R., "Optical Fault Induction Attacks", *Proceedings of Workshop on Cryptographic Hardware and Embedded Systems*, pp. 2-12, San Francisco, 2002

[27] Schneier B., "*Applied Cryptography: Protocols, Algorithms, and Source code in C, 2nd Edition*", ISBN 0-4711-1709-9, WILEY, 1995

[28] Anderson, R., "*Security Engineering -- a Guide to Building Dependable Distributed Systems*", ISBN 0-4713-8922-6, WILEY, 2001

[29] Menzes, A., Oorschot P.C. and Vanstone, S.A., "*Handbook of Applied Cryptography*", ISBN 0-8493-8523-7, CRC Press, 1996

[30] National Institute of Standards and Technology, "Data Encryption Standard (FIPS PUB 46-3)", URL: http://scrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf, October 1999

[31] Daemen J. and Rijmen, V., “The Block Cipher Rijndael”, *Smart Card Research and Applications*, LNCS 1820, Springer-Verlag, 2000, pp. 288-296

[32] Anderson, R., Biham, E. and Knudsen, L., “Serpent: A Proposal for the Advanced Encryption Standard”, *First AES Conference*, California, 1998

[33] Burwick, C., Coppersmith, D. and et al., “MARS - a candidate cipher for AES”, *First AES Conference*, California, 1998

[34] Schneier, B., Kelsey, J. and et al., “Twofish: A 128-bit Block Cipher”, *First AES Conference*, California, 1998

[35] Rivest, R.L., Robshaw, M.J.B., Sidney, R. and Yin, Y.L., “The RC6 Block Cipher”, *First AES Conference*, California, 1998

[36] Diffie, W. and Hellman, M.E., “New Directions in Cryptography”, *IEEE transactions on Information Theory*, 22 (1976), pp. 644-654, 1976

[37] Anderson, R. and Kuhn, M., “Low Cost Attacks on Tamper Resistant Devices”, *Proceedings of 5th International Workshop on Security Protocols*, pp. 125-136, Paris, 1997

[38] Biham, E. and Shamir, A., “Differential Fault Analysis of Secret Key Cryptosystem”, *Proceedings of CRYPTO’97*, pp. 513-525, 1997

[39] Bardsley, A., “*Implementing Balsa Handshake Circuits*”, Ph.D thesis, University of Manchester, 2000

[40] Weste, N. and Eshraghian, K., “*Principles of CMOS VLSI Design - A Systems Perspective, 2nd Edition*”, pp. 231-238, ISBN: 0-2015-3376-6, Addison-Wesley Pub, 1994

[41] Furber, S.B., “*ARM System-on-Chip Architecture, 2nd Edition*”, ISBN: 0-201-67519-6, Addison Wesley, 2000

[42] Verhoeff, T. and Peeters, A.M.G., “The Asynchronous Bibliography”, Eindhoven University of Technology, The Netherlands, URL: http://www.win.tue.nl/cs/pa/wsinap/async.html

[43] Sparsø, J. and Furber S.B., “*Principles of Asynchronous Circuits Design - A System Perspective*”, ISBN: 0-7923-7613-7, Kluwer Academic Publishes, 2001

[44] Bainbridge, W.J., Toms, W.B. and et al., “Delay-Insensitive, Point-to-Point Interconnect Using M-of-N Code”, *Proceedings of Async 2003*, IEEE Computer Society Press, pp. 132-140, May 2003

[45] Fant, K.M. and Brandt, S.A., “NULL Convention Logic: a Complete and Consistent Logic for Asynchronous Digital circuit synthesis”, *Proceedings of International conference on Application-Specific Systems, Architecture, and Processor*, pp. 261-273, 1996

[46] Sobelman, G.E. and Fant, K., “CMOS Circuit Design of Threshold Gates with Hysteresis”, *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 61-64, 1998

[47] Woods, J.V., Day, P., Furber, S.B. and et al., “AMULET1: An Asynchronous ARM Microprocessor”, *IEEE Transactions on Computers*, Vol. 46 No. 4, pp. 385-398, April, 1997

[48] Furber, S.B., Garside, J.D. and et al., “AMULET2e: An Asynchronous Embedded controller”, *Proceedings of Async ‘97*, IEEE Computer Society Press, pp. 290-299, April 1997

[49] Garside, J.D., Bainbridge, W.J., and et al., “AMULET3i - an Asynchronous System-on-Chip”, *Proceedings of Async 2000*, IEEE Computer Society Press, pp. 162-175, April 2000

[50] Furber, S. B., “Validating the AMULET Microprocessors”, *The Computer Journal*, vol. 45 No. 1 BCS, pp. 19-26, 2002

[51] Coron, J.S., “Resistance Against Differential Power Analysis for Elliptic Curve Cryptosystems”, *Proceeding of Workshop on CHES 1999*, pp. 292-302, 1999

[52] Daemen, J. and Rijmen, V., “Resistance Against Implementation Attacks: A Comparative Study of the AES Proposals”, *Proceedings of Second AES Candidate Conference,* Rome, March 1999

[53] Chari, S., Jutla, C., Rao, J.R., and Rohatgi, R., “A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards”, *Proceedings of Second AES Candidate Conference*, Rome, March 1999

[54] Messerges, T.S., “P*ower analysis Attacks and Countermeasures for Cryptographic Algorithms*”, Ph.D. Thesis, University of Illinois at Chicago, 2000

[55] The Mathworks Inc., “*The student Edition of Matlab*”, ISBN: 0-13-855974-0, Prentice Hall

[56] LeCroy Corp., “*LeCroy Digital Oscilloscopes Operator’s Manual*”, NewYork, 1996, URL: http://www.lecroy.com

[57] Weisstein, P., “Cross-Correlation”, URL: http://mathworld.wolfram.com/Cross-Correlation.html, Walfram Research, 1999

[58] Yu, A., “*The Security of a VLSI Implementation of The New AES Against Power and Timing Attacks*”, Ph.D. Thesis, University of Manchester, 2003

[59] Davies, E.R., “*Electronics, Noise and Signal Recovery*”, Academic Press, San Diego, ISBN: 0122-06131-4, 1993

[60] “CRACK A5”, URL: http://www.jya.com/crack-a5.htm, 1998

[61] Anderson, R., “A5”, URL: http://www.ecn.org/crypto/etere/ander.htm

[62] Golic, J., Cryptanalysis of Alleged A5 Steam Cipher”, *Proceedings of EUROCRYPT ‘97*, pp. 239-255, Germany, May 1997

[63] Biryukov, A., Shamir A. and Wagner D., “Real Time Cryptanalysis of A5/1 on a PC”, *Proceedings of 7th Fast software Encryption*, LNCS 1978, pp. 1-18, New York, 2000

[64] Anderson R., “On Fibonacci Keystream Generators”, *Proceedings of 2nd Fast Software Encryption*, pp. 346-352, Belgium, 1994

[65] Golomb, S.W., “*Shift Register Sequences*”, ISBN 0-89412-048-4, Aegean Park Press, 1982

[66] Petlin, O.A. and Furber, S.B., “Build-in Self-Testing of Micropipelines”, *Proceedings Async’97,* IEEE Computer Society Press, pp. 22-29, 1997

[67] Wang, S. and Gupta, S.K., “DS-LFSR: A BIST TPG for Low Switching Activity”, *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and System,* Vol. 21 No. 7, pp. 842-851, July 2002

[68] Peterson, W.W. and Weldon, E.J., “*Error-Correcting Codes, Second Edition*”, ISBN: 0-2621-6039-0, M.I.T. Press, 1972

[69] Lin, S. and Costello, D.J., Jr., “*Error Control Coding: Fundamentals and Applications*”, ISBN: 0-1328-3796-x, Prentice Hall, 1983

[70] “Primitive Polynomial”, WolfRam Research, the World of Mathematics, URL: http://mathworld.wolfram.com/PrimitivePolynomial.html

[71] Sloane, N.J.A., “The on-line Encyclopedia of Integer Sequences”, A011260/ M0107, URL: http://www.research.att.com/~njas/sequences/

[72] Chu, T.A., “*Synthesis of Self-Timed VLSI Circuits from Graph-Theoretic Specifications*”, Ph.D. Thesis, MIT, June 1987

[73] Furber, S. and Day, D., “Four-Phase Micropipeline Latch Control Circuits”, *IEEE Transactions on VLSI system*, Vol. 4, No. 2, pp. 247-253, ISSN 1063-8210, 1996

[74] Sparsø, J. and Staunstrup, J., “Delay-insensitive multi-ring structures”, *INTEGRATION, the VLSI Journal*, 15(3), pp. 313-340, 1993

[75] Patterson, D., Hennessy, J. and Indurkhya, N., “*Computer Organization and Design, 2nd Edition*”, Morgan Kaufmann Publishers, ISBN 1-5586-0428-6, 1997

[76] “Preliminary Power Analysis investigation of the AMULET2e asynchronous CPU”, G3card project internal report, NDS Technologies Israel Ltd., 2000

[77] Temple, S., “Support Pages for the G3card SPA Chip”, URL: http:/ www.cs.man.ac.uk/~temples/g3card/

[78] MAX4212 Datasheet, Maxim, URL: http://pdfserv.maxim-ic.com/arpdf/ MAX4212-MAX4220.pdf

[79] Bainbridge, W.J. and Furber, S.B., “CHAIN: A Delay Insensitive CHip Area INterconnect”, *IEEE Micro special Issue on Design and Test of System on Chip*, pp 16-23, September/October 2002

[80] “SmartMX Microcontrollers Family”, Philips Semiconductors, URL: http:// www.semiconductors.philips.com/news/backgrounders/bg0026/